Delivering Labeled Teaching Images over the Web

Harold P. Lehmann MD PhD, Bach Nguyen MEE, Joan Freedman MS
Office of Medical Informatics Education, Johns Hopkins School of Medicine, Baltimore, MD
hlehmann@jhmi.edu

Abstract

The Web provides educators with the best opportunity to date for distributing teaching images across the educational enterprise and within the clinical environment. Experience in the pre-Web era showed that labels and information linked to parts of the image are crucial to student learning. Standard Web technology does not enable the delivery of labeled images. We have developed an environment called OverLayer that succeeds in the authoring and delivering of such images in a variety of formats.

OverLayer has a number of functional specifications, based on the literature and on our experience, among them, the following: Users should be able to find components by name or by image; to receive feedback about their choice to test themselves.. The image should be of arbitrary size; should be re-usable; should be linked to further information; should be stand-alone files. The labels should not obscure the image; should be linked to further information.. Images should be stand-alone files that can be transferred among faculty members.

Implemented in Java, OverLayer (http://omie.med.jhmi.edu/overlayer) has at its heart a set of object classes that have been reused in a number of applets for different teaching purposes and a file format for creating OverLayer images. We have created a 350-image histology library and a 500-image pathology library, and are working on a 400-image GI endoscopy library.

We hope that the OverLayer suite of classes and implementations will help to further the gains made by previous image-based hyperlinked technologies.

Introduction

Since the days of videodiscs [1] and, later, authoring applications like HyperCard, SuperCard, and ToolBook [2], medical educators have valued the ability to provide students with active labeled image sets. These are images where the identifying labels can be hidden or displayed, as the student desires. This ability is crucial for the student to learn to distinguish important features from unimportant features. We shall call the features the instructor wants the student to identify *components*. They might be histological structures, endoscopic appearances, or electrocardiogram features.

The World Wide Web has swept through medical education, but its core specifications do not make it easy to implement the labeling functionality. For instance, image maps (ISMAPs) can be used to click on a pre-defined component of an image as a means of navitation. However, this navigation does not provide basic visual feedback and its management is cumbersome.

In this paper, we shall discuss the features attendant to the ability to provide labeled images, we shall discuss our design for these functions, we shall present our implementation, and we shall discuss the variety of possible implementations in the age of the Web.

Features

Our perspective regarding labeled images is colored by our students' use of computer-based atlases in a number of areas, primarily histology and pathology. For instance, in 1995 we created Histology Imagebase, a Macintosh application [3]. It contained 350 digitized slides from the teaching collection. On each slide, significant structures were outlined. The following features were identified as important to the user:

- to find components by selecting its name
- to find the name by selecting the component
- to choose when to display labels
- to be able to test oneself on the identities of different components
- to view the image from a number of semantic viewpoints (e.g., phenomenological, nosological, functional)

The last specification was a result of a factorial, controlled trial of an earlier implementation, at our institution [4].

These simple desiderata have their own implications. If the user should be able to explore an image, then the labels must be attached to *each* instance of the component on the image, and not just on convenient examples that make sense when an image is displayed on a printed page. Similarly, as much of an image should be labeled, precisely to encourage this sort of data-driven exploration. The first and last functions

1091—8280/98/$5.00

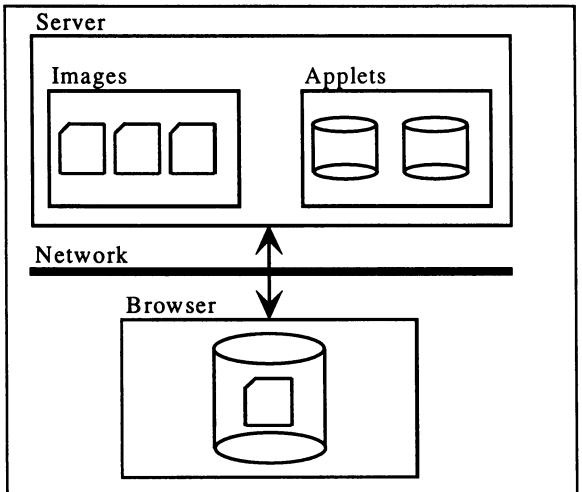

Figure 1. OverLayer system architecture.

are related in that, if a user is to select a component by name, then the *semantic* of naming should be displayed.

In thinking about the use of computer-based images, we developed further specifications. The principles were as follows:

1. Labeled images should be *re-usable* in differently authored Web sites. For example, an image of an infarct of the kidney could be used to teach the identification of necrotic tissue as well as general kidney anatomy. Labeled images might be used in atlases, in virtual slide carousels, in lecture notes, or in quizzes.

2. The labeling should be *independent of the purpose* of the label. This desideratum is a corollary of the preceding specification and argues against the traditional use of arrows or circles, in favor of outlines of the component itself.

3. The label should *not obscure* the image, or, it should *not be part* of the image. This results from the previous specification, and also argues against arrows, on-image labels, and the like.

4. The user should get *feedback* that she has selected a component of the image. For example, the outline might flash or the name of the component highlights.

5. Authors should be able to *link* a labeled area to arbitrary information. For instance, a histology atlas might link a cytological structure to a Web site about mitochondria, while an endoscopy structure might be linked to a virtual-reality (VRML) model of the GI tract, locating the position of the endoscope in the viscus.

6. Users should be able to *test* themselves based on a corpus of labeled images. This feature has been the most popular among students. Further issues arise here. Users should be able to *specify the scope* of the test to include or exclude any material they wish. They should be able to specify whether they want to be asked to *identify* a highlighted component by clicking on its name or to be asked to *locate* a named component by clicking on its image.

7. Images should be *stand-alone files* that can be transferred among faculty members. This specification supports the current way in which teachers assemble most teaching sites: as Web sites comprising files of pages. This specification argues against ISMAP-type solutions, which would entail creating a *folder* of files, because if a single file in the folder were deleted or left behind by mistake, the site would not function properly. On the other hand, a more efficient architecture would be to make each image a record in a database and each page, dynamically generated from that database, but most educators do not have the database capability or the programming abilities to make such sites work.

8. Finally, the *size* of the image is not well specified. On the one hand, it should be as large as possible, since the information is in the picture. On the other hand, it should be as small as possible to ease transfer from server to client. At the least, then, the size should be *flexible* and arbitrary. This specification allows for displaying many different image types, like photomicrographs, pictures of gross specimens, radiographs, endoscopy photographs, and EKG tracings.

Design

While the Web is a client/server environment by definition, it leaves us with degrees of freedom about what functions are performed where. Our solution is to create a proprietary file format for the image, its layers, its labels, and its ancillary information. This file is read into one of a set of Java applets, one applet for each of its intended uses. Fig. 1 shows this system architecture. In the Critique Section we discuss why other solutions were rejected.

File Format

The file must contain a variety of elements: the image, the layers, the text associated with each. The ideal file format for this conglomerate would be GIF 89, as used in animations, because the file could be repurposed as a non-OverLayer file, which would help instructors. However, the GIF89 format is proprietary to CompuServe, and there are no Java classes to

```
[Metadata]
Group=Heart
Filename=101a Myocarditis
Species=Human
...
Description=http://.../101a.html
NumberOfLayers=2
ImageName=htp://.../101amain.jpg
Question=
;
[Layer1]
Filename=Myocytes
Description="These eosinophilic..."
ImageName=http://../Myocytes.jpg
;
[Layer2]
...
```

Figure 2. An example of a configuration file.

deal with it. Thus, we are forced to create a proprietary file format, but based on a standard image format.

Our approach is to append together a number of files into a single Java archive file (jar). The files include the image JPEG file, the layers' black-and-white JPEG files, and the text file containing the rest of the information. The text file is called the *config* file. This file has the format suggested in Fig. 2. The text is parsed by the applets to fill in the Image and Layer objects (see "Implementation," below). Note that descriptions may be strings or URLs. Also, the fields themselves may be domain dependent: A clinical radiology image set would not use the field "Species," and those files would not have that field entry. So, a radiology -specific applet could be created that can read files of a minor modification. The open format of the config file allows for novel features, like multi-dimensional labeling.

Applets

The core applet comprises four classes. The `OverLayerFrame` presents images and layers. The `FileBrowserDialogue` presents the image names, the layer names, and the associated information. The `LayerObject` comprises the layer JPEG image, the associated information, and its position in the "stack" of layers associated with the `Image`, the fourth class.

In the core applet, the `FileBrowserDialogue` instance seen by the user is a separate window from the `OverLayerFrame`. In other applets, they share a space that is in-line to the text. The user can specify the relative widths of the two major interface components, depending on screen and image sizes. A third applet permits viewing of two images simultaneously, for comparison purposes.

Implementation

Fig. 3 shows a screen shot of an OverLayer image (see http://omie.med.jhmi.edu/overlayer), using the core applet. Key points are as follows: The panel on the left—an instance of the `FileBrowserDialogue` class—contains a hierarchical list of the images (if there are more than one), and of labels. Clicking on a label brings up the associated layer in the Image panel. The current default method of highlighting is to flash the target layer in black—to draw the user's attention to the component—and then to present the parts of the image not in the layer as "washed out." This method satisfies the specifications that the labeling does not appear on the image itself nor does it obliterate any of the images. It has been well received by users.

The bottom section of the `FileBrowserDialogue` panel is a truncated Web browser that can display text from the config file or can display a full Web page, as specified by a URL in the config file.

The code for the applets is written in 100% Pure Java, using JDK 1.1, which runs on Netscape Navigator versions 4 and up, and on Microsoft Internet Explorer, versions 4 and up.

Typical html to invoke an OverLayer applet is as follows:

```
1. <APPLET  CODE= OverLayerApplet.class
2. CODEBASE= "/OverLayerApplet"
3. HEIGHT = 413 WIDTH = 633
4. ALT=  "JAVA Applet">
5. <PARAM NAME= "COUNT
6.        VALUE= 2>
7. <PARAM NAME= "NAME0"
8.        VALUE= "http:// .. /101/101a">
9. <PARAM NAME= "NAME1"
10.       VALUE= "http:// .. /101/101b">
11.</APPLET>
```

Lines 1 through 4 invoke the applet. Lines 5 and 6 establish that 2 OverLayer files will populate this instance of the applet. Lines 7 and 8 define the first file: the name is a placeholder, while the URL in Line 8 defines the location of the OverLayer file. Lines 9 and 10 repeat this information for the second file.

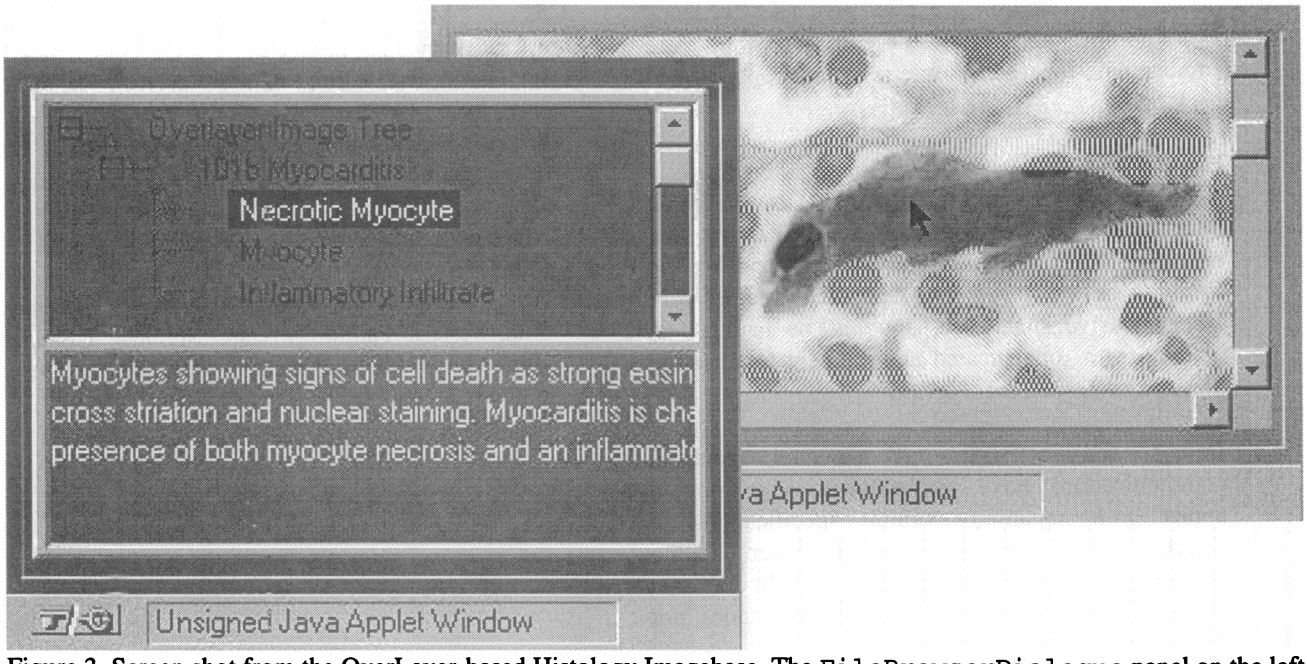

Figure 3. Screen shot from the OverLayer-based Histology Imagebase. The `FileBrowserDialogue` panel on the left can present multiple images within a single OverLayer file (the `FileBrowser`) and can present multiple layers within an image (the `ImageTree`).

We are currently working on three atlases with this environment. The Histology atlas is a port of the Histology Imagebase [3] developed previously for the Macintosh. This is a set of 350 images. We are constructing a Pathology atlas comprising 500 images, and a GI endoscopy atlas of 500 images. The time to create an OverLayer image is about 1 hour: 5 minutes of scanning, 10 minutes of faculty review for choosing what areas to label and to create the ancillary text, 30 minutes of an assistant (medical student, etc.) to label, 5 minutes of faculty review, and 10 minutes to port to OverLayer format.

Performance of the OverLayer environment has been primarily on PCs with 200 MHz Pentium chips. Compared with downloading and viewing of a standard JPEG images, viewing OverLayer files is about 4 times as long. On the Macintosh, performance has been worse. However, downloading a second or third OverLayer file is faster, because only the image needs be transmitted; the code is already resident.

Conclusion

In this paper, we have presented a reasoned approach for dealing with labeled images over the Web. The approach is designed to meet the needs of students and teachers, including flexibility, generalizability, scalability, and future changes in the Web.

The approach is flexible because it uses the Web standard in images—JPEG—so any content can form the basis of an OverLayer image. The system is generalizable because there are no implicit assumptions about the subject domain. The system is scaleable because its heart is the single-file design of the Web. The system should work in future versions of the Web, because it relies on Web standards: Java and JPEG.

The primary problem with our implementation concerns download time. The main cause of this bottleneck is the large size of the files we are currently using: 640 x 480 8-bit images. We expect that, for other domain areas, like GI Endoscopy, with much smaller image sizes, the download time will shrink. Further, as the Web matures, this temporal overhead will be overcome. The other disappointment concerns the Macintosh performance. However, the slow download and run-times there should be obviated by the use of Microsoft Internet Explorer and the new Macintosh Java Virtual Machine.

We are not the only developers to tackle the problem of delivering Web-based labeled images. McEnery and colleagues [5] delivered images with static images as part of a self-teaching module. This approach does not enable to student to explore the image. In the same year, Bradley and colleagues [6] used a database to present images with structures pre-selected. While their technique used a database to present the labeled images, ISMAPs can be used, with static links from the ISMAP to static, labeled images. While

these last two approaches both give user feedback and permit exploring the image, they are cumbersome for teachers and rely on a sophisticated back-end.

Hagler and colleagues [7] have developed a pathology case-teaching environment, using labeled images. Their impelementation uses Javascript that enables students to click on a label and to see feedback on the image. This approach is either very labor intensive, since each page must be hand coded, or requires a sophisticated environment for creating those pages from a meta-script.

Dennis and colleagues (S. Dennis, personal communication) have implemented a java-based applet with apparently similar specifications to OverLayer, providing visual feedback when clicking on the label and has a testing aspect. It does not enable asking where a structure is and does not display ancillary information.

Labeled images will always be a part of education via the Web. The technology is still not straightforward. We are planning on tightening up performance and on creating an editor, enabling easy creation of OverLayer files. Other developers are invited to use the OverLayer format and applets.

Acknowledgments

Thanks to faculty members Renee Dintzis, Lorraine Racusen, Gyongi Nadasdy, and William Ravich for help in specifying OverLayer and in working on content. Thanks to Bonnie Cosner for managing machines and content throughput. This work was funded by NLM grant G08 LM06232-02.

References

1. Stensaas SS. Animating the curriculum: integrating multimedia into teaching. Bulletin of the Medical Library Association 1994;82(2):133-139.
2. Spencer K. HyperCard: Teaching technology for successful learning. J Audiovisual Media Med 1990;13(1).
3. Lehmann HP, Wachter MR. Delivering structured educational images over a network. In: Gardner RM, editor. Nineteenth Annual Symposium on Computer Applications in Medical Care; 1995; New Orleans, LA: American Medical Informatics Association; 1995. p. 989.
4. Lehmann H, Freedman J, Massad J, Dintzis R. A controlled trial on the use of a computer-based histology atlas during a laboratory course. .
5. McEnery K, Roth S, Kelley L. A method for interative medical instruction utilizing the World Wide Web. In: Gardner R, editor. Nineteenth Ann Symp Comp Appl Med; 1995; 1995. p. 502-507.
6. Bradley S, Rosse C, Brinkley J. Web-based access to an online atlas of anatomy: The Digital Anatomist Common Gateway Interface. In: Gardner R, editor. Nineteenth Ann Symp Comp Appl Med Care; 1995; 1995. p. 512-516.
7. Hagler HK, Kumar V, editors. University of Texas Southwestern Pathology Case Studies: Thomson Science; 1997. http://pathcuric1.swmed.edu/PathDemo/MainTofC.htm