

Why Take Notes?
Use the Whiteboard Capture System

Li-wei He
Zhengyou Zhang and Zicheng Liu

September, 2002

Technical Report
MSR-TR-2002-89

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Why Take Notes?

Use the Whiteboard Capture System

Li-wei He, Zicheng Liu, Zhengyou Zhang
Microsoft Research
One Microsoft Way, Redmond, WA, USA
{lhe, zliu, zhang}@microsoft.com

ABSTRACT

This paper describes our recently developed system which captures both whiteboard content and audio signals of a meeting using a digital still camera and a microphone. Our system can be retrofit to any existing whiteboard. It computes the time stamps of pen strokes on the whiteboard by analyzing the sequence of captured snapshots. It also automatically produces a set of key frames representing all the written content on the whiteboard before each erasure. Therefore the whiteboard content serves as a visual index to efficiently browse the audio meeting. It is a complete system which not only captures the whiteboard content, but also helps the users to view and manage the captured meeting content efficiently and securely.

Keywords: Whiteboard capture, meeting archiving, video analysis, audio/video indexing, image classification, corporate knowledge base, digital library.

1 INTRODUCTION

The Whiteboard Capture System is a part of our Distributed Meetings project, the goal of which is to dramatically improve the meeting experience of knowledge workers using ubiquitous computing technologies.

The work presented in this paper focuses on the particular meeting scenarios that use whiteboard heavily such as: brainstorming sessions, lectures, project planning meetings, patent disclosures, etc. In those sessions, a whiteboard is indispensable. It provides a large shared space for the participants to focus their attention and express their ideas spontaneously. It is not only effective but also economical and easy to use -- all you need is a flat board and several dry-ink pens.

While whiteboard sessions are frequent for knowledge workers, they are not perfect. The content on the board is hard to archive or share with others who are not present in the session. People are often busy copying the whiteboard content to their notepads when they should spend time sharing and absorbing ideas. Sometimes they put “Do Not

Erase” sign on the whiteboard and hope to come back and deal with it later. In many cases, they forget and content is left on the board for days. Even if they do come back, they are often confused by the sketchy handwritings, unorthodox acronyms, and wonder how on earth a result is derived.

Our system is an attempt to alleviate meeting participants the mundane tasks of note taking by capturing whiteboard content automatically along with the audio. We use a high-resolution digital still camera to capture one image of the whiteboard about every 5 seconds. We designed an algorithm to distill a small set of key frame images from the captured image sequence. Time stamps of the pen strokes contained in the key frames are also computed. The users can view the key frame images, print them as notes, or cut and paste them into documents. If the users want to find more about the discussion on a particular topic, our browsing software allows them to click some pen stroke associated with that topic and bring up the audio at the moment when the stroke was written.

From our initial deployment, we found our system useful for the meeting participants to review the meeting at a later time, and also for users who did not attend the meeting to understand the gist of the meeting in a fraction of meeting time. We are also pleasantly surprised by some creative ways that participants use to take advantage of the new-found capability in the meeting room.

The paper is organized as follows: Section 2 discusses related works. Section 3 explains the design choices that we made and presents the architecture of our Whiteboard Capture System. Section 4 explains the technical challenges we encountered while building the system. Section 5 reports the details of the deployment process of our system. We discuss the limitations of our system in Section 6 and conclude in Section 7.

Note that the drawings in this paper were done on a whiteboard and were captured by our system.

2 RELATED WORKS

2.1 Whiteboard Capture Devices

Many technologies exist to capture the whiteboard content automatically. One of the earliest, the whiteboard copier, is a special whiteboard with a built-in copier. With a click of a button, the whiteboard content is scanned and printed. Once the whiteboard content is on paper, it can be

photocopied, faxed, put away in the file cabinet, or scanned into digital form.

Recent technologies all attempt to capture the whiteboard into digital form from the start. They generally fall into two categories. The devices in the first category capture images of the whiteboard directly. NTSC-resolution video cameras are often used because of their low cost. Since they usually do not have enough resolution for a typical conference room size whiteboard, several video frames must be stitched together to create a single whiteboard image. The ZombieBoard system [14], deployed internally at Xerox PARC, uses a Pan-Tilt video camera to scan the whiteboard. The Hawkeye system from SmartTech opts for a three-camera array that takes images simultaneously. Another device in this category is the digital still camera. As high resolution digital cameras get cheaper, taking snapshots of the board with a digital camera becomes a popular choice. To clean-up the results, people use software (e.g. Adobe Photoshop or Pixid Whiteboard Photo) to crop the non-whiteboard region and color-balance the images. Our system differentiates from the above systems in that we compute the time stamps of the pen strokes and the key frames by performing analysis on the captured images. The key frames provide the summary notes to the meeting. The time stamps and key frames are effective indices to the recorded audio.

Devices in the second category track the location of the pen at high frequency and infer the content of the whiteboard from the history of the pen coordinates. Mimio by Virtual Ink is one of the best systems in this category. Mimio is an add-on device attached to the side of a conventional whiteboard and uses special cases for dry-ink pens and eraser. The pen emits ultrasonic pulses when pressed against the board. The two receivers at the add-on device use the difference in time-of-arrival of the audio pulses to triangulate the pen coordinates. Since the history of the pen coordinates is captured, the content on the whiteboard at any given moment can be reconstructed later. The user of whiteboard recording can play back the whiteboard like a movie. Because the content is captured in vector form, it can be transmitted and archived with low bandwidth and storage requirement. Electronic whiteboards also use pen tracking technology. They go one step further by making the whiteboard an interactive device. For example, the SMARTBoard from SmartTech is essentially a computer with a giant touch-sensitive monitor. The user writes on the monitor with a special stylus which is tracked by the computer. The computer renders the strokes on the screen wherever the stylus touches the screen -- as if the ink is deposited by the stylus. Because the strokes are computer generated, it can be edited, re-flowed, and animated. The user can also issue gesture commands to the computer and show other computer applications on the same screen. However, electronic whiteboards currently still have limited install base due to their high cost and small sizes (the size of an electronic whiteboard rarely exceeds 6 feet diagonal). Furthermore, those pen-tracking devices have the following

disadvantages: 1) If the system is not on or the user writes without using the special pens, the content cannot be recovered by the device; 2) Many people like to use their fingers to correct small mistakes on the whiteboard in stead of the special eraser. This common behavior causes extra strokes to appear on the captured content. Regarding Mimio, we can mention two more points: 3) People have to use special dry-ink pen adapters, which make them much thicker and harder to press; 4) Imprecision of pen tracking sometimes causes mis-registration of adjacent pen strokes.

2.2 Multimedia Recording Systems

Studies have shown that capturing multimedia experience such as lectures or meetings is useful either for later reviewing by the participants or for the people who did not attend [1,5,6,8,11,16]. A lot of research has been done on the capture, integration, and access of the multimedia experience. People have developed techniques and systems that use handwritten notes, whiteboard content, slides, or manual annotations to index the recorded video and audio for easy access [1,4,8,10,11,15,16,17,18].

Our work is partly inspired by Lisa Stifelman's work on Audio Notebook, a prototype notepad combining pen-and-paper and audio recording [11,13]. Audio Notebook provides structure to the recorded audio based on the synchronization of the key points marked by pen on paper.

Another related work is the Classroom2000 project as reported by Abowd et al. [1,2,3]. In the Classroom2000 project, Abowd et al. used an electronic whiteboard to time-stamp the ink strokes so that the viewer (students) can use the inks as the indexes to the recorded video and audio. They also computed key frames (called pages) based on the erasing events provided by the electronic whiteboard. The main difference between their work and ours is that we use an ordinary whiteboard while they require an electronic whiteboard. With electronic whiteboard, it is trivial to obtain the time stamps of the pen strokes and know when the strokes are erased. But there are many disadvantages from the end user's point of view. First of all, most of the offices and meeting rooms do not have electronic whiteboards installed. Secondly, as reported in [1], people found that it is much more natural to use a regular whiteboard than an electronic whiteboard. Thirdly, images captured with a camera provide much more contextual information such as who was writing and which topic was discussing (usually by hand pointing).

3 SYSTEM DESIGN

The design goals of the Whiteboard Capture System are:

1. Work with any existing whiteboard.
2. Capture the whiteboard content automatically and reliably.
3. Use the whiteboard content as a visual index to efficiently browse the recorded meeting.

In designing the system, we had several design questions: 1) Which capture device to use? 2) How do users record meetings? 3) How do we present the browsing user

interface? 4) How do we keep the recorded meeting data secure? The design is an iterative process. In order to collect feedback early in the design phase, we installed capture systems in three conference rooms nearby. Not only our team members use it frequently as the system evolves, we also encourage other groups to use the system.

3.1 Capture Device

One important design goal of our system is to work with any existing whiteboard. Even though Mimio system works with existing whiteboard, it requires people to use special pens and erasures, and there are various problems as mentioned in the previous section. Therefore we choose to use a direct capture device. Without requiring special pens and erasers makes the interaction much more natural. Since it is taking pictures of the whiteboard directly, there is no mis-registration of the pen strokes. As long as the users turn on the system before erasing, the content will be preserved. Our Whiteboard Capture System uses a 4-mega pixel digital still camera to capture the whiteboard content (see Figure 1). The camera provides images that are 2272 pixels by 1704 pixels -- equivalent to 31.6 dpi for a 6' by 4' board. Using a mass market consumer product as opposed to proprietary hardware can potentially lower the cost.

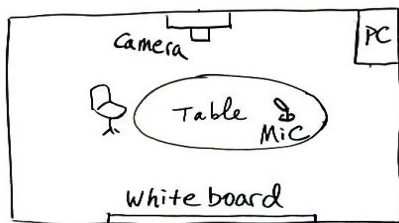


Figure 1: A typical Whiteboard Capture Installation. Note that this image was captured from one of our whiteboard sessions.

3.2 Capture Interface

Another important feature differentiating our system from others is that people are not required to move out of the camera's field of view during capture as long as they do not block the same portion of the whiteboard during the whole meeting.

In our initial implementation, we asked the meeting participants to press a "Key Frame" button on the side of the whiteboard every time before they want to erase a portion of the whiteboard. This design simplifies the computation in two ways: 1) The button is located off the side of the whiteboard, pressing button makes the whiteboard temporarily fully exposed to the camera; 2) The button press events inform the analysis process which image frames are un-obscured key frames.

With the assumption that key frames contain only and all pen strokes, computing the time stamp of each pen stroke becomes straightforward: start from the key frame, simply search forward for the first frame when the same image pattern appears.

However after a short trial period, the users of the system found the interface restrictive. They did not want to remember pressing the button before each erasure. They

simply want the system to figure out the key frames automatically. This UI requirement adds significant complexity to our analysis algorithm because we have to reconstruct the whiteboard content from different snapshots. A description of our algorithm is detailed in Section 4.

3.3 Browsing Interface

Since most people probably do not want to listen to the recorded meeting from start to end, we provide two browsing features to make non-linear accessing of the recorded information very efficient.

1. **Key Frames:** Key frame images contain all the important content on the whiteboard and serve as a summary to the recording. They can be cut and pasted to other documents or printed as notes.
2. **Visual Indexing:** We provide two levels of non-linear access to the recorded audio. The first is to use the key frame thumbnails. The user can click a thumbnail to jump to the starting point of the corresponding key frame. The second is to use the pen strokes in each key frame. Together with the standard time line, these two levels of visual indexing allow the user to browse a meeting in a very efficient way.

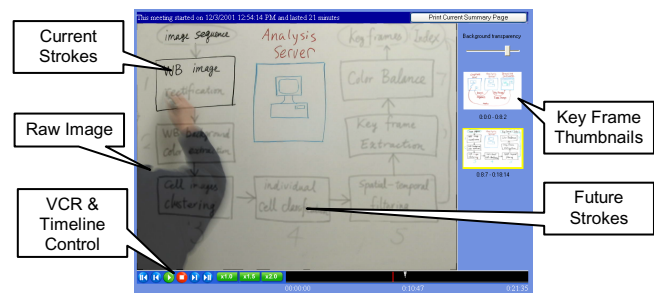


Figure 2: Browsing interface. Each key frame image represents the whiteboard content of a key moment in the recording. The main window shows a composition of the raw image from the camera and the current key frame image. The pen-strokes that the participants are going to write in the future (*Future Strokes*) are shown in ghost-like style.

3.4 Security

Meeting participants are usually apprehensive about recording the meeting because sensitive information might be viewed by unintended people. For them, keep the recorded data secure is a concern. To address this concern, we developed a simple token-based access model.

We ask meeting participants to register with the capture software at beginning of the recording. They can either fill in their email aliases in a dialog box or, to speedup the process, insert their corporate id cards into a smartcard reader to register.

All the recorded sessions reside on a web server. If no one registers, the meeting is posted on a public accessible web page. If at least one participant registered, an access token is generated after the analysis. The token is a long string randomly generated with the unique meeting id. The URL containing the token is emailed to the registered participants. The recipients can click the URL to launch the

web browsing software to review the meeting. They can also forward the URL to people who have not attended the meeting.

This simple Security-by-Obscurity model seems to work well during our initial deployment although more security and privacy measures are needed in a productized system.

3.5 System Architecture

Conceptually, the overall Whiteboard Capture System consists of three components: the capture unit, the processing server, and the browsing software (see Figure 3).

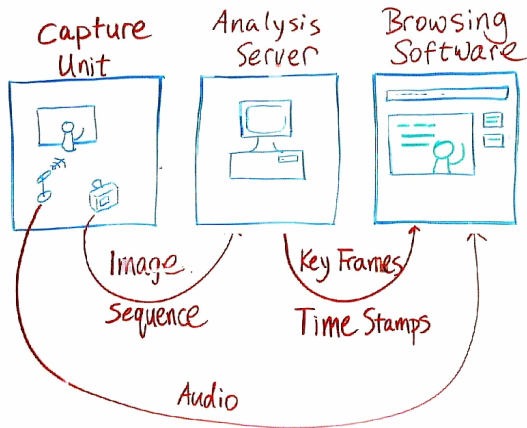


Figure 3: The system consists of three components: the capture unit, the analysis server, and the browsing software.

1. **Capture unit** is installed in the room where meetings take place. It includes a 4 mega-pixel digital camera, a microphone, and a PC.
2. **Analysis server** is located in a central place and stores the recorded data. An analysis program is launched automatically after the user stops the recording in the capture unit. After processing, emails containing the URL to meeting recording are sent to the registered participants.
3. **Browsing software** is a web plug-in installed by the users who wish to view the recordings. Once installed, the users can click the URL to launch the software to access the data on the analysis server.

4 TECHNICAL CHALLENGES

The input to the Whiteboard Capture System is a set of still digital images (see Figure 4). We need to analyze the image sequence to find out when and where the users wrote on the board and distill a set of key frame images that summarize the whiteboard content throughout a session.

Compared to the sensing mechanism of devices like Mimio or electronic whiteboard, our system has a set of unique technical challenges: 1) The whiteboard background color cannot be pre-calibrated (e.g. take a picture of a blank whiteboard) because each room has several light settings that may vary from session to session; 2) Frequently, people move between the digital camera and the whiteboard, and these foreground objects obscure some portion of the

whiteboard and cast shadow on it. Within a sequence, there may be no frame that is totally un-obscured. We need to deal with these problems in order to compute time stamps and extract key frames.

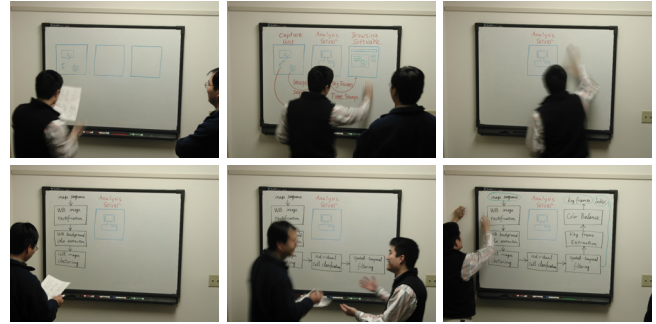


Figure 4: Selected frames from an input image sequence. The session lasts about 16 minutes and contains 195 frames.

4.1 Image Acquisition

Currently we use the Canon PowerShot G2 camera with 4 mega pixels. One important reason that we choose G2 is the availability of a Software Development Kit (SDK) which allows us to write customized software solutions to control the camera from a PC. Our software can specify virtually all the camera parameters on a per-shot basis.

The camera is mounted at either the side or the back of a meeting room. We zoom the camera as close to the whiteboard as possible to maximize the effective resolution. The camera is stationary after the installation and we assume the whiteboard does not move, so the whiteboard images are stationary throughout the captured sequence.

Since the G2 camera has only auto focus mode, the whiteboard might become out-of-focus if an object in front of the whiteboard triggers the attention of the auto focus mechanism of the camera. We mitigate this problem by doing two things: 1) Align image plane of the camera as parallel to the whiteboard as possible to minimize scene depth; 2) Minimize the aperture to increase the depth of field. In practice, only 1-2% of the frames are out-of-focus.

The camera takes the pictures as fast as it can and transfers the images to the PC via USB. Using the G2 camera, we are able to get one JPEG image about every 5 seconds. We keep the exposure and white-balance parameters constant. Assuming the light setting does not change within one session, the color of whiteboard background should stay constant in a sequence.

We found slightly under exposed the images give better color saturation, which makes the stroke extraction process more accurate. A color-balance step is performed at the end to make the grayish whiteboard images more appealing (see Section 4.2.7).

4.2 Image Sequence Analysis

Since the person who is writing on the board is in the line of sight between the digital camera and the whiteboard, he/she often obscures some part of the whiteboard and casts shadow on other part. We need to distinguish among strokes, the foreground object, and the whiteboard. Once

we know the classification results, we can produce the key frame images and an index to be used by the browsing software.

Rather than analyze the images on a per-pixel level, we divide the whiteboard region into rectangular *cells* to lower the computational cost. The cell size is roughly the same as what we expect the size of a single character on the board (about 1.5 by 1.5 inches in our implementation). Since the cell grid divides each frame in the input sequence into *cell images*, we can think of input as a 3D matrix of cell images.

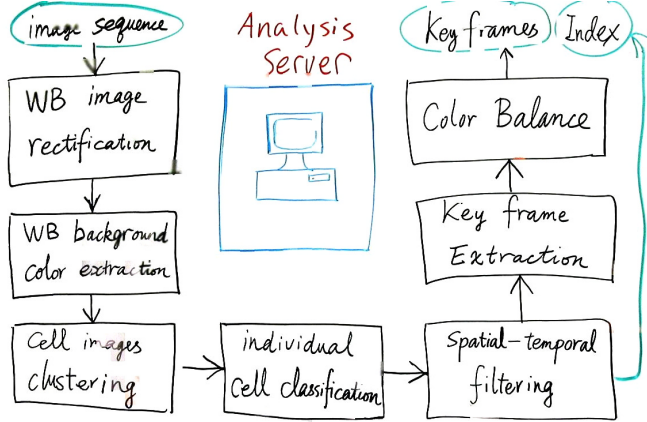


Figure 5: The image sequence analysis process.

Here is an outline of the algorithm that we use.

1. **Rectify** the whiteboard region of every image in the sequence.
2. **Extract** the whiteboard background color.
3. **Cluster** the cell images throughout the sequence for the same cell. If two cell images are considered to be the same, they are clustered in the same group.
4. **Classify** each cell image as a stroke, a foreground object, or the whiteboard.
5. **Filter** the cell images both spatially and temporally to refine the classification results.
6. **Extract** the key frame images using the classification results.
7. **Color-balance** the key frame images.

In the following, we use the running example as shown in Figure 4 to illustrate our algorithm.

4.2.1 Rectify the Whiteboard Images

Before feeding the image sequence to the stroke extraction process, we crop the non-whiteboard region and rectify the images. Because the lens of the G2 camera has fairly low radial distortion, we only need to specify the four corners of the whiteboard. This is currently done manually by clicking a captured image during the one-time calibration step, although this could be done automatically. With the four

corners, a simple bi-linear warp is performed for each image in the sequence using bi-cubic interpolation.

4.2.2 Computing the Whiteboard Color

For the classification of the cells, we need to know for each cell what the whiteboard color is (that is, the color of the whiteboard itself without anything written on it). The whiteboard color is also used for white-balancing in producing the key frames, so it needs to be estimated accurately to ensure the high quality of the key frame images.

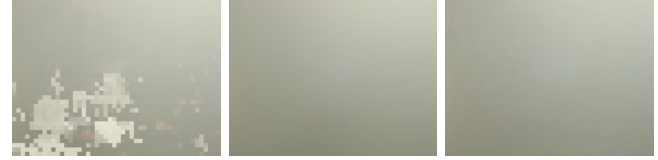


Figure 6: Whiteboard color extraction results. The left image is the result of the 1st strategy, the middle image is the result of the 2nd strategy, and the right image shows the actual blank whiteboard image.

We have experimented with two strategies. The first is based on the assumption that the whiteboard cells have the brightest luminance over time and have small variance (i.e., almost uniform within each cell). This is reasonable since the color of the strokes (red, green, blue or black) will lower the luminance. This, however, may produce holes in the final whiteboard color image, for example, if a cell either contains a stroke or is blocked by a foreground object throughout the sequence. To fill a hole like this, we search its neighborhood, and set its whiteboard color to that of the nearest cell which is not a hole. This strategy usually works quite well, but it fails when a person wears a white T-shirt and/or holds a piece of white paper. The left image of Figure 6 shows the result of the whiteboard color image computed from the input sequence in Figure 4 where a person was holding a white paper in some of the frames. We can see that the computed whiteboard color is corrupted by the white paper.

The second strategy is more sophisticated. The assumption is that a significant portion of the pixels in each cell over time belongs to the whiteboard. By building a histogram of the luminance, the color corresponding to the peak with a high luminance value is very likely the color of the whiteboard. The first step is therefore to compute an initial whiteboard color in this way. It works even if a cell contains a stroke throughout the sequence, but it fails in the case when a person wears a white T-shirt and/or holds a piece of white paper, or when a cell is always hidden by people or other objects. In such cases, the computed whiteboard color image contains *outliers*. The second step is to detect those outliers. The outlier detection is based on a robust technique called *least-median-squares* [11]. Assuming the color varies smoothly across the whiteboard, a plane is fit in the luminance Y or RGB space by minimizing the median of the squared errors. The cells whose color does not follow this model are considered to be outliers and consequently rejected, i.e., they are marked as holes. The interested reader is referred to the Appendix for the details of this technique. The third step is to fill the

holes by using the same procedure as in the first strategy. Finally, to further improve the result, we filter the whiteboard color image by locally fitting a plane in the RGB space. The interested reader is again referred to the Appendix for details. The result obtained with this new technique on the same example is shown in the middle image of Figure 6. We see clear improvements over the result obtained with the first strategy as shown in the left. We also show the actual blank whiteboard in the right image for comparison.

4.2.3 Clustering Cell Images over Time

During the meeting, the content of each cell usually changes over time. For each cell, we would like to cluster all the cell images in the sequence into groups, where each group contains the cell images which are considered to be the same. We use a modified Normalized Cross-Correlation algorithm to determine if two cell images are the same or not. In the following, we describe the Normalized Cross-Correlation technique using one component of the image, but it applies to all RGB components.

Consider two cell images I and I' . Let \bar{I} and \bar{I}' be their mean colors and σ and σ' be their standard deviations. The normalized cross-correlation score is given by

$$c = \frac{1}{N\sigma\sigma'} \sum_i (I_i - \bar{I})(I'_i - \bar{I}')$$

where the summation is over

every pixel i and N is the total number of pixels. The score ranges from -1, for two images not similar at all, to 1, for two identical images. Since this score is computed after the subtraction of the mean color, it may still give a high value even two images have very different mean colors. So we have an additional test on the mean color difference based on the Mahalanobis distance [7], which is given by $d = |\bar{I} - \bar{I}'| / (\sigma + \sigma')$. In summary, two cell images I and I' are considered to be identical and thus should be put into the same group if and only if $d < T_d$ and $c > T_c$. In our implementation, $T_d = 2$ and $T_c = 0.707$.

4.2.4 Classifying Cells

This step is to determine whether a cell image is a whiteboard, a stroke, or a foreground object. We use the following heuristics: 1) a whiteboard cell is uniform in color and is grey or white (i.e., the RGB values are approximately the same); 2) a stroke cell is mostly white or grey with one or two primary colors mixed in; 3) a foreground object does not have the characteristics above. The classification is therefore to determine whether the color distribution of the current cell image and the whiteboard color distribution are the same, or not the same but having strong overlap, or totally different. Again, we use the Mahalanobis distance [7] as described below.

Notice that the whiteboard color has already been computed as described in Sect.4.2.2. Again, we use one component of RGB as an example. Let \bar{I}_w be the whiteboard color and σ_w be the standard deviation (it is a small value since a

whiteboard cell is approximately uniform). Let \bar{I} and σ be the mean and standard deviation of the current cell image. The cell image is classified as a whiteboard cell if and only if $|\bar{I} - \bar{I}_w| / (\sigma + \sigma_w) < T_w$ and $\sigma / \sigma_w < T_\sigma$; as a stroke cell if and only if $|\bar{I} - \bar{I}_w| / (\sigma + \sigma_w) < T_w$ and $\sigma / \sigma_w \geq T_\sigma$; otherwise, as a foreground object cell. In our implementation, $T_w = 2$ and $T_\sigma = 2$.

4.2.5 Filtering Cell Classification

Notice that the above classification algorithm only uses the color information in a single cell. More accurate results can be achieved by utilizing spatial and temporal relationship among the cell groups.

Spatial filtering. The basic observation is that foreground cells should not appear isolated spatially since a person usually blocks a continuous region of the whiteboard. So we perform two operations on every single whiteboard image. First, we identify isolated foreground cells and reclassify them as strokes. Second, we reclassify stroke cells which are immediately connected to some foreground cells as foreground cells. One main purpose of the second operation is to handle the cells at the boundaries of the foreground object. Notice that if such a cell contains strokes, the second operation would incorrectly classify this cell as a foreground object. But fortunately, the following temporal filtering will correct such potential errors.

Temporal filtering. The basic observation is that it is virtually impossible to write the same stroke in exactly the same position after it is erased. In other words, if for any given cell, the cell images of two different frames contain the same stroke, then all the cell images in between the two frames must have the same stroke unless there is a foreground object blocking the cell. This observation is very useful to segment out the foreground objects. Consider the example in the previous section where a stroke cell at the boundary of the foreground object is incorrectly classified as a foreground cell. At the temporal filtering step, this cell will be classified as a stroke as long as it is exposed to the camera before and after the foreground object blocks it.

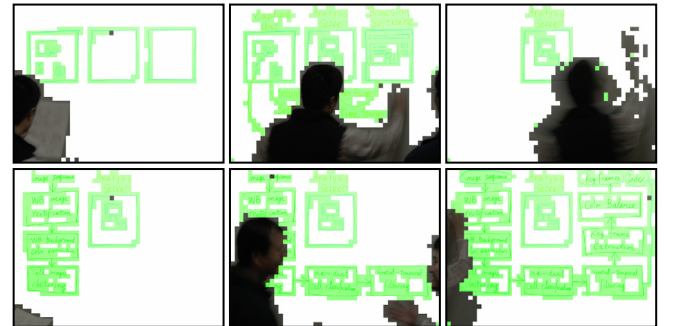


Figure 7: Samples of the classification results. The images above correspond to the images in Figure 4 after cropping and rectification.

Figure 7 shows the classification results for the sample images in Figure 4, where the strokes are in green, the foreground is in black, and the whiteboard is in white.

4.2.6 Key Frame Image Extraction

Key frame images are the summary of a whiteboard session. The user would expect the key frame images to have the following properties: 1) They should capture all the important content on the board; 2) The number of the key frames should be kept to a minimum; 3) They should only contain the pen strokes and the whiteboard, but not the person in front; 4) They should have uniform white background and saturated pen colors for easy cut-and-paste and printing.

The key frame extraction algorithm uses the cell images classification results from the previous step. The algorithm first decides which frames in the sequence should be selected as key frames; it then reconstructs the key frame images.

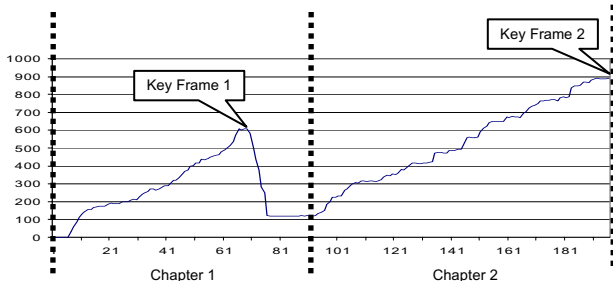


Figure 8: Key frame extraction. A plot of number of strokes vs. time for the sequence in Figure 4. The corresponding two key frame images are shown in Figure 3 and Figure 5.

Key frame selection. There is no unique solution in selecting the key frames -- just as there is no single way to summarize a meeting. In our system, we first divide the meeting into several chapters (topics) then create a key frame image representative of the whiteboard content for that chapter. An erasure of a significant portion of the board content usually indicates a change of topic so we use it as a divider of the chapters. We choose the frame just before an erasure starts as the key frame, which ensures that the content is preserved in those frames. The detailed algorithm works as follows:

1. Count the number of stroke cells for each frame in the sequence. Note that one stroke cell image may span multiple frames -- it is included in the count for each of those frames. Figure 8 shows the stroke cell count plotted against frame number in the example session (Figure 4). A rise in the plot indicates more strokes are written on the board, where a dip in the plot indicates some strokes are erased. As we can see the graph is quite noisy. There are two reasons: 1) The user is constantly making small adjustments on the board; 2) The classification results contain small errors.
2. If we produce a key frame at each dip, we will get dozens of key frames. In order to keep the number

of key frames to a minimum, the data is filtered to retain only the significant erasure events. Our algorithm ignores the fluctuation in the data unless the difference between the adjacent peak and valley exceeds a certain threshold. We use 20% of the maximum stroke count in our system.

3. The valleys in the data divide the session into chapters. The frame containing the peak within a chapter is chosen to be the key frame representing the chapter.

Image reconstruction. Once the frames are selected, we need to reconstruct the images corresponding to what the whiteboard looks like at these points of time. But we cannot simply use the raw images from the input sequence because they may contain foreground objects. We reconstruct the image by gathering the cell images in the frame. There are three cases depending on the cell classification:

1. If a cell image is whiteboard or stroke, its own image is used.
2. If the foreground cell image is within the span of a stroke (i.e., the person is obscuring the strokes on the board), we replace this cell image with the stroke cell image from the neighboring frames.
3. Otherwise, a foreground object must be covering the whiteboard background in this cell, and we simply fill it with its whiteboard color computed in Section 4.2.2.

4.2.7 Key Frame Color Balance

The reconstruction process removes the person from the whiteboard images, but the images still look like the raw images from the input sequence: grayish and noisy. They need to be color balanced. The process consists of two steps:

1. Make the background uniformly white and increase color saturation of the pen strokes. For each cell, the whiteboard color computed in Section 4.2.2, \bar{I}_w , is used to scale the color of each pixel in the cell. $I_{out} = \min(255, \frac{I_{in}}{\bar{I}_w} \cdot 255)$
2. Reduce image noise. We remap the value of each color channel of each pixel in the key frames according to an S-shaped curve.

The final images from our running example (Figure 4) are Figure 3 and Figure 5 which we used earlier in this paper. The beginning and ending times of the chapters and the file names of their key frame images are saved in the index along with the time stamps of the strokes produced in Section 4.2.5.

4.3 Browsing Experience

After the analysis server processed the image sequence and produced the index and key frame images, it sends emails to the session participants with the URL to the processed

recording. The users can click the URL to launch the browsing software (see Figure 2). The goal of the browsing software is to allow users to view the key frame images and quickly access the audio associated with a particular topic.

4.3.1 Image Viewing Mode

The thumbnails of the key frame images are listed in the right pane. Clicking one of the thumbnails brings the corresponding key frame image to the main window at the left and takes the application to the image viewing mode, where the user can zoom in and out, read the text and diagrams in the image, or cut and paste a portion of the image to other documents.

4.3.2 Audio Playback Mode

When the cursor is hovering over a pen stroke cell, the cursor is changed to a hand symbol indicating that it is clickable. Double clicking the cell brings the application to the audio playback mode. The playback starts from the time of the session when the clicked stroke cell was written. The user can still click other stroke cells to jump to other part of the session.

4.3.3 Whiteboard Content Visualization

In the audio playback mode, how do we visualize the whiteboard content in the main window? Given the key frame images and the time stamp information, we can reconstruct an image that corresponds to the whiteboard content at any given time. If we render the image every frame according to the audio playback time, the main window plays back the whiteboard content like a movie. Using this approach, the users will have both the aural and visual context to the session. But they cannot click any pen stroke that takes them forward in time (*Future Strokes*) because these strokes have not yet been rendered in the main window.

In our initial implementation, we borrowed the approach in Classroom2000 [2]: show the future strokes in a washed out mode. However after a short trial period, the users of the browser often confuse the future strokes with the strokes that are not cleanly erased. Another complaint about the interface is that although the users like the whiteboard images without the person in front, they sometimes want to know who wrote the strokes.

After a few design iterations, we decide on the following visualization that addresses all those concerns:


1. Render the current whiteboard content using the key frame image of the current chapter and time stamp information.
2. Render the Future Strokes, convert the results to grey scale, and blur them using a Gaussian filter.
3. Add images from Step 1 and Step 2.
4. Alpha-blend the image from Step 3 with the rectified image from the input sequence. The user can control the alpha value with a GUI slider from 0, showing only the rendered whiteboard image, to 1, showing exactly the original image.

See Figure 2 for an example of such a visualization with $\alpha=0.8$.

5 SYSTEM PERFORMANCE AND USAGE

We have equipped three conference rooms with the Whiteboard Capture System. Information about those three rooms is listed in Table 1. The sizes of whiteboards in those rooms vary and so do the qualities of the key frame images produced. As we can see from the sample images, the writings on a 12'x5' board are fuzzier than the ones on the other two boards because the resolution is maxed out for a 4 mega-pixel input image. Nevertheless, they are still quite legible. Several selected frames from a session using a 12'x5' whiteboard and the key frame images are shown in Figure 9.

Table 1: Information about the three installation sites.

	Room 1	Room 2	Room 3
Board Dimension (ft)	4x3	8x5	12x5
Key Frame Image Dimension (pixel)	1200x900	2400x1500	2400x1000
Resolution (dpi)	25	25	16.7
Sample Images (80x80 pixels, approx. 96 pt font on the board)			

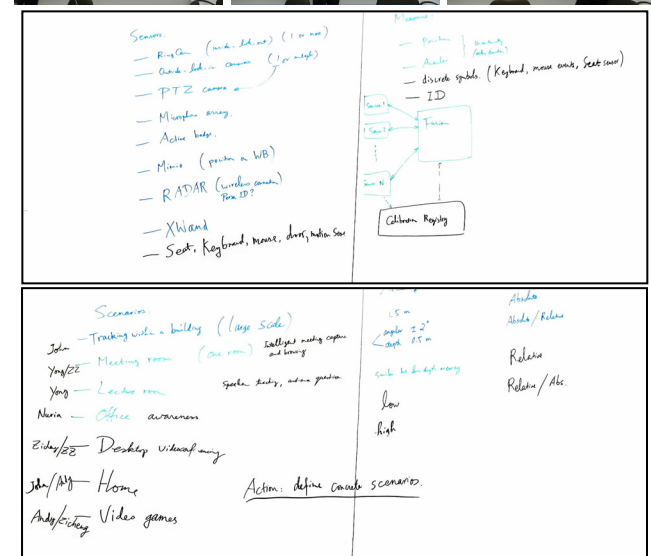


Figure 9: A session using the 12'x5' whiteboard. This session is 1 hour and 5 minutes long. Shown are 6 sample images and 2 extracted key frame images.

The analysis server runs on a Pentium III 800MHz dual CPU PC. The analysis process takes about 20 minutes for every hour of session time. The storage requirement for the 16 bit 11 KHz mono audio takes about 15 Mb per hour using MP3 encoding. The input image sequence requires about 34 Mb per hour using Motion JPEG compression.

The systems installed in three conference rooms are used frequently not only by our own team members but also by colleagues from neighboring groups. Over the course of 6 months, we recorded 108 sessions totaling 48 hours -- averaging 27 minutes per session and 4.5 sessions per week. The average number of key frames per session is 2.7. The key frame images are saved in JPEG format. The average image size is 51.8 Kb. The sizes range from 17 Kb to 150 Kb. Because the JPEG compression works extremely well on the uniform white background, the image size is more related to how much the users write on the board than the image dimension.

All users of our system believe that our system is very useful for meetings that use whiteboard extensively: The key frame images and the visual indexing capability not only allow the participants to review a meeting at a later time, but also allow the users who did not attend the meeting to understand the gist of the meeting in a fraction of the actual meeting time. "I would die to have such a system in all of our disclosure meetings", a patent attorney claimed after using the system to browse the disclosure meeting he just had.

We are pleasantly surprised by some users who found new ways to use the system which we did not intend initially. Take an example of status meetings which usually did not require writing on whiteboard. People still turned on the whiteboard capture system. When it was someone's turn to speak, the manager wrote his/her name on the board so that the speech segments could be easily found later in the recorded audio by clicking on the names in the key frame image. Another example is during a brainstorm session, when someone thought of a good idea, he wrote a star on the side of the board and said it aloud. The audio can then be retrieved later by clicking on the star.

6 LIMITATIONS OF OUR CURRENT SYSTEM

Under some very special circumstances, our system may fail to produce the desired results. For example, if a person stands perfectly still in front of the whiteboard for an extended period, our system would not be able to determine that it is a person. The cells covered by the person would be either treated as strokes or whiteboard depending on the textures of his/her cloth.

If a region of the whiteboard is never exposed to the camera, our system would not be able to figure out the content in that region.

Currently, our analysis algorithm requires the color of the whiteboard background to remain constant in an input sequence. The requirement assumes constant lighting and constant camera exposure setting throughout a meeting which seems to work in our initial deployment, but a more

flexible approach might be to install a known color patch above the top of the whiteboard where nobody can obscure from the camera. The software can then adjust the camera exposure parameters for different lighting conditions on a per-frame basis.

The frame rate of our system is limited by the frame rate of the commercially available still cameras. To achieve higher frame rate, one possibility is to use a high resolution video camera such as HDTV cameras at higher cost. Another possibility is to use super-resolution techniques to increase the resolution of a regular video camera.

7 CONCLUDING REMARKS

Meetings constitute a large part of knowledge workers' working time. Making more efficient use of this time translates to a big increase in their productivity. The work presented in this paper, the Whiteboard Capture System, focuses on the meeting scenarios that use whiteboard heavily: brainstorming sessions, lectures, project planning meetings, patent disclosures, etc.

Our system alleviates the participants of those meetings the mundane note-taking task, so they can focus on contributing and absorbing ideas during the meetings. By providing key frame images that summarize the whiteboard content and structured visual indexing to the audio, our system helps the participants to review the meeting at a later time. Furthermore, the users who did not attend the meeting can often understand the gist of the meeting in a fraction of meeting time.

From our initial deployment, we found the meeting participants started to change their behaviors when the recording and indexing capabilities of the system are discovered. We believe that as the participants become more and more familiar with the system, the summaries and indices produced by the system will become more useful.

Appendix: Plane-based whiteboard color estimation

We only consider one component of the color image, but the technique described below applies to all components (R, G, B, or Y). Each cell i is defined by its image coordinates (x_i, y_i) . Its color is designated by z_i ($z=R, G, B, \text{ or } Y$). The color is computed as described in Section xxx, and is therefore noisy and even erroneous. From our experience with the meeting rooms in our company, the color varies regularly. It is usually much brighter in the upper part and becomes darker toward the lower part, or is much brighter in one of the upper corners and becomes darker toward the opposite lower corner. This is because the lights are installed against the ceiling. Therefore, for a local region (7x7 cells in our case), the color can be fit accurately by a plane; for the whole image, a plane fitting is still very reasonable, and provides a robust indication whether a cell color is an outlier.

A plane can be represented by $ax + by + c - z = 0$. We are given a set of 3D points $\{(x_i, y_i, z_i) \mid i=1, \dots, n\}$ with noise only in z_i . The plane parameters $\mathbf{p} = [a, b, c]^T$ can be estimated by minimizing the following objective function:

$F = \sum_i f_i^2$, where $f_i = ax_i + by_i + c - z_i$. The least-squares solution is given by $\mathbf{p} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{z}$, where

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & 1 \\ \cdots & \cdots & \cdots \\ x_n & y_n & 1 \end{bmatrix} \text{ and } \mathbf{z} = [z_1, \dots, z_n]^T. \text{ Once the}$$

plane parameters are determined, the color of the cell i is replaced by $\hat{z}_i = ax_i + by_i + c$.

The least-squares technique is not robust to erroneous data (outliers). As mentioned earlier, the whiteboard color we initially computed does contain outliers. In order to detect and reject outliers, we use a robust technique to fit a plane to the whole whiteboard image. We use the least-median-squares [11], a very robust technique that is able to tolerate near half of the data to be outliers. The idea is to estimate the parameters by minimizing the median, rather than the sum, of the squared errors, i.e., $\min_p \text{median}_i f_i^2$. We first

draw m random subsamples of 3 points (3 is the minimum number to define a plane). Each subsample gives an estimate of the plane. The number m should be large enough such that the probability that at least one of the m subsamples is good is close to 1, say 99%. If we assume that half of the data could be outliers, then $m = 35$, therefore the random sampling can be done very efficiently. For each subsample, we compute the plane parameters and the median of the squared errors f_i^2 . We retain the plane parameters that give the minimum median of the squared errors, denoted by M . We then compute the so-called robust standard deviation $\sigma = 1.4826\sqrt{M}$ (the coefficient is used to achieve the same efficiency when no outliers are present). A point i is considered to be an outlier and discarded if its error $|f_i| > 2.5\sigma$. Finally, a plane is fit to the good points using the least-squares technique described earlier. The color of an outlier cell i is replaced by $\hat{z}_i = ax_i + by_i + c$.

REFERENCES

1. Abowd, G. D., Atkeson, C. G., Jason A., Brotherton, J. A., Enqvist, T., Gulley, P. & Lemon, J., Investigating the capture, integration and access problem of ubiquitous computing in an educational setting. In the *Proceedings of CHI '98*, pp. 440-447, May, 1998.
2. Brotherton, J. A. & Abowd, G. D., Rooms Take Note: Rooms take notes! Working Papers of AAAI'98 Spring Symposium, March 1998.
3. Brotherton, J. A., Abowd, G. D. & Truong, K. N., Supporting Capture and Access Interfaces for Informal and Opportunistic Meetings, Georgia Tech Technical Report, GIT-GVU-99-06. (1998).
4. Chiu, P., Kapuskar, A., Reitmeier, S., and Wilcox, L. NoteLook: Taking notes in meetings with digital video and ink. *Proceedings of ACM Multimedia '99*. ACM, New York, pp. 149-158.
5. Chiu, P., Kapuskar, A., Reitmeier, S., and Wilcox, L. Meeting capture in a media enriched conference room. *Proceedings of CoBuild '99*. Springer-Verlag LNCS 1670, pp. 79-88.
6. Chiu, P., & Wilcox, L. D., A dynamic grouping technique for ink and audio notes. *Proceedings of UIST'98*.
7. Duda, R.O., Hart, P.E. and Stork, D.G. *Pattern Classification*, Second Edition, John Wiley & Sons, New York, 2001.
8. Ju, S.X., Black, M.J., Minnerman, S. & Kimber D. Analysis of Gesture and Action in Technical Talks for Video Indexing. In *IEEE Trans. on Circuits and Systems for Video Technology*.
9. Moran, T. P., Palen, L., Harrison, S., Chiu, P., Kimber, D., Minnerman, S., Melle, W. v. & Zellweger, P., "'I'll Get That Off the Audio": A Case Study of Salvaging Multimedia Meeting Records," in *Proceedings of CHI '97*, Atlanta, GA, 1997.
10. Pedersen, E., McCall, K., Moran, T. P., & Halasz, F., Tivoli: An electronic whiteboard for informal workgroup meetings. *Proceedings of INTERCHI'93*. pp391-389.
11. Rousseeuw, P. and Leroy, A. *Robust Regression and Outlier Detection*, John Wiley & Sons, New York, 1987.
12. Stifelman, L., The audio notebook. Ph.D. Thesis, MIT Media Laboratory, September, 1997.
13. Stifelman, L.J., Arons, B., Schmandt, C. & Hulthen, E.A. VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker. *Proc. INTERCHI'93 (Amsterdam, 1993)*, ACM
14. Saund, E. Image Mosaicing and a Diagrammatic User Interface for an Office Whiteboard Scanner. Technical Report, Xerox Palo Alto Research Center, 1999.
15. Weber, K. & Poon, A., Marquee: A tool for real-time video logging. *Proceedings of CHI'94*. pp 58-64.
16. Wilcox, L. D., Schilit, B. N. & Sawhney, N., Dynamite: A dynamically organized ink and audio notebook. *Proceedings of CHI'97*. pp 186-193.
17. Whittaker, S., Hyland, P. & Wiley, M., Filochat: Handwritten notes provide access to recorded conversations. *Proceedings of CHI'94*. pp 271-276.
18. Wolf, C., Rhyne, J. & Briggs, L., Communication and information retrieval with a pen-based meeting support tool. *Proceedings of CSCW'92*. pp 322-329.