Real-Time Whiteboard Capture and Processing Using a Video Camera for Teleconferencing

Li-Wei He & Zhengyou Zhang
Microsoft Research, One Microsoft Way, Redmond, WA, USA
{lhe,zhang}@microsoft.com

Microsoft Research Technical Report: MSR-TR-2004-91

ABSTRACT
This paper describes our recently developed system which captures pen strokes on whiteboards in real time using an off-the-shelf video camera. Unlike many existing tools, our system does not instrument the pens or the whiteboard. It analyzes the sequence of captured video images in real time, classifies the pixels into whiteboard background, pen strokes and foreground objects (e.g., people in front of the whiteboard), and extracts newly written pen strokes. This allows us to transmit whiteboard contents using very low bandwidth to remote meeting participants. Combined with other teleconferencing tools such as voice conference and application sharing, our system becomes a powerful tool to share ideas during online meetings.

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

1 INTRODUCTION

The Real-Time Whiteboard Capture System (RTWCS) is a part of our Distributed Meetings project, which aims at dramatically improving knowledge workers' meeting experience 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, and patent disclosures. 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 or the content is accidentally erased by other people. Furthermore, meeting participants who are on conference call from remote locations are not able to see the whiteboard content as the local participants do. In order to enable this, the meeting sites often must be linked with expensive video conferencing equipments. Such equipment includes a pan-tilt-zoom camera which can be controlled by the remote participants. It is still not always satisfactory because of viewing angle, lighting variation, and image resolution, without mentioning lack of functionality of effective archiving and indexing of whiteboard contents. Other equipment requires instrumentation either in the pens such as Mimio from Virtual Ink or on the whiteboard such as SMARTBoard from SmartTech.

Our system was designed with two purposes: 1) to alleviate meeting participants the mundane tasks of note taking by capturing whiteboard content automatically; 2) to communicate the whiteboard content to the remote meeting participants in real time using a fraction of the bandwidth required if video conferencing equipment is used.

To the best of our knowledge, all existing systems that capture whiteboard content in real time require instrumentation either in the pens or on the whiteboard. Our system allows the user to write freely on any existing whiteboard surface using any pen. To achieve this, our system uses an off-the-shelf high-resolution video camera which captures images of the whiteboard at 7.5Hz. From the input video sequence, our algorithm separates people in the foreground from the whiteboard background and extracts the pen strokes as they are deposited to the whiteboard. To save bandwidth, only newly written pen strokes are compressed and sent to the remote participants. Furthermore, the images are white-balanced and color-

enhanced for greater compression rate and better viewing experience than the original video.

There are a number of advantages in using a high-resolution video camera over the sensing mechanism of pen devices or electronic whiteboard. They are: 1) Without requiring special pens and erasers makes the interaction much more natural. 2) Since it is taking images of the whiteboard directly, there is no mis-registration of the pen strokes. 3) As long as the users turn on the system before erasing, the content will be preserved. 4) Images captured with a camera provide much more contextual information such as who was writing and which topic was discussing (usually by hand pointing).

As an extension to our system, we also applied our video algorithm to capture handwriting annotations on printed documents in real time – a common scenario in teleconferencing. As to be described, we found only a minor algorithmic change was needed.

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 system. Section 4 explains the technical challenges we encountered while building the system. We discuss the limitations of our system in Section 5 and conclude in Section 6.

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:

2.1.1 Image Capture Devices

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 [10], 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 Polyvision's Whiteboard Photo) to crop the non-whiteboard region and color-balance the images.

The disadvantages of the image capture devices are as follows: 1) They capture the whiteboard content one snapshot at a time so users have to make a conscious decision to take a snapshot of the whiteboard. 2) There are usually a lag between writing on the board and taking a snapshot. Using these devices in real time teleconferencing scenarios is not very natural and convenient.

2.1.2 Pen Tracking Devices

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 adaptors for dry-ink pens and eraser. The adapted 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 can be reconstructed in real time. And 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.

However, the pen-tracking devices have several disadvantages:

1. It requires instrumentation either to the pens and erasers or to the surface that they are writing on. For example, Mimio uses special adapters for dry-ink pens, which make them much thicker and harder to press. Electronic whiteboards are not even compatible with the existing whiteboards. They use touch screens as their writing surfaces, which limit their install base due to high cost and small size.
2. If the system is not turned on or the user writes or erases without using the special pens or erasers, the content cannot be recovered by the device. Many people like to use their fingers to correct small mistakes on the whiteboard. This common behavior causes extra strokes to appear on the captured content.
3. There is usually minor imprecision in the tracked pen coordinates, which tends to accumulate and cause mis-registrations among the neighboring strokes. Furthermore, it does not allow multiple users to write on the whiteboard simultaneously. The image capture devices do not have this problem since they work in a What You See Is What You Get (WYSIWYG) manner.

2.2 Multimedia Recording Systems

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 [1,2,4,6,7,8,9,10,11,12,13,14].

Inspired by those systems, we also developed a Whiteboard Capture System (WCS) about two years ago [15]. The goal of that project is to build a whiteboard capture system that combines the benefits of both image capture devices and pen tracking devices.

What differentiates our WCS from other systems described in Section 2.1.1 is that we take pictures of the whiteboard continuously instead of occasional snapshots. Thus the entire history of changes to the whiteboard content is captured. If the camera and the whiteboard remain static, robust vision algorithm exists to remove the redundancy in multiple stroke images and to detect the time when the strokes first appear. Compact representation and availability of time stamps are the key benefits of the pen tracking devices.

We chose a Canon G2 digital camera as the input device for its high resolution (4 MP) and the availability of a software development kit, which allows us to control the camera from the PC. Because of the available high resolution, we do not need complex camera control such as in the ZombieBoard system [10]. However, because the camera is connected to the host PC via low bandwidth USB 1.1, the frame rate is limited to 5 second per frame. At such a low frame rate, we made no attempt to use it as a real time conferencing tool. Our algorithm was designed to analyze and browse offline meeting recordings. From the input image sequence, we compute a set of key frames that captures the history of the content on whiteboard and the time stamps associated with each pen strokes. A key frame contains all the visual content before a major erasure. This information can then be used as a visual index to the audio meeting recording (see Figure 1 for such a browsing interface).

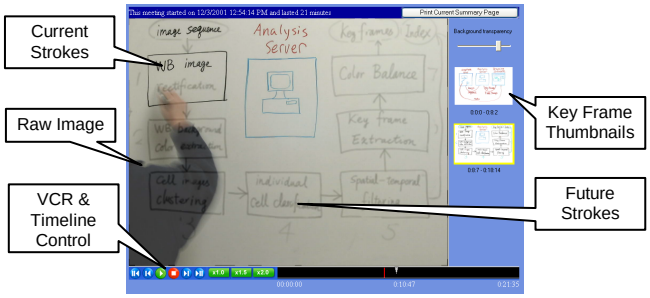

Figure 1: Browsing interface. Each key frame image represents the whiteboard content of a key moment in the recording. The main window shows a composition of the raw image from the camera and the current key frame image. The pen-strokes that the participants have already written are in solid color, while those to be written in the future (*Future Strokes*) are shown in ghost-like style. The user can click on any particular stroke (both past and future) to jump to the point of the meeting when that stroke was written, and listen to what people are talking about.

3 SYSTEM DESIGN

The Real Time Whiteboard Capture System (RTWCS) described in this paper is a real-time extension to our previous WCS system.

3.1 Capture Device

Since building the WCS two years ago, there have been tremendous advances in digital imaging hardware. One notable example is the availability of inexpensive high resolution video cameras and high-speed USB 2.0 connection. For example, with Aplux MU2 video camera connected to any PC with a USB 2.0 port, we are able to capture 1.3 mega pixel images at 7.5 Hz. The resolution of each video frame is 1280 pixels by 1028 pixels --- equivalent to 18 dpi for a 6' by 4' board. At 7.5 Hz, the whiteboard content can be captured in near real time – good enough to use in teleconferences. For our particular scenario, it is a perfect compromise between the NTSC video camera and the high-resolution still image camera. Because of the available high resolution, we do not need complex mechanical camera controls such as in the ZombieBoard system [10].

3.2 Capture Interface Requirements

Like WCS, our system does not require people 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. Unlike WCS, our system does not need special installation or calibration. Sitting on its built-in stand, the video camera can be placed anywhere that has a steady and clear view of the whiteboard. It can be moved occasionally during the meeting. After each move, it will automatically and quickly find the whiteboard region again (see Section 4.1.2). This improvement made our system much more portable and easier to use than the WCS.

Although the camera can be placed anywhere, the intended capture area should occupy as much video frame as possible in order to maximize the available image resolution. For better image quality, it is also better to place the camera right in front of the whiteboard in order to utilize the depth-of-field of the lens to avoid out of focus.

3.3 Automatic Camera Exposure Adjustment

The camera exposure parameter is kept constant. If the light setting does not change, the color of whiteboard background should stay constant in a sequence. We wrote a routine that automatically set the exposure to minimize the number of saturated pixels (i.e. brightness level is 0 or 255). This routine is run once when the system is started and triggered to run again whenever a global change is detected (see Section 4.1.2).

4 TECHNICAL DETAILS

The input to RTWCS is a sequence of video images (see Figure 2). We need to analyze the image sequence in order to separate the whiteboard background from the person in the foreground and to extract the new pen strokes as they appear on the whiteboard.

As mentioned earlier, there are a number of advantages in using a high-resolution video camera over the sensing mechanism of devices like Mimio or electronic whiteboard. However, 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 indoor room has several light settings that may vary from session to session and outdoor room lighting condition is influenced by the weather and the direction of the sun; 2) Frequently, people move between the camera and the whiteboard, and these foreground objects occlude some portion of the whiteboard and cast shadow on it. Within a sequence, there may be no single frame that is completely unoccluded. We need to deal with these problems in order to extract the new pen strokes.

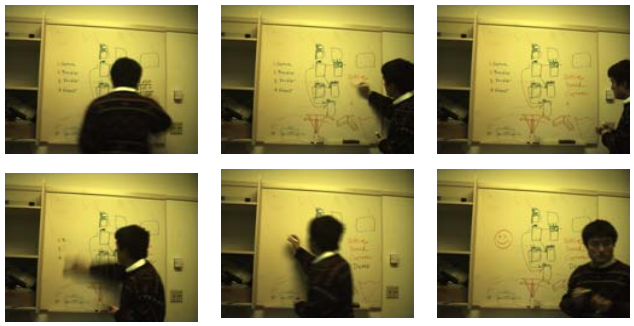

Figure 2: Selected frames from an input image sequence. The sequence lasts 82 seconds.

4.1 Image Sequence Analysis

Since the person who is writing on the board is in the line of sight between the camera and the whiteboard, he/she often occludes some part of the whiteboard. We need to segment the images into foreground objects and whiteboard. For that, we reply on two primary heuristics: 1) Since the camera and the whiteboard are stationary, the whiteboard background cells are stationary throughout the sequence until the camera is moved; 2) Although sometimes foreground objects (e.g., a person standing in front of the whiteboard) occlude the whiteboard, the pixels that belong to the whiteboard background are typically the majority. Our algorithms exploit these heuristics extensively.

We apply several strategies in our analysis to make the algorithm efficient enough to run in real time.

First, rather than analyzing the images at pixel level, we divide each video frame 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 (16 by 16 pixels in our implementation). The cell grid divides each frame in the input sequence into individual *cell images*, which are the basic unit in our analysis.

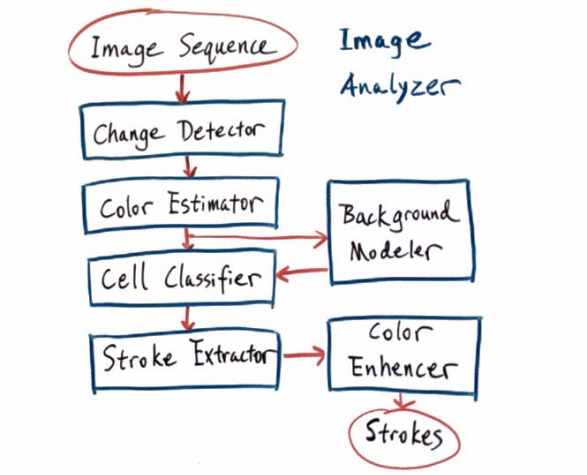

Figure 3: The image sequence analysis process.

Second, our analyzer is structured as a pipeline of six analysis procedures (see Figure 3). If a cell image does not meet the condition in a particular procedure, it will not be further processed by the subsequent procedures in the pipeline. Therefore, many cell images do not go through all six procedures. At the end, only a small number of cell images containing the newly appeared pen strokes come out of the analyzer. The six procedures are:

1. Change detector: it determines if the cell images have changed since last frame.
2. Color estimator: it computes the background color of the cell images -- the color of blank whiteboard.
3. Background modeler: This is a dynamic procedure that updates the whiteboard background model by integrating the results computed from the previous procedure which may have missing parts due to occlusion by foreground objects.
4. Cell classifier: it classifies the cell images into foreground or whiteboard cells.
5. Stroke extractor: it extracts the newly appeared strokes.
6. Color enhancer: it enhances the color of those extracted strokes.

The third strategy is specific to the video camera that we use in our system. The Aplux MU2 allows the video frames to be directly accessed in Bayer format, which is the single channel raw image captured by the CMOS sensor. In general, a demosaicing algorithm is run on the raw image to produce an RGB color image. By processing the cell images in raw Bayer space instead of RGB space and delaying demosaicing until the final step and running it only on the cells containing new strokes, we save memory and processing by at least 66%. An additional benefit is that we can obtain a higher quality RGB image at the end by using a more sophisticated demosaicing algorithm than the one built into the camera driver.

4.1.1 Analysis State

The analysis algorithm keeps some state as it processes the input video frames: 1) The last video frame it has processed; 2) The age of each cell image in the last frame. The age is defined to be the number of frames that the cell image remains unchanged; 3) Cell images with whiteboard content that have been detected so far; 4) The whiteboard background model (see Section 4.1.4 for details).

4.1.2 Assigning Age to Cells

We first assign an age to each cell image. To determine whether a cell image has changed, it is compared against the image of the same cell (e.g., the cell in the same location) in previous frame using a modified Normalized Cross-Correlation (NCC) algorithm. Note that the NCC is applied to the images in the Bayer space.

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 [3], which is given by $d = \left|\bar{I} - \bar{I}'\right| / \sqrt{\sigma^2 + \sigma'^2}$. 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 $T_c = 0.707$.

If the comparison indicates a cell image is changed from the previous frame, its age is set to 1. Otherwise, it is incremented by 1. At each frame, all the cells that have been stationary for more than the age threshold (4 frames in our system -- about 0.5 second at 7.5 Hz) are considered to be the background candidates and fed to the Whiteboard Color Model Update module. If the age is not greater than the age threshold, the cell image is not processed further during this frame. The age threshold is a trade-off between the output delay and analysis accuracy.

We also compute the percentage of the cell images that have changed. If the change is more than 90%, we assume something drastic and global has happened since last frame (e.g. light setting is changed, camera is moved, etc.). In such an event, all state is re-initialized and the exposure calibration routine is called. Other more localized changes (e.g. people moving across, gradual change in sun light) are handled dynamically by the Whiteboard Color Model Update Module.

4.1.3 Computing the Background Color

To classify cells, we need to know for each cell what the whiteboard background color is (i.e. the color of the whiteboard itself without anything written on it). The whiteboard background color is also used in color-enhancing the extracted cell images (see Section 4.1.7), so it needs to be estimated accurately to ensure the quality.

Since the ink absorbs the incident light, the luminance of the whiteboard pixels is higher than pen stroke pixels. The whiteboard color within the cell is therefore the color with the highest luminance. In practice, the colors of the pixels in the top 10th percentile are averaged in order to reduce the error introduced by sensor noise. Hence, the color of each cell is computed by first sorting the pixels of the same color channel (128 green, 64 blue and 64 red values in a 16x16 cell image in Bayer space) and then taking the values of top 10% percentile in each channel.

4.1.4 Updating the Whiteboard Color Model

The color computed from the previous section will give good estimation of whiteboard color for the cells containing some whiteboard background. Though, it will give the wrong color when the cells contain only the foreground or pen strokes (1st image in Figure 4). We have to identify those cells to prevent them from contaminating the whiteboard color model.

Figure 4: 1) Colors of the cell images. Note that the strokes on the whiteboard are removed by our background color estimation algorithm. **2)** Colors of the cell images that go into the Update Module. Note the black regions contain the cell colors that are filtered out by both the change detector and the color estimator. **3)** Integrated whiteboard color model.

We use a least-media-squares algorithm, which fits a global plane over the colors and throws away the cells that contain outlier colors (see Appendix for details). The remaining cells are considered as background cells and their colors are used to update the whiteboard background (2nd image in Figure 4).

We then use a Kalman filter to dynamically incorporate the background colors computed from the current frame into the existing whiteboard background color model. The state for the cell *i* is its color C_i, together with variance P_i representing the uncertainty. P_i is initially set to ∞ to indicate no observation is available. The update is done in two steps:

Integrate. Let O_i be the color of cell *i* computed from the current frame. There is also an uncertainty, Q_i, associated with O_i. In our current system, it can only be one of two values: ∞ if the cell color is an outlier, 4 otherwise (i.e., the standard deviation is equal to 2 intensity levels). Considering possible lighting variation during the time elapsed since the last frame, the uncertainty P_i is first increased by Δ (4 in our system, equivalent to a standard

deviation of 2). C_i and P_i are then updated according to the classic Kalman filter formula:

$$K = \frac{P_i}{P_i + Q_i}$$
$$C_i = C_i + K \cdot (O_i - C_i)$$
$$P_i = (1 - K) \cdot P_i$$

Propagate. In order to fill the holes created by the cells that are occluded by foreground objects and to ensure the color model is smooth, the cell colors are propagated to the neighboring cells. For each cell *i*, it incorporates the 4 of its neighbors' states according to the following:

$$C_i = \frac{C_i P_i^{-1} + \frac{1}{16}\sum_j C_j (P_j + \Delta)^{-1}}{P_i^{-1} + \frac{1}{16}\sum_j (P_j + \Delta)^{-1}}$$

$$P_i = (P_i^{-1} + \frac{1}{16}\sum_j (P_j + \Delta)^{-1})^{-1}$$

Note that we increase the uncertainty of its neighbors by Δ (4 in our system) to allow color variation. A hole of size N generally takes N/2 frames to get filled. Since the uncertainty in the cells with filled values is much larger than the ones with the observed values (due to added Δ), the filled values are quickly supplanted by the observed values once they become available. An example of an integrated whiteboard color is the 3rd image in Figure 4. Note that the bookshelf area in the left side of the image is never filled.

4.1.5 Classifying Cells

This step is to determine whether a cell image is a foreground object or the whiteboard. We perform this in two levels: individual and neighborhood.

At the individual cell level, given a good whiteboard color model, we simply compute the Euclidean distance between the background color of the cell image (computed in Section 4.1.3) and the color of the corresponding cell location in whiteboard background model. If the difference exceeds a threshold (4 brightness levels in our system), the cell image is classified as foreground object.

However, more accurate results can be achieved by utilizing spatial relationship among the cell groups. The basic observation is that foreground cells should not appear isolated spatially since a person usually blocks a continuous region of the whiteboard. So at the neighborhood level, we perform two filtering operations on every frame. First, we identify isolated foreground cells and reclassify them as whiteboard. This operation corrects the mis-classification of the cells that are entirely filled with strokes. Second, we reclassify whiteboard 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. It will be correctly re-classified as whiteboard once the foreground object moves away. Extending the foreground object boundary delays the recognition of strokes by a few frames, but it prevents some parts of the foreground object from being classified as strokes -- a far worse situation.

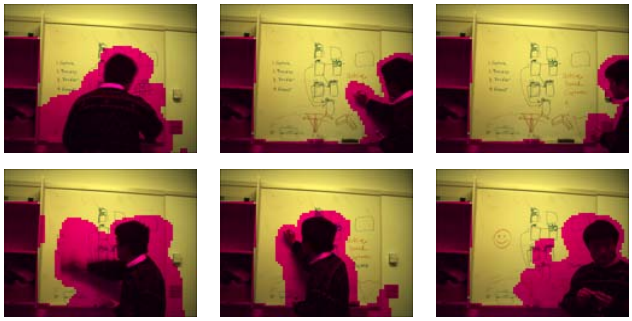

Figure 5: Samples of the classification results. The images above correspond to the images in Figure 2. The classified foreground region is tinted in purple and is larger than the actual foreground object due to motion, shadow, and spatial filtering.

4.1.6 Extracting New Strokes

The cells classified as foreground are not further processed. For cells classified as whiteboard, we check whether there is a whiteboard cell already existing in the same cell location in the output depository. If not, the cell is a new whiteboard cell. If a whiteboard cell does exist, we still need to check whether the existing cell and the current cell image are the same, using the same image difference algorithm in Section 4.1.2. If they are different, the user probably has erased the whiteboard and/or written something new, and therefore the whiteboard cell in the output depository is replaced by the current cell image. Periodically (every 30 seconds), we update all existing whiteboard cells with current cell images to account for possible lighting variations.

4.1.7 Color-enhancing the Stroke Images

At this stage, the newly extracted cell images are finally converted from raw Bayer images into RGB images. The demosaicing algorithm used by the Aplux camera driver is not documented. Instead of the built-in one, we use a demosaicing algorithm proposed in [**Error! Reference source not found.**] which handles edges much better.

After demosaicing, the images still look color-shifted and noisy. They need to be white-balanced and color-enhanced, which also helps in

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 4.1.3, $\bar{I}_w$, 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.

Figure 2 is an example of such color enhancement.

4.2 Teleconferencing Experience

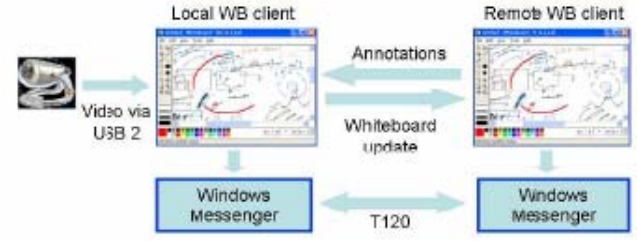

Figure 6: Real-time whiteboard system inside the Windows Messenger.

To test our system in a real teleconference setting, we adapted our system to be a plug-in to the Whiteboard applet of the Microsoft Windows Messenger. The Whiteboard applet allows the users at two ends of a Windows Messenger session to share a digital whiteboard. The user at one end can paste images or draw geometric shapes and the user at the other end can see the same change almost instantaneously. Usually, the user draws objects with his mouse, which is very cumbersome. With our system, the user can write on a real whiteboard instead.

Our system takes 45-50% processing power of a 2.4G Hz Pentium 4 PC. Once launched, our system initializes in about 5 seconds, which include the time to do exposure calibration, initialize the whiteboard color model, and capture the content already existing on the whiteboard.

The changes to the whiteboard content are automatically detected by our system and incrementally piped to the Whiteboard applet as small cell image blocks. The Whiteboard applet is responsible for compressing and synchronizing the digital whiteboard content shared with the remote meeting participant. The remote participant can add annotations on top of the whiteboard image using the mouse. When used with other Windows Messenger tools, such as voice conferencing and application sharing, whiteboard sharing becomes a very useful tool in communicating ideas.

The time delay between the appearance of the stroke in input video and showing up on the local Whiteboard applet is 1 second. Network transport takes additional 0.5 second or more depending on the distance between the two ends.

Because the resulting image contains only uniform background and a handful of colors, the required communication bandwidth after compression is proportion to the amount of content that the user produces. Using GIF compression, a reasonably full whiteboard image at 1.3 MP takes about 200K bytes (Figure 3 takes 70K bytes). After the initial image capture, the whiteboard updates take 50-100 bytes per cell. Since usually only a handful of cells are changing at a time when the whiteboard is in use, the sustained network bandwidth requirement is far below those of video conferencing solutions – suitable even for use in a dial-up network.

The reader is recommended to watch the accompanying video that captures the working of our RTWCS. On the left is the interface which allows us to choose which camera, if there are several, and which demosaicing algorithm (we have implemented several) to use. It also shows a down-sampled live video. On the right is the digital whiteboard application shared with the remote participant, i.e., the remote participant sees exactly what is shown on this window. As can be observed, the person in front of the whiteboard is filtered out. The new strokes show up in the digital whiteboard with very short delay (about 1 second). Note that the order of new strokes appearing in the digital whiteboard is not necessary the same as that of the strokes when they were written. The former depends on when the camera sees the strokes. Because of white-balancing and color enhancement, the quality of the whiteboard contents that the remote participant sees is obviously much better than that of the video.

4.3 Capturing Printed Documents and Annotations

We also tried to use our system to capture handwriting annotations on printed documents in real time – a common scenario in teleconferencing when participants need to share paper documents. We built a gooseneck support so the camera can be pointed downward securely. When capturing 8.5”x11” sized documents, we found that the document image is legible down to 6pt fonts.

The software works in general without modification, except one problem. We discovered that when annotating on paper documents, the user tends to make small movements to the paper. This behavior causes reset to be triggered quite often. Each of such reset causes an additional 5 seconds delay for the annotation to appear. To overcome this problem, we added an efficient homography-based image matching algorithm to align each input video frame to first frame – equivalent to motion stabilizing the input video. This modification removes most of the resets and makes the system much more usable.

This functionality is also demonstrated in the accompanying video.

5 LIMITATIONS OF OUR 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 will obviously not be able to figure out the content in that region. This situation actually happens quite often even for local meeting participants. In that case, the users, remote or local, can ask the presenter to move away temporarily when their views to the whiteboard are blocked. (Remote participants have an advantage over the local ones when the presenter occludes a region already seen by the camera. They can see the contents behind the presenter in their digital whiteboard.)

6 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 Real Time Whiteboard Capture System, allows the users to share ideas on a whiteboard in a variety of teleconference scenarios: brainstorming sessions, lectures, project planning meetings, patent disclosures, etc.

Comparing to video conferencing solutions, our system takes only a fraction of its bandwidth and is suitable even on dial-up networks. Comparing to other whiteboard capture technologies, the users of our system can write naturally using regular board and pens.

With devices like electronic whiteboards and Tablet PCs, the users are promised the ability to write freely in an all electronic medium. But as the cost of those devices is still quite high, we believe that the combination of the omnipresent whiteboard and a low-cost video camera will be a very promising solution for the foreseeable future.

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, or Y). The color is computed as described in Section 4.1.3, and is therefore noisy and even erroneous. From our experience with the meeting rooms in our company, the color on the whiteboard, although not constant, 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 $\{(x_i, y_i, z_i) \mid i=1,...,n\}$ 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 $\mathbf{A}=\begin{bmatrix} x_1 & y_1 & 1 \\ \cdots & \cdots & \cdots \\ x_n & y_n & 1 \end{bmatrix}$ and $\mathbf{z}=[z_1,\ldots,z_n]^T$. 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 [8], 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., $\min_{\mathbf{p}} \text{median}_i f_i^2$. 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 over the whole set of data points. 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. 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.
3. Duda, R.O., Hart, P.E. and Stork, D.G. *Pattern Classification*, Second Edition, John Wiley & Sons, New York, 2001.
4. 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.*
5. Malvar, Henrique S., He, L. and Cutler, R. High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images. *ICASSP 2004*.
6. Moran, T. P., Palen, L., Harrison, S., Chiu, , P., Kimber, D., Minneman, 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.
7. Pedersen, E., McCall, K., Moran, T. P., & Halasz, F., Tivoli: An electronic whiteboard for informal workgroup meetings. Proceedings of INTERCHI'93. pp391-389.
8. Rousseeuw, P. and Leroy, A. *Robust Regression and Outlier Detection*, John Wiley & Sons, New York, 1987.
9. Stifelman, L.J., Arons, B., Schmandt, C. & Hulteen, E.A. VoiceNotes: A Speech Interface for a Hand-Held Voice Notetaker. *Proc. INTERCHI'93 (Amsterdam, 1993),* ACM

10. Saund, E. Image Mosaicing and a Diagrammatic User Interface for an Office Whiteboard Scanner. Technical Report, Xerox Palo Alto Research Center, 1999.
11. Weber, K. & Poon, A., Marquee: A tool for real-time video logging. Proceedings of CHI'94. pp 58-64.
12. Wilcox, L. D., Schilit, B. N. & Sawhney, N., Dynomite: A dynamically organized ink and audio notebook. Proceedings of CHI'97. pp 186-193.
13. Whittaker, S., Hyland, P. & Wiley, M., Filochat: Handwritten notes provide access to recorded conversations. Proceedings of CHI'94. pp 271-276.
14. Wolf, C., Rhyne, J. & Briggs, L., Communication and information retrieval with a pen-based meeting support tool. Proceedings of CSCW'92. pp 322-329.
15. He, Li-wei, Liu, Z. and Zhang, Z. Why Take Notes? Use the Whiteboard Capture System. *ICASSP 2003*.