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
Email: {lhe, zhang}@microsoft.com

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.

1. INTRODUCTION

A whiteboard is an effective and easy to use tool for meetings, especially in scenarios such as brainstorming, lectures, project planning, and patent disclosures. Sometimes, 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.

The system presented in this paper 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 the system directly takes images of the whiteboard, 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).

The paper is organized as follows: Section 2 discusses related works and the design choices that we made. Section 3 explains the technical challenges we encountered while building the system. Section 4 describes how the developed real-time whiteboard system is integrated in the Windows Messenger for remote collaboration on a physical whiteboard. Section 5 concludes the paper.

2. SYSTEM DESIGN

The system described in this paper is a real-time extension to the Whiteboard Capture System (WCS) that we developed two years ago [1]. In WCS, we take pictures of the whiteboard continuously using a Canon G2 digital still image camera. 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 browse the audio meeting recording in a very efficient way.

2.1 Capture Device

Since building the WCS, 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 [1].

2.2 Capture Interface Requirements

Like our previous WCS, our current 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 as long as it 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 3.3). This improvement made our system much more portable and easier to use than our previous 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. TECHNICAL DETAILS

The input to our real-time system is a sequence of high-resolution video images (see Figure 1). 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.

Our system encounters 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 un-occluded. We need to deal with these problems in order to extract the new pen strokes.

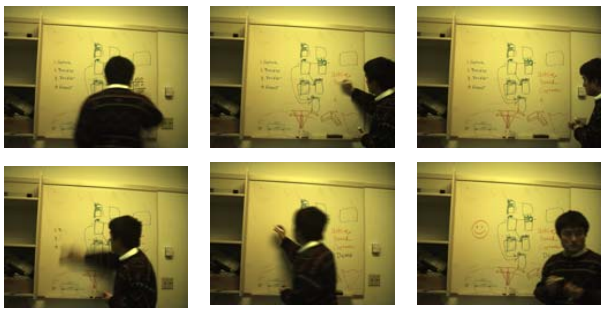

Figure 1: Selected frames from an input image sequence. The sequence lasts 82 seconds.

In order to segment the images into foreground objects and whiteboard, we rely 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.

3.1 Strategies for Real-Time Analysis

We apply several strategies to make the algorithm efficient.

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.

Second, our analyzer is structured as a pipeline of six analysis procedures. 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: determines if the cell images have changed since last frame.
2. Color estimator: 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 or cast shadow by foreground objects.
4. Cell classifier: classifies the cell images into foreground or whiteboard cells.
5. Stroke extractor: extracts the newly appeared strokes.
6. Color enhancer: enhances the color of 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 [2]. 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.

3.2 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 3.5 for details).

3.3 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 the Normalized Cross-Correlation (NCC) algorithm. Note that the NCC is applied to the images in the Bayer space.

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 that case, all state is re-initialized. Other more localized changes (e.g. people moving across, gradual change in sun light) are handled dynamically by the Whiteboard Color Model Update Module.

3.4 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 3.8), 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, we average the colors of the pixels in the top 10th percentile in order to reduce the error introduced by sensor noise.

3.5 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. However, it will give the wrong color when the cells contain only the foreground or pen strokes (the image on the left in Figure 2). We have to identify those cells to prevent them from contaminating the whiteboard color model.

Figure 2: Left: Cell colors. Note that the strokes on the whiteboard are removed by our background color estimation algorithm. Center: Colors of cells that go into the Update Module. Note the black regions are cells filtered out by both the change detector and the color estimator. Right: Integrated whiteboard color model.

We use a least-median-squares algorithm [3], which fits a global plane over the colors and throws away the cells that contain outlier colors. The least-median-squares algorithm uses the median of the errors for selecting the best plane candidate instead of a user-defined threshold. The remaining cells are considered as background cells and their colors are used to update the whiteboard background (the center image in Figure 2).

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:

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

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:

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

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 image on the right in Figure 2. Note that the bookshelf area in the left side of the image is never filled. Although the wall is also classified as whiteboard region, it usually does not affect the overall extraction process.

3.6 Classifying Cells

To determine whether a cell image is foreground or whiteboard, we carry out in two levels: individual and neighborhood.

At the individual cell level, given a good whiteboard color model, we compute the Mahalanobis distance between the background color of the cell image (computed in Section 3.4) and the color of the corresponding cell location in whiteboard background model. If the difference exceeds a threshold, the cell image is classified as foreground object.

This result is then refined 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 3 shows the classification results of a typical input sequence.

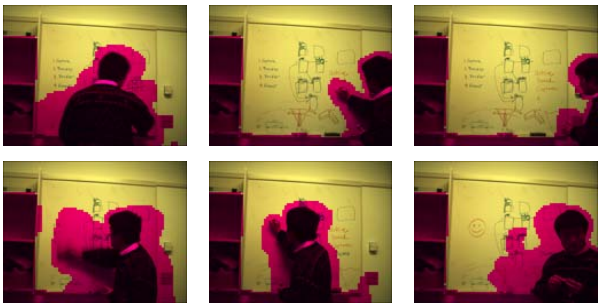

Figure 3: Samples of the classification results for the images in Figure 1. The classified foreground region is tinted in purple and is larger than the actual foreground object due to motion, shadow, and spatial filtering.

3.7 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 3.3 . 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.

3.8 Color-enhancing the Stroke Images

At this stage, the newly extracted cell images are finally converted from raw Bayer images into RGB images. Instead of the built-in demosaicing algorithm in the camera driver, we use a demosaicing algorithm proposed in [2], which handles edges much better.

After demosaicing, the images still look color-shifted and noisy. We use the same procedure used in our previous WCS to white-balance and color-enhance the resulting images [1]. The resulting images contain only uniformly white background, which makes the images print and compress much better.

4. TELECONFERENCING EXPERIENCE

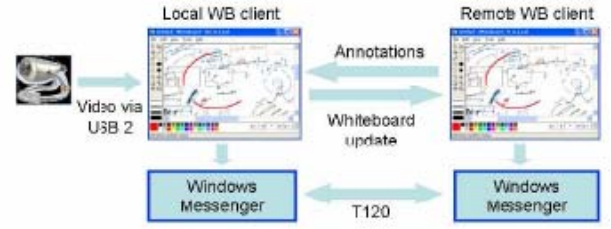

Figure 4: Real-time whiteboard system inside the Windows Messenger.

To test our system in a real teleconference setting, we adapted our system to be a plug-in to the Whiteboard applet of the Windows Messenger (see Figure 5). 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. The changes to the whiteboard content are automatically detected by our system and incrementally pasted 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.

Because the resulting image contains only uniform background and a handful of colors, the required communication bandwidth after compression is proportional 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. 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.

A video that captures our system in action will be shown during the conference presentation and is included in the CD-ROM proceedings. On the left is 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). 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.

5. CONCLUDING REMARKS

The system presented in this paper allows the users to share ideas on a whiteboard in a variety of teleconference scenarios. 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 new 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 high-resolution video camera will be a very promising solution for the foreseeable future.

REFERENCES

[1] He, Li-wei, Liu, Z. and Zhang, Z. Why Take Notes? Use the Whiteboard Capture System. *ICASSP 2003*.

[2] Malvar, Henrique S., He, L. and Cutler, R. High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images. *ICASSP 2004*.

[3] Rousseeuw, P. and Leroy, A. *Robust Regression and Outlier Detection*, John Wiley & Sons, New York, 1987.

[4] Saund, E. Image Mosaicing and a Diagrammatic User Interface for an Office Whiteboard Scanner. Technical Report, Xerox Palo Alto Research Center, 1999.