Whiteboard Scanning: Get a High-res Scan With an Inexpensive Camera

Zhengyou Zhang   Li-wei He
Microsoft Research
Email: zhang@microsoft.com, lhe@microsoft.com

Aug. 12, 2002

Abstract

A whiteboard provides a large shared space for collaboration. In order to capture the contents with readability, a high-res camera is required. If one only has an inexpensive camera such as a laptop/tablet built-in camera or a web cam, she can use the *Whiteboard Scanning* system described in this paper. It captures either a set of snapshots with overlap or a continuous video sequence, and then stitches them automatically into a single high-res image. The stitched image can finally be exported to our *Whiteboard It!* system for further enhancement.

1 Introduction

Imagine that you had a fruitful brainstorming session with all the nice drawings on the whiteboard, and you have to copy them in your laptop but you only have a built-in camera with maximum resolution 640$\times$480 that is not high enough to produce a readable image of the whiteboard. Imagine again that you are in a distributed meeting and you have a document only in paper form to share with other remote meeting participants but you only have a web cam with maximum resolution 640$\times$480 that is not high enough to produce a readable image of the paper document. This paper describes a system that produces a high-res image of a whiteboard or a paper document with a low-res camera by stitching multiple images together.

2 Overview

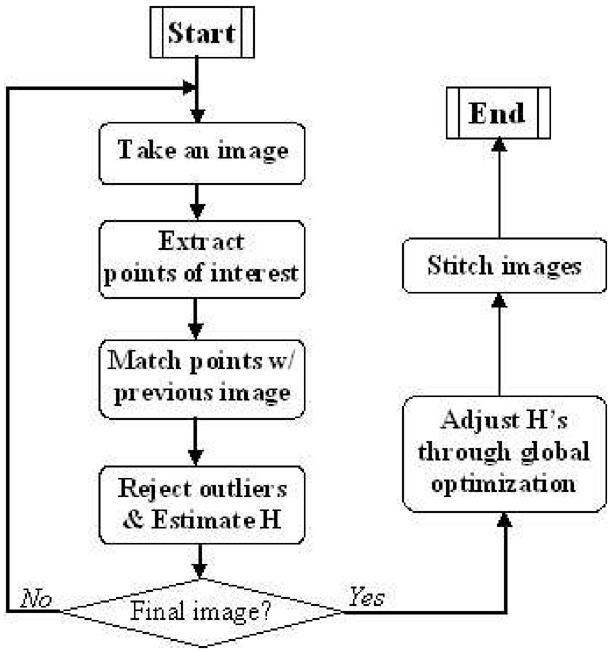
The major steps of the Whiteboard Scanning system is illustrated in Figure 1, and will be explained below.

The system can run in two modes: snapshot or continuous. Although the image acquisition procedure differs for the two operation modes, the stitching process is essentially the same.

In *snapshot* mode, we start taking a snapshot from the upper left corner, a second by pointing to the right but having overlap with previous snapshot, and so on until reaching the upper right corner; move the camera lower and take a snapshot, then take another one by pointing to the left, and so on until reaching the left edge; the process continues in the "S" way until the lower border is captured. Successive snapshots must have overlap to allow later stitching, and this is assisted by providing visual feedback during acquisition. Figure 2a illustrates this image acquisition process.

In *continuous* mode, the user takes images also starting from the upper left corner but continuously following the same "S" way as illustrated in Figure 2b. The difference from the snapshot mode is that the user does not need anymore to wait and position the camera before taking a snapshot. The continuous image acquisition guarantees overlap between successive images. However, motion blur may cause the final stitched image look not as crisp as those obtained with snapshot mode. In order to reduce the blur, the camera exposure time should be set to a small value.

Currently, we have only implemented the snapshot mode. The user interface is shown in Figure 3. We suggest having half image to overlap between successive images. In the viewing region, we show both



Start
Take an image
Extract points of interest
Match points w/ previous image
Reject outliers & Estimate H
Final image?
No
Yes
Adjust H's through global optimization
Stitch images
End

Figure 1: Diagram of the Whiteboard Scanning system

the previously acquired image and the current video view. In order to facilitate the image acquisition, half of the previously acquired image is shown in opaque, while the other half, which is in the overlapping region, is shown in semi-transparent. The current live video is also shown in half opaque and half semi-transparent. This guides the user to take successive images with overlap. Note that the alignment does not need to be precise. Our program will automatically align them. There are also a few buttons to indicate the direction in which the user wants to move the camera (down, up, left, right). The overlapping region changes depending on the direction. We have designed the default behavior such that only the ”down” button is necessary to realize image acquisition in the ”S” way.

The stitching process works very much in a similar way in both operation mode, and is illustrated in Figure 1.

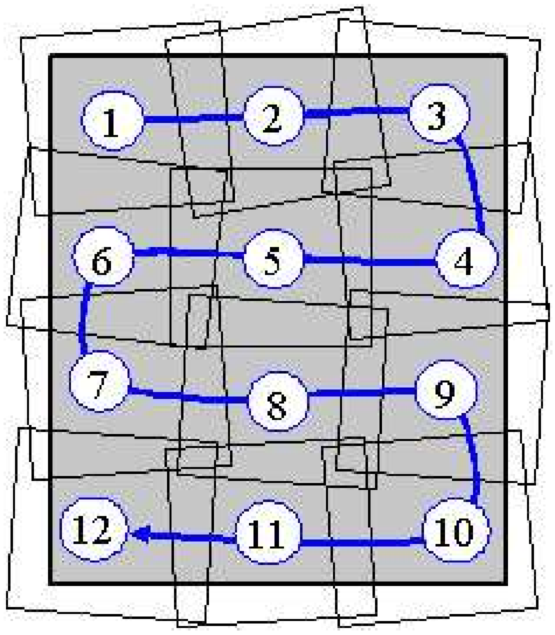
The mathematic foundation is that two images of a *planar* object, regardless the angle and position of the camera, are related by a plane perspectivity, represented by a 3×3 matrix called *homography* $\mathbf{H}$. More precisely, let $\mathbf{m}_1 = [u_1, v_1]^T$ and $\mathbf{m}_2 = [u_2, v_2]^T$ be a pair of corresponding points, and use the notation $\tilde{}$ for $\tilde{\mathbf{m}} = [u, v, 1]^T$, then we have

$$\tilde{\mathbf{m}}_2 = \lambda \mathbf{H} \tilde{\mathbf{m}}_1 \ , \tag{1}$$

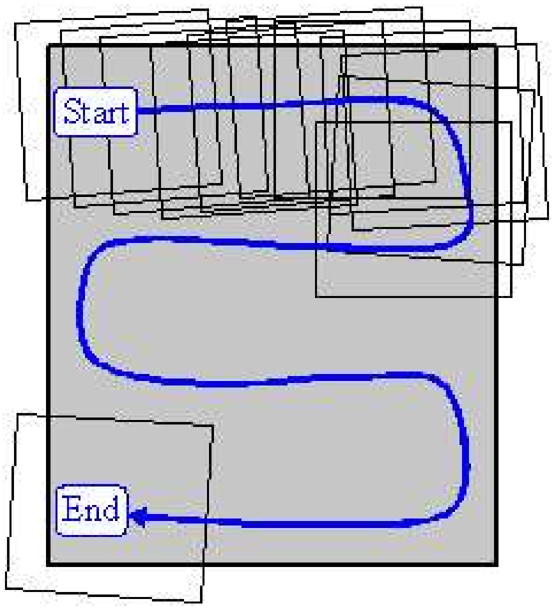
where λ is a scalar factor. That is, $\mathbf{H}$ is defined up to a scalar factor. We therefore need at least 4 pairs of point matches in order to determine homography $\mathbf{H}$.

Referring to Figure 1. For each image acquired, we extract points of interest. We use the Plessey corner detector, a well-known technique in computer vision. It locates corners corresponding to high curvature points in the intensity surface if we view an image as a 3D surface with the third dimension being the intensity. An example is shown in Figure 4, where the extracted points are displayed in red +.

Next, we try to match the extracted points with those from the previous image. For each point in the previous image, we choose an 15×15 window centered on it, and compare the window with windows of the same size, centered on the points in the current image. A zero-mean normalized cross correlation between two windows is computed. If we rearrange the pixels in each window as a vector, the correlation score is equivalent to the cosine angle between two intensity vectors. It ranges from -1, for two windows which are not similar at all, to 1, for two windows which are identical. If the largest correlation score exceeds a prefixed threshold (0.707 in our case), then that point in the current image is considered to be the *match candidate* of the point in the previous image. The match candidate is retained as a *match* if and only



(a) Snapshot Mode



(b) Continuous Mode

Figure 2: Illustration of two image acquisition modes

if its match candidate in the previous image happens to be *the point being considered*. This symmetric test reduces many potential matching errors.

The set of matches established by correlation usually contains false matches because correlation is only a heuristic and only uses local information. Inaccurate location of extracted points because of intensity variation or lack of strong texture features is another source of error. The geometric constraint between two images is the homography constraint (1). If two points are correctly matched, they must satisfy this constraint, which is unknown in our case. If we estimate the homography between the two images based on a least-squares criterion, the result could be completely wrong even if there is only one false match. This is because least-squares is not robust to outliers. We developed a technique based on a robust estimation technique known as the *least median squares* to detect both false matches and poorly located corners, and simultaneously estimate the homography matrix $\mathbf{H}$. More precisely, let $\{(\mathbf{m}_{1i}, \mathbf{m}_{2i})\}$ be the pairs of points between two images matched by correlation, the homography matrix $\mathbf{H}$ is estimated by solving the following nonlinear problem:

$$\min_{\mathbf{H}} \operatorname*{median}_i \|\mathbf{m}_{2i} - \widehat{\mathbf{m}}_{1i}\|^2 , \tag{2}$$

where $\widehat{\mathbf{m}}_{1i}$ is the point $\mathbf{m}_{1i}$ transferred to the current image by $\mathbf{H}$, i.e., $\tilde{\widehat{\mathbf{m}}}_{1i} = \lambda_i \mathbf{H} \tilde{\mathbf{m}}_{1i}$. The optimization is performed by searching through random sampling in the parameter space to find the parameters yielding the smallest value for the *median* of squared residuals computed for the entire data set. From the smallest median residual, we can compute a so-called robust standard deviation $\hat{\sigma}$, and any point match yielding a residual larger than, say, $2.5\hat{\sigma}$ is considered to be an outlier and is discarded. Consequently, it is able to detect false matches as many as 49.9% of the whole set of matches. More concretely, this outlier rejection procedure is implemented as follows:

1. Draw m random subsamples of $p = 4$ different point matches. (We need at least 4 point matches to determine a homography matrix.)

2. For each subsample J, compute the homography matrix $\mathbf{H}_J$ according to (1).

3. For each $\mathbf{H}_J$, determine the median of the squared residuals, denoted by M_J, with respect to the whole set of point matches. The squared residual for match i is given by $\|\mathbf{m}_{2i} - \widehat{\mathbf{m}}_{1i}\|^2$ where $\widehat{\mathbf{m}}_{1i}$ is point $\mathbf{m}_{1i}$ transferred to the second image by $\mathbf{H}_J$.

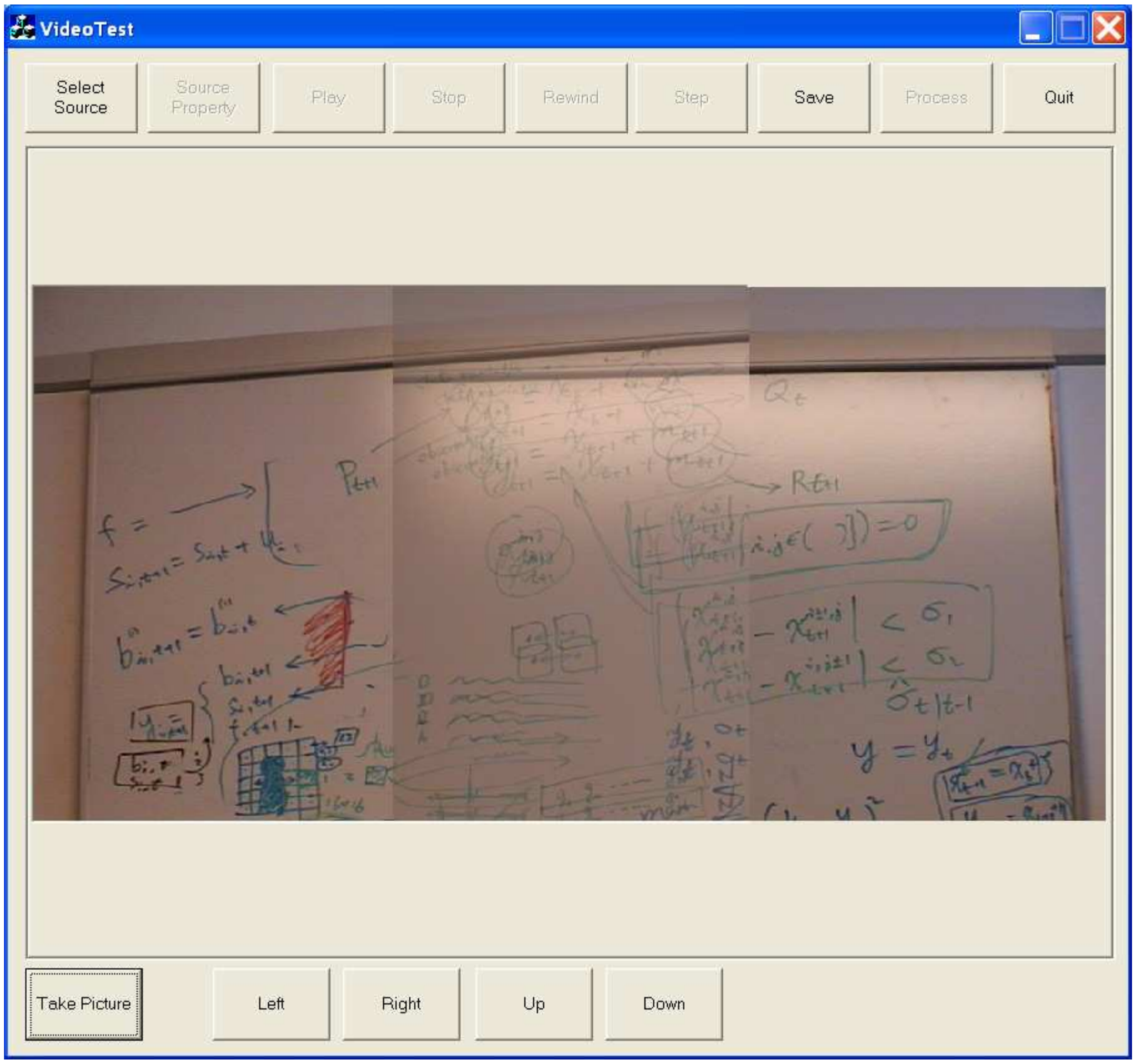


Figure 3: User interface for whiteboard scanning

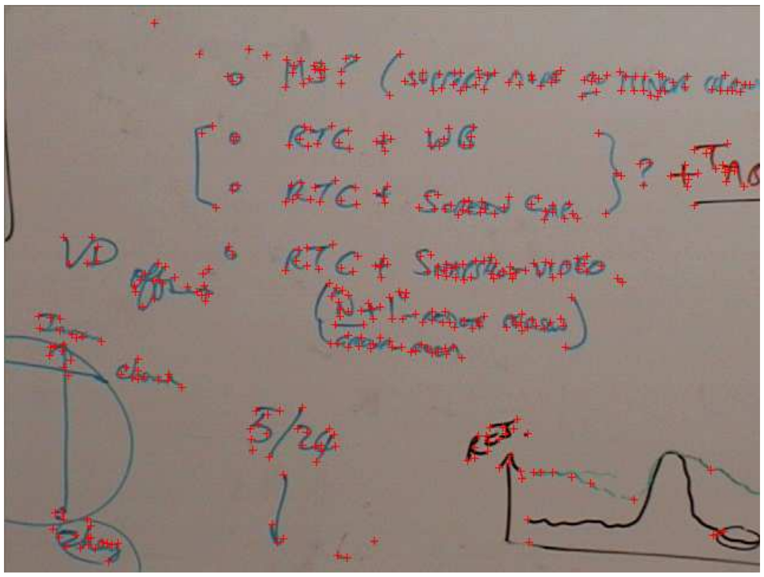


Figure 4: Example of extracted points of interest, shown in +.

4. Retain the estimate $\mathbf{H}_J$ for which M_J is minimal among all m M_J's.
5. Compute the robust standard deviation estimate: $\hat{\sigma} = 1.4826[1 + 5/(n-p)]\sqrt{M_J}$.
6. Declare a point match as a false match if its residual is larger than $k\hat{\sigma}$, where k is set to 2.5.
7. Discard the false matches and re-estimate $\mathbf{H}$ by minimizing the sum of squared errors $\sum_l \|\mathbf{m}_{2l} - \widehat{\mathbf{m}}_{1l}\|^2$ where the summation is over all good matches.

In our implementation, $m = 70$, which gives a probability of 99% that one of the 70 subsamples is good (i.e., all four point matches in the subsample are good) even if half of the total point matches are bad. The last step improves the accuracy of the estimated homography matrix because it uses all good matches.

This incremental matching procedure stops when all images have been processed. Because of incremental nature, cumulative errors are unavoidable. For higher accuracy, we need to adjust $\mathbf{H}$'s through global optimization by considering all the images simultaneously. This is done as follows. Let us assume we have in total N images. Without loss of generality, we choose the first image as the reference image for our global optimization. Let the homography matrix from the reference image to image i be $\mathbf{H}_i$, with $\mathbf{H}_1 = \mathbf{I}$. There are M distinct points in the reference image, which we call *reference points*, denoted by $\hat{\mathbf{m}}_j$. Because of our matching process, a reference point is observed at least in two images. For example, a point in the first image can be matched to a point in the second image, which in turn is matched to a point in the third image; this happens if the first three images shares a common region. Even if a physical point in space is observed in three or more images, we only use one single reference point to represent it. We introduce one additional symbol ϕ_{ij}:

$$\phi_{ij} = \begin{cases} 1 & \text{if point } j \text{ is observed in image } i \\ 0 & \text{otherwise} \end{cases}$$

We can now formulate the global optimization as estimation of both homography matrices $\mathbf{H}_i$'s and reference points $\hat{\mathbf{m}}_j$'s by minimizing the errors between the expected positions and the observed ones in the images, i.e.,

$$\min_{\{\mathbf{H}_i\},\{\hat{\mathbf{m}}_j\}} \sum_{i=1}^{N} \sum_{j=1}^{M} \phi_{ij} \|\mathbf{m}_{ij} - \widehat{\mathbf{m}}_{ij}\|^2 , \tag{3}$$

where

$$\widetilde{\widehat{\mathbf{m}}}_{ij} = \lambda_{ij} \mathbf{H}_i \widetilde{\widehat{\mathbf{m}}}_j .$$

Once the geometric relationship between images (in terms of homography matrices $\mathbf{H}$'s) are determined, we are able to stitch all images as a single high-res image. There are several options, and currently we have implemented a very simple one. We use the first image as the reference frame of the final high-res image, and map successively original images to the reference frame. If a pixel in the reference frame appears several times in the original images, then the one in the newest image is retained.

3 Examples

In this section, we show a few examples. Figures 5 and 6 show the stitching result of six images of a whiteboard. Figures 7 and 8 show the stitching result of a paper document. The stitched paper document is compared in Fig. 8 with a single image of the document, and clearly the stitched image gives a much higher readability.

4 Technical Challenges

This project has been built on top of extensive experience we gained in the past. There are still a number of interesting research issues we should solve:

- Nonlinear camera distortion. The stitching algorithm assumes that the camera undergoes a linear perspective projection, i.e., a line in 3D space is projected as a line in the image. However, an inexpensive camera usually exhibits a strong distortion, especially around the border of the image. We have to correct the distortion either during the stitching process or as a pre-processing through calibration.

- End-user experiences. We have proposed two ways of image acquisition, and they should undergo a usability study. We should also explore other options in order for an end-user to use this system painlessly.

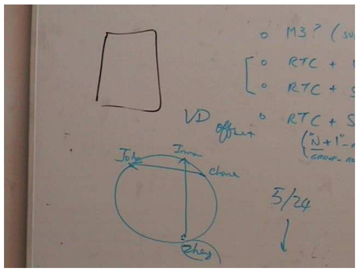


image 1

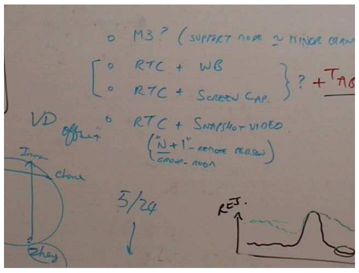


image 2

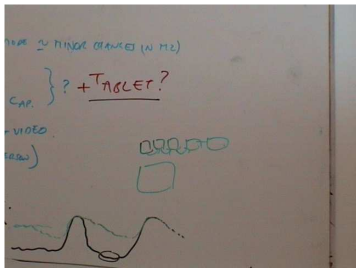


image 3

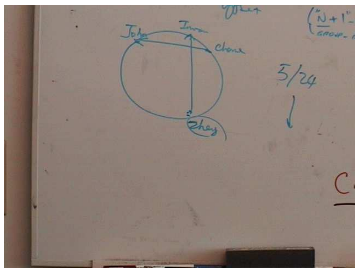


image 6

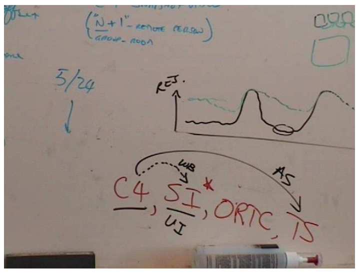


image 5

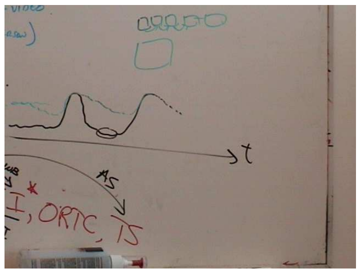


image 4

Figure 5: Six images of the whiteboard in Zhengyou's office.

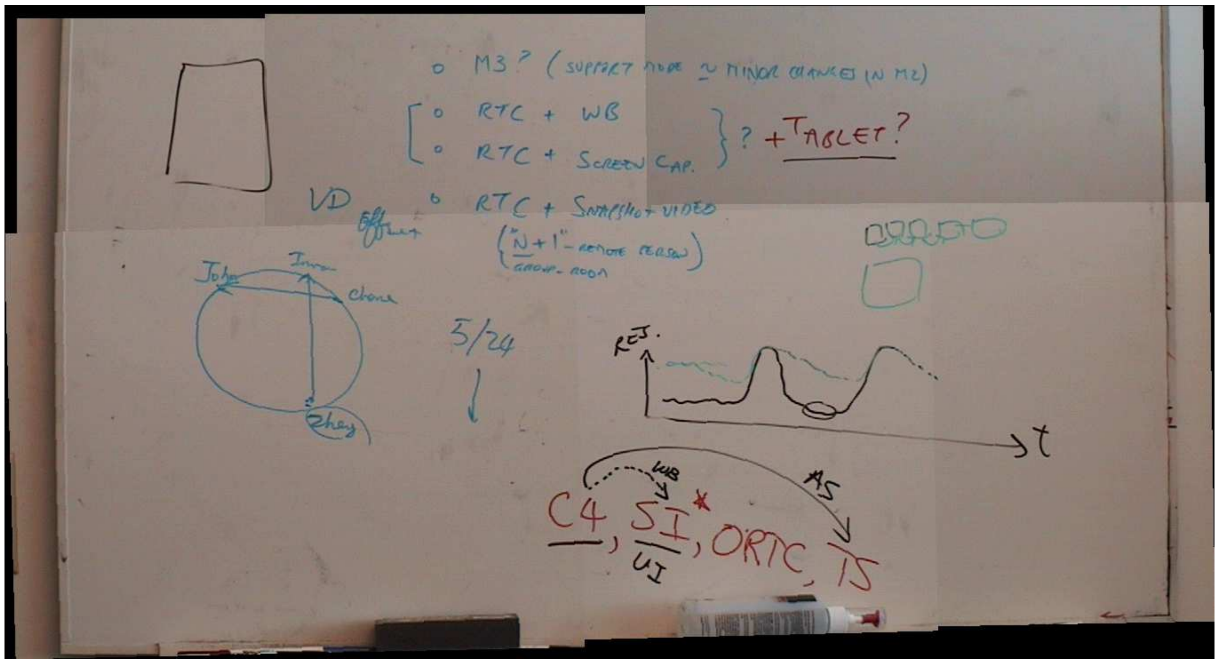
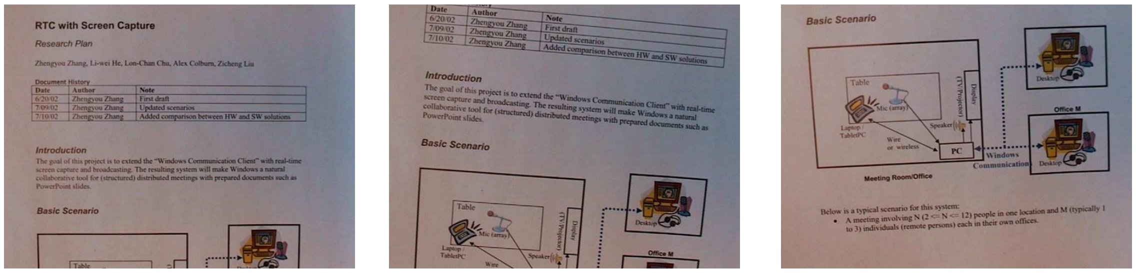


Figure 6: Stitched image from those shown in Fig. 5. The cumulative error is visible at the left border.



RTC with Screen Capture

Research Plan

Zhengyou Zhang, Li-wei He, Lon-Chan Chu, Alex Colburn, Zicheng Liu

Document History

Date	Author	Note
6/20/02	Zhengyou Zhang	First draft
7/09/02	Zhengyou Zhang	Updated scenarios
7/10/02	Zhengyou Zhang	Added comparison between HW and SW solutions

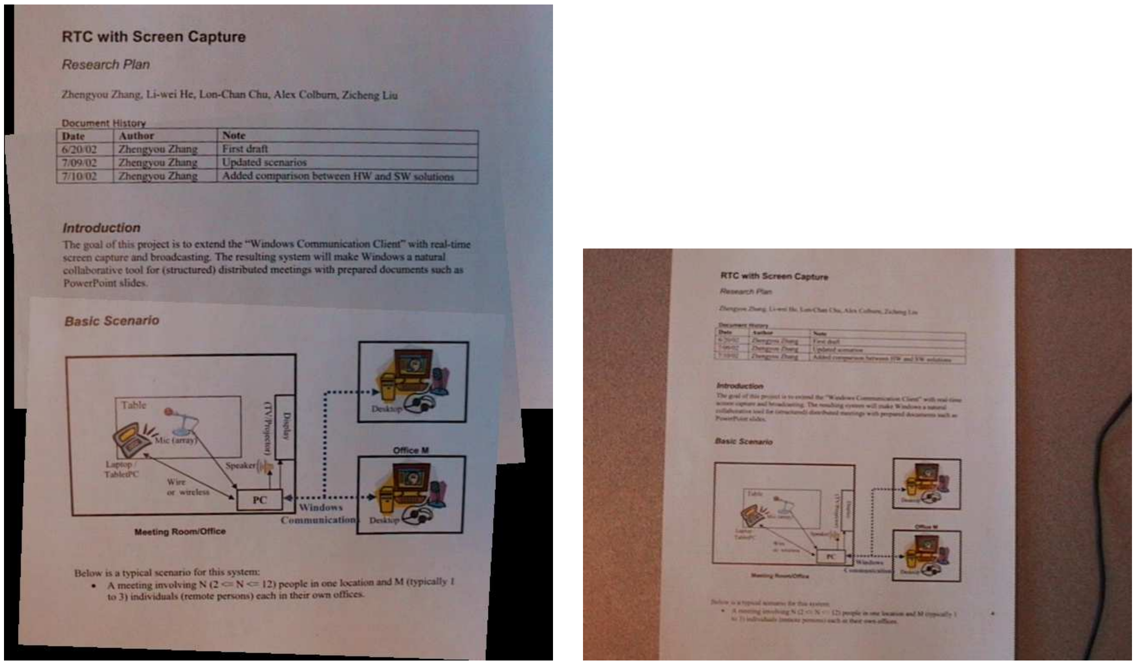
Introduction

The goal of this project is to extend the "Windows Communication Client" with real-time screen capture and broadcasting. The resulting system will make Windows a natural collaborative tool for (structured) distributed meetings with prepared documents such as PowerPoint slides.

Basic Scenario

Below is a typical scenario for this system:
- A meeting involving N (2 <= N <= 12) people in one location and M (typically 1 to 3) individuals (remote persons) each in their own offices.

Figure 7: Three images of a paper document



RTC with Screen Capture

Research Plan

Zhengyou Zhang, Li-wei He, Lon-Chan Chu, Alex Colburn, Zicheng Liu

Document History

Date	Author	Note
6/20/02	Zhengyou Zhang	First draft
7/09/02	Zhengyou Zhang	Updated scenarios
7/10/02	Zhengyou Zhang	Added comparison between HW and SW solutions

Introduction

The goal of this project is to extend the "Windows Communication Client" with real-time screen capture and broadcasting. The resulting system will make Windows a natural collaborative tool for (structured) distributed meetings with prepared documents such as PowerPoint slides.

Basic Scenario

Below is a typical scenario for this system:
- A meeting involving N (2 <= N <= 12) people in one location and M (typically 1 to 3) individuals (remote persons) each in their own offices.

(a) (b)

Figure 8: Comparison: (a) Stitched image from those shown in Fig. 7; (b) The same document captured by the same camera as a single image.