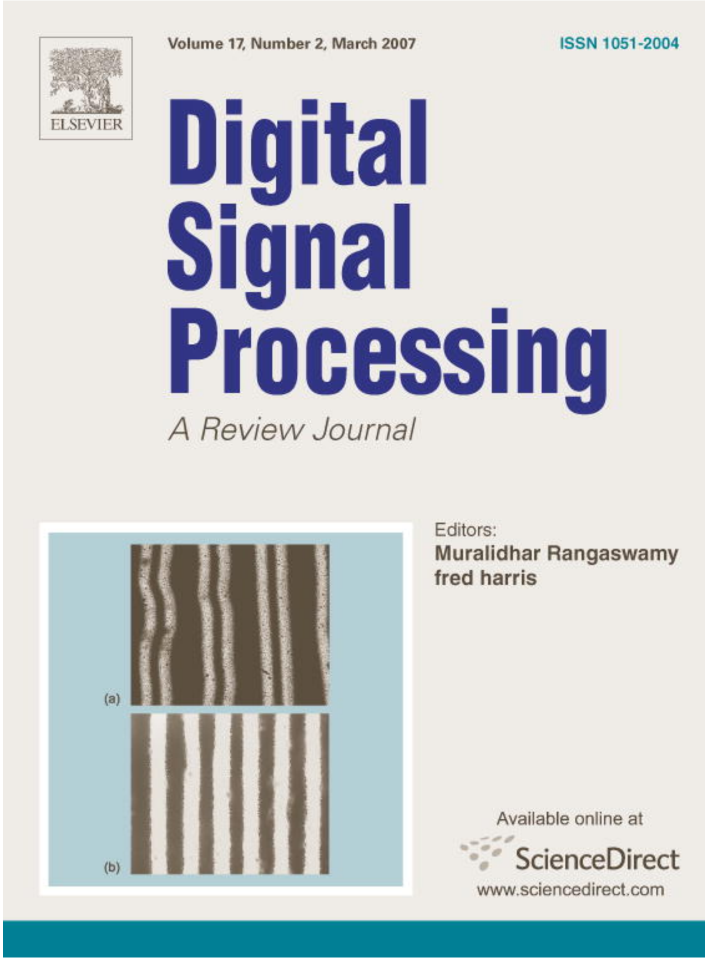


Volume 17, Number 2, March 2007 ISSN 1051-2004

Digital Signal Processing

A Review Journal

Editors:
Muralidhar Rangaswamy
fred harris

Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com
ScienceDirect

Digital Signal Processing 17 (2007) 414–432

Digital Signal Processing

www.elsevier.com/locate/dsp

Whiteboard scanning and image enhancement [☆]

Zhengyou Zhang [*], Li-Wei He

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399, USA

Available online 15 June 2006

Abstract

A whiteboard can be an easy tool for collaboration such as brainstorming, and is widely used, but the content on a whiteboard is hard to archive and share. While digital cameras can be used to capture whiteboard content, the images are usually taken from an angle, resulting in undesired perspective distortion. They may contain other distracting regions such as walls and shadows. The visual quality of those images is usually poor. This paper describes a system that automatically locates the boundary of a whiteboard, crops out the whiteboard region, rectifies it into a rectangle, and corrects the color to make the whiteboard completely white. In case a single image is not enough (e.g., large whiteboard and low-resolution camera), we have developed a robust feature-based technique to automatically stitch multiple overlapping images. The system has been tested extensively, and very good results have been obtained.

1. Introduction

A whiteboard provides a large shared space for collaborative meetings or lectures. 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 whiteboards are widely used, they are not perfect. The content on the whiteboard is hard to archive or share with others who are not present in the session. 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. If you have another meeting right after, you may not have time to copy the contents; if other people reserve the meeting room and use it right after, the contents on the whiteboard will be erased. Because digital cameras are becoming accessible to average users, more and more people are using digital cameras to take images of whiteboards instead of copying manually, thus significantly increasing the productivity. The system we describe in this paper aims at reproducing the whiteboard content as a faithful, yet enhanced and easily manipulable, electronic document through the use of a digital (still or video) camera.

However, images are usually taken from an angle to avoid highlights created by flash, resulting in undesired perspective distortion. They can also contain other distracting regions such as walls. Our system uses a series of image processing algorithms. It automatically locates the boundary of a whiteboard as long as there is a reasonable contrast

[☆] A short version of this paper, entitled "Notetaking with a Camera: Whiteboard Scanning and Image Enhancement," appears in Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004), Montreal, Quebec, 17–21 May, 2004.

[*] Corresponding author.
E-mail address: zhang@microsoft.com (Z. Zhang).
URL: http://research.microsoft.com/~zhang/.

1051-2004/$ – see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.dsp.2006.05.006

near the edges, crops out the whiteboard region, rectifies it to a rectangle with the estimated aspect ratio, and finally corrects the colors to produce a crisp image.

Besides image enhancement, our system is also able to scan a large whiteboard by stitching multiple images automatically. Imagine that you only have a built-in camera with maximum resolution 640 × 480; this is usually not high enough to produce a readable image of a large whiteboard. Our usability study shows that we need about 25 pixels per inch[1] in order to read whiteboard images with normal writing. Our system provides an intuitive interface to assist a user in taking multiple images of the whiteboard with overlap. It then stitches them automatically to produce a high-resolution image. The stitched image can finally be processed and enhanced as mentioned earlier.

The whiteboard scanning subsystem is similar to the ZombieBoard system developed at Xerox PARC [1]. The difference is that they reply on a pan-tilt video camera while we can use a free-moving (still or video) camera as long as there is an overlap between successive views.

The only commercial product we are aware of is Whiteboard Photo from PolyVision [2]. Compared with our system, it lacks two features:

- Aspect ratio estimation: Whiteboard Photo uses either the original image size or the aspect ratio of the bounding box for the final image, therefore aspect ratio of the final image does not correspond to the actual aspect ratio of the whiteboard.
- Whiteboard scanning: Whiteboard Photo does not have the functionality to scan a large whiteboard and stitch multiple images together.

In ICASSP 2003, we presented a whiteboard capture system for a conference room setup [3]. In that system, a high-resolution digital camera is mounted on the opposite wall of the whiteboard and fixed toward the whiteboard, and a microphone is installed in the middle of the table. Both whiteboard content and audio signals are captured during the meeting. The whiteboard image sequence is post-analyzed, and strokes and keyframes are produced and time-stamped. Therefore the whiteboard content serves as a visual index to efficiently browse the audio meeting. On the other hand, the system presented in this paper is very light-weight. It can be used to archive whiteboard content whenever the user feels necessary.

The paper is organized as follows. Section 2 provides an overview of the system. Section 3 describes some details of the image processing techniques implemented in the system. Section 4 presents the whiteboard scanning subsystem. Extensive experimental results with real data are provided. A shorter version of this article appeared in ICASSP 2004 [4].

2. Overview of the system

Before going further, let us look in Fig. 1. On the top is an original image of a whiteboard taken by a digital camera, and on the bottom is the final image produced automatically by our system. The content on the whiteboard gives a flow chart of our system.

As is clear in the diagram shown in Fig. 1b, the first thing we need to decide is whether it is enough to take a single image of the whiteboard. If the whiteboard is small (e.g., $40'$ by $40'$) and a high-resolution digital camera (e.g., 2 mega pixels) is used, then a single image is usually enough. Otherwise, we need to call the whiteboard scanning subsystem, to be described separately in Section 4, to produce a composite image that has enough resolution for comfortable reading of the whiteboard content. Below, we assume we have an image with enough resolution.

The first step is then to localize the boundaries of the whiteboard in the image. Because of perspective projection, the whiteboard in an image usually appears to be a general quadrangle, rather than a rectangle. The quadrangle is localized by detecting four strong edges satisfying certain criteria. If a whiteboard does not have strong edges, a GUI (graphical user interface) is provided for the user to manually specify the quadrangle.

The second step is image rectification. For that, we first estimate the actual aspect ratio of the whiteboard from the detected quadrangle based on the fact that it is the projection of a rectangle in space (see Appendix A for details). Besides the aspect ratio, we can also estimate the focal length of the camera. From the estimated aspect ratio, and by

[1] 1 inch $(1') \approx 2.54$ cm.

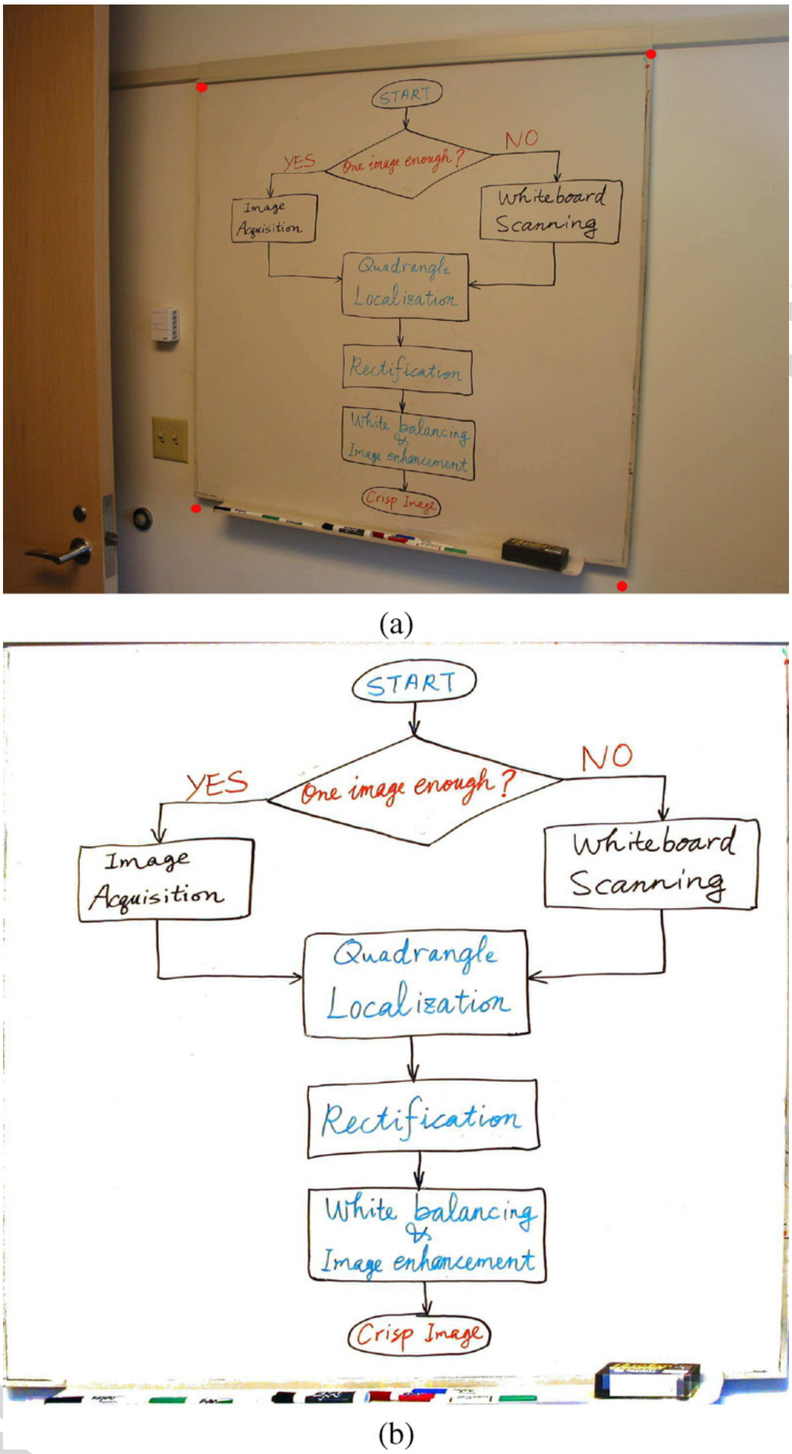


Fig. 1. Diagram of the system architecture drawn on a whiteboard. (a) Original image, (b) processed image.

choosing the "largest" whiteboard pixel as the standard pixel in the final image, we can compute the desired resolution of the final image. A planar perspective mapping (a 3 × 3 homography matrix) is then computed from the original image quadrangle to the final image rectangle, and the whiteboard image is rectified accordingly.

The last step is white balancing of the background color. This involves two procedures. The first is the estimation of the background color (the whiteboard image under the same lighting if there were nothing written on it). This is not a trivial task because of complex lighting environment, whiteboard reflection and strokes written on the board. The second concerns the actual white balancing. We make the background uniformly white and increase color saturation of the pen strokes. The output is a crisp image ready to be integrated with any office document or to be sent to the meeting participants.

Although we have not yet implemented in our current system, image vectorization is logically the final step. It transforms a bitmap image into vector drawings such as free-form curves, lines and arcs. TabletPC inks use a vector representation, and therefore a whiteboard image after vectorization can be exported into TabletPC.

3. Details of image enhancement

We now provide details of the image processing techniques used in our system. The whiteboard scanning system will be described in Section 3.1.

3.1. Automatic whiteboard detection

As was mentioned in the introduction, this work was motivated by developing a useful tool to capture the whiteboard content with a digital camera rather coping the notes manually. If the user has to click on the corners of the whiteboard, we have not realized the full potential with digital technologies. In this section, we describe our implementation of automatic whiteboard detection. It is based on Hough transform, but needs a considerable amount of engineering because there are usually many lines which can form a quadrangle. The procedure consists of the following steps:

(1) Edge detection.
(2) Hough transform.
(3) Quadrangle formation.
(4) Quadrangle verification.
(5) Quadrangle refining.

Combining with the technique described earlier, we have a complete system for automatically rectifying whiteboard images. Experiments will be provided in Section 3.1.2.

3.1.1. Technical details

We describe the details of how a whiteboard boundary is automatically detected.

Edge detection. There are many operators for edge detection (see any textbook on image analysis and computer vision, e.g., [5–7]). In our implementation, we first convert the color image into a gray-level image, and use the Sobel filter to compute the gradient in x- and y-direction with the following masks:

$$G_x = \begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \quad \text{and} \quad G_y = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

We then compute the overall gradient approximately by absolute values: $G = |G_x| + |G_y|$. If the gradient G is larger than a given threshold T_G, that pixel is considered as an edge. $T_G = 40$ in our implementation.

Hough transform. Hough transform is a robust technique to detect straight lines, and its description can be found in the books mentioned earlier. The idea is to subdivide the parameter space into accumulator cells. An edge detected earlier has an orientation, and is regarded as a line. If the parameters of that line fall in a cell, that cell receives a vote. At the end, cells that receive a significant number of votes represent lines that have strong edge support in the image. Our implementation differs from those described in the textbooks in that we are detecting *oriented* lines. The orientation information is useful in a later stage for forming a reasonable quadrangle, and is also useful to distinguish two lines nearby but with opposite orientation. The latter is important because we usually see two lines around the border, and if we do not distinguish them, the detected line is not very accurate. We use the normal representation of a line:

$$x \cos\theta + y \sin\theta = \rho.$$

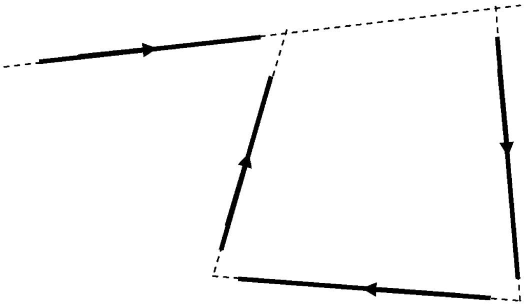


Fig. 2. An example of bad quadrangles.

The range of angle θ is $[-180°, 180°]$. For a given edge at (x_0, y_0), its orientation is computed by $\theta = \operatorname{atan2}(G_y, G_x)$, and its distance $\rho = x_0 \cos\theta + y_0 \sin\theta$. In our implementation, the size of each cell in the $\rho\theta$-plane is 5 pixels by 2°.

Quadrangle formation. First, we examine the votes of the accumulator cells for high edge concentrations. We detect all reasonable lines by locating local maxima whose votes are larger than five percent of the maximum number of votes in the Hough space. Second, we form quadrangles with these lines. Any four lines could form a quadrangle, but the total number of quadrangles to consider could be prohibitively high. In order to reduce the number, we only retain quadrangles that satisfy the following conditions:

- The opposite lines should have quite opposite orientations (180° within 30°).
- The opposite lines should be quite far from each other (the difference in ρ is bigger than one fifth of the image width or height).
- The angle between two neighboring lines should be close to ±90° (within 30°).
- The orientation of the lines should be consistent (either clockwise or counter-clockwise).
- The quadrangle should be big enough (the circumference should be larger than $(W + H)/4$).

The last one is based on the expectation that a user tries to take an image of the whiteboard as big as possible.

Quadrangle verification. The lines detected from Hough space are infinite lines: they do not say where the supporting edges are. For example, the four lines in Fig. 2 would pass all the tests described in the previous paragraph, although the formed quadrangle is not a real one. To verify whether a quadrangle is a real one, we walk through the sides of the quadrangle and count the number of edges along the sides. An edge within 3 pixels from a side of the quadrangle and having similar orientation is considered to belong to the quadrangle. We use the ratio of the number of supporting edges to the circumference as the quality measure of a quadrangle. The quadrangle having the highest quality measure is retained as the one we are looking for.

Quadrangle refining. The lines thus detected are not very accurate because of the discretization of the Hough space. To improve the accuracy, we perform line fitting for each side. For that, we first find all edges with a small neighborhood (10 pixels) and having similar orientation to the side. We then use least-median squares to detect outliers [8], and finally we perform a least-squares fitting to the remaining good edges [5].

3.1.2. Experimental results on automatic whiteboard detection

We have tested the proposed technique with more than 50 images taken by different people with different cameras in different rooms. All the tuning parameters have been fixed once for all, as we already indicated earlier. The success rate is more than 90%. The four failures are due to poor boundary contrast, or to too noisy edge detection. In this section, we provide three examples (Figs. 3–5).

Figure 3 is a relatively simple example because the whiteboard boundary is very clear. The image resolution is 2272 × 1704 pixels. The detected edges are shown in white in Fig. 3b. As can be seen in the Hough image (Fig. 3c), the peaks are quite clear. The corners of whiteboard are accurately estimated, as shown in small white squares in Fig. 3a. The cropped and rectified image is shown in Fig. 3d. The estimated aspect ratio is 1.326, very close to the ground truth 4/3. The estimated focal length is 2149 pixels.

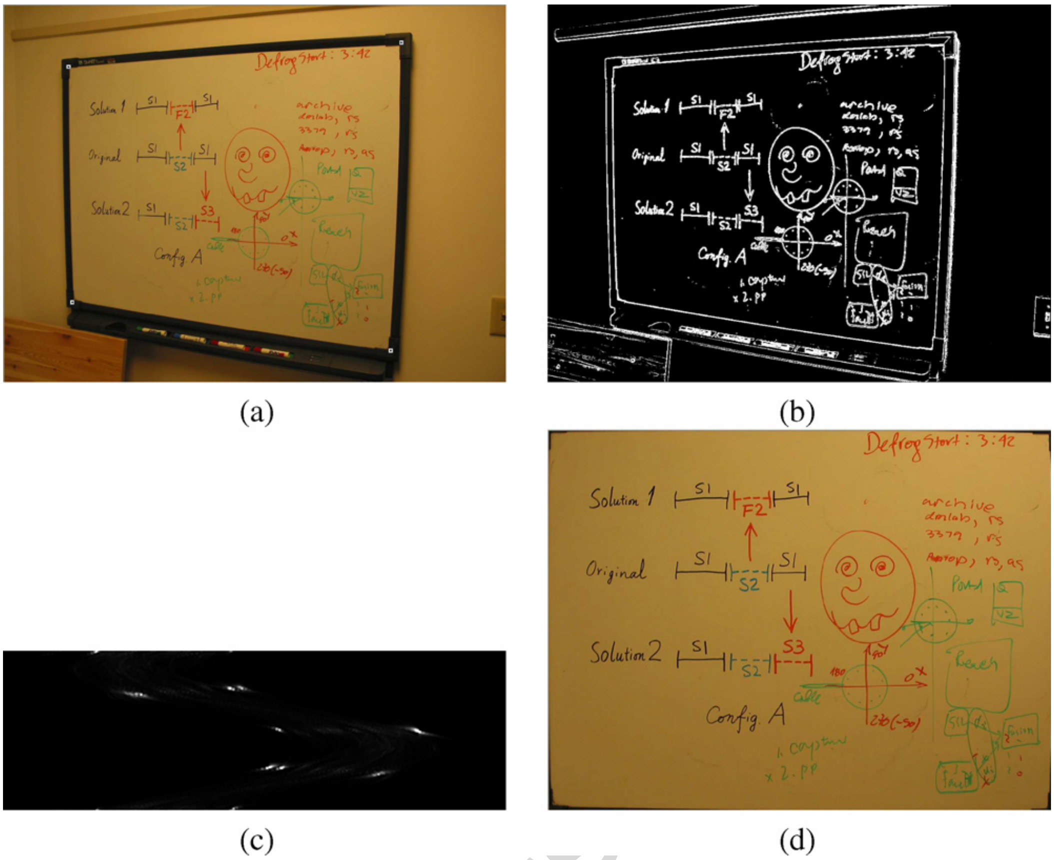


Fig. 3. Example 1. Automatic whiteboard detection and rectification: (a) original image together with the detected corners shown in small white squares, (b) edge image, (c) Hough image with ρ in horizontal axis and θ in vertical axis, (d) cropped and rectified whiteboard image.

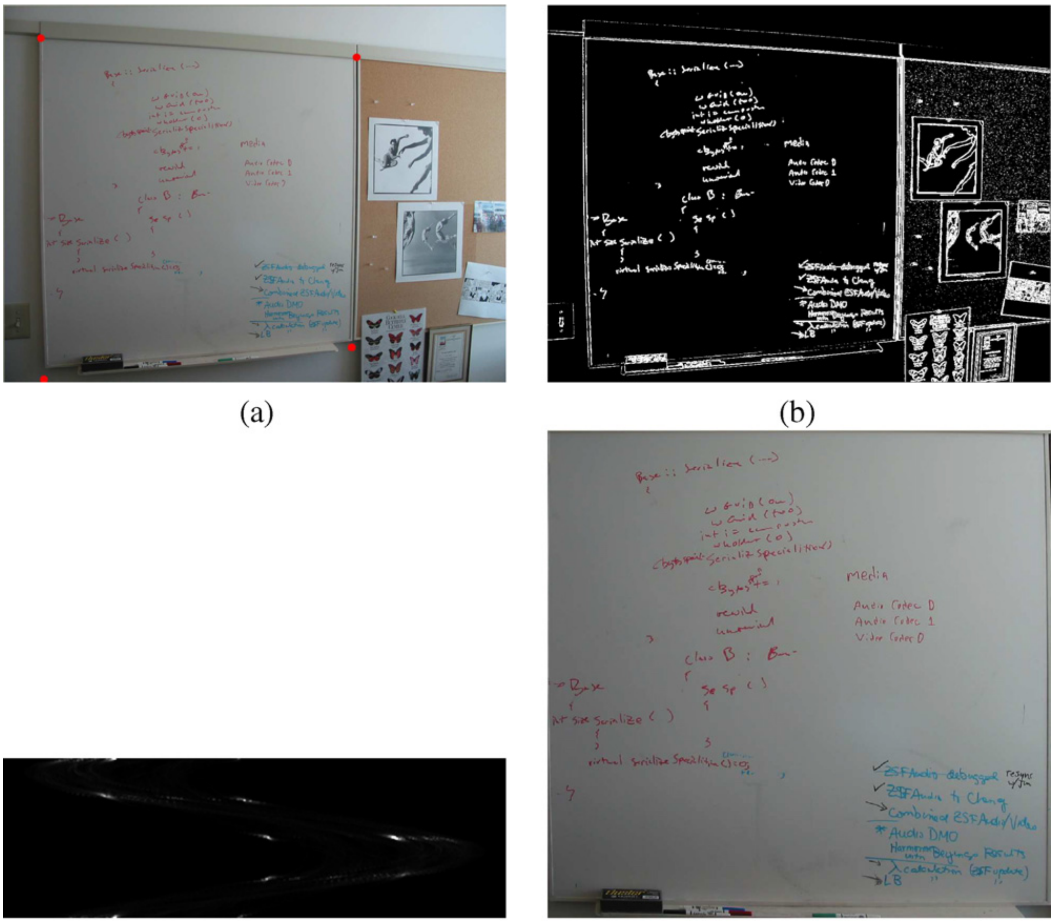
Figure 4 shows a different example. The image resolution is still 2272 × 1704 pixels. As can be seen in the edge image (Fig. 4), the actual lower border of the whiteboard does not have strong edge information. Our technique thus detects the line corresponding to the pen holder, which is perfectly reasonable. The whiteboard corners estimated by intersecting the detected lines are shown in small red dots in Fig. 4a. The cropped and rectified image is shown in Fig. 4d. The estimated aspect ratio is 1.038. The ground truth is 1.05 (the whiteboard is of the same type as in Fig. 7). Since the detected whiteboard includes the pen holder, the estimated aspect ratio (width/height) should be a little bit smaller than 1.05. The estimated focal length is 3465 pixels. We cannot compare the focal lengths because of different zoom settings.

Figure 5 shows yet another example. The resolution is 1536 × 1024 pixels. This example has one particular thing to notice: the upper right corner is not in the image. It does not affect the performance of our technique since we first detect boundary lines rather than corners. In Fig. 5a, the three detected visible corners are shown in small red discs. The fourth corner, although invisible, is also accurately estimated, as can be verified by the cropped and rectified image shown in Fig. 5d, where the invisible region (upper right corner) is filled with black pixels due to lack of information. The estimated aspect ratio is 1.378. We do not have the ground truth because the image was provided by an external person. The estimated focal length is 2032 pixels.

3.2. Determining the physical aspect ratio of a whiteboard

Because of the perspective distortion, the image of a rectangle appears to be a quadrangle. However, since we know that it is a rectangle in space, we are able to estimate both the camera's focal length and the rectangle's aspect ratio.

Single-view geometry of a plane, including plane rectification and mensuration, was addressed in [9]. The case of a rectangular shape was studied in detail in [10]. Here, we address the problem of determining the physical aspect ratio of a whiteboard, which is assumed to be a rectangle, from a single image. Section 3.2.1 derives the basic constraints



(a) (b) (c) (d)

Fig. 4. Example 2. Automatic whiteboard detection and rectification: (a) original image together with the detected corners shown in small red dots, (b) edge image, (c) Hough image with ρ in horizontal axis and θ in vertical axis, (d) cropped and rectified whiteboard image.

from a single view of a rectangle. Section 3.2.2 describes how to use these constraints to estimate the camera's focal length and the actual aspect ratio of the rectangle. Section 3.2.4 provides experimental results with real images.

For general description of projective geometry in computer vision, the reader is referred to [5,11–13].

3.2.1. Geometry of a rectangle

Consider Fig. 6. Without loss of generality, we assume that the rectangle is on the plane $z = 0$ in the world coordinate system. Let the width and height of the rectangular shape be w and h. Let the coordinates of the four corners, M_i $(i = 1, \ldots, 4)$, be $(0, 0)$, $(w, 0)$, $(0, h)$, and (w, h) in the plane coordinate system $(z = 0)$. The projection of the rectangle in the image is an quadrangle. The observed corners in the image are denoted by $\mathbf{m}_1$, $\mathbf{m}_2$, $\mathbf{m}_3$, and $\mathbf{m}_4$, respectively. Furthermore, we will use $\tilde{\mathbf{x}}$ to denote the augmented $\mathbf{x}$ vector by adding 1 as the element to vector $\mathbf{x}$, i.e., $\tilde{\mathbf{x}} = [x_1, \ldots, x_n, 1]^T$ if $\mathbf{x} = [x_1, \ldots, x_n]^T$.

We use the standard pinhole model to model the projection from a space point $\mathtt{M}$ to an image point $\mathbf{m}$:

$$\lambda\tilde{\mathbf{m}} = \mathbf{A}[\mathbf{R} \quad \mathbf{t}]\tilde{\mathtt{M}} \tag{1}$$

with

$$\mathbf{A} = \begin{bmatrix} f & 0 & u_0 \\ 0 & sf & v0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{R} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3]$$

where f is the focal length of the camera, s is the pixel aspect ratio, and $(\mathbf{R}, \mathbf{t})$ describes the 3D transformation between the world coordinate system, in which the rectangle is described, and the camera coordinate system. (In the above model, we assume that pixels are rectangular.) Substituting the 3D coordinates of the corners yields

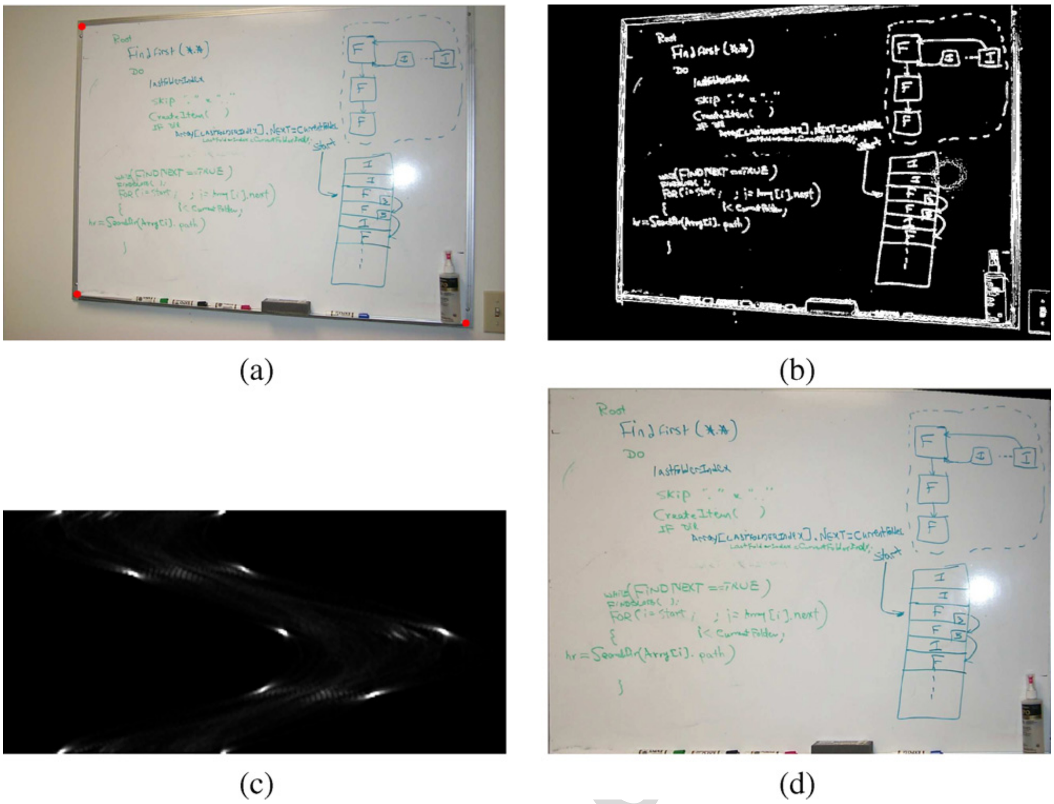


Fig. 5. Example 3. Automatic whiteboard detection and rectification: (a) original image together with the detected corners shown in small red dots (note that the upper right corner is outside of the image), (b) edge image, (c) Hough image with ρ in horizontal axis and θ in vertical axis, (d) cropped and rectified whiteboard image.

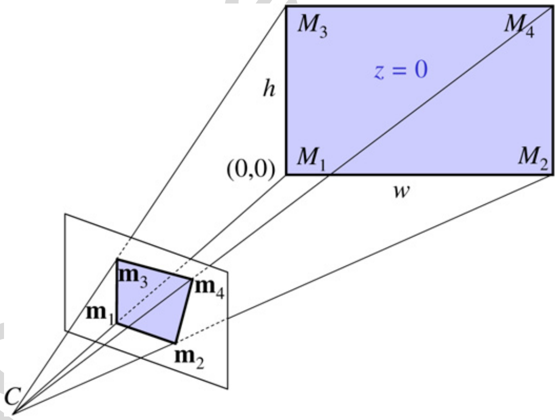


Fig. 6. Geometry of a rectangle.

$$\lambda_1 \tilde{\mathbf{m}}_1 = \mathbf{A}\mathbf{t}, \tag{2}$$

$$\lambda_2 \tilde{\mathbf{m}}_2 = w\mathbf{A}\mathbf{r}_1 + \mathbf{A}\mathbf{t}, \tag{3}$$

$$\lambda_3 \tilde{\mathbf{m}}_3 = h\mathbf{A}\mathbf{r}_2 + \mathbf{A}\mathbf{t}, \tag{4}$$

$$\lambda_4 \tilde{\mathbf{m}}_4 = w\mathbf{A}\mathbf{r}_1 + h\mathbf{A}\mathbf{r}_2 + \mathbf{A}\mathbf{t}. \tag{5}$$

Performing (3)–(2), (4)–(2), and (5)–(2) gives, respectively,

$$\lambda_2 \tilde{\mathbf{m}}_2 - \lambda_1 \tilde{\mathbf{m}}_1 = w\mathbf{A}\mathbf{r}_1, \tag{6}$$

$$\lambda_3 \tilde{\mathbf{m}}_3 - \lambda_1 \tilde{\mathbf{m}}_1 = h\mathbf{A}\mathbf{r}_2, \tag{7}$$

$$\lambda_4 \tilde{\mathbf{m}}_4 - \lambda_1 \tilde{\mathbf{m}}_1 = w\mathbf{A}\mathbf{r}_1 + h\mathbf{A}\mathbf{r}_2. \tag{8}$$

Performing (8)–(6)–(7) yields

$$\lambda_4\tilde{\mathbf{m}}_4 = \lambda_3\tilde{\mathbf{m}}_3 + \lambda_2\tilde{\mathbf{m}}_2 - \lambda_1\tilde{\mathbf{m}}_1. \tag{9}$$

Performing cross product of each side with $\tilde{\mathbf{m}}_4$ yields

$$\mathbf{0} = \lambda_3\tilde{\mathbf{m}}_3 \times \tilde{\mathbf{m}}_4 + \lambda_2\tilde{\mathbf{m}}_2 \times \tilde{\mathbf{m}}_4 - \lambda_1\tilde{\mathbf{m}}_1 \times \tilde{\mathbf{m}}_4. \tag{10}$$

Performing dot product of the above equation with $\tilde{\mathbf{m}}_3$ yields

$$\lambda_2\big(\tilde{\mathbf{m}}_2 \times \tilde{\mathbf{m}}_4\big) \cdot \tilde{\mathbf{m}}_3 = \lambda_1\big(\tilde{\mathbf{m}}_1 \times \tilde{\mathbf{m}}_4\big) \cdot \tilde{\mathbf{m}}_3,$$

i.e.,

$$\lambda_2 = k_2\lambda_1, \quad k_2 \equiv \frac{(\tilde{\mathbf{m}}_1 \times \tilde{\mathbf{m}}_4) \cdot \tilde{\mathbf{m}}_3}{(\tilde{\mathbf{m}}_2 \times \tilde{\mathbf{m}}_4) \cdot \tilde{\mathbf{m}}_3}. \tag{11}$$

Similarly, performing dot product of (10) with $\tilde{\mathbf{m}}_2$ yields

$$\lambda_3 = k_3\lambda_1, \quad k_3 \equiv \frac{(\tilde{\mathbf{m}}_1 \times \tilde{\mathbf{m}}_4) \cdot \tilde{\mathbf{m}}_2}{(\tilde{\mathbf{m}}_3 \times \tilde{\mathbf{m}}_4) \cdot \tilde{\mathbf{m}}_2}. \tag{12}$$

Substituting (11) into (6), we have

$$\mathbf{r}_1 = \lambda_1 w^{-1}\mathbf{A}^{-1}\mathbf{n}_2 \tag{13}$$

with

$$\mathbf{n}_2 = k_2\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1. \tag{14}$$

Similarly, substituting (12) into (7) yields

$$\mathbf{r}_2 = \lambda_1 h^{-1}\mathbf{A}^{-1}\mathbf{n}_3 \tag{15}$$

with

$$\mathbf{n}_3 = k_3\tilde{\mathbf{m}}_3 - \tilde{\mathbf{m}}_1. \tag{16}$$

From the properties of a rotation matrix, we have $\mathbf{r}_1 \cdot \mathbf{r}_2 = 0$. Therefore, from (13) and (15), we obtain

$$\mathbf{n}_2^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{n}_3 = 0. \tag{17}$$

Again, from the properties of a rotation matrix, we have $\mathbf{r}_1 \cdot \mathbf{r}_1 = 1$ and $\mathbf{r}_2 \cdot \mathbf{r}_2 = 1$. Therefore, from (13) and (15), we obtain, respectively,

$$1 = \lambda_1^2 w^{-2}\mathbf{n}_2^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{n}_2, \tag{18}$$

$$1 = \lambda_1^2 h^{-2}\mathbf{n}_3^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{n}_3. \tag{19}$$

Dividing these two equations gives the aspect ratio of the rectangular shape:

$$\left(\frac{w}{h}\right)^2 = \frac{\mathbf{n}_2^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{n}_2}{\mathbf{n}_3^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{n}_3}. \tag{20}$$

This equation says clearly that the absolute size of the rectangle cannot be determined from an image. This is obvious since a bigger rectangular shape will give the same image if it is located further away from the camera.

3.2.2. Estimating camera's focal length and rectangle's aspect ratio

In the last section, we derived two fundamental constraints (17) and (20). We are now interested in extracting all useful information from the quadrangle in the image.

We do not assume any knowledge of the rectangle in space (i.e., unknown width and height). Since we only have two constraints, we have to assume some knowledge of the camera. Fortunately, with modern cameras, it is very reasonable to assume that the pixels are square (i.e., $s = 1$) and the principal point is at the image center (i.e., known u_0 and v_0). Given u_0, v_0 and s, we are able to compute the focal length f from Eq. (17). Indeed, we have

$$f^2 = -\frac{1}{n_{23}n_{33}s^2}\bigl\{\bigl[n_{21}n_{31} - (n_{21}n_{33} + n_{23}n_{31})u_0 + n_{23}n_{33}u_0^2\bigr]s^2 + \bigl[n_{22}n_{32} - (n_{22}n_{33} + n_{23}n_{32})v_0 + n_{23}n_{33}v_0^2\bigr]\bigr\} \tag{21}$$

where n_{2i} (respectively n_{3i}) is the ith component of $\mathbf{n}_2$ (respectively $\mathbf{n}_3$). There exists singularity, and we will address it in Section 3.2.3.

As soon as f is estimated, the camera's intrinsic parameters are all known, and the aspect ratio of the rectangle is readily computed by Eq. (20).

Equation (20) can be used in a different way. If the aspect ratio of the rectangle is given, we can also use that equation to estimate the focal length. Together with (17), we'll have two equations to estimate the focal length, leading a more reliable estimation. However, this is not what we assume in this work.

Once $\mathbf{A}$ is known, the pose of the rectangular shape can be determined. We have from (13)

$$\mathbf{r}_1 = \mathbf{A}^{-1}\mathbf{n}_2 / \|\mathbf{A}^{-1}\mathbf{n}_2\| \tag{22}$$

and from (15)

$$\mathbf{r}_2 = \mathbf{A}^{-1}\mathbf{n}_3 / \|\mathbf{A}^{-1}\mathbf{n}_3\|. \tag{23}$$

In turn,

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2. \tag{24}$$

The translation vector can be determined from (2), i.e.,

$$\mathbf{t} = \lambda_1 \mathbf{A}^{-1}\tilde{\mathbf{m}}_1. \tag{25}$$

Note that the translation can only be determined up to a scale factor λ_1, which depends on the size of the rectangle as can be seen in (18) and (19). This is obvious since a bigger rectangular shape will give the same image if it is located further away from the camera.

3.2.3. Singularity

Obviously, no solution to (21) exists when $n_{23} = 0$ or $n_{33} = 0$. It occurs when $k_2 = 1$ or $k_3 = 1$, respectively. Let us examine it in details.

If $k_2 = 1$, then from (11), we have

$$(\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1) \cdot (\tilde{\mathbf{m}}_4 \times \tilde{\mathbf{m}}_3) = \mathbf{0} \tag{26}$$

or in other words,

$$\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1 = \alpha(\tilde{\mathbf{m}}_4 - \tilde{\mathbf{m}}_3) \quad \text{or} \quad \mathbf{m}_2 - \mathbf{m}_1 = \alpha(\mathbf{m}_4 - \mathbf{m}_3) \tag{27}$$

where α is a non-zero scalar. That is, side $\mathbf{m}_1\mathbf{m}_2$ must be parallel to side $\mathbf{m}_3\mathbf{m}_4$.

This leads to $\mathbf{n}_2 = \tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1$. Because the third element of $\mathbf{n}_2$ is zero, and without loss of generality assuming $u_0 = v_0 = 0$ and $s = 1$ (since they are known), constraint (17) becomes

$$(\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1) \cdot \mathbf{n}_3 = 0 \quad \text{or} \quad (\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1) \cdot (k_3\tilde{\mathbf{m}}_3 - \tilde{\mathbf{m}}_1) = 0. \tag{28}$$

Note that the focal length f has been canceled out. The above equation can be rewritten as

$$(k_3 - 1)\tilde{\mathbf{m}}_3 \cdot (\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1) + (\tilde{\mathbf{m}}_3 - \tilde{\mathbf{m}}_1) \cdot (\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1) = 0. \tag{29}$$

Because $\tilde{\mathbf{m}}_2 - \tilde{\mathbf{m}}_1$ is parallel to the image plane, in order for the first term to be 0, either $k_3 = 1$ or $\mathbf{m}_3 = \mathbf{0}$ (i.e., at the image center, which is a very special case and will not be considered further). In order for the second term to be 0, side $\mathbf{m}_1\mathbf{m}_2$ must be orthogonal to side $\mathbf{m}_1\mathbf{m}_3$.

Since $k_3 = 1$, from (12), we have

$$(\tilde{\mathbf{m}}_3 - \tilde{\mathbf{m}}_1) \cdot (\tilde{\mathbf{m}}_4 \times \tilde{\mathbf{m}}_2) = \mathbf{0}. \tag{30}$$

This means that side $\mathbf{m}_1\mathbf{m}_3$ must be parallel to side $\mathbf{m}_2\mathbf{m}_4$.

In summary, the degenerate case for estimating the focal length when $k_2 = 1$ is when the projection of the rectangle onto the image plane is a rectangle. When $k_3 = 1$, we arrive at the same conclusion.

Fortunately, even though f cannot be estimated when the image of the rectangle is a rectangle (no perspective effect), we can still estimate the aspect ratio. Indeed, since $n_{23} = 0$ and $n_{33} = 0$ because $k_2 = 1$ and $k_3 = 1$ in the

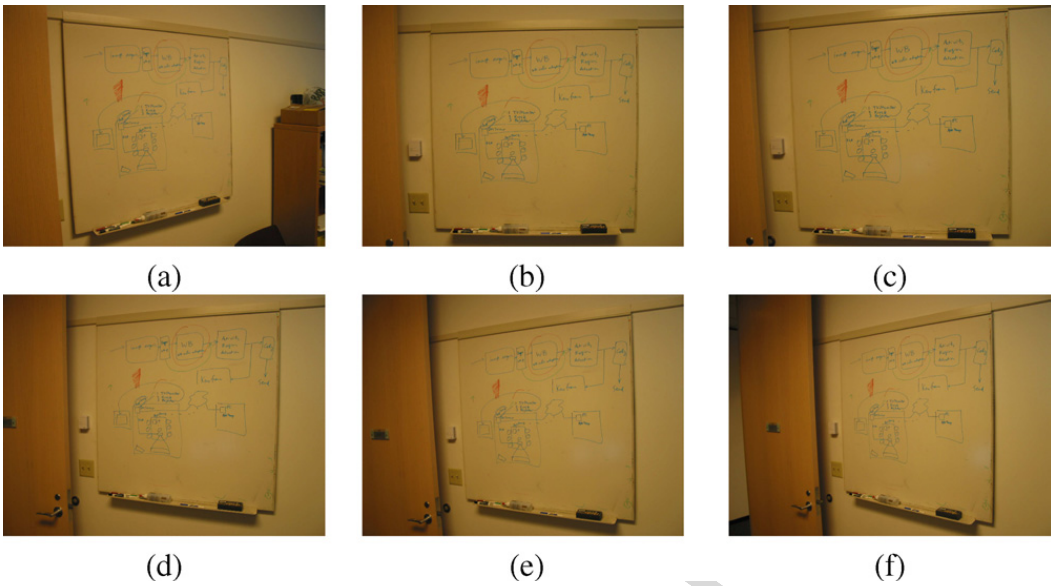


Fig. 7. Six images of the same whiteboard, taken from different angles.

Table 1
Results with images shown in Fig. 7

Image	a	b	c	d	e	f
Aspect ratio	1.028	1.035	1.031	1.021	1.019	0.990
Error (%)	2.1	1.4	1.8	2.8	3.0	5.7
Bounding box	0.966	1.035	0.981	0.892	0.843	0.727
Difference (%)	5.1	1.4	6.6	15.1	19.7	30.8
Focal length	2202	2442	2073	2058	2131	2030

degenerate case, after simple algebra, Eq. (20) becomes

$$\left(\frac{w}{h}\right)^2 = \frac{(n_{21}^2 + n_{22}^2)}{(n_{31}^2 + n_{32}^2)}. \tag{31}$$

Not surprisingly, the physical aspect ratio is equal to the aspect ratio of the rectangle image in this special case.

3.2.4. Experimental results on aspect ratio estimation

In this section, we provide experimental results with real data. Six images of the same whiteboard, as shown in Fig. 7, were taken from different angles. The most frontal view is image b. We manually measured the whiteboard with a ruler, and the size is about $42' \times 40'$ (note: $1' \approx 2.54$ cm). The aspect ratio is therefore 1.05, and we use this as the ground truth.

In each image, we manually clicked on the four corners of the whiteboard, and use the technique described in the last section to estimate the focal length of the camera and the aspect ratio of the whiteboard. The results are shown in Table 1. The second row shows the estimated values of the aspect ratio, while the third row shows its relative error compared with the ground truth. The error is mostly less than 3%, except for image f which was taken from a very skewed angle. There are two major sources contributing to the errors: the first is the precision of the manually clicked points; the second is lens distortion that is currently not modeled. Lens distortion can be clearly observed in Fig. 7. The error of the estimated aspect ratio tends to be higher for images taken from a larger angle. This is expected because the relative precision of the corner points is decreasing. For reference, we also provide the aspect ratio of the bounding box of the whiteboard image in the fourth row of Table 1, and its relative difference with respect to the ground truth in the fifth row. The relative difference can go up to 30%. It is clear that it is not reasonable to use the aspect ratio of the bounding box to rectify the whiteboard images. The sixth row of Table 1 shows the estimated focal length, which varies around 2200.

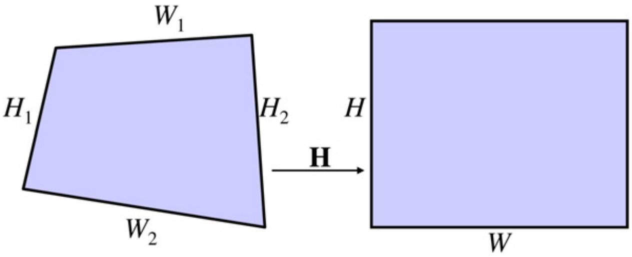
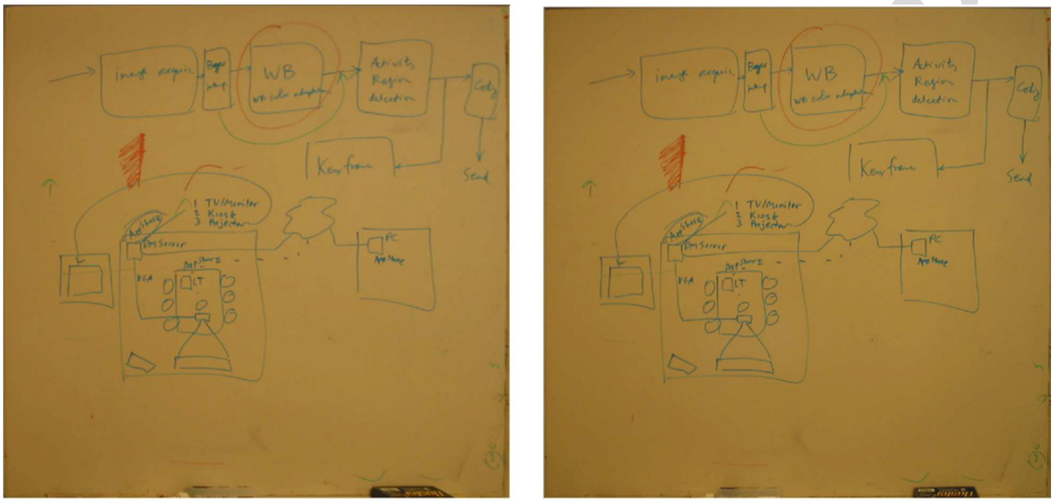


Fig. 8. Rectification of an whiteboard image. Left: original shape, right: rectified shape.



(a) (b)

Fig. 9. Rectified version of the first two images shown in Fig. 7, using the estimated aspect ratios.

Note that when an image of the rectangle is close to be rectangular (e.g., Fig. 7b), it is near the singularity for estimating the focal length, so the estimated focal length is not accurate. On the other hand, the physical aspect ratio is closer to the aspect ratio of the bounding box, so the estimated aspect ratio is most accurate.

3.3. Rectification

The next task is to rectify the whiteboard image into a rectangular shape with the estimated aspect ratio. For that, we need to know the size of the final image. We determine the size in order to preserve in the rectified image maximum information of the original image. In other words, a pixel in the original image should be mapped to at least one pixel in the rectified image. Refer to Fig. 8. The side lengths of the quadrangle in the original image are denoted by W_1 and W_2 for the upper and lower sides, and by H_1 and H_2 for the left and right side. Let $\hat{W} = \max(W_1, W_2)$ and $\hat{H} = \max(H_1, H_2)$. Let $\hat{r} = \hat{W}/\hat{H}$. Denote the estimated aspect ratio by r. We determine the size of the rectified image as follows: $W = \hat{W}$ and $H = W/r$ if $\hat{r} \geqslant r$; otherwise, $H = \hat{H}$ and $W = rH$. Once the size is determined, the rectifying matrix **H** (homography) can be easily computed, and the color in the rectified image is computed through bilinear or bicubic interpolation from the original image.

Figure 9 shows two rectified images of the whiteboard using the estimated aspect ratios. They correspond to images a and b in Fig. 7. The rectified images look almost identical despite that the original images were taken from quite different angles. Other rectified images are also similar, and are thus not shown.

In the examples shown in Figs. 3, 4, and 5, the last picture is respectively the rectified image in each example.

3.4. White balancing and image enhancement

The goal of color enhancement is to transform the input whiteboard image into an image with the same pen strokes on uniform background (usually white). For each pixel, the color value C_{input} captured by the camera can be

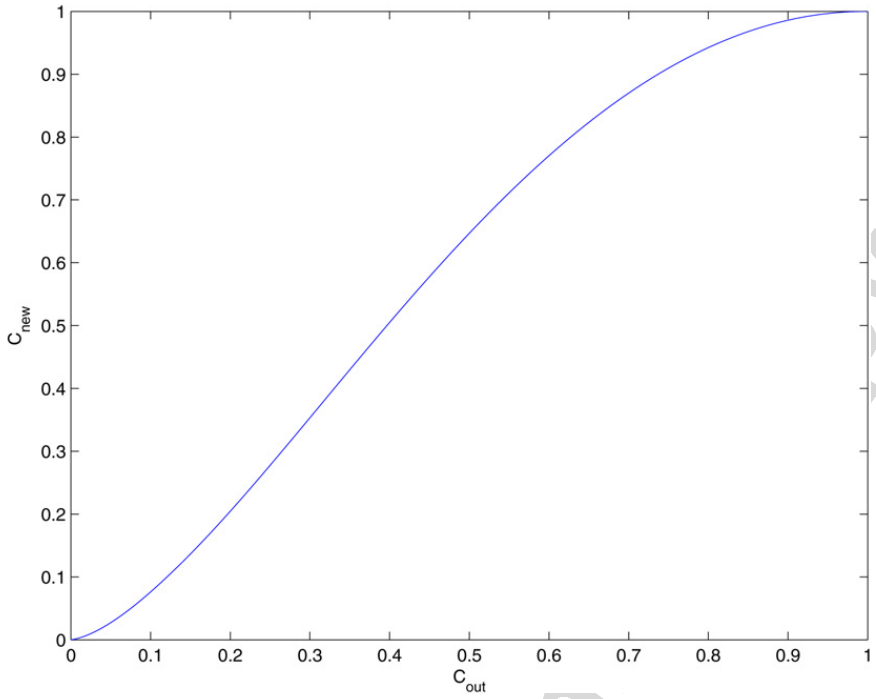


Fig. 10. An example of the S-shaped curve.

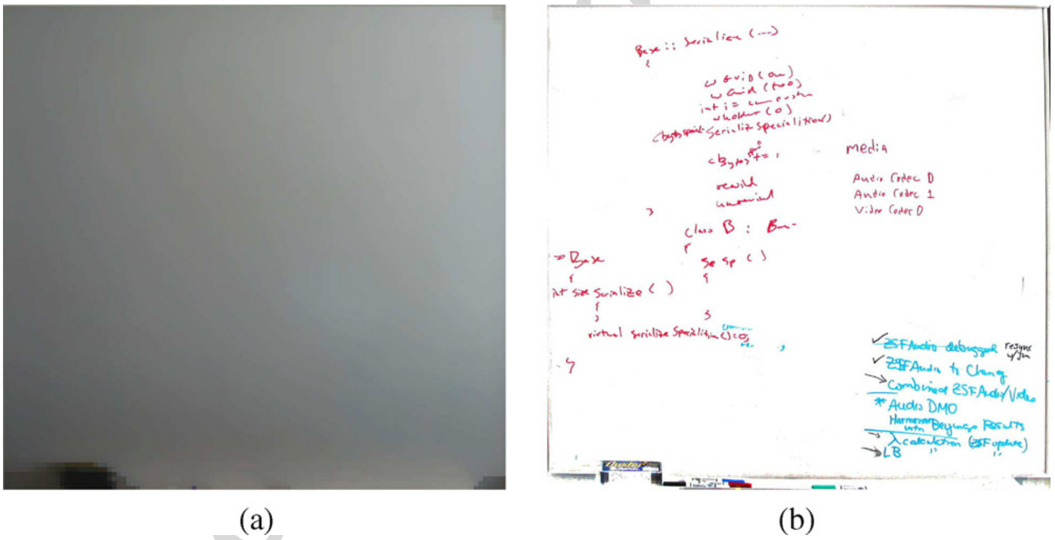
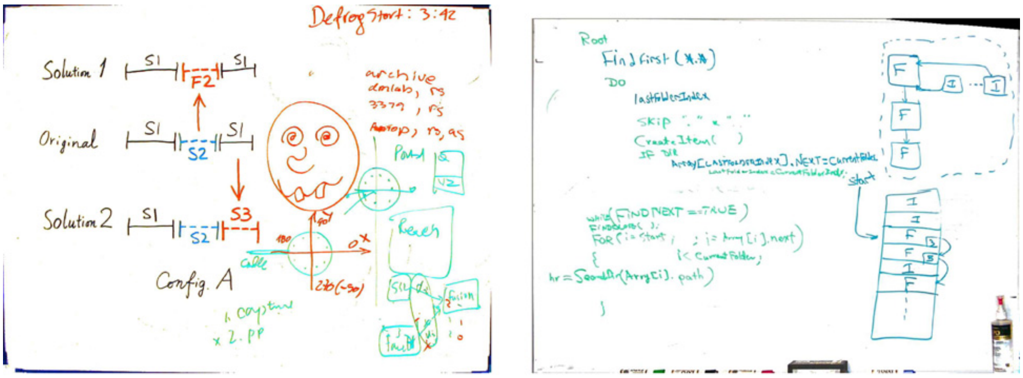


Fig. 11. Whiteboard image enhancement of Example 2, shown in Fig. 4: (a) estimated whiteboard color, (b) final enhanced image.

approximated by the product of the incident light C_{light}, the pen color C_{pen}, and the whiteboard color C_{wb}. Since the whiteboard is physically built to be uniformly color, we can assume C_{wb} is constant for all the pixels, the lack of uniformity in the input image is due to different amount of incident light to each pixel. Therefore, the first procedure in the color enhancement is to estimate C_{light} for each pixel, the result of which is in fact an image of the blank whiteboard.

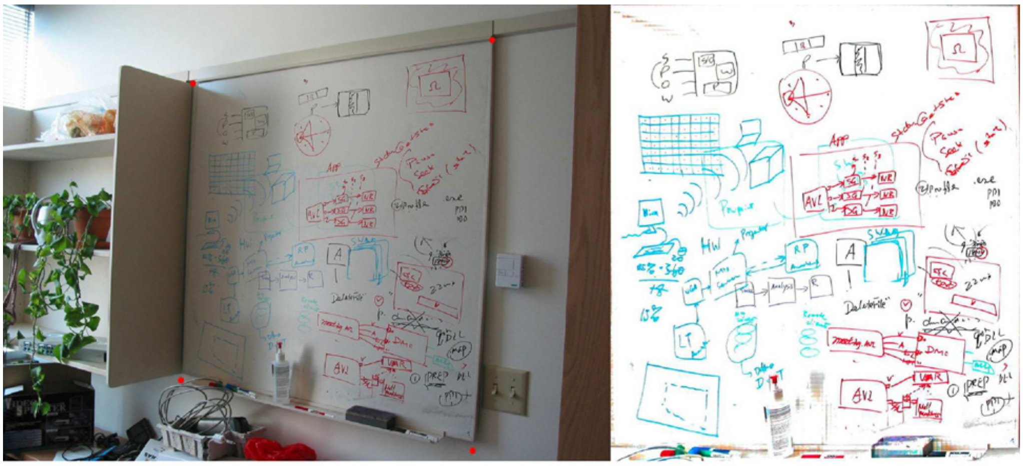
Our system computes the blank whiteboard image by inferring the value of pixels covered by the strokes from their neighbors. Rather than computing the blank whiteboard color at the input image resolution, our computation is done at a coarser level to lower the computational cost. This approach is reasonable because the blank whiteboard colors normally vary smoothly. The steps are as follows:

1. Divide the whiteboard region into rectangular cells. The cell size should be roughly the same as what we expect the size of a single character on the board (15 by 15 pixels in our implementation).



(a) (b)

Fig. 12. Whiteboard image enhancement of Examples 1 and 3: (a) enhanced image from Example 1, shown in Fig. 3, (b) enhanced image from Example 3, shown in Fig. 5.



(a) (b)

Fig. 13. Whiteboard image enhancement in a cluttered office: (a) original image together with the detected corners shown in small red dots, (b) final enhanced image.

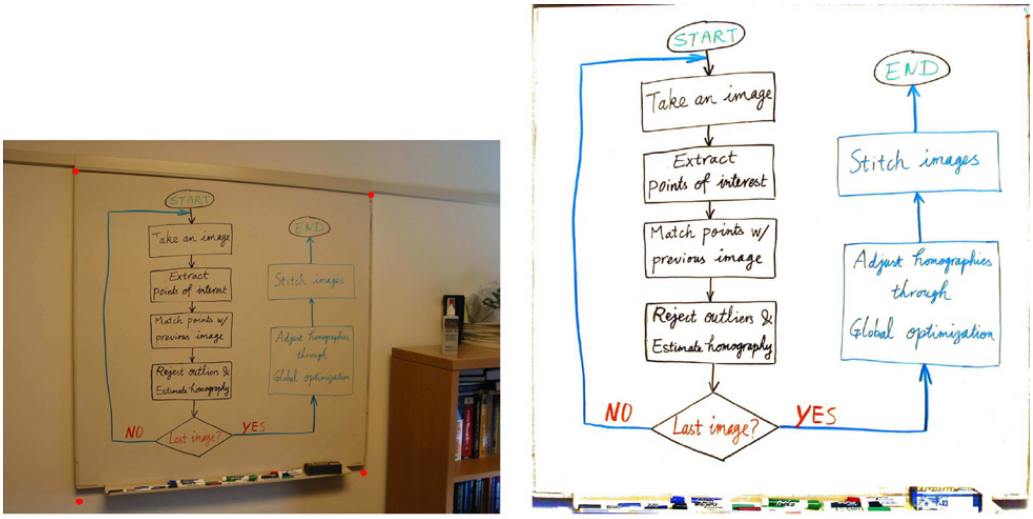
2. Sort the pixels in each cell by their luminance values. Since the ink absorbs the incident light, the luminance of the whiteboard pixels is higher than stroke pixels'. The whiteboard color within the cell is therefore the color with the highest luminance. In practice, we average the colors of the pixels in the top 25 percentile in order to reduce the error introduced by sensor noise.
3. Filter the colors of the cells by locally fitting a plane in the RGB space. Occasionally there are cells that are entirely covered by pen strokes, the cell color computed in Step 2 is consequently incorrect. Those colors are rejected as outliers by the locally fitted plane and are replaced by the interpolated values from its neighbors.

Once the image of the blank whiteboard is computed, the input image is color enhanced in two steps:

1. Make the background uniformly white. For each cell, the computed whiteboard color (equivalent to the incident light C_{light}) is used to scale the color of each pixel in the cell: $C_{\text{out}} = \min(1, C_{\text{input}}/C_{\text{light}})$.
2. Reduce image noise and increase color saturation of the pen strokes. We remap the value of each color channel of each pixel according to an S-shaped curve: $0.5 - 0.5\cos(C_{\text{out}}^{p}\pi)$. The steepness of the S-curve is controlled by p. In our implementation, p is set to 0.75, and the corresponding curve is shown in Fig. 10.

3.4.1. Experimental results on image enhancement

Figure 11 shows the result on the example shown in Fig. 4. Figure 11a shows the estimated whiteboard color as if there were nothing written on it, and Fig. 11b shows the final enhanced image.


START
Take an image
Extract Points of interest
Match points w/ previous image
Reject outliers & Estimate homography
Last image?
NO
YES
Adjust homographies through Global optimization
Stitch images
END

(a) (b)

Fig. 14. Diagram of the scanning subsystem: (a) original image, (b) processed image.

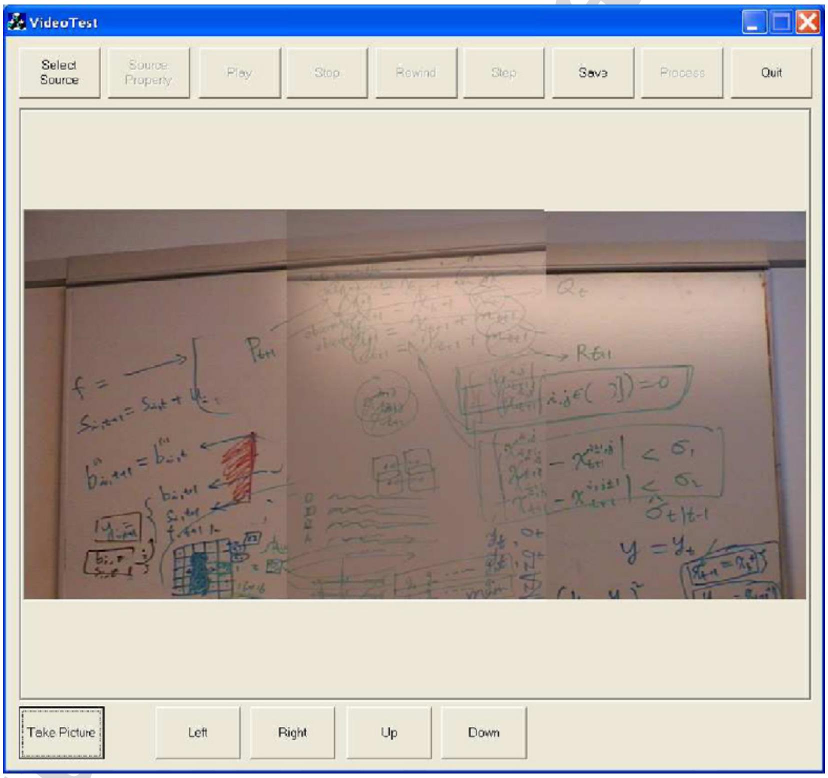


Fig. 15. User interface for whiteboard scanning. Note that half of the previous acquired image and half of the live video are shown in transparent to guide the user to take the next snapshot.

Figure 12 shows the enhanced images from the examples shown in Figs. 3 and 5, respectively.

Figure 13 shows a whiteboard in a cluttered office. As can be seen, the image contains a significant portion of distracting objects, and our software correctly identifies the whiteboard, and does a great job in cleaning up the image.

4. Whiteboard scanning subsystem

The major steps of the Whiteboard Scanning system is illustrated in Fig. 14, and will be explained below. The mathematic foundation is that two images of a *planar* object, regardless the angle and position of the camera, are

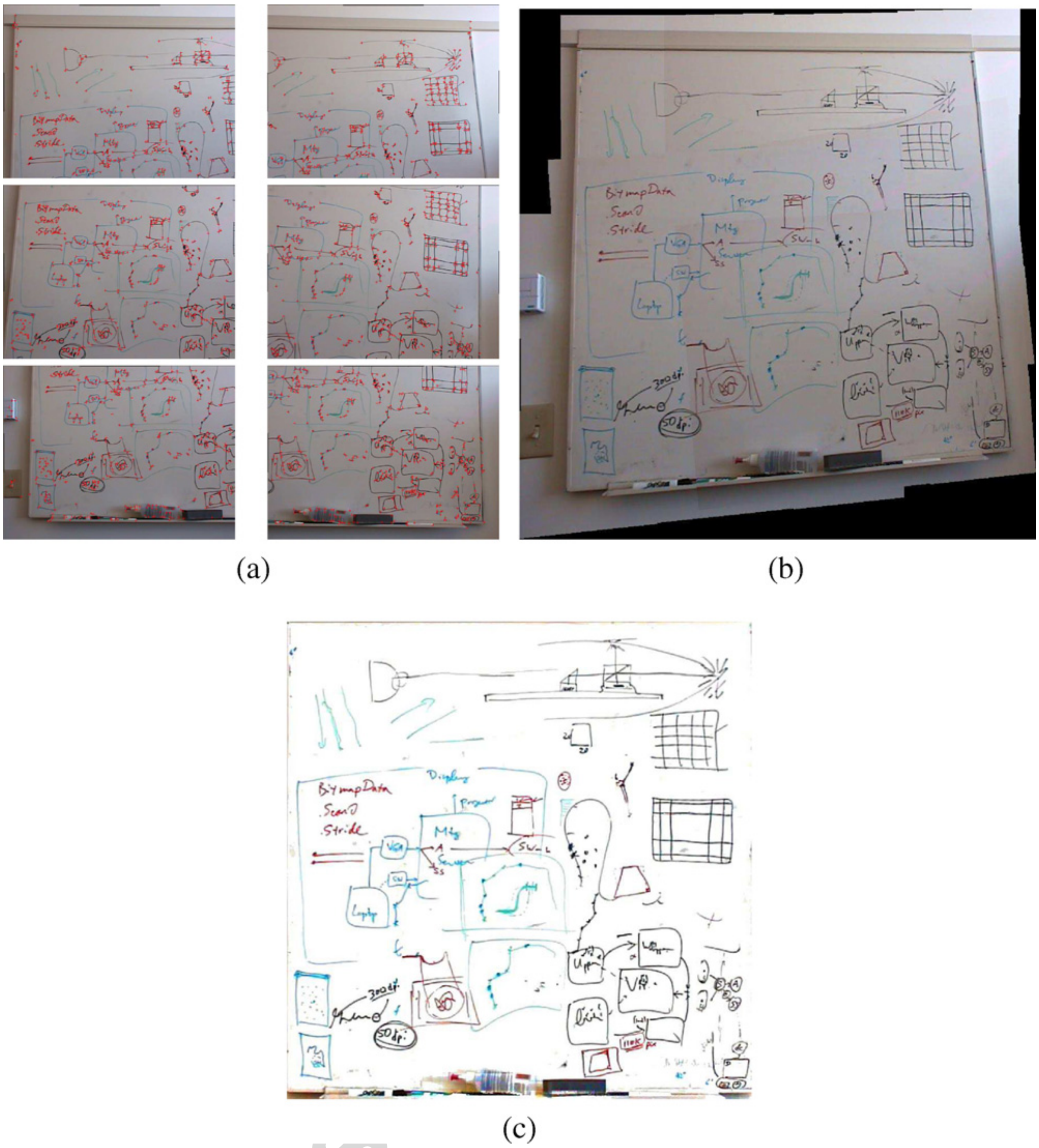


Fig. 16. An example of whiteboard scanning. (a) Original images overlayed with detected points of interest, (b) stitched image, (c) processed image using the technique described in the last section.

related by a plane perspectivity, represented by a 3 × 3 matrix called *homography* **H** [12,13]. The stitching process is to determine the homography matrices between successive images, and we have developed an automatic and robust technique based on points of interest. This has several advantages over classical stitching techniques based on minimizing color differences: (1) less sensitive to color changes between images due to, e.g., different focus; (2) less likely converged to local minima because the points of interest contain the most useful information and because other textureless whiteboard pixels, which would be distracting in color-based optimization, are discarded; (3) robust to large motion because a global search based on random sampling is used.

During whiteboard scanning, 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 with visual feedback during acquisition, as shown in Fig. 15. In the viewing region, we show both the previously acquired image and the current video view. 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

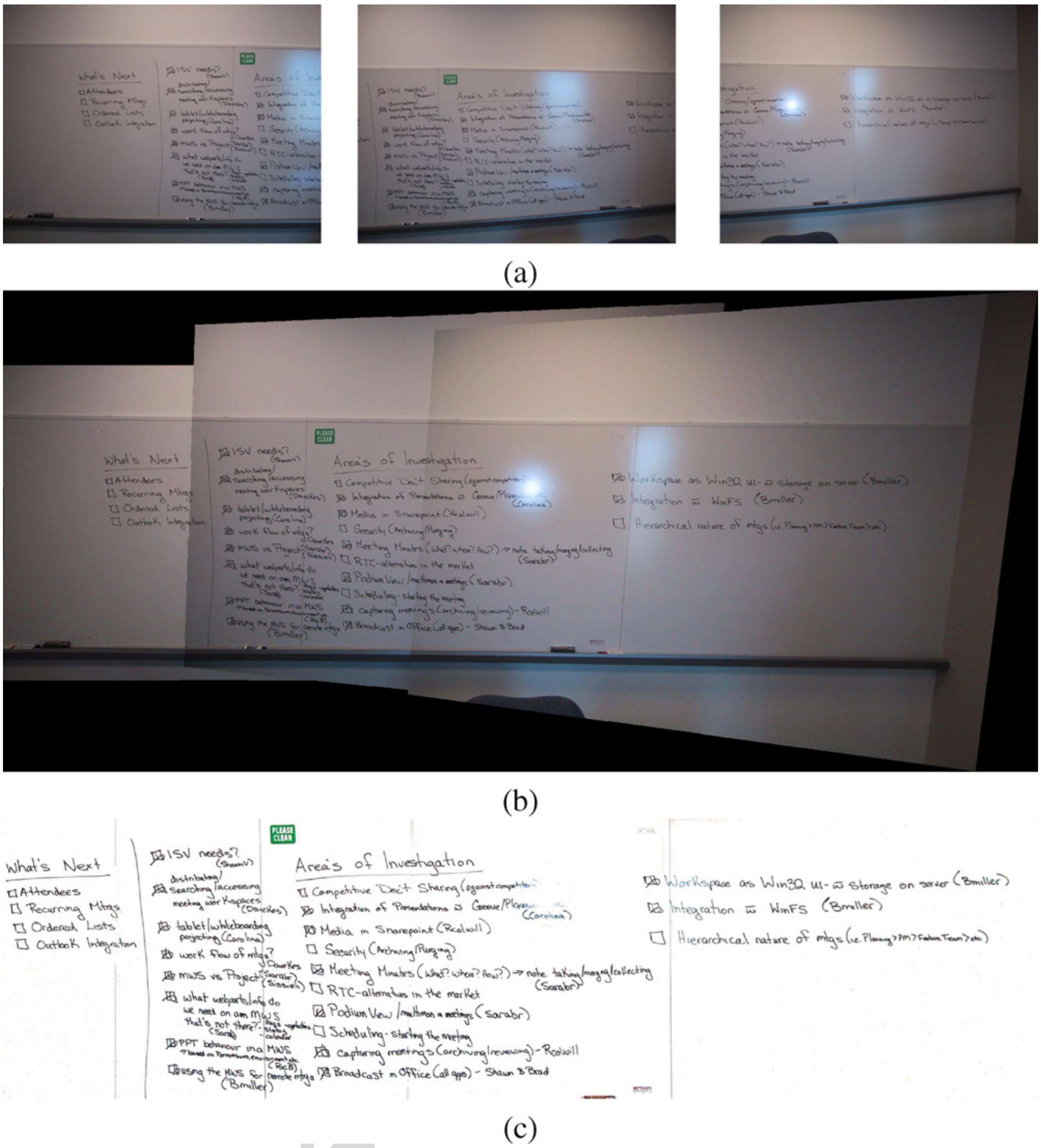


Fig. 17. A second example of whiteboard scanning. (a) Three original images, (b) stitched image, (c) final processed image.

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.

Referring to Fig. 14. For each image acquired, we extract points of interest by using the Plessey corner detector, a well-known technique. These points correspond 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 Fig. 16a, where the extracted points are displayed in red +.

Next, we try to match the extracted points between image. For each point in the previous image, we choose an 15×15 window centered at it, and compare that window with windows of the same size, centered at the points in the current image. A zero-mean normalized cross correlation between two windows is computed. 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 if its match candidate in the previous image happens to be the point being considered. This two-way symmetric test reduces many potential matching errors.

The geometric constraint between two images is the homography constraint. If two points are correctly matched, they must satisfy this constraint, which is unknown in our case. The set of matches established by correlation usually contains false matches because correlation is only a heuristic and uses only local information. Inaccurate location of extracted points because of intensity variation or lack of strong texture features is another source of error. 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* (see, e.g., [8]) to detect both false matches and poorly located corners, and simultaneously estimate the homography matrix **H**.

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 **H**'s through global optimization by considering all the images simultaneously.

Once the geometric relationship between images (in terms of homography matrices **H**'s) are determined, we are able to stitch all images as a single high-resolution 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 image, and map subsequent 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.

Two examples are shown in Figs. 16 and 17.

5. Conclusions

We have presented a digital notetaking system by scanning the content on a whiteboard into the computer with a camera. Images are enhanced for better visual quality. The system has been tested extensively, and very good results have been obtained. Because digital cameras are becoming ubiquitous, our technology may contribute to a significant increase in productivity.

Acknowledgments

The authors would like to thank the reviewer for recommending us to look at in depth the singularity of estimating the focal length.

References

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

[2] PolyVision, Whiteboard Photo, http://www.polyvision.com/products/wbp.asp.

[3] L. He, Z. Liu, Z. Zhang, Why take notes? Use the whiteboard system, in: Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP2003), vol. V, Hong Kong, April 2003, pp. 776–779.

[4] Z. Zhang, L. He, Notetaking with a camera: Whiteboard scanning and image enhancement, in: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004), vol. III, Montreal, Quebec, May 17–21, 2004, pp. 533–536.

[5] O. Faugeras, Three-Dimensional Computer Vision: A Geometric Viewpoint, MIT Press, Cambridge, MA, 1993.

[6] R.C. Gonzalez, R.E. Woods, Digital Image Processing, second ed., Prentice Hall, New York, 2002.

[7] R. Jain, R. Kasturi, B.G. Schunck, Machine Vision, McGraw–Hill, New York, 1995.

[8] Z. Zhang, Parameter estimation techniques: A tutorial with application to conic fitting, Image Vision Comput. 15 (1) (1997) 59–76.

[9] A. Criminisi, Accurate Visual Metrology from Single and Multiple Uncalibrated Images, Springer-Verlag, New York, 2001.

[10] D. Liebowitz, Camera Calibration and Reconstruction of Geometry from Images, Ph.D. dissertation, University of Oxford, 2001.

[11] K. Kanatani, Geometric Computation for Machine Vision, Oxford Univ. Press, London, 1993.

[12] R. Hartley, A. Zisserman, Multiple View Geometry, Cambridge Univ. Press, Cambridge, UK, 1998.

[13] O. Faugeras, Q.-T. Luong, The Geometry of Multiple Images, MIT Press, Cambridge, MA, 2001.

Zhengyou Zhang is a Principal Researcher with Microsoft Research, Redmond, USA. He is an IEEE Fellow, an Associate Editor of the *IEEE Transactions on Multimedia*, an Associate Editor of the *International Journal of Computer Vision (IJCV)*, and an Associate Editor of the *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*. He served on the Editorial Board of the *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* from 2000 to 2004. He is a member of the Multimedia Signal Processing Technical Committee of the IEEE Signal Processing Society. He received the B.Sc. degree in electronic engineering from the University of Zhejiang, China, in 1985, the M.Sc. degree in computer science from the

University of Nancy, France, in 1987, the Ph.D. degree in computer science from the University of Paris XI, France, in 1990, and the Doctor of Science (*Habilitation à diriger des recherches*) diploma from the University of Paris XI, in 1994. He has been with INRIA (French National Institute for Research in Computer Science and Control) for 11 years and was a Senior Research Scientist from 1991 until he joined Microsoft Research in March 1998. In 1996–1997, he spent one-year sabbatical as an Invited Researcher at the Advanced Telecommunications Research Institute International (ATR), Kyoto, Japan. He has published over 150 papers in refereed international journals and conferences, and has co-authored the following books: *3D Dynamic Scene Analysis: A Stereo Based Approach*, Springer-Verlag, Berlin/Heidelberg, 1992; *Epipolar Geometry in Stereo, Motion and Object Recognition*, Kluwer Academic, Dordrecht/Norwell, MA, 1996; *Computer Vision*, Chinese Academy of Sciences, 1998 [in Chinese]. He has been a member of the program committees or a program chair for numerous international conferences, and recently, he is a Technical Co-Chair of the IEEE International Workshop on Multimedia Signal Processing (MMSP), Victoria, BC, Canada, October 2006.

Li-Wei He is a Research Software Design Engineer with Microsoft Research, Redmond, USA. He received the B.Sc. degree in computer science from the University of Washington, USA, in 1996. He joined Microsoft Research in April 1998. He has published over 10 conference and journal papers on multimedia and computer graphics. He is the lead author of the paper that won the best paper award in ACM Multimedia Conference 1999.