Image Mosaicing and a Diagrammatic User Interface for an Office Whiteboard Scanner

Eric Saund
Xerox Palo Alto Research Center
3333 Coyote Hill Rd., Palo Alto, CA 94304
saund@parc.xerox.com

Abstract

This paper presents two computer vision components underlying a novel camera-based whiteboard scanner now in routine use at our research center. First, a fully-automatic feature-based mosaicing algorithm enables high-resolution whiteboard imaging using a video resolution pan/tilt camera, requiring minimal frame overlap even in the presence of sparse curvilinear data. Second, real-time activity detection coupled with grouping-based line drawing analysis enable a Diagrammatic User Interface whereby commands are issued to the system by drawing on the whiteboard itself. This application represents a step toward reducing the barrier between the physical and electronic workspaces.

1 Introduction

Few creative workplaces lack a whiteboard or chalkboard. The form factor of whiteboard-scale surfaces affords pacing, gesticulating, sharing with large and small groups, and stepping back to get a look at the big picture. Small office whiteboards support conversations, lists, and notes; medium size conference room whiteboards participate in presentations and group collaborations; large whiteboard walls maintain organizational reference material including schedules, timetables, and assignment postings.

The existence of several commercial devices for online or offline whiteboard image capture suggests that indeed the material drawn on whiteboards often is valuable and worth preserving. Most of these involve a specially instrumented device that severely limits drawing area [10, 11, 2]. By contrast, camera-based whiteboard image capture using a single snapshot severely limits image resolution[14].

Image mosaicing permits the capture of large areas at high resolution, e.g. [13, 3, 15]. Solutions involving hand-held video cameras are, however, cumbersome and daunting for users, especially in business environments where priorities of meeting content and social interaction preclude virtually any form of "fighting with technology."

We have therefore endeavored to build office appliances that employ computer vision and other technology in order to offer users electronically-enabled enhanced functionality, but in a "calm" setting [16]. By mounting a pan/tilt camera in the ceiling, the physical machinery for whiteboard capture is moved out of the way, to a place where it can double as a user interface input device. Capture of the needed zoomed-in frames is fully automated; a "photograph" of the whiteboard appears on a nearby printer a few minutes after a scan command is issued.

In addition to a web-based user interface and a physical "GO" button, we provide a *Diagrammatic User Interface* (DUI), that continually monitors activity in front of the whiteboard and watches for users to draw and annotate "buttons" indicating commands and their associated parameters. Whereas a physical button is the most straightforward mechanism for getting a machine to do something, a diagram is often the most effective way of indicating spatial information, while handprinting is a natural medium for conveying symbolic data. Stafford-Fraser describes an initial exploration of this idea [12].

Our system, called "ZombieBoard" (it brings to electronic life the ink marks on a whiteboard), has successfully been in routine use at our research center since the spring of 1997, and during that time has delivered several hundred images to non-expert users. This paper is in two major sections. Section 2 describes the feature-based mosaicing algorithm; Section 3 describes the Diagrammatic User Interface. By necessity of brevity, this overview of ZombieBoard omits many details and important considerations. A more complete description of the system is presented in [8].

2 Feature-Based Image Mosaicing

The automatic construction of image mosaics from multiple overlapping images has recently become a

popular outgrowth of computer vision research. The basic mosaicing problem is to determine image transformation parameters (e.g. pure translation, affine, true perspective) for all component snapshots that will align snapshots' overlapping regions without showing seams. Two basic approaches have been taken; both seek transformation parameters that optimize an objective function. Motion-based alignment methods seek to minimize some cost function (such as sum of squared error) of pixel-by-pixel intensity differences in the overlap regions of overlapping snapshots. These methods are related to optical flow algorithms by the use of intensity gradients to direct the search over transformation parameters. Alternatively, feature-based alignment methods first identify corresponding features in overlapping snapshots, then seek image transformation parameters that optimally align the features.

Motion-based methods typically require substantial overlap between successive image snapshots, and typically require significant image texture in order to align the frames. For a whiteboard scanning application, however, it cannot be guaranteed that markings will be present over the entire board surface. In fact it is common to encounter a great deal of blank space. Furthermore, in the interest of image capture and processing efficiency, we seek to grab as few frames as possible with a minimal amount of overlap between them. For these reasons we have developed a fully-automatic feature-based mosaicing technique.

By employing a computer-controlled pan/tilt camera, we gain control over the input data in comparison with images from hand-held cameras. Through calibration of the camera's position and orientation with respect to the whiteboard, an initial dead-reckoning estimate of the perspective transformation from image coordinates to reconstructed whiteboard coordinates is computed from the pan/tilt calibration parameters and the ostensible pan and tilt settings for each snapshot. Unavoidable errors in camera calibration, however, lead to residual mismatch between overlapping tiles. The bulk of the computational effort is therefore directed to finding corresponding features and refining the snapshot transform parameters that will lead to a coherent mosaic image without seams.

2.1 Selection of Features

The first step of the mosaicing procedure is to find corresponding image features for every pair of image snapshots predicted to overlap one another based on dead-reckoning information. In a whiteboard scanning application it is not uncommon for a snapshot overlap region to contain markings consisting only of relatively straight lines and therefore providing only one direction of constraint for feature matches. Figure 1 illustrates this "aperture" problem. Our feature selection algorithm therefore preferentially selects image patches providing constraint in two dimensions, and secondarily, image patches providing constraint along only one direction.

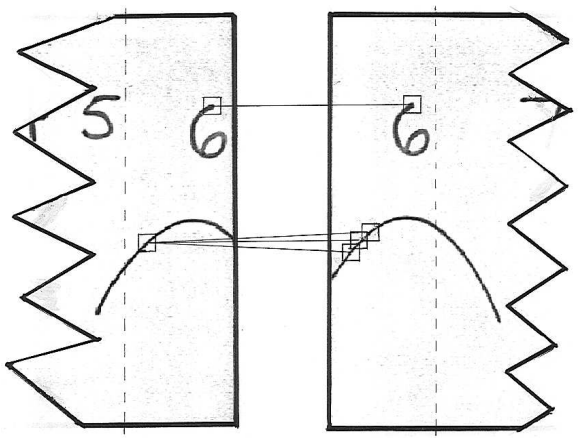

Figure 1: "Aperture" problem in feature correlation. Shown are the overlapping regions of two image snapshots. The image patch at the top of the "6" can be well localized, while the line-like image patch below can slide along the corresponding image region in the other snapshot.

2.2 Feature Matching

Each feature of every snapshot is matched with corresponding features in overlapping snapshots using coarse-to-fine correlation. Experience has shown it important to perform feature correlation to subpixel accuracy. This is obtained by fitting an ellipsoid to the correlation scores in the vicinity of the best pixel-quantized correlation match.

The ellipsoid fit also performs an important function regarding intermediate quality features that constrain the match along only one direction. The orientation and aspect ratio of the ellipsoid are taken as "variance"-like indicators of the one-dimensional sliding constraint provided by aperture-limited features [1]. Figure 2 depicts the feature matches and constraint geometries provided by features in a typical snapshot overlap region. In addition to these true feature matches, each snapshot maintains four virtual features located at the corners of the snapshot's image frame, related directly to the whiteboard coordinate system as estimated by dead-reckoning. These ensure that some reasonable solution can be found in blank (featureless) whiteboard regions.

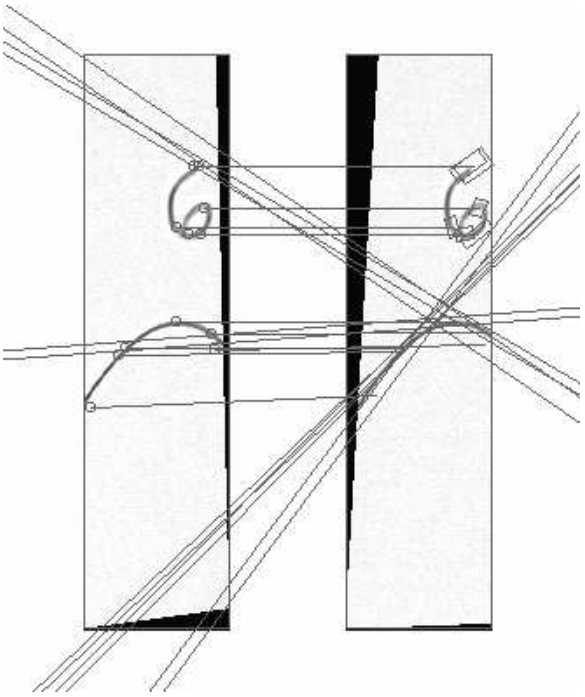

Figure 2: Feature matches. Lines indicate matches between reference feature patches (centered at circles in the left image) and corresponding locations in right image. Each match is characterized by an oriented rectangle indicating the orientation and translational constraint along and across the feature's orientation as determined by an ellipsoid fit to the correlation surface.

2.3 Transformation Refinement

The perspective transformation between image points (X, Y) and locations in the whiteboard coordinate system (x, y) is given by,

$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \qquad (1)$$

$$x_i = \frac{x'_i}{z'_i} = \frac{t_{11}X_i + t_{12}Y_i + t_{13}}{t_{31}X_i + t_{32}Y_i + 1}$$

$$y_i = \frac{y'_i}{z'_i} = \frac{t_{21}X_i + t_{22}Y_i + t_{23}}{t_{31}X_i + t_{32}Y_i + 1}$$

Given at least four image points and their (purportedly) known locations on the whiteboard, we can obtain a least-squares solution for the transformation parameters T by rearranging these expressions to form a system of linear equations (see Mundy and Zisserman [4], pg.482). To accommodate the aperture-limited constraints available from line-like features we apply rotation and scaling to each of the features' entries in the linear system to weigh the penalty for feature location errors according to the orientation and "variance" parameters of the ellipsoidal correlation constraint model associated with each correlation match. In general more than four feature matches are found, and the overconstrained matrix equation is solved for the weighted least-squares optimal transformation parameters using singular value decomposition.

2.4 Collective Refinement of Snapshots' Transformation Parameters

The most popular strategy for building image mosaics, known as the "painting" strategy, is to first select one snapshot as an anchor which is copied or transformed into the whiteboard "canvas." Then successive snapshots are placed by aligning with whatever image features can be matched on the canvas. For example, one might start by placing the upper-left snapshot by dead-reckoning, then fill in the remainder of the whiteboard image by tracing a zig-zag path with the remaining tiles.

This approach is problematical because of accumulation of errors. When snapshot alignments are constrained by matching features on only one or two sides, small errors in feature location estimates can lead to significant distortions on the unconstrained sides. These distortions accumulate and lead to unacceptably distorted and misaligned mosaics.

An alternative strategy made possible by the feature-based approach is to iteratively refine all snapshots' transformation parameters collectively. For each feature whose image coordinates are localized in a pair of overlapping snapshots, take as the estimate of the whiteboard coordinates (x, y) of this feature the midpoint of the projections of the feature into whiteboard coordinates according to each of the two snapshots' current estimated transform parameters. Thus at each iteration every snapshot adjusts itself to improve its alignment with its neighbors on all sides.

2.5 Rejection of Incorrect Feature Matches

Occasionally the correlation-based feature matching process results in incorrect feature matches. This happens, for example, through aliasing of image patches containing closely spaced parallel lines. To keep mismatching features from disrupting the placement of snapshots in the mosaic, these need to be detected and eliminated.

Our approach is to examine each feature's strain energy, which we take as the weighted squared distance between the feature's predicted whiteboard locations according to each of the overlapping snapshots the fea-

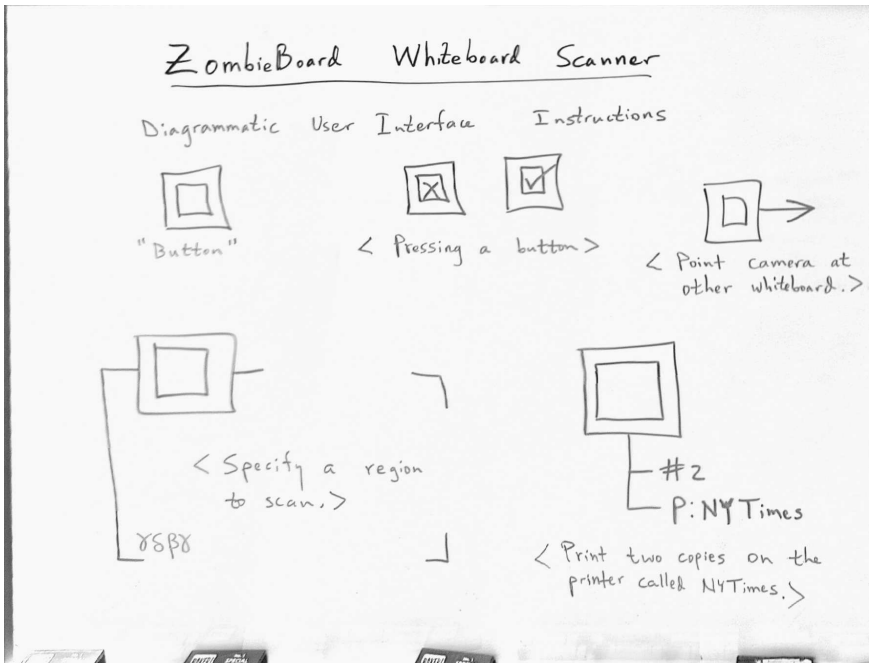

Figure 3: Scanned whiteboard image. The content written on this whiteboard are instructions-by-example for using the Diagrammatic User Interface.

ture is found in. The weighting factor is simply the weighting along and across the feature's local orientation as estimated by the oriented ellipsoid fit to the correlation surface. As relaxation proceeds, most features' strain energies will decrease as their projections into world coordinates are brought into line with one another. Mismatching features' strain energies will remain high, however, and we may reject features whose strain energies fall above a threshold. The threshold is adjusted downward during the course of the settling process according to an annealing schedule. Virtually all features are accepted at first, when all features' strain energies are high due to errors in the dead reckoning estimate. As the snapshots begin to align with one another, the mismatching features begin to stick out and are rejected. As mismatching features are rejected, correctly matching features can be brought into even better alignment, and the strain energy threshold is lowered. Overall, with four threshold levels in the annealing schedule the entire process typically takes between 25 and 50 relaxation cycles, with perhaps 2% of features rejected as erroneous matches.

A completed image mosaic is shown in figure 3.

3 Diagrammatic User Interface

A great deal of activity in computer vision has recently been directed toward Human/Computer Interface applications supported by analysis of faces, hand and arm gestures, and body motions. Common to these modalities is an interaction paradigm involving transient information, for example, the direction of a gaze, the posture of a hand, or the wave of an arm. In many of these cases users must learn to execute prescribed spatio-temporal motor sequences. By contrast, a statically available drawing can be created at the user's own pace, and edited to their satisfaction. We have chosen to explore this direction for a user interface application of computer vision.

The ZombieBoard Diagrammatic User Interface consists of two main functions. First a real-time Activity Analysis module filters an image stream to extract subimages that could possibly represent a diagrammatic command. Then, a line drawing analysis module detects and interprets any visible commands by extracting and analyzing the spatial pattern of whiteboard markings.

We have designed a simple diagrammatic command language which is depicted explicitly for users in an "Instructions" page mounted beside the whiteboard. (The content of these instructions are replicated in handwritten form in the scanned image of Figure 3.) The *Nested Box Button* (NBB) figure is the key shape whose recognition initiates additional analysis to gather diagrammatic annotations.

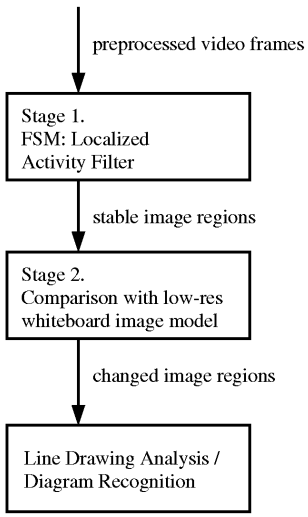

Figure 4: Stages of activity analysis in the Diagrammatic User Interface.

3.1 Activity Analysis

When not engaged in collecting zoomed-in snapshots of the whiteboard, the camera is zoomed back to view the entire whiteboard area. Detection and interpretation of diagrammatic commands is performed using images captured at this relatively low resolution. For the Activity Analysis module, camera images are high-pass filtered to remove sensitivity to lighting changes.

In general the detailed analysis of the markings on a whiteboard is a compute-expensive job, even when the system is looking only for a stereotypical pattern such at the key Nested Box Button. Relatively little of the raw input stream is new material drawn on the whiteboard though; most of the time, most of the input images contain whiteboard material previously seen and analyzed, or else people engaged in whiteboard work.

We have therefore designed a two-stage Activity Analysis filter whose function is to pass to the line drawing analysis module images only of newly modified persistent image content exemplified by material newly written on the whiteboard. See Figure 4. The image is partitioned into local image blocks on which the Activity Analysis filter works in parallel. Each block implements a finite state machine. In the first Activity Analysis stage, image blocks cycle through the following states:

- The **Resting** state reflects the absence of any image changes in a block whose image content has been fully analyzed.

- The **Motion** state indicates that the image block is undergoing change. Change is detected by detecting interlacing and by frame differencing, and may be due to a person moving in front of the whiteboard and possibly drawing on it.

- The **Stable After Motion** state indicates that after an image block had been registered in the Motion state, it has remained unchanged for a certain period of time (e.g. 2 seconds). An image block in this state could contain material newly drawn on the whiteboard.

In the second Activity Analysis stage, Stable After Motion image blocks are compared with a stored image representing the current model of the stable content of the scene. If these match, such as occurs when a person walks in front of the whiteboard without modifying its content, then the image block simply reverts to the Resting state. If however these are sufficiently different, such as when markings are added to or erased from the whiteboard, the contents of the current scene model are updated with the content of that image block, and that block is passed (with others) on to the line drawing analysis module.

As with many computer vision algorithms, the actual implementation of this module is somewhat more involved than portrayed in this overview.

3.2 Line Drawing Analysis

Due to unconstrained imaging geometry and tremendous variability by people in drawing even simple figures such as the Nested Box Button, the line drawing analysis module must be extremely tolerant to deviations from the prototypical line drawing commands. Most techniques used in engineering line drawing analysis are unsuitable. Direct template matching would be out of the question. For simply detecting NBBs, it is possible that a brute-force neural network technique such as used to detect faces in images [5] would succeed, but the computational cost would be quite high, and this approach would gain nothing toward further analysis of diagrammatic annotations to the basic Nested Box Button "Go" command.

Instead, the approach we take is based on perceptual grouping [6, 9]. The incoming line drawing image is subjected to center-surround filtering, thresholding, and thinning. Curvilinear lines are collected by tracing, and perceptually salient corners are found by a multiscale corner detection algorithm (see [7]). The

result is a set of primitive curve element tokens representing relatively straight curvilinear contour segments. Tokens then undergo a series of grouping operations designed to make explicit spatial structure such as extended curvilinear arcs, corners, parallels, and nested corners. See figure 5. For this level of structure, rules governing spatial configurations of tokens satisfying corner, parallel, and nested-corner relations are hand-tuned.

Recognition of NBB figures is done by model-based matching to a prototype configuration of nested-corner and parallel features. Means and variances of the pose parameters (relative location, orientation, and scale) of these features were refined incrementally by training on positive exemplar NBBs gathered from naive users over time by accumulating images that cruder versions of the program failed to recognize. Initial hypotheses for NBBs are generated from Nested-Corner features oriented to possibly form the upper left corner of an NBB.

Once a Nested-Box-Button is recognized, additional routines are run to gather annotations of the command: (1) The NBB is tested to see if is "pressed" by virtue of possessing curve fragments forming an X or check mark drawn inside; (2) Curvilinear tokens extending from one side of the NBB could possibly represent an arrow, in which case curve fragments case the arrowhead are tested; (3) Curves extending from both sides indicate an encircling region, in which case curve tracing routines are brought into play. The encircling command prototcol permits the encircling region to be noncontinuous, so long as it is reasonably evident where the boundaries of an enclosing polygon lie. The curve tracing routine is thus an iterative function that traces continuous curve sections by following tokens linked end-to-end. Where a curve terminates, a "flashlight" algorithm is used by which an expanding search beam is directed outward to find curve fragments representing its continuation elsewhere on the whiteboard.

In operation, when running on a Sparc 20 the DUI normally responds to hand-drawn commands within 5 to 10 seconds. This amount of delay is very significant and understandably annoying to users and we anticipate improving response time through the use of faster computing and frame-grabbing hardware.

Acknowledgements

Much credit is due to Dietmar Aust for developing and implementing the initial DUI Activity Analysis module.

References

[1] Anandan, P.; [1989]; "A computational framework and an algorithm for the measurement of visual motion," *IJCV* 2, 283-310.

[2] Copyboard products; *Kintronics (http://www.kintronics.com/plus.htm,panaboard.html)*

[3] Irani, M., Anandan, P., and Hsu, S.; [1995]; "Mosaic Based Representations of Video Sequences and Their Applications," in *Proc. 5th Int. Conf. Computer Vision*, pp. 605-611, June 20-23, Cambridge, MA.

[4] Mundy, J. and Zisserman, A., eds., *Geometric Invariance in Computer Vision*, MIT Press, Cambridge, MA. 1992.

[5] Rawley, H., Baluja, S., and Kanade, T.; [1996]; "Human Face Detection in Visual Scenes," in Touretzky, D., Mozer, M., and Hasselmo, M., eds., *Advances in Neural Information Processing Systems (NIPS 8)*, pp. 875-881.

[6] Saund, E.; [1990]; Symbolic Construction of a 2-D Scale-Space Image. *IEEE TPAMI*, **12**:8, 817-830.

[7] Saund, E.; [1993]; "Identifying Salient Circular Arcs on Curves." *CVGIP: Image Understanding*, **58:**3, 327-337.

[8] Saund, E.; [1998]; "The ZombieBoard Whiteboard Scanner," Technical Report, Xerox PARC, (forthcoming).

[9] Saund, E., and Moran, T.; [1994]; A Perceptually Supported Sketch Editor. *Proc. ACM Symposium on User Interface and Software Technology(UIST'94)*, 175-184.

[10]; SmartBoard, *Smart Technologies, Inc. (http://www.smarttech.com).*

[11]; Softboard, *Microfield Graphics, Inc. (http://www.micg.com).*

[12] Stafford-Fraser, Q.; [1996]; "BrightBoard: A Video-Augmented Environment," *Proc. ACM CHI '96.*

[13] Szeliski, R.; [1994]; "Image Mosaicing for Tele-Reality Applications," 2nd IEEE Workshop on Appl. of Computer Vision (WACV '94), Sarasota, Florida.

[14]; Tegrity, Inc.*http://www.tegrity.com.*

[15]; VideoBrush Whiteboard, *Videobrush Corporation (http://www.videobrush.com).*

[16], Weiser, M., and Brown, J. S.; [1996]; "Designing Calm Technology," *PowerGrid Journal*, v. 1.01, *http://powergrid.electriciti.com/1.01*

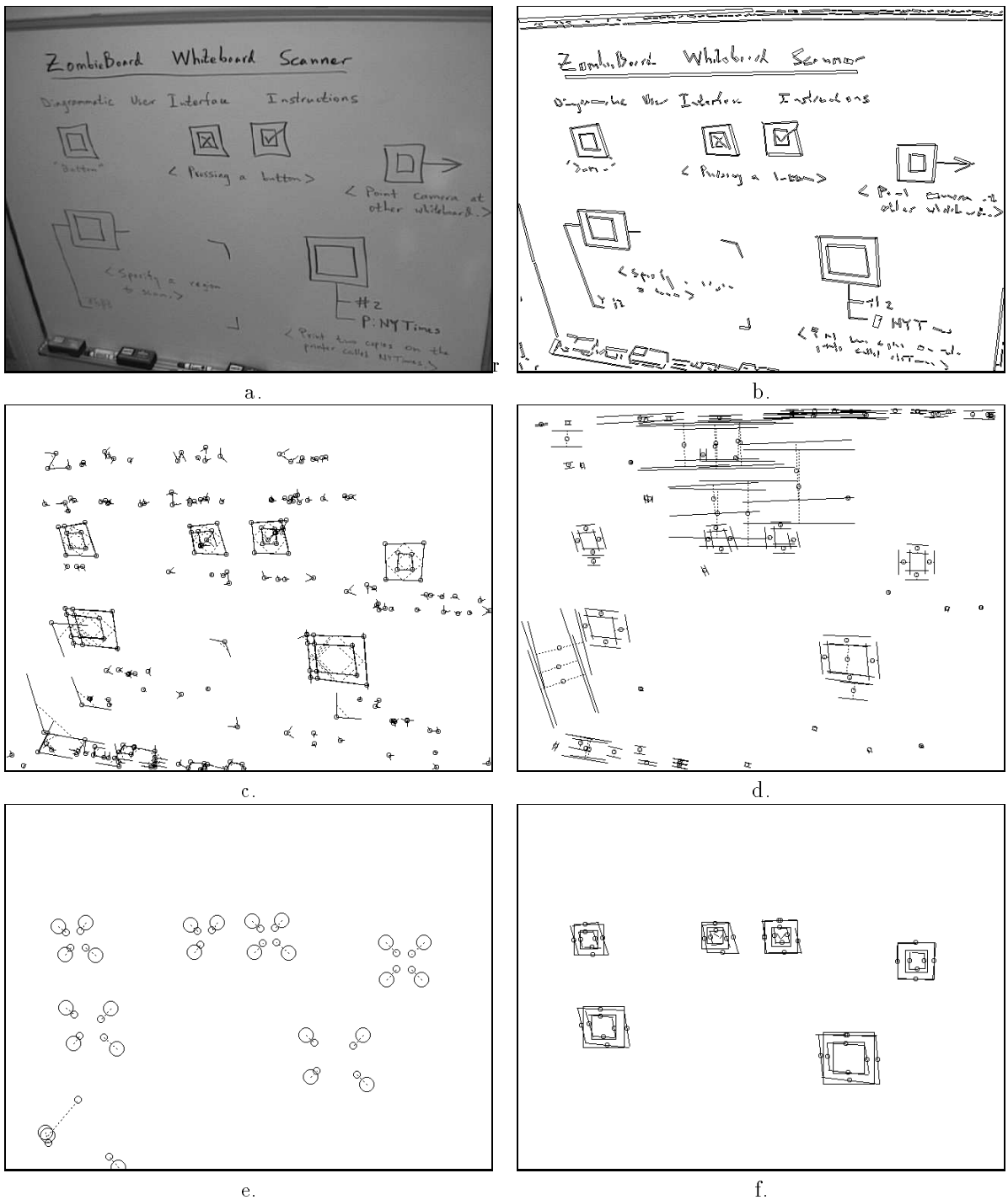

Figure 5: a. 640x480 input image. b. Bounding boxes of curve fragments. c. corners. d. parallels. e. nested corners. f. Nested Box Buttons

References

[1] Anandan

[2] copyboards

[3] Irani-etal

[4] Mundy and Zisserman

[5] Rawley et al

[6] Saund curvilinear grouping

[7] Saund salient arcs

[8] Saund ZB TR

[9] Saund and Moran

[10] Smartboard

[11] Smartboard

[12] Stafford-Fraser

[13] Szeliski

[14] Tegrity

[15] VideoBrush

[16] Weiser and Brown