THE LIVE PAPER SYSTEM

CHARLES ROBERTSON

Your file Votre référence
ISBN: 0-612-99040-0
Our file Notre référence
ISBN: 0-612-99040-0

THE LIVE PAPER SYSTEM

BY

CHARLES ROBERTSON, B. ENG.

MAY 2004

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
ELECTRICAL ENGINEERING

FACULTY OF ENGINEERING AND APPLIED SCIENCE
MEMORIAL UNIVERSITY OF NEWFOUNDLAND
ST. JOHN'S, NEWFOUNDLAND AND LABRADOR

CANADA

ABSTRACT

For the past 30 years, most interactions with computers have been through monitors, keyboards, and more recently mice. However, with ever increasing and more ubiquitous computing power, it is important to research other interaction modalities that are simultaneously richer and easier to use. By using computers to augment objects with illusionary intelligence and capabilities, it is possible for people to use these objects in new and exciting ways. Video cameras allow computers to monitor an environment non-intrusively, and react when appropriate.

This thesis presents a realized Video Augmented Environment, called *Live Paper*, for enhancing paper documents on a tabletop. To the user, it appears as if the paper gains new visual and auditory features. A piece of paper can be removed and returned to the tabletop, and it will regain the same features it previously exhibited. A data projector, connected to the computer, projects annotations onto the paper and the tabletop. The computer uses a video camera, mounted next to the projector, to capture images. The system analyzes the images to determine page locations, orientations, and identifications. Thus the projected annotations appear locked to the page, moving and rotating as the user moves and rotates the page.

Live Paper provides several page-based applications, called *transparencies*, including a 3D architecture visualizer, a music player, and a collaboration tool. All transparencies use a consistent set of projected buttons, with which the user interacts using his or her finger.

The page-finding algorithms extract features from the captured images by segmenting (using Otsu grey-level thresholding) and analyzing the boundaries (using Freeman-Davis corner finding). The feature analysis stage examines these boundaries to find the pages. The thesis also presents the successful analysis of overlapping and occluded pages, based on a new application of perceptual occlusion techniques.

To identify paper, *Live Paper* uses a novel method that does not require fiducials, works in non-ideal conditions, and uses low resolution video images. The system determines the similarity of found and stored pages using a modified Hausdorff Distance. The thesis presents a comparison of the Hausdorff measure with a Euclidean measure, using a set of sample pages including hand-written pages, presentations, journals, and paintings.

The thesis reviews the transformations used to synchronize the locations from the captured image with the internal models and the projected workspace. The system design section includes details on how the hardware and software modules interact, and how programmers can add additional transparencies to the system.

The *Live Paper* system successfully demonstrates a unique augmented environment where ordinary paper is the interface with the computer. This thesis lays a foundation for future research and development of a general purpose augmented tabletop.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. John Robinson, for his support and guidance throughout my graduate studies, and for the time that he spent reviewing my thesis.

I am appreciative for the wonderful colleagues at the Multimedia Communications Lab, and later the Computer Engineering Research Lab. I want to acknowledge Li-Te Cheng, Rahim Pira, Reza Shahidi, and Li Cheng for their support and assistance. I want to especially acknowledge Li-Te Cheng for his excellent MCLGallery library.

A note of thanks to all my close friends - Douglas and Khristie Howse, Derek Butler, and Joshua and Shirley Swamidas. Your encouragement over the years is greatly appreciated.

I would like to thank my colleagues at Verafin Inc. for their support in granting me time off work so that I could finish my studies. I would also like to thank NSERC, Dr. John Robinson, the Faculty of Engineering and Applied Science, and the School of Graduate Studies for financial assistance during my programme.

I want to dedicate this thesis to my Mom and Dad, whose complete and unwavering confidence and encouragement allowed me to finish.

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

LIST OF SYMBOLS & ABBREVIATIONS

COMMONLY USED

3D	Three Dimensional
ANSI	American National Standards Institute
API	Application Programming (or Programmer's) Interface
dpi	dots per inch
cm	Centimetre
CRT	Cathode Ray Tube
DUI	Diagramatic User Interface
FIR	Finite Impulse Response (filter)
fps	frames per second
GUI	Graphical User Interface
I/O	Input/Output
IACF	Inter-Application Communication Facilitator
IGF	Inverse Goodness-of-Fit
IIR	Infinite Impulse Response (filter)
KB	Kilobyte (2^{10} bytes)
LCD	Liquid Crystal Display

LED	Light Emitting Diode
MCL	Multimedia Communications Laboratory
MB	Megabyte (2^{20} bytes)
MHD	Modified Hausdorff Distance (also $h_M(A,B)$)
MHz	Megahertz
mm	millimetre
NTSC	National Television Systems Committee
O(*n*)	Order of *n* (Big O Notation)
OCR	Optical Character Recognition
PC	Personal Computer
RAM	Random Access Memory
RGB	Red-Green-Blue
UI	User Interface
URL	Uniform Resource Locator
VAE	Video Augmented Environment
XGA	Extended Graphics Array
XML	Extensible Markup Language
u[n]	Discrete-time unit step function
δ[n]	Discrete-time unit impulse function

SYMBOLS FOR OTSU THRESHOLD DETERMINATION

L	The number of grey levels
k	The number of classes

t_n	Threshold at grey level n
C_K	Class K of grey level distribution
n_i	The number of pixels with intensity i
N	Total number of pixels
p_i	Normalized value at intensity i
λ, κ, η	Class separability measures
σ_B^2	Between-Class Variance
σ_W^2	Within-Class Variance
σ_T^2	Total Variance
ω_K	Zeroth-order Moment for class C_K
$\omega(x)$	Zeroth-order Cumulative Moment
μ_K	First-order Moment for class C_K
$\mu(x)$	First-order Cumulative Moment (mean)
μ_T	Total first-order moment (mean grey level of image)
T	Final grey level threshold for *Live Paper* segmentation

SYMBOLS FOR BOUNDARY FEATURE DETECTION

s	The number of consecutive links terminating at a node
a_j	The link pointing to the j^{th} point
L_j^s	The line segment terminating with link a_j
X_j^s	x component of line segment L_j^s
Y_j^s	y component of line segment L_j^s

l_j^s	Length of line segment L_j^s
θ_j^s	Local curvature of line segment L_j^s
δ_j^s	Incremental curvature of line segment L_j^s
K_j	Cornerity at point j

Symbols for Moving Averages Algorithm

P	Grey level image
p_n	Pixel n in P
T	Thresholded (output) image
t_n	Pixel in T corresponding with p_n
t	Threshold value (0 – 100)
g_n	Approximation to running average for p_n
h_n	Average of current running average and average on previous row
s	Length of approximated running average

Symbols for Hausdorff Distance

$H(A,B)$	Hausdorff Distance between sets A and B
$h(A,B)$	Directed Hausdorff Distance from set A to set B
$d(a,B)$	Minimum distance between point a and all points in set B
$\|\|a - b\|\|$	Distance between points a and b
$L_1(a,b)$	L_1-Norm between points a and b

$L_2(a,b)$	L_2-Norm between points *a* and *b*
$L_{infinity}(a,b)$	$L_{infinity}$-Norm between points *a* and *b*
$h_M(A,B)$	Modified Hausdorff Distance between sets *A* and *B*

SYMBOLS FOR REGISTRATION

(u,v)	Coordinate before transformation
(x,y)	Final corresponding coordinate after transformation
(u_C,v_C)	Centre of radial distortion
r	Radial distance from coordinate to centre of radial distortion
r'	Corresponding radial distance after transformation
C	Correction constant
C'	Inverse correction constant

Chapter 1

INTRODUCTION

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

Mark Weiser, "The Computer for the 21st Century"

1.1 MOTIVATION

For most people, the focus of interaction with a computer is its monitor, its keyboard, and its mouse. They use a graphical user interface that hasn't fundamentally changed for over twenty years [Myers 1998]. However, by simply dropping their eyes, they leave the realm of computers and arrive in the real world. With the ever-increasing computational power available, why hasn't computer interactivity moved beyond the virtual desktop to the real tabletop?

It is compelling to think of a general augmented environment for the home or office. Instead of moving to a specific interface to use the computer, a person could interact with a computer that would actively respond to his or her needs. Traditional graphical user interfaces are insufficient to this vision. It would create too much clutter to start placing keyboards, mice, and screens throughout the environment. Other technologies such as touch screens or remote controls are more reasonable, but they still do not provide easy access to all the power that computers can provide.

What if we could point at an object to indicate a request to the computer? What if we could verbally ask the computer about an object we are holding in our hands? What if the computer could respond by displaying information directly on the object of our focus? A comprehensive and perfect implementation of this vision is still in the realm of science fiction, and is likely to remain such for some time. However, the technology exists today to start creating this vision.

There are two possible ways, which are not necessarily exclusive, that technology can evolve over the next decade in order to provide an experience akin to "ubiquitous computing", a phrased coined by Mark Weiser [Weiser 1991]. The most likely prospect is that computer power will be embedded in ordinary objects, and cost little more than objects without this power. We will be able to buy pens that track their motion and store their writing or send it to other devices [OTM 2003]. With cheap, flexible displays in abundance, we will buy mugs that show animated pictures as well as the temperature of the beverage within – similar to the animated cereal box advertising in the movie "Minority Report" [Spielberg 2002]. Furniture will change patterns to provide different moods for different functions. Objects will become smarter about themselves, will be able to tell us about their state, and will be able to talk to each other to provide sophisticated emergent behaviours.

The other possibility is less likely to happen in the near future, but ultimately is more compelling. Instead of providing certain objects with embedded intelligence, we will be able to augment any ordinary object with illusionary intelligence. This means that although the computing power will not reside within the object, it will seem as if it does. The object is still the interface to the computer. It will not matter to the user that the intelligence is not in the object. The user will still have access to the writing associated with a pen, and still be able to see the temperature of a mug. To create such an illusion, the computer must be linked to sensors distributed throughout the environment.

One advantage of illusionary systems over embedded systems is that we will not have to replace our existing objects. We can continue to use our favourite pen – in fact, we can use any pen that happens to be handy, as the system could easily associate the writing with the user, and not the particular object. It will also be easier to add new capabilities – we don't need to replace our mug because it won't become obsolete. These capabilities can be more computer-intensive. And it will be easier to add new objects – with a software upgrade we could use a coffee table to read and compose email.

There are some disadvantages of illusionary intelligence. The zone of augmentation is fixed, either to a person (augmented reality) or a place (augmented environments). Unlike embedded intelligence, the object cannot be taken out of the zone and still function. Illusionary intelligence might also miss some of the subtleties of user interaction.

Although sensitive sensors and more powerful algorithms would help, there will always be situations where embedded sensors will be better. Finally, the initial cost of such a system would be expensive compared to small groups of embedded-intelligence objects.

As mentioned, there are two principle ways to define a zone of illusionary intelligence. The zone could be personal, as in the case of augmented reality. Such a system would have only one point of view, which would be tied to the point of view of the user. The number of input devices would be limited to what the user could carry. The practical quality of these devices would likely be poor in terms of robustness and subtleness as the system would have to compensate for the motion of the user. There are many situations where the portability of augmented reality would be crucial, but in general such a system would be inferior to a permanent environment.

The alternative zone is an augmented environment, which would be fixed in a specific area. It could be in a room of a house, an office, a boardroom, or a shared public space. Because it would be permanent with more mountings available, the augmented environment could use a variety of input devices such as microphones, video cameras, infrared cameras, touch pads and screens, and switches. Some devices, such as video cameras, would passively gather user input. The devices would be scattered throughout the environment to provide multiple points-of-view. The system could also use a variety of output devices, ranging from traditional displays to movable projectors, lasers, haptics, and speakers. An augmented environment would be better able to determine how to respond to the user.

However, for an augmented environment to be truly useful, the design must recognize that people want to use the functionality provided and not to merely interact with the environment. The focus must be on augmenting objects within the environment. To that end, there are many research questions about designing and implementing augmented environments.

There are many user interface questions. How would someone know if an object was augmented? How would someone understand what enhancements were available? How would someone control the enhancements? How would someone know if his or her command was understood and acted upon?

There are system design questions. How should the system bind enhancements to objects? Should all objects of the same type be treated identically, or should individual objects be treated as unique entities. How should the architecture be structured? How can new functionality be added? What algorithms for input processing need to be designed?

In the Live Paper project, the research investigates a number of these issues by taking a potentially compelling augmented environment and discerning the most important technology and interface issues. To that end, the environment is that of an ordinary office desktop where paper documents serve as the interactive augmented objects. For input device, a video camera provides a flexible, low cost, and non-intrusive solution. For output, a standard data projector is also very flexible, and can display onto the user's focus. Using one input and one output device simplifies the system architecture and reduces the necessary computational power, while still providing a basis to answer some of the questions. Many of the solutions the research develops are useful in augmented environments other than *Live Paper*.

Although several researchers have considered the issue of desk-based video augmented environments [Wellner 1993a] [Kobayashi 1998] [Underkoffler 1998], they have emphasized creating desktops that provide new features and modes of interaction. In *Live Paper*, the system is a means to augment objects on the tabletop, rather than the table itself. A system that could find, identify, and track pieces of paper would provide the user with a strong illusion that the paper itself is augmented. Illuminating the paper with spatially registered computer generated projections would complete the illusion. This is the concept of *Live Paper*.

1.2 RESEARCH OBJECTIVES

The objective of this research is to create a video augmented tabletop system that uses image processing to find, identify, and track paper, while providing a consistent projected user interface.

The Motivation section, after first introducing the vision of a general augmented environment, presents the practical tabletop system that forms the basis of this research. Because the focus of *Live Paper* is on paper, the research investigates what image processing algorithms are needed to separate writing surfaces from the tabletop. This task is difficult because the lighting might not be consistent, and the page might be partially obscured by the user's hand. There might also be clutter, such as mugs, rulers, and pens, which the system must ignore.

The research also investigates what algorithms are needed to reliably identify the located pieces of paper. In order for *Live Paper* to be generally applicable, the identification method does not use explicit and unique marks (as per [Robinson1997] and [Kobayashi 1998]). Furthermore, the research determines a suitable scheme for registering the projections, the camera, and the tabletop.

Finally, the research considers what graphic user interfaces are appropriate to a tabletop environment. *Live Paper* must project augmentations onto physical pages such that the user can readily comprehend what is content and what is interface. The research also investigates a suitable technique for a user to interact with interface elements such as buttons. All of these developments must fit within a practical framework of software components that can provide a platform for future research on *Live Paper*.

1.3 CONTRIBUTIONS

With the *Live Paper* research, I have created an innovative and usable tabletop augmented environment for paper documents. This thesis presents innovations in three areas of augmented environments.

1.3.1 Finding Paper Pages

I introduce a robust framework for finding pages on a desk, where I integrated existing algorithms and then extended them to better match this application. Extensions include the use of an optimized search pattern to minimize false edges at page boundaries. The framework uses a standard video camera, works under ordinary office lighting, and uses inter-frame information to track pages. I also present work on the use of perceptual methods to find overlapping and occluded pages, including systematic testing and results from real-world tests.

1.3.2 Identifying Paper Pages

In this thesis I also present a framework of algorithms for identifying pages using its distribution of markings. Unlike existing systems, *Live Paper* does not rely on pre-rendered fiducials (identifying marks). I make a unique use of the Hausdorff distance to measure the similarity between observed and stored pages. *Live Paper* is the first system to recognize paper documents on a tabletop using the markings themselves.

1.3.3 Implementation

The user interface should help the user to focus on the objects (pieces of paper) on the tabletop, and not the tabletop itself. In *Live Paper*, objects gain abilities while they are within the augmented zone. This leads to new concepts such as the *transport*, which is a visual bridge between the virtual domain and the physical domain. I also introduce *transparencies*, which overlay the physical object with new abilities and have a visual anchor in the transport. The user interacts with his or her finger to select buttons using a click-less mechanism that is entirely based on image processing. The mechanism uses a new method to prevent false selections. Finally, I present a flexible software architecture that acts like an operating system for the augmented environment.

1.4 THESIS ORGANIZATION

The eight chapters of this thesis are organized as follows:

- Chapter 1, Introduction, gives the motivation behind the development of *Live Paper*, and the problem statement and objectives for research.

- Chapter 2, Review of Video-Augmented Environments, reviews existing projects and products that are related to *Live Paper*. These systems include augmented desktop, interactive whiteboards, and paper-based projects.

- Chapter 3, The Design of *Live Paper*, presents the vision of the project and then explains the design principles. The chapter concludes by giving examples of how the final system accomplishes them.

- Chapter 4, Finding Paper on a Desktop, presents a number of strategies used by *Live Paper* to locate and then track pages on the tabletop. The experiments show the success of the strategies at finding overlapping and occluded pages.

- Chapter 5, Identifying Paper, presents the procedure for determining the identity of a page from its contents, without the need for fiducials. Experimental evidence demonstrates that the method is successful in a VAE.

- Chapter 6, Registration, summarizes the transformations for converting between locations in the captured images, the tabletop, and the projector display.

- Chapter 7, System Design, gives information about the technical architecture of *Live Paper*, including the hardware. The software section presents the interfaces for each module, the class structure, and creating *Live Paper* applications.

- Chapter 8, Contributions, presents conclusions about the project and recommendations for future work.

Chapter 2

Review of Video-Augmented Environments

The potential of augmented environments has influenced a number of research projects and commercial products. This chapter presents technologies that fit within an augmented office environment, usually where the user is required to perform writing of some kind. Most of the projects use video cameras as input, and thus can be broadly categorized as Video-Augmented Environments (VAEs) [Stafford-Fraser 1996b]. These projects have influenced the design and the research of *Live Paper*.

2.1 Whiteboards

The whiteboard is a versatile tool found in offices and educational environments. It is similar to a chalkboard, but provides a higher contrast surface for writing. A number of researchers and companies have investigated means to add storage capabilities and interactivity to whiteboards.

2.1.1 VideoBrush Whiteboard

The *VideoBrush Whiteboard* [VideoBrush 1997] was a commercial product that allowed a user to sweep a computer-connected video camera across a whiteboard and generate a high-resolution, colour mosaic of the contents. This mosaic could then be saved and printed or shared electronically. The system did not interpret the written information. It is no longer in production.

The product is representative of very basic video augmentation, which requires a high degree of user-involvement to transfer data to the computer. The other whiteboard projects in this section automate the input process.

2.1.2 BrightBoard

The *BrightBoard* [Stafford-Fraser 1996a] uses a video camera to add digital recording and some interactivity to a standard whiteboard. A computer watches the board via a video camera, and waits for the person to write special symbols. Figure 2-1 shows symbols used to print an area of the whiteboard.

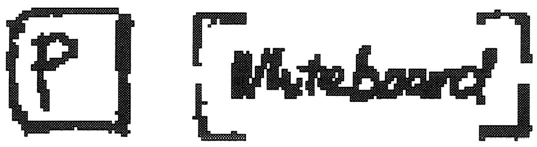


Figure 2-1: Example of *BrightBoard* symbols

(from [Stafford-Fraser 1996a])

Areas of the board can be saved, printed, faxed, or emailed; the commands are written directly on the whiteboard. Thus the user can remain focused on the board at all times. The *BrightBoard* generates a sound to indicate processing of a command. Each command is a pre-programmed sequence of letters and numbers inside a box. To associate a command with a particular area, the user denotes the region with corner symbols.

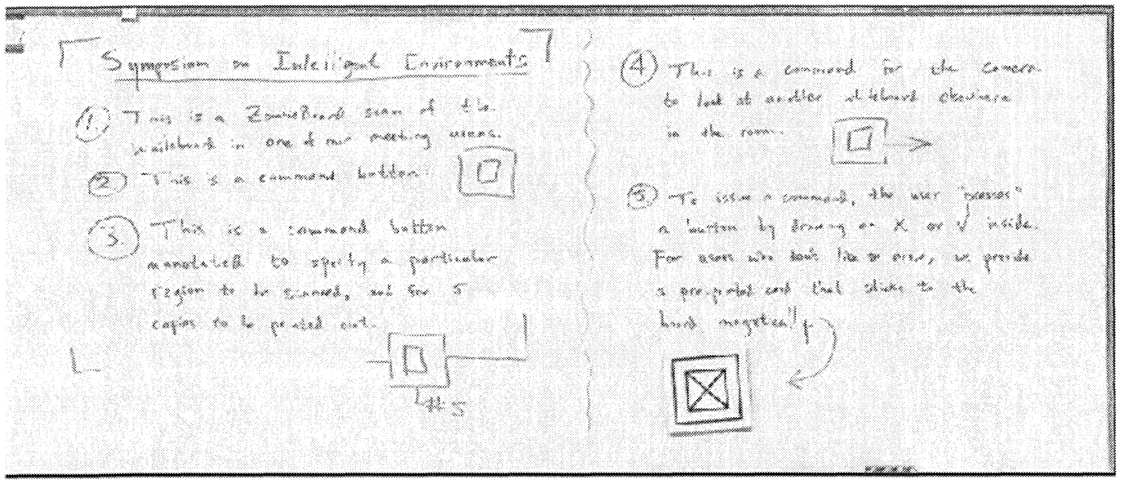
The *BrightBoard* operation is fast, but not quite real-time; images are captured at about 2 frames per second (fps). The system waits for a trigger indicating that the current whiteboard image should be processed. Once the trigger is tripped, a high-resolution grey-level image is processed for commands. The BrightBoard requires 4.5 seconds on a SPARCstation 2 to complete processing.

The trigger is a software-based motion detection algorithm that is tripped when there is no movement. This indicates that the user has stepped out of the camera's field of view. The motion detection algorithm uses low-resolution images of 40x30 pixels. Thus the processing requirements are quite low until a scene is analyzed.

The *BrightBoard* thresholds the high-resolution image of the board using a *moving-averages* algorithm (see Section 5.3.1 for details). The algorithm separates the darker writing on the board from the background. The algorithm is specifically designed to compensate for any gradients imposed by uneven lighting.

2.1.3 ZombieBoard

The *ZombieBoard* [Saund 1998] [Black 1998] is a whiteboard scanner developed at the Xerox Palo Alto Research Center. The system captures high-resolution images of the whiteboard, analyzes gestures, and provides a Diagrammatic User Interface (DUI) – see Figure 2-2 for a sample scan of a whiteboard.



Symposium on Intelligent Environments

(1) This is a ZombieBoard scan of the whiteboard in one of our meeting areas.

(2) This is a command button.

(3) This is a command button annotated to specify a particular region to be scanned, and for 5 copies to be printed out.

#5

(4) This is a command for the camera to look at another whiteboard elsewhere in the room.

(5) To issue a command, the user "presses" a button by drawing an X or √ inside. For users who don't like to draw, we provide a pre-printed card that sticks to the board magnetically.

Figure 2-2: *ZombieBoard* sample scan

(from [Saund 1998])

To obtain high-resolution images, the *ZombieBoard* mosaics low-resolutions images captured from a pan/tilt NTSC video camera. The camera scans the whiteboard such that each captured frame is only a portion of the entire board. For an 8-foot by 4-foot whiteboard, a total of 14 frames are used. Because the computer can control the pan and tilt angles of the camera, it is easy to estimate the mosaicing parameters. However, since it is difficult to accurately register the captured images to the whiteboard, the system uses features in the overlapping images to refine the generated mosaic.

The *ZombieBoard* analyzes gestures that a user performs with a 'phicon', or physical icon. The phicon has a distinct colour that can be easily found in a video image. The system recognizes a set of six gestures: start, cut, print, save, clear, and quit.

The DUI allows users to draw buttons and commands on the board (similar to the *BrightBoard*). The architecture is more flexible and extensible than the one used in the *BrightBoard* system. For example, the user may select any closed polygonal region in the *ZombieBoard* system, as compared to a rectangular region in the *BrightBoard*.

2.1.4 Tegrity Digital Flipchart

Another commercial product that uses a whiteboard is the *Tegrity Digital Flipchart* [Tegrity 1998]. The Tegrity system allows annotations written on a projected computer-based presentation to be captured. It also allows for control of the presentation from the board itself – the user can 'press' projected buttons.

The system (see Figure 2-3) consists of a standard whiteboard, a data projector, a video camera, and a host computer. The whiteboard, projector, and camera must be initially calibrated. The system projects onto the whiteboard several control buttons. The user moves so that his or her body does not obstruct the view of the camera, and uses his or her finger to select the buttons. Visual feedback indicates to the user that the button was successfully selected.

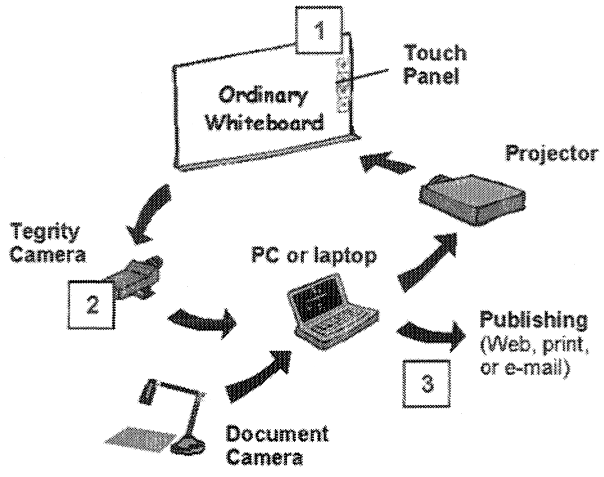


Figure 2-3: System diagram for the *Tegrity Digital Flipchart*

(from [Tegrity 1998])

One of the command buttons allows the user to capture any markings on the whiteboard. The projected image is blanked during the capturing process, which takes about a second. The captured image is merged with the original presentation slides and stored.

By 2002, the software was incorporated into the *WebLearner* product, and the interactive elements de-emphasized. Instead, the software provided a focus on capturing presentations for later playback.

2.1.5 Magic Board and Magic Table

The Magic Board project [Crowley 2000] combines an ordinary whiteboard with a video projector and pan/tilt camera. Instead of continuously capturing physical markings, it only extracts them during operations such as copying or saving. A user can select a drawing with his finger, digitize it, and then move the digitized version about the whiteboard.

The Magic Table applies the Magic Board technology to a horizontal whiteboard surface [Bérard 2003]. Multiple people can use circular tokens to quickly select written areas for digitization, and then move, rotate, and zoom the digitized ink.

2.1.6 Smart Classroom

The Smart Classroom [Shi 2003] merges several technologies to allow remotely located students to watch and listen to a teacher in a classroom with local students. The room contains several fixed cameras and video projectors. The teacher uses the Media Board, a touch sensitive video augmented whiteboard, to display prepared slides and write notes. Remote students can see the contents of the Media Board on their computer screens. The Smart Cameraman software automatically selects among several cameras to give the remote students the best view of the room. The Media Board also allows gesture recognition, as captured by the video cameras.

On the side of the room is the Student Board, which displays all of the remote students. The teacher interacts with this board by gesturing with a laser pointer to select students and permit them to ask questions.

2.1.7 ClearBoard

The ClearBoard project uses the metaphor of writing on a glass window while facing a collaborator. In the ClearBoard-1 [Ishii 1992], the user faces an angled half-mirror on which he writes with a color fluorescent marker. A video camera above the mirror records the user's reflection and his drawings, and then sends the feed to a remote identical setup. Here a rear projector displays the image onto a half-mirror. To the remote user it appears as if the first user is behind the mirror and is writing on it. The reverse video feed happens as well, so that both users have the illusion of writing on the same glass window.

The ClearBoard-2 [Ishii 1994] overcomes a few issues in ClearBoard-1, where the users are unable to erase remote markings, and the session cannot be recorded. The ClearBoard-2 has transparent overlay sheets on the display so that users draw in a computer-based paint application with a digitizer pen. The illusion that the remote collaborator is behind the screen is maintained.

2.1.8 The Significance of Whiteboard Augmented Environments

The whiteboard projects demonstrate how video can be used to construct an interactive environment. The concept of writing commands in the *BrightBoard* and the *ZombieBoard* environments is interesting, but is not as useful in a paper environment. The user is unlikely to be comfortable placing marks on every piece of paper. However, issuing commands from the work surface, either by writing or by selecting a virtual touch panel as in the *Tegrity* product, is compelling.

Therefore, a projected touch panel was considered for use in *Live Paper*. The implementation has to be more sophisticated because the buttons in *Live Paper* can occur anywhere on the desktop, and thus are more susceptible to accidental selection. Section 3.3 describes the *Live Paper* variation on projected buttons.

2.2 Tabletops

The graphical user interface common on personal computers uses a desktop metaphor to create an easy to use environment. In many office spaces, the computer shares a physical desktop with other items such as books, notebooks, papers, and pens. Some research projects in augmented environments try to create a user interface for desktops, and in doing so add new capabilities to the objects already found there.

2.2.1 Marcel

Marcel [Newman 1992] is a prototype French-English translation assistant. The user sits at a desk with a document of French text. When the user wants a translation of a word, he or she points at the word with a cordless digital pen. A list of possible English translations for the word appears on the desk inside a projected window, which the user may move and remove.

The researchers designed the system for user testing. All documents are scanned at high resolution and converted to text before the session begins. The system uses low-resolution video images captured from a camera over the desk to determine which page is present. A digitizing tablet under the page allows the user to precisely point to words with the pen. After the user selects a word, *Marcel* determines the corresponding stored word and consults an online dictionary for a translation.

Marcel corrects for skew using a parallel element locating technique. To determine which page is present, *Marcel* uses a description of the page using line lengths, paragraph heights, and word breaks.

2.2.2 DigitalDesk

The *DigitalDesk* [Wellner 1993a] is a project developed by Pierre Wellner at Cambridge Computer Labs in collaboration with Rank Xerox EuroPARC. The *DigitalDesk* was the first real attempt to add general computer interactivity to an ordinary desktop, and has become an inspiration to many of the other video augmented desk systems mentioned in this document.

The system receives information from two cameras mounted over the desk. A microphone mounted under the desk detects when a user taps the desktop with his or her finger. Information is shown to the user via a mounted data projector, also over the desk. See Figure 2-4 for an early prototype of the *DigitalDesk.*

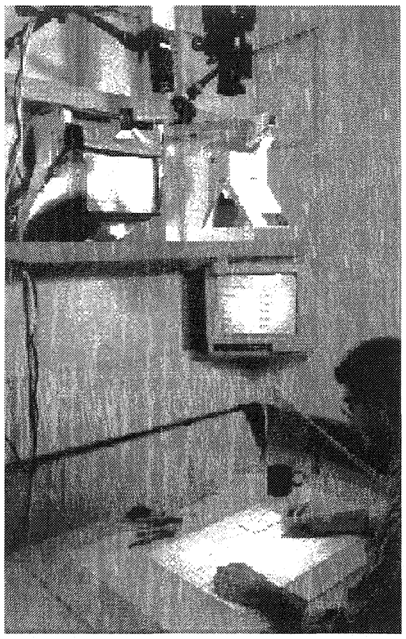


Figure 2-4: ***DigitalDesk*** **prototype**

(from [Wellner 1993a])

One of the cameras captures images at a very high resolution, and is focused on a relatively small area of the desktop. It captures numbers and other printed text, and passes the images to an OCR engine. Thus the user can enter numbers from a typed list by placing them under the high-resolution camera and pointing at them with a finger. The system also has a digitizing tablet on the desktop; the user may use a digitizing pen to select real and projected objects.

Wellner implemented several applications, including the *DigitalDesk Calculator* to enter numbers from a printed sheet, *PaperPaint* to combine print and projected media, and *DoubleDigitalDesk* for collaboration between remote users. The *Digital Drawing Board* is a closely related system, which has a tilted drawing surface.

2.2.3 Ariel (High Road Demonstrator)

A version of the *DigitalDesk*, called *Ariel*, was designed for annotating engineering drawings, and was part of the EuroCODE project [Mackay 1995]. The environment for *Ariel* is a distributed co-operative work place. As the engineers at a bridge building site update their drawings with minor changes, they must communicate their changes to each other. The system allows for an intelligent way to digitally store the changes without forcing the users to become acquainted with another computer system.

One input to the computer is via a pointer on the page. This pointer is a source of red light, such as an LED or laser beam. The system also has a video camera, a digitizing tablet, a bar-code reader, and a hand-held scanner. The result is a system which is reasonably natural to use, but which primarily accommodates the work habits of its users.

2.2.4 LightWorks and CamWorks

Both the *LightWorks* and *CamWorks* projects are video-scanning systems for printed documents [Black 1998]. The *LightWorks* system is based on the *DigitalDesk* – feedback is projected onto the printed material. The user selects the text to scan using a mouse.

CamWorks is a more fully developed system that is based on a simpler architecture [Taylor 1998]. This system also employs a video camera positioned over a desktop, but the selection of the text is performed within an on-screen video area (see Figure 2-5). *CamWorks* is able to automatically detect skew, column boundaries, and word boundaries. Thus users can select text on a word-by-word basis, even if the text on the page is skewed with respect to the camera capture area.

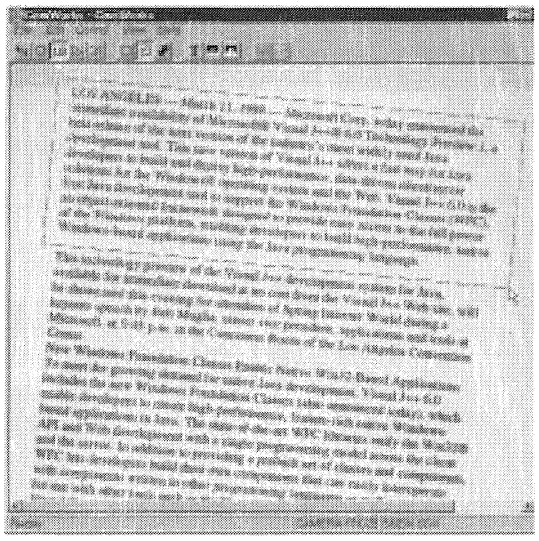

[illegible]

Figure 2-5: Skewed rectangular selection in *CamWorks*

(from [Taylor 1998])

2.2.5 Origami

The *Origami* project is a general research programme that is investigating video-augmented environments in the role of education. [Robinson 1997] The *DigitalDesk* acts as a base for the *Origami* project.

The user creates documents that contain both printed material, and material that will be projected. The printed document contains special marks for finding and recognizing the pages (see Figure 2-6). This includes a unique identifier printed in an optical character recognition (OCR) font. Once the system recognizes a printed page, the system consults a registry for information on rendering the active projections.

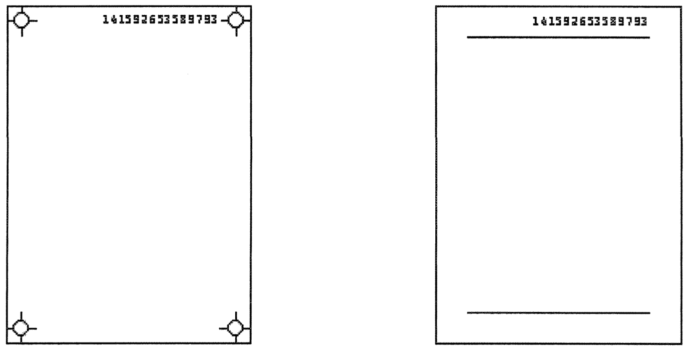


Figure 2-6: Special marks and text to identify pages in the *Origami* project (from [Robinson 1997])

For pointing, the user uses a pen with an LED in its tip. The light is captured by the video camera, and recognized by the system.

2.2.6 InteractiveDESK

The *InteractiveDESK* [Arai 1995] is a video-augmented desk that contains both a display on the desktop, with a corresponding pen-input system, and an ordinary upright display, with a keyboard. The focus is to assist the user in switching between electronic documents on both displays. The *InteractiveDESK* also allows for some links between real objects and electronic files. However, these objects each require a bright, uniquely coloured tag.

2.2.7 EnhancedDesk

The *EnhancedDesk* project [Kobayashi 1998] has a similar hardware configuration to the *DigitalDesk* (see Figure 2-7). The project automatically displays digital information linked to a page, and then allows the user to directly interact with the display. The information could be dynamic, such as a movie.

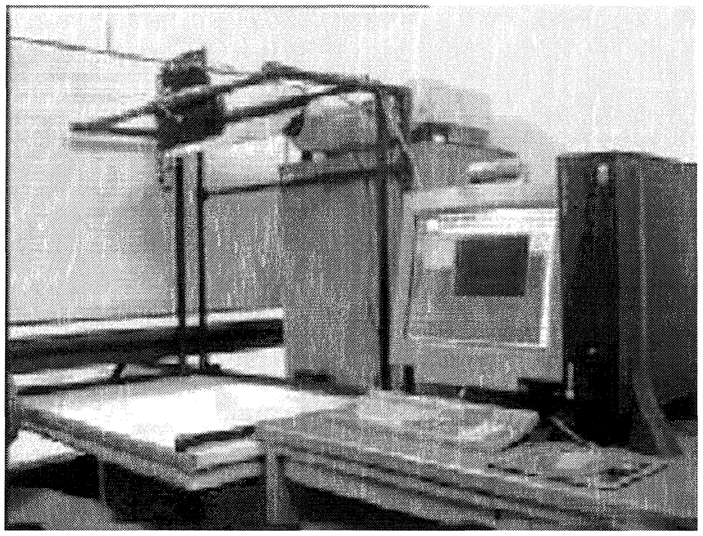


Figure 2-7: The *Enhanced Desk* setup

(from [Kobayashi 1998])

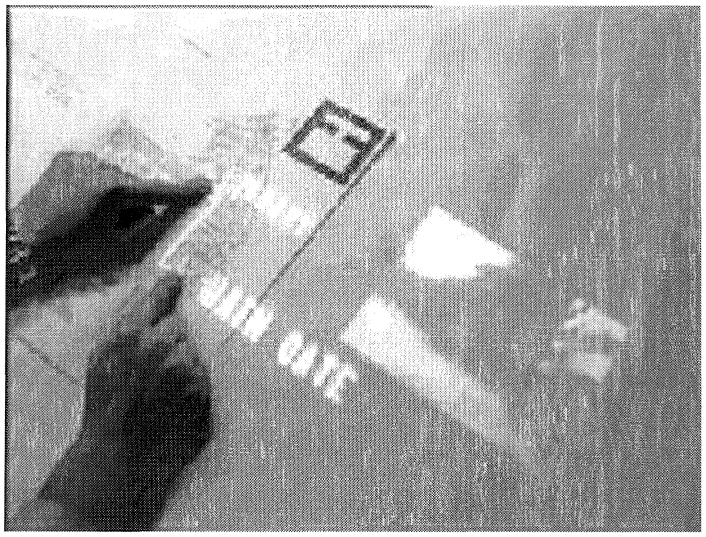


Figure 2-8: The geographical information system

(from [Kobayashi 1998])

The system uses a 5x5 grid of binary cells, called a matrix code, to identify each page. The page in Figure 2-8 contains such a grid. The matrix code is used to reference the stored physical characteristics of the page. This includes the page size and locations of regions that the user may select.

Users of the *EnhancedDesk* may use their fingers to interact with documents and projected data. The system's finger finding algorithm recognizes the location of the hand by background differentiation and then using skin colour to identify the hand. In later work by the authors, they used infrared cameras to track hand movements [Sato 2000]. Document 'hot spots' may be selected with a finger, and appropriate information is displayed.

The researchers have also examined the use of the system for education: in a sample application, they describe a Physics textbook containing matrix codes. When students come to a page with a matrix code, appropriate interactive applications are projected onto the desk - in this case, a spring with a weight. A student can change the position of the weight with his or her finger to see the effect on the motion of the spring.

2.2.8 metaDESK

The Tangible Media Group at the MIT Media labs has been working on a number of projects involving 'tangible user interfaces'. The *metaDESK* [Ullmer 1997] is an investigation into using physical objects to interact with a computer display. In addition to the desk, which incorporates a back-projected display, the *metaDESK* also includes two movable 'lenses'. Figure 2-9 has two pictures of the *metaDESK*.

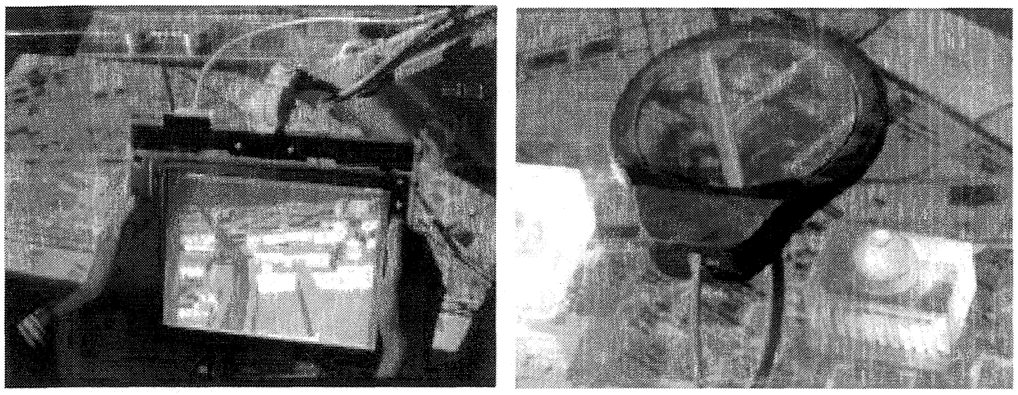


Figure 2-9: The *metaDESK*

(a) Example of active lens (b) The passive lens (from [Ullmer 1997])

The *metaDESK* uses a number of optical and mechanical sensing devices, such as infrared and standard video cameras, to determine the position and orientation of the physical objects. The lenses show additional information about the underlying display area. In the project, the *metaDESK* was used to display a map of the MIT campus. The active lens shows a 3D view of the campus buildings, and the passive lens shows an aerial photograph. To activate the MIT map, a representation of the 'Great Dome' building is placed onto the desk. The Great Dome is used as an anchor, and the map appears in relation to the dome. Another building may also be placed on the desk to scale and rotate the map.

The use of the buildings as physical icons, or phicons, is an interesting concept that has been extended by the group. For example, the *mediaBlocks* project [Ullmer 1998] uses blocks with some digital storage capabilities to store handles to media such as video, pictures, audio, and whiteboard data. These blocks can be inserted into browser devices to view the data, or in some cases dropped into a printer reader to produce a printout.

2.2.9 Illuminating Light

The MIT Tangible Media Group's project *Illuminating Light* [Underkoffler 1998] was also inspired in part the *DigitalDesk*. The project uses a flat desk surface, and introduces the concept of an I/O bulb. An I/O bulb is a grouping of a data projector and a coincident or near-coincident video camera.

An I/O bulb can be classified as one of three different scales. A large-scale I/O bulb is mounted on a ceiling in a computer-controlled gimbal system. This system can rotate to display and capture in different parts of the room. A medium-scale version is a mounted projector and camera. By having a fixed installation, a number of issues involving the relationships between real-world and projected elements become much easier. The small-scale I/O bulb's projector/camera combination is mounted on a Luxo-style device, which is a multi-segment movable arm found on desk lamps. The user moves the I/O bulb to illuminate an area with additional information. Sensors on the joints of the arm provide a computer with the position of the arm, which can then display the appropriate information.

The *Illuminating Light* project focuses on an application for teaching and simulating holography. Students can manipulate, on a tabletop, physical objects that represent common holography devices such as a laser, a beamsplitter, and a lens. The I/O bulb's camera captures the scene, and a computer analyzes the scene for coloured dots acting as fiducials. These dots identify the type and orientation of each object. The system then projects the path of the laser beam (see Figure 2-10). The objects behave appropriately – the laser appears to split at the beamsplitter, and diverges at the lens. When the holography set-up is correct, then a simulated hologram is projected at the output.

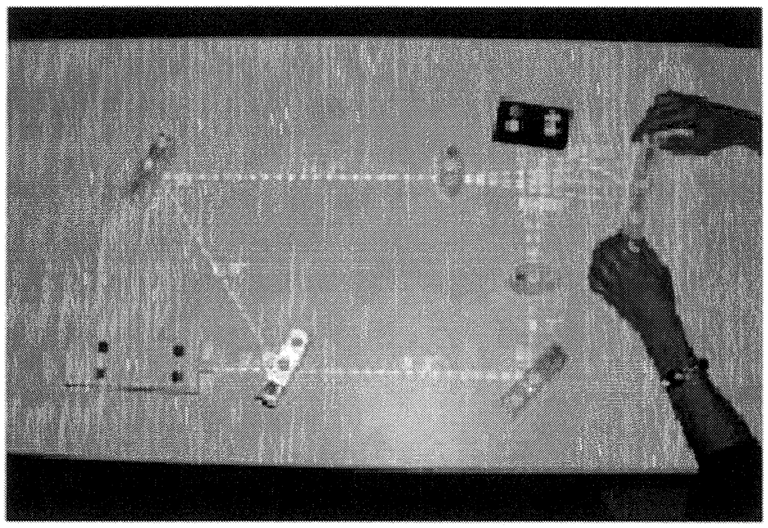


Figure 2-10: ***Illuminating Light*** **holography application**

(from [Underkoffler 1998])

The project is free-form; the objects do not need to be placed in precise locations on the table, nor do they have to be placed in a certain order. The environment is naturally collaborative – multiple students can move the objects simultaneously, with real-time feedback of the laser path. Distances between objects and the angles between reflected beams are displayed on the tabletop next to the laser beam in question.

The simulated laser light is projected as grey dotted lines that slowly progress in the direction that the light is travelling. The authors found that colour projections could confuse the algorithms to extract the coloured dots. Thus there is no processing done to remove the effect of projections in the display area.

Because the system uses coloured dots, the system does not need to do a detailed analysis of the scene. One computer performs both the image analysis, at a frame rate of 8 to 12 frames per second, and the holography simulation. The authors intend to build a version using template-matching to remove the reliance on coloured dots for object recognition.

2.2.10 Augmented Surfaces

The *Augmented Surfaces* project [Rekimoto 1999] provides laptop users with a shared desktop area where they can place media and other data. The system integrates multiple laptops, handhelds, an augmented tabletop (the *InfoTable*), and a display projected on a wall (the *InfoWall*). Figure 2-11 shows the system in use. When a user sits at the table with a laptop, the system uses printed matrix codes (as per [Rekimoto 1998]) on the laptop to identify it. By using the pointing device in the laptop, the user can drag media from their workspace to the tabletop or wall. Users can associate media to any physical object that is tagged with a matrix code; these objects can be removed and returned to the table with the data association intact.

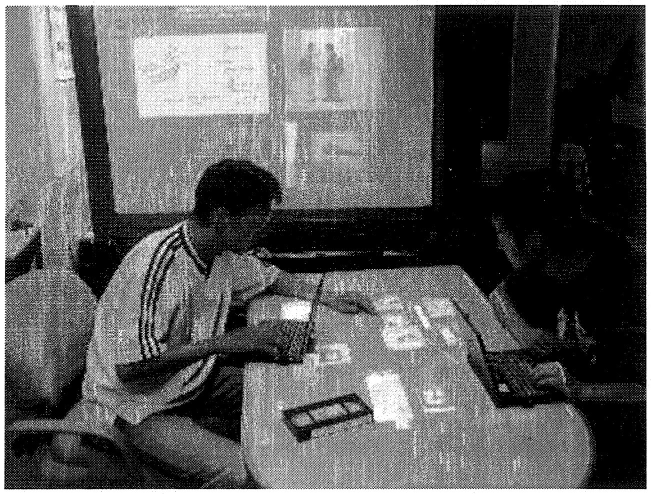


Figure 2-11: The *Augmented Surfaces* environment in use.

(from [Rekimoto 1999])

The *InfoTable* supports multiple users. A user can select media placed by the other user, and move it to his own laptop. When a cursor is on the table or wall, a visible line is drawn back to the laptop to show the association.

The system captures images of the workspace from a fixed camera and a pan/tilt camera. If the system detects motion from the low resolution fixed camera, then it uses the pan/tilt camera to find and read the matrix codes. The system does not sense hands or fingers.

The *SmartSkin* project [Rekimoto 2002], also by the same author, does sense the position of hands and fingers. *SmartSkin* is a capacitive sensing mesh that is built into a tabletop. As a hand approaches the table, the capacitances at the sensors change. This allows the system to locate the hand in two dimensions, and estimate the height above the desk, without any image processing. The system can differentiate between a finger, multiple fingers, and the entire hand. When the tabletop is augmented with a data projector (see Figure 2-12), the user can use various hand gestures to interact with the projected media.

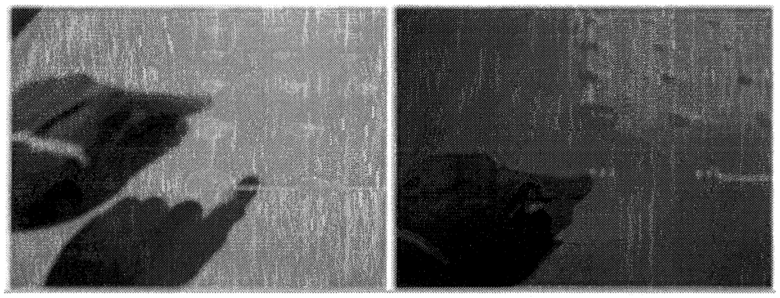


Figure 2-12: A *SmartSkin* application that shows the touch sensitivity of surface. (from [Rekimoto 2002])

2.2.11 Everywhere Displays Projector

The Everywhere Displays project [Pinhanez 2001][Pingali 2003] combines a steerable projector with a pan/tilt video camera to provide input and output capability throughout a room. The display can be placed on any surface within view of the projector, and is geometrically corrected for distortions due to oblique surfaces. Input regions are defined using an XML-based definition language, and an API allows Java applications to communicate with the system services. Users can select projected buttons and move sliders with their fingers.

The BlueSpace project [Lai 2002] is a general workplace environment that uses an Everywhere Display projector in conjunction with repositionable LCD monitors, active badges, and sensors for measuring ambient conditions. The Everywhere Display can project information on the cubicle wall, on the desk next to the user, or on a desktop for collaboration.

2.2.12 The Escritoire

The Escritoire [Ashdown 2003] is a digitizing desktop augmented with two overlapping projected displays. One covers most of the desk with a low-resolution display, while the other focuses on a small area with high-resolution. The two areas are calibrated so that the user can drag virtual paper across the different resolutions – bringing the documents into the high-resolution area for editing and annotating.

A similar concept is the Focus Plus Context Screen [Baudisch 2001]. The latter project uses an LCD screen for the high-resolution area and is arranged as a wall display. Input is via a standard keyboard and mouse.

2.2.13 The Significance of Tabletop Augmented Environments

The augmented environments based on a tabletop design encompass a wide range of approaches to the design, and a wide range of applications. The design of *Live Paper* has been affected by these projects, both by what they can do and by what they can not do.

For example, the *DigitalDesk* was the first augmented environment to explore a variety of tasks, and it provides the basic inspiration for the hardware set up of *Live Paper*. Both projects use a camera and a projector. However, *Live Paper* does not use a digitizing tablet, as this would not be compatible with an ordinary tabletop.

The image processing and other algorithms required for each project depends on the task and on the choice of input devices. For projects such as *Origami*, *EnhancedDesk*, and *Illuminating Light*, the researchers have devised ways to label objects such that they can be quickly detected and quickly identified. These techniques are useful in some contexts, but are not useful for *Live Paper* in the context of ordinary pages with markings.

Live Paper also differs from these projects in that *Live Paper* provides a framework for augmenting paper on a typical desktop, as opposed to a system that is limited to one or two predefined tasks. The existing projects might use a library of functions to develop certain task responses, but they do not strive to produce a common framework that can be extended for different applications.

2.3 PEN AND PAPER

Another common writing tool is the paper pad. It is lightweight, portable, and very flexible. Its can store a reasonable amount of writing, and its pages can be torn out. Several researchers have developed means to augment a paper pad. There have also been several research projects that looked at augmenting the capabilities of writing instruments such as pens, markers, and highlighters.

2.3.1 Digital Pens

The *VideoPen* [Arai 1997] is a pen that contains a retractable highlighter, a video camera, and a switch on the tip. The field of view of the NTSC video camera is 2 inches by 1.5

inches and includes the pen tip. The *VideoPen* is part of a system called *PaperLink*, which can associate written marks with electronic content, and can extract words from a printed page as input. The prototype pen is cumbersome, as the miniature camera is attached to the side of the pen, and requires a wire connection to a computer.

A consumer pen product developed by Anoto [Anoto 2003][Logitech 2003] uses an optical sensor to determine the pen's location on special paper. The pen records handwriting and uploads to a desktop computer when the pen is docked. OTM Technologies [OTM 2003] has developed a digital pen that uses a low power laser and an optical sensor to determine movement. Its pen does not require specialty paper, and communicates to a computer or other digital device in real-time using wireless Bluetooth technology [Bluetooth 2003].

2.3.2 Paper User Interface

The Paper User Interface [Johnson 1993] is a system where pages with identifying glyphs (distinct shapes) and standard markings (see Figure 2-13) are used to interact with a computer. A 'paper server' called XAX provides a framework for paper applications. A prototype document management system called *Protofile* allows users to store, distribute, and retrieve paper documents. The user presents a cover sheet to *Protofile* to initiate an action. To store pages, the user follows a cover sheet with the pages to store.

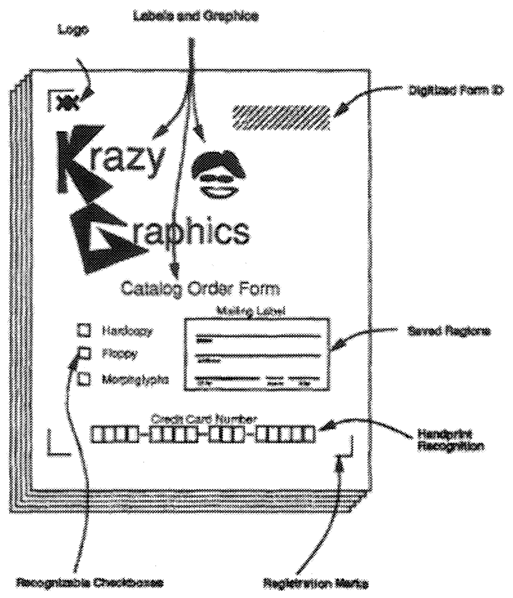


Figure 2-13: Cover sheet in the Paper User Interface

(from [Johnson 1993])

2.3.3 Video Telewriter

In the *Video Telewriter* system [Robertson 1997], a video camera views a stationary page. The system processes successive video images in order to create a low-datarate bitstream suitable for transmission to remote sites. The system extracts 'marks' from the image while ignoring the user's hand and pen.

In its first version, the system did not have any network connectivity, and did not attempt to differentiate pages. Whenever motion was detected, the system processed for marks. One requirement was that the user removes his or her hand from the image when not writing. Thus, when no motion was detected for a couple of seconds, an update mechanism reprocessed the image to find error marks. Although most extracted marks correspond to writing, some artefacts such as dark shadows and the pen tip could be erroneously stored as marks. The deletion of error marks also allowed for page changes – so long as the user paused after removing a page to allow the update mechanism to execute. However, the system did not store old pages.

In a later version, more functionality was added, including network capabilities and an improved computer interface. Users at remote sites could join a telewriting session and see what was being written on the paper page. A user could use the mouse to annotate his or her screen, and this was displayed on all users' screens. Instead of deleting all marks when a page was removed, a paging feature was added to detect page changes and store the old marks. Each user could then view older pages without affecting others.

2.3.4 Tele-Graffiti

Tele-Graffiti is a telewriting system that projects onto paper the real-time contents of a remote page [Takao 2003]. Two computers are linked via a high-bandwidth network, and a video camera at each end captures paper attached to a clipboard (see Figure 2-14). As one writes on the paper, a colleague at the other computer can interact with the writing on his or her own page. The projection tracks the page as one moves or rotates it. *Tele-Graffiti* makes full use of OpenGL [Shreiner 1999] for mapping remote images to the local page location before projecting the images.

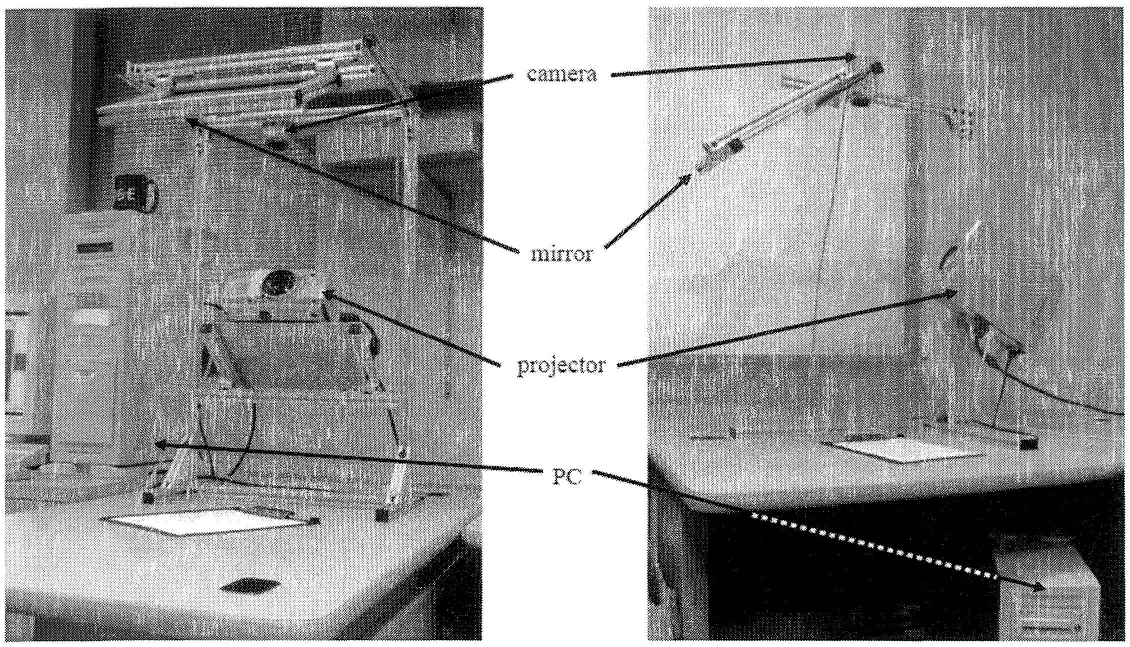


Figure 2-14: The Tele-Graffiti system hardware (from [Takao 2003])

Tele-Graffiti demonstrates robust paper tracking, but requires the page to be on a clipboard with known characteristics. The system uses these to determine the location and orientation of the page. The system cannot find more than one page, and does not keep a record of the page past the end of the session. To find the hand of the user, *Tele-Graffiti* uses background image subtraction from a reference image.

2.3.5 The Significance of Pen & Paper Augmented Environments

These projects show some potential applications of augmenting pen and paper, including hyper-linking, document storage and retrieval, and telewriting. The objects themselves provide direct access to these features. The user does not need to leave the object and go to a computer terminal, and thus interrupt the task.

Only *Tele-Graffiti* provides feedback to the user at the task location. With the augmented pens, a user experiences most interactivity when a display device is nearby. *Protofile* provides no interactive feedback at all. Both *Video Telewriter* and *Tele-Graffiti* illustrate a good application of video augmentation, but they have constraints of a single paper page for a single application. There is opportunity to advance the state of the art by providing an augmented environment for paper that allows for multiple simultaneous pages and applications on the same desktop.

2.4 THE SIGNIFICANCE OF LIVE PAPER

The projects in this chapter represent the state of the art with respect to augmented environments similar to *Live Paper*. The remainder of this thesis presents original research that advances the field, both in general terms and with algorithms suited for paper-centric tabletops. The research fuses new algorithms and concepts to the state of the art, and creates a new system.

Existing desktop systems either use fiducials to find, track, and identify objects (such as [Kobayashi 1998] and [Underkoffler 1998]), or make highly restrictive assumptions about the environment (as in [Takao 2003]). In an ordinary office environment, it is necessary to be able to identify paper based on its intrinsic properties. This thesis presents a unique framework of algorithms suitable for finding and identifying paper pages.

A new contribution is the application of a virtual touch panel to objects in a desktop environment. This touch panel, called a *transport* (see Section 3.3), expands on existing ideas (like [Tegrity 1998]) by visually tagging an object and being context specific. The input mechanism is also novel. Unlike mechanisms such as [Sato 2000], *Live Paper* uses a simple algorithm and confirmation to provide quick and robust selection by a finger. *Live Paper* also differs from previous projects by allowing the same type of objects (such as letter-size paper) to have different combinations of augmentations. One page might have an architectural rendering and be remotely viewable, while another is a remotely viewable telewriting page.

A final contribution is in treating the augmented environment framework as an operating system. The augmentations are similar to applications in traditional computer operating systems like Microsoft Windows or Unix. *Live Paper* provides low-level features for finding pages and associating applications to them, but the real power of the system lies in the applications themselves. Other projects have looked at single applications, or have looked at processing libraries, but none have viewed the desktop as a general augmented environment.

Chapter 3 presents the applications and user interface contributions in more detail. Chapters 4, 5, and 6 present the algorithms and experimental work for finding and identifying paper pages. Chapter 7 has an overview of the *Live Paper* architecture and describes the process for adding new capabilities.

Chapter 3

THE DESIGN OF *LIVE PAPER*

This chapter presents the basic principles behind the design of the *Live Paper* system, and gives examples of *Live Paper* applications. The requirements for these applications point to a number of research tasks, which the following chapters of this thesis investigate.

3.1 THE LIVE PAPER VISION

The vision for the *Live Paper* system is a computationally enhanced tabletop environment that augments ordinary pieces of paper. To a person (the user) sitting at the table, it will appear as if the paper gains new passive and active, visual and auditory features (see Figure 3-1). A piece of paper that is removed, but then returned to the desk, will regain the same features that it previously exhibited. The focus on paper as the interface is a subtle, yet key, difference from previous augmented environment projects. The desk is simply a tool by which the user adds enhancements to paper pages. To the user, the experience should be that of paper suddenly becoming 'alive' when laid on the desk.

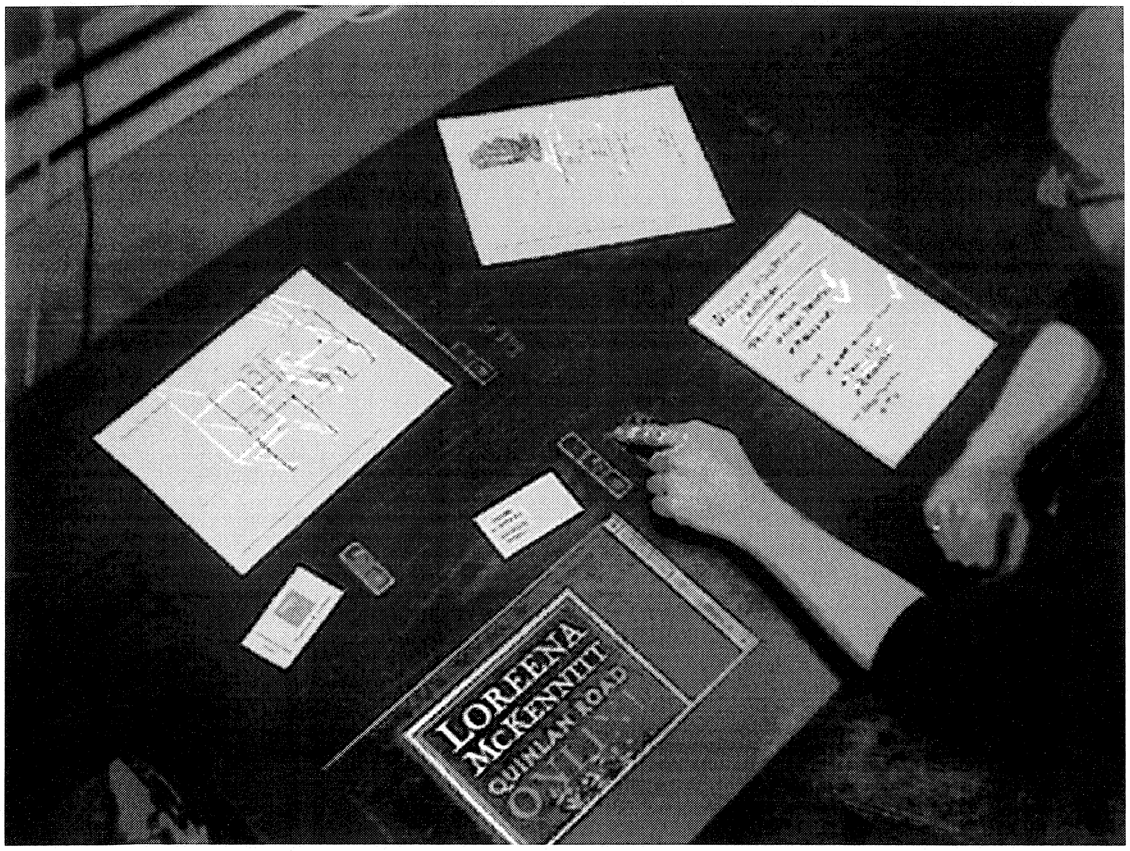


Figure 3-1: The *Live Paper* system in use

Figure 3-2 is a diagram of how the *Live Paper* desk looks from overhead; the user is located at the bottom of the diagram. Pieces of paper may be placed anywhere on the desktop, and may overlap each other. Projected annotations are displayed on both the pages and the desk surface. Some annotations are passive; although they would follow a moving page, the user would not be able to interact with them. Other projected elements act as buttons that the user can select. Smaller pieces of paper, like business cards, have information shown next to them or in a set region in the lower left-hand corner of the desk. The projector throw area is physically limited to the left two-thirds of the desktop, but pages laid outside are still recognized by the system.

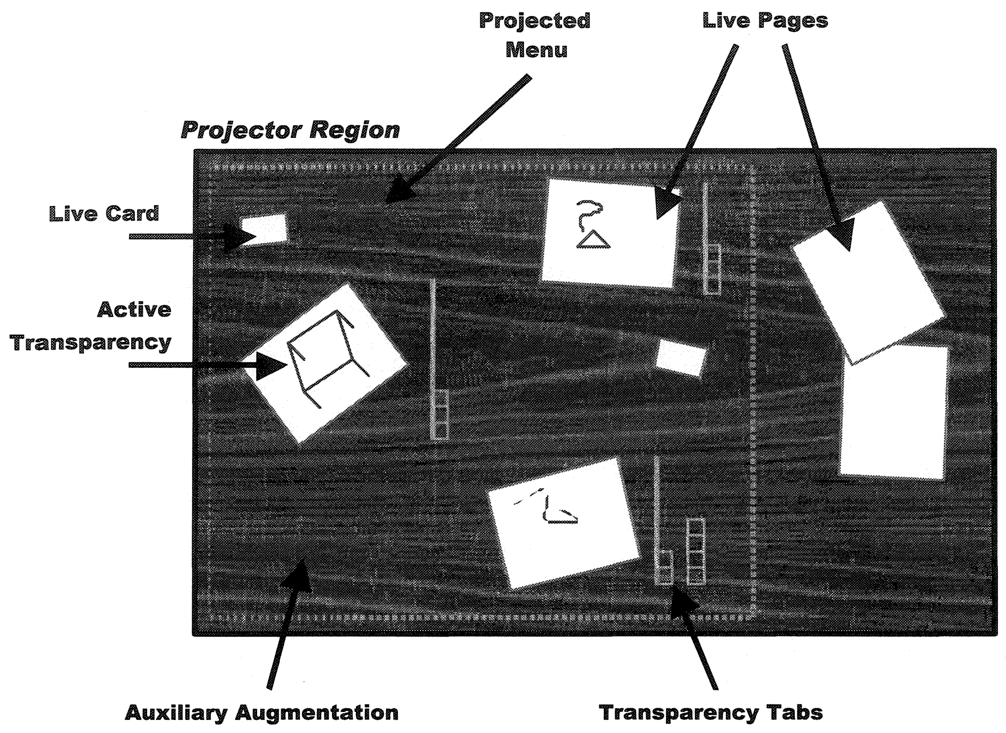


Figure 3-2: Implementation of the *Live Paper* system

The environment within which the user will work is a standard office desktop, but the system will use a ceiling mounted data projector to augment paper with video. The user will be able to write on pages using ordinary pens and to move or remove pages at will. The system will have collaborative abilities so that people at remote sites may view selected pages, even as the user writes on them. Information for a specific page will move with that page.

The final *Live Paper* system implements the vision by adhering to fundamental design principles, as identified in the next section. The system architecture that emerges from these principles helped to identify the research tasks that this thesis presents in subsequent chapters.

3.2 Design Principles

The *Live Paper* system investigates the virtual enhancement of ordinary objects with new properties and abilities. The system implementation follows general design principles that specify how these objects should be enhanced. The principles determine how the user interface behaves and how the system handles task interaction.

3.2.1 Zones of Augmentation

This research looked at objects without built-in enhancements; each object only gains enhanced properties in certain locations. These locations are termed *Zones of Augmentation*. In this case, the area immediately above the *Live Paper* desktop is its augmented zone – the user only has access to new features when he or she places the paper on the desk. Other zones could include wearable augmented-reality (AR) computer systems, notepads, and intelligent rooms.

The true utility of these zones will only be realized when the zones are interconnected and allow the user to seamlessly move among them. For example, a person could wear an AR system to see annotations and other media on a piece of paper that was enhanced during a desktop session. Thus zones could be public and fixed, or personal and portable.

The design of this project focuses on a single zone – an office desktop; however, the general design is applicable to systems for other Zones. The overall architecture is also conducive to a multiple zone system. For example, object model storage could be shared among systems.

3.2.2 Paper as Centre of Focus

Several research projects have examined enhancing different types of objects, including pens and clothing. For this project, the enhanced objects are pieces of paper. Although on-line displays have supplemented or replaced many tasks for which people have traditionally used paper, paper remains the better for reading [O'Hara 1997]. Until interactive displays become extremely cheap and light (as ultimately envisioned by [E Ink 2004]), paper will be the superior medium for annotating, navigating quickly, and spatially organizing content.

The user must have an effective illusion that a page gains new properties and abilities when laid on the desktop. The desktop itself should be viewed as a location where paper is enhanced. Thus the name of the project is *Live Paper*, as opposed to Live Desk.

For the system to succeed with this illusion, the user must be able to place any piece of paper on the desk, at any orientation, under any illumination, on top of other pages, and possibly occluded, and still see the augmentations appear quickly and correctly.

Visual enhancements do not necessarily have to appear on the pages themselves, but should be obviously linked to a specific page. There should be some indication that an audio enhancement is emanating from a particular piece of paper.

3.2.3 Cards, Pages, Books

Each piece of paper provides a number of clues about its primary function, such as for reading or content creation. The clues would certainly include the markings on the paper, but also include more subtle, but easier to discern, hints such as the position of the paper and its size. The distinct size differences of cards, pages, and books, and their corresponding functions, suggest that particular augmentations can be automatically associated with a piece of paper.

Cards are pieces of semi-rigid paper the size of business cards. Traditionally, they have been used for identifying a person with a cluster of attributes (such as address and affiliation), for certifying something about a person (for example, a license), or for representing value (such as a coupon). At the other end of the size scale is the book, which has high inherent information capacity. In an augmented zone, it can be enriched by multimedia versions of the content, including *in situ* footnotes, and interactive diagrams.

In both of these cases, the printed material is generally permanent, suggesting that enhancements can be added at the time of content creation. If the intended use of the material permits, then distinct markings (glyph codes) can also be printed (such as [Kobayashi 1998]). The augmented zones can use these glyphs as triggers to enhanced content. However, existing cards and books would only benefit from glyphs if the user explicitly added markings. Ideally, a zone would use the structure of the printed material, possibly including optical character recognition (OCR), to uniquely identify any card or book. Then users could add their own enhancements, such as audio notes or links to pages on the World Wide Web.

In contrast to the card or book, the third paper size is the page, which is an individual sheet of paper such as a letter or form. Pages are the most complicated of paper types, as they may be blank or contain writing when they are first presented to the system. The user often modifies the content of the pages with markings. A zone of augmentation therefore must track where the page is, ignoring occlusions and projected markings, and also must redefine its internal description of the page as the user modifies it.

The emphasis in the *Live Paper* research is on the page – a system that can handle pages with permanent markings should also be capable of dealing with cards and books with minor modification. The system will allow a variety of page sizes, but will not infer roles from their sizes.

3.2.4 Transient, Temporary, and Permanent Actions

The actions of the user in the *Live Paper* environment can be broadly classified into three groups. The nature of the actions helps identify how the user interface should be designed.

Transient actions are those that the user quickly performs, and can promptly forget about. The actions do not have any affect on the real world. A gesture such as pointing would be a transient action. *Temporary* actions affect the real world, but can easily be undone, such as moving a piece of paper. *Permanent* actions affect the real world in such a way that the action cannot be easily undone. For example, marking on a piece of paper with ink is a permanent action.

In the *Alternate Reality Kit* [Smith 1987], Smith suggested that interfaces embody a tension between *literal* interaction, such as direct manipulation of an object, and *magical*, such as pop-up menus, which do not occur on real objects. Literal interactions are easier for novices because their functionality is more obvious. However, magical interactions provide more power and flexibility.

In general, temporary and permanent actions are more suited for literal interactions, while transient actions are suited for magical interactions. For general interaction with the system, such as controlling what augmentations are displayed, the user would use the transient action of pointing a finger at a 'magical' button. When the user does a permanent action such as mark on a page, he or she would expect a response appropriate to the task. So if the user drew a floor plan for a house, then he or she would expect projected walls to appear. This annotation should then track the page as the user moves it about the tabletop.

So the magical augmentations provide the user with a means to control the way the system behaves, while literal augmentations are related to the physical characteristics of the pages themselves.

3.2.5 User Interface

The interface between the user and the *Live Paper* system involve two modalities – the way the user interacts with the focused task, and the way the user issues controls to the system. In the first case, the interface of the application is specific to the task, and can be considered very literal. For example, the annotations involved with landscape modelling will be different than those for remote collaboration. As new applications are developed, new interfaces will also have to be developed.

However, in the second modality, the user is issuing commands to the system. There is no corresponding non-magical means of controlling the system; the user will have to learn the basics of how to interact with *Live Paper*. To assist the user, the design of the user interface must conform to a number of basic principles:

1. Obvious. The user must be able to determine which projections are meant for control, and which ones are related to the task.

2. Non-interfering. While working with the paper, the user should not have to worry about the interface.

3. Easily accessible. The user must be able to quickly get to the controls for a needed page.

4. Consistent. The basic controls for different applications should be consistently presented to the user, so that he or she can use existing knowledge of how the system works.

The user should control the system through magical interactions. Thus the user should use gestures to select a projected button, not make permanent marks on a page (or the tabletop) for control purposes. Similarly, the system should not, in general, require special marks on the page for identification or to trigger applications.

In addition to these basic principles, other practical aspects are important. Ideally, the interface should only require a video camera for input – a purely vision based interface. In principle, audio could be used for voice commands; however, audio would not be used for detecting when the user tapped the tabletop [Wellner 1993a]. A digitizing tablet would not be available, as the tabletop is a standard wooden desktop.

3.3 REALIZATION OF DESIGN PRINCIPLES

This section describes how the final *Live Paper* system realizes the design principles in Section 3.2. The basic means of augmenting a piece of paper is with one or more *transparencies*. These are small application objects that provide enhanced abilities, such as data projections, inter-networking features, information storage, and audio processing. To add a new ability to a piece of paper, the system associates an existing transparency with that page. Multiple transparencies can be placed on a page to provide complementary functions.

Each page or card has a standard *transparency portal*, or *transport*, with a suite of tools for the user. The transport is projected onto the tabletop next to the page or card. It provides a visual anchor for the individual transparencies, and clearly indicates transparency status. Through the transport, the user can associate, activate, deactivate, and modify the properties of a transparency. Input from the user is captured and passed to the transparencies, including writing and finger selections. The system stores all necessary information for a live page on the hard drive of the host computer.

The remainder of this section presents in detail three transparencies: a music player, an architectural renderer, and a remote collaboration tool. In addition, the *Live Paper* system has transparencies for sharing pages, browsing the web, and debugging; the descriptions for these transparencies are less detailed. Section 3.4 highlights the research problems that were investigated in order to be able to create the transparencies.

3.3.1 Transparency Interface

The user needs a visual anchor for the transparencies – to answer the question "how do I interact with the enhanced pages?" The system provides a transport at the right edge of each page (see Figure 3-3(a)). Each transport has a series of tabs, where each tab either represents the portal itself or a particular transparency. Thus a new page will have a transport with one tab. As more transparencies are added to the page, its transport grows larger to accommodate all of the tabs.

By selecting the tab, the user can show or hide a particular user interface for that transparency (see Figure 3-3(b-c)). Current transparencies show a stacked set of control buttons, which the user can select. These selection features are activated through hand gestures – pointing at a tab or button with a finger will display a confirm button. If the user then confirms the selection by pointing at it, the appropriate action will occur. When the user occludes multiple buttons, the system decides the selected button by choosing the one furthest from the user.

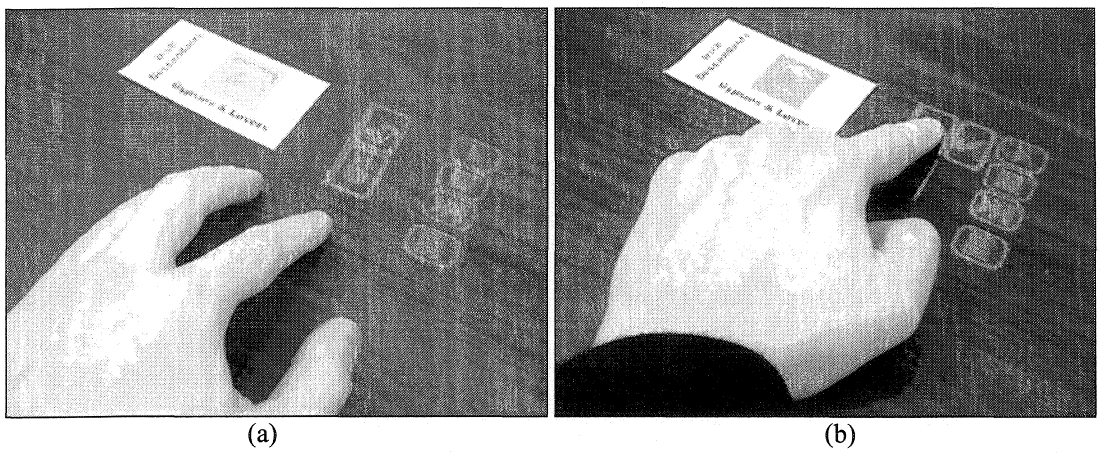
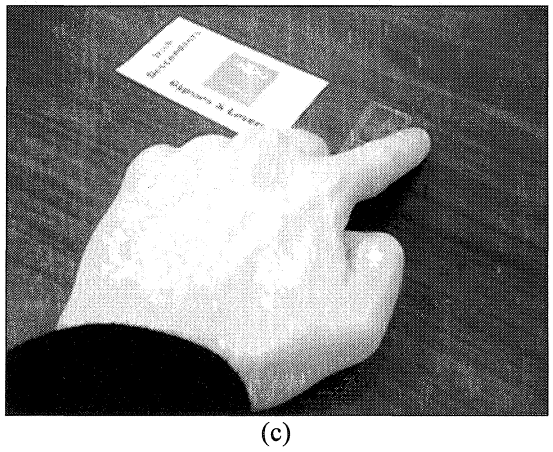



Figure 3-3: The transparency interface

(a) Page (card-sized) with Transport and Music user interface. (b) User is selecting the Music tab. (c) Tab selection has been confirmed, thus hiding the Music interface.

In addition to selecting buttons, the user can also interact with a transparency in a more basic way: by moving the page. For an application such as the architectural renderer, translating and rotating the page provides the user with a direct way to vary the view of the enhanced data. All transparencies have access to information such as the orientation, position, and printed content of the page, and thus can react to any changes that the user makes to these properties.

Another feature of the *Live Paper* system is a communication channel that can be established between transparencies, even if they belong to different pages. This allows the transparencies to exchange information, such as the current page content. The local page sharing transparency uses this feature.

A number of research problems were addressed while developing the transparencies, including page detection and recognition, image/projection registration, and finger detection. The relative importance of the solution to each problem varied with each of the transparencies. For example, the music player does not require that page locations be particularly accurate, but does require good recognition and finger detection. However, the architectural renderer also needs very precise measurements of a page's location.

3.3.2 Music Player

The Music Player is a transparency that plays a stored list of music. It reads a script file when created, which it then uses to determine the order and location of music files. The user may select the music tab icon to display or hide a strip of buttons (see Figure 3-4), which allow the user to play, pause, stop, skip to the next track, skip to the previous track, and to display a menu of songs (see Figure 3-5 for list of buttons, and Figure 3-6 for an example sequence of user interaction).

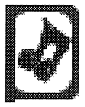


Figure 3-4: The music transparency tab icon

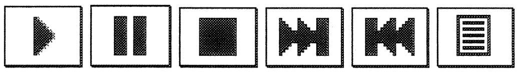


Figure 3-5: Buttons for the music transparency

From left to right: play, pause, stop, next track, previous track, and show//hide menu.

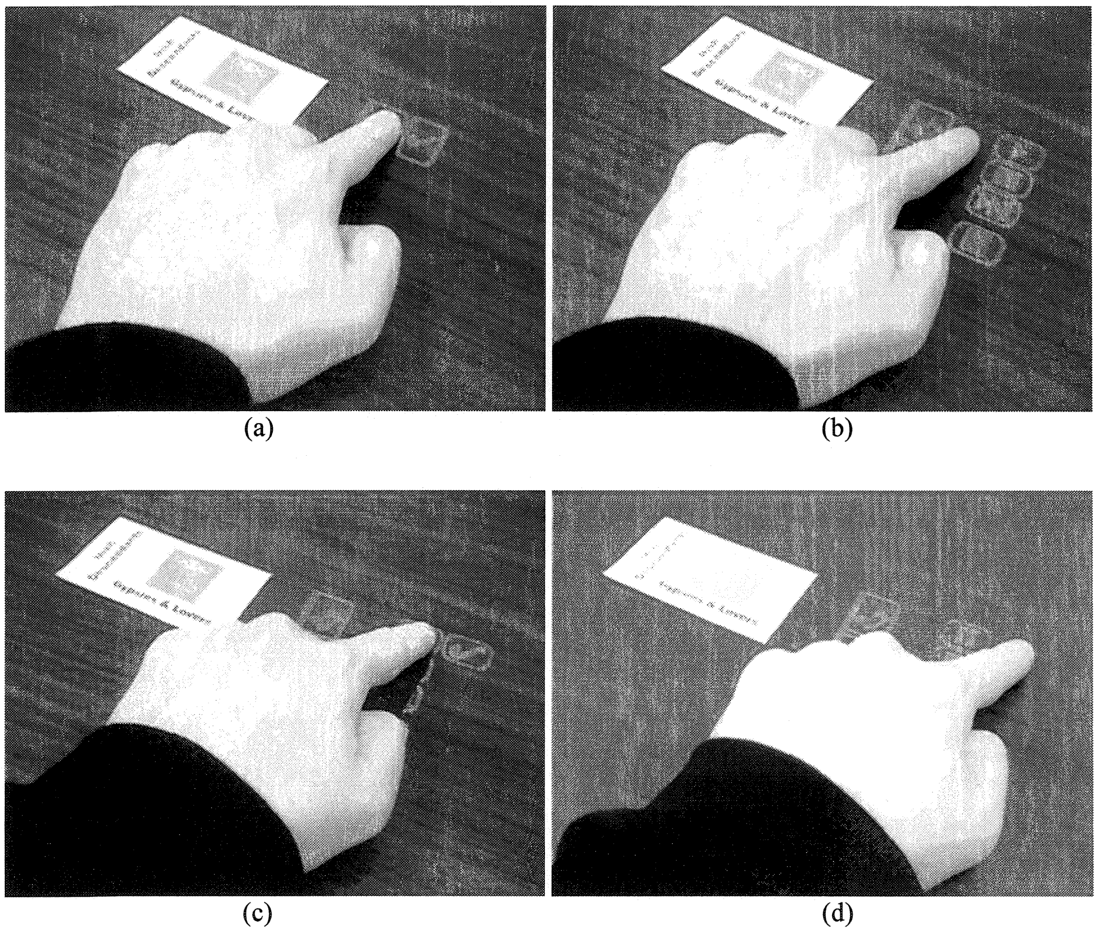


Figure 3-6: Controlling the Music Player transparency

(a – d) The sequence of photographs shows the user pausing music playback.

By default, the transparency begins playing music when its associated page is first laid on the desktop. The play button automatically updates as the playing status changes; for example, if a song is playing, then the button displays a pause icon. Although no printed overlays are generated by the transparency, the button stack does move as the paper is moved.

3.3.3 Architectural Renderer

The Architectural Renderer transparency displays a perspective-corrected, three-dimensional rendering of walls, doors, and windows related to a floor plan. The rendering appears to be attached to the page, and thus will rotate and move with the page. The transparency assumes that the user is standing in front of the desk, and thus uses a point of view 30 cm in front of, and 80 cm above, the desktop. A three dimensional house, for example, appears to sit on the desk, and the perspective will change appropriately as the page is moved (see Figure 3-7). Please see Figure 3-20 for another architectural example.

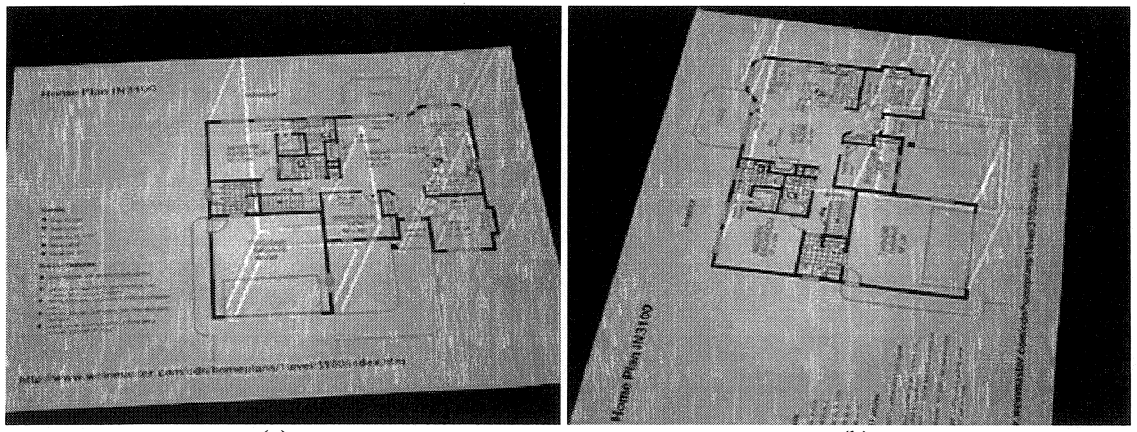

(a) (b)

Figure 3-7: The Architectural Renderer transparency in action

The three dimensional rendering of the house changes as the page is rotated or translated.

As with the music transparency, the user may select the architectural renderer's tab icon on the transport to show or hide a stack of buttons (see Figure 3-8). There are three buttons, which allow the user to show or hide the rendering, to show the rendering as a stereo view (requiring a pair of red/blue glasses) or in a monocular view, and to display the rendering in wireframe mode or with hidden lines removed (see Figure 3-9).

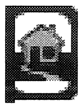


Figure 3-8: Architectural Renderer Tab

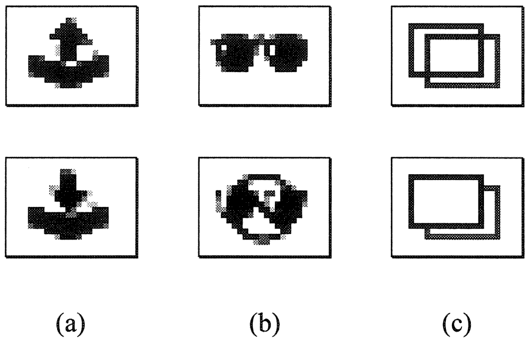


Figure 3-9: Architecture toggle buttons

(a) Show/Hide (b) Stereo/Mono (c) Wireframe/Hidden line removal.

With a pair of red/blue stereoscopic glasses, the user can view the three-dimensional rendering of the building in stereo. Because the point of view is fixed, the user must stand in set location to receive the best effect. A worthy upgrade to the system would be to incorporate some means of head-tracking. This could provide the *Live Paper* transparency with a changing point-of-view, and thus the user could look at different sides of a building by moving his or her head as well as moving the page.

Figure 3-10 shows a sequence of user interactions, that begins with the user selecting and confirming the tab icon to show the button stack, and ends with the selection and confirmation of the button to hide the 3D rendering.

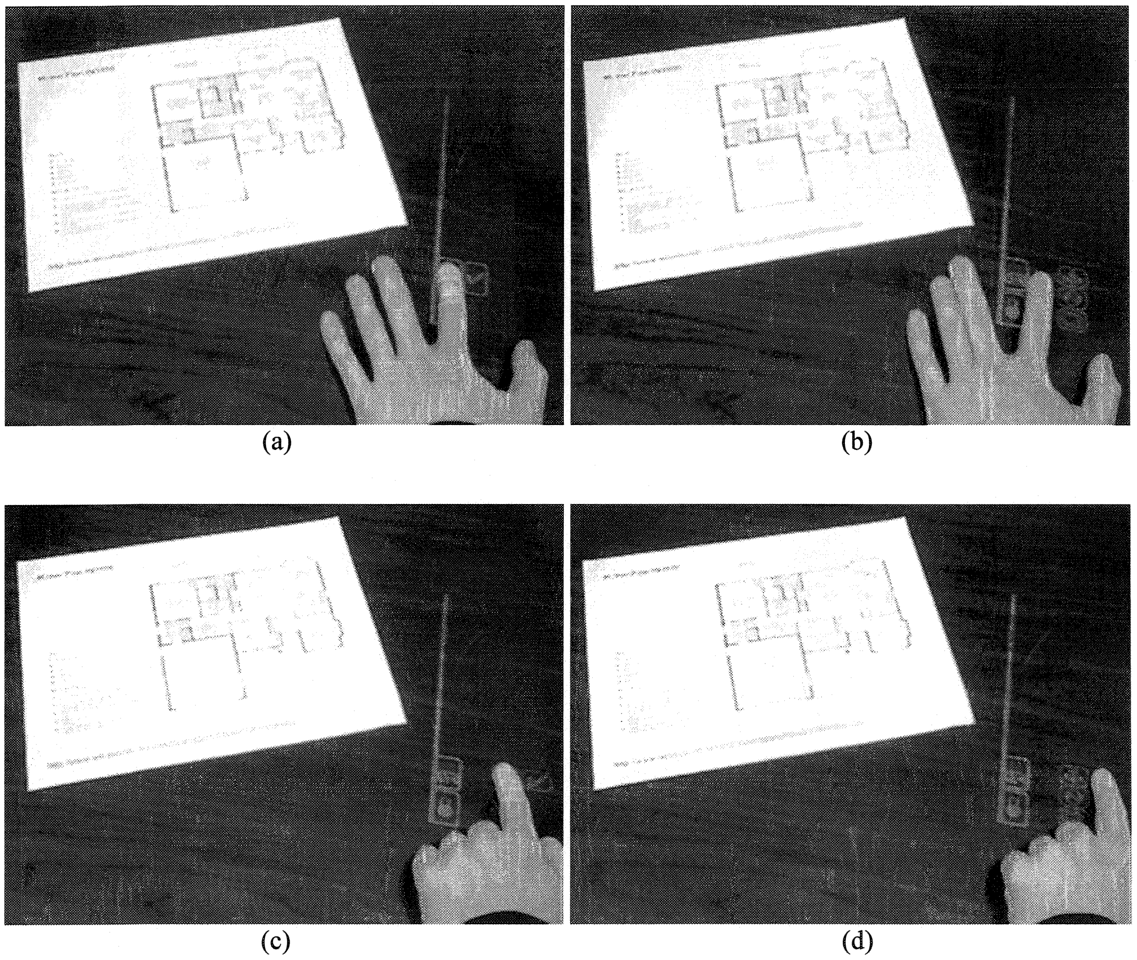


Figure 3-10: Sequence of user actions with the Architectural Renderer

3.3.4 Remote Collaborator

The Remote Collaborator transparency establishes a link with networking facilities of *Live Paper*, and makes the image of the page available for transmission. Users located at remote sites can connect to the *Live Paper* system and view any page that has this transparency activated. While viewing a page, the remote user can use a mouse to add annotations. These appear on both the remote user's screen and on the page on the *Live Paper* desk (see Figure 3-11).

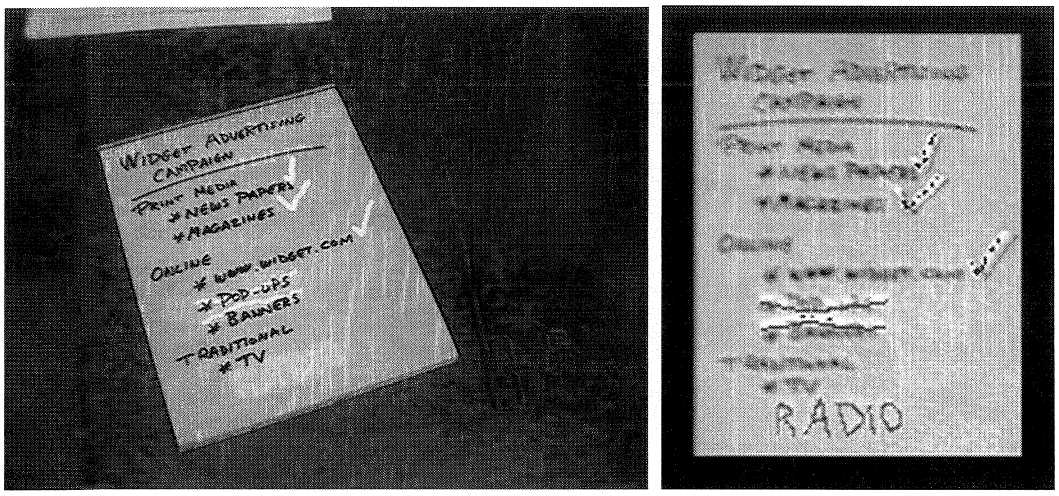


Figure 3-11: An example of the Remote Collaborator in action

***Live Paper* displays the remotely-created annotations directly onto the page on the tabletop (shown on the left). The remote user creates the annotations on a virtual page displayed on a monitor (shown on the right).**

The tab icon of the Remote Collaborator (see Figure 3-12) changes from a dark red arrowhead to a bright green, animated arrowhead when a remote user connects to the system (see Figure 3-13). This is a visual clue to the local user that a remote user is currently able to view the page.

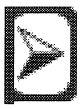


Figure 3-12: The Remote Collaborator tab icon

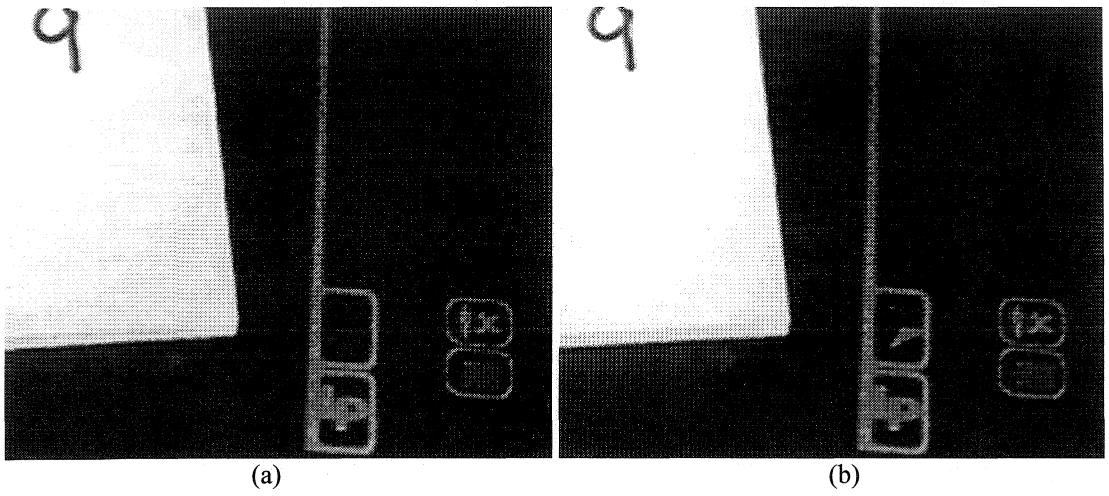


Figure 3-13: An animated icon in the tab shows the status of connected remote users. (a) When there are no remote users, the arrow in the tab is red and static. (b) When a remote user connects, the arrow changes to green and begins moving slowly.

The local user may choose actions from two toggle buttons (see Figure 3-14), which appear when the user selects the tab icon. One button turns on or off the annotations drawn by remote users. The other button permits or refuses remote users from viewing the page.

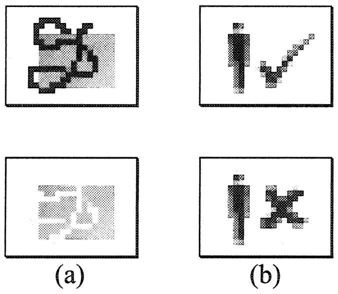
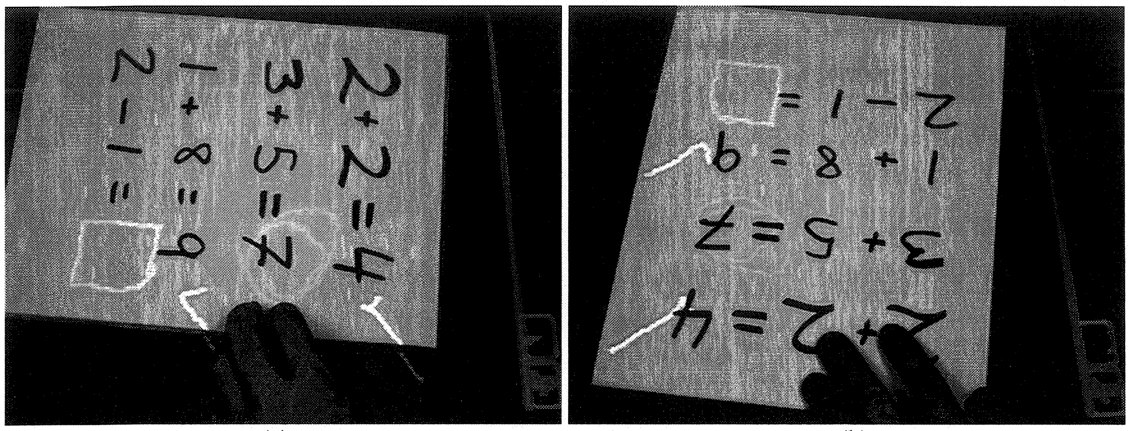


Figure 3-14: Buttons available in the Remote Collaborator transparency (a) Toggle remotely-created annotations on/off (b) Permit/refuse remote users

The annotations are locked onto the page in a similar way as the Architectural Renderer transparency. As the user moves the page on the desk, the annotations track the page (see Figure 3-15 for an example).



(a) (b)

Figure 3-15: Rotating a page with a Remote Collaborator transparency

(a,b) The annotations are locked on the page, so rotating or moving the page causes the annotations to track appropriately.

3.3.5 Other Transparencies

The *Live Paper* system contains several other transparencies, including a web transparency, a page sharing transparency, and a debugging transparency.

The Web transparency (see Figure 3-16 and Figure 3-17) displays a simple and relatively small window of a World Wide Web page on the desk. When activated, the window appears with the contents of the web site. Standard Windows scroll bars are present in the window, but they cannot be selected by finger gestures. A web site at any fixed URL address can be displayed.

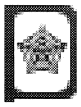


Figure 3-16: The tab icon for the Web transparency

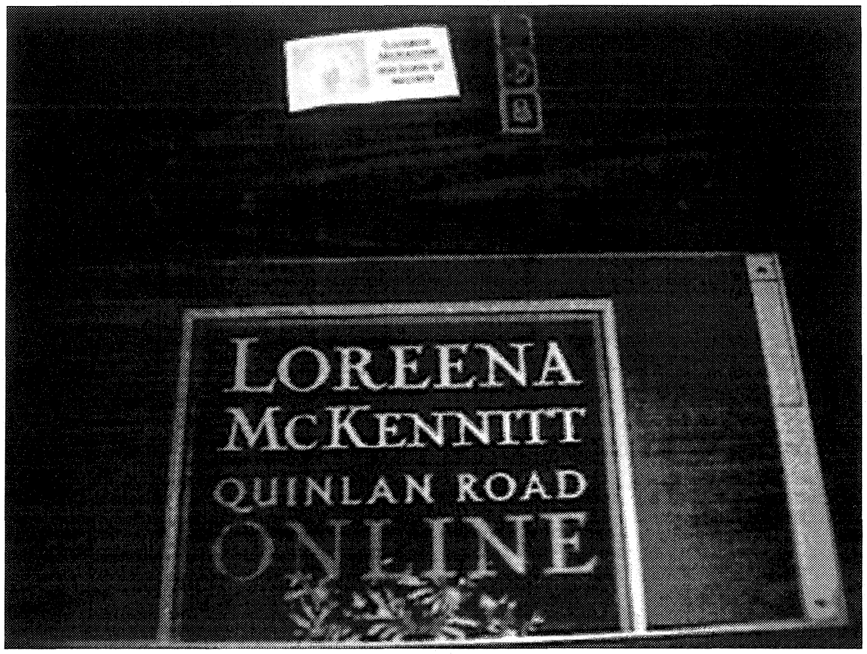


Figure 3-17: Web transparency in action

The user may toggle the visibility of the web page by selecting the web tab icon. However, because each projected pixel is about 1mm x 1mm, even a small web page takes up significant space on the desktop. Without a higher resolution projection, the utility of the Web transparency for all but the smallest of web pages is limited.

The PageShare transparency (see Figure 3-18 and Figure 3-19) shares the content of two physical pages, similar to Tele-Graffiti [Takao 2003] but with fewer constraints. The content of each page is extracted, transmitted, and then projected onto the other page. Ideally, the two pages would be at different Live Paper systems, but for this project both pages were on the same tabletop. Conceptually, this is the same as two remote desks and illustrates the feasibility. The transparency could be used to remotely mark-up page content, or to collaboratively brainstorm. PageShare does not have any independent buttons other than its tab icons.

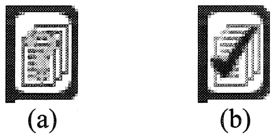


Figure 3-18: The tab icons of the PageShare transparency

(a) This icon is shown if no page link has been established (b) once two pages are shared, both show this icon in their transport.

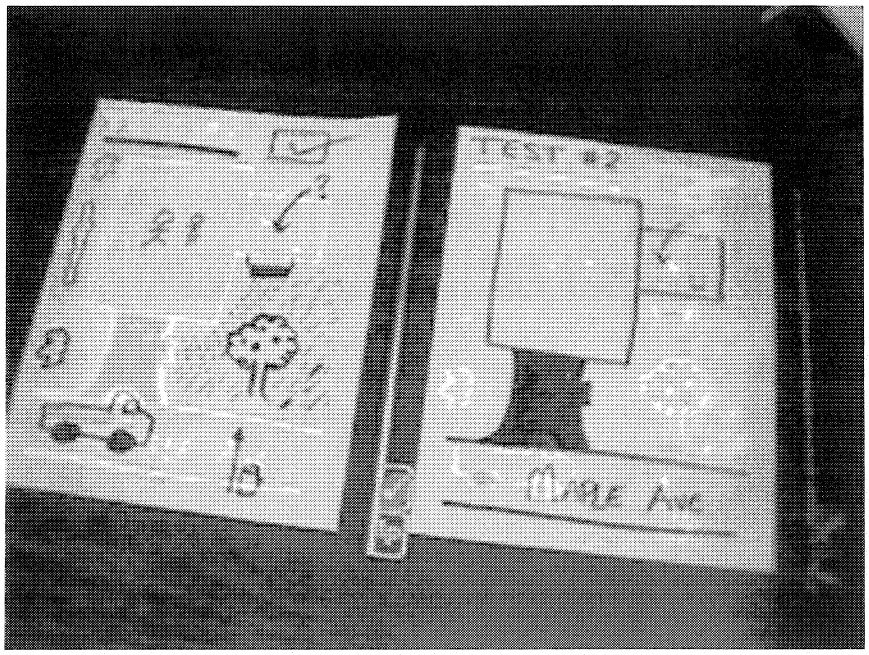


Figure 3-19: The PageShare transparency in action.

Content from each page is extracted, inverted, and displayed on the other page.

The Debugger transparency (Figure 3-20) provides some simple facilities for checking the position of the current page. At each of the four corners a crosshair target and a label is displayed. The label is one of UL (for upper-left), UR (upper-right), LR (lower-right), and LL (lower-left). The labels are established based on the camera perspective, and so appears to be rotated 180 degrees with respect to the user. As the page is rotated, the labels stay associated with the appropriate corner, based on the original viewing of the page.

Below each of the corners, the system displays 3 sets of coordinates, relating to the position of the crosshair in the image, desktop, and projector coordinate systems. The debugger application also displays the original aspect of the page, either *Landscape* or *Portrait*. The tab icon of the debugger (see Figure 3-21) is animated – as long as the system is processing normally, the icon refreshes. A frozen icon indicates abnormal system behaviour. The Debugger transparency was very useful during development, but is not intended to be normally available on the tabletop.

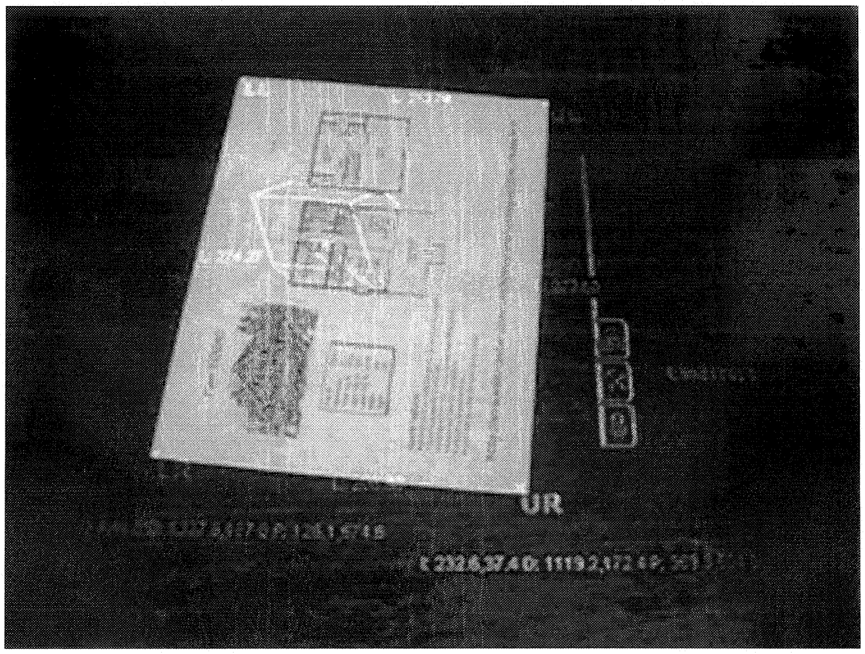


Figure 3-20: Debug transparency in action

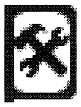


Figure 3-21: The animated tab icon for the Debugging transparency

3.4 RESEARCH TASKS

The implemented transparencies required a number of research tasks to be solved. These included:

- Determining the locations of all pieces of paper on a desktop, whether they are placed alone, overlapped, or occluded.

- Uniquely identifying these found pages with pages stored in a database. The system must determine if a page is not in the database, in which case the system must add the page.

- Registering a projected annotation with a physical page. The annotation should track a moving page, even when the page is partially occluded.

- Developing a suitable system architecture that supports the transparency framework and the other components of the user interface. The architecture must incorporate the image processing algorithms.

The follow chapters examine each of these problems in detail.

Chapter 4

FINDING PAPER ON A DESKTOP

This chapter addresses the issue of processing a captured image of a desktop to determine the number and location of pages contained thereon (see Figure 4-1). It describes novel applications and extensions of segmentation and boundary extraction methods. It introduces and evaluates a graph-based approach for dealing with overlapping and occluded pages.

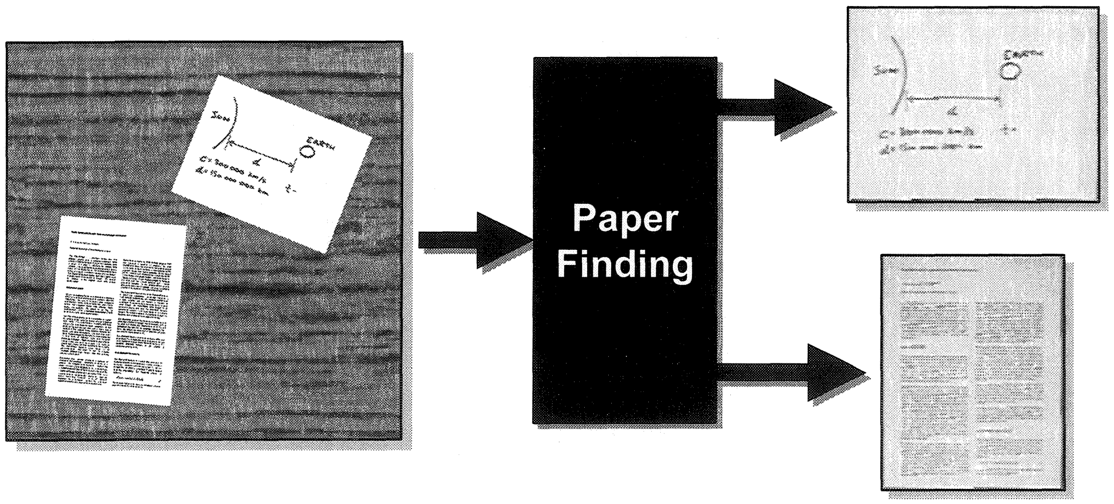


Figure 4-1: The process of finding paper pages, viewed as a black box

4.1 MOTIVATION

Before the *Live Paper* system can enhance a piece of paper, *Live Paper* must first determine where the paper is located on the desktop and extract its image. The issue of finding paper is one that most researchers in the area of Video Augmented Environments have ignored. Those that have attempted to find pages have used identifying marks affixed or embedded in the page. However, the intention of the author was for *Live Paper* to be more generally applicable, which required that the system be able to locate any piece of paper that is laid on the desk surface (see Figure 4-2).

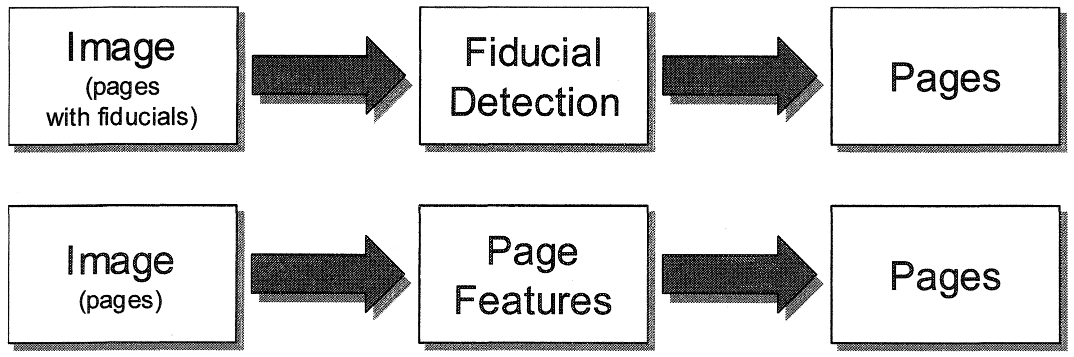


Figure 4-2: Alternative processes for page-finding.

The assumptions of what information is contained on a page will affect the overall process of finding that page. *Live Paper* follows the lower sequence.

The *Live Paper* system is to be applicable in a general office environment, without the need for specialized hardware. Thus the only input device is a video camera, and the writing surface is an ordinary desktop. The only information available to the paper finding algorithms is a series of video images that might or might not contain paper. Given an individual image of a desktop, the system must be able to determine the location of pages. Once the system knows that there are pages present, it can take advantage of this information when processing subsequent images.

This chapter investigates a series of algorithms that are suitable for locating paper on a desktop. Section 4.2 presents an overview of the methodology, which is then explained in detail in subsequent sections. These sections explain the algorithms used and show how they were applied in new strategies suitable for the research problem of finding paper. The chapter concludes with demonstrations of the validity of these methods.

4.2 METHODOLOGY

4.2.1 Requirements

The *Live Paper* system must operate in real-time, so that a user can interact with the projected annotations. Ideally, the system should process captured images at the same rate at which the image capture card provides them (up to 30 fps). In practice, the system must be able to process frames fast enough so that lag between the user's actions (such as moving a page) and the system response is not seriously compromised.

The algorithms must also provide a good level of accuracy by maximizing the rejection rate of false positives (areas of the desktop that look like pages) and the acceptance rate of false negatives (pages that are not located). Accepting false positives is the larger problem, as there is little that the user can do to accommodate system errors in this respect.

The algorithmic analysis must balance computational speed and correctness, with refinements as development progresses.

4.2.2 Assumptions

The basic assumptions about the desktop, the video camera, and the features of paper pages must be broad enough to be practically useful, but narrow enough so that a solution can be found. The prime assumption is that page-finding involves white paper laid flat on a desktop with a dark finish. The paper must be in good shape, rectangular, and have a perceptible margin (approximately 1 cm). Analysis can take advantage of these properties so to be fast and robust.

The relationship between the camera and desktop cannot change except under computer control in a known manner. This is the case with a pan-tilt video camera. The physical location of the camera is fixed with respect to the desktop.

After the pages are found and identified, the *Live Paper* system will augment them with projected overlays of information. An assumption is that the video projections will not interfere with the effectiveness of page finding via image processing. Later tests will show the validity of this assumption.

4.2.3 Process

Finding paper consists of two key tasks: feature extraction and feature analysis (see Figure 4-3). Feature extraction consists of a segmentation stage and a boundary analysis stage. Segmentation converts a masked version of the captured image into two sets of pixels – those that are deemed part of a page, and those that are not. Boundary analysis examines this set of pixels, and extracts a set of boundaries that represent the edges between paper and desktop. These are then analyzed to determine the salient features of corners and edges.

Feature analysis uses the extracted features to determine the locations of pages on the desk. It has three stages, each of which is more thorough but more computationally expensive than the previous stage. However, features that are used in one stage are removed before the next stage begins, thus reducing the amount of processing required. Isolated paper analysis uses a simple set of criteria to locate single pieces of paper. The page tracking stage uses the database of existing pages with the newly extracted pages and extracted page features to determine the new locations of pages that might have moved. Those features that the system has not removed are then provided to the overlapping and occluding analysis stage. Due to the nature of the extracted features, analysis of isolated papers is significantly different from cases where papers overlap or when they are partly occluded by another object.

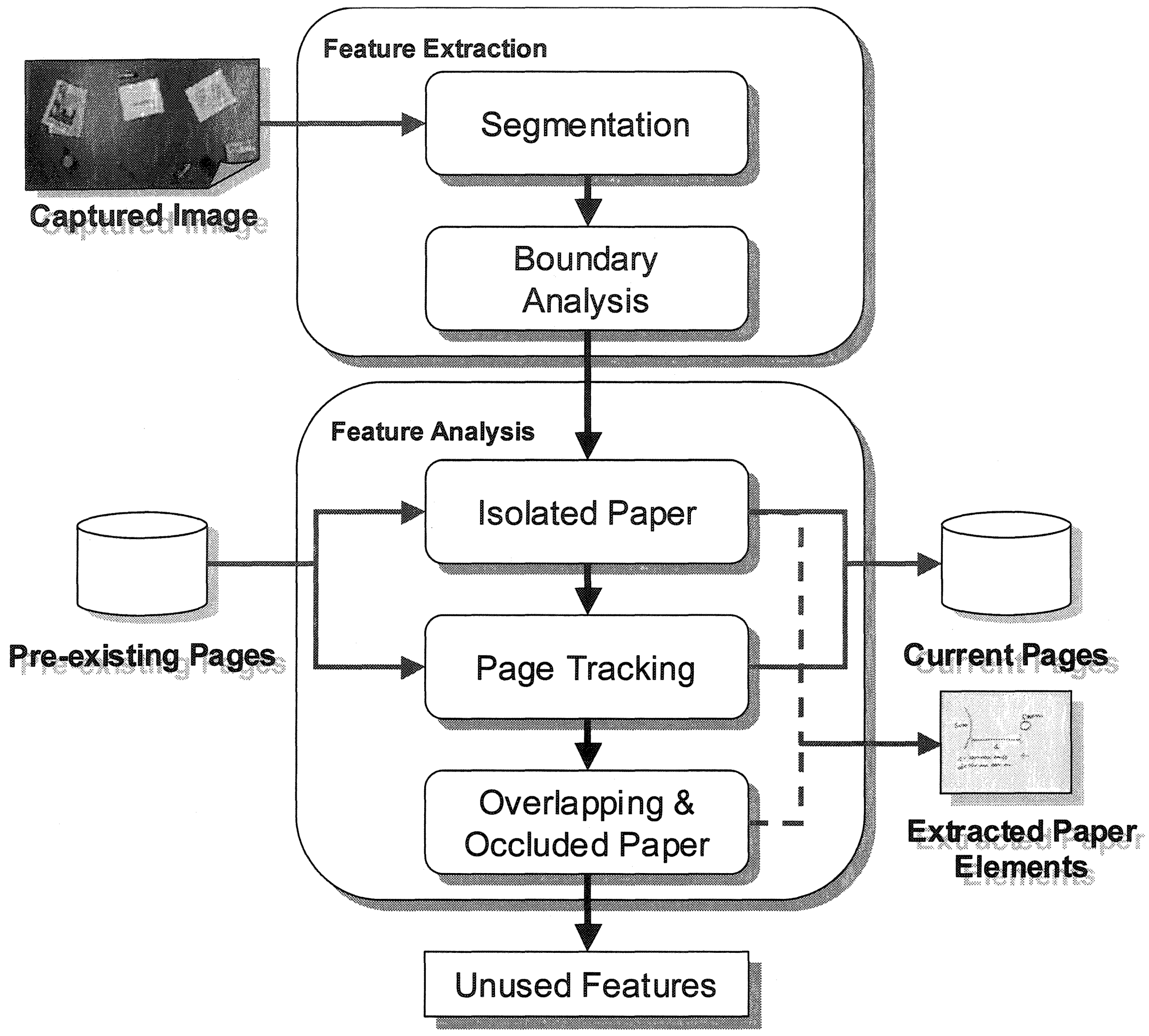


Figure 4-3: The key tasks for finding paper pages.

4.2.4 Background on Finding Paper

As indicated earlier, research projects that enhance paper on a desktop using a camera and data projector have either ignored the problem of finding paper pages or have used unique identifying marks. An example of the former is the *DigitalDesk* [Wellner 1993a], which did not attempt to find individual pages. The *Origami* project [Robinson 1997] is an example of the latter.

The *DigitalDesk* did not need to locate individual pages because it did not treat pages as autonomous units. For example, to copy a set of real marks into a digital version, the user selected an area with a digitizing pen. The system then projected the extracted marks onto the desk, where the user could move or copy them. The user could create an amalgamation of real and projected drawings, but these marks were not registered; moving the page did not move the digital projections.

The *Origami* fiducials were four crosshair-like shapes, one located at each corner of a page. The system would use image-processing techniques to find these shapes, and then determine the page location. The main drawback is that all pages must contain the marks, which is a problem for existing pages. There are also aesthetic issues, especially if the fiducials are large.

The *EnhancedDesk* [Kobayashi 1998] finds pages by using a unique identifying matrix code [Rekimoto 1998] affixed to a page. The shape, a square box with a grid of black-and-white pixels, is unique enough to be easily discriminated from the other markings on a page. The desk uses the matrix code to identify the page (see Chapter 5), which in turn allows the system to accurately place the page because the relationship between the matrix code and the page is programmed into the system.

The use of pre-generated markings means that these methods are not suitable for detecting arbitrary pages, and thus not suitable for *Live Paper*.

4.3 FEATURE EXTRACTION

4.3.1 Masking

Before the captured image is processed, *Live Paper* extracts the desk area. Although desk finding is another potential area of research, the current system assumes that the desk will not move with respect to the camera. Therefore the desk mask is set once. *Live Paper* only processes objects that are on the desk.

When the system is initialized, each pixel position in the captured image is checked to see whether or not it is on the desktop. The system does this by transforming the image coordinate to the desktop coordinate. If the desktop coordinate is within the bounds of the desktop (a rectangular area) then the corresponding mask pixel is set to 1; otherwise, it is set to 0. When subsequent images are captured, any pixels at the same location as those zero pixels in the mask are erased.

4.3.2 Segmentation

Image segmentation is "the decomposition of a scene into its components." [Jain 1988] After segmentation, the image has the same dimensions, but each individual pixel is classified as an object or as the background. Segmentation is useful for image analysis when the image contains well-defined objects that can be clearly separated.

There are many characteristics that can be used for segmenting an image, including intensity, local texture (homogeneity), edge strength, or colour. The choice of which characteristic to use depends on both the nature of the objects and any computational restrictions. For example, the segmentation technique of thresholding is computationally less expensive than using a measure based on texture, but might not provide sufficient discrimination.

Pages on a dark desktop have a number of distinguishing characteristics, including high intensity, sharp corners, strong edges, and lack of colour (for white pages). All or some of these features can be used to separate the page regions. The question to be addressed here is whether a thresholding segmentation algorithm using just the grey-level intensity of each pixel, and no other information, is effective for *Live Paper*.

4.3.2.1 Thresholding

A simple segmentation method is grey-level thresholding, which uses the intensity value of each pixel to classify that pixel. One or more intensity levels are chosen as thresholds, and all pixels that have intensities between these thresholds are classified as one object. For *Live Paper*, grey-level images have 8-bits of resolution, and thus can have up to 256 distinct levels of intensity. Thus if a single threshold of 100 is selected for a particular image, then all pixels with intensities of 0 to 100 are classified separately from those pixels with intensities of 101 to 255.

The *Live Paper* system captures colour images, with a separate 8-bit intensity value for each of red, green, and blue. To use an image with grey-level thresholding, the average of the three intensities may be calculated for each pixel:

$$I(r,c) = \frac{R(r,c) + G(r,c) + B(r,c)}{3} \tag{4-1}$$

Determining the optimal threshold to segment an image is the most important step. The threshold can be either adaptive or non-adaptive. An adaptive threshold changes its value in response to local characteristics around a pixel.

For ideal grey-level thresholding, the intensity values of the pixels corresponding to an object are different from those of the background object. Figure 4-4 is an example intensity histogram of an image that consists of an object that is uniformly more intense than the background. Since there is no overlap of the intensity values, choosing a threshold in between would perfectly segment the object from the background. In practice, the range of pixel intensities will overlap, as illustrated in Figure 4-5. Whatever the threshold selected, there will be object pixels segmented as background, and background pixels segmented as the object. The best threshold is the one that minimizes the number of misclassified pixels; however, this is often difficult to determine accurately. Simple heuristics include the intensity value midway between the peaks, and the lowest value in the valley between the peaks.

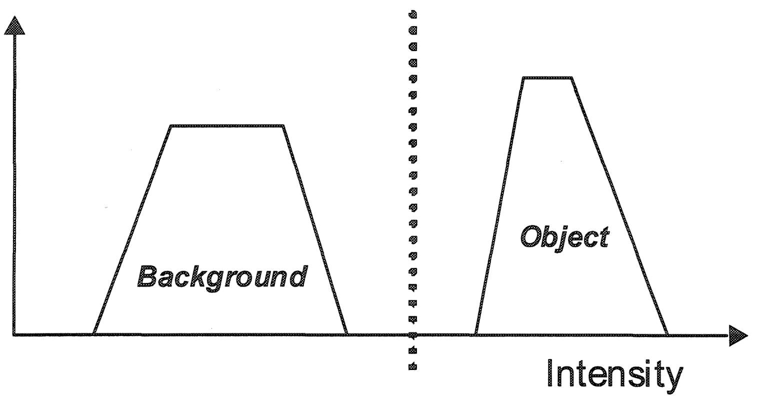


Figure 4-4: Example histogram of ideal distribution of grey-level intensity values.

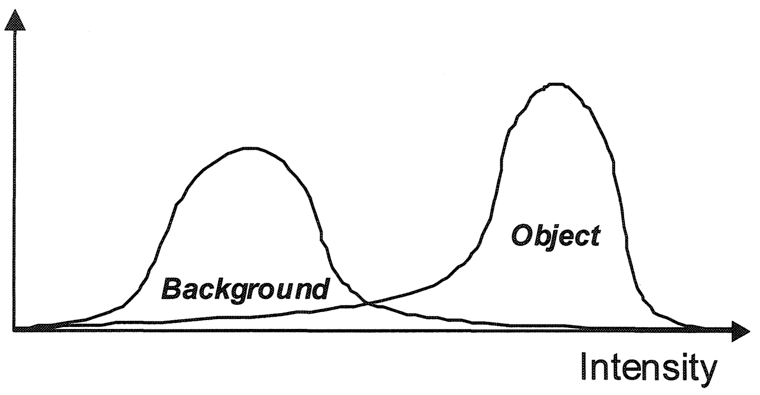


Figure 4-5: Example histogram of a typical real-world intensity distribution.

4.3.2.2 Otsu Threshold Determination

The Otsu thresholding method [Otsu 1979] uses discriminant analysis to divide the histogram into two or more classes. The thresholds chosen are optimal from the viewpoint of discriminant analysis; that is, the resulting classes have minimum inter-class spread and maximum intra-class separability. The thresholds can then be used to segment the image.

Live Paper uses Otsu thresholding to separate pixels belonging to paper pages from the background (see section 4.3.2.3). The following general formulation, for an arbitrary number of grey-levels and classes, provides details on the algorithm.

Let L be the number of grey levels (intensities) in the image. If the image is to be divided into k classes, then the number of thresholds is k-1. The thresholds, denoted by t, have grey level values such that $0 \le t_1 \le t_2 \le \ldots \le t_{k-1} \le L-1$. The grey levels are distributed among the classes such that class $C_0 = \{0, \ldots, t_1\}$, class $C_1 = \{t_1+1, \ldots, t_2\}$, …, and class $C_{k-1} = \{t_{k-1}, \ldots, L-1\}$.

The histogram is considered to be a probability distribution of the pixels in the image. First, the histogram is normalized, so that if the number of pixels with intensity i is n_i, and $N = n_0 + n_1 + \ldots + n_{L-1}$, then the normalized value at intensity i is $p_i = \frac{n_i}{N}$.

The best division of the histogram into k classes is the one that creates the maximum separability. There are three equivalent but different forms of defining class separability based on three measures of variance:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \kappa = \frac{\sigma_T^2}{\sigma_W^2}, \eta = \frac{\sigma_B^2}{\sigma_T^2}. \tag{4-2}$$

The term $\sigma_B{}^2$ is the between-class variance, $\sigma_W{}^2$ is the within-class variance, and $\sigma_T{}^2$ is the total variance. They are defined as:

$$\sigma_B^2 = \sum_{j=0}^{k-1} \omega_j \left(\mu_j - \mu_T\right)^2, \tag{4-3}$$

$$\sigma_W^2 = \sum_{j=0}^{k-1} \omega_j \sigma_j^2 , \tag{4-4}$$

and

$$\sigma_T^2 = \sum_{i=0}^{L-1} (i - \mu_T)^2 p_i . \tag{4-5}$$

The simplest form of class separability to compute is the measure $\eta = \sigma_B^2 / \sigma_T^2$. Because σ_T^2 is constant, the maximum value of σ_B^2 also gives the maximum value of η. Thus the optimal set of thresholds, $\{t_1^*, \ldots, t_{k-1}^*\}$, satisfies:

$$\sigma_B^2(t_1^*, \ldots, t_{k-1}^*) = \max_{0 \le t_1 \le \ldots \le t_{k-1} \le L-1} \sigma_B^2(t_1, \ldots, t_{k-1}) . \tag{4-6}$$

We define $t_0 = 0$ and $t_k = \text{L-1}$. To calculate σ_B^2, it is necessary to try all threshold combinations. Starting with the class occurrence levels, calculate for class C_j

$$\omega_j = \omega(t_{j+1}) - \omega(t_j) . \tag{4-7}$$

The value $\omega(x)$ is the zeroth-order cumulative moments of the histogram up to the x^{th} grey-level, and is defined as

$$\omega(x) = \sum_{i=0}^{x} p_i . \tag{4-8}$$

Thus $\omega(\ 0\ ) = 0$ and $\omega(\ \text{L-1}\ ) = 1$. The mean for class C_j is defined by

$$\mu_j = \frac{\mu(t_{j+1}) - \mu(t_j)}{\omega(t_{j+1}) - \omega(t_j)} , \tag{4-9}$$

where $\mu(x)$ is the first-order cumulative moment up to the x^{th} grey-level. It is defined as

$$\mu(x) = \sum_{i=0}^{x} ip_i \, , \tag{4-10}$$

The value $\mu(0) = 0$. The total mean level of the picture is μ_T, given by

$$\mu_T = \mu(L-1) = \sum_{i=0}^{L-1} ip_i \, . \tag{4-11}$$

For the case of a single threshold, the between-class variance is

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \, . \tag{4-12}$$

This can be simplified so that for a given threshold t, the variance is

$$\sigma_B^2(t) = \frac{[\mu_T\omega(t) - \mu(t)]^2}{\omega(t)[1-\omega(t)]} \, . \tag{4-13}$$

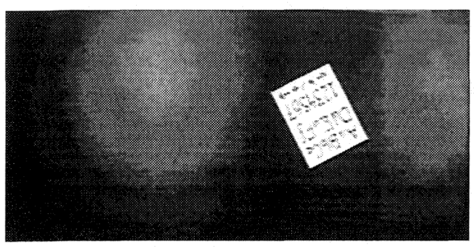
In the case of two thresholds, the between-class variance is

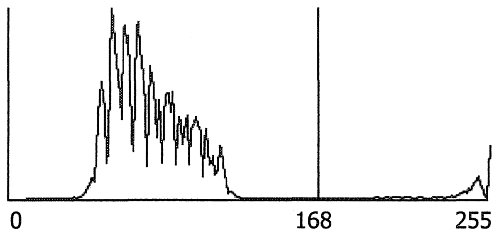
$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 + \omega_2(\mu_2 - \mu_T)^2 \, . \tag{4-14}$$

The complexity of calculating the single threshold variance is O(n), where n is the number of grey-levels. The two threshold variance has a complexity of O(n^2).

4.3.2.3 Application

Figure 4-6 is an image of a desk containing one page, along with its histogram of grey-level intensities. There are two distinct peaks in the histogram. The broad one on the left represents the dark areas of the desktop, as well as the writing on the page. The peak on the right contains those pixels in the page area. On the desk, the variations of brightness due to specular reflection and grains in the wood cause the left peak to be very wide.





(a) (b)

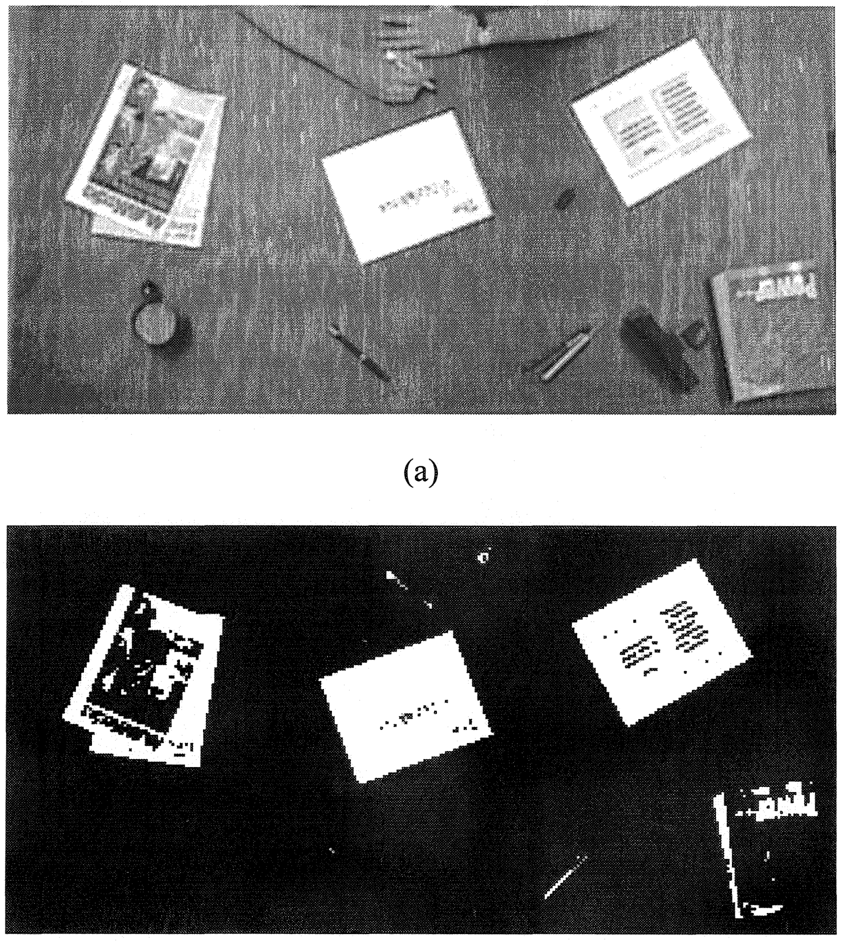
Figure 4-6: Determining the segmentation threshold.

(a) Image of page on a tabletop (b) Corresponding grey-level histogram (the value 168 was chosen by *Live Paper* as the threshold for the image).

Automatic setting of the threshold is essential to *Live Paper*. The optimal threshold depends on factors such as the gain of the camera's optical system, the lighting of the room, the desk surface, and the amount and thickness of the paper. The threshold varies as these factors change. Even the presence of windows in the room is enough to cause a change in the optimal threshold throughout a day. Automatic Gain Control on the camera might adapt, in a difficult to predict manner, to variation in the amount of paper in view. *Live Paper* must automatically determine a good threshold (close to the optimal) for separating a page on a desktop from the background.

In the preliminary design of *Live Paper*, the research tested a variety of automatic thresholding algorithms. These included selecting a grey-level from the valley between two peaks [Parker 1997], a locally adaptive method (see Chapter 5 [Wellner 1993b]), and the Otsu method described previously (Section 4.3.2.2). Of these, the Otsu method proved to be the most reliable. *Live Paper* uses an algorithm (explained later in this section) incorporating this method to automatically select a threshold for page segmentation. In Figure 4-6(b), the value of 168 is the threshold level that *Live Paper* selected for that image.

Figure 4-7 shows the result of thresholding a grey-level desktop image with an Otsu-determined grey-level threshold of 136. The white pixels in the thresholded image have been classified as belonging to a page (object); the black pixels are all other regions of the desk (background).


(a)

(b)

Figure 4-7: Thresholding an image of the tabletop.

(a) Grey-level image of a desktop (b) Thresholded version of grey-level information in (a)

Some non-page items, such as the magazines, pens, and books, have been misclassified as page regions. Likewise, the writing on each page has also been wrongly segmented. Further processing is necessary to isolate the pages from other falsely classified objects.

In most cases, the single threshold Otsu method is sufficient for segmenting the desk image. However, in some cases, such as when paper dominates the desk, the threshold is not in the histogram valley, but within the peak corresponding to paper. The method creates a class for the light areas of the paper, and one for the darker areas (classifying shadows with markings and the desk). The two-threshold version of the Otsu method always has one good threshold that is suitable for separating pages from the desk. This was verified in testing with over 200 images including a variety of scenarios such as only blank pages, pages with markings, and pages partly occluded by the hand of the user.

The two-threshold algorithm attempts to separate the histogram into three classes. These three classes are usually around the peak of the desk pixels, around the peak of paper pixels, and the valley in between. Most of the pixels with grey-levels in the valley correspond to the edges between the paper and the desk. Thus the lower threshold is normally optimal. However, at times the Otsu algorithm calculates the lower threshold to be within the range of intensities for the desk. It is necessary for *Live Paper* to detect this situation and switch to the higher threshold.

While a more complicated histogram shape analysis could be done, testing shows that the Otsu thresholds themselves can be used to determine which threshold to use. If the histogram grey-level count at the lower threshold is significantly below the count at the upper threshold, then the lower threshold is the optimal one. This level of difference was measured over a large set of images, and it was found that a factor of 3 worked consistently well. Thus for the final threshold T, Equation 4-15 gives the relationship when $a = 1/3$.

$$T = \begin{cases} t_1 & if \quad n_{t_1} \le a n_{t_2} \\ t_2 & otherwise \end{cases} \tag{4-15}$$

Because the two-threshold algorithm is O(n^2), where *n* is the number of grey-levels, the histogram is reduced from 256 grey-levels to 64 levels. This significantly decreases the computation time while essentially leaving the computation of the optimal threshold unaffected. The implementation pre-calculates all of the zeroth-order and first-order cumulative moments. To determine whether or not there is a page in the image, the system uses the total variance, which is low when there are no pages present.

4.3.3 Boundary Analysis

Boundary analysis takes a set of pixels that have been classified and determines the location of the characteristic features of the boundaries between the sets. The relevant issues include finding and extracting the boundaries, and then detailed analysis to find these salient features.

In finding the boundaries, the *Live Paper* system needs to be able to eliminate some of the more obvious false boundaries. These often arise when markings on pages are incorrectly classified as part of the desktop. One tool for doing this is morphological filtering.

4.3.3.1 Digital Image Morphological Operations

Morphology relates to the form and structure of an object (Parker, 1997). Morphological operations are useful for changing the shape of an object, such as increasing its size or removing spurious points or holes. For digital images, there exist different implementations of morphological operations for two-level, greyscale, or full-colour data.

Two-level images are assumed to consist of pixels that belong either to an object or to the background. Two basic operations are erosion and dilation. Structuring elements can be used in these operations, but are not necessary. Without a structuring element, erosion will eliminate an object pixel unless it is totally surrounded by object pixels. Dilation will change a background pixel into an object pixel if any of its neighbours are object pixels. For more flexibility, structuring elements can be used to enforce certain patterns.

An opening operation consists of an erosion operation follow by a dilation operation. Thin elements (such as lines of one or two pixels in thickness) will be eliminated, but thicker elements will not be affected. The complementary operation is a close operation, which is a dilation followed by an erosion. Closing will fill any small holes that exist in an object. Figure 4-8 presents an example of opening and closing operations.

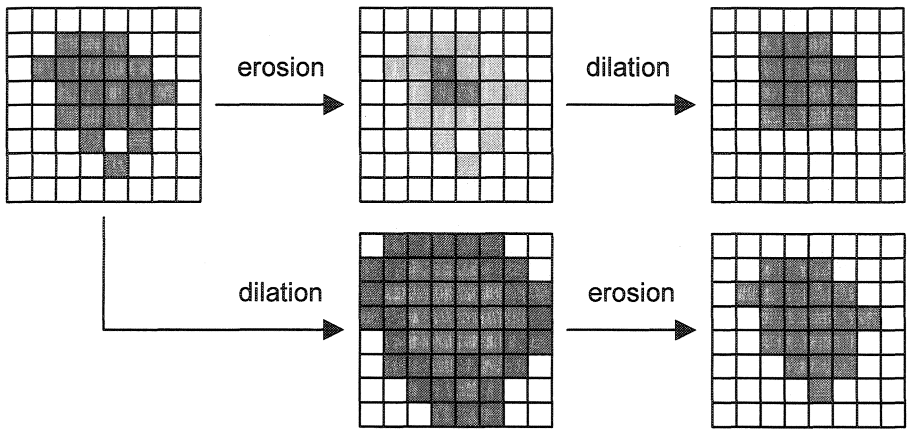


Figure 4-8: Binary morphological operations of erosion and dilation

Two-level morphological operations are used in some of the *Live Paper* feature analysis stages (see section 4.3.3.4).

4.3.3.2 Boundary Determination

A pixel is considered to be on the boundary of an object if it is part of the object and has at least one neighbour that is the background. The connectivity scheme used by the algorithm will affect the shape and thickness of the boundary. A 4-connected scheme considers only those pixels horizontally or vertically adjacent as neighbours. In an 8-connectivity scheme, diagonal pixels are also considered neighbours. The 4-connectivity scheme is simpler to implement and produces thinner borders.

A two-level segmented image can be converted to a two-level edge-map image where all object pixels are edge pixels (see Figure 4-9). The original image should be padded with background pixels to ensure that any object pixels on the image edge will not cause processing errors. Simply ignoring the edges will not work unless special care is taken to compensate for gaps. Creating an edge-map image is a simple process of scanning through the image pixel-by-pixel and examining the neighbours of an object pixel. If any neighbour is not an object pixel, then the pixel under consideration is an edge pixel.

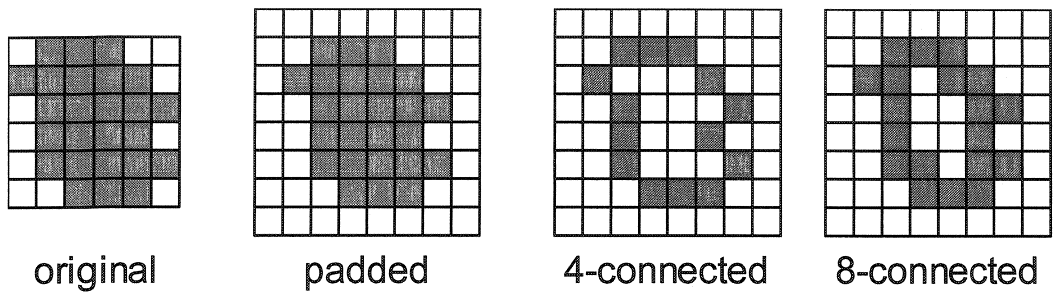


Figure 4-9: Effect of connectivity on boundary determination.

4.3.3.3 Boundary Extraction and Storage

An edge-map image only contains the position of edge pixels, not their direction or connectivity. A routine must scan through the image to find an edge pixel, and then generate a connected boundary map of the edge. If the edge-map was generated using a 4-connected scheme, then the routine must check the 8-connected neighbours to find all connected edge pixels. In closed contours, the first edge pixel that the routine finds will have two neighbours, and thus the extraction routine must decide which neighbour pixel to follow.

As the boundary is generated, the edge pixels are removed from the image. When the last edge pixel is removed, the routine checks for a closed boundary by determining whether the last pixel is adjacent to the starting pixel. Large simple objects such as pieces of paper have simple closed edges that do not branch. The presence of small objects could cause branches to appear (see Figure 4-10).

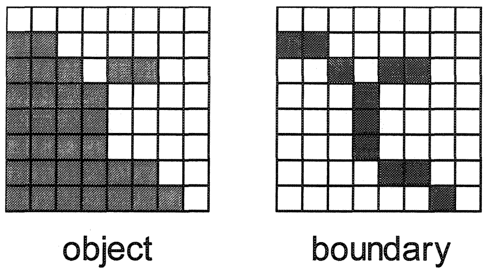


Figure 4-10: Example of a small object causing a branch to appear in a boundary.

A simple but useful way to store the extracted boundary is with a Freeman chain code [Freeman 1977], which is a list of links that reconstruct a raster-based contour. Once a boundary is encoded, salient features such as corners and straight edges can be found (see section 4.3.3.5). An *n*-link Freeman chain is defined by

$$A^n = a_1 a_2 a_3 \ldots a_n , \tag{4-16}$$

where a_i is the *i*th link in the chain. If A^n is a closed chain, then $a_i = a_{i \pm n}$. The links have one of eight values, $a_i \in \{0, \ldots, 7\}$, which indicate the direction from the current pixel in the chain to the next pixel (as shown in Figure 4-11).

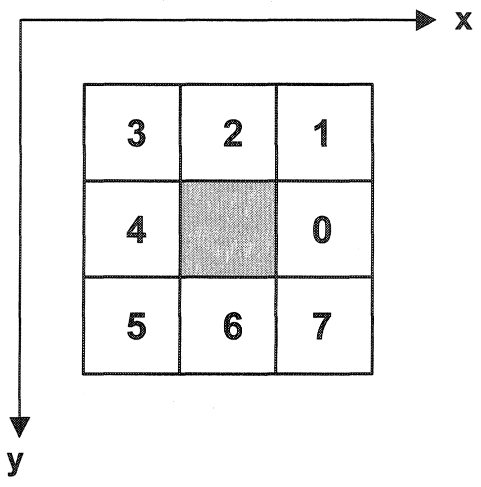


Figure 4-11: Directions corresponding to each value of a_i.

The x and y components of link a_i are a_{ix} and a_{iy}, respectively, where $a_{ix}, a_{iy} \in \{-1,0,1\}$. Thus if $a_i = 0$, then $a_{ix}=1$ and $a_{iy} = 0$, and the x coordinate of a_{i+1} is 1 more than a_i.

4.3.3.4 Boundaries Extraction in Live Paper

As the example in Figure 4-7 indicated, the purpose of boundary analysis in *Live Paper* is to convert a binary image of page/non-page pixels into a boundary representation of edges. The result is another two-level image, with the pixels that are 'on' corresponding to the edges in the original grey-level image (see Figure 4-12).

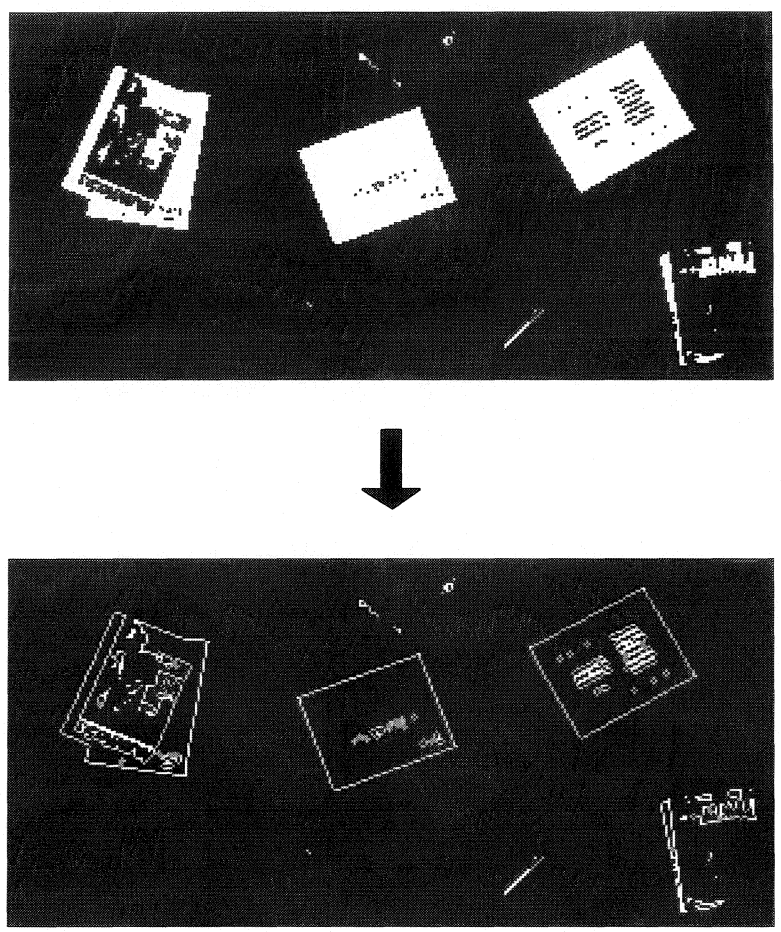


Figure 4-12: Locations of boundaries.

Segmenting a captured image into page and non-page objects is difficult to do perfectly. There are many potential objects that have similar intensities to white pages. If the segmentation were perfect, then the analysis would be straightforward – at least for the single page case. If misclassifications have occurred, then some object (page) pixels are in fact not pages (either desktop or other objects), or some pixels classified as desktop are page pixels.

In terms of computational efficiency, there is a trade-off between improving the quality of the segmentation or improving the boundary analysis. Fast and reasonably accurate segmentation is good; perfect is probably not necessary. Simple segmentation will not take too much time. Thus simple techniques to improve the segmented image (such as certain image morphology operations) will likely be worthwhile. Because images are much larger than boundaries (10 to 100 times more points), it would be better to spend additional computational resources on boundary extraction and analysis. These considerations are applied in *Live Paper* in the following way.

First the segmented image is converted to a boundary image. The segmented image is scanned row by row, from left to right. If the pixel under consideration is on and has at least one 4-connected neighbour that is off, then the pixel is classed an edge pixel. Before processing, the segmented image is padded with off pixels. Although isolated pixels in the segmented image could be eliminated at this stage, the boundary extraction method successfully copes with small objects and isolated pixels.

Live Paper next scans the boundary image row by row, from left to right. Once a boundary pixel is found, a contour-following algorithm traces the boundary using eight-connectivity. All contours are traced in a counter-clockwise manner, and stored in a Freeman chain code. The standard algorithm has been modified to account for the current direction of following the contour (see Figure 4-13). In the figure, the arrows indicate the direction from the previous border pixel to the current pixel. The numbers indicate the search order for the next border pixel. The number 8 indicates the previous border pixel –

if the search does not find a neighbouring pixel before reaching position 8, then the contour has ended. The modifications are necessary for the cases when the boundaries diverge, and encourage the algorithm to follow the outermost boundaries.

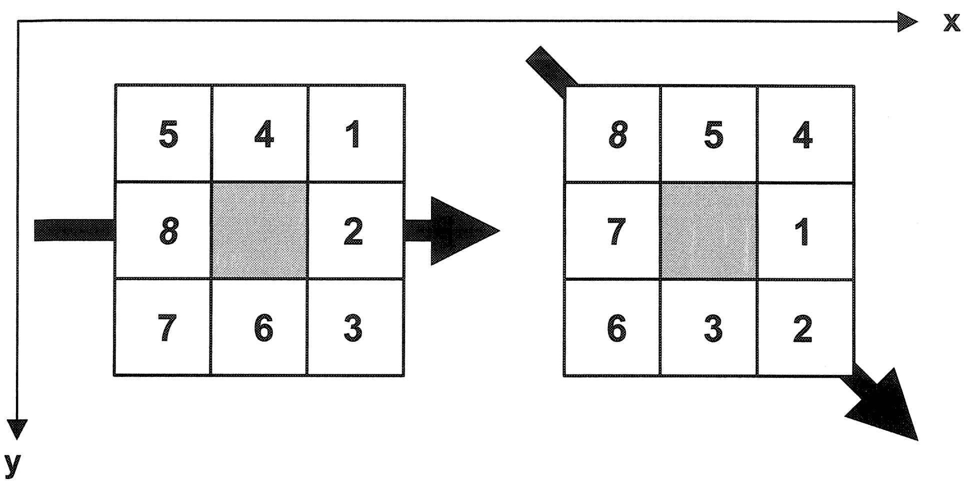


Figure 4-13: Example of direction of search for pixels.

The numbers indicate the order of searching neighbouring pixels.

The contours can be open or closed – if the last extracted point is next to the first extracted point, then the contour is closed. The contours of interest to the system are those corresponding to pages, and they should be closed. Thus if a contour is open, the system checks the last three points on the contour for branches. If one is found, then the other branch is taken and boundary extraction continues with the new branch. If two open contours have coincident start or end points, they are joined.

Live Paper continues by applying mechanisms for pruning contours that are unlikely to represent pages. Very short contours are eliminated. The system also eliminates those contours inside simple closed contours; the internal contours are created by markings on the pages. Figure 4-14 shows the final extracted boundaries for the example input in Figure 4-7; these boundaries are actually stored as chain codes, and not as a two-level image.

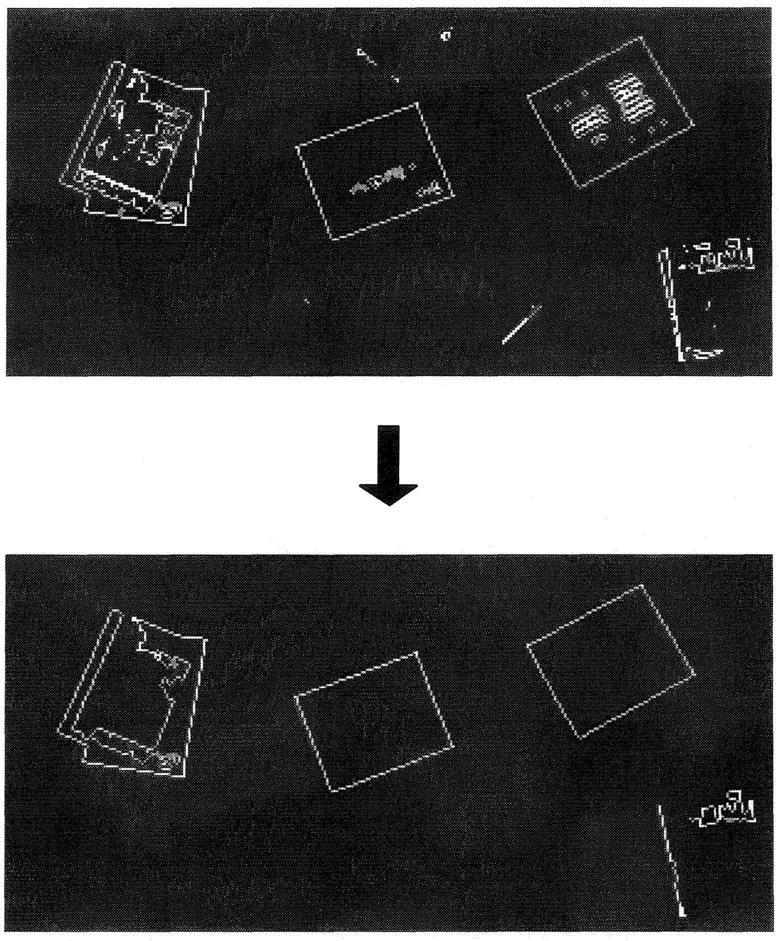


Figure 4-14: Elimination of obvious false boundaries.

A third operation is performed when the overlapping and occlusion feature analysis stage is reached. The system applies a morphological closing operation to a copy of the segmented version of the captured image. Edge detection is then applied to the newly segmented image. If a boundary has no pixel in common with this modified edge image, then the boundary is eliminated.

4.3.3.5 Boundary Feature Detection

Once a boundary has been stored, features that are informative for page finding can be detected. These features can be used to determine whether or not the boundary is a page or part of a page. Two types of features are of interest: corners, and straight-line segments. One strong element of pages is that corners are quite sharp, at 90 degrees. Unfortunately, due to digitization effects, corners will not necessarily appear within a single pixel.

Live Paper starts by identifying the corners in each of the extracted chain codes, using the Freeman-Davis corner-finding algorithm [Freeman 1977]. There are many other such algorithms from which to choose (such as [Teh 1989]), but the Freeman-Davis algorithm is adequate in this case due to the sharpness of the page corners. It is also efficient, working in O(n) time, where n is the number of points in the chain code. This implementation requires only a single pass of the chain code data. The following discussion is included so to help the reader understand how the algorithm finds corners. The mathematical description of a chain code is the one given in section 4.3.3.3.

For a consecutive series of s links terminating at the node to which link a_j points, the line segment connecting the endpoints is given by L_j^s. That is,

$$L_j^s = a_{j-s+1}a_{j-s}\ldots a_j, j = 1,2,\ldots,n. \quad (4\text{-}17)$$

The x and y components of the line segment are

$$X_j^s = \sum_{i=j-s+1}^{j} a_{ix} , \tag{4-18}$$

and
$$Y_j^s = \sum_{i=j-s+1}^{j} a_{iy} . \tag{4-19}$$

The length l_j^s of L_j^s is

$$l_j^s = \sqrt{\left(X_j^s\right)^2 + \left(Y_j^s\right)^2} . \tag{4-20}$$

The angle θ_j^s of L_j^s is the local curvature, and is the measure between the x-axis and the forward direction of the line segment. It is given by

$$\begin{aligned} \theta_j^s &= \tan^{-1}\left(\frac{Y_j^s}{X_j^s}\right), \quad \text{if} \left|X_j^s\right| \geq \left|Y_j^s\right| \\ &= \cot^{-1}\left(\frac{X_j^s}{Y_j^s}\right), \quad \text{if} \left|X_j^s\right| < \left|Y_j^s\right| . \end{aligned} \tag{4-21}$$

The incremental curvature δ_j^s provides some smoothing, and is twice the mean over two adjacent angular differences:

$$\begin{aligned} \delta_j^s &= 2\left[\frac{\left(\theta_{j+1}^s - \theta_j^s\right) + \left(\theta_j^s - \theta_{j-1}^s\right)}{2}\right] \\ &= \theta_{j+1}^s - \theta_{j-1}^s \end{aligned} \tag{4-22}$$

Freeman and Davis introduce the term ‘cornerity’ to indicate the sharpness of a corner. The cornerity at point *j* is K_j, given by

$$K_j = \sqrt{t_1} \times \sum_{i=j}^{j+s} \delta_i^s \times \sqrt{t_2}\,, \tag{4-23}$$

where

$$t_1 = \max\left\{t : \delta_{j-v}^s \in (-\Delta, \Delta),\ \forall 1 \le v \le t\right\}$$
$$t_2 = \max\left\{t : \delta_{j+v+s}^s \in (-\Delta, \Delta),\ \forall 1 \le v \le t\right\}$$

$$\Delta = \tan^{-1}\left(\tfrac{1}{s-1}\right).$$

The t_1 and t_2 values are the contributions of the arms around the corners. The middle term is a measure of slope discontinuity. Thus shallow corners with short arms have low measures of cornerity, while sharp corners with short arms and shallow corners with long straight arms have greater values of cornerity (see Figure 4-15).

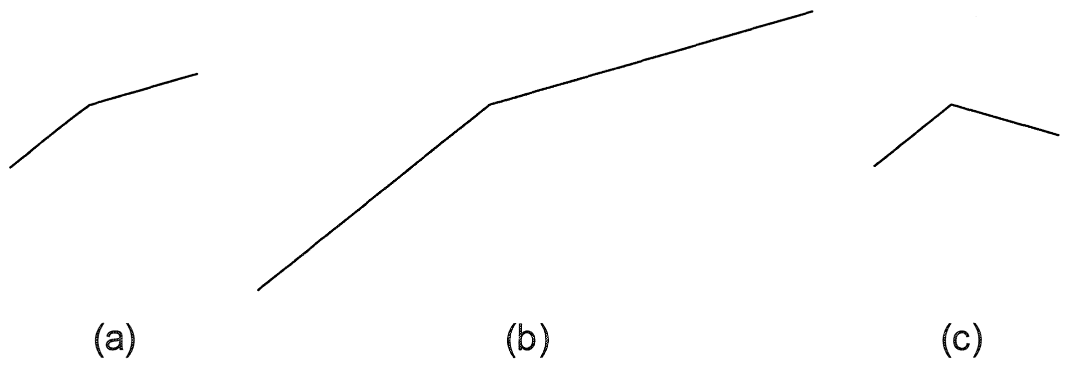


Figure 4-15: Effect of arm length and corner angle on corner definition. (a) Short arms with shallow angle – ambiguous corner. (b) Long arms with shallow angle – corner present. (c) Short arms with sharp angle – corner present.

Live Paper uses the Freeman-Davis algorithm just described to calculate the cornerity value *K* at each point in the chain code. It then scans through the chain code looking for the largest *K* values that exceed a minimum value. If the cornerity at multiple adjacent points exceeds the minimum value, then the point with the largest *K* is used.

The result is a list of corner points for the contour; if the contour has four corners, it is likely to be a page. Since pages have straight edges but a four-corner contour might not, the system checks the straightness of the edges between each pair of consecutive corners. Unfortunately, simply counting the pixels between corner points is not sufficient (see Figure 4-16). Instead, the system finds the pixel that is furthest from the straight line connecting the corners. If the pixel is more than two pixels from the line, then *Live Paper* considers the line to be curved.

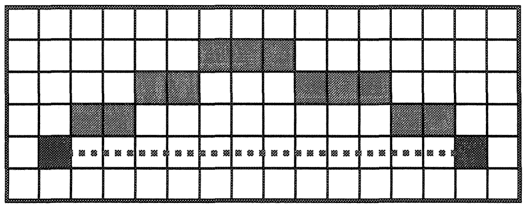


Figure 4-16: Determining if a line is straight.

The number of pixels between end points is the same for both the straight line and the curved line.

The algorithm does not use the perpendicular distance, but either the horizontal or vertical distance based on the slope of the line. The point which is furthest vertically (or horizontally) from the line can be used to determine how much the line deviates from straight. The exact distance of this point is determined afterwards (see Figure 4-17).

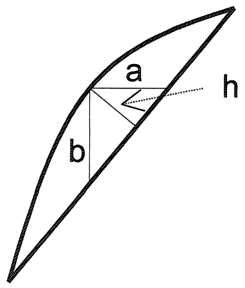


Figure 4-17: Determining maximum deviation from a straight-line.

Finally, the angles of the corners are checked to ensure they are close to 90 degrees. Due to aliasing and perspective effects, the corners can vary between 70° and 110°.

At this point we can return to the question posed in section 4.3.2 as to whether thresholding is adequate for segmentation, and the suggestion in section 4.3.3.4 that deficiencies in segmentation can be remedied by improved boundary analysis. Many test scenes with varied lighting conditions have been examined, and the results show a consistently high extraction of page features.

4.4 FEATURE ANALYSIS

Live Paper feature analysis is new work, since the consideration of going from page features to full desk description is addressed for the first time here. As indicated in Figure 4-3, the features are analyzed in three stages, where each stage has an escalation of feature processing. As each stage extracts pages, that stage removes the page features from the data set.

4.4.1 Escalation Mechanism

To reduce the computational requirements of the *Live Paper* system, it is best for the page finding methods to be done incrementally. This can be based on the complexity of finding particular pages. Single unobstructed pages are relatively easy to find. Overlapping or obstructed pages require much more analysis. In addition, the *Live Paper* system is a dynamic system, as it is constantly maintaining a list of pages that it has found on the desktop. Algorithms can take advantage of this information to search the desktop for existing pages. These pages can even be moving on the desktop, provided the location difference between frames is not excessive.

The escalation mechanism for page extraction begins with isolated page extraction. This stage examines the list of all corners and straight edges. If the features match those of a page, then the stage creates a representative page object, and deletes the associated contour from the boundary set. When this stage finishes, the number of contours has potentially been reduced. This new boundary set is used by the page tracking stage, which removes references to corners and edges that can be matched to stored pages. Then the final stage analysis for occlusion and overlap can perform detailed calculations on a reduced data set.

4.4.2 Isolated Paper

The isolated paper finding mechanism works in two steps. The first step is to analyze each of the extracted contours to determine if it is a page. There are five criteria used by the mechanism to determine whether or not the contour is a page:

1. The contour must be closed.
2. The contour must have four corners.
3. All lines between the corners must be straight.
4. All of the corners must be right angles.
5. All lines between the corners are greater than a miminum length (4 cm).

If a contour matches all of these criteria, then the stage creates a representative page candidate object, and deletes the contour from the set. The stage repeats this process on all contours in the set. When complete, all extracted pages, as well as the unprocessed features, are passed to the next stage.

4.4.3 Page Tracking

The current version of *Live Paper* attempts to track pages between captured image frames. Early versions of the system used only page identification (as presented in Chapter 5) on each frame to determine which pages were still present. However, this was unsatisfactory because transparencies could not move as occluded pages were moved. By taking advantage of pre-existing page location information, the page tracking stage can maintain a lock on most pages without a need for the more computationally expensive algorithms of the dedicated overlap and occlusion finding stage.

To determine if a page is still present, *Live Paper* uses the page corners as indicators. This occurs in two passes: the first pass is concerned with full pages, and the second pass analyzes the remaining boundary features.

On the first pass, the stage checks each of the extracted full-page candidates against the list of pre-existing stored pages. If the extracted page is near to the position of a stored page, then a match has been found. The existing page is transferred to the list of current pages for the new image frame; all of the associated information of the page, such as its transparencies and identification images, are also transferred. The page position is updated if the page has moved. The likelihood of a mismatch is very low because all four corners of the candidate must be near the four corners of the stored page. If the candidate page cannot be found in the stored list, it is tagged as a new and unknown page. *Live Paper* will then determine if it has a record of the page in its database (see Chapter 5).

On the second pass, *Live Paper* analyzes the remaining pages in the pre-existing page list. For each page in the list, the system looks for boundary corners that are close to the position of the pages corners. If two or more corners match the page, that page is considered to be still on the desk. Two tolerances are used: a loose one to determine if the corner matches, and a tight one to determine if the corner has moved. The tight tolerance eliminates any jitter due to digitization effects in the captured image.

The frame rate of *Live* Paper affects the rate at which a page can be tracked. In the development system, which has a frame rate of about 3 fps, the maximum speed at which paper can be moved is 15 cm/s.

4.4.4 Paper Overlap & Occlusion

4.4.4.1 Perceptual Occlusion

Perceptual occlusion [Saund 1999] investigates the manner in which boundaries form visual groupings such that they appear to belong to overlapping surfaces. It builds on previous work in the areas of gestalt grouping [Alquier 1996] and perceptual grouping [Feldman 1995].

One of the key analysis aspects of perceptual occlusion is discerning whether two boundaries are in alignment accidentally or because they are the same edge that has been occluded by another surface. To do this, Saund creates a graph of nodes and links. Boundaries between surfaces, and junctions between boundaries, are both nodes. Alignment links and coincidental links are used to join the nodes. Surfaces are found by analyzing the graph based on the calculated energies of position, contour smoothness, surface colour, and figural convexity. Other energies are possible. The lower the total energy of a proposed global solution, the more likely that solution is the correct one.

This perceptual occlusion work and its predecessors were the inspiration for the development of a new mechanism for interpreting overlapping and occluded pages in *Live Paper*. The mechanism takes advantage of the simpler nature of the page shapes to reduce the complexity of the graphs in the perceptual occlusion work, and then to extend the idea in a new direction, as explained below.

4.4.4.2 Overlap and Occlusion Analysis in Live Paper

On any normal office desktop, paper is stored in stacks, and often overlaps. The page on which the user is focusing is often partly occluded. Figure 4-18 shows the series of modules developed to analyze those boundary features that were not removed in the page tracking or isolation page detection stages. Analysis is based on a graph of nodes and links, where the nodes are straight-line segments, and the links are relationships between the segments.

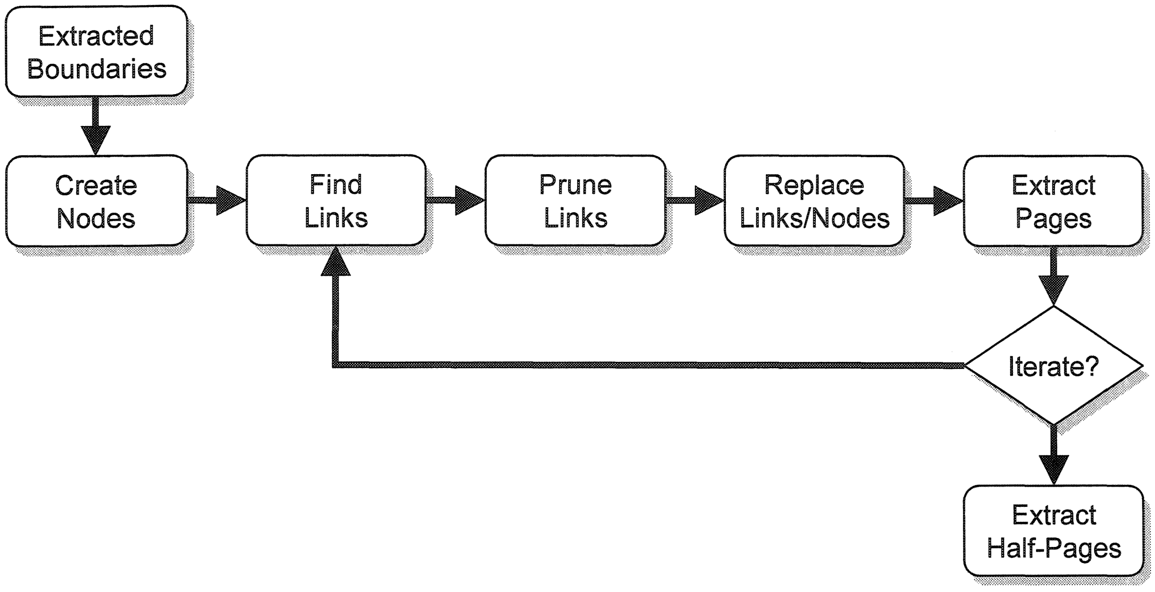


Figure 4-18: Block diagram of algorithm for page overlap and occlusion.

The following sections describe each of the modules in Figure 4-18 in detail using a single example input file. The extracted boundaries are created from the image in Figure 4-19.

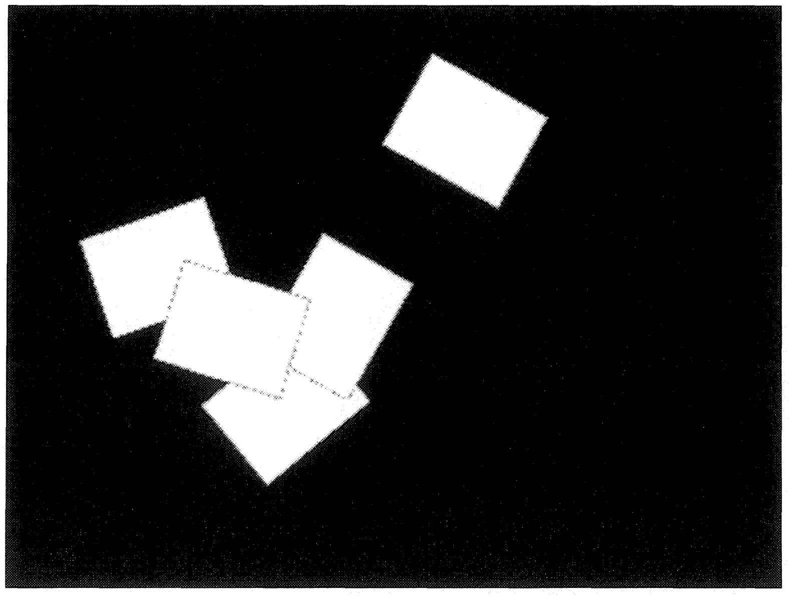


Figure 4-19: Original generated image of five randomly placed and rotated pages.

4.4.4.3 Create Nodes

Using information from the boundary analysis stage, this module creates a set of nodes. Each node represents a straight-line segment, which separates a desk region and a paper region. The node points in a particular direction: from the perspective of an observer at the start point looking towards the end point, the page lies to the right of the segment and the desk lies to the left (see Figure 4-20). The orientation of the node does not affect this geometry.

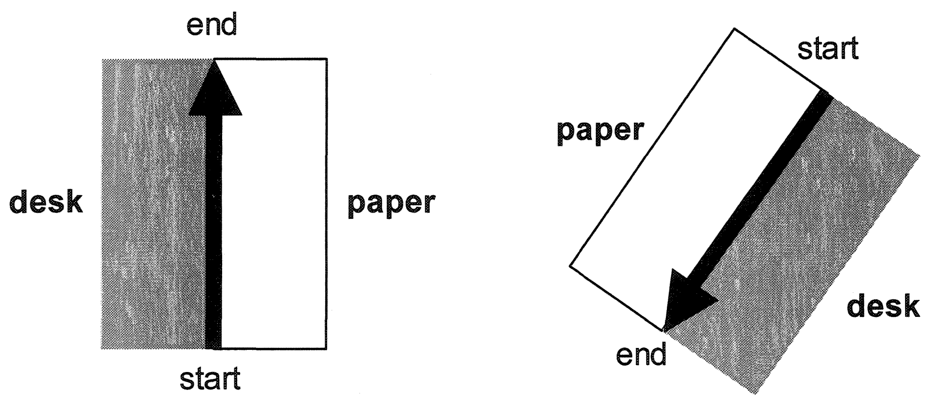


Figure 4-20: A node signifies the boundary between the desk and paper.

Due to the way in which the scanning of the segmented image is performed, all boundaries are extracted in a clockwise manner. Thus forward parsing of the contours will ensure that straight-line segments are extracted from their starting points to their end points. The image in Figure 4-21 shows the output of plotting the 18 nodes found from the boundaries of Figure 4-19. Each node has been plotted as a straight-line segment between its start point and end point.

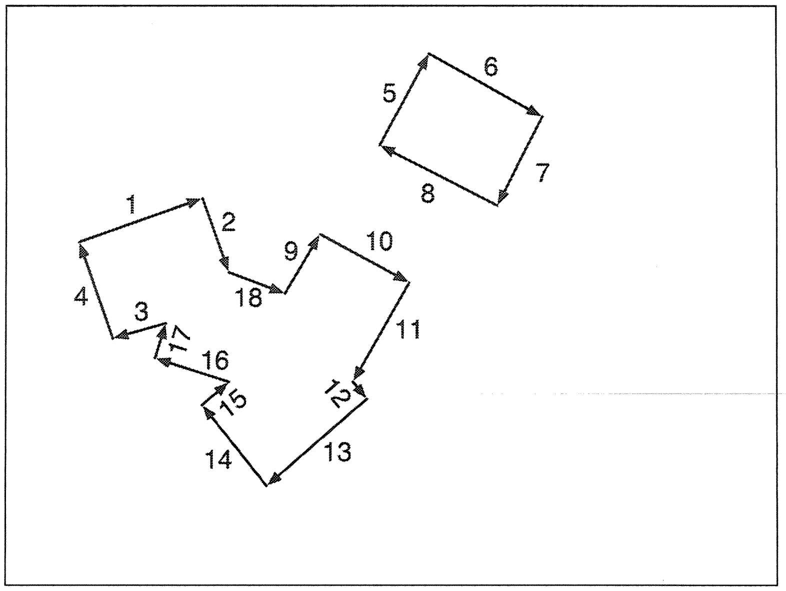


Figure 4-21: Nodes plotted as straight line segments.

The Create Nodes module has extracted 18 nodes from Figure 4-19.

4.4.4.4 Find Links

On each iteration of the overall algorithm, this module establishes links between the straight-line nodes. There are only three links that are of interest to paper detection: *corners*, *collinear*, and *right angle* (see Figure 4-22). These three alignments can occur around the boundaries of a single occluded or overlapped page. The module checks each node against all other nodes in the graph to determine which of the three links apply. Each node can have more than one link after this module is finished; however, ultimately only one link will be chosen as valid.

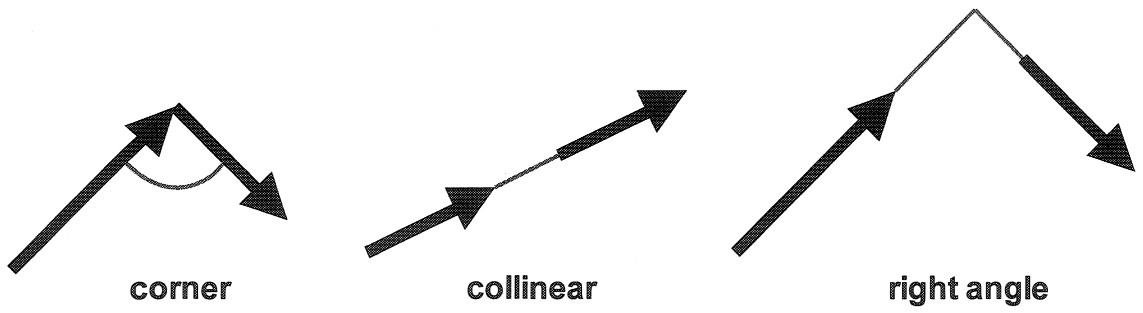


Figure 4-22: Valid links between straight-line segment nodes.

The relative orientation of the nodes is important. Figure 4-23 gives examples of node alignments that are similar to the valid alignments in Figure 4-22, but that cannot belong to the boundary of the same page. They only occur when two pages accidentally align.

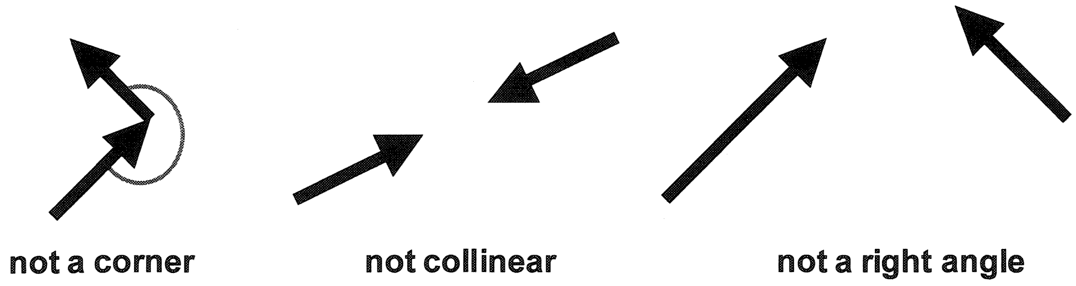


Figure 4-23: Node alignments that do not have associated link types.

Corners occur when the end-point of one node is common with a start-point of another node, and the angle between them is 90 degrees. However, the second node must lie to the right (as shown in Figure 4-22). Thus in Figure 4-21, there is a corner link from node 1 to node 2, but not from node 18 to node 9. Collinear means that two nodes point in the same direction along the same line. In Figure 4-21, nodes 9 and 5 are collinear. A right angle means that two segments lie in such a way that a page corner might lie at their intersection. In Figure 4-21, there is a right angle from node 2 to node 3.

For each node-link-node combination, the module generates an inverse goodness-of-fit (IGF) measure that is equal to the difference in area between the ideal case and the actual case. A small IGF indicates that the node-link-node combination is close to the ideal. For corners and right angle links, the IGF is equal to the area swept out by the longer of the two nodes. In the case of collinear links, the calculation is based on the orientation of the nodes. Both calculations are detailed below. The next module, *Prune Links*, uses the IGF to keep or reject links.

For corner detection, the algorithm checks each node to determine if its end point is adjacent to the start point of another node. If so, then it ensures that the angle is approximately 90 degrees – this allows for some variation due to digitization effects. Smaller lines are allowed more digression, as single pixel errors in the location of the start or end points would cause significant deviation in the direction of the node.

In calculating the IGF, the algorithm uses the longer of the two nodes to determine the area. Thus as nodes lengthen, they have a stricter requirement to be close to 90 degrees. From Figure 4-24, assume l_1 and l_2 are the lengths of two nodes that meet at a single point. If $l_1 > l_2$, and the angle of error between a right angle and the actual angle is given by η (in radians), then:

$$IGF = 0.5 \times \eta \times l_1^2 \tag{4-24}$$

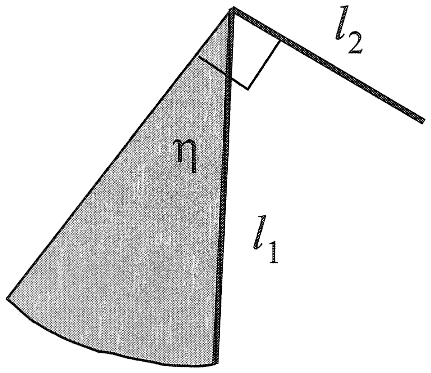


Figure 4-24: Measuring the IGF of two line nodes.

When two nodes meet at a corner, the inverse goodness-of-fit measurement is calculated based on the difference between the actual and ideal angles.

For right angles, the calculation of the IGF is similar, except that the line lengths are measured with respect to the projected intersection point. The module also checks that the lengths are reasonable for 8.5 x 11 inch pages. Using an approximate size of 5 mm per pixel, the module immediately rejects links with excessive lengths. Then if the IGF is below a threshold, the link is accepted.

For collinear links, the nodes must point in the same direction with a reasonable total length, similar to right angles. This length restriction eliminates some accidental alignment. An example where two nodes falsely appear to be collinear would be when the top of one page aligns with the bottom of another page.

The module finds collinear links by checking all pairs of nodes to see if their slopes are approximately the same. The tolerance used depends on the length of the nodes, with shorter nodes having a higher tolerance value. Aligned nodes are then checked to ensure that they are correctly positioned – that is, approximately collinear. If so, then the IGF for the collinear line is calculated. Similarly to the corner link calculation, the IGF for collinear lines uses the deviation areas (the shaded areas in Figure 4-25) of nodes from the collinear line. There are three possible orientations of the nodes, and thus three different calculations. In the case of perfect collinearity, the IGF is 0.

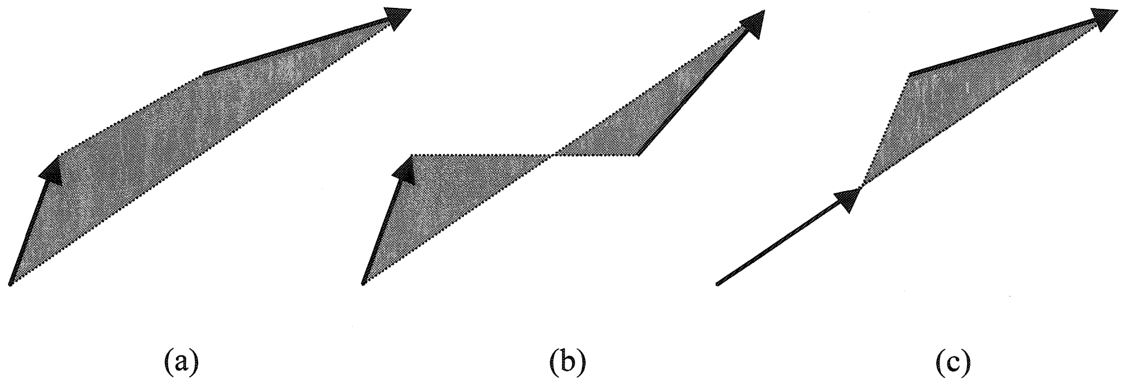


Figure 4-25: Deviation areas for collinear links.

(a) Both nodes lie to the same side of the collinear link, and so the area is a trapezoid. (b) The nodes lie on opposite sides of the link; the total area is the sum of the two triangular areas. (c) One node lies along the link, and so only one triangular area needs to be calculated.

Figure 4-26 shows the eighteen links found for the nodes of Figure 4-21. The thirteen corner relationships are denoted with a thin green line. Collinear links are shown with a dark blue dashed line – there is a link from node 15 to node 9, and from node 9 to node 5. The three right angle links are shown as purple lines in an L-shape. These connect from node 2 to node 3, node 15 to node 12, and node 17 to node 18.

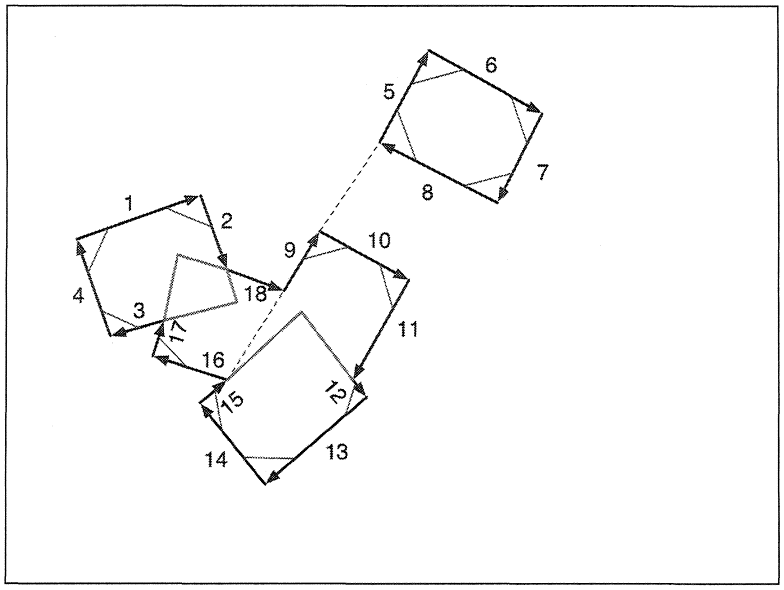


Figure 4-26: The eighteen links found for nodes in Figure 4-21.

There are thirteen corners, two collinear links, and three right angle links.

The output of this stage includes real relation links between nodes as well as many incorrect links due to accidental alignments. The next stage tries to find the best links, which are ideally the correct ones.

4.4.4.5 Prune Links

Any node that is part of a page must ultimately have a single forward link to another node. Nodes that have multiple links must be pruned. The *Prune Links* module assumes that only one link is true and attempts to determine which one.

The core research problem of the overlap and occlusion work is to choose the best link. If this module selects an incorrect link, then the corresponding page will go undetected. However, the *Find Links* module completed most of the work of determining the best link when it generated the IGF value. The pruning module uses this value to sort the links. Starting with the link with the highest IGF value (that is, the poorest link), the module checks the nodes at both ends. If the start node is the source for another link, then the current link is not the best link, and is deleted. For example, if a collinear link and a right angle link both start from the same node, then the module selects the poorer link first, and deletes it. If the end node is the destination for another link, then again the current link is deleted. If neither node has a competing link then the link under review is optimal, and is kept. When the module completes, only one link remains between any two nodes, and no node has more than one link starting or ending at it.

In Figure 4-27, the above algorithm has been applied to the graph, and both collinear links have been removed. Although this example doesn't show it, the *Prune Links* module can also choose remove corner links in favour of collinear or right angle links.

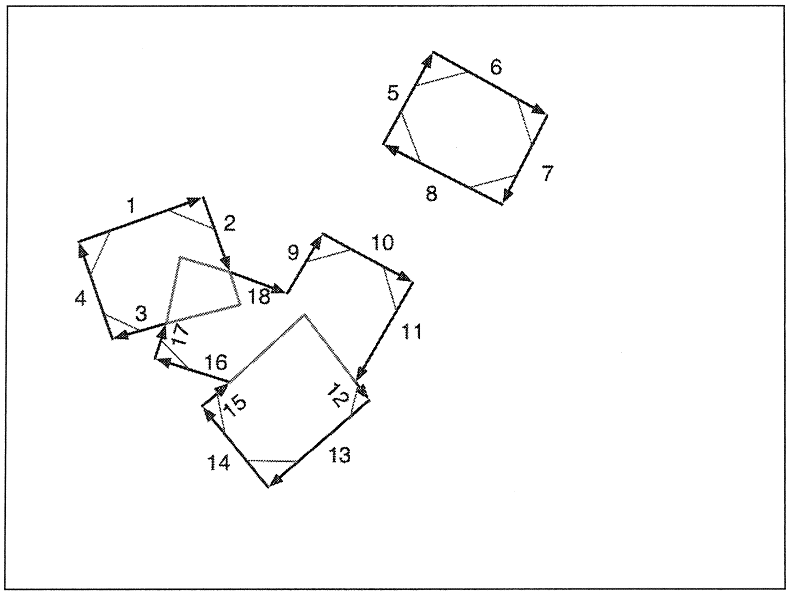


Figure 4-27: Links remaining from Figure 4-26 after pruning.

4.4.4.6 Replace Links & Nodes

This module selectively replaces links and nodes in the graph. Because the pruning method is comprehensive, each node will have at most a single forward link to another node, and a single link that starts at another node. This module replaces a node-link-node with either one node (in the case of collinear) or two nodes and a corner link (for a right angle link). In the second case, the module has to estimate the location of the corner, and extend the nodes appropriately. A corner link is never replaced.

Figure 4-28 illustrates the replacement of links and nodes in the example graph. The three right-angle links have been replaced with corner links, and the adjacent nodes have been modified appropriately. For example, nodes 2 and 3, and their right angle link, have been replaced with nodes 19 and 20 respectively, with a new corner link.

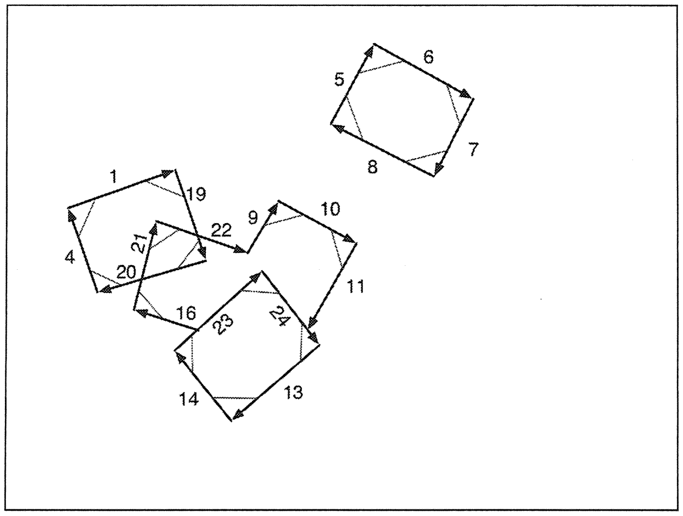


Figure 4-28: Replacing links and nodes.

All collinear and right-angle links, and their adjoining nodes, have been replaced with corner links and resized nodes.

4.4.4.7 Extract Pages

A page is only extracted if there are four nodes connected by corner links. Each node must have a single corner link to the next node. Once a page is extracted, all of the associated nodes and links are removed from the graph. Figure 4-29 shows the nodes and links that were extracted from Figure 4-28. Three pages were found in the first iteration of the algorithm. The page represented by nodes 13, 14, 23, and 24 was underneath two other pages in the original image (see Figure 4-19). It was found in the first pass because three corners were visible, and the algorithm was able to reconstruct the hidden corner.

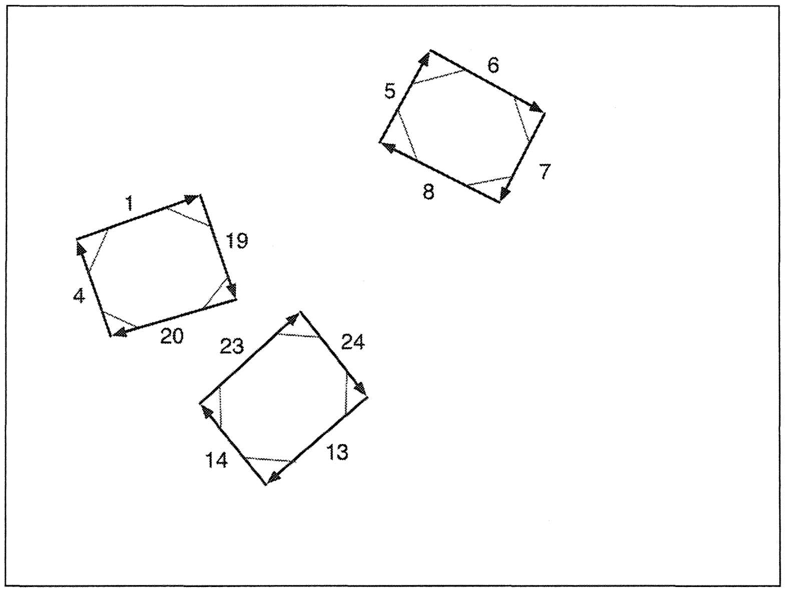


Figure 4-29: The three full pages extracted from the graph.

Each page consists of four nodes (edges) and four links (corners).

4.4.4.8 Iterate

Not all corner links can be found on a single pass through this stage. Some corners are only detected when nodes have been modified by the replacement of collinear or right-angle links. Sometimes corners are found to be false after node modification. This module checks whether the stopping conditions – that the nodes and links in the graph have remained the same for two subsequent passes – have been met.

4.4.4.9 Extract Half-Pages

After all full pages are extracted, the algorithm checks for the presence of partial pages. To do this, the algorithm must know the sizes of pages for which it is checking. Of the previous modules in the algorithm, only *Find Links* required some knowledge of the page sizes, and that was to eliminate excessively long accidental alignments. This module needs the valid sizes of pages in pixels, which can be acquired through the registration process presented in Chapter 6. After registration, it is possible to determine how many pixels in the captured image correspond to a unit of measurement on the tabletop. For the Live Paper set-up, one inch on the tabletop corresponds to five pixels in the image, so a 8.5 x 11 inch page is two sides of 43 pixels and two of 55 pixels.

In the example of Figure 4-19, two pages are positioned such that one edge is totally overlapping another page (Figure 4-30) – the half-page extraction module can detect this situation.

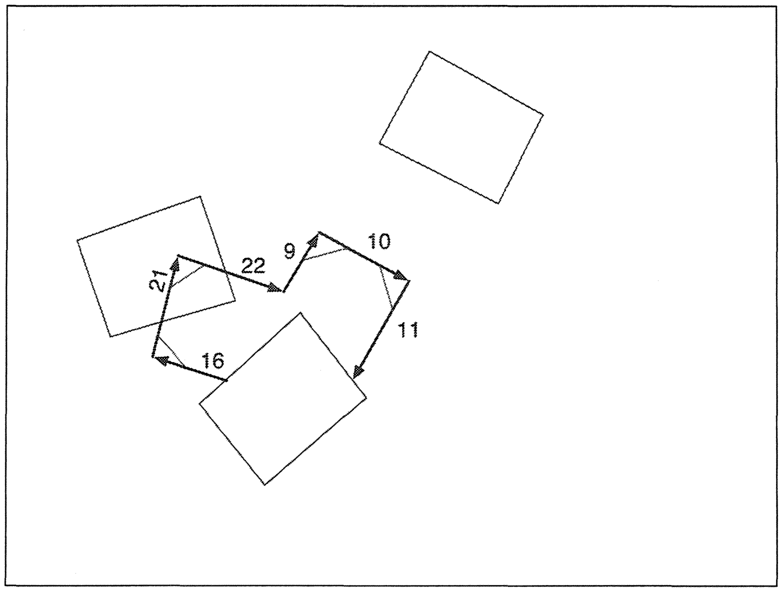


Figure 4-30: Of the original five pages, two pages could not be detected by the previous modules.

Half-page extraction requires that three sides (nodes) are present, and that two corners connect them. The middle node must correspond to one of the page sides (either 43 or 55 pixels long); the two adjacent nodes must not be longer than the remaining side. Thus if the middle node corresponds to the 43 pixel (8.5 inch) side, then the other two nodes must be 55 pixels (11 inches) long or less. If this condition is met, then the module assumes that it has found a half-page, and extracts it. The module generates the full-page by extending the adjacent nodes to be the correct length, and creates a new node for the missing side. The module inserts corner links between the adjacent sides and the missing side.

If other page sizes are present on the desktop, then the utility of this step is compromised. For example, an occluded 11 inch by 14 inch page could be falsely extracted as a letter sized page. However, as shown in the next section, the potential success rates of finding pages increases dramatically when the half-page extraction module is included. For example, in the case where 20 simulated pages are randomly placed on a 4 x 5 foot desktop, the success rate rises from 21% to 54% with the half-page extraction module activated. Figure 4-31 shows the successful extraction of the two half-pages.

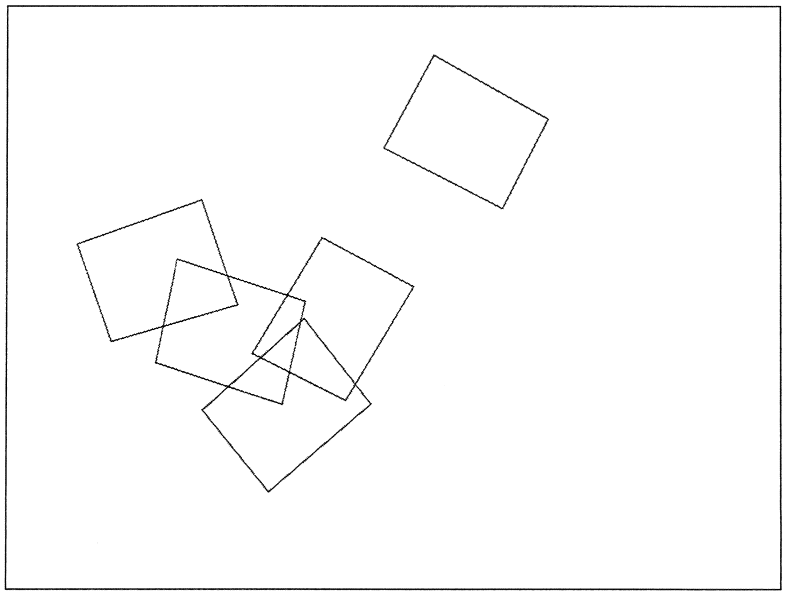


Figure 4-31: All five pages have been found succesfully.

Attempting to perform an extraction with less information than three sides is not useful unless the system can determine more details about the rest of the image. The current analysis does not distinguish between image regions that are part of the desktop, and regions that belong to a user's body or other occluding objects. Thus the system does not have enough information from a single corner to determine whether the other corners are hidden due to overlap, or due to occlusion. Thus it cannot determine where the page boundaries are located. Admittedly, the same situation exists with half-pages, but a better guess can be made.

4.4.4.10 Representative Examples

This section presents two examples. The first (Figure 4-32) shows the results of extracting 15 simulated pages using the algorithms for finding overlapping and occluding pages. The second example (see Figure 4-33) shows how an image of 15 simulated pages and 10 occluding objects is analyzed.

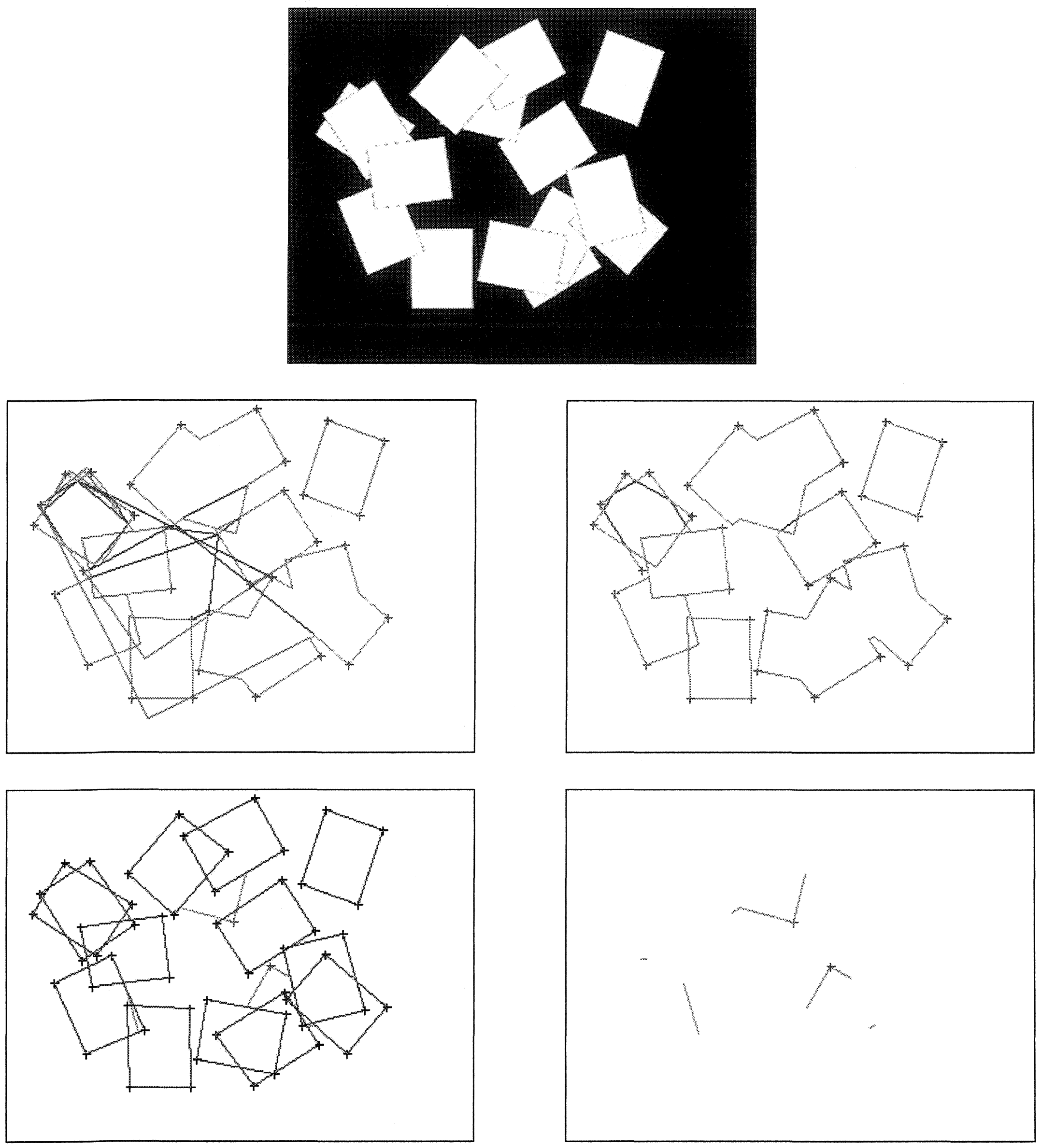


Figure 4-32: Page-finding example where 13 of 15 pages have been located.

Image description by row: (a) Generated image with 15 pages. (b) Output of Find Links module. (c) Output of Prune module. (d) Thirteen pages found and extracted; two pages had too much overlap for the algoithm to compensate. (e) Nodes and links that were not used by the algorithm..

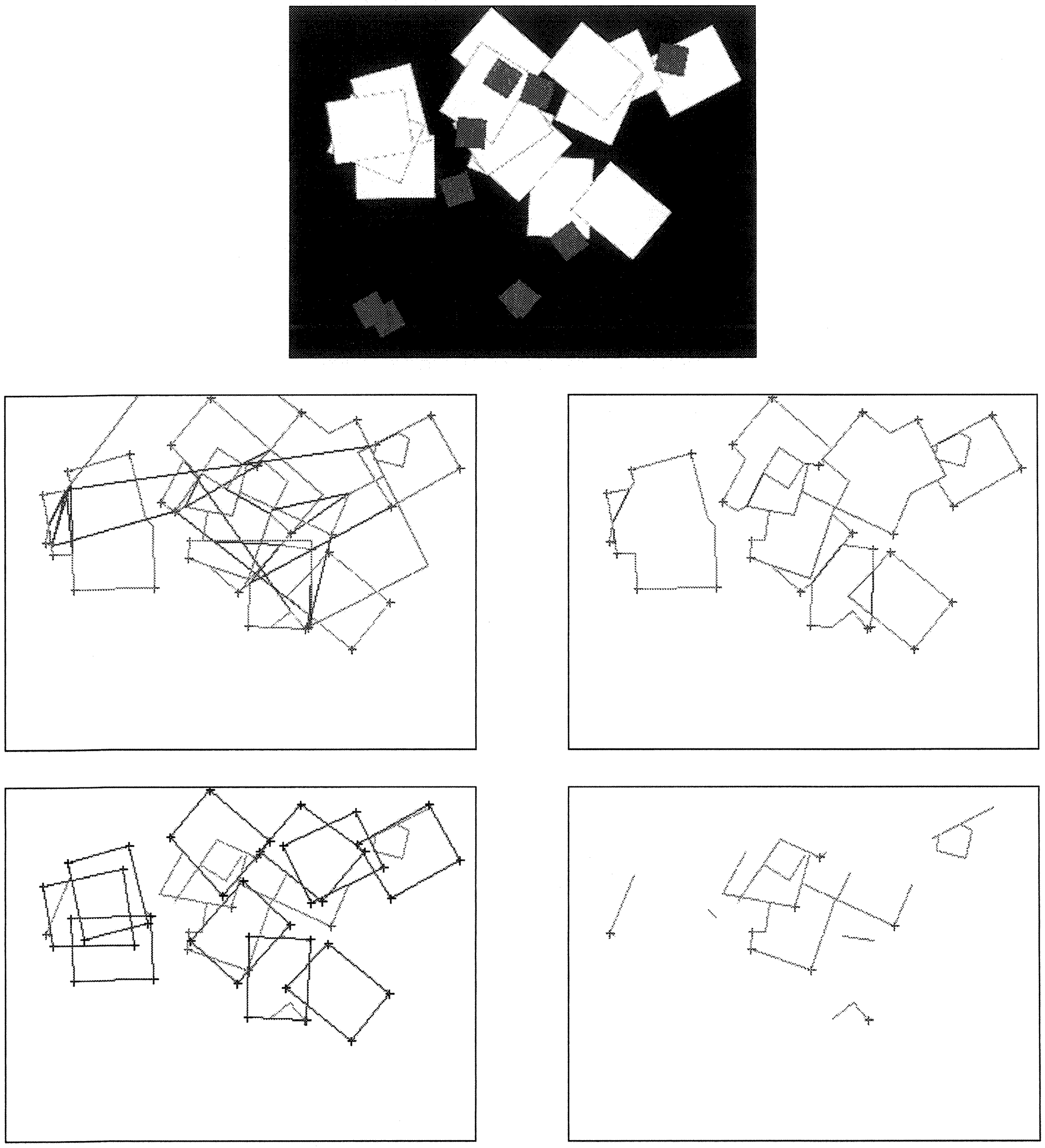


Figure 4-33: An example of page finding with overlapping pages and occluding objects. Image description by row: (a) Generated image with 15 pages and 10 overlapping objects. (b) All links found by the algorithm on the first iteration. (c) The remaining links after the pruning operation. (d) There have been 10 pages extracted (using iteration). (e) The links and nodes not used by the algorithm – some of these elements have been generated by the replace stage.

4.4.5 Overlapping Pages Experiment

The following experiment compares the success rates of using the isolated page-finding algorithm, the iterative part of the overlapping and occluded pages algorithm, and the full algorithm including half-page detection.

The experiment generates images of a number of blank letter-sized pages (from 1 to 20) that are randomly rotated and placed on a simulated desktop. The experiment uses a uniform distribution for both the rotation and position. The size of the desk is equivalent to a 5 foot, 4 inch by 4 foot desktop. As the number of pages increases, the number which overlap also increases. For each image, the experiment applies all three algorithms separately. The numbers of found pages are recorded after some additional analysis that ensures the pages are letter-sized. For each given number of pages, the experiment is repeated 500 times, for a total of 10000 generated images.

This method has the benefit of generating a large number of artificial scenarios very quickly. It is suitable for showing how successful the algorithms are for random locations of pages. The 'Isolated Page Algorithm' is actually a modification of the overlapping and occluded algorithm, but is comparable to the existing isolated page finding algorithm.

Figure 4-34 presents the experimental results of the investigation. All three techniques are successful at finding isolated pages. However, the performance of the isolated page technique degrades rapidly as more pages are added, with a success rate of 0.509 at 4 pages, and 0.028 at 20 pages. When the algorithm for extracting overlapping pages is used without the final half-page detection stage, then the success rates improves such that half the pages are found for up to 10 pages on the desk. At 20 pages, the success rate is 0.214. Incorporating the half-page extraction algorithm increases the success rate such that at 20 pages on the desktop, over half of the pages are still detected (a rate of 0.539).

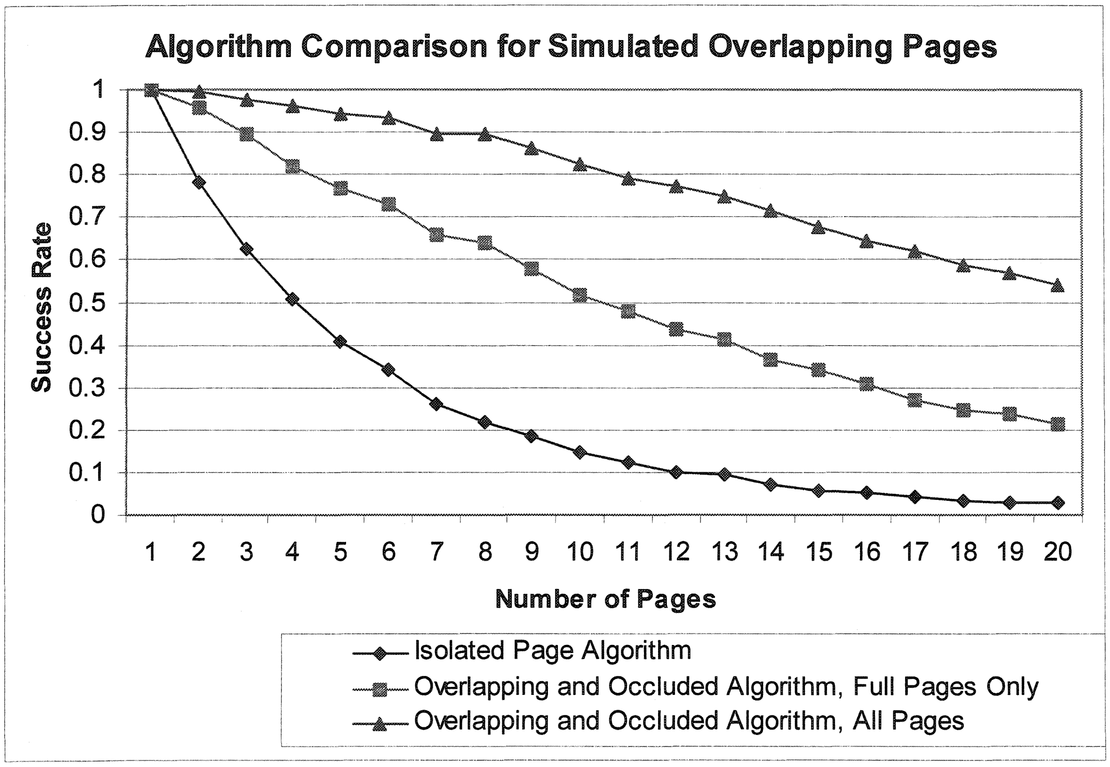


Figure 4-34: Experimental results comparing the success rates of the three algorithms.

The experiment only simulates a desktop – in the real world, the user is unlikely to distribute pages totally at random. However, the experiment does provide a good basis for comparing the variations in algorithms. The success rate of the half-page extraction algorithm demonstrates that the algorithm is effective.

4.4.6 Overlapping and Occluded Experiment

This experiment is similar to the overlapping experiment of section 4.4.5, with the addition of occluding elements. The size of the simulated desk and the distribution of pages are the same. Each element is a dark square that is equivalent to 4 inches by 4 inches in size. The experiment randomly rotates and places the elements after placing the pages. Not all of the elements actually block pages, and some blocks overlap. Although the shape is representative of only a small proportion of real-world objects that could occlude the paper, the elements do represent the objects that would likely cause the most problems for the full procedure. The images in Figure 4-35 show an example of a generated overlapping and occluded page set, and the set of extracted pages and remaining nodes. When the generated image is thresholded (using a fixed threshold), the occluding elements appear to be part of the background.

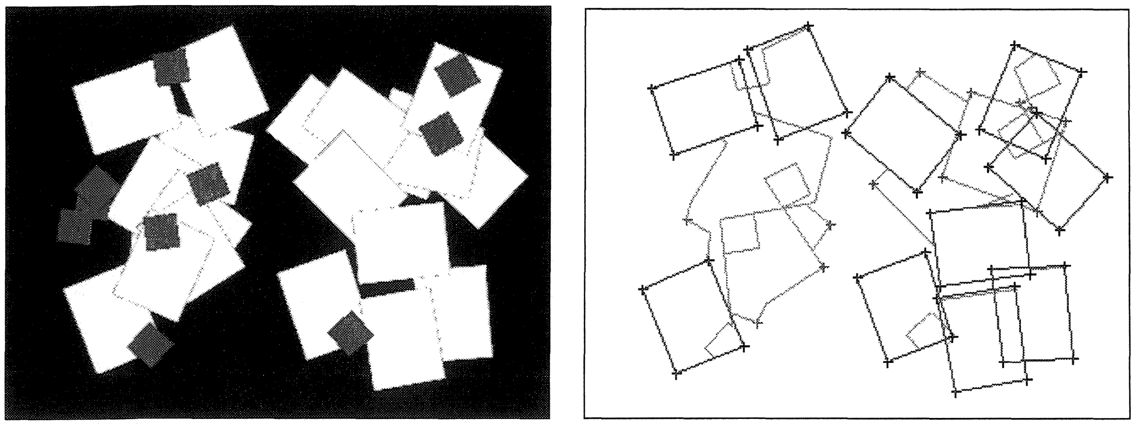


Figure 4-35: Generated overlapping and occluded page set.

(a) The original image with 20 pages with 10 occluding blocks. (b) The final data, including 10 extracted pages and many unused nodes.

The results in Figure 4-36 show that approximately half (0.494) of the generated single-page images contain an occluded page. Many of the failures in the final extraction algorithm (both full pages and half pages) are due to accidental alignments of the blocking element with the page. In the pruning stage, which removes links based on the local measure of inverse goodness-of-fit (IGF), these situations are not corrected.

However, the rate of performance decrease for both the overlapping-page and all-page extraction algorithms is shallow, so that at 20 pages on the desk, the success rate of the full algorithm is 0.442 (compared to 0.549 in section 4.4.5).

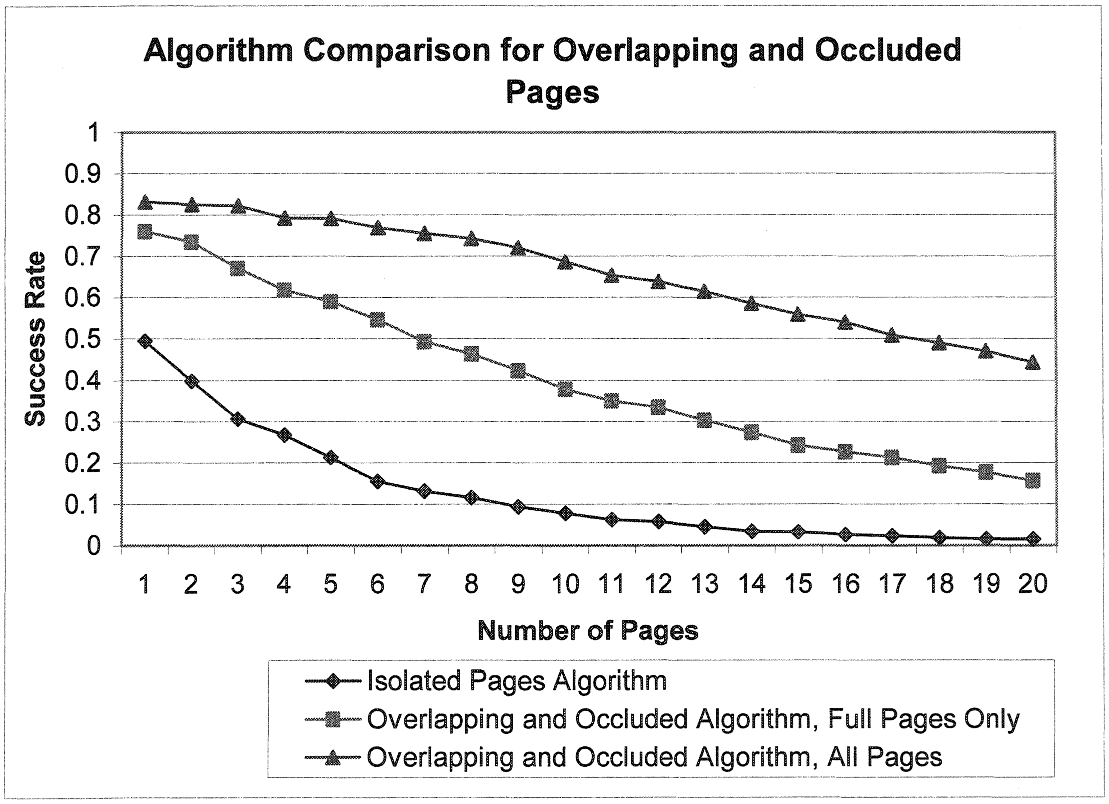


Figure 4-36: Success rates of the three algorithms for overlapping pages and occluding objects.

4.4.7 Comparison of Simulated Pages with Real Pages

The previous two experiments demonstrated the effectiveness of the three algorithms over large sets of simulated pages. This experiment confirms that these results are valid for real pieces of paper. Sets of images with between 1 and 10 pages are created and stored; each set contains 100 images. The distribution and rotation of the pages are uniform over a tabletop that is smaller - 5 feet by 2 feet 6 inches – than in the previous experiments, but which matches with the size of the real tabletop. An edge detection algorithm is also applied to each image, with the resulting image stored.

Figure 4-37 shows two pairs of sample images - the generated one is on the left, and the real tabletop is on the right. During image capture, the capture application shows the generated image next to a live video stream of the tabletop with the edges transparently overlaid. As he positions the pages on the tabletop, the experimenter views the alignment in the application on-screen. When the pages line up correctly with the edges, an image from the stream is captured and stored, and the next generated image is loaded. In a few cases, not all of the generated pages are visible - the remaining pages are hidden underneath. In those cases, the real pages are also placed underneath the visible pages. Five representative sets of real-page images are captured; for the cases where 1, 2, 3, 5, or 8 pages are present on the desk.

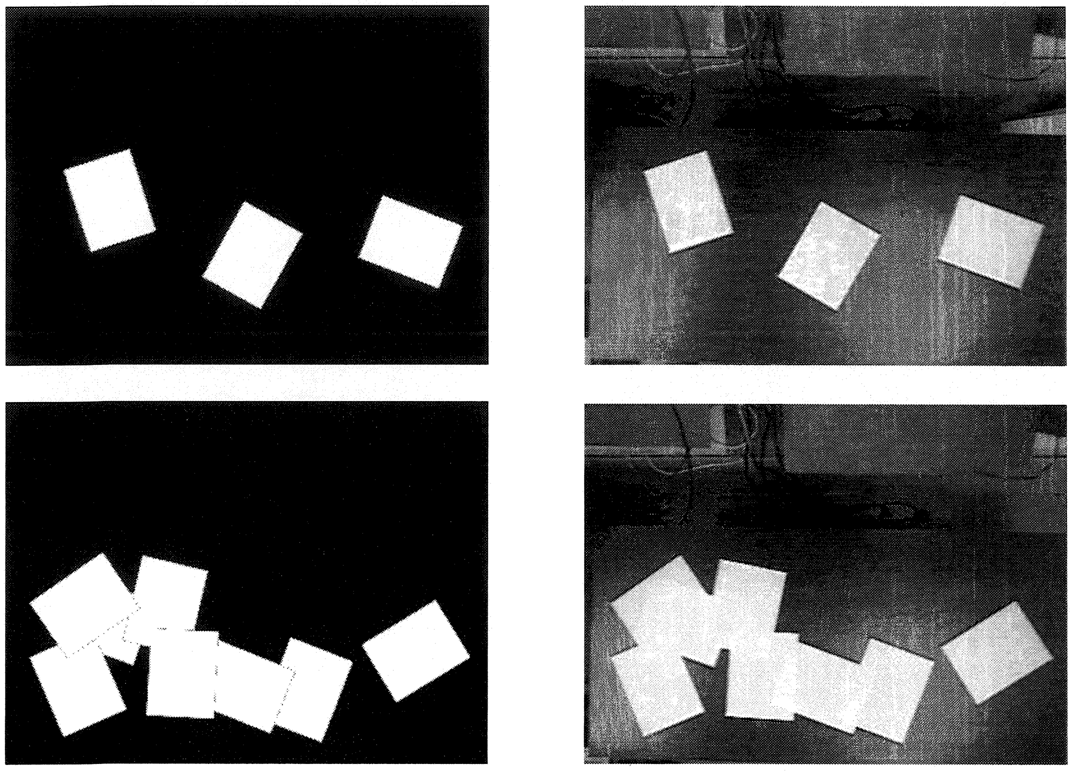


Figure 4-37: Images of simulated and real paper.

The simulation images on the left were used to create the real-world layouts on the right.

The graph in Figure 4-38 shows the results for the three algorithms – isolated page finding, overlapping and occluded page finding, and overlapping and occluded page finding with half-page detection – for both the generated images and the real-world images. For each trend line, the results for 5 sets of pages are plotted. The results for both cases match well. When compared to the results in Figure 4-34 and Figure 4-36, the success rate appears to decline much faster for a given number of pages. This is due to the smaller tabletop size, which increases the probability of overlap.

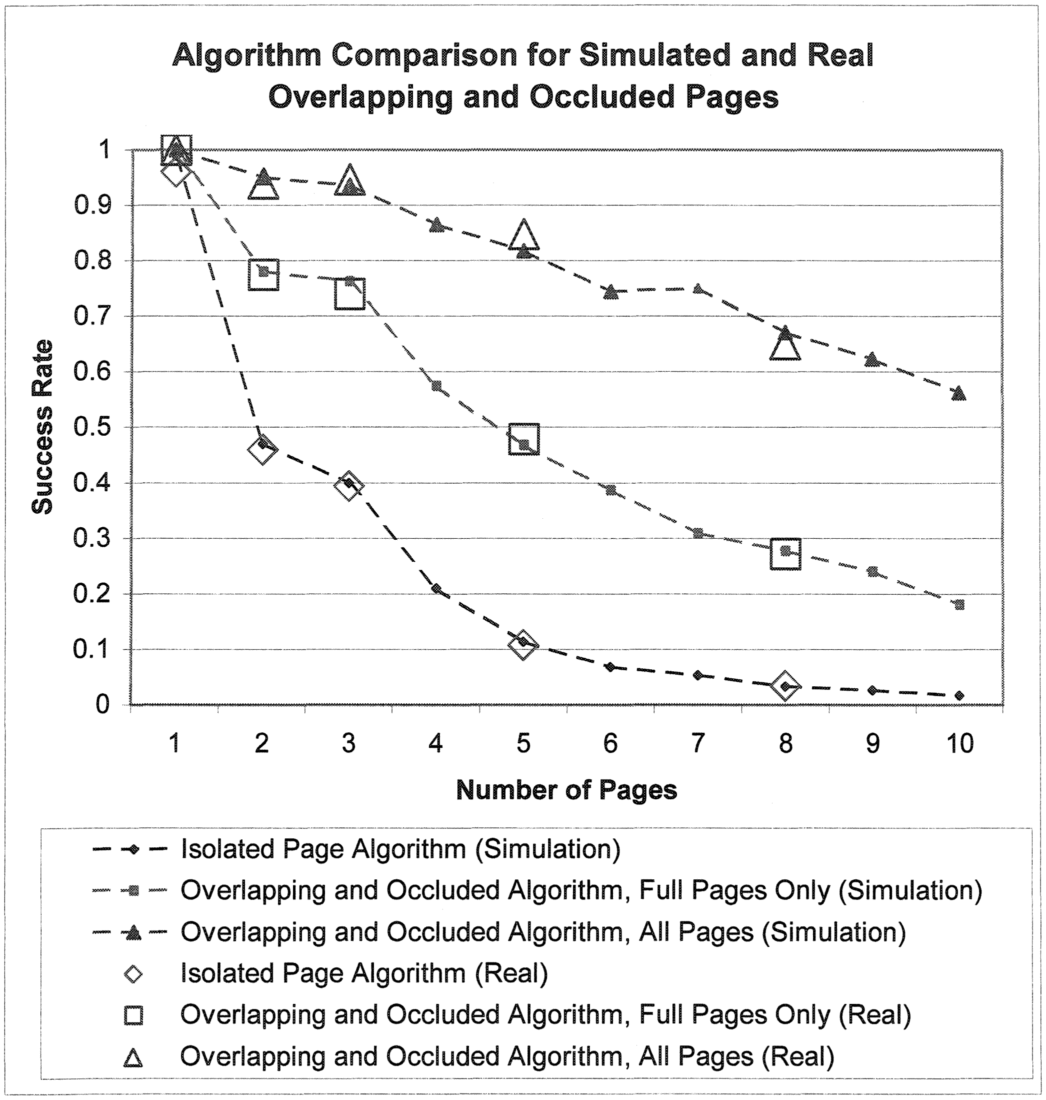


Figure 4-38: Algorithm success rates for simulated and real desktops.

The differences in success rates could be due to several issues. The alignments between the simulated and real pages are not perfect. Also, a fixed threshold is used to segment the artificial pages, but for the real pages, the Otsu method was used. Despite these differences, the agreement between the two sets is good, verifying that the results from the previous two sections also apply to real paper on real tabletops.

4.4.8 Discussion

The investigation of the overlapping and occlusion detection process raised a number of interesting issues that are suitable for further research. The calculation of the IGF is based purely on local characteristics of the nodes, without any consideration for the overall shape of the page. However, since *Live Paper* would be able to infer the sizes of pages from existing ones, the system could use this information to modify the IGF of links according to how well the associated quadrilaterals fit the pages. Alternatively, it would be possible to extract all possible page-like shapes from the graph, and apply an IGF-like calculation to the entire shape. These approaches would not invalidate the work presented here; the flexibility of the current approach allows *Live Paper* to find new page sizes.

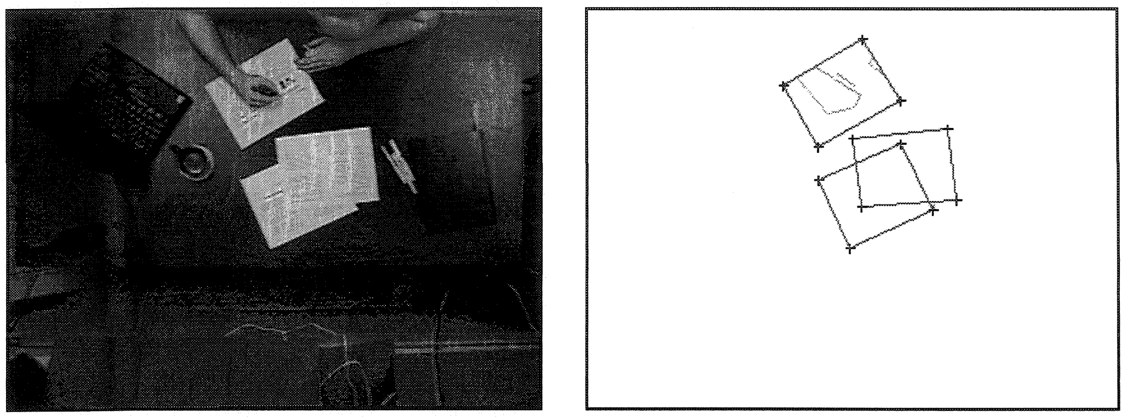
Another issue for further investigation is a means to allow for 'undoing' of a replace stage operation. In the current implementation, once the replace stage finishes, the original node information is lost. One possibility would be to create a new graph at the replace stage based on the new and unmodified nodes, and extract pages based on this new graph. The re-iteration stage would then re-integrate the graphs by removing the original nodes that correspond to extracted pages. Then the entire graph would be re-analyzed from the create links module. This would permit the prune module to make mistakes without necessarily losing the ability to extract a particular page.

Notwithstanding these possible improvements, the new method outlined in this chapter does successfully find many of the pages scattered on a tabletop in a random, chaotic fashion.

4.5 VERIFICATION WITH REAL PAGES

In addition to the experiment in section 4.4.7, images of real pages on a tabletop were digitally captured in order to judge the performance of the full page-finding algorithm. Unlike the previous experiments, these are simply representative examples. All of the pages have markings or printed material. Figure 4-39 to Figure 4-42 present a series of example images captured with the *Live Paper* video camera of a real tabletop with real paper. Key stages of the analysis are shown. The results show the overall success of the methods developed in this chapter.

In Figure 4-39, the first image shows an image of the tabletop that is being used to physically annotate some paper. In addition to the user's hands and arms (which are occluding one page), a variety of other objects are present. The second image shows the result after masking the first image for the tabletop area, and then applying the methods of this chapter. All three pages have been found.



(a) (b)

Figure 4-39: Verification with three pages.

(a) Image capture of the Live Paper desktop (b) Result of applying Overlapping & Occluding module to the image. All three pages were located.

In Figure 4-40 and Figure 4-41, five pages were distributed on the tabletop. In the first figure, the data projector has enhanced the pages with simple annotations. In the second, the pages are not enhanced. In both cases, all five pages were found.

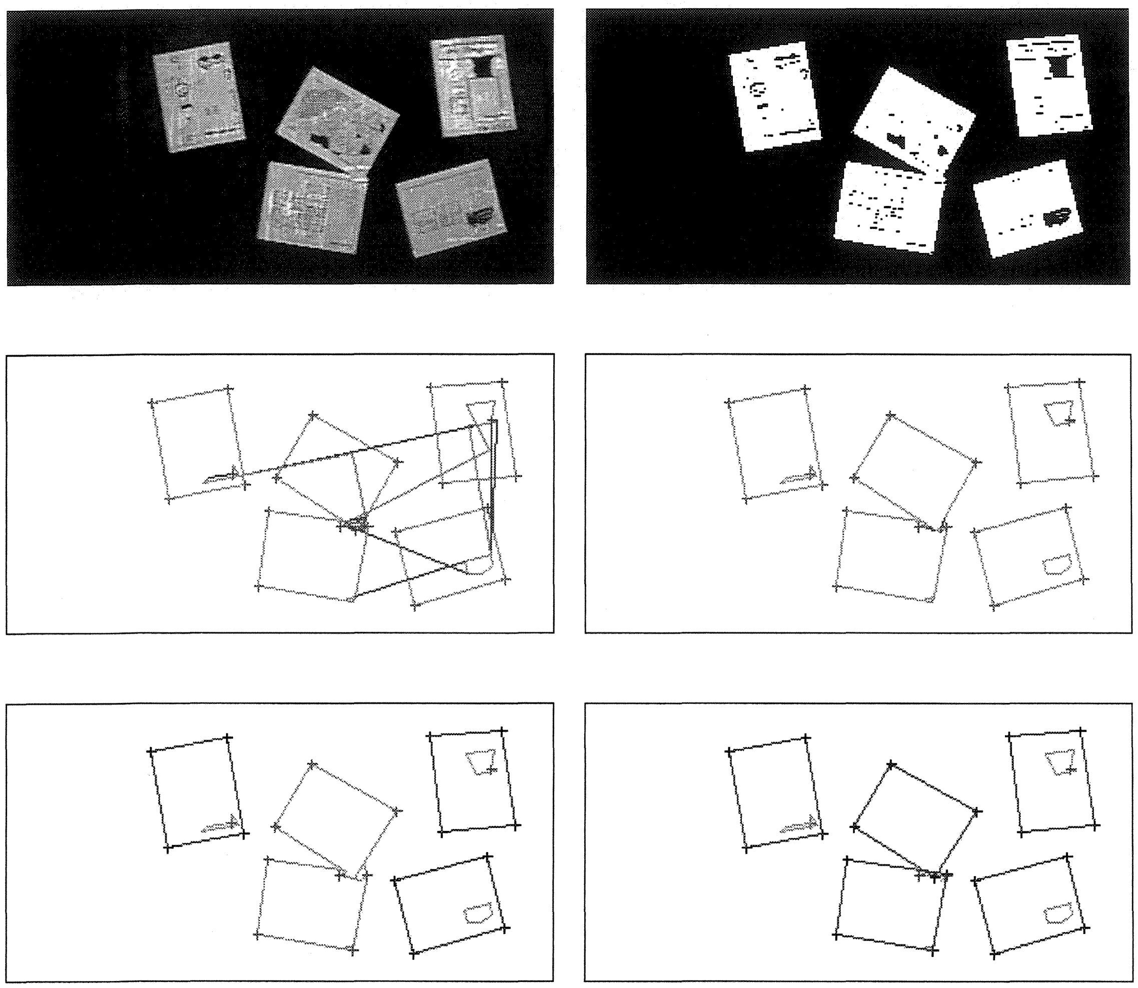


Figure 4-40:Analyzing stages of 5 pages with projections.

From top left, by row: (a) Original image (b) Segmentation via Otsu's method (c) All nodes and links found by the overlapping and occluding method. (d) Pruned links (e) All pages found on the first iteration of the method. (f) All pages found.

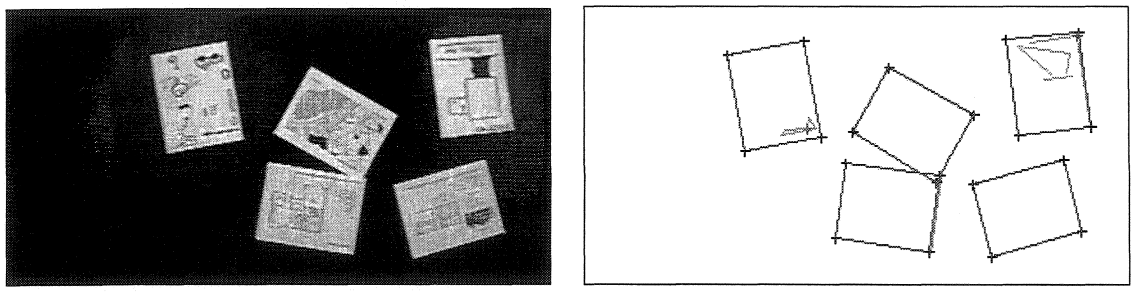


Figure 4-41: Pages without projections

(a) The five pages from Figure 4-40 in the same position. (b) All pages found.

In the final example, in Figure 4-42, eight pages were placed, with varying amounts of overlap, on the tabletop. The methods of this chapter were able to determine the number and location of all eight pages.

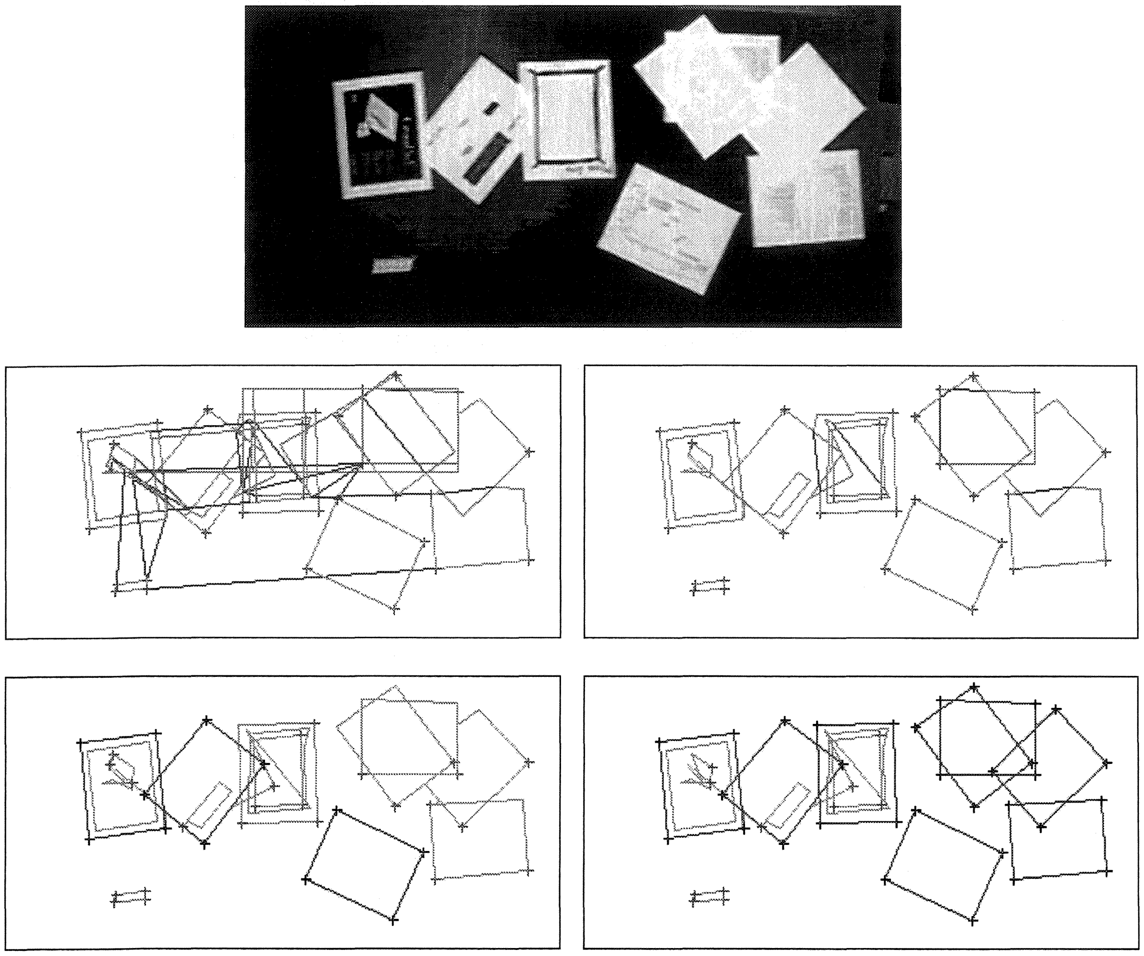


Figure 4-42: A captured image of 8 pages in various overlapping positions.

Image description by row (left to right): (a) Captured image masked for desktop area. (b) All nodes and links found. (c) Pruned links (no iteration). (d) Full pages extracted on first iteration. (e) All pages found and extracted.

Chapter 5

Identifying Paper

This chapter presents techniques for determining which page, in a database of stored page images, matches a newly extracted page. Using the extracted image, a series of algorithms produce suitable metrics for measurement. This chapter presents experimental results for a number of variations on the Hausdorff distance measure, and identifies the best of these. A comparison to a typical grey-level metric shows that the Hausdorff approach is superior for typical pages.

5.1 Motivation

Because it is a real-time system, *Live Paper* must rapidly determine if a page on the desk has already been stored. Part of the procedure presented in Chapter 4 discusses tracking pages from one captured video frame to the next. So long as the tracking is successful, *Live Paper* does not need to do further analysis to determine the identity of the page. However, if the system loses track of a page – for example, the user removes it from the desk for a period of time – then the system must be able to scan a page database to find the matching page. Only then can the system reactivate associated transparencies.

As indicated in Chapter 4, the pages on a *Live Paper* desktop do not have fiducials. The system must use inherent information about the page to identify it. As a result, *Live Paper* treats all pages with identical markings as the same page – it cannot distinguish between an original page and its photocopy. Additionally, all blank pages are considered to be the same page, and, more importantly, the system does not allow the user to enhance a blank page. The reason for this is that as soon as the system loses track of the blank page, it would not know if a new blank page was the same physical page. It is certainly possible for a future version of the *Live Paper* desktop to allow a generic blank page to receive certain enhancements.

Pages on the desktop could contain printed material, or not; hand-written words, or not; drawings, or not; pictures, or not. A suitable generic approach would not make an assumption about the material on the page, but instead use the structure of markings as a unique identifier. If the system could rapidly determine whether or not a mark was writing, and then if it could determine what was written and in what language, then that information (along with its position) could be used reliably to identify the page. However, the poor accuracy and added complexity of performing optical character recognition on a low-resolution video image means that this is not suitable.

This chapter presents a process for quickly identifying pages based on low-resolution images.

5.2 METHODOLOGY

5.2.1 Requirements

The identification algorithm must not require any special fiducials or other identifying marks on the page. The algorithm cannot use optical character recognition as the sole means to identify a page. The algorithm must be fast and not too computationally expensive – suitable to be included as one part of a real-time system. The algorithm must handle pages that contain colour markings.

The algorithm must accurately determine whether a page is new to the system or is already stored in the page database. The page images will be captured at a relatively low resolution (about 5 pixels per inch, or 2 pixels/cm), although higher resolution images will be available when the video camera zooms in (up to 15 pixels per inch, or 6 pixels/cm). The identification algorithm must be able to work with page images of various resolutions.

5.2.2 Assumptions

The assumptions for the identification algorithm are the same as those presented in section 4.2.2. One additional assumption is that the projections will not interfere with the identification algorithm. Pages will be identified before the system places annotations on them. Once a page has been identified, the tracking algorithm presented in section 4.4.3

will track that page between successive video frames, and thus the identification algorithm will not subsequently operate on that page.

5.2.3 Process

Identification takes place in three steps (see Figure 5-1). After the page has been located (see Chapter 4), the page image is extracted. Then the resulting image is segmented into writing and non-writing areas. Finally, the *Live Paper* system uses a Hausdorff distance algorithm to compare the segmented image with a database of stored pages, where each page has an associated image. The smallest distance indicates the best match. If this distance is too large, then the *Live Paper* system adds the page to the database as a brand new page.

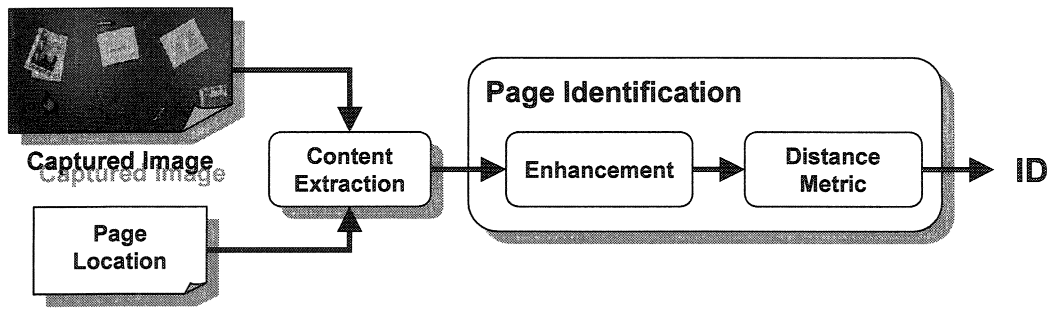


Figure 5-1: Procedure for Page Identification

Content extraction is accomplished by using texture mapping based on the four page corners. The enhancement stage modifies the content so that it is suitable for look-up in the database. The *Live Paper* uses an operation proposed by Wellner to convert the grey level page image into a binary image. The distance metric is a variation on the Hausdorff distance measure. However, the identification block could be replaced with other techniques, such as contrast stretching and blurring, or a grey level Euclidean measure. Section 5.4 explores other options, and compares them with the Hausdorff measure.

5.3 BACKGROUND ON IDENTIFYING PAPER

5.3.1 Page Identification Techniques

Several groups have tackled the problem of page identification within the context of a video augmented environment like the *Live Paper* tabletop. However, all of these solutions use markings on the objects for identification, and so the solutions are not suitable for ordinary pieces of paper.

Those involved with the Origami project proposed using cross-hair-like fiducials in each of the page corners, and a numerical identification code printed along a page in the margin [Robinson 1997]. The identifying code is printed in a font typeface that can be easily read using optical character recognition on the video image.

Another solution is the use of Cyber Codes [Rekimoto 1998] [Rekimoto 2000]. Originally termed matrix codes, these are small fiducials that contain black and white squares in a grid-like pattern. By rearranging the positions of the squares, unique codes can be created. To locate the page, a system using a Cyber Code must first locate the code, and determine what page is present. Each individual square of the code must be large enough so that the system can determine the code from a video image of the page.

The size and shape of a Cyber Code is such that normal markings on the page are unlikely to be mistaken for a code.

Related to Cyber Codes is the use of coloured dots in the Illuminating Light project [Underkoffler 1998]. Although these dots were attached to physical blocks representing holographic equipment such as lasers, the same process could be applied to paper. As with Cyber Codes, there is still a possibility that ordinary markings on a page could mistakenly be interpreted as fiducials.

Researchers at Xerox developed a method called DataGlyphs [Dymetman 1998] [PARC 2002] that encode binary data into fine printed strokes (glyphs). These glyphs can be easily encoded as part of grey-scale pictures, or printed as a light grey background. The data rate at 600 dpi is approximately 1KB per square inch. A scanner must be used to retrieve the data, which is encoded with synchronization, randomization, and error correction data. In the original proposal for *Intelligent Paper*, the plan was to use the DataGlyphs to store page identification and location information. The user would use a hand-held scanner to select links at predetermined locations on the page to access dynamic data on a computer.

These solutions are not incompatible with *Live Paper*; a video-augmented tabletop system could support pages with and pages without identifying markings. However, because ordinary pieces of paper do not already have fiducials embedded, a new approach is necessary.

5.3.2 Finding Documents in Databases

Researchers have examined a number of techniques for matching a document to others in a database. One technique is to extract the locations and shapes of blocks of text and pictures on a scanned page. A similar method [Casey 1992] used the location and length of horizontal and vertical lines on a scanned form to automatically classify the form for further processing. Another method [Doermann 1997] detected duplicates in a document database by extracting signatures based on the shape of representative text. These techniques are not suitable for *Live Paper*, as they make assumptions about the type of documents that will be present – for example, that all pages will have printed text. *Live Paper* requires an algorithm that allows any type of writing or printing on a page.

5.3.3 Binarization

Analysis of segmentation techniques, similar to those discussed in chapter 4, showed that shadows and uneven lighting on a page would often cause a histogram with significant overlap of object and non-object regions. Thus the author investigated several adaptive thresholding algorithms.

The *DigitalDesk* uses the fast thresholding algorithm called Moving Averages, developed by Pierre Wellner [Wellner 1993b]. The algorithm was specifically designed for scanned document images of black text on a white page, and works well even in the presence of non-uniform lighting [Parker 1997]. However, its performance as a general segmentation algorithm is mixed (see Figure 5-2).

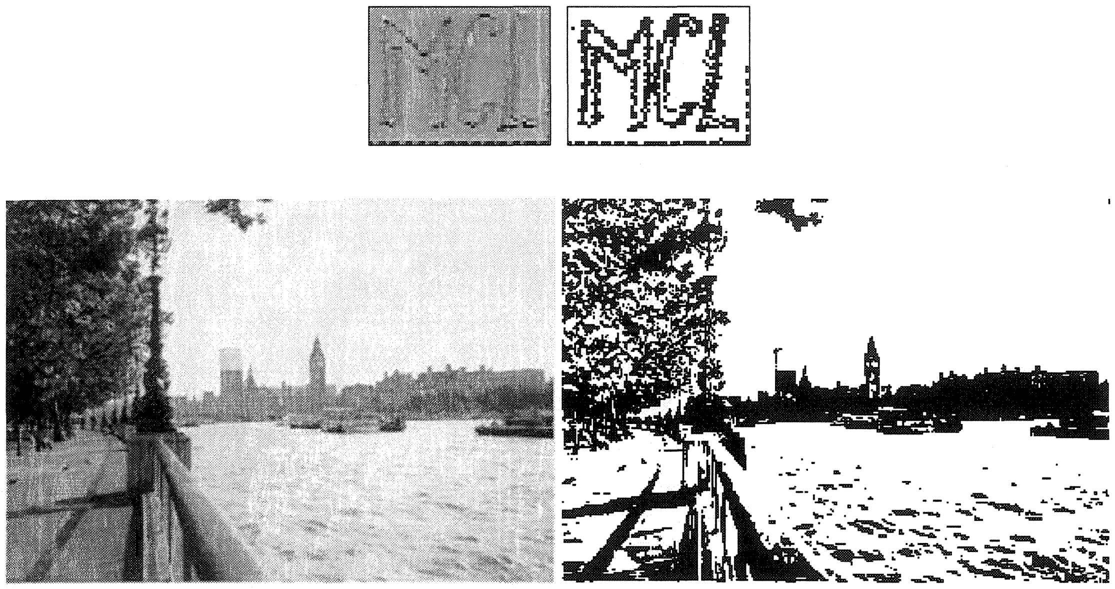


Figure 5-2: Application of Moving Averages to images of a page and a natural scene.

The algorithm works by maintaining an average of the last *s* pixels encountered while traversing each row of an image. If a newly encountered pixel is significantly darker than this moving average, it is classified as foreground. Otherwise, it is classed as background.

Because the algorithm makes only one pass through the image, it is very fast. To exploit the natural similarity that occurs at the start (and end) of two consecutive rows, the algorithm scans the image in alternating directions. To eliminate streaks that naturally occur, the algorithm uses the average of the current running average and the running average at the same horizontal position on the previous row.

Live Paper uses the moving averages algorithm without modification. A more detailed definition of the algorithm follows.

Definition

Given an image P, let p_n represent the value of the current pixel being analyzed. Let T be the final thresholded image, and t_n be the pixel in T which corresponds to p_n. The value of t_n is given as

$$t_n = \begin{cases} 1 & \text{if} \quad p_n < \frac{h_n}{s}\left(\frac{100-t}{100}\right) \\ 0 & \text{otherwise} \end{cases} . \tag{5-1}$$

The value t is the threshold value – a pixel is 'on' (value of 1) if it is t percent darker than the background.

The value h_n is the average of the current running sum of pixel values and the running sum at the same horizontal position on the previous row:

$$h_n = \frac{g_n + g_{n-width}}{2} . \tag{5-2}$$

Thus in Equation 5-1, h_n/s is the running average at point p_n.

In Equation 5-2, the value *width* is the width (row length) of the image P. The value g_n is an approximation to a running sum of s pixels in length. It is defined as

$$\begin{aligned} g_n &= g_{n-1} - \frac{g_{n-1}}{s} + p_n \\ &= \sum_{i=0}^{n} \left(1 - \tfrac{1}{s}\right)^i p_{n-i} \end{aligned} . \tag{5-3}$$

The value for g_0 is arbitrary, but Wellner suggests using $127s$, based on an 8-bit grey-scale image. Another possible value is sp_0, although the value p_0 might not be representative of pixel values in P.

The scanning of the image takes place in a boustrophedon manner – each row is scanned in an alternating direction (see Figure 5-3). By convention, the algorithm starts in the upper left corner of the image, and progresses down the image row by row.

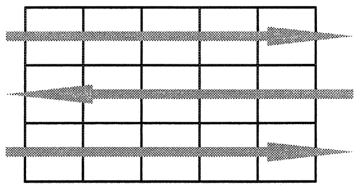


Figure 5-3: Boustrophedon Scanning

In a more conventional filtering view, the moving averages algorithm can be viewed as a series of four operations, two of which are filters (see Figure 5-4). The input is the 1-dimensional discrete signal produced by scanning across the columns of the image, row by row. The first stage reverses the order of every second row of the image. The second stage approximates the running average using an infinite impulse response (IIR) filter with an impulse response of $a^n u[n]$, where u[n] is the discrete-time unit step function. The third stage averages the current value with the one on the previous row, equivalent to a finite impulse response (FIR) filter with an impulse response of $\delta[n] + \delta[n-w]$, where $\delta[n]$ is the discrete-time unit impulse function and w is the width of the image. However, because the previous row of the image is generally similar to the current row, the effect is to approximate a non-causal IIR filter with impulse response $a^{n}u[n] + a^{-n}u[-n]$. The last stage thresholds the data to create the binary output.

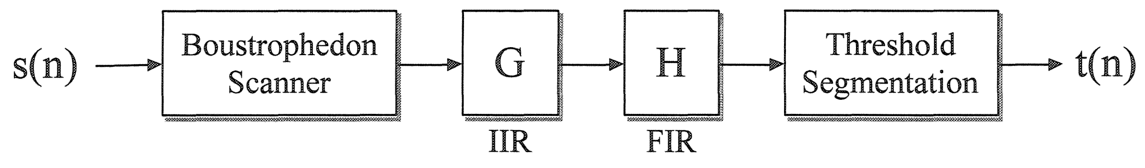


Figure 5-4: The series of operations for the moving averages algorithm.

5.3.4 Distance Measures

5.3.4.1 The Hausdorff Distance

Once the page image has been converted to two-levels, *Live Paper* compares the image to others that have been stored. *Live Paper* uses the Hausdorff distance to measure the similarity of pairs of pages. The Hausdorff distance is small for two pages that are visually similar, and large for two pages that are visually distinct. In the case of a perfect match, the distance between the pages is 0.

The Hausdorff distance has been successfully used [Huttenlocher 1993][Dubuisson 1994][Hull 1997] in the areas of image and document similarity detection. The Hausdorff distance is a metric that measures the similarity of points in two finite sets, but does not find a one-to-one correspondence between points in each set. Usually, in the case of similarity detection, two-level edge images are used to generate the point sets, but this is not always so [Hull 1997].

The Hausdorff distance is very tolerant of small differences in position [Huttenlocher 1993]. A slight distortion in an image will not cause significant changes in the Hausdorff distance. The modified Hausdorff distance is also tolerant of extraneous (error) pixels [Dubuisson 1994]. Another useful feature is the potential to detect additions to an image.

Definition

Assume two sets A and B, containing N_A and N_B number of points respectively. The Hausdorff Distance is defined as

$$H(A,B) = \max(h(A,B), h(B,A)). \tag{5-4}$$

The function $h(A,B)$ is the directed Hausdorff Distance from set A to set B, and is the maximum of the distances from each point in A to the closest point in B. It is defined as

$$h(A,B) = \max_{a \in A} d(a,B). \tag{5-5}$$

The function $d(a,B)$ is the minimum distance between point a and all points in the set B, and is defined as

$$d(a,B) = \min_{b \in B} \|a - b\|. \tag{5-6}$$

The term $||a - b||$ is the distance between two points, or the norm of the vector from point a to point b. A standard definition of $||a - b||$ is the Euclidean, or L_2-, Norm. The L_1-Norm and the $L_{infinity}$-Norm could also be used. Their definitions are

$$L_1(a,b) = \|a-b\|_1 = \sum_{r=1}^{n} |a_r - b_r|, \tag{5-7}$$

$$L_2(a,b) = \|a-b\|_2 = \sqrt{\sum_{r=1}^{n} |a_r - b_r|^2} \text{, and} \tag{5-8}$$

$$L_\infty(a,b) = \|a-b\|_\infty = \max_r |a_r - b_r|, \tag{5-9}$$

where a_r is the component value of point a along the r-axis, and n is the number of axes. Another possible definition for $d(a,B)$ is to take the minimum squared L_2-Norm:

$$d(a,B) = \min_{b \in B} \left[\|a-b\|_2\right]^2 \tag{5-10}$$

$$[L_2(a,b)]^2 = \left[\|a-b\|_2\right]^2 = \sum_{r=1}^{n} |a_r - b_r|^2 \tag{5-11}$$

The advantage of the L2-Norm squared is enhanced discriminability when used with the Modified Hausdorff Distance below (Equation 5-12). In that case, the overall measure is very similar to the Mean Squared Error.

For an example calculation, see Figure 5-5, which shows two raster images. The five black pixels in image *A* become points in set *A*, and the eight black pixels in image *B* become the points in set *B*. Four of the points in image *A* are in image *B*. The fifth point in *A* is diagonally adjacent to a point in *B*, with a difference of 1 unit horizontally and 1 unit vertically. Thus the measure *h(A,B)* is 2 for the L_1-Norm, 1.414 for the L_2-Norm, and 1 for the $L_{infinity}$-Norm.

The directed Hausdorff distance from B to A also varies, so that $h(B,A)$ is 5 for the L_1-Norm, 4.123 for the L_2-Norm, and 4 for the $L_{infinity}$-Norm. The Hausdorff distance is the maximum of the two directed distances, and so $H(A,B) = h(B,A)$ for all three measures.

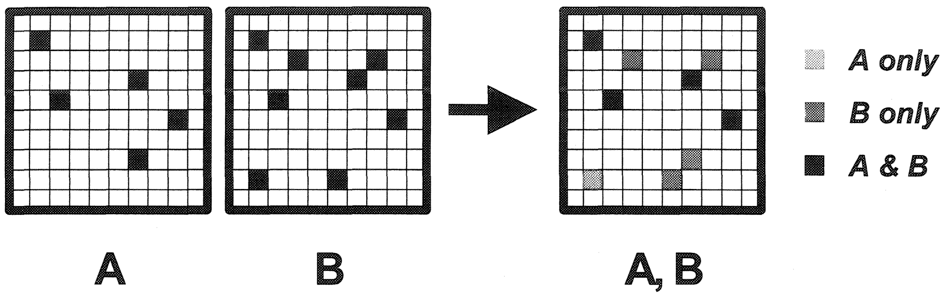


Figure 5-5: Hausdorff Calculation Example

The Modified Hausdorff Distance

The standard directed Hausdorff distance is very susceptible to noise in the image. Dubuisson and Jain proposed an alternate method to address this problem while maintaining a high level of discrimination among images [Dubuisson 1994]. The Modified Hausdorff Distance (MHD) simply averages the distances from all points. Thus $h(A,B)$ is replaced in Equation 5-4 by:

$$h_M(A,B) = \frac{1}{N_A} \sum_{a \in A} d(a,B) \tag{5-12}$$

where N_A is the number of points in set A. The function $h(B,A)$ is similarly replaced in Equation 5-4.

Other Variations

The overall Hausdorff distance definition (Equation 5-4) can also be modified to be the average of the two directed distances, as in

$$H(A,B)=\frac{h(A,B)+h(B,A)}{2}. \tag{5-13}$$

5.3.4.2 Grey-level Euclidean Distances

The grey-level Euclidean distance measures the similarity of two images on a pixel-by-pixel basis. Here, the research uses the Euclidean distance as a representative grey-level classifier. Other classifiers, such as the Minimum Intra-Class Distance, would not offer significant improvement because the number of training samples is sparse compared to the number of dimensions [Duda 1973]. The number of training samples per page would be the number of captured images of that page, while the number of dimensions would be the total number of pixels on a page (55 x 43, for a total of 2365 dimensions).

The intensity value of a pixel in one image is directly compared with the corresponding pixel in the other image. The total of the absolute intensity differences indicates the overall similarity of the two images. If the sum is zero, then the two images match perfectly.

The formulation of the grey-level Euclidean distance is similar to that given in Equation 5-8. Both images must have the same dimensions – N rows and M columns. The intensity representation for both images must also be the same; for example, an 8-bit intensity resolution will give 256 distinct levels of grey. The formulation for the Euclidean distance assumes that the intensity changes linearly.

$$\|a-b\|_2 = \left(\sum_{i=1}^{N} \sum_{j=1}^{M} \left| a_{ij} - b_{ij} \right|^2 \right)^{\frac{1}{2}} \tag{5-14}$$

As Equation 5-14 indicates, if images *a* and *b* are identical, then the difference at each pixel location is zero, and the grey-level Euclidean distance is also zero.

5.3.4.3 Hausdorff Fraction and Eigenspaces

A variation on Hausdorff uses subspace recognition methods to approximate a measure called the Hausdorff fraction [Huttenlocher 1999]. Instead of calculating a distance, it measures the fraction of points in one set that is within a distance *d* of points in another set. If two sets match well, then their Hausdorff fraction approaches 1; totally mismatched sets will have a fraction close to 0 for all but the largest values of *d*.

Model images are treated as column vectors, and are then combined to form a matrix. The method uses eigen-decomposition on this matrix to find the *k* eigenvectors with the largest eigenvalues. When comparing a new image with the stored model images, these *k* eigenvectors are used to approximate the Hausdorff fraction. The method provides for compact storage and fast indexing, and is robust to partial occlusion.

This thesis does not investigate the use of the Hausdorff fraction and eigenspaces, but the method would be worth examining for future use in page recognition. One key issue to investigate would be whether a representative training set exists. Also, the variation of markings on pages might be such that the value of k would have to be very large to provide adequate discriminability. There might also be a problem accommodating new pages into the eigenspace of the model images.

5.4 EXPERIMENTS

The author performed a number of experiments to determine the best Hausdorff distance measure for *Live Paper*, and to determine the accuracy. A commercial pan/tilt/zoom camera captured images of letter-size (8.5x11 inches) pages on a desktop at a distance of 180 to 190 cm. The size of the page image was approximately 55x43 pixels in the original 320x240 image, and this extracted image (after application of the Moving Averages algorithm) became the basis for the Hausdorff point set. Section 5.6 provides details on the final system implementation. The Voronoi surface for each extracted page image was generated using a 13x13 distance template at each point. Figure 5-6 shows the corresponding point sets and Voronoi diagrams for two sample pages.

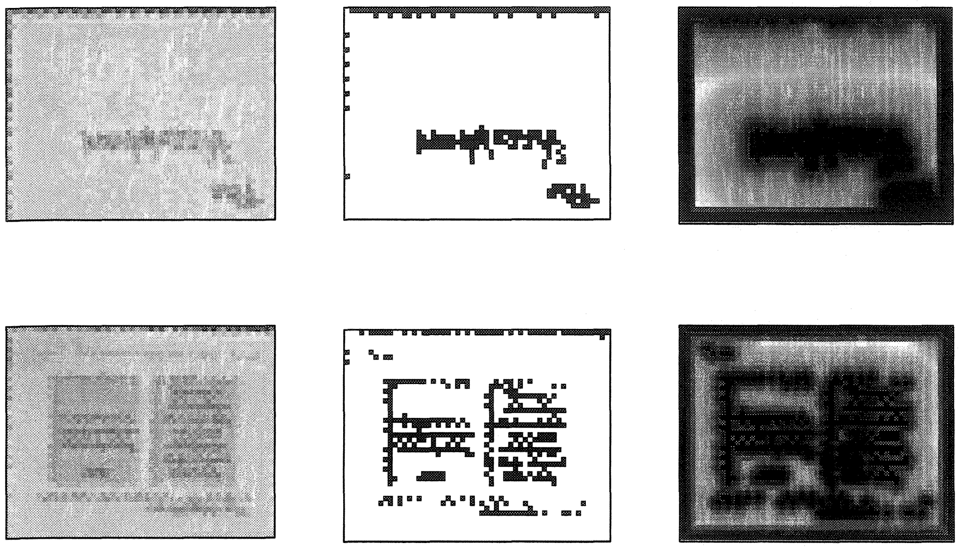


Figure 5-6: Sample page images, along with their corresponding binary versions and Voronoi diagrams.

In the experiments, 171 different pages were compared. The pages encompassed three different styles, including hand-drawn (Figure 5-7), presentation (Figure 5-8), and journal articles (Figure 5-9). The hand drawn images are sketches or writing, and are generally distinct. The presentations are two sets of computer-generated slides. Slides in each set are visually similar. The journal images are published papers that contain printed text and some pictures. These provided a good breadth of potential documents that users of *Live Paper* would present to the system. The representative pages include many similar samples.

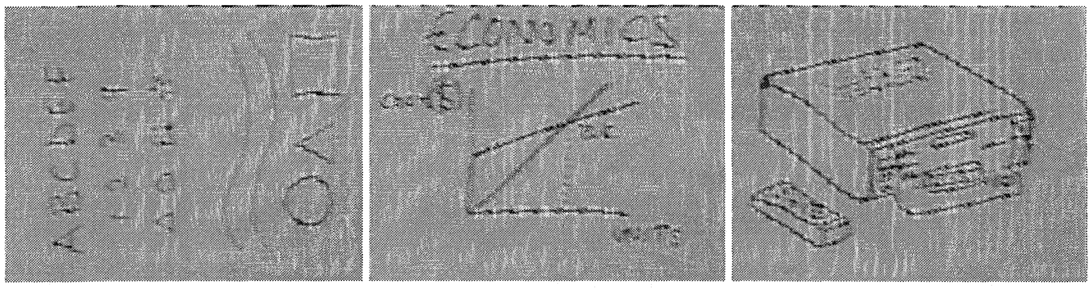


Figure 5-7: Samples of pages with handwritten content, as captured by the system during testing.

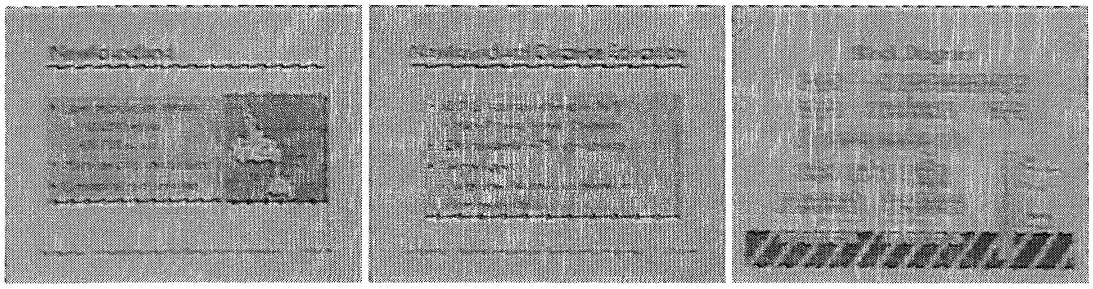


Figure 5-8: Samples of pages containing presentation content as captured by the system.

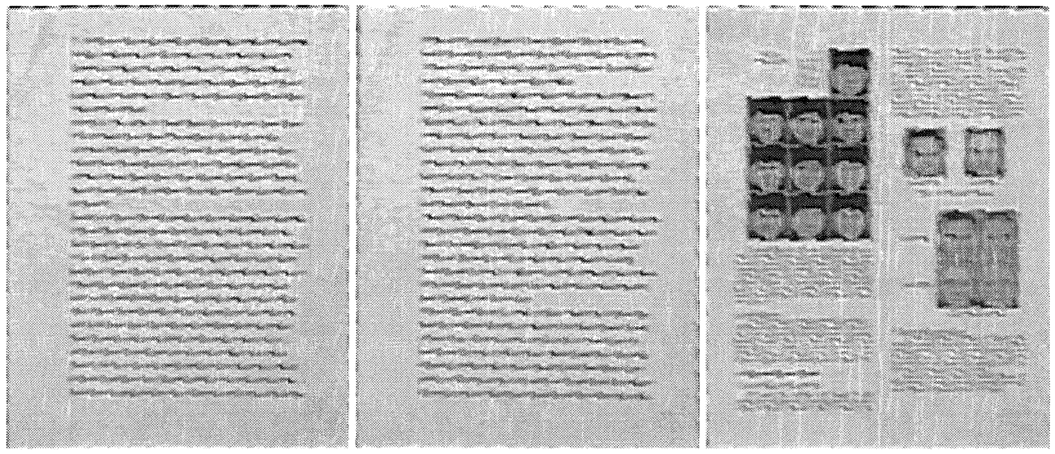


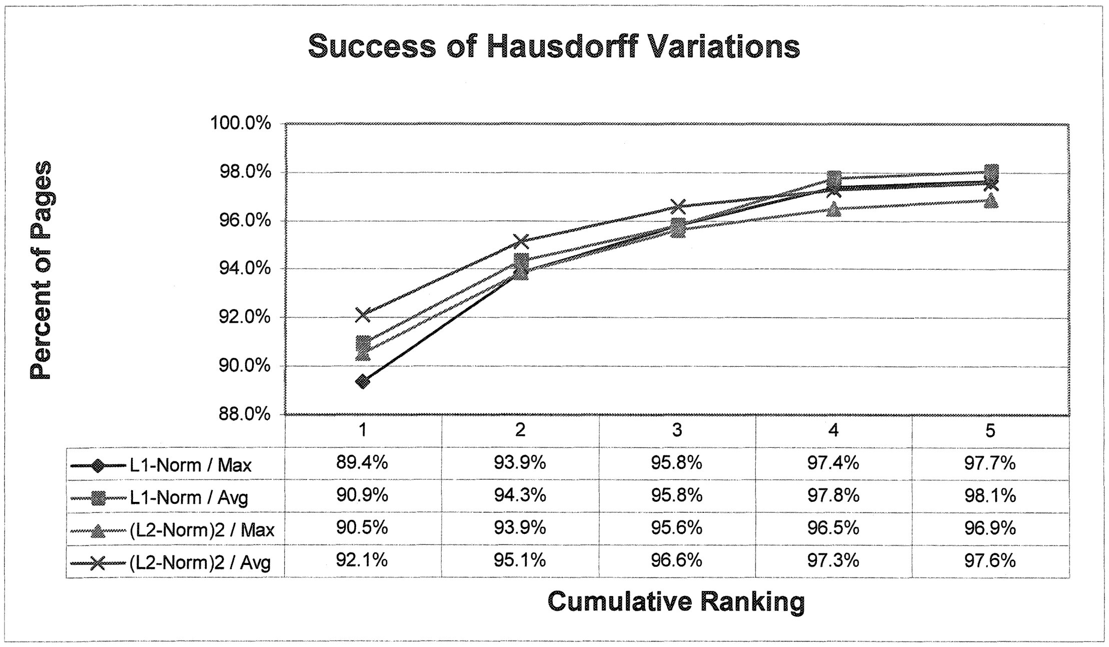
Figure 5-9: Sample journal pages captured by the system.

5.4.1 Best Hausdorff Distance Form

Table 5-1 compares four variations of the Hausdorff distance. The results give the rank of model images to their corresponding test image. A rank of 1 indicates a correct match, while a rank of 2 indicates that the one other model image was closer to the test image. To generate the set of test pages, an image of each page is stored as a model image, and then each page was captured at six different positions and orientations. Each set of test images was compared independently against the model set, and the results aggregated. The total page count is 1026 (6 sets of 171 unique pages). The graph in Figure 5-10 compares the four variations.

Measure Used	*Moving Average*		*Ranking Results (Percentage)*					
	s	*t*	*1*	*2*	*3*	*4*	*5*	*6+*
L_1-Norm, MAX (Equations 5-5, 5-7)	*9*	*3*	89.38	4.48	1.95	1.56	0.29	2.34
L_1-Norm, AVG (Equations 5-13, 5-7)	*9*	*3*	90.94	3.41	1.46	1.95	0.29	1.95
$(L_2\text{-Norm})^2$, MAX (Equations 5-4, 5-11)	*15*	*3*	90.55	3.31	1.75	0.88	0.39	3.12
$(L_2\text{-Norm})^2$, AVG (Equations 5-13, 5-11)	*16*	*3*	92.11	3.02	1.46	0.68	0.29	2.44

Table 5-1: Results of various Hausdorff Distance calculations (6 sets of 171 pages each)



	1	2	3	4	5
◆ L1-Norm / Max	89.4%	93.9%	95.8%	97.4%	97.7%
■ L1-Norm / Avg	90.9%	94.3%	95.8%	97.8%	98.1%
▲ (L2-Norm)2 / Max	90.5%	93.9%	95.6%	96.5%	96.9%
✕ (L2-Norm)2 / Avg	92.1%	95.1%	96.6%	97.3%	97.6%

Figure 5-10: A comparison of the cumulative success of each Hausdorff Distance variation.

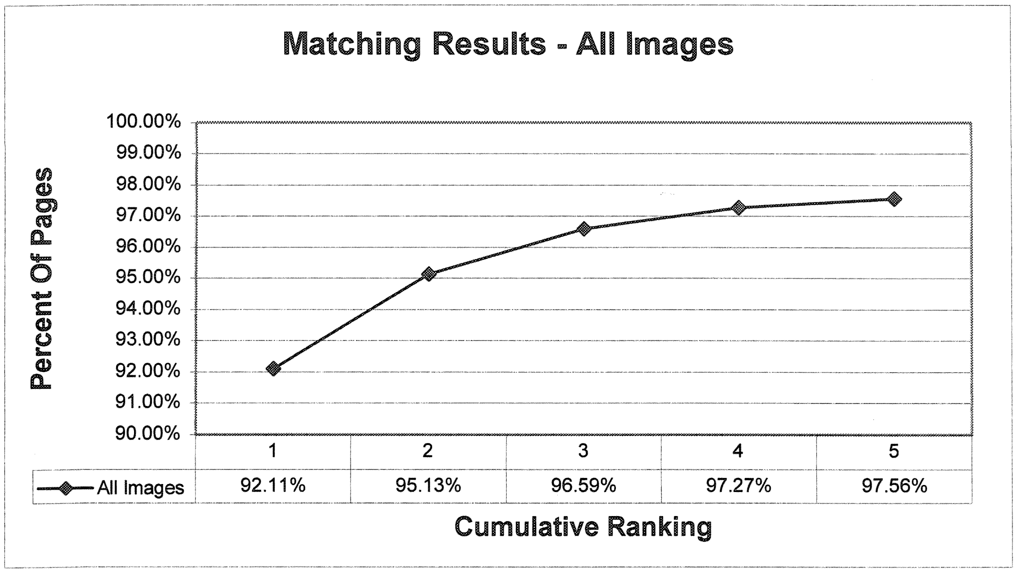
In contrast to previous results [Dubuisson 1994], this research showed a difference between averaging the directed distances and using the maximum directed distance for the Hausdorff distance (see Equations 5-13 and 5-4). This research also showed that using the (L_2-Norm)^2 to measure the distance between points produces more accurate results - 92.1% of the page images were correctly matched, and 96.6% were matched within the top 3.

Table 5-2 examines in detail the case of using the (L_2-Norm)^2 measure and averaging the directed distances. The results for the three styles of pages are broken out. The third column indicates whether the ranking results are based on the best Moving Average parameters for the given page style, or the best for the entire set. The optimal values of the Moving Average parameters vary for different types of pages and illumination conditions. For all images, the best (fixed) parameters were s = 16 and t = 3. Note that in the all image case, images of one set could be matched against images of a different set.

Image Set	*Number*	*Moving Avg. Parameters*	*Ranking Results (Percentage)*					
			1	*2*	*3*	*4*	*5*	*6+*
All Images	*1026 (171 x 6)*	*Optimal & Fixed s=16 t=3*	92.11	3.02	1.46	0.68	0.29	2.44
Hand Drawn	*300 (50 x 6)*	*Optimal s=5 t=6*	100.00	0.00	0.00	0.00	0.00	0.00
		Fixed	97.00	1.00	1.33	0.33	0.33	0.00
Presentations	*450 (75 x 6)*	*Optimal s=8 t =5*	92.22	4.44	1.78	1.11	0.00	0.44
		Fixed	90.00	4.89	2.67	1.11	0.44	0.89
Journals	*276 (46 x 6)*	*Optimal s=9 t =3*	90.94	1.45	1.81	0.00	1.45	4.35
		Fixed	90.22	2.90	1.09	0.36	0.00	5.43

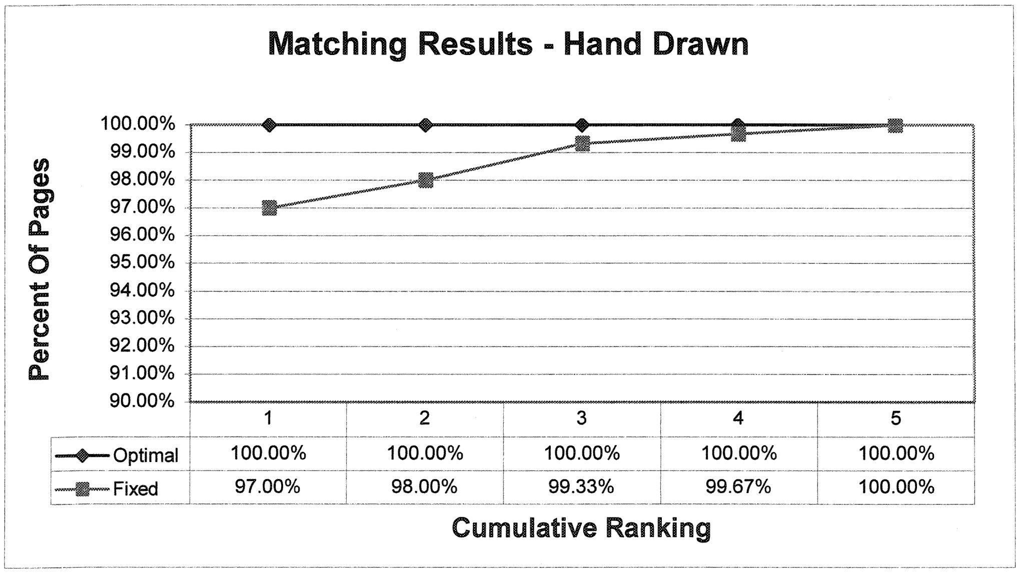
Table 5-2: Detailed results for (L_2-Norm)^2 average Hausdorff distance

The graphs in Figure 5-11, Figure 5-12, Figure 5-13, and Figure 5-14 show the cumulative ranking results from Table 5-2.



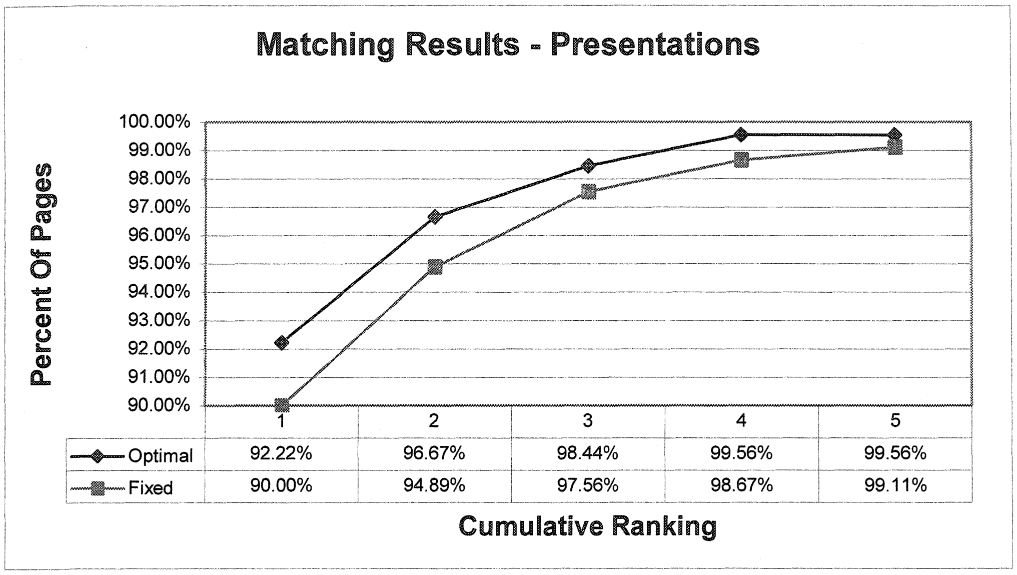
	1	2	3	4	5
All Images	92.11%	95.13%	96.59%	97.27%	97.56%

Figure 5-11: Cumulative rankings for all pages using the (L_2-Norm)^2 average Hausdorff distance.



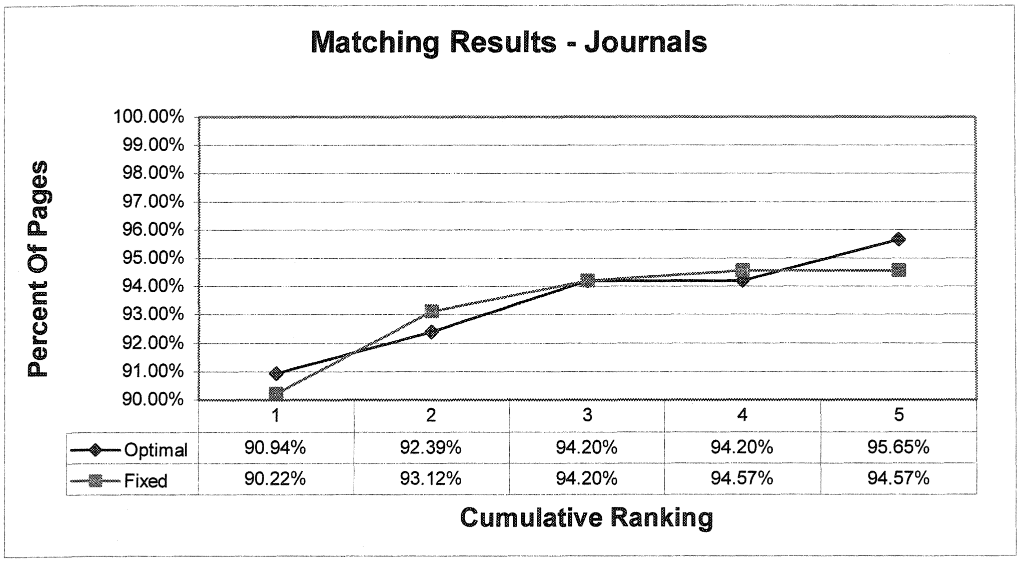
	1	2	3	4	5
Optimal	100.00%	100.00%	100.00%	100.00%	100.00%
Fixed	97.00%	98.00%	99.33%	99.67%	100.00%

Figure 5-12: Cumulative rankings for hand-drawn pages using the (L_2-Norm)^2 average Hausdorff distance.



	1	2	3	4	5
Optimal	92.22%	96.67%	98.44%	99.56%	99.56%
Fixed	90.00%	94.89%	97.56%	98.67%	99.11%

Figure 5-13: Cumulative rankings for presentation pages using the (L_2-Norm)^2 average Hausdorff distance.



	1	2	3	4	5
Optimal	90.94%	92.39%	94.20%	94.20%	95.65%
Fixed	90.22%	93.12%	94.20%	94.57%	94.57%

Figure 5-14: Cumulative rankings for journal pages using the (L_2-Norm)^2 average Hausdorff distance.

5.4.2 Effect of Page Image Size

Because the Live Paper uses a computer controlled pan/tilt/zoom camera, the system can direct the camera to zoom in on a page. At zoom level of 2, a letter-sized page covers approximately 110x85 pixels. At a zoom level of 3, a letter-sized page has a size of about 165x128 pixels.

These sizes were used in extracting 76 journal pages to examine the accuracy of the measurement. For zoom level 2, 96.9% of the pages were matched correctly, and 98.2% were in matched within the top three. At a zoom level of 3, 99.6% of the pages were matched correctly, and all pages were matched within the top three.

These results suggest two different implementation schemes to increase the accuracy of matching, with the trade-off of increased computation. One implementation would use an increased resolution for all matching. The system would need to capture desktop images with a video camera and frame grabber card that support a higher resolution. The second implementation would use lower resolution images for most page matching, but direct the camera to zoom in to obtain higher resolution images when necessary.

Neither solution is optimal for the hardware used in *Live Paper* (see Section 7.2.2). Because there is only one commercial quality video camera, the system cannot retrieve higher resolution images without zooming in. However, the system would not be able to monitor the tabletop while it was retrieving the detailed image, and this process would take on the order of 10 seconds.

5.4.3 Detecting False Matches

The ability to detect false matches is important. There are two measures that can be used to determine the confidence of the match. When a candidate page is the same as one stored, the Hausdorff distance should be very low. Figure 5-15 shows the distribution of distances for correct matches and the closest mismatch for the set used in Table 5-2. Although the average distance for the matches is 0.728, compared to 1.896 for the mismatches, the spread of the mismatch is large. No one distance can threshold matches from mismatches.

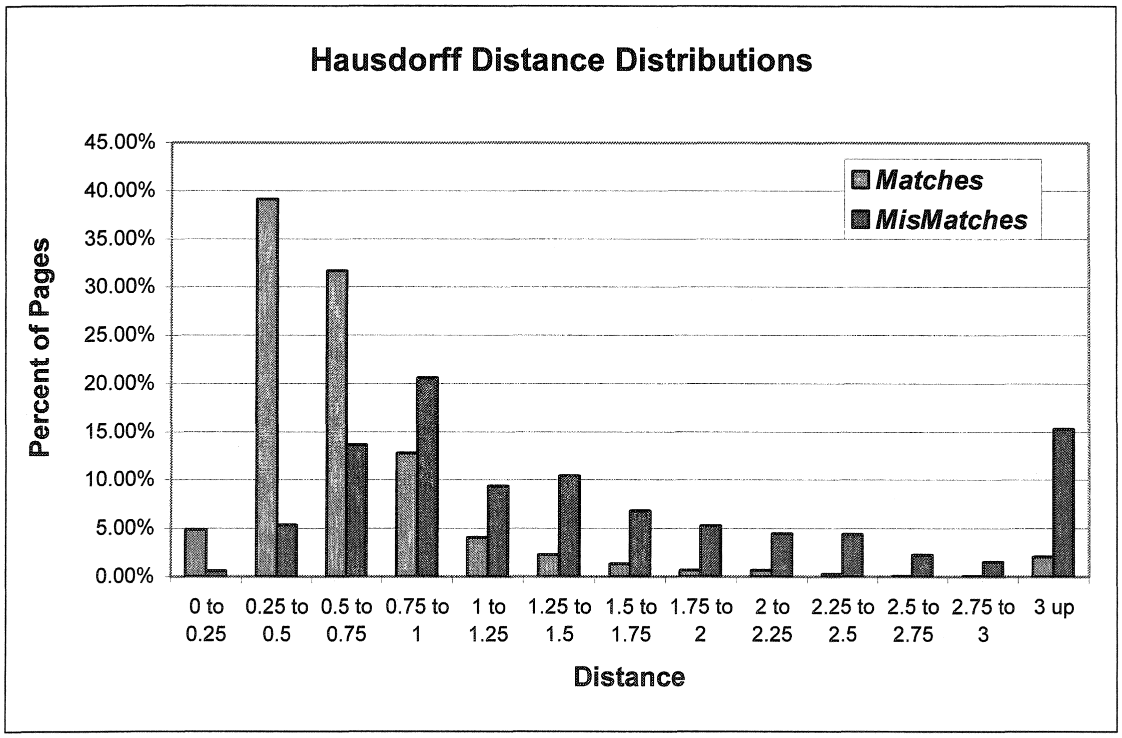


Figure 5-15:Distribution of distances for 171x6 pages to correct match and closest mismatch.

Another confidence measure is the ratio of the second best match to best match. When the best match is correct, the ratio should be large. When the best match is wrong, the ratio should be small because the two distances are more likely to be similar. Figure 5-16 shows the distribution of ratios for the set in Table 5-2. Using this measure, it is possible to separate most of the mismatches.

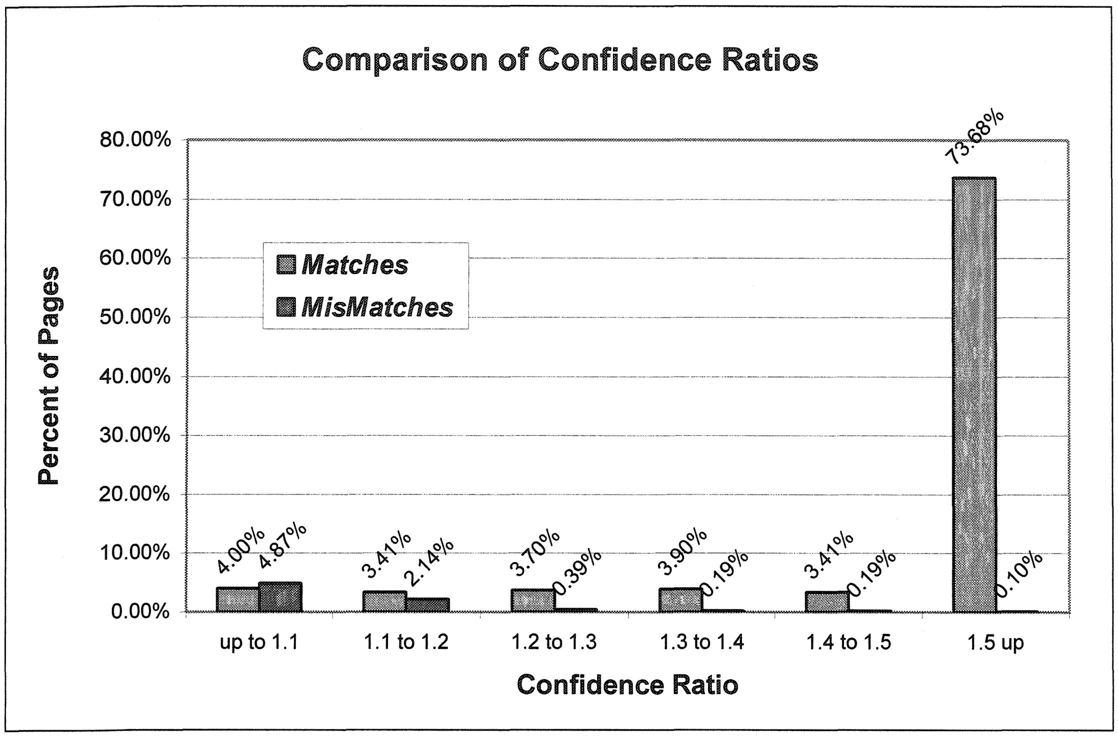
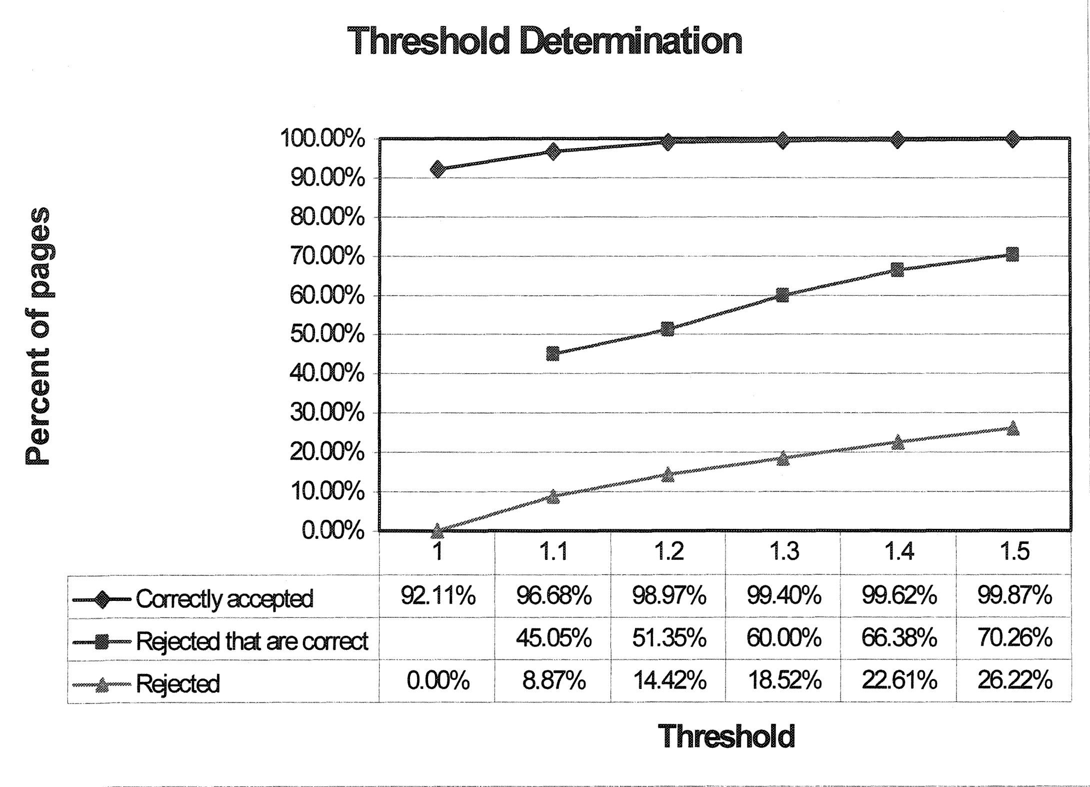


Figure 5-16: Ratios of best to second best matches (171x6).

Figure 5-17 shows the effects of choosing various confidence ratio thresholds. Choosing a value of threshold of 1.2 means that 14.42% of the pages would be rejected, but that 98.97% of the accepted pages are correctly identified. This information could determine when to use a larger image size for the Hausdorff distance calculation.



	1	1.1	1.2	1.3	1.4	1.5
◆ Correctly accepted	92.11%	96.68%	98.97%	99.40%	99.62%	99.87%
■ Rejected that are correct		45.05%	51.35%	60.00%	66.38%	70.26%
▲ Rejected	0.00%	8.87%	14.42%	18.52%	22.61%	26.22%

Figure 5-17: Effects of various threshold values on the success of correctly identifying a page.

5.4.4 Timing

Table 5-3 shows the timing results of the $(L_2\text{-Norm})^2$ implementation on a PC with an Intel 166MHz Pentium, 128 MB of RAM, and the Microsoft Windows 95 operating system. The last line in the table is a calculated value given for comparison purposes. The results show that this implementation of the moving averages algorithm and the $(L_2\text{-Norm})^2$ modified Hausdorff distance measure has a speed suitable for a real-time system. Each page-to-page comparison is the equivalent of two full Hausdorff distance calculations, due to the possibility of 180° rotation.

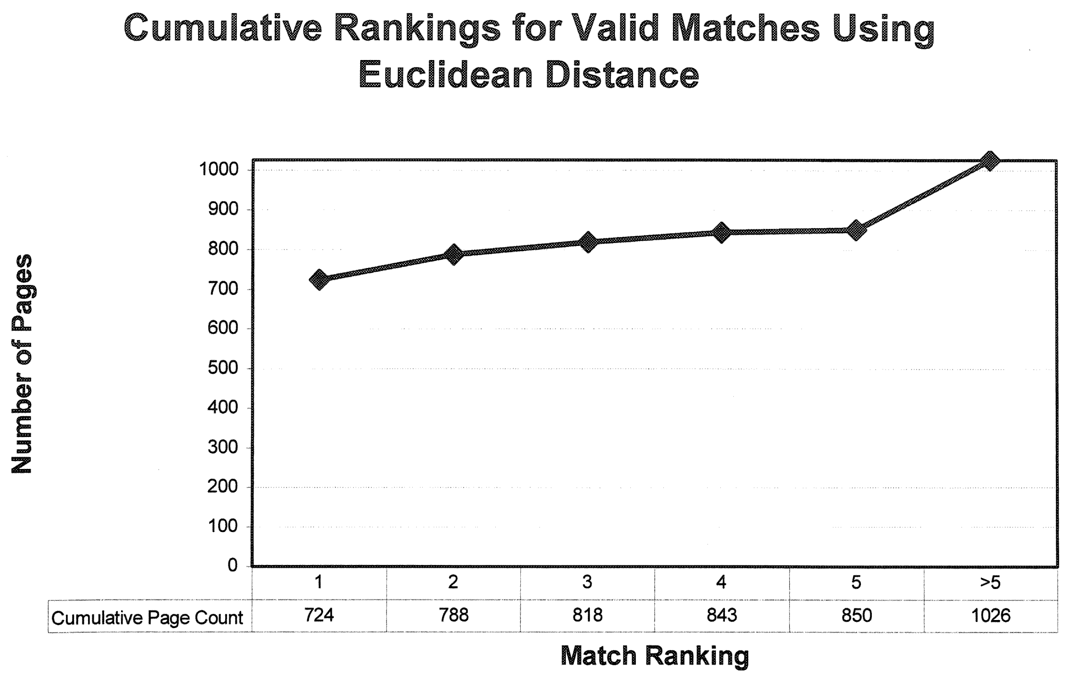
Execution time for Moving Average algorithm	6.46 ms
Time to create point list & Voronoi	18.6 ms
Time for Search of Page-to-Stored	105 ms
Time for a Page-to-Page Compare	0.373 ms
Number of pages searched in one second	2613 pages

Table 5-3: Timing Results of the $(L_2\text{-Norm})^2$ Implementation

5.4.5 Grey-level Euclidean Measures

The Hausdorff distance is a measure that calculates similarity between sets of two-level images. Given that the original images of the pages are grey-scale, a multi-level measure was selected for comparison. Initial results using the Euclidean distance measure were extremely poor, and the extracted page images were enhanced by histogram equalizing and low-pass filtering (with a 5x5 Gaussian filter). The histogram equalization corrected some variations in lighting, and the low-pass filter reduced the effects of aliasing, both in the edges of strong marks and at the corners of the pages.

For the set of pages captured under the conditions outlined at the beginning of section 5.5, the tests showed a successful match rate of 70.6% for the Euclidean distance measure. By comparison, the best match rate for the Hausdorff distance was 92.1% for the $(L_2\text{-Norm})^2$. Figure 5-18 shows the ranking results for the Euclidean distance. A rank of 2 means that the correct match for the candidate page was the second best match. The cumulative match rate increases slowly as the rank is increased, and reaches a cumulative rate of 82.8% for rank 5. This indicates that even if the user has the ability to tell *Live Paper* that a match is invalid, the system would have difficulty finding the correct match.



	1	2	3	4	5	>5
Cumulative Page Count	724	788	818	843	850	1026

Figure 5-18: Cumulative Rankings for Valid Matches Using the Euclidean Distance Measure

Figure 5-19 partly explains the poor results of the Euclidean distance measure. There is significant overlap between the distances to the correct matches and the distances to the closest incorrect matches.

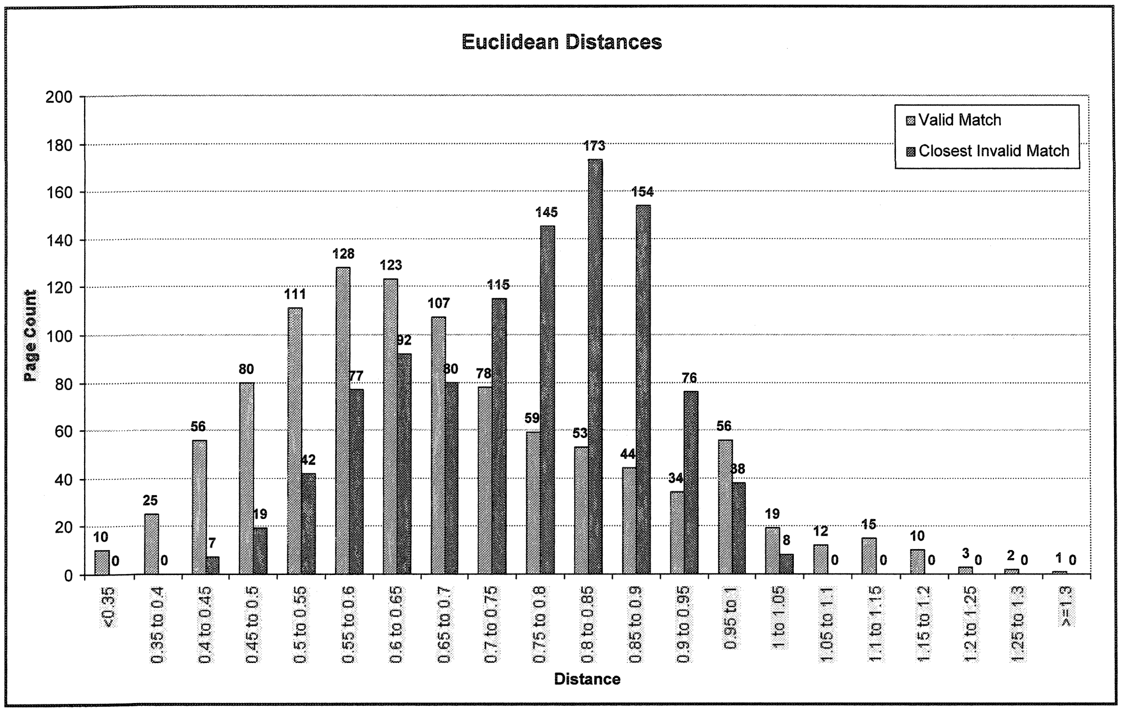


Figure 5-19: Distribution of Euclidean Distances for Valid Matches, and the Closest Invalid Match

It is also apparent from Figure 5-20 that the ratios of the best match to the second best match (the confidence ratio) are generally low for valid matches. Figure 5-21 shows the success rates for various possible confidence ratio thresholds. It is possible to achieve a high correct match rate by setting the threshold at 1.2, but this would reject 61.3% of all matches.

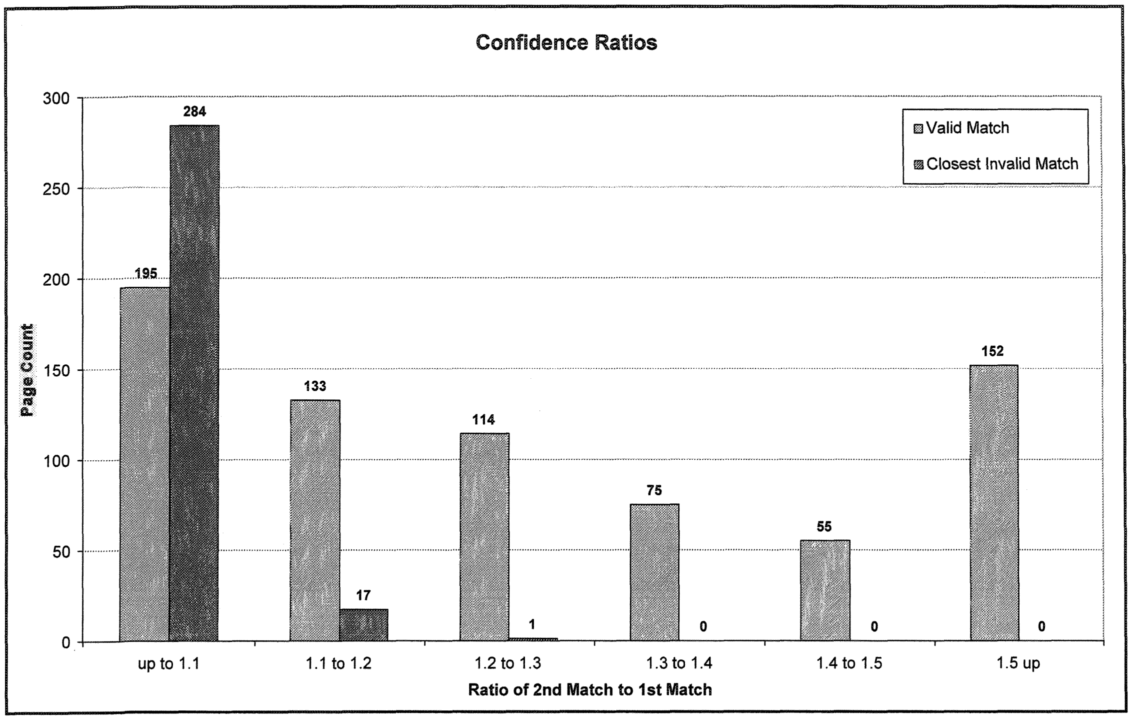
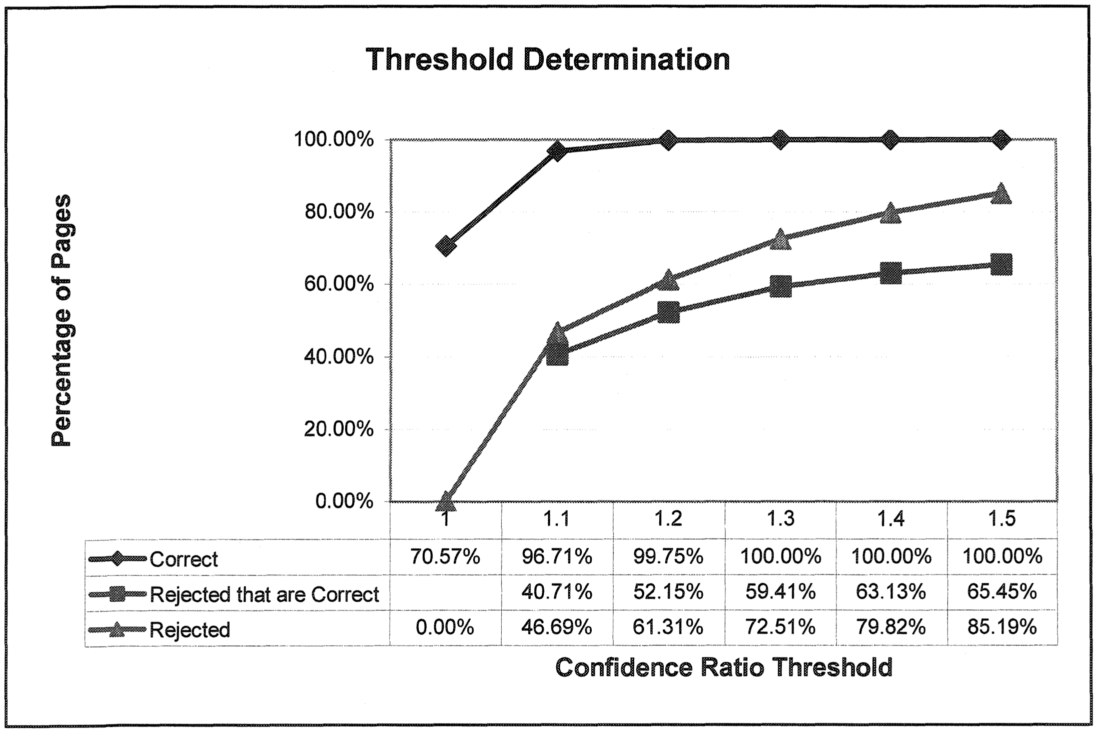


Figure 5-20: Confidence Ratios for Euclidean Distance Measure



	1	1.1	1.2	1.3	1.4	1.5
Correct	70.57%	96.71%	99.75%	100.00%	100.00%	100.00%
Rejected that are Correct		40.71%	52.15%	59.41%	63.13%	65.45%
Rejected	0.00%	46.69%	61.31%	72.51%	79.82%	85.19%

Figure 5-21: Success Rates for Various Confidence Ratio Thresholds

The Euclidean is a poor measure for identifying candidate pages against a database of general page images. This is not surprising considering that the page images contain significant areas of bi-level intensity. A more sophisticated grey-level classifier is unlikely to do better with the test set due to the low number of samples (six) compared with the number dimensions (the size of the page image, which is 53 x 44, or 2332).

5.4.6 Paintings

For most documents that would be laid on a desk in an office environment, common sense indicates that they would be textual documents such as reports or letters. Some of these pages might have large pictures on them, such as graphs or screenshots.

To test the extreme, the author created 49 pages, each of which contains a single large grey-scale picture that encompasses most of the page. Figure 5-22 has three sample images of pages containing paintings. It was hypothesised that the Euclidean distance measure would work very well on this test set, and this was found to be true. The success rate was 99.7% (1 page was missed). The graph in Figure 5-23 shows the distribution of distances.

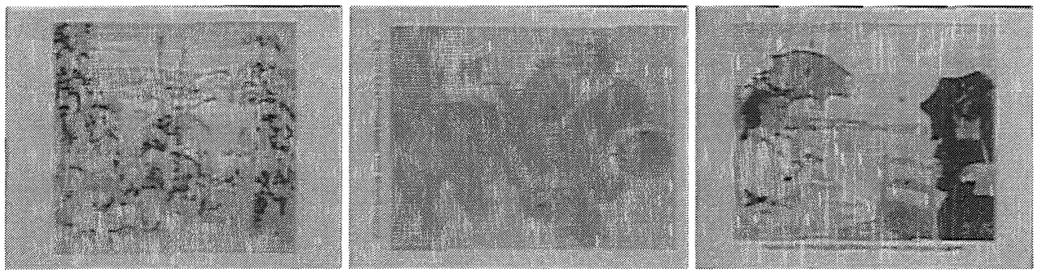


Figure 5-22: Sample Images of Pages with Paintings

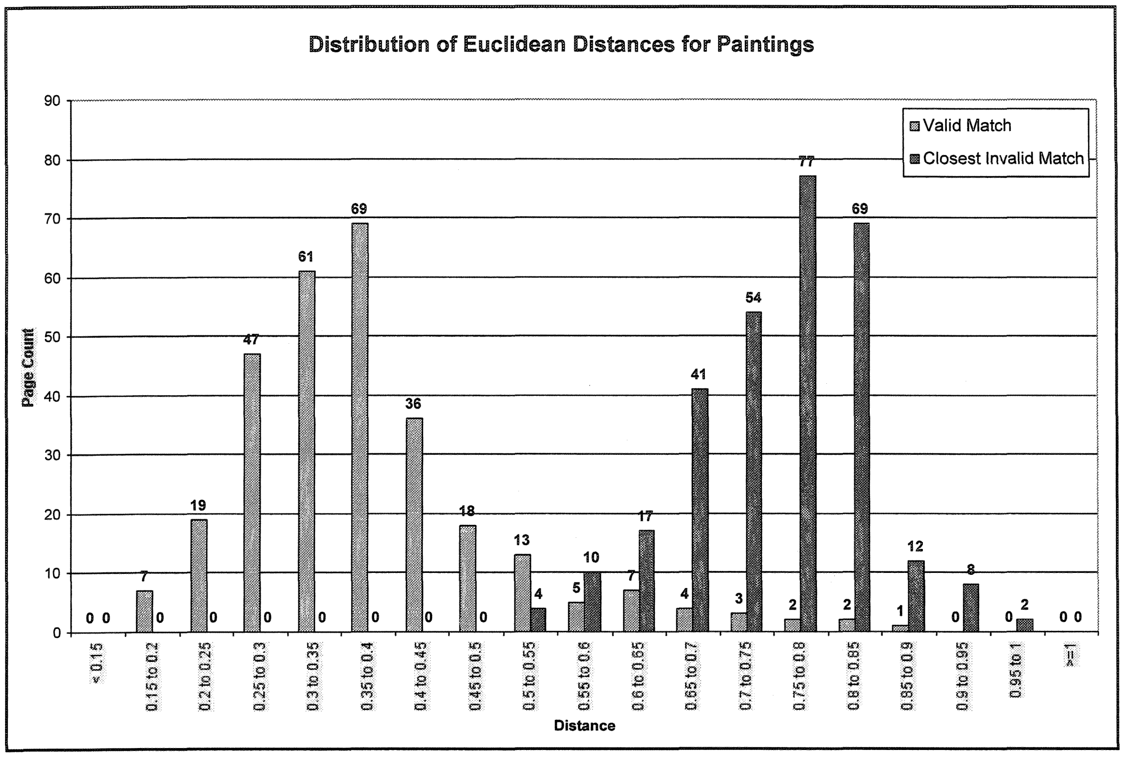


Figure 5-23: Distribution of Euclidean Distances for Painting Test Set

The combination of the Wellner algorithm (section 5.3.1) and the Hausdorff distance (section 5.3.4.1) also handles the case of paintings with reasonable accuracy. On the set of 49 pages, testing showed that the algorithms produced an accuracy of 92.5% using parameters for the Wellner algorithm suitable for general pages (s = 16 and t = 3). The graph in Figure 5-24 shows the distribution of distances. The distribution unfortunately does not easily separate into two groups.

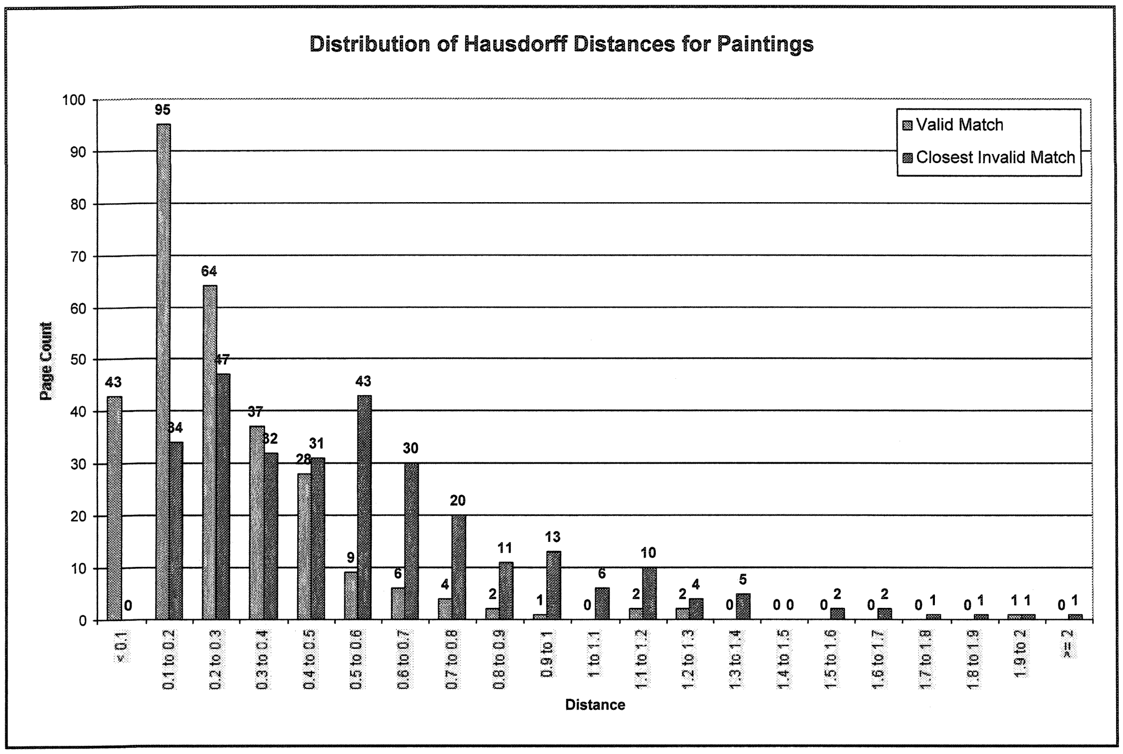


Figure 5-24: Distribution of Hausdorff Distances for Painting Test Set

In a separate experiment where the parameters for the Wellner algorithm were optimal for paintings (s = 11 and t = 7), the success rate for the Hausdorff method was 98.1%.

The results indicate that the Hausdorff distance performs adequately on pages with significant grey-level regions. If the specific task for which *Live Paper* required such pages, then the grey-level Euclidean distance measure could replace or supplement the Hausdorff distance.

5.4.7 K-Nearest Neighbours

Another test used the K-Nearest Neighbour method [Therrien 1989] with the Hausdorff distance measure to classify the pages. The method uses the *K* page images most similar to the inspected page image to determine the matching page. The match is the page whose image is most common in the set of images. However, if no page image is the most common, then the method cannot determine the matching page.

For each page, there were 6 captured images for the test set, and 1 additional image used as the candidate page, for a total of 7 images. By considering 6 images at a time with an additional test image, there are 6 separate image sets for each page. The results can be seen in Figure 5-25, based on a test set of 257 pages (including hand-drawn, presentation, journal, and painting). When $K = 1$, 92.1% of the pages were matched correctly. However, as K increases, the success rate drops.

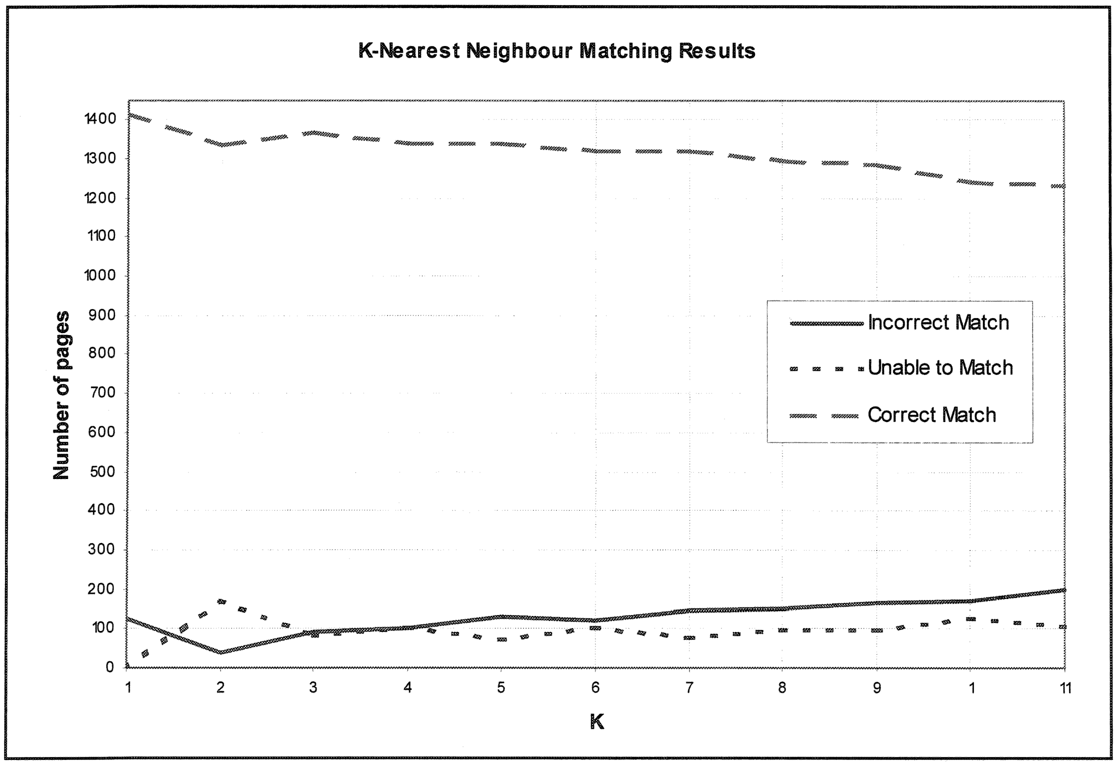


Figure 5-25: Results for K-Nearest Neighbour matching for increasing *K*.

The best value of K is 1, which corresponds to using the single best match. This indicates that there is overlap between classes, and that there is significant variance within the class. The results might be improved by capturing more images for each page, thus expanding the number of samples in each class.

5.5 Experimental Conclusions

The Hausdorff distance works well in selecting the correct page from a database of stored page images. Although the Euclidean distance measure works better when pages are dominated by grey-level, for pages with text, the Hausdorff distance is a better measure.

Ideally, *Live Paper* would be able to perfectly identify every page it sees, and determine whether or not the page is new. However, the system is interactive, and this means that the user has the opportunity to correct the system. A simple user interface would let the user indicate to *Live Paper* that it found the wrong page. This would require some additional research and design to determine a suitable interface so that the user would quickly know which page was found. For some transparencies, a mismatched page would be immediately obvious, but others would be more difficult for the user to recognize. In addition, the system itself would have to maintain more information about the page, so that it could effectively undo the erroneous recognition. For example, once a page has been recognized, *Live Paper* currently updates the last image of the page. To undo this, the system would have to store older versions of the image as well.

The results from the higher resolution experiment indicate that a hierarchical scheme should be investigated further. Pages could be extracted at low resolution and compared with the database. If the distance was too great, or the confidence ratio too low, then a higher resolution image could be captured and used for matching.

5.6 IMPLEMENTATION

5.6.1 Texture Mapping

From a set of four corners, *Live Paper* uses a bilinear texture-mapping algorithm [Wolberg 1990] to map pixels from a rotated page in the desk image to the pixels in the page image. The texture-mapping algorithm uses nearest neighbour interpolation. The algorithm does not account for perspective distortion.

5.6.2 The Hausdorff Distance

The moving averages algorithm is applied to each page that is texture mapped from the original desk image. In the resulting two-level image, each pixel that is on is considered to be included in the set of points associated with that image. The new image is compared with every other image already stored in the database by calculating the modified Hausdorff distance (Equation 5-12) using the $(L_2\text{-Norm})^2$ (Equation 5-11). The smallest distance, provided it is below a set threshold, indicates which page matches with the new one. Since the orientation of the captured page image could be rotated 180° with respect to the stored image, both orientations must be tried.

Real-Time Execution

It is not feasible to fully compute the Hausdorff procedure for each page-to-page comparison in a real-time system such as the Live Paper. *Live Paper* might have to search hundreds of pages for each time it finds a new page. Several techniques used by *Live Paper* to decrease execution time of the Hausdorff algorithm are described below.

The first technique is to pre-calculate the distance transform of each image. Each position in the distance transform, which is a rasterized approximation to the Voronoi surface [Huttenlocher 1993], gives the distance to the nearest 'on' pixel in the corresponding page image. The result is that a search for the nearest point is replaced by a lookup in the transform. Figure 5-26 shows sample L2-Norm Voronoi surfaces (normalized to 256 levels of grey). By using the distance transform, the system only needs to calculate the detailed distance once.

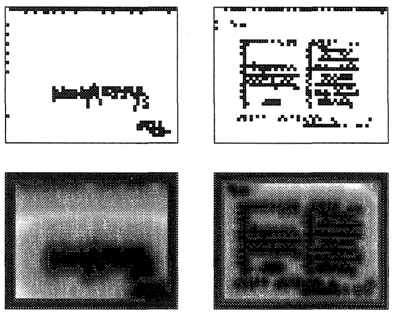


Figure 5-26: Voronoi surfaces for given pages.

The second technique is to use a hard-coded distance template when creating a transform. The template is centred on the location in the Voronoi surface that corresponds to the point in the original image. Those pixels that are greater than the pixels in the template are replaced with the lower value. Unless the template is larger than the source image, the calculated Hausdorff distance will be approximate; however, small templates allow for fast creation of the surfaces.

Another technique is to convert the original image into a list of coordinates. Thus the entire image will not need to be scanned. If the image needs to be rotated, a second list of rotated coordinates can be stored. The Voronoi surface can also be generated from this point list. The final technique is to use the smallest possible image size. This reduces the total possible number of pixels that are 'on' in an image.

5.6.3 System Implementation

Live Paper uses the Hausdorff distance to determine whether or not the pages (candidates) found in the most recently captured desk image are already stored in the system. *Live Paper* also uses the distance as a backup measure to the tracking algorithm. The following procedure is used:

1. All candidate pages (including parts of pages) are extracted from the captured video image of the tabletop.

2. Page tracking, as described in chapter 4, is used to associate as many candidates with the model pages (those pages considered to be on the tabletop) as possible. In Figure 5-27, model pages are those that are in the 'ON' or 'Removal' state. Every successful match removes a candidate from the list.

3. If any full-page candidates remain in the list, then each is ranked against the full-page database using the Hausdorff distance.

4. *Live Paper* then starts with the best ranks, and steps through the list of remaining model pages. For each model page, the system checks the candidates' match for the given rank. If the model page matches the candidate at that rank, and if the Hausdorff distance is below a threshold, then the match has been found. The system removes the candidate and the model pages from their respective lists.
   The system then increments the rank, and tries again with the remaining model and candidate pages. This continues until all model pages have been matched, or the rank becomes too large. Based on the experiments given earlier in this chapter, a rank of 5 is used as the stopping value.

5. The model pages consist of active ('ON') pages, and those in a deactivating ('Removal') state (Figure 5-27). Old deactivating pages are removed from the model list – their state is set to 'OFF'. Any remaining active pages are set to deactivating.

6. All remaining page candidates are checked to see whether they are new pages, or inactive pages already in the database. If the best match (rank 1) has a Hausdorff distance less than a threshold, then a match has been found, and the page is activating and updated. Otherwise, the page is new, and is thus created and added to the database.

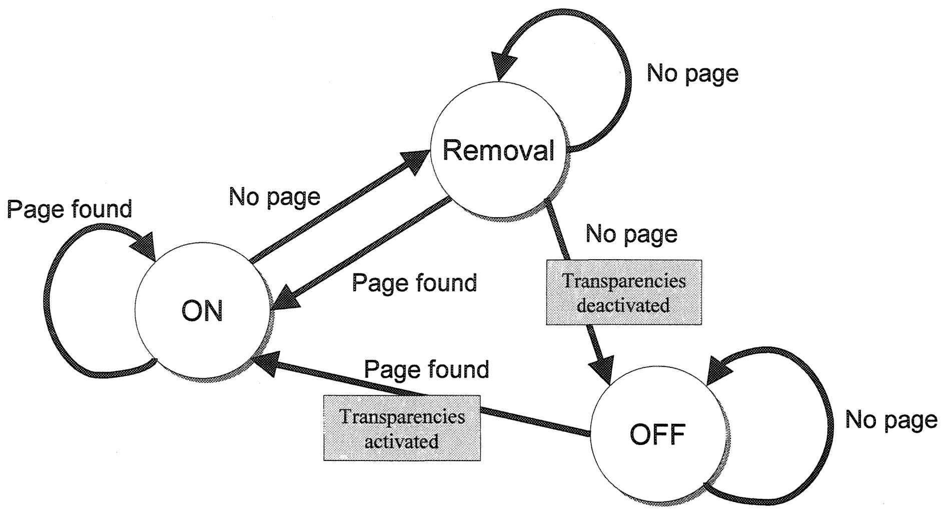


Figure 5-27: Page updating states.

The update mechanism has the advantage that transparencies are not deactivated immediately if the system loses track of the page for a couple of frames. The disadvantage is that the transparencies do not deactivate immediately after the user removes the page from the table. The projected annotations remain on the tabletop for a short but noticeable period of time (about 3 seconds).

From informal experiments and general usage of the system, this algorithm seems to work well. The system can locate and recognize a page, and display the annotations, within two seconds of placing the page. When a page moves, the projections follow it. Further optimizations could be added. For example, the page-ranking algorithm ranks the candidate against all pages in the database; instead, it could use a threshold value to quickly reject those pages that could not be a match. In step 6, the system could also request confirmation from the user that the page is new.

Chapter 6

REGISTRATION

This chapter explains the need for accurate registration between the image captured by the video camera, the image displayed by the projector, and the tabletop. Registration is the process of establishing point-to-point correspondences amongst the regions. The chapter explains the mathematics involved with several registration schemes, and then reports on alternatives tested for use in *Live Paper*.

6.1 MOTIVATION

The basic functionality of *Live Paper* is to provide dynamic virtual annotations to physical pieces of paper. The system must know how to project the annotations such that they appear in the correct positions either on or next to the paper. The system must also determine when the user has used his or her finger to select a projected user interface element. Thus a fast and accurate registration scheme is needed so that the *Live Paper* system can map amongst elements in the captured image, in the projected image, and on the physical tabletop (see Figure 6-1).

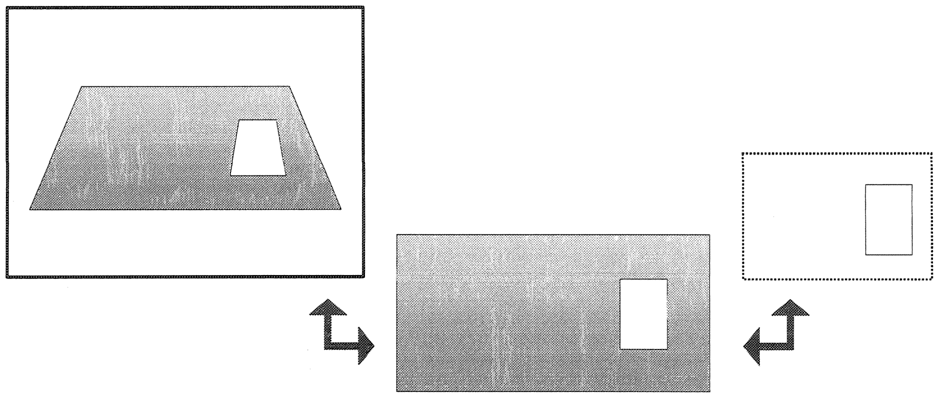


Figure 6-1: Reconciling page positions.

The captured image (left), the desktop (middle), and the projector space (right).

When the various regions are registered, the system can determine the true size and location of pages found in the captured image. This information can be used to track moving pages across multiple frames. The system can determine whether the velocity is reasonable. It can also calculate the position of hidden corners of the page, even if only one corner is visible.

Annotations that are tightly tied to a particular location on a page, such as in the architecture transparency, require very good registration between the projector, the tabletop, and the captured image. Interactive annotations, such as buttons and menus, also need good registration so that the system can determine when the user has selected an item.

6.2 BACKGROUND ON REGISTRATION

A spatial transformation is "a geometric relationship between each point in the input and output images." [Wolberg 1990] Transformations are used to translate points to new positions, either in the same spatial domain or mapped to a new domain. In the context of the Live Paper, all transformations are two-dimensional.

There are several types of spatial transforms listed in the general literature. The perspective and bilinear transforms are the simplest four-point geometric transforms. This section briefly examines these two transforms, as well as a transform for radial correction.

6.2.1 Perspective

Perspective transformations correspond to perspective projections – transformed lines that were originally parallel appear to converge to a single point, unless they were also parallel to the projection plane. Parallel lines in the output set grow closer together as they approach the perspective point. All lines are preserved, including diagonals.

If (u, v) are the original coordinates, then the following is a perspective transformation that will give the coordinates (x', y'):

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{6-1}$$

or equivalently:

$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{12}v + a_{13}}{a_{31}u + a_{32}v + 1},$$

$$y = \frac{y'}{w'} = \frac{a_{21}u + a_{22}v + a_{23}}{a_{31}u + a_{32}v + 1}. \qquad (6\text{-}2)$$

There are eight coefficients in the perspective transformation. The ninth coefficient (a_{33}) is normally considered as a 1, and is shown as a 1 in Equations 6-1 and 6-2. However, if a_{33} is not 1, then normalizing the transformation matrix by dividing with a_{33} will not affect x or y. Thus four pairs of non-collinear points are necessary to fully define the transform.

Figure 6-2 gives an example perspective transform of a square grid.

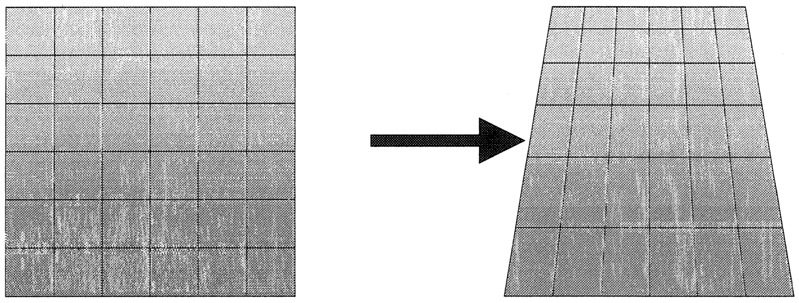


Figure 6-2: Perspective Transform Example

6.2.2 Bilinear

The *bilinear* transformation is a simple four-point warping algorithm that preserves straight lines from the input point set. The *bilinear* transformation from (u, v) to (x, y) is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_3 & a_2 & a_1 & a_0 \\ b_3 & b_2 & b_1 & b_0 \end{bmatrix} \begin{bmatrix} uv \\ u \\ v \\ 1 \end{bmatrix}, \tag{6-3}$$

or equivalently:

$$x = a_3uv + a_2u + a_1v + a_0,$$

$$y = b_3uv + b_2u + b_1v + b_0. \tag{6-4}$$

To solve, four points are needed in the input set and their corresponding locations must be known in the second set. Solving the *a* and *b* coefficients requires the solving of two matrix equations. For example, to solve for the *a* coefficients, the following matrix equation must be solved:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 & u_0 & v_0 & u_0v_0 \\ 1 & u_1 & v_1 & u_1v_1 \\ 1 & u_2 & v_2 & u_2v_2 \\ 1 & u_3 & v_3 & u_3v_3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}. \tag{6-5}$$

Figure 6-3 gives an example of a bilinear transform applied to a square grid.

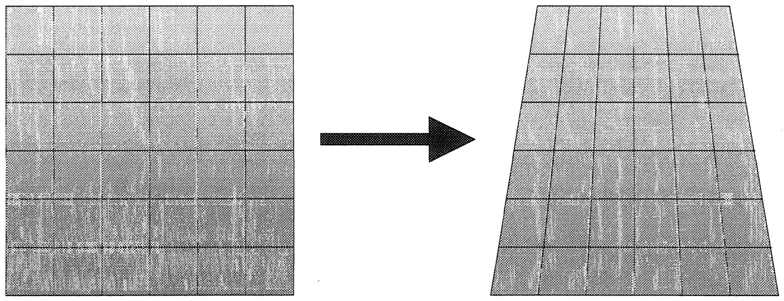


Figure 6-3: Bilinear Transform Example

6.2.3 Radial Correction

Radial correction can be used to correct for barrel and pincushion distortion caused by the lens optics. The amount of correction for a position is related to its distance from the centre of distortion. An explicit model of radial and decentering distortion [Mohr 1996] uses a 6th order polynomial, but a simpler approximation can be made (see Equation 6-7).

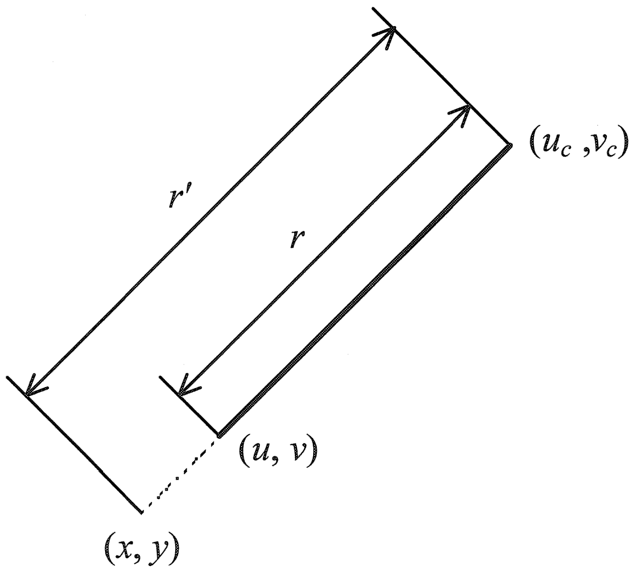


Figure 6-4: Radial Correction

In Figure 6-4, u_c and v_c are the center of the radial distortion. The point (u, v) is the original position, and (x, y) is the position after radial correction. Based on Figure 6-4, the following relationships hold true:

$$\begin{aligned} r^2 &= (u-u_c)^2 + (v-v_c)^2 \\ r'^2 &= (x-u_c)^2 + (y-v_c)^2 \ . \end{aligned} \tag{6-6}$$

The basic equation for radial correction is

$$r' = r(1 - Cr^2), \tag{6-7}$$

which can be rewritten as

$$\frac{r'}{r} = 1 - Cr^2 \, . \tag{6-8}$$

The ratio of the new radius to the old radius holds in both the x-axis and y-axis, as follows:

$$\frac{x-u_c}{u-u_c} = \frac{y-v_c}{v-v_c} = \frac{r'}{r} \ . \tag{6-9}$$

Thus we can rewrite Equation 6-8, substituting the x-terms for the radius ratio:

$$\frac{x-u_c}{u-u_c} = 1 - Cr^2 \, . \tag{6-10}$$

Solving for x gives

$$x = (1 - Cr^2)(u - u_c) + u_c \ . \tag{6-11}$$

The process of rearranging Equations 6-10 and 6-11 can be duplicated for *y* to give:

$$y = (1 - Cr^2)(v - v_c) + v_c \ . \tag{6-12}$$

Because the same centre point is used for both *r* and *r'*, the following inverse transforms take the same form as those in Equations 6-11 and 6-12:

$$u = (1 - C'r'^2)(x - u_c) + u_c , \tag{6-13}$$

$$v = (1 - C'r'^2)(y - v_c) + v_c \ . \tag{6-14}$$

The correction constant is different for the inverse transform. It can be determined after a forward calculation has been completed to give one (*u*, *v*)-(*x*, *y*) pair.

$$\begin{aligned} u &= (1 - C'r'^2)(x - u_c) + u_c \\ 1 - C'r'^2 &= \frac{u - u_c}{x - u_c} \\ C'r'^2 &= 1 - \frac{u - u_c}{x - u_c} = \frac{x - u}{x - u_c} \\ C' &= \frac{x - u}{(x - u_c)r'^2} \end{aligned} \quad . \tag{6-15}$$

Thus full radial correction requires determination of the constant *C*, and the centre of radial distortion (u_c, v_c).

The images in Figure 6-5 provide a series of comparison images to illustrate the nature of barrel and pincushion distortion in an image from a video camera. The original image in Figure 6-5 (a) exhibits slight barrel distortion, evident in the curves along the edges of the tabletop.

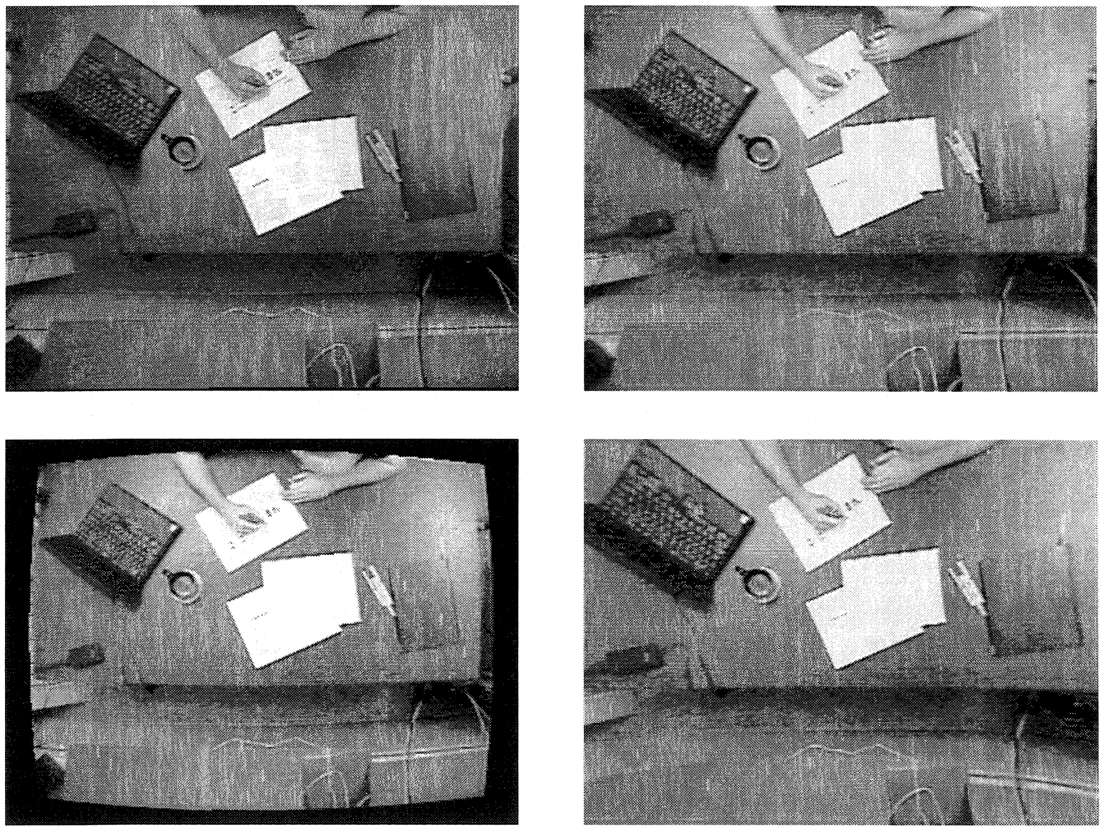


Figure 6-5: Examples of radial distortion and correction. Brightness and contrast have been adjusted to enhance details.

(a) Original image with slight barrel distortion (b) Correction applied

(c) Additional barrel distortion applied to original image (d) Too much correction applied, resulting in pincushion distortion.

6.3 IMPLEMENTATION

6.3.1 Requirements

Live Paper must be able to determine whether or not an extracted page candidate is of the correct size to be a letter-size page or the size of a business card. Projected annotations, especially those associated with elements on a page, must be laid precisely. The system must be able to determine the locations of user interface projections, such as buttons, in the captured image.

6.3.2 Assumptions

The research assumes that the natural limits of precision of the equipment are reasonable. For the camera, one captured pixel corresponds to approximately 5 mm on a side of captured tabletop space. Each pixel from the projector occupies approximately 1 mm on a side. Thus annotations that are projected onto an extracted page could be up to 3 mm away from the correct position, even for perfect registration.

6.3.3 Process

Live Paper performs registration by establishing independent coordinate systems for each reference frame, and then using a set of transformations to convert coordinates between systems. Each transformation is based on a quadrilateral region, and is set during a manual registration phase. The *Live Paper* has three key coordinate systems: *image*, *desk*, and *display*. It also has a fourth coordinate system, called *camera*, which can be used to centre the camera on a particular position on the tabletop.

The *desk* coordinate is the base coordinate – all other conversions between coordinates use the *desk* coordinate. Thus to convert from image to display, requires a conversion from *image* to *desk*, and *desk* to *display*. This reduces the number of transform matrices required.

A set of spatial transforms is used by the system to do the conversion. These are established by manually registering to points on the desk surface. Each set requires four pairs of points.

Due to the complex interaction between camera position and zoom settings, and the location of the desk in the resulting image, *Live Paper* sets a specific transformation for each set of camera parameters. Thus the coordinates of objects in an image acquired while the camera is not in a previously analyzed position will not be converted to desk coordinates.

6.3.4 The Coordinate Systems

Image coordinates indicate the position of an object in the image captured by the camera. The *desk* coordinates give the location of an object on the desktop in discrete units. The *display* coordinates are used by the projector system to place displayed information. *Camera* coordinates give the pan and tilt values (in degrees) of the computer-controlled camera to place an object at the centre of the viewing area. See Figure 6-6 for a diagram of some of the coordinate system values.

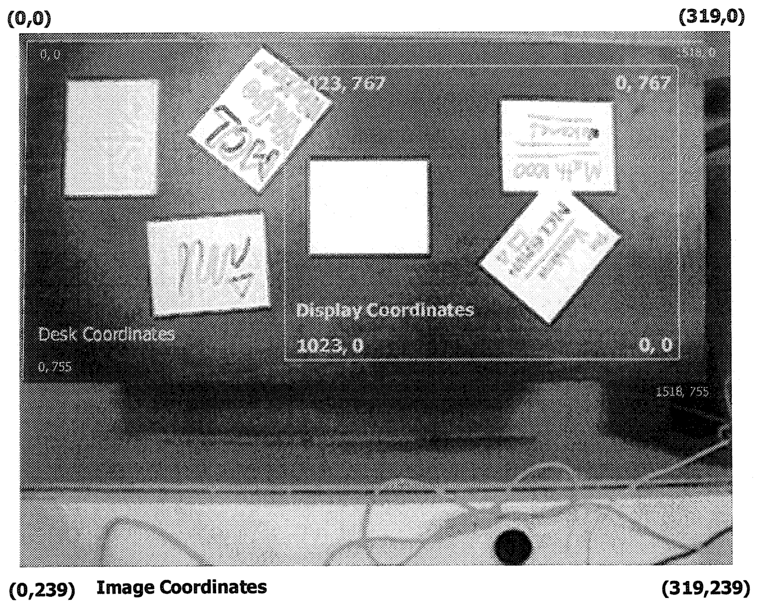


Figure 6-6: Relationship between *Image*, *Desk*,and *Display* coordinates

6.3.4.1 Image

Each image is captured as a raster 24-bit colour image with a width *w* and a height *h*. *Image* coordinates are simply discrete values used to address individual pixels. The valid range for *image* coordinates depend on the image size – they range from (0,0) to (*w*-1, *h*-1). The upper left pixel of an image always has a coordinate of (0,0).

The mapping of *image* coordinates to other coordinate systems depends on the pan, tilt, and zoom values of the camera. In theory, mappings to other coordinate systems could be set up for any arbitrary pan-tilt-zoom combination. However, in practice for the *Live Paper* system, this added level of complexity was not needed. The current system establishes two settings for the camera, and sets up independent mappings for each. One setting encompasses the entire desktop, and the other focuses on the writing area immediately in front of the user.

6.3.4.2 Desk

The decision about the form of *desk* coordinates considered units of relative and absolute sizes. A relative unit would be the result of establishing a fixed grid, say 1000x1000 units, for a desktop. The physical size of each unit would depend on the size of the desk. Conversely, absolute units would establish a physical size for each unit. Each desktop would have a different number of units in both the horizontal and vertical directions.

Live Paper uses absolute units of one millimetre for the *desk* coordinates. The main advantage is that a distance in *desk* coordinates immediately gives the true distance in millimetres. The upper left corner of the desk, as viewed by the camera, has a value of (0,0).

6.3.4.3 Display

Display coordinates are discrete numbers that correspond to locations on screen. When the data projector is in use, the Live Paper's output is redirected from the computer screen to the desktop. The range of the display coordinates depends on the settings of the video card and capabilities of the data projector. For the current setup, the screen resolution is 1024x768, and so the range of the display coordinates is (0,0) to (1023,767).

6.3.4.4 Camera

Camera coordinates are real numbers, given as a pan/tilt pair. The values are in degrees, and can be both positive and negative. The limits are given by the physical properties of the computer-controlled camera. For the Canon VC-C3 video camera used in the *Live Paper* set-up, the pan limits are left and right 90 degrees, and the tilt limits are up 25 degrees and down 30 degrees.

6.3.5 Implementation Alternatives

At the extreme, there are two alternatives for registration algorithm. One possibility would be to construct a single lookup table that would include the transformed position for every original position. Although the size would be quite large (3MB to transform a projector coordinate into a single other coordinate system), the quantity of RAM in standard desktop computers today is about 2 orders of a magnitude bigger. The look-up table would be constructed by collecting a sample of points and then interpolating the remaining. In the case where automated feedback could be used, such as registering the projector directly with the captured image, then nearly every point of interest could be sampled. However, although the amount of computer memory available is large, there are other aspects of the system that could also require significant memory (such as the page database).

A second possibility would be to fully model the camera intrinsic parameters [Mohr 1996]. This includes features such as radial and decentering distortion. Although this model can be extremely accurate ($2x10^{-5}$ of the image size), the required equations are computationally expensive.

The speed of the first alternative and the sub-pixel accuracy afforded by the second alternative were not critical for *Live Paper*, especially when the tradeoffs were taken into consideration. The research explored two alternatives between the extremes during the development of *Live Paper*.

6.3.6 Region Transform Implementation

The original implementation of *Live Paper* used the bilinear transform to convert between the coordinate systems. The bilinear transform is computationally simpler than the perspective transform (only multiplications and additions are required). However, the distortion of the camera optics is such that a number of non-overlapping quadrilateral regions had to be used to convert between *image* and *desk* coordinates. Figure 6-7 shows how the quadrilaterals were arranged – one row and four columns. The use of a single quadrilateral caused significant error in the calculated position of the pages, which is especially noticeable when projecting information.

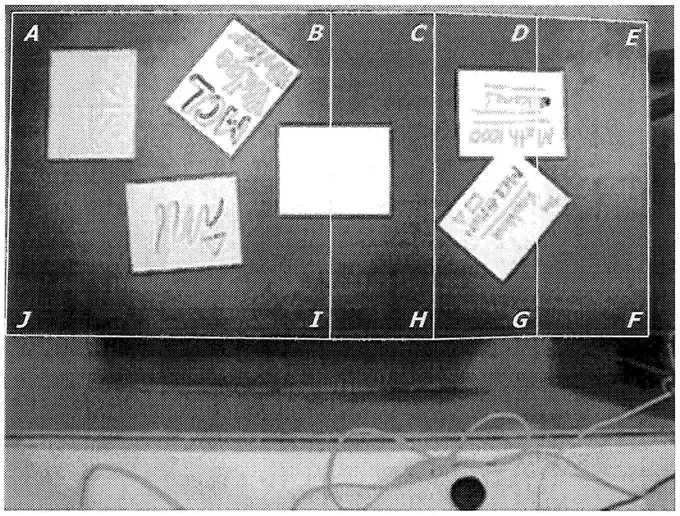


Figure 6-7: Individual transformation quadrilateral regions

(for warping between *image* and *desk* coordinates)

This scheme worked reasonably well for tracking pages and displaying projections to the side of pages. However, the accuracy of the projections onto pages was often poor, and the system could not reliably extract finger information from areas corresponding to projected buttons. Although it would be possible to continue to add regions, the author sought a simpler alternative that would provide greater accuracy.

6.3.7 *Live Paper* Implementation

The research investigated whether the use of perspective in a single quadrilateral would provide the required accuracy for finger detection and annotations on a page. Although the bilinear transform is faster than a perspective transform, it does not match the deformations in the captured image or in the projector image.

By itself, the perspective transform did not provide accurate results. However, the distribution of errors suggested a radial distortion. After applying radial correction to a perspective-transformed point, adequate accuracy was achieved for both *image-desk* and *desk-display* coordinate transforms (see Figure 6-8) within the region of interest. Projections are accurate to within 1 mm of their calculated position over most of the tabletop; the worst case is 3 mm in a small (50 mm x 50 mm) area. For applications that involve *image-desk-display* transformations, such as projecting an annotation onto a page or detecting a finger within a projected button, the system is consistently accurate to within 5 mm. When the camera has zoomed to focus on the writing area, the accuracy is within 3 mm.

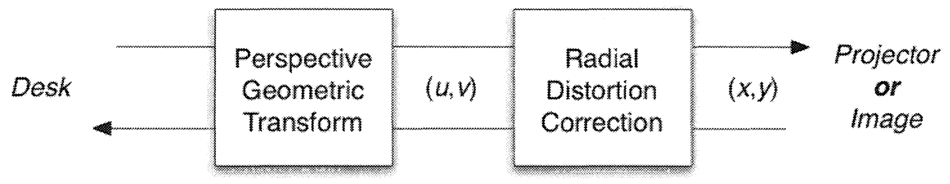


Figure 6-8: Order of operations when converting from *desk* coordinates to *projector* or *image*, or the reverse conversion.

The perspective transform and radial correction can be combined into a single set of operations. It is for this reason that the perspective transform was used and not a region-based bilinear transform. The radial distortion is due to the optics of the system; if a different camera was used, where the radial distortion was negligible, then the region-based bilinear transform would probably be adequate, and would be computationally less expensive.

6.4 COMMENTS

The registration algorithm in *Live Paper* has a level of accuracy adequate to demonstrate the concept of a VAE tabletop. A number of issues arise from the implementation, and these suggest several avenues of future research.

In *Live Paper*, basic page tracking was implemented using two or more corners to update a page location. Thus the user can occlude one or two corners and still move the page. However, because the corners are not located with sub-pixel resolution, inaccuracies arise if a page is moved. Future research could increase the accuracy of the corner extraction algorithm, and thus a single corner would be sufficient for tracking a page through both translation and rotation. Accurate registration is necessary so that the shape of the page can be preserved – it is rectangular in real coordinates, but slightly distorted in image coordinates.

The accuracy of transforming *desk* coordinates to *camera* coordinates was never fully examined. Under normal operation of *Live Paper*, it is not practical to change the camera position to view a specific area of the desk with a higher resolution. It takes up to 3 seconds to move the camera, and one additional second for vibrations to cease. While zoomed, the system is not able to track objects over the entire desk. For moving the camera to be practical, the system needs a second fixed camera that views the entire tabletop, all the time. In such a system, the camera's pan and tilt settings have to be registered with the *desk* coordinates. In addition, a model would have to be developed such that *image* coordinates from an image with an arbitrary set of pan, tilt, and zoom factor values could be transformed to *desk* coordinates. In such a set up, a lookup table scheme could not be used for the registration.

One decision that made it difficult to register the system was to set the lower-right corner of tabletop as the origin of the *desk* coordinates. If the desk was shifted during registration, then the process had to be restarted. However, once registered, the desk could be moved without affecting the transformations. If the system could arbitrarily select an origin for the tabletop, then registration process would be more robust as only relative distances would have to be measured.

Finally, future research should investigate automatic and semiautomatic registration. *Live Paper* could automatically determine the registration parameters between the projector and the camera. A suitable pattern would be projected such that the system could accurately locate it in the captured image. This might require that the user cover the tabletop with a white background. Semiautomatic registration would require that the user place and move a target on the tabletop. The system would be able to determine the location of target markings in the captured image, and would know the real-world distance between the markings. Thus the system would build up a set of point-to-point correspondences for use in registration.

Chapter 7

System Design

This chapter details the hardware and software framework of the *Live Paper* system. *Live Paper* is a first-approximation of a full operating system for augmented environments. This prototype is useful both as a model and for its lessons on what needs to change. The effort expended on *Live Paper* can be harnessed for future development.

The purpose of this chapter is two-fold. The previously presented concepts and algorithms are applied in a functioning system. The chapter also provides background for future development of a tabletop operating system; recommendations are presented in Chapter 8.

Following an overview of the fundamental *Live Paper* design, the section on hardware presents the key physical components of the system. The software section describes the code for the user interface, the storage, the image processing, and the transparencies. The architecture incorporates the algorithms and design solutions presented in previous chapters. The chapter concludes with short descriptions of event processing and how new transparencies are added to the system.

7.1 Fundamental Design

The requirements of *Live Paper* include image-processing routines that can detect and uniquely identify pages on the desktop. The system requires storage for observed pages and their augmentations, and a user interface to interact with the user. Another requirement is that the system be able to run in real-time; there should be minimal delay between the time a user performs an action and the time the system responds to that action.

The *Live Paper* system architecture consists of three key modules (see Figure 7-1), which provide services for image processing, modelling and storing objects, and user interaction. A particular system task, such as detecting the orientation of a page or determining whether the user has selected a button, will often involve routines in more than one module.

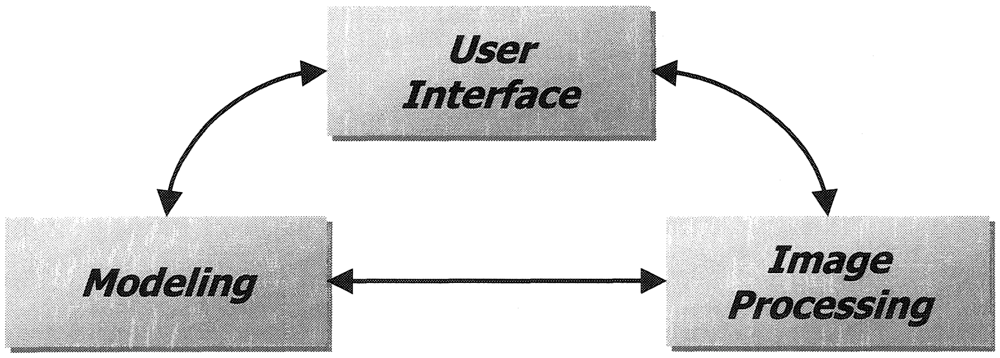


Figure 7-1: ***Live Paper*** **Software Architecture**

7.1.1 Image Processing Module

The Image Processing module captures and analyzes video images, with the intent to find, track, and identify paper. The module detects general motion and the presence of fingers on a button. These routines operate quickly so that the system can respond to user actions in real-time.

Information passed between the Image Processing module and the Modeling and Transparency modules include page manipulations such as moving, creating, hiding, showing, and updating.

7.1.2 Modeling Module

The Modeling module maintains a model of the real-world state of the desk and the pages seen. For example, as pages are identified and tagged, the module stores their locations, orientations, and contents. It also stores references to any associated transparencies.

When updating a page model, this module will call routines with the Image Processing module. The module will also notify the User Interface module when events, such as a page appearing, occur.

7.1.3 User Interface Module

The User Interface (UI) module is responsible for all interactions with the user, including both input and output modalities. Most transparency routines, and basic display widgets like pictures, buttons, and menus, are located in this module.

The UI module interacts with the Modeling module to determine the location of pages, and thus the best location for interface elements. The Image Processing module provides information on selection tasks by the user.

7.2 HARDWARE

The hardware for the system consists of four components: a desk, a computer, a video camera, and a projector. The desk is 30 inches x 60 inches (1518mm x 755mm) with a glossy dark wood finish that is marked by several small scratches and pits. Although the desk has five drawers, the *Live Paper* system does not interact with them.

7.2.1 Computer System

The current computer system has an Intel Pentium III microprocessor operating at 933MHz and uses the Microsoft Windows 98 operating system. The computer has a video/audio capture card (Winnov Videum AV), an Ethernet network card, and a second video display card to which the data projector is connected. The computer has two serial ports for use with the camera and projector.

The available computing power evolved over the duration of the research, with the core microprocessor speeds ranging from 166MHz, to 400MHz, and finally to 933MHz. The capability of the system also grew during this time, with the result that the overall system performance was reasonably consistent.

7.2.2 Image Capture

The video camera is a Canon VC-C3 Communication Camera, which is connected to the computer via a composite cable and an RS-232 cable. The composite cable carries a standard NTSC video signal. The RS-232 cable allows the computer to control camera features such as zoom, pan, tilt, gain, and focus. The VC-C3 can tilt up a maximum of 25° and down 30°; it can pan 90° both left and right (see Figure 7-2).

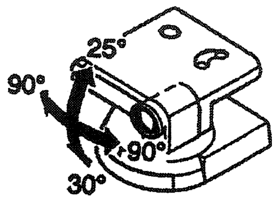


Figure 7-2: Maximum Range of Motion of the Canon VC-C3 Pan/Tilt/Zoom Camera

The camera has a horizontal resolution of 450 TV lines and a vertical resolution of 350 TV lines, which it scans using a 2:1 interlaced method. It has a maximum 10x optical zoom. An attached polarizing filter reduces most of the reflection from the data projector off the tabletop.

The camera is mounted above the desktop on a sub-plate attached at an angle of 25° to a vertical steel plate. The main plate can be repositioned along a horizontal steel beam. See Figure 7-3 for a diagram of the mounting, and a photograph of the set-up. The plate is aligned over the back edge of the desk, at a height of 170 cm. This permits the camera to tilt such that the back edge of the desk is always within the field of view.

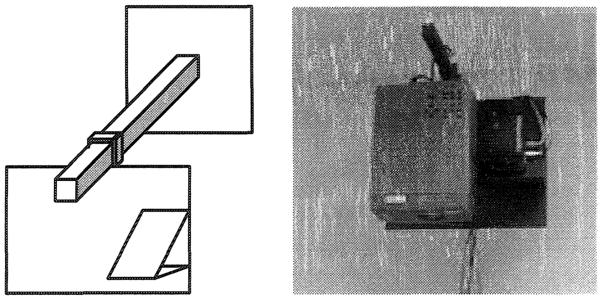


Figure 7-3: (a) Mounting Diagram (b) Photograph of Mounted Projector and Camera

7.2.3 Data Projection

The data projector (see Figure 7-4) is a Telex P600 LCD projector capable of displaying true XGA resolution (1024 x 768). It has a brightness rating of 750 ANSI lumens, which is bright enough for the user to see projected annotations on the desktop in normal office lighting. In addition to the RGB video cable, the projector is also connected to the computer by an RS-232 cable. The computer can thus activate or suspend the video output of the projector as needed.

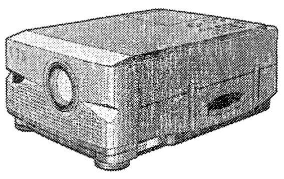


Figure 7-4: Telex P600 LCD Projector

The projector is mounted next to the video camera on the vertical steel plate (see Figure 7-3). It projects directly onto the desktop, with a maximum throw area of 922mm x 694mm.

7.3 SOFTWARE

All of the image processing and storage functions of *Live Paper* are implemented in software. *Live Paper* was written in the C++ programming language, and runs as a Microsoft Windows 98 application. The *Live Paper* application is started using the Windows Explorer file viewer. The primary monitor is used to display information about the internal state of *Live Paper*, while the second monitor output is directed to the data projector and onto the tabletop. Some system calls are made through a library called MCLGallery, which was created by Li-Te Cheng for the Multimedia Communications Lab [Cheng 1998]. MCLGallery provides a simple software interface to several system APIs. For example, it contains functions for image capture and communications via a network or a modem. The library was written in the Sybase Power++ development environment in the C++ programming language.

Figure 7-5 shows the overall architecture of *Live Paper*. Each layer will be described briefly in this section, along with a summary of code modules. *Live Paper* consists of about thirty-one thousand lines of code.

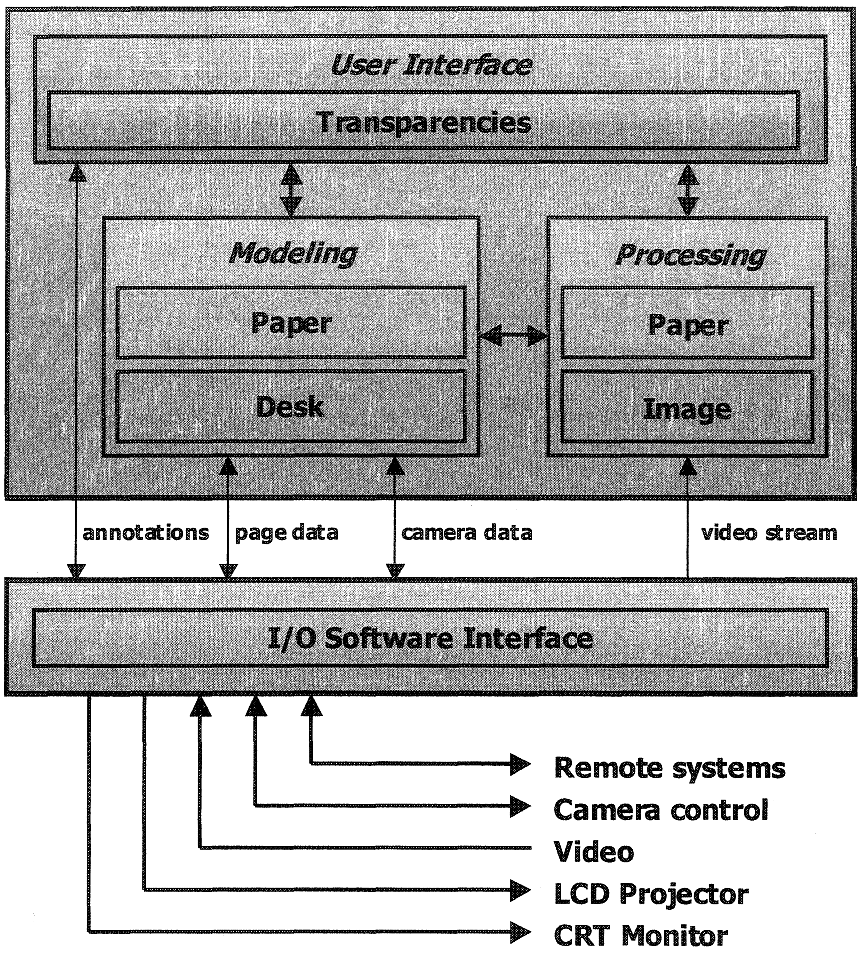


Figure 7-5: ***Live Paper*** **Software Architecture**

7.3.1 Input/Output Software Interface

This layer provides abstraction between the specific hardware and the processing algorithms. Thus *Live Paper* could use a different video camera, video capture board, or communication device with no changes to the modeling, processing, or user interface layers. This component contains specialized routines for the hardware used – for example, there are routines for turning on and off the data projector, and for controlling the camera. It also allows access to a variety of other technologies, including Windows device drivers and DirectX. Operations such as video capture are handled by the Windows operating system.

The implementation of *Live Paper* provides a standard Microsoft Windows GUI to debugging and other system details that would not normally be available to a user. The GUI is displayed on a CRT monitor beside the *Live Paper* tabletop. The data include processing information about the captured image, a list of pages in the database, a list of pages known to be on the tabletop, and representations of Hausdorff data for a selected page. It is through this interface that the user can add transparencies to a new or existing page, and can pause or quit the *Live Paper* application.

Table 7-1 lists four of the most significant classes in this layer and provides a brief description of their purposes. This layer has a total of 3475 lines of code.

Class Name	Purpose	Lines of Code
Camera	Interfaces to capture hardware and camera control commands.	975
Network	Opens a network port for communication with remote networked applications.	262
3D	Renders 3D wireframe and hidden-line drawings.	1536
MusicBox	Interface to system for music playback.	452

Table 7-1: Significant classes for I/O Software Interface Layer

7.3.2 Processing

The processing layer provides functions both for processing captured images of the desk (as described in Chapter 4), and for analyzing the extracted images of pages (as described in Chapter 5). Image processing functions include finding the optimal paper-tabletop segmentation threshold and finding, extracting, and analyzing boundary contours. Paper processing extracts the page images and creates Voronoi diagrams and point sets for Hausdorff analysis.

Table 7-2 lists five significant classes that are part of the processing layer. The entire layer consists of 7647 lines of C++ code. The class relationship is shown in Figure 7-7.

Class Name	Purpose	Lines of Code
ImageAnalyzer	Contains the algorithms for finding pages and extracting contours.	558
ChainCode	Stores and analyzes boundaries.	1626
Hausdorff	Creates and stores data structures for comparing Hausdorff distances.	752
Image	Stores images (of tabletop, pages, or widgets) and provides functions such as copying, basic image processing, and merging.	2707
Segmentation	Provides functions for finding and applying thresholds (such as the Otsu) to images.	711

Table 7-2: Significant classes for Processing Layer

7.3.3 Modeling

The modeling layer provides two sets of *Live Paper* functionality: it models the current state of the tabletop, including which pages are present and where, and it models the contents, both physical and augmented, of individual pages.

The layer models the tabletop from information extracted from the captured images. On each update cycle of *Live Paper*, the Desk (tabletop) class grabs the most recently captured image from the Camera class, and uses the ImageAnalyzer class to segment the image, to find the page boundaries, to extract the boundaries and find the salient corner features, and then to extract page candidates. The Desk class then attempts to match the candidates to its existing list of pages using the corner locations. Remaining page candidates are matched against the full page database based on the Hausdorff distance metric. Unmatched page candidates are added to the database, and pages which were present on the desk in the previous cycle are marked as removed. The Desk class uses a Folder container class to store the pages. When *Live Paper* starts, the folder is loaded from permanent storage, and is saved when *Live Paper* shut downs.

Each page is stored in an instantiation of the Page class. The Page class uses the processing layer (and the Hausdorff class specifically) to create and maintain the Voronoi diagrams for the page. The PageLocation class stores the location of page, and determines the orientation of page. The Page class also has the last extracted image of the page.

Figure 7-6 gives the key for the class diagram in Figure 7-7 and Figure 7-8. Figure 7-7 gives the relationships for classes in the processing and modelling layers. Table 7-3 presents the five key classes or class groups in modelling layer. The total number of lines of code is 7068 lines.

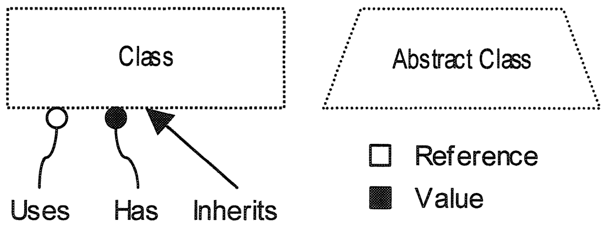


Figure 7-6: Symbol key for class diagrams in Figure 7-7 and Figure 7-8

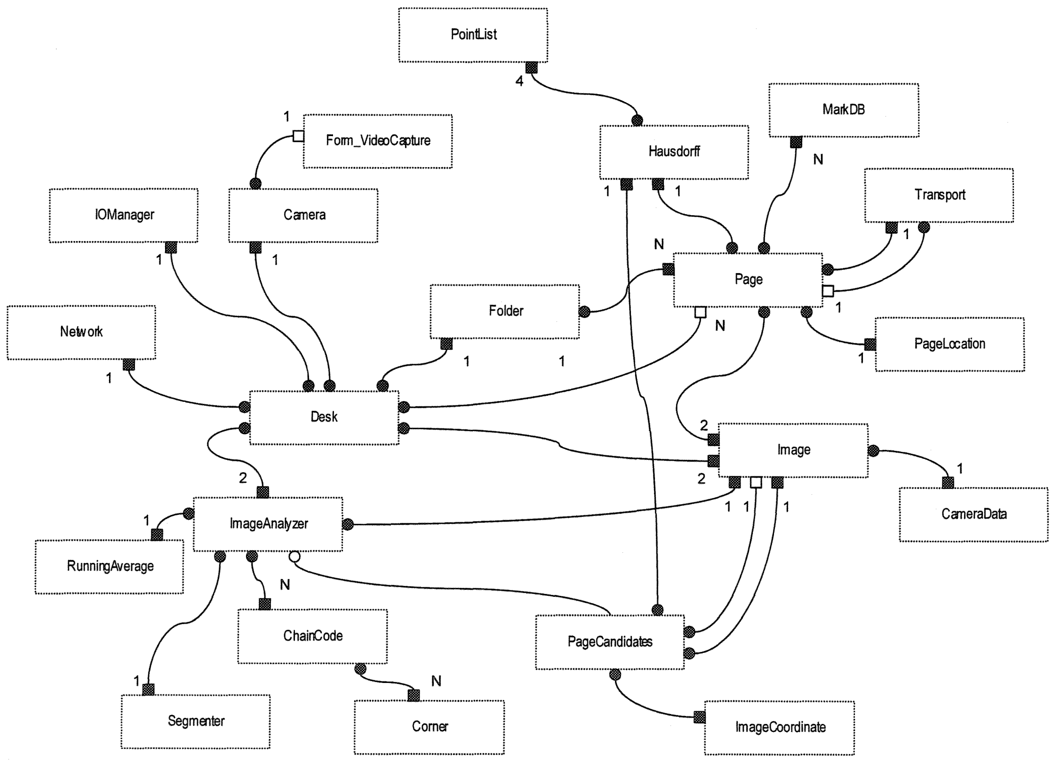


Figure 7-7: Class diagram for the processing and modelling layers

Class Name	Purpose	Lines of Code
PageCandidate	Stores candidates for pages; also provides functions for matching to existing pages and tracking orientation.	317
Coordinate (all)	Provides easy conversion between coordinate systems (such as desk, image, projector, etc.)	1530
Desk	Updates pages based on information from the captured video images.	518
Folder	Container classes for pages. Allows for easy searching and ranking of pages.	1586
Page	Stores an individual page, including location and image of contents.	758

Table 7-3: Significant classes for Modeling Layer

7.3.4 User Interface

The user interface layer is responsible for activating and updating the transparencies, and for displaying transports and I/O widgets. It contains the display component and the core I/O management component. The I/O manager maintains a list of active widgets and their locations, and notifies the widgets if the user has selected them with his or her finger.

The Transport class is owned by the Page class, and contains the list of all transparencies that have been added to that page. It is responsible for activating and deactiving the transparencies. *Live Paper* currently has seven transparency classes, including an

Architect class, a Music class, and a Debugging class. Section 7.5 presents instructions on how to add new classes to *Live Paper*.

When a page is placed on the tabletop, the associated Transport object adds all of its widgets to the IOManager class. Then each of its transparencies also adds its own widgets. The IOManager class maintains a list of all the widgets and active applications, and notifies the widgets when the user has selected them. Although the primary interface is via finger selection, the IOManager also allows the user of the system mouse to select a widget. If a page is moved, then the widgets automatically move on the next update cycle of *Live Paper* because the IOManager notifies them to redraw. The IOManager also permits communication between transparencies via the Inter-Application Communication Facilator (IACF) class.

The user interface has many common GUI elements, such as button, menus, and pictures. Each element is its own class, derived from an abstract base class called Widget. Interaction elements, which can be selected with a user's finger or the mouse, are derived from an IOWidget class (see Figure 7-8). Widgets are logically attached to a visual anchoring element on a page. When a page is moved, the widgets use the new location on the next update cycle.

The Display class is a drawing space that corresponds to the output display device. In *Live Paper*, there is a single derived class, ProjectorDisplay, which represents the display surface of the data projector unit. All widgets and any other output objects are rendered in the ProjectorDisplay object. The ProjectorDisplay also provides functions for turning on and off the data projector, and thus could also be classified as part of the I/O Software Interface layer.

The diagram in Figure 7-8 shows the class structure of the User Interface layer.

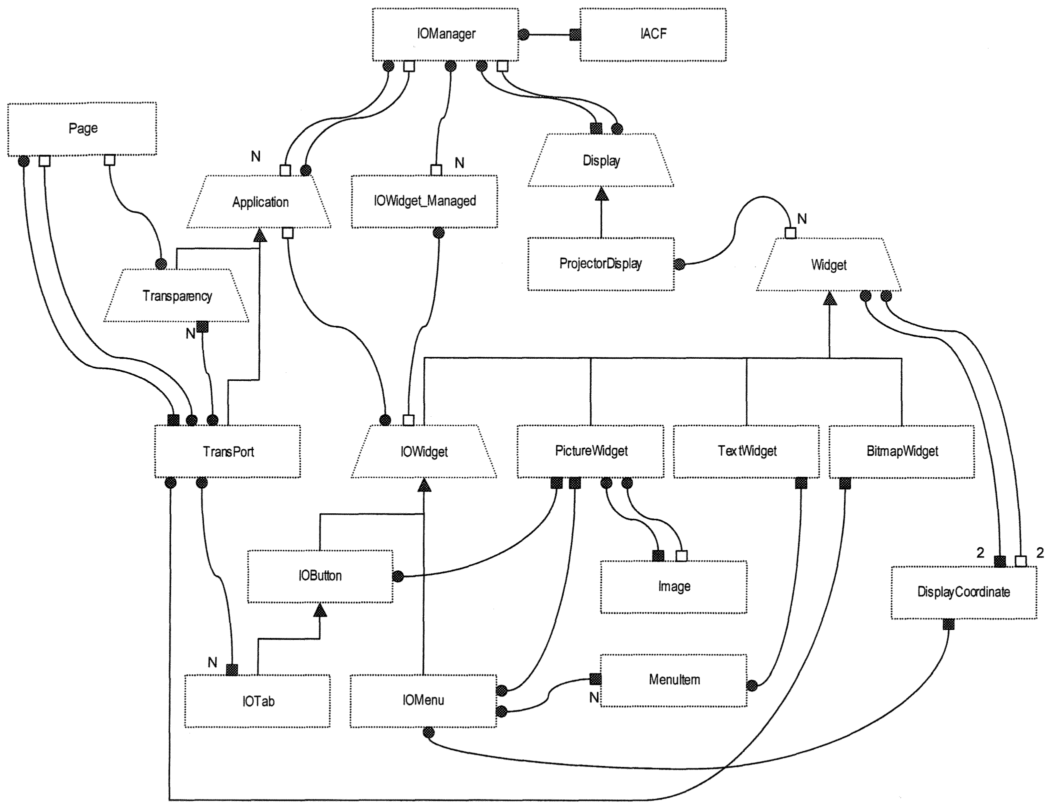


Figure 7-8: Class Diagram of the User Interface layer

Table 7-4 lists all of the major classes or groups of classes for the User Interface layer. The layer contains a total of 7719 lines of code.

Class Name	Purpose	Lines of Code
Transport	Stores the transparencies, and displays the transport tab bar interface element.	387
Transparencies (all)	Provide specific functionality that can be applied to a page.	2525
IOManager	Maintains a list of all applications and widgets, and ensures they are updated appropriately.	473
Widgets (all)	Provide display features, and some also provide input functionality.	2663
Display, ProjectorDisplay	Provide a canvas in which widgets are displayed.	523

Table 7-4: Significant classes for User Interface Layer

7.4 EVENT-ACTION PROCESSING

The basic mechanism of *Live Paper* is the update cycle. This consists of a series of fundamental steps. The first step is to capture the image of the tabletop. A check for motion is performed – if there is no significant motion, then the system can assume that the tabletop has not been disturbed since the last captured frame, and further image processing can be skipped. Image processing follows the outline of events in Chapter 4 and Chapter 5. Contours are found by first segmenting the tabletop image. An ImageAnalyzer object examines these contours to find whole pages. The Folder object then attempts to match these page candidates with stored active pages.

Unmatched active pages are then matched to overlapping and occluded pages on the desk based on their locations. Next, page candidates are matched to the remaining stored pages, either active or inactive, using the Hausdorff distance measure. If a page candidate cannot be matched, then it is treated as a new page, and added to the database. If a previously active page is not found in the new set of candidates, then it enters a deactivating state. It remains in this state for several cycles, and is removed if it is not found again. See Figure 5-27 for an illustration of the states.

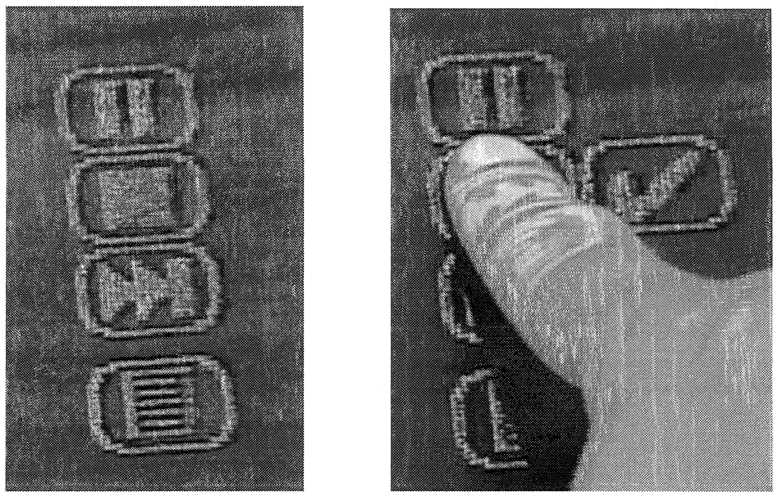
When a page is matched with its candidate, its position is updated. After the new list of pages is finalized, then the IOManager object receives an Idle call, which in turns triggers an Idle event to each of the active applications (transports and transparencies), as well as a check for the user's finger. If the user is selecting or selected a button, then that button receives the appropriate message. Finally, the screen is redrawn and the widgets are updated.

7.5 BUTTON SELECTION

As mentioned in section 3.3.1, the user has the ability to select projected buttons by using his or her finger. When the user touches the button, a confirmation check box appears to the immediate right. By moving the finger to the check box, the user confirms the selection and triggers the associated action. Moving the finger to another button, or away from the tabletop, lets the user cancel the action. The gesture does not require the user to physical tap on the desk (such as required by [Wellner 1993a]).

The image analysis is simple, and does not involve finger tracking (such as [Hardenberg 2001] and [Sato 2000]). When the system displays or moves a button, the corresponding location in the captured video image is sampled and stored. On each update cycle, the IOManager compares the stored sample with the current sample at that position. If the new sample is significantly brighter (at least 12 pixels increase in brightness by 15 grey levels), the button is potentially being selected. The detection algorithm takes advantage of the polarization filter on the camera, which eliminates most of the specular reflection. However, when the user places his or her finger on a button, the finger is strongly lit with a diffuse light.

On each update cycle, the IOManager compares samples at every projected button. When multiple buttons appear to be selected, then the button furthest away from the user is chosen as the candidate (see Figure 7-9 for an example). The IOManager then creates a confirmation check button and projects it to the right of the candidate. The IOManager waits for either the confirmation button to be selected, or for both the candidate and the confirmation button to be unselected.



(a) (b)

Figure 7-9: User selecting a button.

(a) Buttons displayed on the tabletop for the Music Player transparency. (b) The user selecting the stop button. Several other buttons have also been occluded by the finger, but because the stop button is further away from the user, it becomes the candidate button.

7.6 CREATING TRANSPARENCIES

Each transparency in the *Live Paper* system is a C++ class that is derived from a base *Transparency* class. There are a number of functions that the new class must implement, as well as several optional ones. This section briefly describes what must be written to create a new transparency, and in so doing, will give an indication of the difficulty of adding new transparencies.

There are four key functions that must be overridden (see Figure 7-10). The *Create* function is called when the transparency is first created – either when the system is started up and the page is generated, or when that transparency is added to an existing page. *Live Paper* calls the *Activate* function when the associated page appears on the tabletop. At this point, the transparency must register its user interface widgets with the IO Manager.

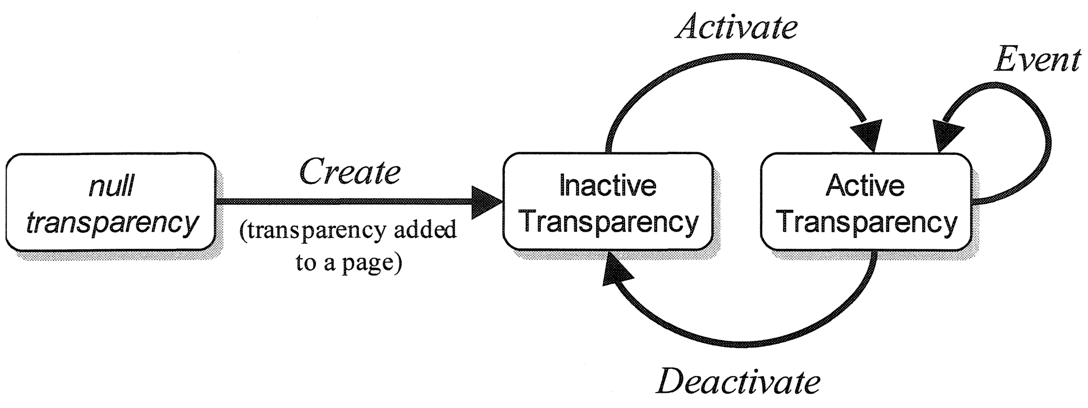


Figure 7-10: Relationship between transparency states and the functions called by *Live Paper*.

While a transparency is active, it will continually receive event messages from the IO Manager. There are two messages implemented in *Live Paper* – one for selection events, which occurs when the user selects a button, and one for update events, which occur on a regular basis. A transparency must implement the *Event* function in order to handle events related to moving the page or to user interactions.

The final function is a *Deactivate* function, which is called when a page is removed from the tabletop. The transparency must use this action to remove all widgets from the IO Manager and stop any activities in progress – for example, the music transparency will stop playing the current piece of music.

Two other functions that the transparency can implement are for saving and loading data. If a page is saved, then a *Save* call is made to each of its transparencies. Likewise, when the page is loaded with the *Load* function, each of its transparencies is created and then has an opportunity to load data from a file.

In addition to code, the new transparency should also have a unique icon that will be displayed on the transport bar. The name of the transparency is tied to a particular icon. Multiple icons can be used for animations by changing the name of the transparency slightly each time the IO Manager asks for its name. This would involve overriding the *GetTabName* function.

7.7 SUMMARY

The *Live Paper* system architecture provides the core functions to support a video augmented environment for a tabletop. The transparency mechanism is flexible and extensible, and able to add new task-oriented applications to the system. The framework lays a foundation for future development of a tabletop operating system.

Chapter 8

CONTRIBUTIONS

As mentioned in the introduction, *Live Paper* creates a basic zone of intelligence where ordinary paper pages gain illusionary computational abilities. *Live Paper* uses an environmental sensor to make paper an interface to the computer. It provides a user interface that matches magical interactions with physical interactions. New applications, in the form of transparencies, can be added to those already existing in the system. The technologies developed to find, identify, and track pages, support the illusion that the paper has built-in computational ability. *Live Paper* contributes to future development of generalized augmented environments where many objects gain illusionary intelligence.

This chapter presents the contributions of the research documented throughout this thesis. It also makes recommendations for future work on augmented environments.

8.1 CONCLUSIONS

The key contribution of the *Live Paper* system is to further develop the concept of paper-as-interface to a computer, first explored in the *DigitalDesk* project [Wellner 1993a].

However, the *Live Paper* system extends this concept to include individual pieces of paper as the focus of the interface – virtual annotations appear to be attached to the paper as it is moved around the desktop. The flexibility of the system is such that any arbitrary piece of paper can be used, and not just paper with explicit identifying codes.

Live Paper provides a framework for a Tabletop Operating System, as it demonstrates how task specific augmentations can be separated from system commands. With respect to the user interface, the transports act as bridges between the task-specific elements in the transparencies and the more magical interactive elements such as the tabs, buttons, and menus. The user is aware of the purpose of these elements, as they are obvious, accessible, and consistent. With respect to system architecture, the transparencies are general applications that use a consistent object-oriented interface to interact with the core framework of *Live Paper*. Each transparency provides unique augmentations that are focused on the single piece of paper to which the transparency is attached.

The concept of transparencies is extremely powerful. The transparencies are separate computational units that address specific tasks. Each transparency is dedicated to a single real object – a page in *Live Paper*. Each object, in turn, can have multiple transparencies. This articulation of virtual, portable computational units has not been expressed before in the field of augmented environments.

The implementation of finger detection is a robust and novel method to determine when the user has selected a button without explicitly tracking the hands. The user can employ any finger, or another object like a pen, to select the button, while the confirmation button eliminates false selections.

The problem of finding pages on the desk is not trivial; the presented research shows that the methods in *Live Paper* for locating and tracking individual pages are reliable. Separating overlapping and occluded pages is more difficult, but the presented methods work well.

Page identification using the Hausdorff measure also works well, especially on small sets of reasonably distinct pages. However, as the number of stored pages increases, there is an increase in the likelihood that a new page will be similar to an existing page. Feedback from the projected image can also cause some false identifications. Some techniques to reduce false matches have been investigated in this project, and have been shown to produce positive results.

The registration algorithms in *Live Paper* are sufficiently accurate to augment pages with projected annotations, and to detect button selection.

The implementation of *Live Paper* successfully demonstrates what can be accomplished with the hardware and proposed software architecture. The system provides a unique means to enhance ordinary paper documents with virtual information and other interactive features without requiring the user to consult a separate display device.

8.2 Recommendations

The potential for an enhanced *Live Paper* system is enormous. This thesis lays the groundwork for future research and development.

8.2.1 General

Live Paper should move beyond individual pages to collections of pages, both in loosely coupled forms such as printed documents and in book forms. Pages exist in relation to other pages, and the system can take advantage of this. A simple example would be the transport that appears next to a page. In a document, a similar bar would appear, but subtly different in look; each individual page might not have its own transport. Moreover, a common set of transparencies could be added to the entire document at a time.

The system shows the strength of changing the focus of augmentation from the desktop itself to pages on the desktop. Extending the enhancements to objects other than paper would reveal additional research problems. A key set of objects for the desktop is pens. For example, a system that dynamically stored writing, and could associate that writing with its writer, would be able to provide smart mark-up and note-taking features. Being able to use a pen to project a pointer would be useful in remote collaboration.

There should be further development to increase the speed and accuracy of the system. Preliminary research would investigate the necessary response times of core activities, such as responding to button selection and tracking moving pages. The response time is directly related to the frame rate of capturing video images. For example, if the system must respond to a button selection within 1/6th of a second, then the frame rate must be at least 12 frames a second to ensure that the user has stopped moving his or her hand.

The system should use multiple cameras to watch the desktop (such as in [Rekimoto 1999]). One camera would have a fixed viewing angle that includes the entire desktop. One or more pan-tilt-zoom cameras would work independently to allow enlargements of arbitrary locations on the desk. A pan-and-scan technique to increase the resolution of captured pages would then be feasible, as the system would know whether or not a page has moved from the main camera. Using a single pan-tilt-zoom camera is not a realistic option for a fixed-environment augmentation system. *Live Paper* could not take advantage of the mobility of the camera without ignoring the entire tabletop for a significant period of time (about 10 seconds).

Finally, the variety of input/output widgets should be increased. Desktop operating systems, such as Microsoft's Windows or Apple's Mac OS X, provide many standard user interface elements, as well as provide mechanisms for programmers to create their own elements. General guidelines for their use should be created.

8.2.2 System Design

The system architecture should be split into separate stand-alone processes that would communicate via network services. They could then be hosted on separate computers. For example, one process would connect directly to the video cameras and perform image processing. A second process would maintain a software model of the physical elements on the desktop. The third process would be responsible for human-object interaction, including projecting information via the overhead projector. This solution provides flexible availability of supplemental computing power.

Similarly, development should add threading to the system software, whether or not multiple processes were created. This would allow the system to fully exploit the power of multiprocessor systems.

Another useful addition would be an infrared illuminator positioned so as to light objects over the desktop, and an infrared camera to image these objects (such as investigated in [Sato 2000]). Two cameras would be sufficient for tracking the user's arms and hands, allowing for 3D gestures and pointing.

In *Live Paper*, adding a new kind of transparency required a recompile of the entire system. Adding new transparency types should not require this. They should be created using a scripting language or compiled into dynamic link libraries. This would require further articulation of a standard programming interface within *Live Paper*.

The transparency applications should be further developed. The Architectural Renderer is very compelling, and could gain new features such as head tracking. The perspective projection of the walls could then update to match the centre of vision. The walls could be painted solidly, and there could be an option to 'see-through' the front-most wall to objects inside.

8.2.3 Research Topics

Future development of the algorithms should continue to refine and optimize the current set of algorithms. If multiple cameras are added to the system, then the algorithms could be extended to this scenario as appropriate.

Experiments should be conducted where real objects such as mugs, staplers, and pens are used to occlude paper. The experiments presented in this thesis did not attempt to address the performance of the algorithms with real objects in a consistent repeatable way. In informal usage of *Live Paper* with real objects, the algorithms did perform adequately.

With respect to finding pages, using sub-pixel estimation might increase the accuracy of determining the location. Due to aliasing effects, page corners are often indistinct, and thus might not appear in extracted page boundaries. Future research could address this issue.

An investigation into the trade-offs between documents without fiducials and ones with fiducials might be warranted. Although determining what page is present without fiducials is critical for a general purpose VAE desktop, documents and other objects with fiducials do allow for certain applications. For example, if a user sees a business card with a fiducial, then he or she knows that the card is already enhanced. A simple to use application to generate documents with enhancements would be a good development.

Live Paper does not automatically correct for mis-registered projections. There could be development of a control loop mechanism that uses the projected augmentations, or special projected fiducials, to refine the page location as well as the registration between camera, desk, and projector.

With respect to page identification, the Hausdorff distance calculation could be extended in several ways. Experiments showed that increasing the resolution of the capture page image would result in an increase in the accuracy of the Hausdorff calculation. Thus there could be research into an hierarchical Hausdorff algorithm that automatically determines what image resolution to use in comparisons. The algorithm would start with a low-resolution image, and increase as necessary. There would an interesting trade-off between accuracy and computational efficiency.

A second extension would be to modify the Hausdorff calculation for use with overlapping and occluded pages. In the case of overlapping pages, the system would not know which page was on top, and so various combinations of partial pages might have to be compared. The investigation would have to determine when pages could be reliably matched against stored pages.

A third extension for the Hausdorff calculation would be to integrate the match ranking into the *Live Paper* user interface, so that the user could tell the system when it had the wrong match. The interface would have to allow the user to quickly switch to the next possible match, and provide good feedback as to what page is currently shown.

The registration used in *Live Paper* works well, but takes time to set up. Future research could investigate techniques for self-registration of the system. The registration between projector and camera could be fully automatic. Registration between camera, desk, and projector would require a target to determine real-world distances. However, this could be made very easy for the user.

8.3 FINAL REMARKS

The *Live Paper* system successfully demonstrates a unique, working, augmented environment where ordinary paper becomes the interface to the computer. The system fulfills the objective of creating a video augmented tabletop system that uses image processing to find, identify, and track paper. *Live Paper* provides the user with a consistent projected interface. This thesis lays a foundation for future research and development of a general purpose augmented tabletop.

REFERENCES

[Alquier 1996] Alquier, L. and Montesinos, P., "Segmentation of lines using perceptual organization with active contour functions," Proceedings of the 13th International Conference on Pattern Recognition, Vol. 1, August 1996, pp. 25-29.

[Anoto 2003] Anoto Inc. "The Digital Pen", Web site, 2003. "http://www.anoto.com/?url=/technology/pen/"

[Arai 1995] Arai, T., Machii, K., Kuzunuki, S., and Shojima, H., "InteractiveDESK: A Computer-augmented Desk Which Responds to Operations on Real Objects", *Proceedings of CHI '95*, ACM, 1995, pp. 141-142.

[Arai 1997] Arai, T., Aust, D., and Hudson, S., "PaperLink: A Technique for Hyperlinking from Real Paper to Electronic Content", Proceedings of CHI 97, ACM, 1997, pp. 327-334.

[Ashdown 2003] Ashdown, M., and Robinson, P., "The Escritoire: A Personal Projected Display", Proceedings of the 11th International

Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2003), February 2003, pp. 33-40.

[Baudisch 2001] Baudisch, P., Good, N., and Stewart, P., “Focus Plus Context Screens: Combining Display Technology with Visualization Techniques”, Proceedings of UIST’01, ACM, 2001, pp. 31-40.

[Bérard 2003] Bérard, F., “The Magic Table: Computer-Vision Based Augmentation of a Whiteboard for Creative Meetings”, PROCAM workshop of IEEE International Conference in Computer Vision, IEEE, October 2003.

[Black 1998] Black, M., Bérard, F., Jepson, A., Newman, W., Saund, E., Socher, G., and Taylor, M., “The Digital Office: Overview”, AAAI Spring Symposium on Intelligent Environments, March 1998, pp. 1 - 6.

[Bluetooth 2003] Bluetooth SIG, “The Official Bluetooth Membership Site”, Web site, 2003.
“http://www.bluetooth.org/”

[Casey 1992] Casey, R., Ferguson, D., Mohiuddin, K., and Walach, E., “Intelligent Form Processing”, Machine Vision and Applications, Vol. 5, 1992, pp. 143-155.

[Cheng 1998] Cheng, L-T, and Robinson, J., “MCLGallery: A Framework for Multimedia Communications Research”, CCECE ’98 Proceedings, IEEE, 1998, pp. 413-416.

[Crowley 2000] Crowley, J., Coutaz, J., and Bérard, F., “Things That See”, Communications of the ACM, Vol. 43, No. 3, March 2000, pp. 54-64.

[Doerman 1997] Doermann, D., Li, H., and Kia, O., “The Detection of Duplicates in Document Image Databases”, Fourth International Conference on Document Analysis and Recognition, 1997, pp. 314-318.

[Dubuisson 1994] Dubuisson, M-P, and Jain, A., “A Modified Hausdorff Distance for Object Matching”, Twelfth IAPR International Conference on Pattern Recognition, IEEE, 1994, pp. 566-588.

[Duda 1973] Duda, R. and Hart, P., *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

[Dymetman 1998] Dymetman, M. and Copperman, M. "Intelligent Paper", Electronic Publishing, Artistic Imaging, and Digital Typography, R. Hersch, J. Andr, H. Brown, eds., Springer-Verlag, April 1998, pp. 392-406.

[E Ink 2004] E Ink Corporation, Web site, 2004. ”http://www.eink.com/”

[Feldman 1997] Feldman, J., "Regularity-based Perceptual Grouping", Computational Intelligence", Vol. 13 Num. 4, 1997, pp. 582-623.

[Freeman 1977] Freeman, H. and Davis, L., “A Corner-Finding Algorithm for Chain-Coded Curves”, IEEE Transactions on Computers, Vol. C-26, March 1977, pp. 297-303.

[Hardenberg 2001] Hardenberg, C., and Bérard, F., “Bare-Hand Human Computer Interaction”, Proceedings of the ACM Workshop on Perceptive User Interfaces, ACM, 2001.

[Hull 1997] Hull, J. and Cullen, J., “Document Image Similarity and Equivalence Detection”, Fourth International Conference on Document Analysis and Recognition, IEEE, 1997, pp. 308-312.

[Huttenlocher 1993] Huttenlocher, D., Klanderman, G., and Rucklidge, W., “Comparing Images Using the Hausdorff Distance”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 9, IEEE, 1993, pp. 850-863.

[Huttenlocher 1999] Huttenlocher, D., Lilien, R., and Olson, C., “View-Based Recognition Using an Eigenspace Approximation to the Hausdorff Measure”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 9, 1999, pp. 951-955.

[Ishii 1992] Ishii, H. and Kobayashi, M., "ClearBoard: A Seamless Media for Shared Drawing and Conversation with Eye-Contact", Proceedings of Conference on Human Factors in Computing Systems (CHI'92), ACM, 1992, pp. 525-532.

[Ishii 1994] Ishii, H., Kobayashi, M., and Arita, K., "Iterative Design of Seamless Collaboration Media", Communications of the ACM, Vol. 37, No. 8, August 1994, pp. 83-97.

[Jain 1988] Jain, A., *Fundamentals of Digital Image Processing*, Prentice Hall, 1988.

[Johnson 1993] Johnson, W., Jellinek, H., Klotz, L., Rao, R., and Card, S., "Bridging the Paper and Electronic Worlds: The Paper User Interface", InterCHI '93, 1993, pp. 507-512.

[Kobayashi 1998] Kobayashi, M. and Koike, H., "EnhancedDesk: Integrating Paper Documents and Digital Documents", Asia Pacific Computer Human Interaction 1998, Japan, IEEE, 1998.

[Lai 2002] Lai, J., Levas, A., Chous, P., Pinhanez, C., and Viveros, M., "BlueSpace: Personalizing Workspace through Awareness and Adaptability", International Journal of Human Computer Studies, Vol. 57, No. 5, Elsevier, 2002, pp.415-428.

[Logitech 2003] Logitech Inc., “The io™ Personal Digital Pen”, Web site, 2003. “http://www.logitech.com/index.cfm?page=products/details&contentid=6150”

[Mackay 1995] Mackay, W., Pagani, D., Faber, L., Inwood, B., Launianinen, P., Brenta, L., and Pouzol, V., “Ariel: Augmenting Paper Engineering Drawings”, Video, Proceedings of CHI ’95, ACM, 1995.

[Mohr 1996] Mohr, R. and Triggs, B. “Projective Geometry for Image Analysis”, Tutorial at ISPRS, Vienna, 1996.

[Myers 1998] Myers, B., “A Brief History of Human Computer Interaction Technology”. ACM Interactions, March 1998, pp. 44-54.

[Newman 1992] Newman, W. and Wellner, P., “A Desk Supporting Computer-based Interaction with Paper Documents”, Proceedings of Conference on Human Factors in Computing Systems (CHI ’92), ACM, 1992, pp. 587-592.

[O’Hara 1997] O’Hara, K., and Sellen, A., “A Comparison of Reading Paper and On-Line Documents”, Proceedings of CHI ’97, ACM, 1997, pp. 335 – 342.

[OTM 2003] OTM Technology., Vpen™, Web site, 2003. “http://www.otmtech.com/vpen.asp”

[Otsu 1979] Otsu, N., "A threshold selection method from gray-level histograms," Transactions of Systems, Man, and Cybernetics, Vol. 9, 1979, pp. 62-69.

[PARC 2002] Palo Alto Research Center Incorporated, "DataGlyphs®: Embedding Digital Data," Web site, 2002. "http://www.parc.com/solutions/dataglyphs/"

[Parker 1997] Parker, J., Algorithms for Image Processing and Computer Vision, John Wiley & Sons, 1997.

[Pinhanez 2001] Pinhanez, C., "Using a Steerable Projector and a Camera to Transform Surfaces into Interactive Displays", Proceedings of CHI'01, ACM, 2001, pp. 369-370.

[Pingali 2003] Pingali, G., Pinhanez, C., Levas, A., Kjeldsen, R., Podlaseck, M., Chen, H., and Sukaviriya, N., "Steerable Interfaces for Pervasive Computing Spaces", Proceedings of IEEE International Conference on Pervasive Computing and Communications (PerCom'03), IEEE, 2003, pp. 315-322.

[Rekimoto 1998] Rekimoto, J., "Matrix: A Realtime Object Identification and Registration Method for Augmented Reality," *Asia Pacific Computer Human Interaction* (APCHI '98), July 1998

[Rekimoto 1999] Rekimoto, J. and Saitoh, M., "Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments", Proceedings of CHI'99, 1999, pp. 378 – 385.

[Rekimoto 2000] Rekimoto, J. and Ayatsuka, Y., "CyberCode: Designing Augmented Reality Environments with Visual Tags," *Designing Augmented Reality Environments* (DARE 2000), 2000.

[Rekimoto 2002] Rekimoto, J., "SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces", Proceedings of CHI 2002, pp. 113 – 120.

[Robertson 1997] Robertson, C., "Telewriting in a Video Augmented Environment", Engineering 9888 Report, Memorial University of Newfoundland, April 18, 1997.

[Robinson 1997] Robinson, P., "The Origami Project", Proceedings of EUROGRAPHICS '97, Vol. 16, No. 3.

[Sato 2000] Sato, Y., Kobayashi, Y., and Koike, H. "Fast Tracking of Hands and Fingertips in Infrared Images for Augmented Desk Interface," Fourth International Conference on Automatic Face- and Gesture-Recognition, Grenoble, France, March 2000.

[Saund 1998] Saund, E., "Image Mosaicing and a Diagrammatic User Interface for an Office Whiteboard Scanner". (unpublished paper) "http://www.parc.xerox.com/spl/members/saund/papers/zombieboard98.ps.Z"

[Saund 1999] Saund, E., "Perceptual Organization of Occluding Contours of Opaque Surfaces", Computer Vision and Image Understanding, Vol. 76. No. 1, October 1999, pp. 70-82.

[Shi 2003] Shi, Y., Xie, W., Xu, G., Shi, R., Chen, E., Mao, Y., and Liu, F., "The Smart Classroom: Merging Technologies for Seamless Tele-education", IEEE Pervasive Computing, Vol. 2, No. 2, IEEE, April-June 2003, pp. 47-55.

[Shreiner 1999] Shreiner, D., "OpenGL® Reference Manual (3rd Edition)", Addison-Wesley, 1999.

[Smith 1987] Smith, R., "Experiences with the Alternate Reality Kit: An Example of the Tension between Literalism and Magic," Proc of Human Factors in Computing Systems and Graphics Interface, CHI+GI 1987, Toronto, Canada, April 5-9, 1987, pp 61-67.

[Spielberg 2002] Spielberg, S. (director) "The Minority Report," 20th Century Fox, Released June 17, 2002.

[Stafford-Fraser 1996a]

Stafford-Fraser, Q. and Robinson, P., "BrightBoard: A Video-

Augmented Environment", Proceedings of CHI '96, ACM, 1996, pp. 134-141.

[Stafford-Fraser 1996b]
Stafford-Fraser, Q., "Video-Augmented Environments," Doctoral Dissertation, University of Cambridge, 1996.

[Takao 2003] Takao, N., Shi, J., and Baker, S., "Tele-Graffiti: A Camera-Projector Based Remote Sketching System with Hand-Based User Interface and Automatic Session Summarization", International Journal of Computer Vision, Vol. 53, No. 2, July 2003, pp 115-133.

[Taylor 1998] Taylor, S., Dance, C., Newman, W., Taylor, A., Aldhous, T., and Taylor, M., Augmenting Paper: Using a Video Camera to Support Selective Scanning from Paper to Screen, Xerox Ltd., Technical Report EPC-1998-105.

[Tegrity 1998] Tegrity Inc., Interactive Presentation Station, Web site, 1998. "http://www.tegrity.com/"

[Teh 1989] Teh, C. and Chin, R., "On the Detection of Dominant Points on Digital Curves", IEEE Trans. Pattern Analysis and Machine Intelligence, v. 11, no. 8, August 1989.

[Therrien 1989] Therrien, C., *Decision, estimation and classification*, John Wiley & Sons, New York, 1989.

[Ullmer 1997] Ullmer, B. and Ishii, H. "The metaDESK: Models and Prototypes for Tangible User Interfaces", Proceedings of UIST '97, ACM, October 1997, pp. 223-232.

[Ullmer 1998] Ullmer, B., Ishii, H., and Glas, D. "mediaBlocks: Physical Containers, Transports, and Controls for Online Media", Computer Graphics Proceedings / SIGGRAPH '98, ACM, July, 1998, pp. 379-386.

[Underkoffler 1998] Underkoffler, J., and Ishii, H. "Illuminating Light: An Optical Design Tool with a Luminous-Tangible Interface", Proceedings of CHI '98, April, 1998, pp. 542-549.

[VideoBrush 1997] VideoBrush Corporation, VideoBrush Whiteboard, web site, 1997. "http://www.videobrush.com/whiteboard/info.html"

[Weiser 1991] Weiser, M., "The computer for the 21st century", Scientific American, Sept. 1991, v. 265, pp. 94-104.

[Wellner 1993a] Wellner, P., "Interacting with Paper on the DigitalDesk", Communications of the ACM, Vol. 36 No.7, July 1993, pp. 86-96.

[Wellner 1993b] Wellner, P., “Adaptive Thresholding for the DigitalDesk,” Rank Xerox Ltd., Technical Report EPC-1993-110.

[Wolberg 1990] Wolberg, G. Digital Image Warping, IEEE Computer Society Press, Los Alamitos, CA, 1990.