The LivePaper System: Augmenting Paper on an Enhanced Tabletop

John Robinson
Department of Electronics
University of York
Heslington, York, United Kingdom
jar11@ohm.york.ac.uk

Charles Robertson
Faculty of Engineering and Applied Science
Memorial University of Newfoundland
St. John's, Newfoundland, Canada
charlesr@engr.mun.ca

Abstract

The LivePaper system augments ordinary pieces of paper with projected information. Pages, cards and books are placed on an instrumented tabletop to activate their enhancement. To the user, it appears as if the paper gains new visual and auditory features. Projected annotations track the orientation and location of pages as the user moves them on the desktop. A piece of paper that is removed, but then returned to the desk, regains the same features that it previously exhibited. The LivePaper system accomplishes this by using features extracted from written material on the page, not from glyphs or other artificial marks.

The paper describes both the system as a whole, and a number of sample applications we have developed to illustrate the feasibility of the LivePaper system. These applications include an architectural visualization tool, which projects a 3D hidden-line rendering of walls onto a page. The user may rotate and move the page to view the rendering from different angles. Another application is an audio player, which begins playing when a page (such as a business card) is laid on the desk. The user may control playback with his or her finger via projected buttons. Other applications include page-sharing, remote collaboration, and World Wide Web page viewing. From the user's perspective, all of these applications are attributes of the particular page, not features of the tabletop.

Particular attention is given to the design of interaction: LivePaper is object-oriented, because the individual sheets are treated as computational units, but it also provides functions that involve several objects. The design principles applied to handle the different kinds of functionality are explained and illustrated in the LivePaper system, but are also proposed for wider use in augmented reality.

Keywords

Video augmented environment, enhanced tabletop, human-computer interface

1. Introduction

The LivePaper system is a video-augmented environment that enhances ordinary pieces of paper with projected information. Figure 1 shows the system in use, with various applications bound to paper cards and pages. Later sections outline the methods by which the system analyses images of paper on a table top, interprets them, adds functionality to them, and provides novel applications, but our primary concern is to show how LivePaper embodies user-interface principles that we believe are widely applicable in augmented reality. Briefly these are the identification of functionality as *object-oriented*, *multi-object* or *environmental*; the division of object-oriented functionality into *literal* and *magical* parts with different implications for information presentation and interaction; the provision of replicated control for *multi-object* functionality; and the creation of *virtual environment objects.* Section 2 explains how these terms and principles arise from a consideration of ubiquitous computing, augmented reality on wearable computers, and video-augmented environments. Section 3 describes how they are realized in the LivePaper system. Section 4 gives a brief technical overview and Section 5 summarizes the implications of LivePaper and the proposed user interface principles.

2. Interacting with Augmented Objects and Environments

2.1 Classes of augmentation functionality in ubiquitous computing, wearable augmented reality and video augmentation

The vision of ubiquitous computing (ubicomp) [28] is to enhance everyday objects such as pens, paper, mugs, chairs and walls, with computational capabilities. People interact with embedded processors by handling the objects as they normally do, and new functionality is encountered as augmentation of conventional functionality. Some simple manifestations of ubicomp, such as self-monitoring coffee cups (which squawk when in need of cleaning) and responsive drapes (that open and close subject to both lighting change and voice command), augment single independent objects. But the real promise of ubiquitous computing is networked objects that exchange information to provide an integrated intelligent environment. This presents a user-interface challenge. So long as people interact with particular objects, ubicomp offers a natural, object-oriented, user interface. This, indeed, is one of its attractions. But when communicating objects generate information that transcends particularity, it is difficult to say

how and where that information should be presented to the user. Similarly, how should the user control functions that integrate capabilities from several objects?

In wearable computing [23], interaction is usually subject-centric. Although a typical general-purpose wearable provides the user with simultaneous access to dynamic computed information and to the real world – for example, by presenting the first on a see-through head-mounted display – there is often no visual or conceptual connection between the two domains. Email and web browsing, for example, run on a wearable with no reference to the user's real environment. Reinforcing the subjective orientation, displayed information moves with the user's head, and input is by a special-purpose hand-held device. So the real and computed worlds experienced by the user are superimposed but isolated. Researchers in wearable computing have long been interested in bridging the two worlds (e.g. [6]). Their augmented-reality applications overlay computed visual information on a real scene, generated by a computer that is aware of the environment and the view the user has of it. Now it becomes possible to envisage object-oriented interaction with a wearable. If true augmented reality is achieved – with overlays sticking to surfaces in the real scene, even as they are viewed dynamically from different viewpoints – then object-centric computation and interaction can be simulated. A user may grab an object, manipulate it, and the overlays will not only follow, but also change in line with the user's actions. Such real-time, registered, augmentation would give a very strong illusion that the object itself is the computational unit. We have conducted some investigations into such object-oriented interaction for wearables [4], though the practical difficulties of registration have limited the applications. Indeed, the problem of maintaining lock between overlays and reality is a very hard one [3] especially if the user is allowed to move freely. When and where registration is good enough to support robust augmented reality, wearable computing can simulate ubiquitous computing. It therefore presents a new version of the user interface problem already mentioned. If we can augment objects so that manipulating them individually provides a natural focus for computer-user communication, how do we deal with functions that transcend individual objects?

In this paper, we concentrate on a different kind of augmented reality, a video-augmented environment (VAE), where one or more data projectors augment a scene that is monitored by one or more video cameras. VAEs may be thought of as enhancing a fixed workspace, but in our research we have concentrated on maintaining a strong association between augmentations and individual objects. We augment paper, which, in everyday use, is scattered and stacked on untidy surfaces. So object-orientation is natural: we have to work at the level of the sheet, card or book, not at the level of the augmentation surface [17]. Yet we encounter the same problem as in ubicomp and wearables – some functionality is particular to individual objects, some functionality is integrative. How do we provide a user interface that is both object-oriented and capable of multi-object and environmental interaction? Further, how do we structure the object-oriented augmentations to give the right cues to users about what can be done?

For all three cases we have discussed – real objects with embedded processors, wearable augmented reality simulating enhanced objects, and VAEs projecting augmentations onto objects – we identify the following loci of augmentation:

- *Object-oriented.* When functionality is strongly associated with a particular object, even if there are many objects with the same functionality, we say it is object-oriented. For example, the ability to share a sheet of paper by transmitting its contents to a remote location and having remote annotations projected onto it, is object-oriented. Other sheets may or may not be similarly shared. There is no necessary linkage between objects offering the common functionality. Clearly it makes sense to associate the user interface for this type of functionality with the object in question.
- *Multi-object.* Functionality that involves interaction between a plurality of objects is multi-object. A special case of this is when only one kind of augmentation can apply at once. For example, in the LivePaper system, audio augmentation is via a single speaker so it is not desirable to augment more than one piece of paper with audio at a time. There is contention between objects that are attempting simultaneously audio augmentation, and implicit communication between them is required to resolve which one speaks. Multi-object functionality applies generally to information exchange between collaborating objects in an augmented environment.
- *Environmental*. Functionality that applies to the whole of the augmented area is called environmental. In the case of ubicomp, environment denotes integrative functionality that applies to a whole room, while on a wearable, any operation that is context sensitive, but not associated with particular objects, would be environmental.

Because projected information and displays on wearables do not always enhance real objects or the real environment and yet are still sometimes called augmented reality, we add the term *Unbound* to denote functionality that has no physical anchor and *Subject-centric* to denote functionality that is private to the wearer of a computer.

2.2 Literal and Magical Subclasses of Object-oriented Functionality

Object-oriented functionality is tightly bound to particular objects, and we have stated the implication that user interaction should likewise be object-oriented. A characteristic property of individual objects is their physical extent. This not only constrains the arrangement of visible enhancements, it also provides a very strong organizational framework for the user interface. Here we explain how we recognize this framework, and use it in LivePaper, based on insights first suggested by Smith [21].

The idea that interfaces embody a tension between the *literal* and the *magical* was first suggested and studied in the context of the *Alternate Reality Kit*. *Literal* features exploit a metaphor such as direct manipulation to define the modes of use of an object, and they are therefore easy to learn. *Magical* features, like pop-up menus, have little to do

with the way real-life objects work, so are harder to learn but provide the user with more power. Smith plotted functionality against learning time for a variety of interactors in a conventional graphical user interface, showing the opposing influences of literalism and magic.

We argue that in a mixed reality the physical objects themselves provide anchors for literalism. Augmentations that that are bound to these anchors are interpreted literally. Other augmentations, associated with, but not bound to, the physical objects, can provide *magical* features of the interface. To explain this more clearly, consider the types of operation a user could do with Live Paper on an enhanced table top:

The user can place paper at will, position it, stack it, and even fold it or crush it (though our implementation does not yet handle these later cases). The user can make gestures, either over the paper or beside it. The user can make marks (for example, with ink), either on the paper or beside it. We note that paper is naturally manipulated by placing it and marking it; people do not normally gesture over paper. Conversely, people do not make markings adjacent to paper on a desktop --- such annotations add nothing to the paper once it is moved; they stay behind as dereferenced graffiti on the desk. Therefore, the paper itself, its markings and the augmentations projected onto it, are seen as comprising a physical, tangible unit. The place for *magic* is not on this literal field, but outside it. Directly adjacent to a sheet of paper is ideal --- users do not do *literal* interaction there, but the spatial association is obvious.

There are important design consequences of this reasoning. All live paper applications involve *literal* interaction on the paper. Thus, annotations of printed information are registered to the page and overlay it. As the page is moved, these annotations translate and rotate with it, so they remain locked in place. Markings that the user makes on the paper may result in responses by the system, but these will be changes in the representation of the object, not control responses. For example, writing on the paper may cause the vectorization of markings for transmission to a remote graphical shared space, but it will not cause a *magical* change in functionality. By contrast, each live page has a *transport* which is projected adjacent to the page, which tracks its position but is always oriented vertically. The spatial association of the transport and the paper is clear, but the fixed format and orientation signals to the user that the transport is not part of the literalism of the LivePaper interface. On the transport we provide controls, over which the user gestures (see below). The examples of applications given later further illustrate the clear separation of user interface operations. Perhaps the most compelling is the architectural application of section 3.3, where the augmentations on the paper change as the page is turned, but in accordance with a literal/physical interpretation of what is on the paper.

The separation of literal and magical functionality that we propose and implement in the LivePaper system is a direct consequence of treating pieces of augmented paper as units of interaction on a surface that is itself augmented. Earlier systems have freely mixed magical functionality with literalism. For example the Brightboard ([22], see also section 3.1) encourages the objects of interest --- boards full of information --- to be impregnated with *magical*

symbols. We suggest that our clear separation of literal objects and magical adjuncts gives a much clearer model of what can be done in a mixed reality. The same idea can be extended, for example, to a wearable augmented reality system, where the detection of known objects in the scene allows for their literal augmentation, with graphics that lock into place even as the user moves in space. At the same time they can be magically augmented with controls that may move with the user, rather than with the object.

2.3 Magical Interaction for LivePaper

Given the separation between literal and magical interaction, the LivePaper system must recognize changes in markings on paper, and hand gestures adjacent to paper. Later sections of this article outline what is involved in tracking marking changes, and in detecting the user's finger against paper and the table top. Here we briefly describe our design of the *magical* interaction mechanism.

The visual anchor for the magical interaction is the vertical *Transparency Portal*, or *transport*, at the right side of each page (see Figure 2 and later examples). Each transport has a series of tabs, where each tab represents a *transparency*. A transparency is a bundle of functionality that can be added to the page by its original creator or the user. As more transparencies are added to the page, its transport grows larger to accommodate all of the tabs.

By pointing at a tab, the user can view the particular user interface for that transparency, typically a stacked set of control buttons that 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.

The transport user interface is therefore a kind of cascading pop-up menu. However, because the user has no way of indicating a button press (we do not, for example, detect tapping of the table top), the final branch of the menu is a confirm button. Control is therefore purely by continuous 2D motion.

The transport automatically stacks transparency tabs in a column on the tabletop. This magical organization of the access to functionality contrasts with the way transparencies overlay their literal functionality on the paper itself: they superimpose, just like transparent sheets laid on top of the paper (hence the name). There is the potential for confusion between overlaid transparencies, but we have never encountered this in practice.

2.4 Integrating the Interface

The separation into literal and magical interface components can be extended to multi-object functionality. Clearly the control of such functionality is magical, but integrative applications may yield literal information, which has to

be presented somewhere. For example, consider an application that looks at two open diaries and projects information about timeslots that are open on both.

We have implemented control of multi-object functionality through replicated tabs. Objects that interact have characteristic communication tabs in their transports. Manipulation of the controls of one object can result in simultaneous projection of menus on both objects. Figure 3 shows two documents that are linked for collaborative drawing or for unblocked brainstorming [12]. (The projections are shown in reverse video for clarity.) Obviously these two documents could be located anywhere within the augmented area. They could also be on remote systems. Because the control of integrative functionality is replicated on both transports, the functionality is operated identically whether the sharing is local or remote.

We implemented two kinds of interface for environmental functionality. The first, designating an area of the augmented surface to global functions, corresponds to the introduction of a *virtual environmental control object*. This gives a clear spatial cue for the organization of control, but it takes up real estate. The alternative is massive replication of controls through all objects' transports. This alternative has been used for our one global application – debugging, illustrated in figure 4.

3. The LivePaper Implementation

3.1 Previous Work on Video Augmented Environments

Prior work in video augmentation has generally focused on creating a workspace with enhanced abilities rather than simulating embedded intelligence in the objects. Several projects since the early 1990's have used video and other sensors, coupled with data projectors, to create a Video Augmented Environment (VAE) based on a desktop model.

The *DigitalDesk* [13][29] was a pioneering attempt to add general computer interactivity to an ordinary desktop, and has become an inspiration for many other video augmented desk systems. The system receives information from two cameras mounted over the desk. Information is displayed on the desk via a data projector. Direct successors of the *DigitalDesk* include the *LightWorks* project, a video-scanning system for printed documents [2], *Origami* [19] and the *EnhancedDesk* [11]. Both *Origami* and the *EnhancedDesk* locate and identify pages using fiducials.

The *metaDESK* [25] is an investigation into using physical objects to interact with a computer display. In addition to the back-projected display desk, the *metaDESK* also includes two movable 'lenses' that display information about the underlying area. The *metaDESK* uses buildings as physical icons, or phicons to rotate and set the scale of maps. The *mediaBlocks* project [26] uses blocks with some digital storage capabilities to store handles to media such as video, pictures, and audio. 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. *Illuminating Light* [27] focuses on teaching and simulating

holography. Students manipulate physical objects that represent common holography devices. A computer analyzes the scene for coloured dots acting as fiducials. These identify the type and orientation of each object. The system then projects the path of the laser beam, which interacts appropriately with the objects. Although the shape of each object suggests a function, the object itself has no intrinsic utility outside of the zone of augmentation.

The *BrightBoard* [22] 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. Areas of the board can be saved, printed, faxed, or emailed; the commands are written directly on the whiteboard. 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 ZombieBoard [2] [20] is a whiteboard scanner. The system captures high-resolution images of the whiteboard, analyzes gestures, and provides a Diagrammatic User Interface (DUI). To obtain high-resolution images, the *ZombieBoard* mosaics low-resolutions images captured from a pan/tilt NTSC video camera. 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.

There have been several research projects that look at augmenting the capabilities of writing instruments such as pens, markers, and highlighters. Additionally, novel methods of organizing and registering overlay augmentations have been applied to various common objects. Important influences on LivePaper were references [1],[15],[24].

3.2 LivePaper System Overview

The LivePaper VAE has been implemented in a practical tabletop system. The user sits in front of the table. A camera positioned overhead has a generally unobstructed view of the table top. A computer digitizes the video signal and performs image analysis to determine the location, size, and content of the pages. It then feeds output to a data projector to augment the paper.

The desk is 30"x60" (1518mm x 755mm) with a dark wood finish. The surface has not been altered, and is thus pitted with numerous scratches and is quite shiny. The computer is a standard personal computer with an Intel Pentium III microprocessor operating at 933MHz and Microsoft Windows 98 operating system. It has a video/audio capture card (Winnov Videum AV), an Ethernet network card, a second video display card, and two serial ports for control of the camera and projector. The Canon VC-C3 Communication Camera is connected to the computer via a composite video cable and an RS-232 cable for controlling 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 and has maximum 10x optical zoom. The camera is mounted above the desktop on a subplate attached at an angle of 25° to a vertical steel

plate. The main plate can be repositioned along a horizontal steel beam. See Figure 5 for a diagram of the mounting, and a photograph of the setup. 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. The data projector is a Telex P600 LCD projector with display resolution 1024 x 768 and 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.

The Live Paper software system consists of three types of support modules: image processing, including finding and identifying paper, motion detection, and finger tracking; state handling, including monitoring the positions, contents and associated transparencies of all pages known to the system; and user interface. The software also provides an interface for transparency modules. Transparencies are akin to LivePaper applications, and can be added to the system easily, via a clean API. LivePaper consists of 20,000 lines of purpose-written C++ code and is implemented on top of custom libraries for support of multimedia communication, developed by L-T Cheng at Memorial University. End-to-end latency (from camera acquisition to projected output) is on the order of 250ms to 500ms, depending on the number of active applications.

3.3 Applications

LivePaper applications are implemented in *transparencies* – bundles of functionality that can be applied to pages that the system has seen. In some cases a transparency will require access to stored data to associate with a page (for example, a CAD model to associate with an architectural plan). At present this information must be entered through a keyboard. 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.

We describe 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.

Music

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 fig 6), which allow the user to play, pause, stop, skip, and display a menu of songs.

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 transport button stack does move as the paper is moved.

Architecture

The Architecture Renderer transparency displays a perspective-corrected three-dimensional extrusion of 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's viewpoint is approximately 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 fig 7). 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. 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.

Collaboration

The Remote Collaborator transparency establishes a link with the networking facilities of *Live Paper*, and shares the image of the page. Users of conventional personal computers located at remote sites can connect to the *Live Paper* system and view the pages that have this transparency. 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 fig 8.). The tab icon of the Remote Collaborator changes from a dark red arrowhead to a bright green, animated arrowhead when a remote user connects to the system. This is a visual clue to the local user that a remote user is currently able to view the page. 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 (fig 9).

4 Technologies

In order to create an augmented desk environment that successfully simulates paper with embedded computing capabilities, we have investigated a number of issues. In this section, we outline some of the research we have done in four key areas.

4.1 Page finding

In a desktop environment, the user can easily vary the location of pages both during the process of writing and when finished. Before the system can process a page, it must successfully locate that page and extract a perspective corrected image of the marks. The current implementation uses four steps to locate non-overlapping pages:

segmentation, edge finding, edge extraction, and page finding. Figure 10 shows an example image of the desktop as it passes through each step. When overlap is detected, a further four steps are triggered (fig 11) to estimate the locations of hidden corners.

Implementation

Segmentation is based on grey-level thresholding. In the simple case of one page on a desktop, the grey-level histogram for the image is well separated into two regions. However, as more objects and pages are added to the desk, the separation in the histogram is reduced, and multiple peaks can form. A fixed threshold is not used because there are variations in brightness due to lighting and camera properties.

We use a variation on the method developed by Otsu to automatically select a threshold for picture segmentation. The Otsu thresholding method [14] uses discriminant analysis to divide the histogram into two or more classes. The thresholds chosen are optimal from the viewpoint of discriminant analysis. We have found that searching for the best two-class division of the histogram does not always return an appropriate threshold, especially when there are large amounts of clutter. However, the three-class division is quite robust, and a simple heuristic is used to determine which of the two thresholds is best for separating the pages from the desk regions. Boundaries are found by examining the neighbours of each pel classified as belonging to a page. These boundaries are then extracted and stored in a chain-code using a contour-following algorithm. The algorithm uses the previous direction to determine the search order for neighbouring pels. Extremely short contours, and those wholly within other contours, are immediately eliminated. The boundary chain-codes are examined to find corners and straight line segments [8]. This method is described in more detail in [16].

When overlap is detected, the system feeds the information already derived into a geometric inference module. This looks for candidate hidden corners between lines that do not meet. Suitable pairings are organized into a graph that is progressively pruned to estimate the most likely configuration of corners. The method is described in more detail in [17].

4.2 Page Identification

The system must rapidly determine if a sheet of paper on the desk has already been analyzed and stored. It is a waste of both computer and network resources to fully reprocess a stored sheet. The best technique to identify a sheet is to use its current location and the location of similar sheets or sheet elements in the newly captured frame. For small time differences, it is valid to map sheet elements based on their locations and orientations. However, if the system does lose track of a sheet, then an alternate method of identification is necessary. The augmented desk stores images of all processed sheets (see Figure 12 for sample pages), and uses a Hausdorff distance measure to compare a new sheet of paper image to the stored images.

Implementation

To identify a page, the system first locates the page, and then extracts the image area using a bilinear texture-mapping algorithm. The page image is segmented into writing and non-writing areas by application of the Moving Averages thresholding algorithm [29]. The algorithm was designed for scanned document images of black text on a white page, and works well even in the presence of non-uniform lighting. The system then uses the Hausdorff distance measure to compare this binary image with a database of stored images. The Hausdorff distance is a measure of similarity between sets of points, but it does not establish a one-to-one correspondence. A number of researchers have investigated its use for image similarity [5][9][10]. For binary page images, the mark pels are used to generate the point sets. The smallest distance indicates the best match – a cut-off distance is used to decide whether the system has already stored the page or not. Further details of our procedure may be found in [16].
By using the Hausdorff distance on the two-level page image directly, it is possible to identify each page rapidly without requiring glyph codes, explicit identification tags [19], or large matrix codes [11]. In experiments on a set of 171 pages, we realized an accuracy of page identification of 93% using an image resolution of 5 dpi. By zooming the camera head to obtain a resolution of 15 dpi, the accuracy is over 99%. The technique is also fast, even when comparing a page with a database of several hundred stored pages. The use of the Hausdorff distance does not preclude other identification techniques.

When the page is changed through annotation or other writing (see Section 4.4. below), a new version is stored for future matching.

We have not yet conducted a systematic evaluation of page identification performance when parts of a page are occluded. But because augmentation provides direct feedback to the user, errors in identification of overlapped sheets are immediately evident, and the user can quickly correct the system by moving the page in question so that it is on top.

4.3 Finger tracking

The restriction of gestural interaction to transports and their cascading menus simplifies the location and tracking of the user's fingers. The system generates a list of hotspots on the tabletop, according to the location of buttons. These locations are transformed to regions in the acquired image. When the user points at a button, their finger is illuminated by the projected augmentation. The result is a significant change in the luminance of the imaged button area. When the system detects that a significant number of pixels in a hotspot exceed a change threshold, the region is triggered. A pointing finger triggers all the hotspot regions it occludes. The one furthest from the user corresponds to the fingertip, and is therefore taken as the activated button. This method proves to be robust as well as fast.

4.4 Videowriting

Videowriting is the use of a video camera to extract, in real-time, new writing from a sheet of paper. This capability is built into all LivePaper applications so paper can be annotated (and the page image changed) at will. In addition to the basic tasks of extracting high-quality writing while ignoring the user's hand, the resulting mark data must be compressed and stored, and transmitted during collaboration sessions. Because the extraction algorithm will likely make some mistakes, the system must be capable of deleting falsely extracted marks, both locally and remotely. The ideal methodology is one that incorporates various algorithms to robustly detect writing on different surfaces, at non-ideal angles, and with any writing instrument. Currently, several basic assumptions are made to reduce the complexity. The writing surface must be a sheet of white paper. The writing instrument is a dark ink marker, with a pen tip sufficiently wide to be detected at the captured resolution. With these restrictions, we have developed a methodology which reliably extracts writing in real-time. Details are provided in [18].

One practical consideration that will always be present is the occlusion of newly written material by the user's hand. This might require the user to be mindful of periodically removing his or her hand from the page, so that the system can update the list of marks. A VAE desktop with projective capabilities could prompt the user when necessary.

Implementation

To incrementally extract marks, three steps are used. The captured image of the page is adaptively thresholded to separate the marks from any other clutter on the page. The thresholding process does misclassify some areas of the user's hand and pen as writing, and so the system classifies a mark based on its temporal permanence. Finally, the difference between previously extracted marks and those in the current frame is used to find new marks. Videowriting uses an adaptive grey-level thresholding algorithm designed similar to Moving Averages [29] which visits each pel on horizontal and vertical passes. For each pass, the algorithm averages the current pel with l pels before and after it. If the current pel is t levels darker than the average in either pass, then it is considered to be a mark pel.

Permanence of a mark is currently established by ANDing several consecutive thresholded images. Pels that were set as mark pels repeatedly are considered permanent mark pels. To extract marks incrementally, the system compares the image generated by adaptive thresholding with an image created from marks stored in the database. New marks are extracted, and then stored as binary arrays, along with additional information such as its coordinates and identification number. Updates occur when there is no motion for a short period of time, which indicates that the user has remove his or her hand from the page. Any pel that has been mistakenly classified as a mark pel is deleted from the appropriate mark. Each mark is permitted to have a few false pels, but if a significant number of pels are deleted, the entire mark is purged.

5 Conclusion

Through design of the LivePaper system, we have developed guidelines for user-interface organization in augmented reality systems. These are summarized:

- Divide functionality into that which is associated with particular objects, that which involves cooperation between objects, and that which pertains to the whole environment.
- Determine the literal part of object-oriented function, being the augmentation that corresponds directly to the physical form and conventional use of the object.
- Provide user access to literal functionality by direct object manipulation, and display of literal information by registered graphical overlay on the object.
- The remaining object-oriented functionality is its magical component, being the augmentation that does not correspond to the conventions for the object.
- Display magical functionality so that its association with the object is clear, but so that it forms a detached annotation rather than an overlay.
- Allow user manipulation of magical controls in a way that has nothing to do with the literal, conventional use of the associated object.
- Where objects have multiple augmented functions, superimpose the literal overlays on the object, but spatially organize the magical functionality so that the alternatives are clear.
- Support cooperation between objects by displaying appropriate literal information overlaid on each and by duplicating the magical part of the interface. Other magical cues, such as displaying links overlaid between objects and using flashing tabs, may be used to highlight the integration of several objects' functionality.
- Support environmental functionality with a virtual environmental control object which is simply a reserved location in the environment, or by replication of magical tabs on all objects. Environmental functionality should not be embedded in the literal augmentation of objects.

These rules have been embodied in LivePaper, and we offer them as possible guidelines for design of other augmented and mixed reality systems.

References

[1] Billinghurst M, Kato H. Collaborative Mixed Reality. International Symposium on Mixed Reality (MR '99), Yokohama, Japan, March 1999, pp. 261-284.

[2] 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.

[3] Cheng L-T, Robinson JA. Dealing with Speed and Robustness Issues for Video-Based Registration on a Wearable Computing Platform. Proceedings of the Second International Symposium on Wearable Computers, Pittsburgh, PA, Oct 19-20, 1998, pp 84-91.

[4] Cheng L-T, Robinson JA. Personal Contextual Awareness through Visual Focus. To appear in IEEE Intelligent Systems Journal.

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

[6] Feiner S, MacIntyre B, Seligmann D. Knowledge-Based Augmented Reality. Communications of the ACM, Vol 36, No. 7, July 1993, pp 53-62.

[7] Fox D. Composing Magic Lenses. Proceedings of CHI 98. Los Angeles, CA, April 1998, pp. 519-525.

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

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

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

[11] Kobayashi M, Koike H. EnhancedDesk: Integrating Paper Documents and Digital Documents. Asia Pacific Computer Human Interaction 1998. IEEE, Japan, 1998, pp. 57-62.

[12] Hymes CM, Olson GM. Unblocked Brainstorming Through the Use of a Simple Group Editor. Proceedings of CSCW '92. ACM, Nov 1992, pp. 99-106.

[13] Newman W, and Wellner P. A Desk Supporting Computer-based Interaction with Paper Documents. Proceedings of CHI '92. ACM, 1992, pp. 587-592.

[14] Otsu N. A Threshold Selection Method form Gray-Level Histograms. IEEE Trans. on Systems, Man, and Cybernetics. SMC-9, no. 1, Jan 1979. pp. 62-66.

[15] Rekimoto J. NaviCam: A Magnifying Glass Approach to Augmented Reality. Presence: Teleoperators and Virtual Environments. vol. 6, no. 4, August 1997. pp. 399-412.

[16] Robertson C, Robinson JA. Page Similarity and the Hausdorff Distance. Proceedings of the Seventh International Conference on Image Processing and its Applications, Manchester, UK, July 1999, pp 755-759.

[17] Robertson C, Robinson JA. Live Paper: Video Augmentation to Simulate Interactive Paper. Proceedings of ACM Multimedia 1999, Orlando, Florida, Oct 1999, part 2, pp 167-170.

[18] Robertson C. Live Paper. PhD Thesis, Memorial University of Newfoundland, in preparation.

[19] Robinson P. The Origami Project: A framework for interacting with paper. Proceedings of EUROGRAPHICS '97. Vol. 16, No. 3. Available at http://www.cl.cam.ac.uk/Research/Origami/Origami1997c/index.html

[20] Saund E. Image Mosaicing and a Diagrammatic User Interface for an Office Whiteboard Scanner. Technical report, Xerox Palo Alto Research Center, 1998.

[21] Smith RB. 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.

[22] Stafford-Fraser Q, Robinson P. BrightBoard: A Video-Augmented Environment. Proceedings of CHI ’96. Vancouver, Canada, 1996. pp. 134-131.

[23] Starner T, Mann S, Rhodes B, Levine J, Healey J, Kirsch D, Picard RW, Pentland A. Augmented Reality through Wearable Computing. Presence: Teleoperators and Virtual Environments. Vol. 6, No. 4, August 1997. pp. 386-389.

[24] Stifelman L. Augmenting Real-World Objects: A Paper-Based Audio Notebook. Proceedings of CHI ’96. Vancouver, Canada, 1996, pp. 199-200.

[25] Ullmer B, Ishii H. The metaDESK: Models and Prototypes for Tangible User Interfaces. Proceedings of UIST ’97. pp. 223-232.

[26] Ullmer B, Ishii H, Glas D. mediaBlocks: Physical Containers, Transports, and Controls for Online Media. Computer Graphics Proceedings / SIGGRAPH ’98. ACM, July, 1998. pp. 379-386.

[27] 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.

[28] Weiser M. The computer for the 21st century. Scientific American. Sept, 1991, pp. 94-104.

[29] Wellner P. Interacting with Paper on the DigitalDesk. Communications of the ACM. Vol. 36 No.7, July 1993, pp. 86-96.

Figure Captions

Figure 1. The LivePaper System in Use

Figure 2. LivePaper Components on the Tabletop.

Figure 3. Collaborative Drawing and Brainstorming application

Figure 4. Debugging (also showing the architectural application)

Figure 5 (a) Mounting Diagram (b) Photograph of Mounted Projector and Camera. The projector is attached to a vertical plate hanging from a reinforced horizontal bar. The pan-tilt-zoom camera is adjacent, attached to an angled plate, ensuring the full tilt range can be exploited.

Figure 6 Music Transparency

Figure 7 Architecture Transparency

Figure 8 Remote Collaboration with Traditional Personal Computer User

Figure 9 Example of how displayed annotations appear to be locked to the page.

Figure 10 Procedure for Detecting Sheets

Figure 11 Procedure for dealing with overlapped sheets

Figure 12 Examples of extracted pages

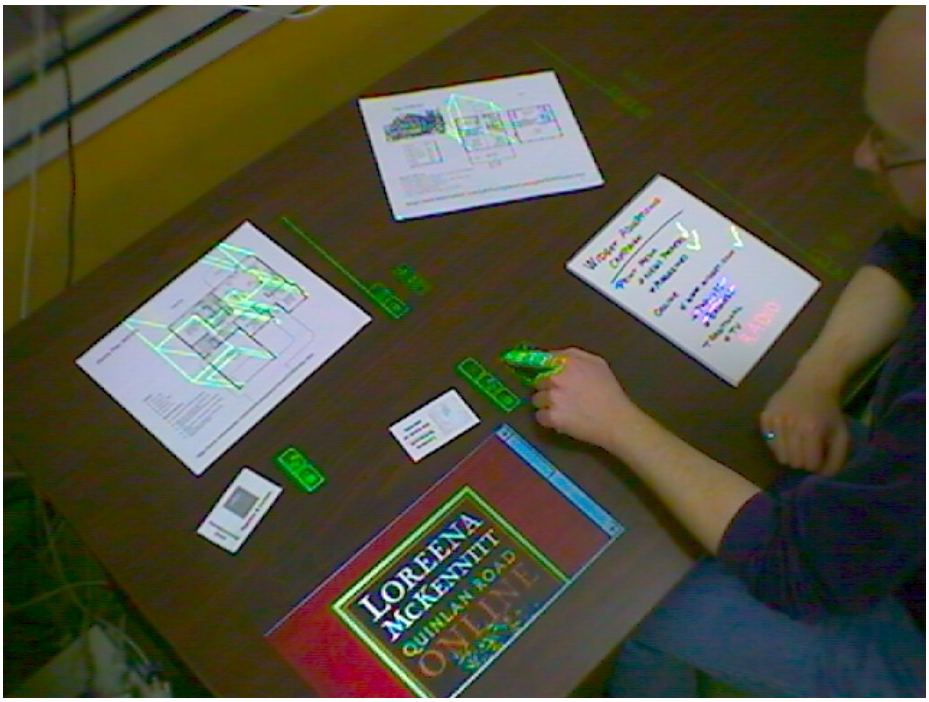

Figure 1. The LivePaper System in Use

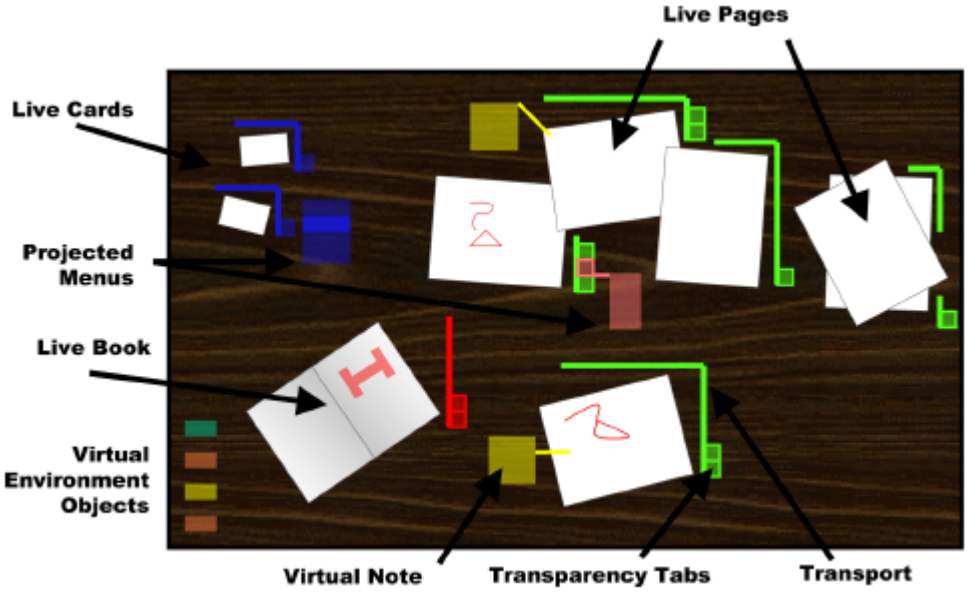

Figure 2. LivePaper Components on the Tabletop.

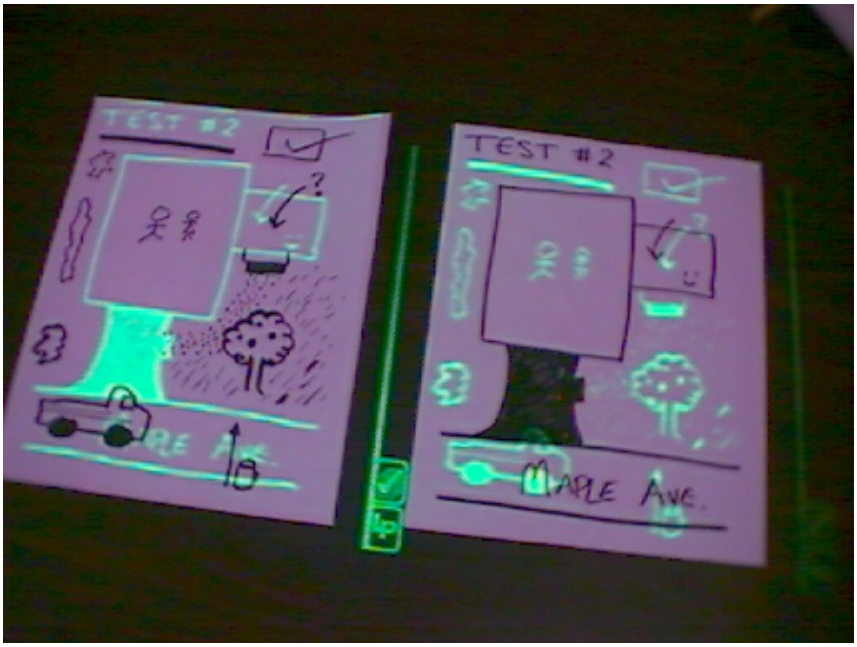

Figure 3. Collaborative Drawing and Brainstorming application

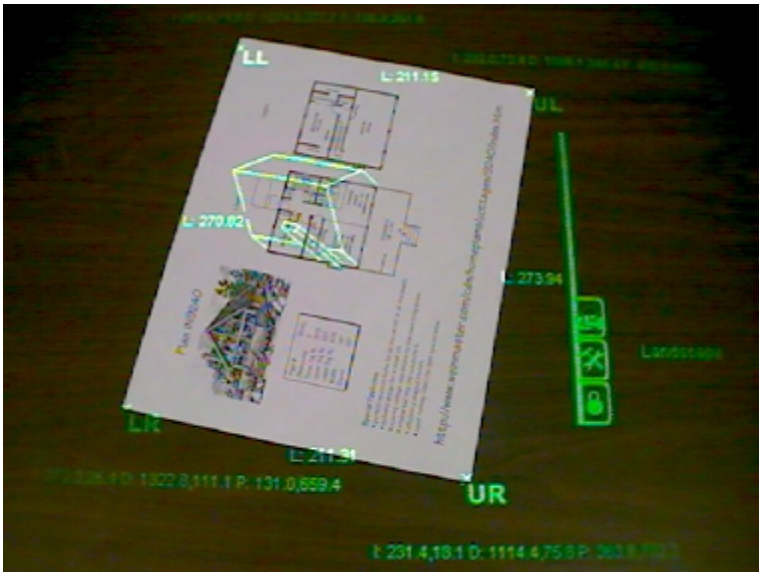

Figure 4. Debugging (also showing the architectural application)

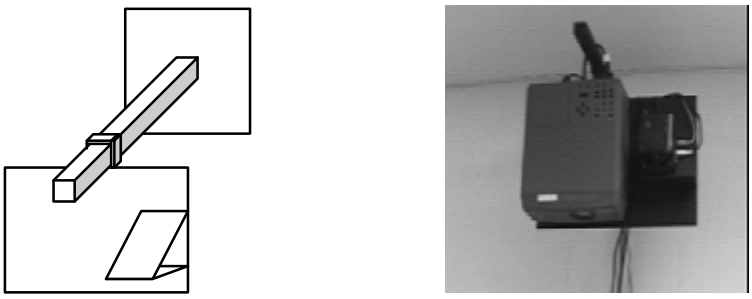

Figure 5 (a) Mounting Diagram (b) Photograph of Mounted Projector and Camera. The projector is attached to a vertical plate hanging from a reinforced horizontal bar. The pan-tilt-zoom camera is adjacent, attached to an angled plate, ensuring the full tilt range can be exploited.

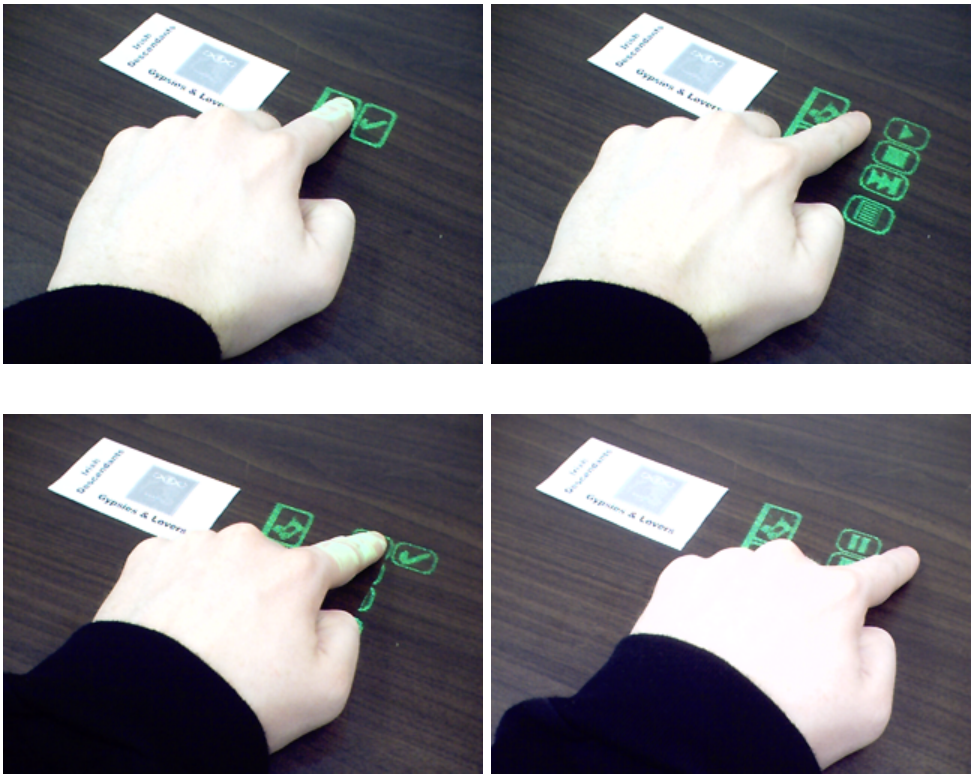

Figure 6 Music Transparency

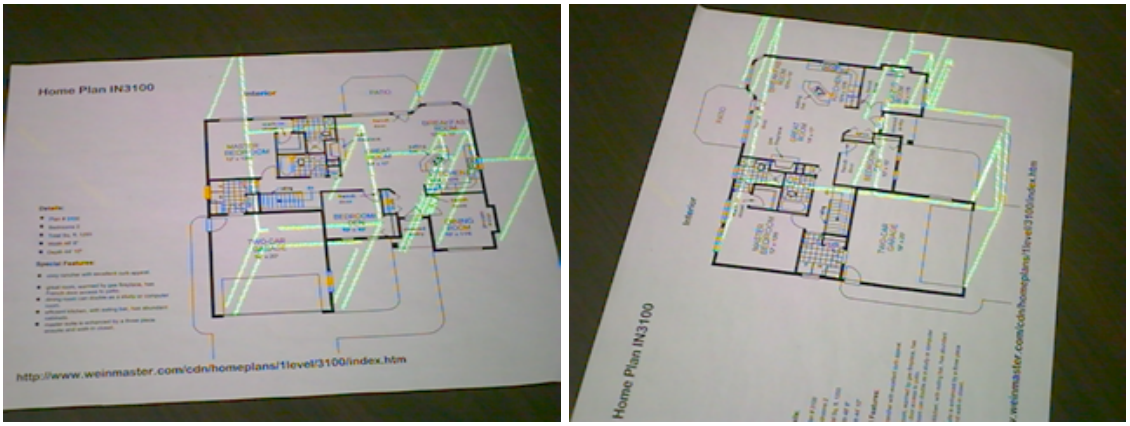

Figure 7 Architecture Transparency

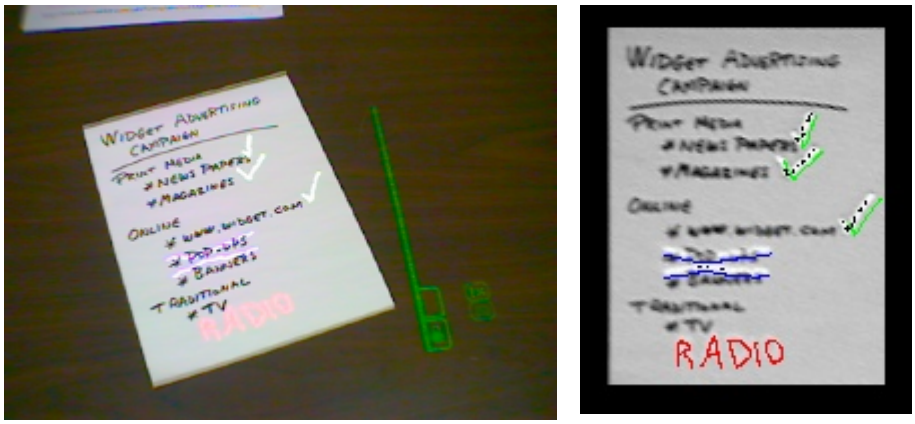

Figure 8 Remote Collaboration with Traditional Personal Computer User

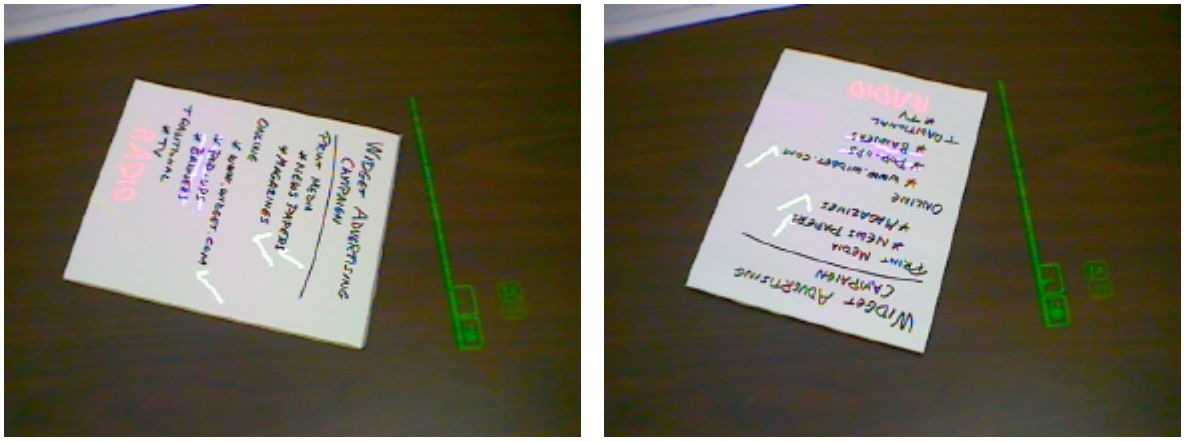

Figure 9 Example of how displayed annotations appear to be locked to the page.

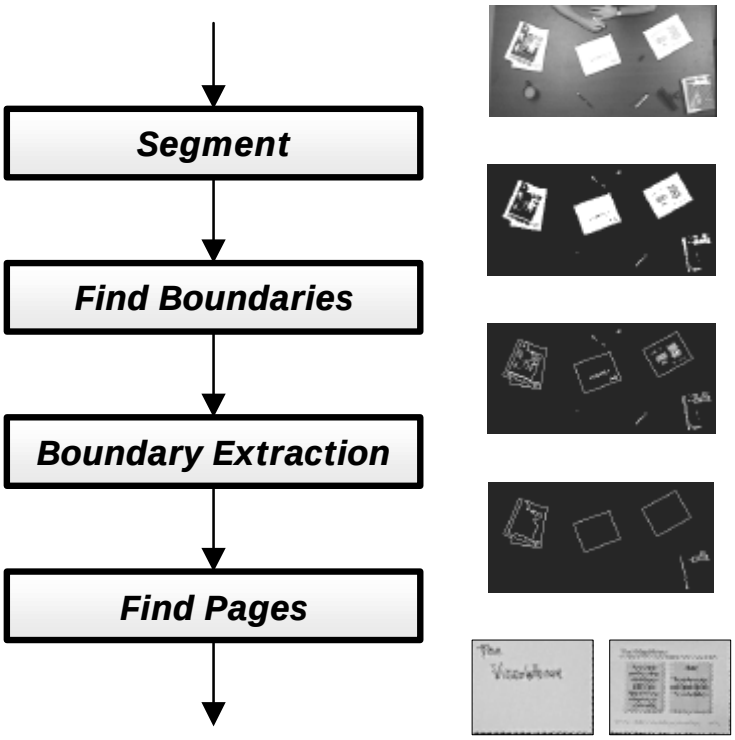

Figure 10 Procedure for Detecting Sheets

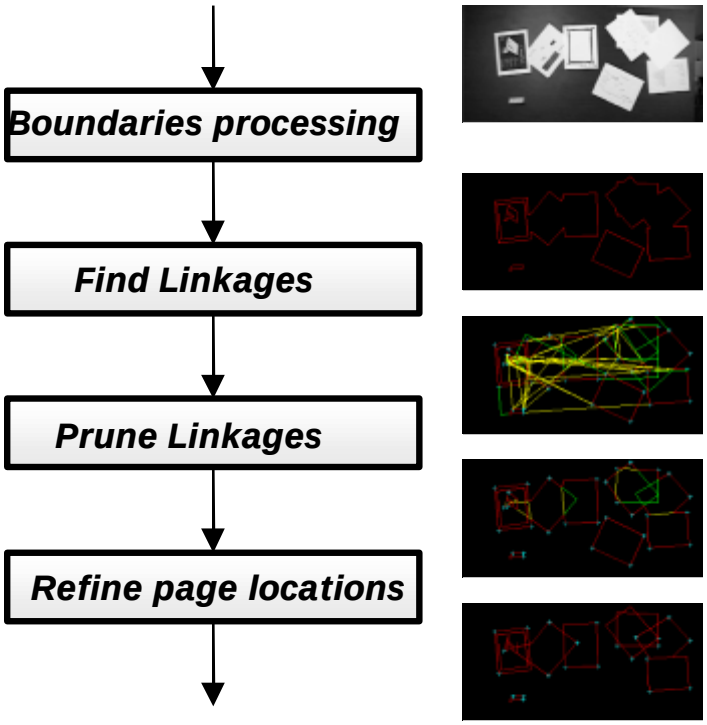

Figure 11 Procedure for dealing with overlapped sheets

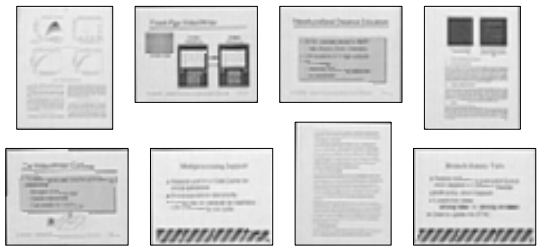

Figure 12 Examples of extracted pages