The Intelligent Interactive Book

by

Maosha Shi

Thesis

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

September 2009

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Maosha Shi

September 8, 2009

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Maosha Shi

September 8, 2009

Acknowledgments

I would like to thank Dr Kenneth Dawson-Howe for all his help, guidance and advice throughout the course of this project. He was always helpful and very enthusiastic in his approach to this project.

I would also like to express my appreciation to all my wonderful friends who provided me with love, caring, and inspiration during difficult times.

Finally, I am so very grateful for the family I am blessed with. To my mother and father, who put a lifetime of effort and love into my upbringing. Thank you for everything you have done for me.

Maosha Shi

University of Dublin, Trinity College
September 2009

The Intelligent Interactive Book

Maosha Shi
University of Dublin, Trinity College, 2009

Supervisor: Kenneth Dawson-Howe

We developed an intelligent interactive book using computer vision techniques to identify the page and monitor user input in this dissertation. When reading this book, reader can use their fingers to interactive with a paper based page. A fixed webcam used to "read" the current page's content and user's action, and then show the result of interaction on a screen.

In the run-time, the system automatically find current page's corresponding reference image, and location pointing finger's position to decide what kind of result should be shown. We developed a method to recognize the page based on template matching. The user input has been located as the combination of motion and skin color detection result.

Taking advantages of a popular webcam as input results a low cost system. And the innovative marker design brings reader better experience while preserves the natural reading habit well.

Contents

Acknowledgments **iv**

Abstract **v**

List of Tables **ix**

List of Figures **x**

Chapter 1 INTRODUCTION **1**

1.1 MOTIVATION 1

1.2 GOAL 2

Chapter 2 STATE OF THE ART **4**

2.1 BACKGROUND 4

2.2 RELATED WORK 4

2.3 RECOGNITION TECHNIQUES 6

2.3.1 Markers 6

2.3.2 Key Feature Points 7

2.3.3 Manually matched by the user 8

2.3.4 Embedded ID 8

2.3.5 Conclusion 9

2.4 INTERACTION WITH THE PAPER 9

2.4.1 Pen Based 10

2.4.2 Marker Based 11

2.4.3 Finger Based 11

2.5 RESULT DISPLAY 14

2.6 SUMMARY 14

Chapter 3 DESIGN **15**
3.1 PAGE RECOGNITION 16
3.1.1 Requirements 16
3.1.2 Potential Solutions 16
3.2 USER INPUT 17
3.2.1 Requirements 17
3.2.2 Potential Solutions 17
3.3 SYSTEM ARCHITECTURE 18
3.4 ABOUT OpenCV 19

Chapter 4 PAGE RECOGNITION **20**
4.1 SURF 21
4.1.1 Fast-Hessian Detector 22
4.1.2 SURF Descriptor 22
4.1.3 Matching 23
4.1.4 Implementation Using OpenCV 23
4.1.5 Result 23
4.2 HOMOGRAPHY 26
4.2.1 Definition 26
4.2.2 OpenCV Functions 27
4.3 TEMPLATE MATCHING 28
4.3.1 Template Matching Algorithm 29
4.3.2 Markers for Location and Transformation 31
4.3.3 Transformation and Matching 33
4.4 COMPARISON AND CONCLUSION 35

Chapter 5 INTERACTION **37**
5.1 SKIN COLOR DETECTION 38
5.1.1 HSV Color Space 38
5.2 MOTION DETECTION 41
5.2.1 Background Substraction 41
5.2.2 Motion Template 42

Chapter 6 EVALUATION **44**
6.1 RESULT 44
6.1.1 Page Recognition 44
6.1.2 User Input and Result display 46

Chapter 7 CONCLUSION **48**
7.1 CONTRIBUTION 48
7.2 FUTURE WORK 49

Appendices **51**

Bibliography **53**

List of Tables

2.1 Conclusion of existing system 14

List of Figures

1.1 Examples of interactive books. A children story book(top left); a electronic version of Vogue(top right), and a ebook for learning Chinese. . . 2

2.1 2D matrix code examples (left: original,right: restored from video). . . 6
2.2 A book with two-dimensional matrix code. 7
2.3 RFID transponders (tags). Flexible copper printed onto clear plastic, embedded between two printed sheets to create one page thickness for the Listen Reader books. 8
2.4 Anoto Technology. 10
2.5 Extraction of hand region by an infrared camera. 12
2.6 The state transition diagram of WikiTUI. Shapes with dotted lines represent future features of the system. All users start at the idle/hover state. When a user pauses her fingertip on top of a trace for 2 seconds, the system enters the object selection state. If she pauses on the object for another 2 seconds, then the associated digital medium will either start playing or become minimized depending on the systems status; the system then arrives the restore/minimize state. If she decides to move her fingertip instead of pausing longer, then the selected object is also moved; thus the system become in the moving objects state. 13

3.1 The overall system architecture of Intelligent Interactive Book. 18

4.1 Input images 20
4.2 A reference image with feature points. 24
4.3 A correct sample of SURF. 24

4.4 An incorrect sample of SURF. 25
4.5 View of a planar object as described by homography. 26
4.6 Template Matching Example 29
4.7 Image processing to find markers. 32
4.8 Corners's position and order passed to calculate the transform matrix. . 33
4.9 Page's layout in a frame. 34
4.10 Frame Transformation Result and a reference image. 34
4.11 Template Matching vs. SURF 36

5.1 The input image of finding User Input. 37
5.2 HSV arranged as a cylinder.. 38
5.3 Skin color samples. 39
5.4 Skin color extraction result. 40
5.5 Marker color extraction result. 40
5.6 Background image without any moving object. 41
5.7 From left to right: the current frame captured from webcam; the image after compute the difference between background and current frame; result after connected component operation. 42
5.8 Motion Template Diagram. 42
5.9 Input images 43

6.1 Page recognition result of six sample pages. 45
6.2 Finger location and interaction result display. 46
6.3 Open/Close a sample image by pointing to a picture. 47

7.1 The improved system architecture. 49
7.2 A thick book which show deformed pages when open it. 49
3 Class Diagram. 52

Chapter 1

INTRODUCTION

Reading online is popular nowadays, as there is abundant book resource there. What's more, some electronic books bring readers interactive experience. There are a lot of interactive book designed for children, student and adults(Figure 1.1).

Those books enriched reading experience by qualified pictures, lively animations, beautiful music and so on. When a child open a interactive story book, he or she can click a button to ask the book to read stories to them, or use mouse play games with characters in the story. If you use an interactive book to learn language, you can click on any work in a sentence, and the book will show you how the pronunciation and meaning of the word. If you read a interactive magazine, you can click on a picture to zoom in, and reading with a beautiful music. Those media and interactive really make our reading more interesting, our learning more effective.

1.1 MOTIVATION

But reading on screen also brings problem. It makes our eyes tired after long time staring at the screen. People still like paper which can be read easier, and also portable. Human reading habit is formed during reading a paper based book. The problem is, reading on paper will lose such colorful experience brought by media and online

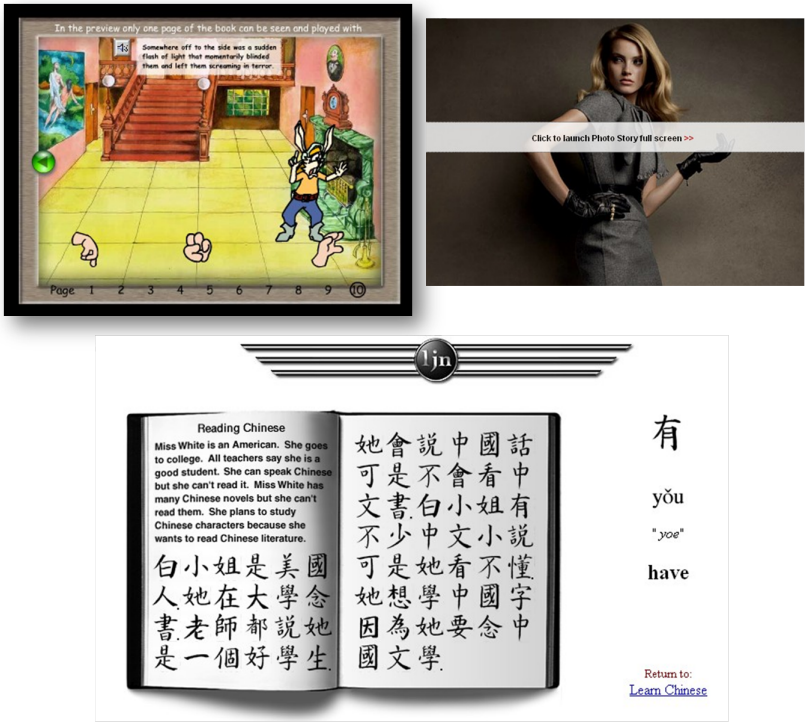


Figure 1.1: Examples of interactive books. A children story book(top left); a electronic version of Vogue(top right), and a ebook for learning Chinese.

resource. Our project's idea comes from here. We want to build a bridge between the paper based book and hyper-media.

1.2 GOAL

The goal of this project is to develop a reading interface (paper based) which can enrich user's reading experience. We want to develop a system with a paper based book and a webcam which is popular today, to achieve human interaction with the paper book. If reader click on some area (such as figures, buttons) the system will show some result(play a video, audio; Open a website...). With that, when user reading a paper book, he or she can also enjoy the benefit brings by interaction and media.

- In chapter 2, we will introduce you the related projects in this area.You can see

the research of this has started from 1990s, and also some interesting books.

- Chapter 3 shows our design of the system, what kind of implementation issues we are facing and potential solution to them.
- Chapter 4 and 5 show the details of how we solve those problem and the underlying techniques in computer vision.
- Chapter 6 show the how the system works in some applications.
- Chapter 7 will conclude all of this for you.

Chapter 2

STATE OF THE ART

2.1 BACKGROUND

The e-book and the Physical book both have its own advantages and disadvantages. The challenge to reading system design is to provide the best things of both types of books. The researches in this area started from 1990s. Sections below will show you those systems and we will also consider them pro and con.

2.2 RELATED WORK

The classical approach to interacting with the book is represented by the DigitalDesk[18], developed in 1993. The DigitalDesk is a real physical desk, but it is enhanced to provide some characteristics of an electronic workstation. A computer display is projected onto the desk to show the result of interaction, and the sense what the user is doing is captured by a video camera which pointed down at the desk. The images captured by the camera will be processed by an image-processing system. This system has 3 important characteristics:

- it projects electronic images down onto the desk and onto paper documents;

- it responds to interaction with pens or bare fingers;
- it can read paper documents placed on the desk.

Koike et al.[11] developed an augmented desk interface system, EnhancedDesk, and introduced Interactive Textbook, an application developed on this system, which is aimed at providing an effective learning environment. When a student opens a page which describes experiments or simulations, Interactive Textbook automatically retrieves digital contents from its database and projects them onto the desk. Interactive Textbook also allows the student hands-on ability to interact with the digital contents. This system uses the matrix code which is attached to the pages to identify the pages and an infrared camera to monitor the user's hand.

While some system focus on enhancing the content by linking it digitally with content on another surface, some others developed a new digitally enhanced books which fully augmenting the real content. For example, The mixed reality book[9] symbiotically merges different type of media in a seamless approach. This solution utilizes recent developments in computer vision tracking, advanced GPU technology and spatial sound rendering. The systems collaboration capabilities also allow other users to be part of the story. The user can use a handheld display to experience the augmented book as well as use a marker to interact with the book.

In the latest research on enhancing physical books, WikiTUI [20] brings digital media to physical paper books. This system allows readers to add and access digital content through fingertip interactions on books, and enables them to share information with other readers using wiki technology.

When designing an enhanced physical book, the problems that researchers faced can be divided into two aspects.

First Page identification - how to detect the current page ID from the physical book?

Second How the interaction on the book can be implemented?

In the sections below we present the existing solutions to those issues.

2.3 RECOGNITION TECHNIQUES

The main goal of this section is to detect the page number from the physical book. The techniques we show can be deviled into four categories:

2.3.1 Markers

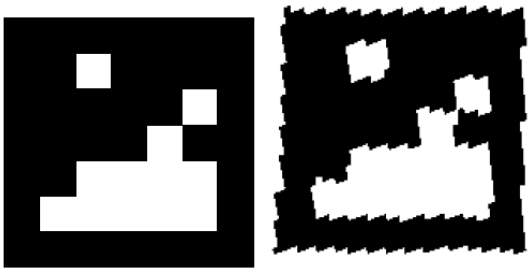


Figure 2.1: 2D matrix code examples (left: original,right: restored from video).

The "Matrix" method identifies real world objects and estimates their position and orientation using a combination of visual markers (Figure2.1) and a video camera.[15] The 2D matrix marker is a square shaped barcode that can identify 2^{16} different objects. By analyzing the distortion of the rectangular shape of the matrix code frame, the system estimates the position and orientation of the video camera.

In EnhancedDesk [11] a matrix code is attached to the page which has digital contents(Figure 2.2).

Each matrix code corresponds to each application program. When the user opens a page containing a matrix code, the system recognizes the matrix code and projects the digital content onto the desk. The system not only identifies the unique ID of the matrix code but also recognizes its size and orientation. By using such size and orientation information, the system decides where to project the digital contents.

This method seems to be the easiest way to locate and identify the page, but the marker breaks the origin style of a physical book and makes the illusion less convincing.

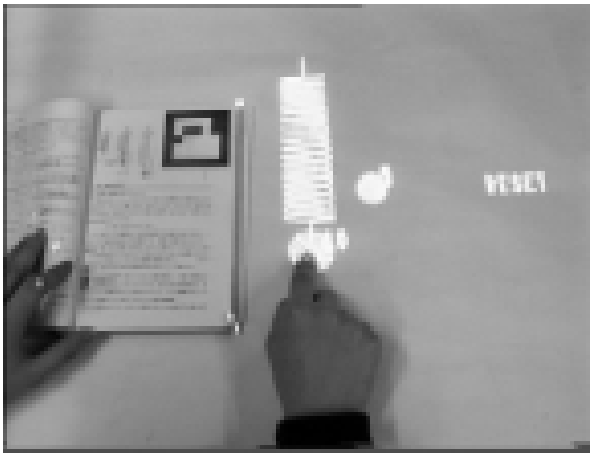


Figure 2.2: A book with two-dimensional matrix code.

2.3.2 Key Feature Points

The Haunted Book animates the illustration of a poetry book using the method described in [14]. The book registration is completely based on natural features. The registration system relies on one single reference image for each page to augment, for which the artist designed animated layers using AfterEffects. At run-time, the system estimates the homography registering the captured frame and the correct reference image, and applies it to the animated layer for augmentation. When this system starts, the software keeps looking for the same page for the last captured frame. This will keeps a short delay - in the worst case, the system will try all ten pages in the book before finding the correct one. One crucial feature of Keypoint Recognition which used by the Haunted Book is the book can be read also without a camera and Augmented Reality.

The main problem of this technique is it will take several seconds to find the correct page then remain interactive. The reaction time of the system can be very long when the book has hundreds of pages.

2.3.3 Manually matched by the user

In the WikiTUI prototype[20], rather than detecting page numbers from the physical book, the system projects page numbers corresponding to the location of digital annotations. The page numbers are projected at the two bottom corners of the desktop on either side of the book. User must manually match the page numbers for digital annotations to whatever page they are reading in the book using the projected page numbers. They can place their index finger on the projected page number at the bottom right to advance pages of digital annotations. Thus the projected page numbers act as page turn buttons, and are used to keep the displayed annotations in sync with the physical book page.

2.3.4 Embedded ID

Listen Reader [1] is an electronically augmented paper-based book. It combines the look and feel of a real book with the rich, evocative quality of a movie soundtrack. The book provides a naturally interaction with human user by using a new RFID technology (the TIRIS TagIt system from Texas Instruments, with simultaneous ID) to recognize what page a book is open to[2].

Figure 2.3: RFID transponders (tags). Flexible copper printed onto clear plastic, embedded between two printed sheets to create one page thickness for the Listen Reader books.

A thin flexible transponder tag(Figure 2.3) with a unique ID is embedded in the

paper of each page, and a special tag-reader is affixed to the binding of the back of the book. As the pages turn, the tag-reader notices which tags are within its read range and which have moved out of its range (which is about four inches).

Geometry in tag layout on the pages may cause problem. In this system, two tags may overlap by 80% or more, but not completely. Another problem of this recognition technique is those embedded tags will rise the cost of printing the book.

2.3.5 Conclusion

In a word, web camera based page recognition is the cheapest and the easiest to achieve our goal, however, it just works under particular lighting condition. Instead of identifying the nature page feature, we can detect the matrix marker attached on each page which contains book's position, orientation, and current page number information by the camera. But the markers are unreadable for human, and a book with markers may lose its nature style. Embedded ID can provide us a beautiful binding, good printed paper pages book, as well as a page ID can be easily read by a tag-reader. But the embedded tag brings geometry overlap problem, and makes our system too expensive. Manually match the pages needs user to perform a specific act. If we got a book which has hundreds of pages, in the worst situation, user has to turn the pages hundreds of times. This method brings much more troubles to the user.

2.4 INTERACTION WITH THE PAPER

The main goal of this section is to capture the user's input to the system. There are three main kinds of user input in existing systems:

1. Pen based;
2. Marker based;
3. Feigner based.

Sections below show how those kinds of techniques can be used in an AR book system.

2.4.1 Pen Based

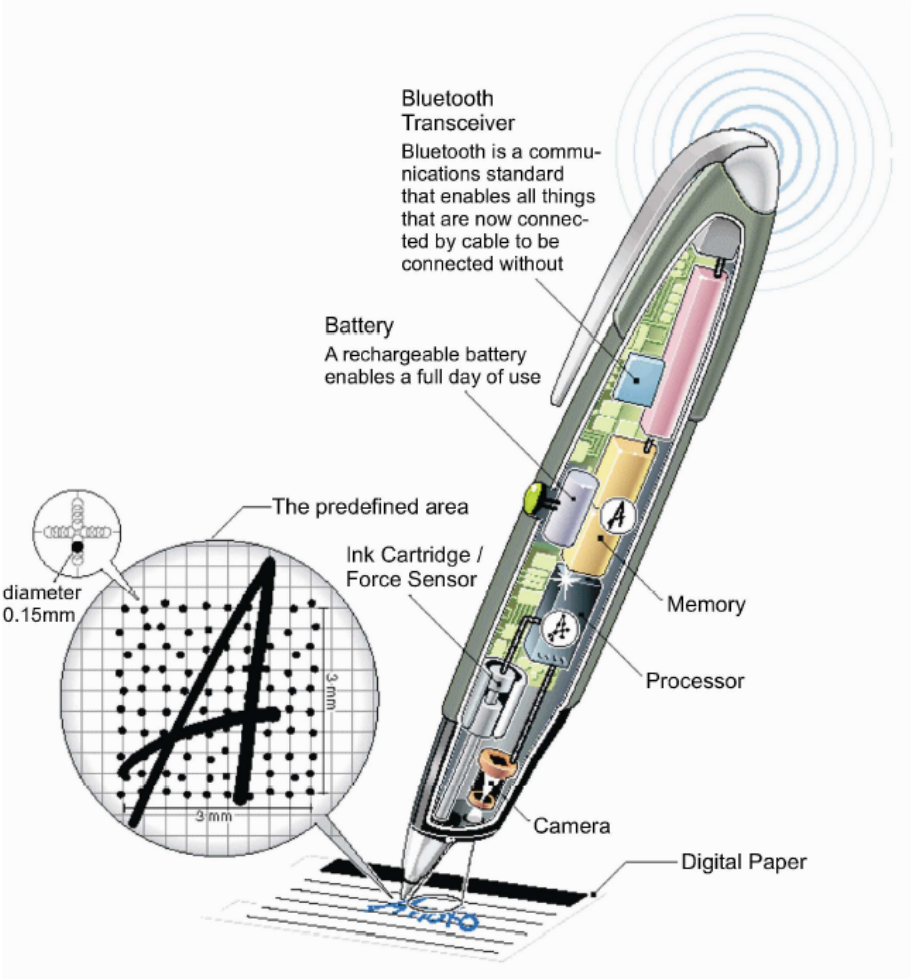


Figure 2.4: Anoto Technology.

Most of the pen based input technology use Anoto digital pen and digital paper (Figure 2.4). Paperproof [17] use this technology to allow users to edit digital documents by means of gesture-based markup of their printed versions. The Anoto technology is able to track and store the position of a pen on paper through a combination of

a special dot pattern printed on the paper that encodes position information and a camera inside the pen. By using this technology we can get accuracy position of the reaction and the camera in the pen can capture all the details in the pages. This is a new way to achieve an accurate interaction with paper, however, to have a digital pen and print a book on digital paper is too expensive for our system.

2.4.2 Marker Based

The Mixed Reality Book[9] uses a marker to interact with the book. The user can use a tangible interaction devices (paddles, cube) to trigger animations in a sequence, like moving the video of user's image around. This kind of interaction is more like playing a game with some tools and it loses the nature of reading habit.

2.4.3 Finger Based

Most of Interactive Books we are interested in like DigitalDesk[18], WikiTUI[20] and EnhancedDesk[11] all use fingers as the input of their system.

DigitalDesk[18] look for the motion of the hands and assume other objects on the desk do not move. They capture successive frames and examine the image by subtracting successive value of each pixel in the two frames then pick out moving object from the background effectively. Determining when the user taps on the desk is difficult to when only processing image from an over-head camera. Their solution to this problem is using a microphone to determine when the user taps. A microphone is attached to the bottom of the desk and the system monitors the amplitude of the signal to determine when the user taps on the desk. It is easy to achieve but the microphone sometimes confuses other taps, bumps on the desk, or hand claps with a finger tap. And the finger following and tap-monitoring tasks must be carefully synchronized.

EnhancedDesk[11] use an infrared camera to monitor user's motion. It first finds the human skin region and then tests the temperature of this region. Figure 2.5 left and middle images show one example of an input image from the infrared camera, and

Figure 2.5: Extraction of hand region by an infrared camera.

a region of human skin extracted by binarization of the input image. To remove those regions other than human skin, they first remove small regions, then select the two regions with the largest size. If only one region is found, they consider that only one arm is observed on the desk. Once regions of user's arms are found in an input image, they use template matching for the fingertips. This method can get accuracy result of both hands' position and achieve multi-finger interaction with the paper. But the infrared camera is not popular today and it will raise the cost of this system too.

In WikiTUI[20], they developed a gesture-based finger-paper interface. possible finger interactions are shown in the diagram in Figure 2.6 and are grouped into five categories:

- Interactions based on action time
- Interactions based on hand shapes
- Interactions based on drawing
- Interactions based on movement
- Interactions with two hands involved

For this technology, user has to spend some time to adapt to the state transition system, and the reaction may be slow.

There are some other finger-paper interaction systems: Bare-hand human-computer interaction[16] gives a finger-finding and hand posture recognition algorithm and shows

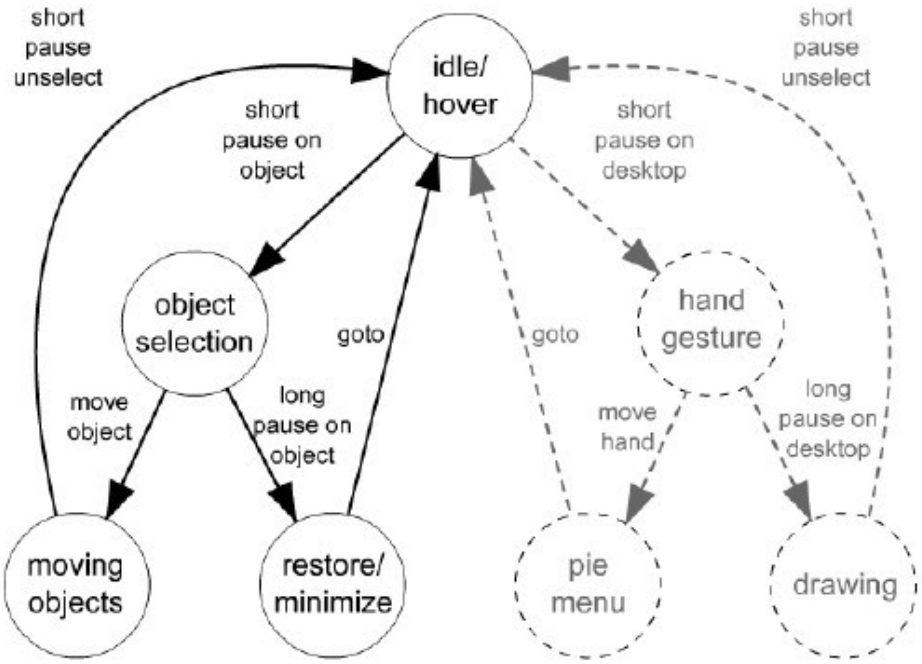


Figure 2.6: The state transition diagram of WikiTUI. Shapes with dotted lines represent future features of the system. All users start at the idle/hover state. When a user pauses her fingertip on top of a trace for 2 seconds, the system enters the object selection state. If she pauses on the object for another 2 seconds, then the associated digital medium will either start playing or become minimized depending on the systems status; the system then arrives the restore/minimize state. If she decides to move her fingertip instead of pausing longer, then the selected object is also moved; thus the system become in the moving objects state.

solutions to issues such as motion blurring and provide a stable finger tracing system.

Visual tracking of bare fingers for interactive surfaces[12]shows the details of foreground extraction and automatics thresholds to the fingertips. Its design is based on modeling of two classes of algorithms that are key to the tracker: Image Differencing Segmentation (IDS) and Fast Rejection Filters (FRF). It also introduces a new chromatic distance for IDS and a FRF that is independent to finger rotation.

Multi-finger interactions with papers on augmented tabletops[6] introduces a novel method which enables multiple users to use both fingertips and physical papers as mediums for interaction.

2.5 RESULT DISPLAY

We can get the location and orientation information from the page identification stage and project the result to the approximate place. Projected Augmentation[7] use a projector that can be rotated and in other ways controlled remotely by a computer, to follow objects carrying a marker.

2.6 SUMMARY

Table 2.1 shows the basic information of the existing system we introduced above.

Name	Input	Page Identification	Output	Applications
DigitalDesk [18]	Camera; Microphone	Printed Number	Projector	Calculator; PaperPaint
Interactive Textbook[11]	Infrared camera	2D matrix code	Projector	Physics Books
The mixed reality book[9]	Markers	Markers	Handheld Display	History book
WikiTUI [20]	Camera	Manually match	Projector	All kinds of books
The Haunted Book	Camera	Key feature Points	Animation on screen	Poetry book
Listen Reader	Tag-reader	Embedded Tag	Soundtracks	Children's book

Table 2.1: Conclusion of existing system

Chapter 3

DESIGN

After looking at the existing systems and making comparison between the technologies they used, we can specify our goals here. Our system must have features below:

- Original book style.
- Be readable even without a webcam and AR.
- Easy to use.

To achieve those features, we define our solutions as below:

- Identify pages by nature feature - without marker or merge the markers into the page design.
- Use a popular webcam to "read" the book and user's input.

The design of the system relies on Computer Vision techniques. "Computer vision is the transformation of data from a still or video camera into either a decision or a new representation. All such transformations are done for achieving some particular goal."[5] We can analysis frames captured from the camera and get useful information from them by using computer vision techniques. If we compare our system to a human

body, the webcam would be eye, and computer vision is the thoughts in brain to process what people have seen.

3.1 PAGE RECOGNITION

The first task of our system is to identify the page number of current page in a captured frame. We store one single reference image for each page in the book. When the system runs, start to find the correct reference image to current page. The "find" process is the recognition of this page. After the correct image has been found, return the current page number to the system, so that system can decide what kind of reaction should display.

3.1.1 Requirements

That information should be detected in this stage:

1. Page Location
2. Compare with the reference image, the captured page image has been transform to certain degree. So the next information we need is page's perspective transform matrix.
3. A transformed frame image.
4. Matching result with a reference image
5. Find the correct reference image and return the page number.

3.1.2 Potential Solutions

In many computer vision applications, find correspondences between two images of same object is an important task. Solutions locate this issue include:

- Feature point finding and matching. We will introduce SURF(Speeded Up Robust Features)[3], a novel scale- and rotation- invariant interest point detector and descriptor to our system.
- Template Matching. Template matching is a very simple technique where a subimage is searched for within an image. It is very useful for searching objects in a scene, so it has been used in many recognition applications.

3.2 USER INPUT

The second important problem need to be solved is to define user input of this system. For the main goal of our application is enrich user's reading experience based on user's natural reading habit, In addition, when people read a book, they just use their hand or a pen to interact directly with the book, we define the user input as bare finger.

3.2.1 Requirements

In this part those information should be known:

- Fingertips' position. Since we define the only gesture of our system is pointing, we only need one finger (index finger in most situation)'s position.
- "Tap" action. Once we have got the fingertips position, we also need to determine when the user taps on the book.

3.2.2 Potential Solutions

One way of this task is video-based finger tracking. In some other applications obscuration of fingers by other finger of part of body can be a problem. But in a reading system, this kind of problem is not existing, since we just interested in one pointing finger and the hand's action has be limited in a 2D plane - the desk. To get the position

of the hand, we first need to segment the hand region from the background. Image differencing segmentation can achieve when we assume all object on the desk cannot move. However, that is imposable while reading. You have to turn pages and may move the book during the reading. Even a tap action cause motion on the page. So we need combine the motion result with a skin color region extract to get the hands region. Finally we have to decide the fingertip's position in the hand region. At this stage we assume the webcam has been fixed in front of the user, so that we can define the fingertip's position as the "top" of hand region.

3.3 SYSTEM ARCHITECTURE

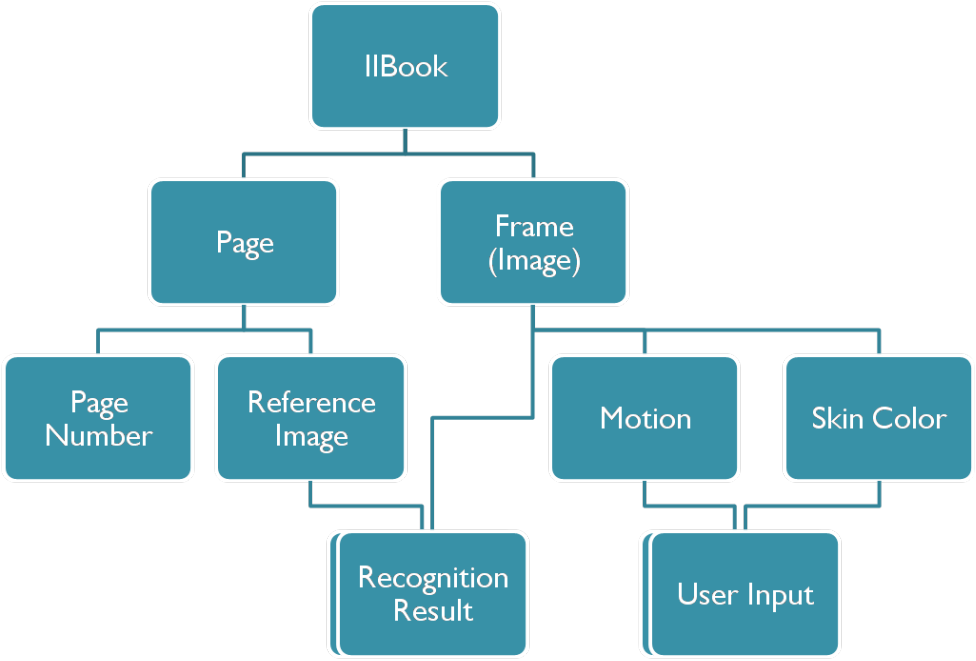


Figure 3.1: The overall system architecture of Intelligent Interactive Book.

To conclude the discussion above, Figure 3.1 shows the overall system architecture. In this system we take page objet and frame object as resources. Though page we can

get each page's page number and reference image. Though frame image, we combine the motion region and skin color to get the user input. Frame image matches with the reference image provide us the page recognition results.

In next two chapters, we will give show all details of how we get page recognition result and user input.

3.4 ABOUT OpenCV

Most of the Implementation in this system using OpenCV library.

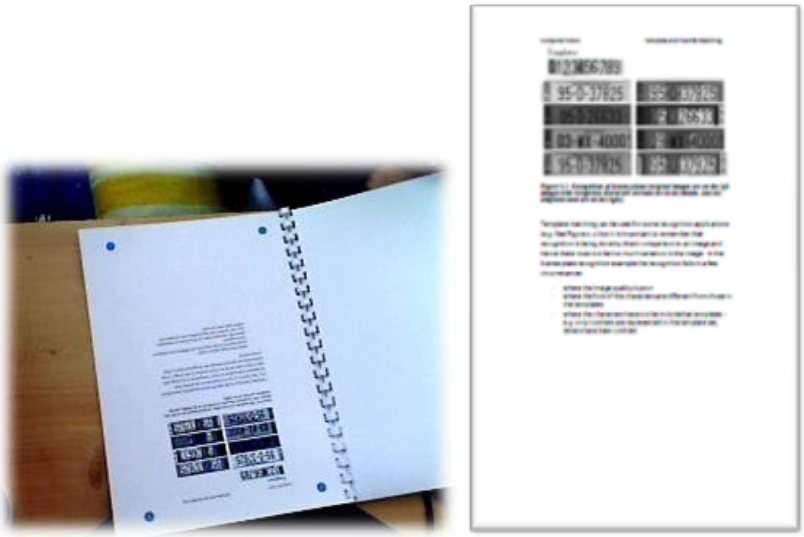
"OpenCV is an open source (see http://opensource.org) computer vision library available from http://SourceForge.net/projects/opencvlibrary." "OpenCV was designed for computational efficiency and with a strong focus on realtime applications."[5]

we can achieve some computer vision algorithm with openCV functions easily, and openCV's simple but powerful GUI can help show the image processing result.

Chapter 4

PAGE RECOGNITION

In this chapter we introduce you the technical details of getting the page recognition result. Figure 4.1a is a frame capture by webcam. Figure 4.1b is a reference image



(a) (b)

Figure 4.1: Input images

of one page. The input images of this task are the last captured frame and reference images of all pages in the book.

As mentioned above, we have two ways within computer vision to find and track objects. The two sections below show the theory of them, and we will choose one to be used in our system after comparison.

4.1 SURF

SURF(Speeded Up Robust Feature)[3] is developed to find correspondences between two images of the same object or scene. It has features blow:

- Fast interest point detection;
- Distinctive interest point description;
- Speeded-up descriptor matching;
- Invariant to common image transformations(Image rotation, Scale changes,Illumination change, Small change in Viewpoint).

The search process can be divided into three main steps:

First, "Interest points" are selected at distinctive location in the image. This location could be: corners, blobs, and T-junctions. If an interest point can be found under different situation and viewpoint, they consider this point's detect is stable. We can consider that point has a "big" feature.

Second, The neighborhood of every interest point is represented by a feature vector. The feature vector is the descriptor of this point, and this descriptor should be distinctive, robust to noise, detection errors and geometric and photometric deformation.

Finally When match two images, the descriptors are matched between them. This matching is often based on a distance between the vectors.

A large variety of feature descriptors had been proposed before SURF developed. But SURF can complete this task better and faster. Then we will introduce the unique features of its detector and descriptor.

4.1.1 Fast-Hessian Detector

Its detector is based on the Hessian matrix[13], but uses a basic approximation which is a basic Laplacian - based detector. Given a point

$$X = (x, y)$$

in a image I, Hessian-based interest point localization in X at scale σ is:

$$H(X, \sigma) = \begin{bmatrix} L_{xx}(X, \sigma) & L_{xy}(X, \sigma) \\ L_{xy}(X, \sigma) & L_{yy}(X, \sigma) \end{bmatrix}$$

where $L_{xx}(X, \sigma)$is the Laplacian of Gaussian of the image. It relies on integral images to reduce the computation time and we therefore call it the Fast-Hessian detector.

4.1.2 SURF Descriptor

SURF descriptor is inspired by SIFT[10] which computes a histogram of local oriented gradients around the interest point and stores the bins in a 128-dimensional vector (8 orientation bins for each of the 44 location bins). There are two steps need to be done in this stage:

First of all , fix a reproducible orientation based on information from a circular region around the interest point.

Then construct a square region aligned to the selected orientation, and extract the SURF descriptor from it.

4.1.3 Matching

After finding all detectors and extracting descriptors, we can match the two images. Fast index the sign of the Laplacian for the underlying interest point is included in this stage. The sign of the Laplacian distinguishes bright blobs on dark backgrounds from the reverse situation. we only compare features if they have the same type of contrast in this stage.

4.1.4 Implementation Using OpenCV

In OpenCV 1.1 version, the SURF function has been added to help our implementation. The Implementation below is using Function 1

Program 1 cvExtractSURF Function.

```
void cvExtractSURF( const CvArr* image, const CvArr* mask,
                    CvSeq** keypoints, CvSeq** descriptors,
                    CvMemStorage* storage, CvSURFParams params );

 CvSURFPoint;
```

This function finds features in the image. For each feature it returns its location, size, orientation and the descriptor. Figure 4.2 shows an example, we illustrate all the feature points out on this image by circles. After finding out all features in page image and frame image, we matching those keypoints up and find pairs of the same points as described above.

4.1.5 Result

Figure 4.3 shows SURF find corresponding features pairs and locate the page successfully. (The page's position in frame image is calculated by a perspective transform matrix which we will introduce in next section.) However, SURF dose not work well in

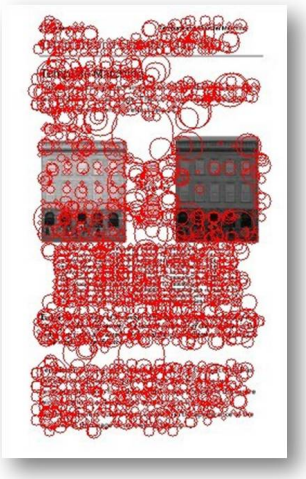
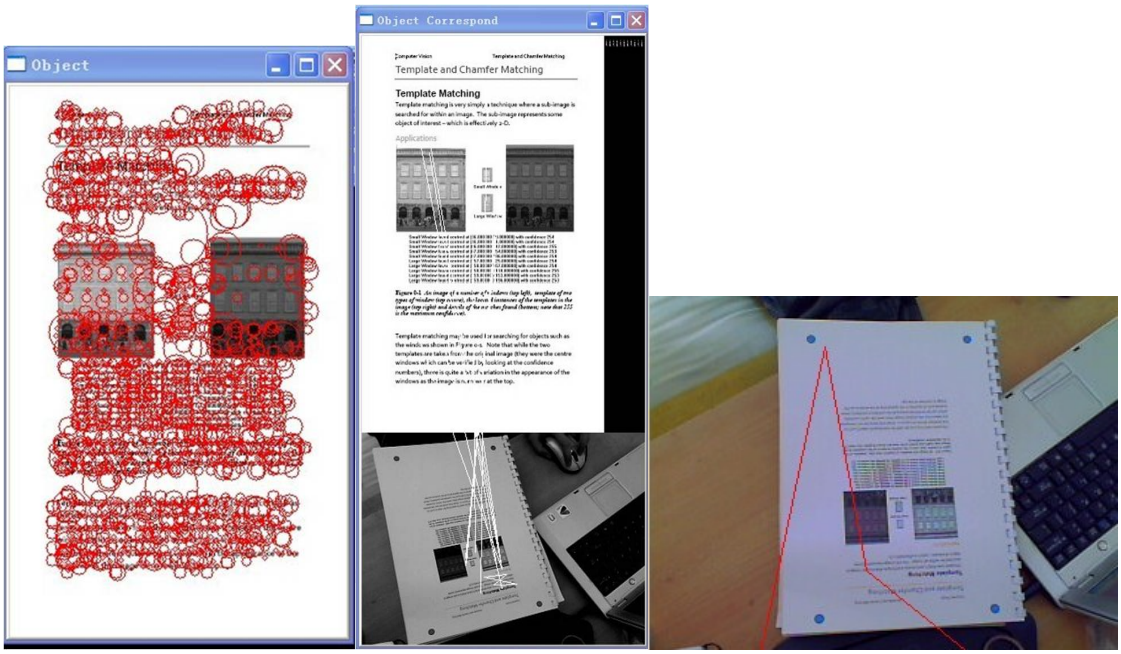


Figure 4.2: A reference image with feature points.

(a) Reference image (b) Matching result and page location

Figure 4.3: A correct sample of SURF.

our application. Figure 4.4 shows a false sample. It shows that the most robust key-points locate in the image region of a page. The corresponding point from frame image mostly in the image of the page. If the image just takes a small region of the page,



(a) Reference image (b) Matching result (c) Incorrect page location

Figure 4.4: An incorrect sample of SURF.

it is really hard to locate the page correctly. What's more, we also use the keypoints pairs to calculate the transform matrix. If we got one incorrect pairs in this stage, the matrix could be totally wrong. We list problems with SURF we found here:

First , It is unstable and not suit for all objects.

Second , It is also very slow. We usually find more than 1000 keypoints in a reference image as well as more than 300 keypoints in a frame. It will take around 360ms to match two images. (In the worst situation the system need to compare N times to find a correct images, N = Total page number of the book). "360ms" may takes a long time if we get hundreds of pages.

Considering of all those issues, we need to try another solution - Template matching.

4.2 HOMOGRAPHY

Before the illustration of our another method of page recognition, I would like to write about homography first. Since this is a very important information for us to locate the page in the frames and determine the reaction to user input.

4.2.1 Definition

In computer vision, we define planar homography as a projective mapping from one plane to another[5]. Figure 4.5 shows a mapping - from the object plane to the image

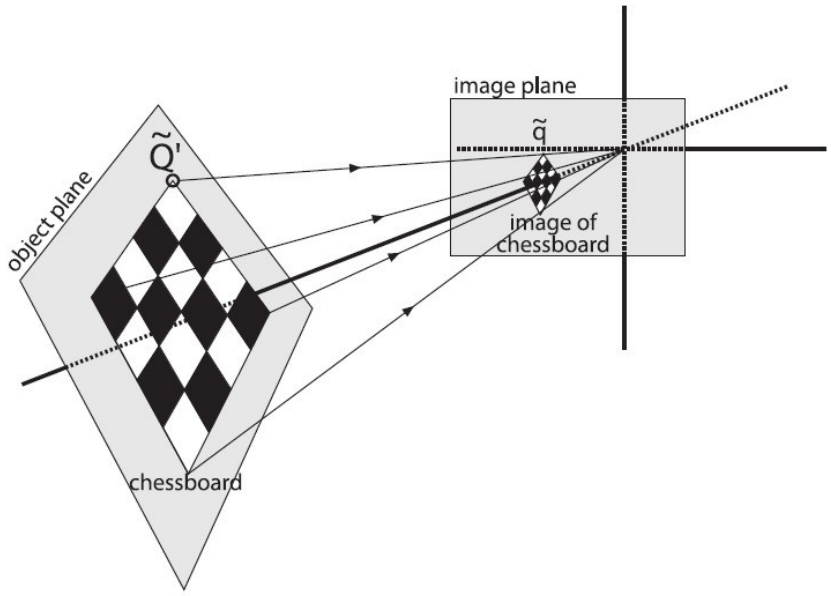


Figure 4.5: View of a planar object as described by homography.

plane that simultaneously comprehends the relative locations of those two planes as well as the camera projection matrix. We can express this mapping in terms of matrix multiplication if we use homogeneous coordinates to express both the viewed point Q

and the point q on the imager to which Q is mapped. We can define, $Q = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$, $q = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, and the transform matrix H.

In addition, without loss of generality, we can choose to define the object plane so that $Z = 0$. and,$Q = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$, $q = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, and $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$ then,

$$q = HQ.$$

The homography matrix H relates the positions of the points on a source image plane to the points on the destination image plane (usually the imager plane) by the following simple equations:

$$p_{dst} = Hp_{src}, p_{src} = H^{-1}p_{dst}$$

4.2.2 OpenCV Functions

Go through the definition of homography matrix, we can see that we can compute H without knowing anything about the camera intrinsics. Actually, OpenCV provides us with a handy function, cvFindHomography(), which takes a list of correspondences and returns the homography matrix that best describes those correspondences. We need a minimum of four points to solve for H, but we can supply many more if we have them. The fact is using more points is beneficial, because we can minimize the noise by this.

The homography matrix derived can be used by cvWarpPerspective(), which can apply perspective transformation to the image.

Program 2 cvFindHomography Function.

```
void cvFindHomography(
        const CvMat* src_points,
        const CvMat* dst_points,
        CvMat* homography
);
```

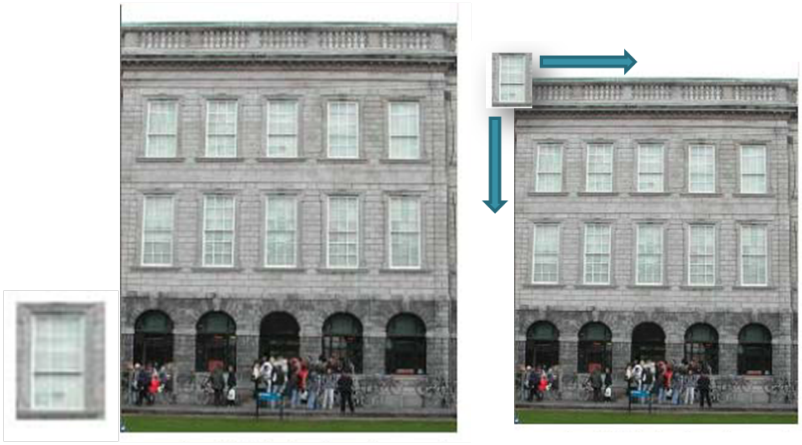
Program 3 cvWarpPerspective Function.

```
void cvWarpPerspective( const CvArr* src, CvArr* dst,
                        const CvMat* map_matrix,
                        int flags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS,
                        CvScalar fillval=cvScalarAll(0) );
```

4.3 TEMPLATE MATCHING

Template matching is very simply a technique where a sub-image is searched for within an image. This is a very useful technology for searching for objects in a scene. And it can be used in recognition applications as well.

4.3.1 Template Matching Algorithm



(a) (b) (c)

Figure 4.6: Template Matching Example

The inputs to template matching are two images - the object image include what we are searching for(Figure 4.6a), and the image in which we want to search(Figure 4.6b). The algorithm sweeps the object image across another image looking for matches(Figure 4.6c).

for every possible position of the object in the image **do**
  Evaluate a match criterion
end for
Search for local maxima (or minima) of the match criterion above some threshold

In OpenCV we can use cvMatchTemplate() to do that.

Program 4 cvMatchTemplate Function.

```
void cvMatchTemplate(const CvArr* image,
                     const CvArr* templ,
                     CvArr* result,
                     int method);
```

The output of matching will be put in the result image, which is a single-channel byte or floating-point image of size ($images.width - objectImage.width + 1, images.height - objectImage.height + 1$) The "matching criterion" has been defined as matching method in that function. There are three different methods we can use in OpenCV (We define I as the input image, T the template, and R the result):

- Square difference matching method (method = CV_TM_SQDIFF)

  These methods match the squared difference, so a perfect match will be 0 and bad matches will be large:

$$R_{sq\_diff}(x,y) = \sum_{x',y'} \left[T(x',y') - I(x+x',y+y')\right]^2$$

- Correlation matching methods (method = CV_TM_CCORR)

  These methods multiplicatively match the template against the image, so a perfect match will be large and bad matches will be small or 0.

$$R_{ccorr}(x,y) = \sum_{x',y'} \left[T(x',y') \cdot I(x+x',y+y')\right]^2$$

- Correlation coefficient matching methods (method = CV_TM_CCOEFF)

  These methods match a template relative to its mean against the image relative to its mean, so a perfect match will be 1 and a perfect mismatch will be -1; a value of 0 simply means that there is no correlation (random alignments).

$$R_{ccoeff}(x,y) = \sum_{x',y'} \left[T'(x',y') \cdot I'(x+x',y+y')\right]^2$$

$$T'(x',y') = T(x',y') - \frac{1}{(w \cdot h)\sum_{x'',y''} T(x'',y'')}$$

$$I'(x+x',y+y') = I'(x+x',y+y') - \frac{1}{(w \cdot h)\sum_{x'',y''} I'(x+x'',y+y'')}$$

In our system we choose Correlation matching methods, since we can't achieve a perfect match. With Correlation matching methods, we can compare the exact matching result, and take the biggest one with its image as our recognition result.

Finally, when using template matching we have to notice that there must not be too much variation in the image. Here is the problem, our input images for template matching can be seen as Figure 4.1. They are quite different from each other. In order to use them in template matching, we have to transform the frame image by to an image which is similar to the reference image. We have to know the transform matrix, and the location of book in a frame. We need markers now.

4.3.2 Markers for Location and Transformation

As we have shown in chapter 2, some existing system uses a matrix code as markers to help identify the book. But those markers break the nature style of a book, and it is unreadable to a reader.

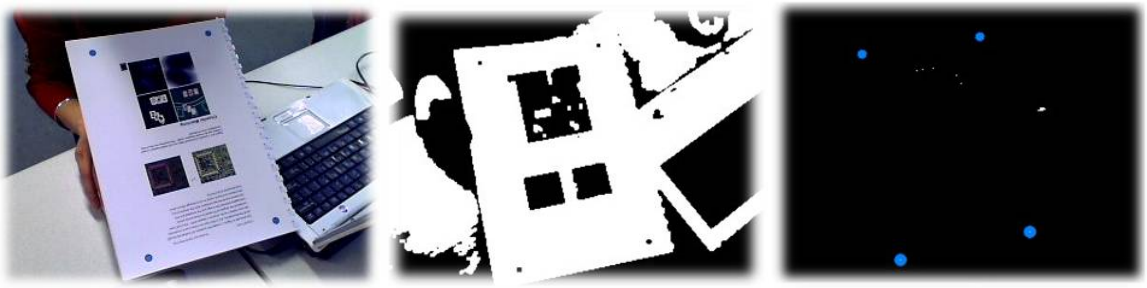
We try to merge our markers into page design and finally develop a mark system as four circles with solid color. Those markers locate on the four corners of book page, to show us those corners' position in frame. We can take those circles as little decorations on the page.

Next step we need to find those markers' position in a frame. We use some basic computer vision techniques to solve that.

A frame image like Figure 4.7a is the input of this stage. Then we do steps below on this image:

1. Convert it to grey-scale.
2. Threshold the grey-scale image to a binary image. This step segments the markers' from white background of the page. The algorithm is quite simple:

   for all pixels (i, j) **do**



(a) Source Image (b) After Threshold and Closing (c) Founded Markers

Figure 4.7: Image processing to find markers.

if $f(i,j) \geq T$ **then**
$g(i,j) = 1$
else
$g(i,j) = 0$
end if
end for

3. Then do closing on the binary image. The purpose of doing this is join objects which close to each other and fills in holes within objects. The result we can get after this step is shown in Figure 4.7b.

4. Find contours in the image. In computer vision, contour is a curve in an image. If we want more information from an edge detected image, we need to extract the contours from edges.

   In our application contours can be found in a threshold created image, in which the edges are implicit as boundaries between positive and negative regions. The results of finding contours is sequences.

5. The next step is to check the circularity of contours. For our markers are all circles, we choose to use cvFitEllipse() to achieve that. This function calculates ellipse that fits best to a set of 2D points. The returning structure of this function include the size and position information of the ellipses.

6. The final step is to confirm if the ellipses color is our defined markers color.

If we can find four ellipses with the markers' color in one frame, we define the markers have been found. Figure 4.7c shows the result of finding markers.

4.3.3 Transformation and Matching

Before we pass the marker's position into the homography equation to calculate the transform matrix, we have to sort the order of the corners to the same order as reference image's. As we can see from Figure 4.8, we define $P_A = (0, 0)$, $P_B = (Page.width, 0)$,

Figure 4.8: Corners's position and order passed to calculate the transform matrix.

$P_c = (0, Page.height)$, $P_D = (Page.width, Page.height)$. We have to find the corresponding points to A, B, C, D. in a frame. To achieve a fast recognition, we assume the camera is in front of the reader, and the reader would not rotate the page by a large degree. The layout of a page in one frame can be classified into three situations shown by Figure 4.9 According to the marker's layout the position features, we can figure out which order we should choose. And pass the point pairs into the equation to get our transform matrix. The matrix derived can be used by cvWarpPerspective(). Figure 4.10 shows the result of perspective transform of a frame, and the corresponding reference image.

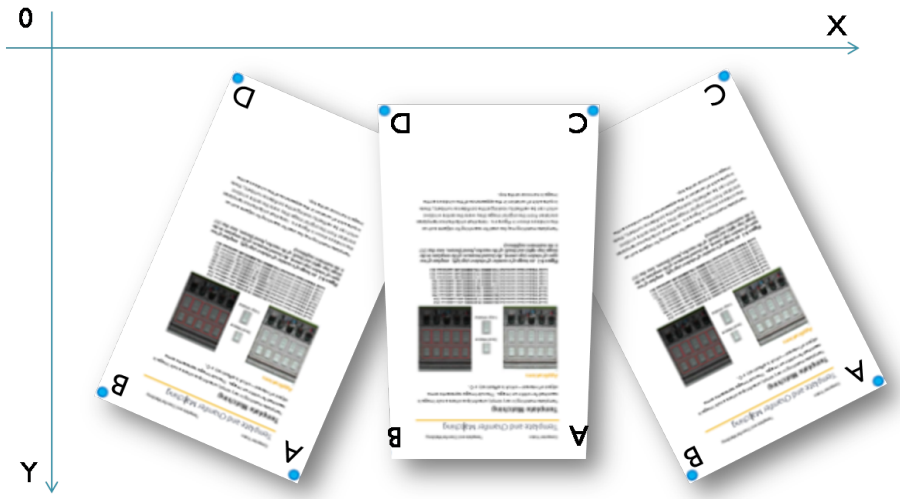


Figure 4.9: Page's layout in a frame.

Figure 4.10: Frame Transformation Result and a reference image.

At run-time, the system matches the transformed image with all the reference images in the database at the same size. There will be only one matching result of each matching. We store that result as the reference image's property. When the all the matchings of one frame image have finished, we search for the largest matching result, and return the corresponding reference image as the result of page recognition. We conclude the matching algorithm here.

if Find four makers in a frame **then**

Pass corner points pairs to calculate the homography matrix

```
    Transform the frame image
    for All reference images do
      Template matching(ref image, frame image)
    end for
    for All reference images do
      Find the largest matching result
    end for
    Result = Largest value's image
  end if
```

With that result, the system can decide what kind of application should be applied on this frame.

4.4 COMPARISON AND CONCLUSION

Figure 4.11 illustrate the differences between Template Matching and SURF. They both have advantages and disadvantages.

We choose to Template Matching in page recognition, for it achieves our goal well. It is fast, stable and does not break the nature reading habit.

	SURF	Template Matching
Average FPS	3	Over 12
Applicable pages	Just works well when pages have a lot of images Doesn't work for pages without picture.	Works well for most of the pages
Page Orientation	All orientation	Pages can't be rotate over 45 degree.*
Illumination condition	Can work in almost all Illumination condition	Color calibration needed.
Viewpoint	Small viewpoint change is acceptable	Very Small viewpoint change is acceptable
*Can be improved by using deferent color to fill the markers.		

Figure 4.11: Template Matching vs. SURF

Chapter 5

INTERACTION

The next problem need to be solved is how to capture the user input and apply related application according to the User Input. As we mentioned before, in this stage we need to find the fingertip's position, which can also be considered as the interaction point. To achieve that, we first find the skin color region in the frame.

Figure 5.1: The input image of finding User Input.

5.1 SKIN COLOR DETECTION

Skin color has proven to be a useful cue for locate and track the hand. In general application, the skin color of hands is different from the background. In our application, the skin color is also different from the background which can be considered as white paper. Despite there may be some figure in the book show the same color as skin, we also use skin color as one cue of hands' location for that situation is rare.

5.1.1 HSV Color Space

We find skin color pixels in HSV (hue, saturation, and value) color space. This color space has been considered more intuitive to use, closer to how an artist actually mixes colors. [8] shows that the HSV color space can segment the skin region from background well. In addition, the RGB color space is sensitive to the changing of illumination condition, so we finally choose HSV color space to help us with extract skin color pixel.[19] We often describe HSV space as color points fill in a cylinder(called a color solid)(Figure

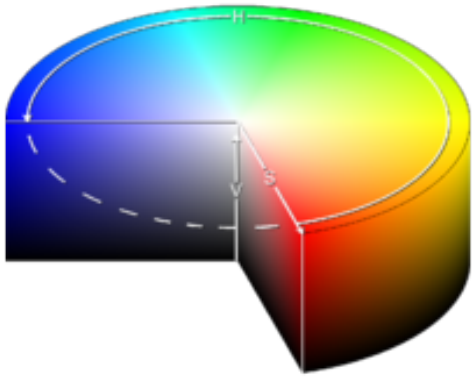


Figure 5.2: HSV arranged as a cylinder..

5.2). This cylinder's central axis ranges from black to white with interpolation between the pure black and the white region. The angle around the axis illustrates "hue", the distance from the axis corresponds to "saturation",and the perimeter of the cylinder corresponds to "value".

In OpenCV, the transformation between RGB and HSV color model is been defined

as:

$$V \leftarrow max(R, G, B)$$

$$S \leftarrow \begin{cases} (V - min(R, G, B))/V & V \neq 0, \\ 0 & otherwise \end{cases}$$

$$H \leftarrow \begin{cases} (G - B) * 60/S & V = R \\ 180 + (B - R) * 60/S & V = G, \\ 240 + (R - G) * 60/S & V = B \end{cases}$$

if $H < 0$ then $H \leftarrow H + 360$

We can use cvCvtColor(const CvArr* src, CvArr* dst, CV_BGR2HSV) to convert our frame to HSV color space. To pick out the skin color pixel, it is like threshold in all the 3 color channel. If one pixel's H, S and V value locates in our skin color range, we turn that pixel to with. To get a good skin color range, we use webcam take some pictures of different people's hands, then pick and record the color information from those pictures. Here is the result, we picked 50 points as a sample.Figure 5.3 According

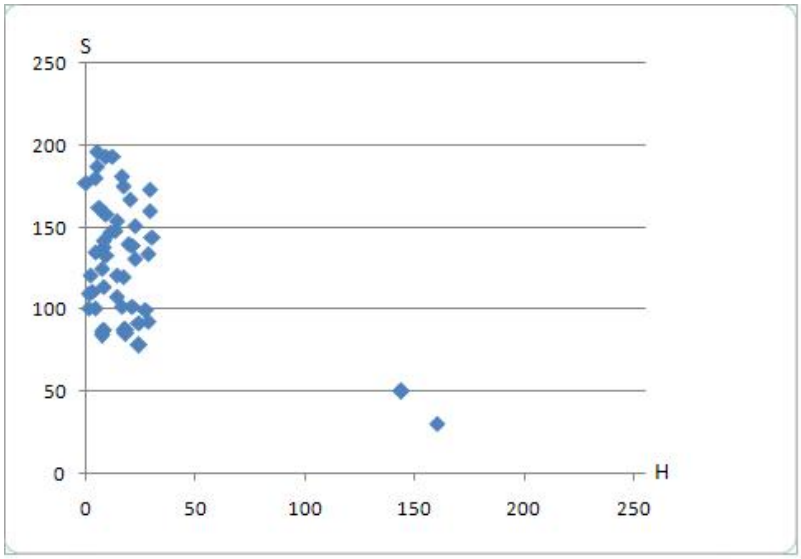


Figure 5.3: Skin color samples.

to the result we set our skin color range as:

$$H \leftarrow (0, 30)$$

$$S \leftarrow (75, 200)$$

With those value and functions, we get a binary image shows the skin region. After that we do an opening operation on it to removes noise, narrow features and smoothes the object boundaries. The result of the skin color can be seen as Figure 5.4. This figure also tell us system consider wooden desk's color as skin color. To avoid that and also avoid taking similar color on the page as skin color, we need more restriction in hand region extraction. We also find markers' color in HSV space using similar

Figure 5.4: Skin color extraction result.

method(Figure 5.5).

Figure 5.5: Marker color extraction result.

5.2 MOTION DETECTION

As skin color may provide us other region whose color looks like skin, we need more restriction in hand detection. We can observe that while reading a page or interact with the page, reader hardly to move the book. Motion can be our another way to get the hand's position, what's more, we also need to analyse motion to capture the "tap" action. Motion detection can help us determine if there is any motion in the image, accordingly localise moving object.

5.2.1 Background Substraction

Assuming the webcam is fixed during the reading, we choose background substraction to detect motion. This is a simple and effective way. To do this, we need two inputs image - current frame $f_1(i,j)$ and a background image $f_2(i,j)$,then we can identify the moving objects the scene:

$$d(i,j) = \begin{cases} 0 & \text{if } |f_1(i,j) - f_2(i,j)| < \varepsilon \\ 1 & \text{otherwise} \end{cases}$$

Figure 5.6: Background image without any moving object.

Figure 5.6 shows the background and Figure5.7 shows the result of background substraction. This result combine with the skin color can provide us a better result. But it is also affected by the motion of the background. Such as shadows of the fingers and slight moving of the pages. So we improve the background substraction by motion template which can also trace hands.

Figure 5.7: From left to right: the current frame captured from webcam; the image after compute the difference between background and current frame; result after connected component operation.

5.2.2 Motion Template

Motion templates were invented in the MIT Media Lab by Bobick and Davis[4]. Motion templates are an effective way to track general movement and are especially applicable to gesture recognition.[5] Using motion templates requires a silhouette of an moving object. From the discussions above we know that the simplest method of obtaining object silhouettes is to employ differencing between successive frames. This will give us the moving edges of objects. Now we got silhouette of moving object as Figure 5.8(A),

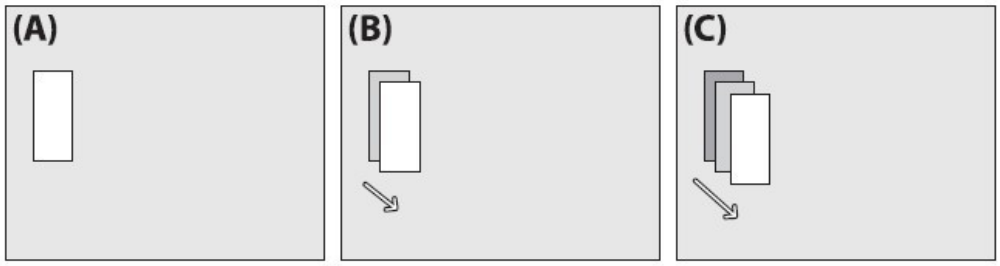
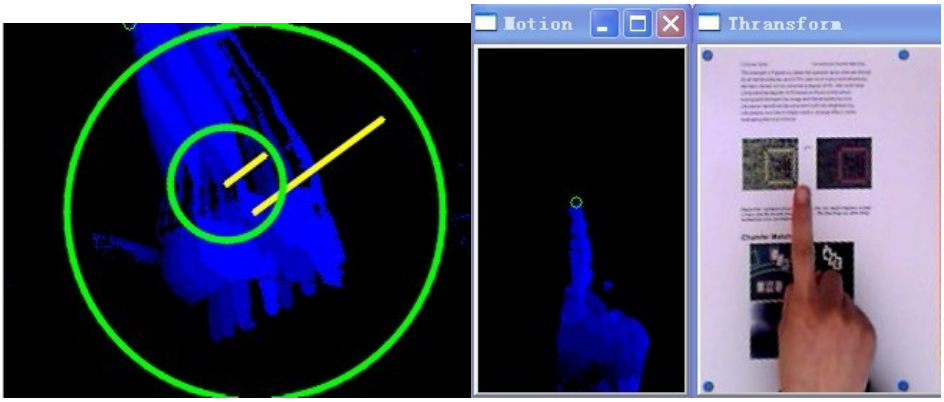


Figure 5.8: Motion Template Diagram.

the white area show the moving object. we can consider Figure 5.8(A) as the next frame. White region is object's current position, and darker one shows same object's position in previous frame. We record object's history positions in a certain time-stamp as Figure 5.8(C). After that we can get a motion gradient similar to the arrow shown in Figure 5.8(C) Taking that gradient as a template, we can identify some gestures. Figure 5.9a shows a vector with magnitude and direction(small circle), and the bigger circle gives the overall direction of all motion is found.



(a) Motion gradient of the moving hand (b) User input result.

Figure 5.9: Input images

We try to use motion template to determine when the user taps on the book. Unfortunately, the tap action is not significant enough for that. So we just use the position of the finger instead. The hand's position can be traced by the combination of skin color region and a motion history image. We also define the interaction point as the top of the hand's region since we assume the camera is fixed in front of reader. To avoid noise outside the page(such as desk's color), we transform the motion image and reset the interest region as the same size of the page. Figure 5.9b illustrate the final result of user input. The green circle shows the interaction point.

Chapter 6

EVALUATION

We develop some small application in a 6 pages vision note to evaluation our system. The first section below shows the result of evaluation.

6.1 RESULT

Basically, we evaluate our system in two aspects. The first one is to test if the system can find all pages reference image correctly and efficiently. The second one is if system can locate user input and show the result correctly.

6.1.1 Page Recognition

We use LOGITECH QuickCam Pro4000 to capture image. Figure 6.1 show 6 frames from the recognition process.(The FPS shown in those images is low because there is a screen record software running at the same time.) The corresponding reference image of each frame shows beside it.

- When we turning the pages, the system locates the page's position and returns the correct reference image.

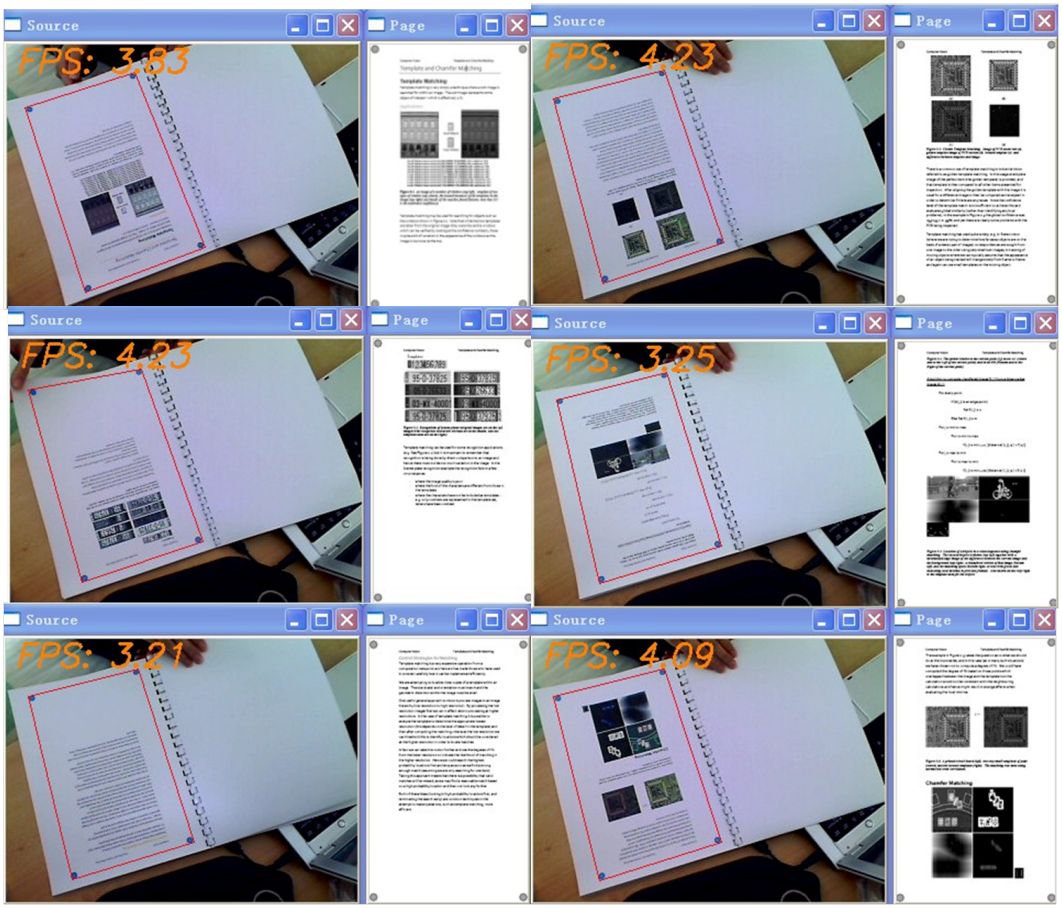


Figure 6.1: Page recognition result of six sample pages.

- We can see recognition works fine in a good illumination conditions.
- Template matching can recognize most of pages in the book, even two pages which have same pictures, or a page without picture.
- Reaction time is acceptable. It takes no more than 300ms to find a correct image in our test.
- The recognition works on slightly deformed pages.
- Incorrect transformation happens when turning the page, but won't affect the interaction result.

6.1.2 User Input and Result display

Figure 6.2 show the result of locating and transforming pointing fingers position. According to the result, our method we can get an accuracy location of the inter-

Figure 6.2: Finger location and interaction result display.

action point. It can identify a small region like the small picture "E" shown in figure 6.2. We also develop some application using our system. In the "Template Matching" chapter of vision system, if reader point to the template matching picture of windows, the system will run a demo in TIPS show how the template matching works. If user pointing on the picture in page two, it will open or close a sample image for user.(Figure 6.3)

At this stage we can say we have achieved our goal of developing this system. The book works fine and can show the correct result to the user under an acceptable illumination condition.

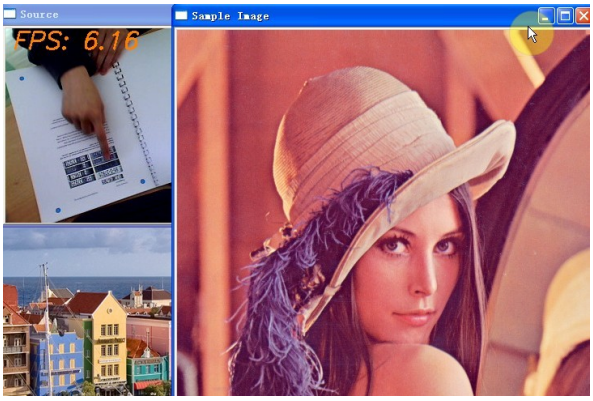


Figure 6.3: Open/Close a sample image by pointing to a picture.

Chapter 7

CONCLUSION

7.1 CONTRIBUTION

In this system, we implement page recognition and user input identification using OpenCV library. Figure 7.1 shows the completed system architecture. The input of the system is frames captured by webcam, and reference images of a book stored in our database. When the system starts we load those page images first. Then image processing on the captured frame. Through this stage, we get markers position; transform the frame image; locate user input; matching the transformed frame with reference images; and display the result of interaction. Through the evaluation we can see our system works well and it do build a bridge between the paper pages and electronic media resources.

At the beginning of the thesis, we briefly introduced the idea of our project. What we want to do and why we want to do that. Then we gave a detailed illustration about what we already have in related area. In addition, we compared existing systems and gave our opinions on that. Those systems inspired us a lot, and we start our design based on them and our dissertation's goal. The main part of the thesis is the illustration of how we identify pages and user input. Finally we show the applications and evaluation of the system.

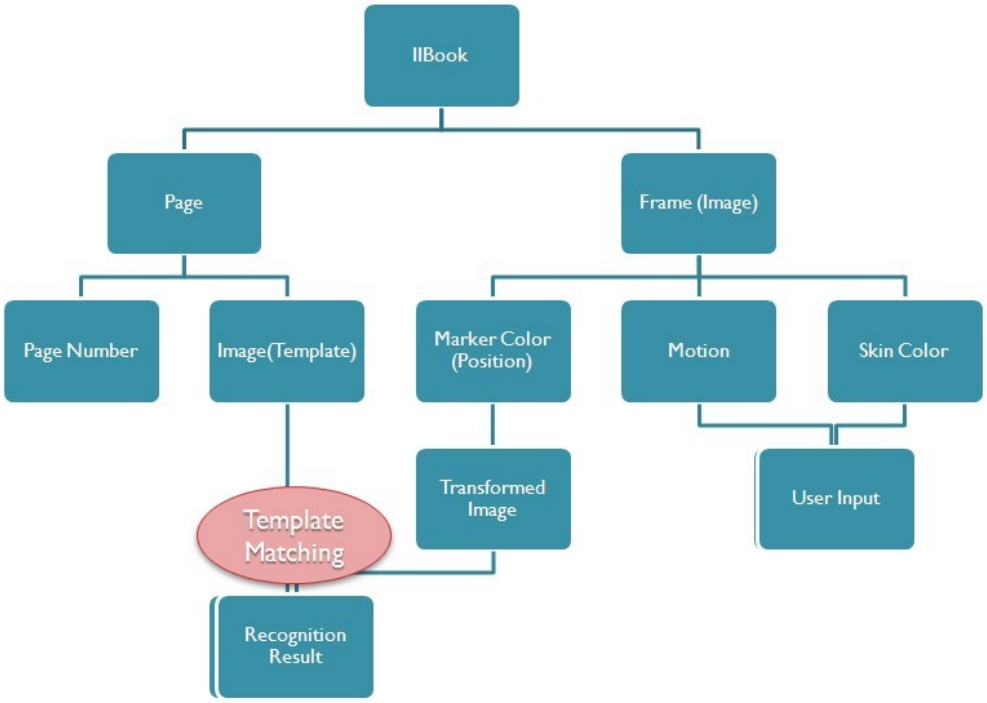


Figure 7.1: The improved system architecture.

7.2 FUTURE WORK

In our system we focus on the main implication issues and solved them. However, there is still some place can be improved to make our system better.

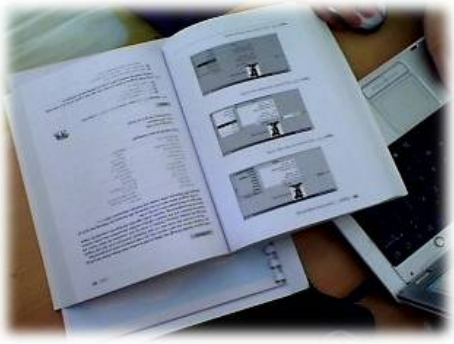


Figure 7.2: A thick book which show deformed pages when open it.

- We can animate the result and display it around the interaction point, such as a bird flying after user's finger. That will be more interesting.
- Project the result beside the book / on the blank of the page. That let the reader get rid of the screen, and make the reading easer.
- We also can develop more gestures, and use a microphone attached on the table to help determining when the user taps. That can enhance the user input.
- For many books which have hundreds of pages often show deformed the pages when we open the book (Figure 7.2). We can design other types of marker to show the curvature of pages. With that our system can identify deformed pages.

Appendix

Class Diagram of the system.

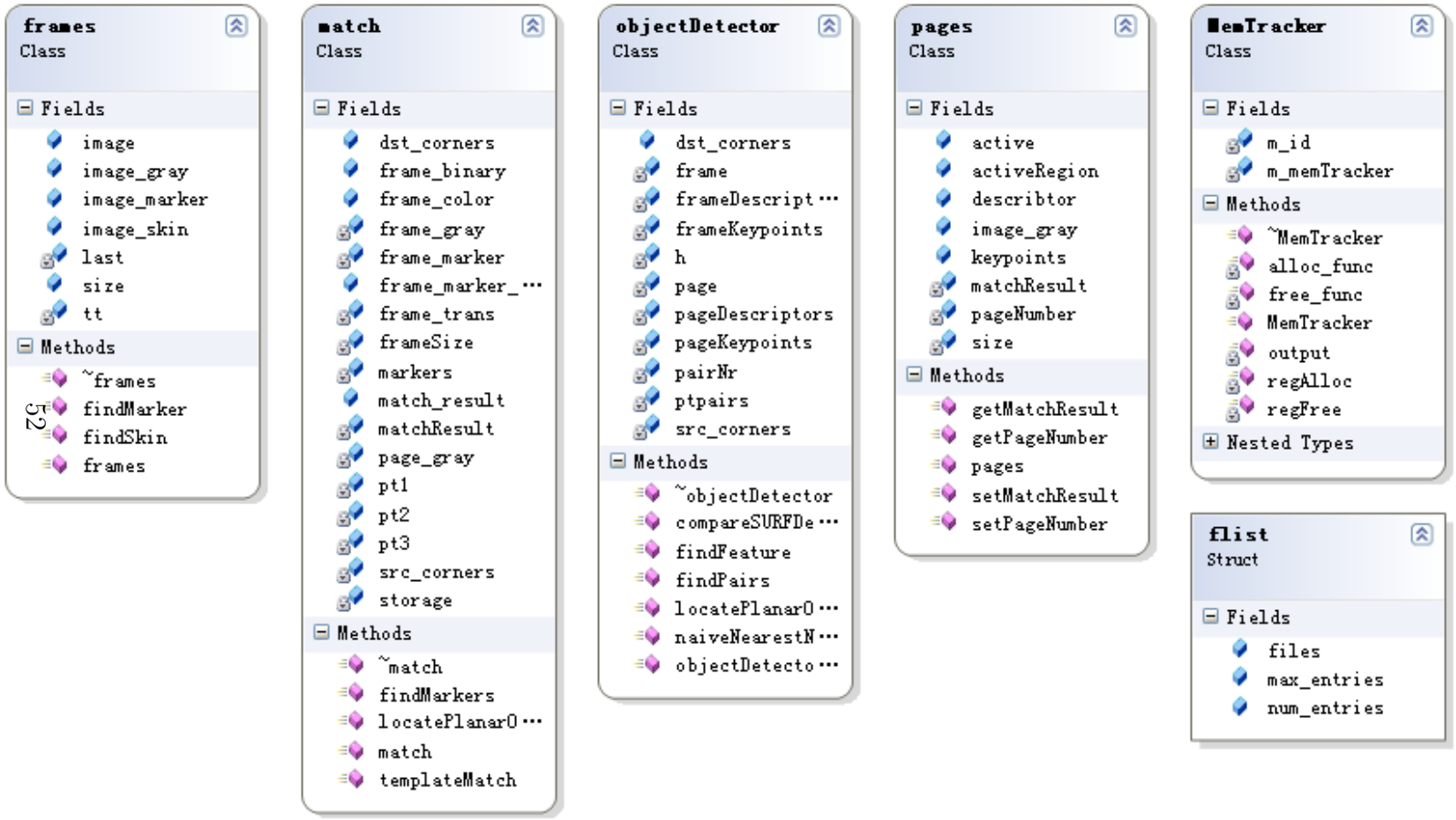


Figure 3: Class Diagram.

Bibliography

[1] Maribeth Back, Jonathan Cohen, Rich Gold, Steve Harrison, and Scott Minneman. Listen reader: an electronically augmented paper-based book. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 23–29, New York, NY, USA, 2001. ACM.

[2] Maribeth J. Back and Jonathan Cohen. Page detection using embedded tags. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 159–160, New York, NY, USA, 2000. ACM.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, 2008.

[4] Aaron F. Bobick, James W. Davis, Ieee Computer Society, and Ieee Computer Society. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:257–267, 2001.

[5] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[6] Son Do-Lenh, Frédéric Kaplan, Akshit Sharma, and Pierre Dillenbourg. Multi-finger interactions with papers on augmented tabletops. In *TEI '09: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 267–274, New York, NY, USA, 2009. ACM.

[7] Jochen Ehnes, Koichi Hirota, and Michitaka Hirose. Projected augmentation - augmented reality using rotatable video projectors. In *ISMAR '04: Proceedings of*

the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, pages 26–35, Washington, DC, USA, 2004. IEEE Computer Society.

[8] Christophe Garcia and Georgios Tziritas. Face detection using quantized skin color regions merging and wavelet packet analysis, 1999.

[9] Raphael Grasset, Andreas Duenser, Hartmut Seichter, and Mark Billinghurst. The mixed reality book: a new multimedia reading experience. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 1953–1958, New York, NY, USA, 2007. ACM.

[10] Yan Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 506–513, 2004.

[11] Hideki Koike, Yoichi Sato, Yoshinori Kobayashi, Hiroaki Tobita, and Motoki Kobayashi. Interactive textbook and interactive venn diagram: natural and intuitive interfaces on augmented desk system. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 121–128, New York, NY, USA, 2000. ACM.

[12] Julien Letessier and François Bérard. Visual tracking of bare fingers for interactive surfaces. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 119–122, New York, NY, USA, 2004. ACM.

[13] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. volume 1, pages 525–531 vol.1, 2001.

[14] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[15] J. Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *APCHI '98: Proceedings of the Third Asian Pacific Computer and Human Interaction*, page 63, Washington, DC, USA, 1998. IEEE Computer Society.

[16] Christian von Hardenberg and François Bérard. Bare-hand human-computer interaction. In *PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*, pages 1–8, New York, NY, USA, 2001. ACM.

[17] Nadir Weibel, Adriana Ispas, Beat Signer, and Moira C. Norrie. Paperproof: a paper-digital proof-editing system. In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 2349–2354, New York, NY, USA, 2008. ACM.

[18] Pierre Wellner. Interacting with paper on the digitaldesk. *Commun. ACM*, 36(7):87–96, 1993.

[19] Wikipedia. Hsl and hsv.

[20] Chih-Sung (Andy) Wu, Susan J. Robinson, and Ali Mazalek. Turning a page on the digital annotation of physical books. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 109–116, New York, NY, USA, 2008. ACM.