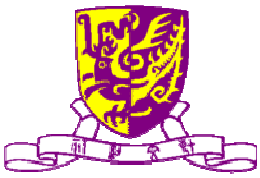


香港中文大學

計算機科學與工程學系

Department of Computer Science and Engineering,

The Chinese University of Hong Kong

Research on Tablet
Tablet Apps for Supporting Research Activity

Li Wai Lun (1009615163)

Supervised By

Prof. LYU Rung Tsong Michael

Abstract: To use the tablet devices for supporting Research Activities, we may need to install a number of apps to perform various tasks, such as paper/reference library searching, PDF reader & annotation tool, file sharing, and group communication client. In this project, we would like to develop an application to help in supporting research activities in CSE Department. The final product would include the following three features: paper or reference management, research activity log and communication. A server is required to be used as paper/reference repository, research activity synchronization and comments referencing.

Table of Contents

1. Introduction

1.1 Motivation

Tablet devices gain greater and greater importance around the world (Figure 1.1). The portable design attracts users for reading information and watching videos. Although there are still limitations of tablet devices and it is not possible to replace the traditional personal computer, the tablets can be further utilized in many aspects.

Figure 1.1 Tablets PC

Seeing the advantages of the portability and the ease of reading books (Figure 1.2), the tablets should be promoted to perform academic objectives. We would like to introduce it to research activities. However, the design of tablet aims at the simplicity. As mentioned previously, the tablet is different from personal computer. Too complex tasks should not be implemented. So what can be done without a substantial keyboard? Yes, the task should be performed without large amount of input data. We noticed that the large screen of tablet eases the difficulties of reading books. Papers are important components of research activity, and that is what can be easily done without a substantial keyboard.

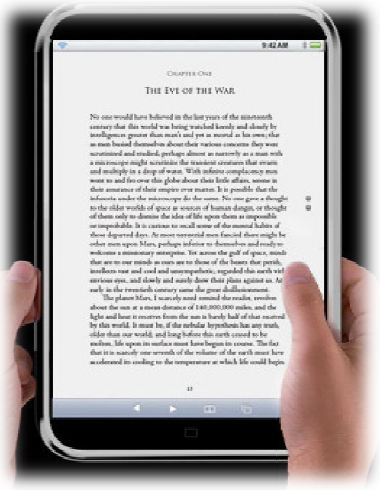


Figure 1.2 Using tablet to reading e-book

Purely reading paper seems an easy job. It seems far easier than e-books, because e-books may not available on the internet while huge resources of paper are easily "Googled" (since there is "Google scholar" http://scholar.google.com) on the web (Figure 1.3). Additionally, PDF reader is common installed in tablets.

Figure 1.3 Google Scholar

Figure 1.4 Paper in hard copy

Nevertheless, what we hope to do is not only giving a PDF reader to read paper. We would like to build up on the existing resources, and develop a tailor-made application for users to view paper easily. For example, are there ways to help us to view the paper easier? Paper seems like ordinary printings (Figure 1.4) with long long pages of words, but they are indeed very different. References are especially for academic papers. Would the application improve the ease of viewing the reference? And though it is mainly used for viewing paper, we believe that more stuff can be completed with the assistance of the tablet. Extra featured are expected to achieve.

We hope that the product constructed can relieve much difficulties of viewing paper and help in research activity. We envisage this product can truly reach the objectives with the rapid popularization of the tablets,

1.2 Objective

Our final project is hoped to develop a tablet application to fulfill the following four objectives.

- Improve efficiency
- Exchange ideas easily
- Make reading paper simply
- Minimize workload of managing resources

1.2.1. Improve efficiency

Obviously, the tablet devices aim to help saving much more time. We can treasure our time by doing vivid tasks anytime and anywhere – and what the users focusing is not just “what they can do”, but “how they can complete their tasks simply and efficiently”. For the word “simple“, we would like to define the word as doing much more things with minimal work done on it. We try to achieve this “simple” which has been analyzed in the last paragraph.

So what is “efficient” mean? To make it more materialized, we make it related to time consuming. Time is definitely our major concern, and finish the tasks in a simple way is one way to increase the efficiency. Besides, efficiency requires strong supports behind simplicity. Strong, powerful and well-designed database and libraries can greatly devote to the enhancement of the efficiency, which let you locate what you need within the minimal time spend.

In fact, efficiently cannot be easily measured. Comparing to the effectiveness, we may quantify some targets to see whether it can be completed fully or partially, or in how much percentage it completed, while efficiency is heavily depends on many different tiny stuffs. We hope to achieve this by having research on the methodology of reading papers and design our system carefully.

1.2.2. Exchange ideas easily

Research papers, to a certain extent, are the concepts of spreading the themes and theories – or we name it "ideas" – around the world. Research activity involves a great scale of exchanging ideas in principle, in order to invoke much more new inventions. Sharing papers are what we can commonly do nowadays, yet would there be rooms to improve it? Can the communication be further enhanced? If yes, then how?

Figure 1.5 Exchange ideas between different people

Building upon reading research papers, we may add features to further boost the circulation of ideas. We should have experienced such kind of situation that we would have some questions or challenges to raise, or we would have our own ideas or different point of views concerning certain point described about the topic (Figure 1.5). These should all be considered as academic sharing of ideas and they may stimulate us to rethink deeply. Sparks invoked may contribute to new ideas too.

1.2.3. Make reading paper simply

Reading paper is quite a simple task. However, it is sometimes quite annoying – especially for reading papers. Let's think of the situation of flipping a hardcopy of a paper. Browsing the pages one by one, looking at the dense lines of words and several pieces of images and diagrams, everything runs smoothly. However, some not special but common nagging digits start to annoy us quickly and frequently. The digits within the [ ] bracket show its prevalence in the whole passage, if you would like to know what is that means, you should flip to the last page to have a note – usually you know nothing because the line tells you a string of unknown article names and strange names of researcher – to have a look. Repeat the procedure for 10 times or above – see? Yes, it is annoying!

The above case seldom appears in daily life indeed, since we are not common to refer to reference in daily reading. Nonetheless, the reference weighs heavily in the scholars' writing. Tones of unknown phrases and findings may be shown and these are all valuable to be referred.

Of course, the problem is easily solved using the PC by opening the file twice. The problem can also be addressed on the hardcopy by tearing the pages out. But we would like to make everything simpler, to view the reference or annotation effortlessly by just one click. If the tool is powerful enough, we expect the tool can directly show us the referred articles too. All these helps make things simpler.

1.2.4. Minimize workload of managing resources

Information is flooding, especially on the Internet (Figure 1.6). Millions and millions of information floods when we look for a certain topic. Perhaps it helps us to include as much as information as can, while we often spend quite a lot time for filtering and locating what we actually need. Duplication of information always occurs, which prevents us from extracting the pieces we want.

Figure 1.6 Internet contains lots of information

Figure 1.7 Green Computing

To avoid this, good management may address the problem. If the system is clever and mature enough to manage paper neatly and avoid the repeats, workload of researchers can be minimized. Much more time can be spent to concentrate on the research topic. All these help are more than just saving time of the researchers and scholars – these also save the resources of our computer to achieve green computing in some sense (Figure 1.7).

1.3 Project Overview

To achieve our objectives, we should implement our project steps by steps. This report will discuss each part clearly, and here will give a whole picture of our project.

1.3.1 Research on relevant topics

Chapter 2.2 gives some analysis of our survey which has been conducted before starting our project with coding. Following are the area we would like to study:

A) Format of research paper: what can be shown to user?

Understanding the skeleton of papers makes the filter of information run smoothly. Some parts are essential to identify the uniqueness of the paper and building index of the paper library, just like the title and authors of a paper. But would there be more information can be easily recognized by the server, so as to improve the accuracy of storing paper or showing useful information to the users?

B) Occurrence of the phrases: how to make the paper readable for computer?

Besides understanding the paper format, the phrases used by the papers are important for teaching the system too. Keep in mind that papers are designed to be read by human. Languages of people, unlike the languages of computers, can vary within a certain range. Different wordings (even with "syntax errors") may tell the same ideas. Analysis of the occurrence of some key phrases improves the accuracy for computer to locate the data.

C) Format of PDF files: what PDF does?

It is not surprising that most papers are saved as PDF format (Figure 1.8). Printing as PDF prevents it from editing or mix-up due to careless mistakes and make it portable to various computer working environment and operating system. Of course, we are not trying to analyze "why PDF is used", but "what PDF does". Format of a paper is readable for human beings, while PDF format is known by computer. Studying more and getting familiarized to this format would lead us to better development of the application and the buildup of server.

Figure 1.8 A PDF file

D) Tools help to build the app: any API available?

Building an application on the tablet devices is not easy. Android platform, to a certain extent, is quite a new platform to us. It is quite hard for us to start to learn to write a PDF viewer from knowing nothing. We would like to find out if there is any APIs can help us to build our application smoothly. After all, learn to build a PDF viewer is not our ultimate goal. We would like to save time so as to run more towards our goals.

E) ***More features can be utilized: how to achieve more?***

Not just referencing is significant; we should investigate more characteristics to enrich the application and accomplish our aims. For example, are there any communication features have been implemented in the tablet that can be made good use of to facilitate the exchange of ideas? Or are there any open source software can be imported to help the development runs smoothly? Just have a research to find out the answers.

F) Methods to locate targets: any searching algorithm helps?

Searching activities are so common that it seems not a worth talking part in the whole system. Various online search engines are common, highly accessible and easily adoptable. However, papers are quite different from the other web resources online. Each paper, obviously, contains few more characteristics, like official keywords noted by the authors. We therefore wish to tailor-make a search algorithm for our system to improve the effectiveness of the searching activities within the paper repository.

G) Linkage between papers: how papers relate to each other?

Referring to the accurate paper give great convenience as mentioned before. We would like to analysis the relationship between papers to enhance automatic reference parsing. To achieve this, we probably focus on how the format of the reference items varies. For example, does the reference item contain authors' name first? or the title of the paper first? Or is the name of the journal or the conference included? All these help to build up a creditable linkage and even help to search the target missing paper online.

H) Duplication checking: what makes each paper distinct?

After all, computers are not intelligent enough to read though the while file and check the uniqueness among all files. With different file name and metadata of the file, the files will be treated as distinct, although the content may exactly the same. Hence, we need to see what makes each paper different from the others, like title, authors and year released, etc.

1.3.2 Design of the whole system

Chapter 3 concludes the overall design of our system. Our system includes server side and client side development, that both sides heavy workload to handle.

For the server side, it has been developed using PHP web server language. The server side aims to do every parsing and analysis on the PDF files and manage the database. PHP is chosen because PHP has strong libraries for reading the PDF files. This make the analysis on the parsing paper runs smoothly.

For the client side, our application has been implemented on the Android platform. Originally it is expected to be implemented on the iOS (Figure 1.9). Yet, due to the limitation of hardware resource that we do not have iOS development platform, we give up the iOS and switch to Android platform (Figure 1.10). The Android application is built by Java language (Figure 1.11) and the development is comparatively portable to the application runs on iOS. Reiterate that our focus for once – we would like to make research activity simpler on tablet devices but not on MACs only.

Figure 1.9 iOS5

Figure 1.10 Android

Figure 1.11 Android

To ensure the progress of the project, the project is roughly divided into two parts. One student focuses on client side and the other student focuses on the server side.

There three main modules to achieve our goals:

1) Paper management
2) Communication
3) Research Log

Further explanations on design and implementation are included in Chapter 3 and 4.

1.3.3 Division of works into two phases

To make the tasks clearly, we divided the works into two phases. We have tried to conduct a comprehensive survey and implement some key features of the applications during the first phase (Figure 1.12). In the second phase, we would aim at increase the accuracy of the application and make fine tunes to improve the efficiency (Figure 1.13).

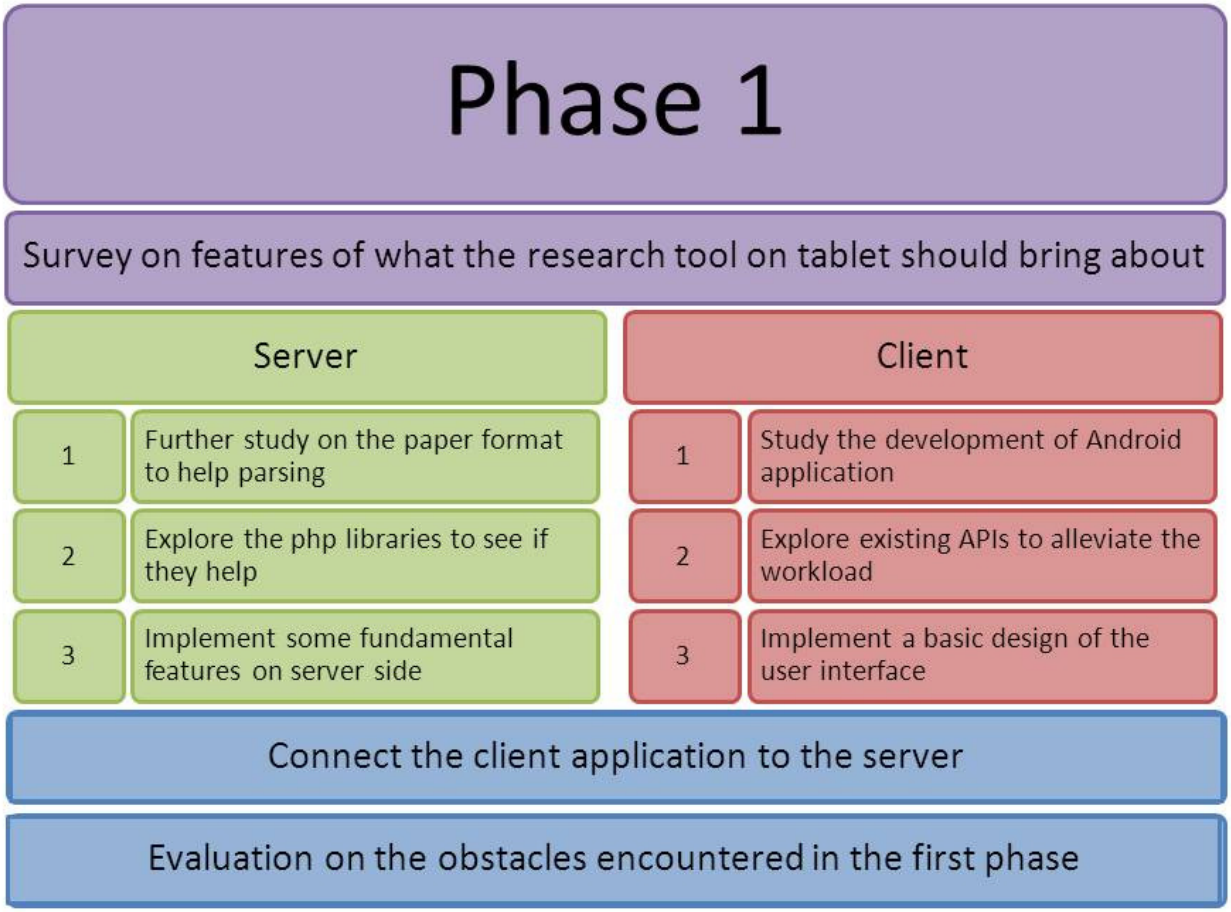


Figure 1.12 Works in phase one

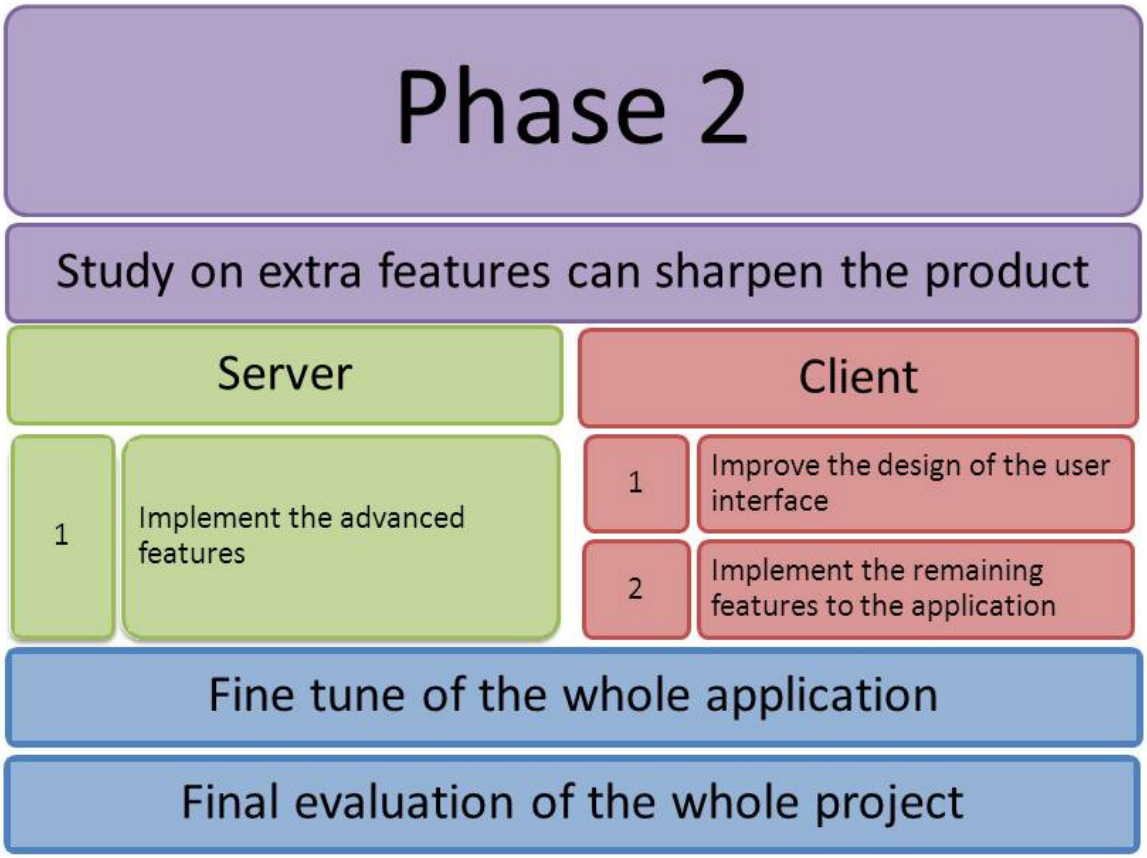


Figure 1.13 Works in phase two

2. Background

2.1. Raising population of tablet device

When it comes to tablet devices, they are seldom used in academic aims. Chasing back to the time when Steve Jobs presented his "truly magical product" – iPad [1], he defined it with many functions: "browsing the Web, sending e-mail, sharing photos, watching videos, listening to music, playing games and reading e-Books". Since this most popular tablet in the world released, we almost found most of users treat it just as a smartphone with a larger screen. Surely most of features of the tablets, not matter iPad or Android tablets, can be found on the iPhones or Android smartphones. The main differences between the tablets and smartphones are the calling and SMS functions and the screen size (Figure 2.1).

Figure 2.1 Screen Size difference between smartphones and tablets

However, we thought that these portable devices with a larger screen size should not be treated like an alternatives of smartphone. They are truly different, and we should fully utilize the differences. Once we noticed what Steve Jobs presented, one point should be focused – the reading e-Books function of the iPad (Figure 2.2). As one of experienced smartphone users, this point woke us up. Among the abovementioned features, we are comforted to perform most of them. We may browse the web for dealing with some urgent stuff or finding emergent information, yet we seldom use the phone to read books. Of course, reading books is possible on phone devices, while the small size screen is not preferable compared with the tablet. And this is what we looking for – the absolute difference between tablet devices and the small touchscreen gadget. On recognizing the difference, we decided to develop a tool to make good use of it.

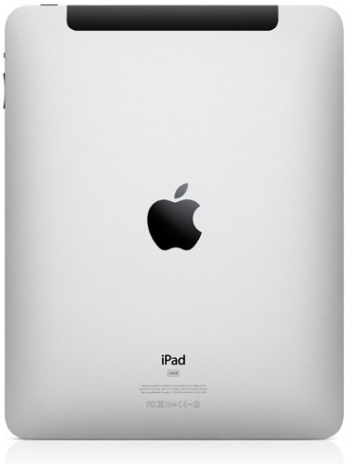


Figure 2.2 iPad

2.2. Lack of academic application for tablets

We would like to introduce the tablet devices to academic aspects. Take a look at the Google Play (Figure 2.3). Click on the category tab of “Books and References” – the one seems related to academics research – and check if there is anything helps in research or paper referencing. However, we would get a list of e-books or e-book readers. Even we click onto the tab of “education”, we also get similar results. How about we type “research” and some relevant terms? Most of the results show application related to business or stocks. One result of “research paper” catches our eyes – well, the app “Writing Your Research Paper” seems just an E-book, not an app help me to writing the paper (Figure 2.3).

The case may not such extreme indeed. If we search "research Android App" on Google, meaningful results can still be found. However, there are still few meaningful results. The two applications we found is just like a database and libraries for us to search for materials [2, 3]. There is lack of strong and useful applications for supporting activity.

Figure 2.3 Google Play is lack of academic application

2.3. Reading paper takes the advantage of tablets

As mentioned in the motivation part, complex task should be avoided violating the original concept of the tablet design. This tells reasonably that nowadays the application available only provide the function of searching papers or research resources for users.

Taking of advantages that tablet can read books and keeping the origin of tablet (Figure 2.4), we would like to develop an application related to reading research paper. Reading paper seems quite a simple task. It seems far simple than e-books. Books may not be available on Internet, while there are great resources of paper on the web.

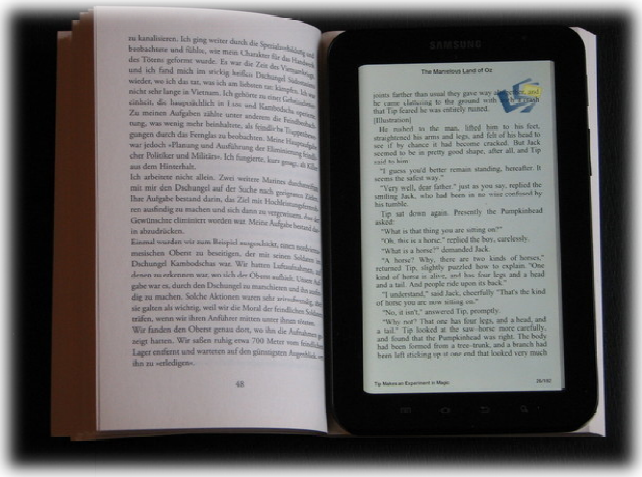


Figure 2.4 The large screen of tablet is good for reading books and papers

Yet when we read papers, no matter the hardcopy or softcopy on the tablet, we find it annoying to clip the pages over and over, to grab the reference and chase where the ideas come from. The clipping maybe not the great problem, but the referencing can surely be done better and better. Somehow, just clipping to the last page we just know more about a list of strange titles and authors. What we want is the paper! And that is what we looking for – a paper reader with automatically referenced and linked to others.

2.4. More features beyond reading

If a platform is designed for reading research paper, then could it be further extend to a research tool? Quite more than reading paper and required little data input. We would like to improve and sharpen the product in order to realize a helpful research tools that truly cater the needs of researchers. This goal prompts us to think more, that will be further clarified in the research part and the design part.

3. Research on relevant topics

Research tools are not rare in personal computer. What we are now seeking for is not just software of a research tool, but an application specifically catering the aim of reading paper and using the tablet devices. Yet, Scholars' writings, though have been quite restricted in the writing formats, vary much because they are written by human. To make the paper more manageable, we should first start to see the difference and tell the computer what parts to look at. There are four main topics we have conducted relevant surveys.

3.1. Format of research paper: what can be shown to user?

To start with, we should clearly define what the research paper is. Somehow our product may not powerful enough to recognize all PDF files without a standard. Quoted from Wikipedia: "Research paper, (also called scholarly paper), which is published in academic journals and contains original research results or reviews existing results" [4]. Under such definition, we try to restrict the paper format fulfill the rules of the publication of the academic journals.

Firstly, generalizing on the ideas of the some reference page of the University of California [5] and the Rice University [6] from USA, the research paper have to include the title, author(s) (and maybe the contact information), the abstract and the references. Of course there are introduction and contents, but abstract is enough for achieving our goals.

Second, we conduct a rough survey on a pool of research papers. The pool of papers contains 169 PDF files and 149 of them fulfill the definition of a scholarly paper (Figure 3.1). We analyze the position of title and author, the appearance of abstract and keywords (Figure 3.2), and how the reference show. Nearly 95% of the papers have an abstract, and 90% of them start with the word “Abstract”. The keywords appear in 50% of the papers. Almost 95% of the references indexed by the [ ], while few of them using other format or with no index (just show a list of references). Some of research papers consist of unstandardized format, such as author goes before title, or in the form of report. 15% of them start with a cover page which is not a standard.

To conclude, over 80% of the scholar papers in the research pool follow the format that includes title, authors, abstract, introduction, content and references (Figure 3.3). Though it may not be the real statistics of all the papers available on the web, it is worth to have a try in parsing papers in the format investigated.

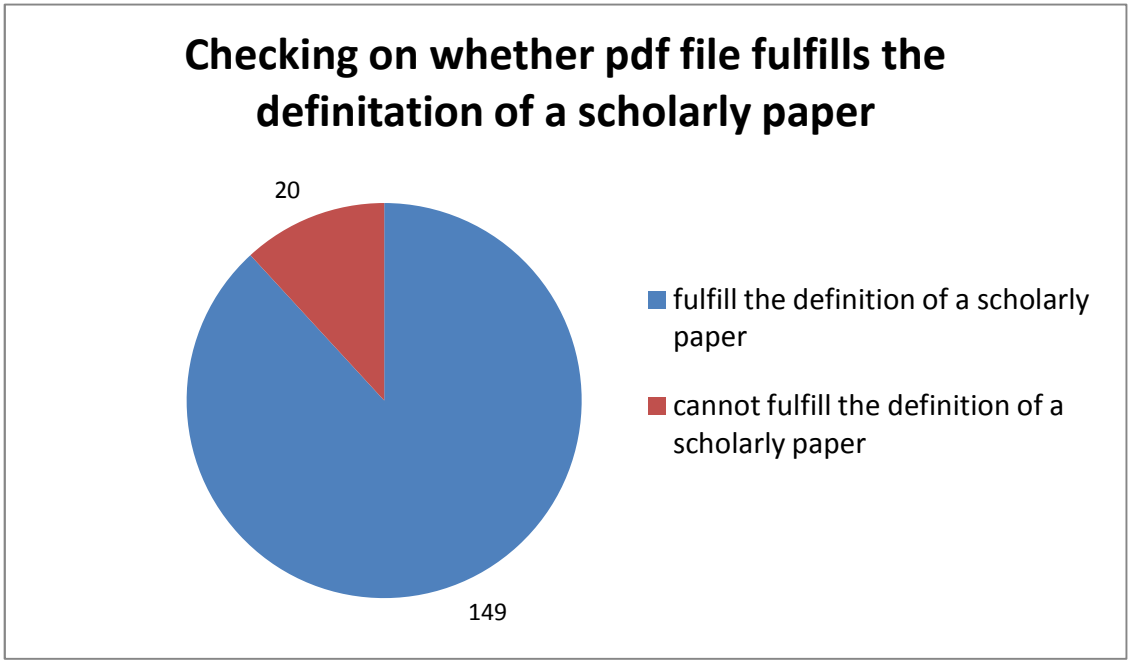


Figure 3.1 Chart of checking on whether pdf file fulfills the definition of a scholarly paper

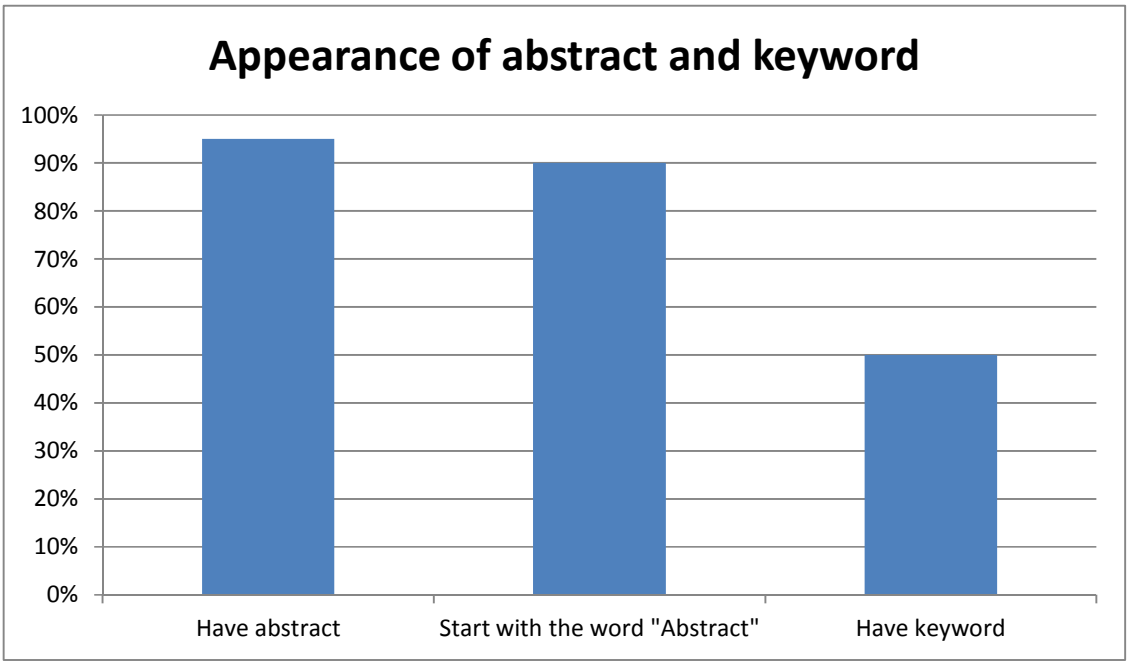


Figure 3.2 Chart of appearance of abstract and keyword

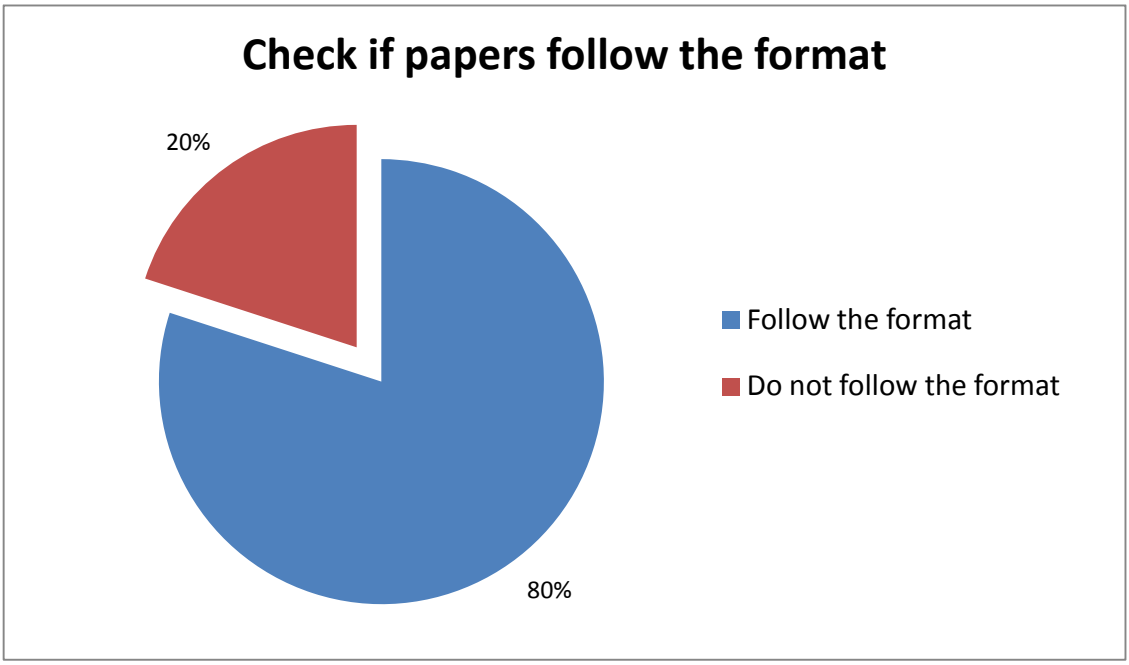


Figure 3.3 Chart of checking if papers follow the format that includes title, authors, abstract, introduction, content and references

3.2. Occurrence of the phrases: how to make the paper readable for computer?

To study how to parsing the paper, we analyze the phrase occurred in the paper which may make the paper more understandable to the computer.

Title

95% of the papers do not have the “title” word to indicate the title of the paper. Usually the font size of the title is the largest word appeared in the whole file. Few of the titles appear after the author name. Titles may not the first line of the paper, because there may be headers of the page or some further information.

Author

There are a huge number of papers which are written or collaborated by more than two authors. Besides the name of authors, more than a half of papers we surveyed contain a line of information and contact information about the authors. For those do not show author information at the top of the paper, the information may be added at the end of the paper (after the appearance of references).

Keywords

Presence of keyword is not a necessary part of a standard scholarly paper. Yet, after consideration, we would like to check if there is any presence of the keywords column. If there is, the keywords will be recorded in our database to assist the searching of paper. The keywords usually appear with the presence of different noticing words like “keywords”, “index terms” or “general terms”.

Abstract

More than 80% of abstract is written following the presence of the "Abstract" word. Some of them are written directly after the name / the contact information of the authors. When the abstract ends, introduction or keywords will be presented.

References

Nearly all the papers have the "References" or "Bibliography", while there is a small variation between paper and papers. Statistic shows 90% of references have shown in the index form [ ]. The rest shows without [ ] but numbers only, and some of them without indexing. Those lists of references without indexing will not be considered in this project because of the meaninglessness.

3.3. Format of PDF files: what PDF does?

According to Wikipedia: "Portable Document Format (PDF) is an open standard for document exchange. This file format, created by Adobe Systems in 1993, is used for representing documents in a manner independent of application software, hardware, and operating systems."[7]

Structure of PDF files

The PDF files consists of eight primitive objects including Boolean values, numbers, strings, names, arrays, dictionaries, streams, and the null objects[8]. PDF contains everything as objects, to standardize different data printed by various files originally. Take text as one of the examples. Adobe creates an element for the text and makes it as stream [7]. The primitive type "streams" is used for storing a large amount of data. The element records not only the text wordings, but also the font size, the font family, etc. PDF also records the layers of each object.

Metadata

There is one more significant feature of PDF files. There is metadata of PDF files. The metadata records much key fields of the title, author and update dates. The metadata seems to be powerful, which could help us a lot in completing or goals. PDFmeat [9] is a powerful open-source tool to acquire the metadata of PDF Files and we have done a research on the metadata using the PDFmeat. Though there are few errors, over 70% of papers are correctly located.

PDFmeat

Figure 3.4 Icon of PDFmeat

PDFmeat (Figure 3.4) stands for PDF Metadata Acquisition Tool [9]. The tool provides great support for analyzing the metadata in PDF header and managing the PDF files. It aims at convert the metadata to BibTex. To run the program, PDFmeat provides a user-friendly GUI which embedded in the Firefox browser. On click it will generate BibTex data to catch the metadata of PDF file. However, the GUI is not feasible for realize our aim. We would like to have the CLI program to run and catch data. We have done many searching and refer to the official page for many times but we still can access the PDFmeat successfully. Unless we can achieve it successfully, we are impossible to use the metadata.

PHP Library - PDFlib

There were many share libraries available for PHP, and some of them related to PDF files. Nonetheless, most of them are aimed to create PDF files rather than parsing / reading the PDF files. Filtering those useless, we focus on the PDFlib.[10] (Figure 3.5). The PDFlib get various value like font size, position in X and Y, page numbers etc. [11] All these features shall be taken at note for implementation.

Figure 3.5 Icon of PDFlib

Xpdf – pdftotext

Though PDFlib can read the PDF files, most of the functions build-in are expected to be used in creating or editing PDF files. Like the functions in PDFlib, these is no any function can directly extract the text of the existing PDF. Parsing PDF text may be not relied on the PHP library. Nonetheless, there are other open-source programs help to extract the text in the PDF, like PDFBox [12] and Xpdf [13] (Figure 3.6). We would first implement Xpdf for experiment.

Figure 3.6 Icon of Xpdf

3.4. Tools help to build the app: Any API available?

To alleviate our workload, we choose to use the API of PDF viewer. We have found several in the Internet, such as 'APV - Android PDF Viewer', 'vudroid', ebookdroid' and 'apdfviewer'. We have tested their performance by reading different types of PDF files.

	APV	vudroid	ebookdroid	apdfviewer
Only text	2	3	1	4
Only image	2	4	3	1
Both text and image	1	3	2	4
Small file size	1	3	2	4
Very large file size	2	4	1	3

(1 is the best, 4 is the worse)

We have tested the four APIs by five different types of PDF files, one is only containing texts, one is only containing images, one is containing both texts and images, one is of small file size and the final one is of very large file size.

APV - Android PDF Viewer

'APV' has the best overall performance, and does best when reading PDF containing both texts and images with small file size.

vudroid

'vudroid' does worst when reading PDF with only images and of very large file size. Its overall performance is the worst among the four APIs.

ebookdroid

'ebookdroid' has very good performance when reading PDF with only texts and of very large file size. However it does not perform good when reading images.

apdfviewer

'apdfviewer' does the best when reading images, however, it has the worst performance when reading texts.

Conclusion

Although 'ebookdroid' does better than 'APV' when reading PDF with texts only and of very large file size, a paper contains both texts and images, and its file size would not be very large, we believe that 'APV' is the best API to read papers, and we choose to use 'APV' in our project.

3.5. More features can be utilized: how to achieve more?

We have done some searching in further exploration on the functionality of the application. We tried to start with our objectives set, and some sparkles invoked.

To improve the efficiency?

As undergraduate students, we have never got a chance to perform a formal and comprehensive research on any topic. However, we believe that every time we do research and investigation, we need plenty of information and bibliography to support what we are saying. Quite annoying scenario arouses – when we try to look for some new information, we read and find some unknown words and terms. Then we try to make it clear, and we search for other pages of information and more and more terms will come out. Just like some kind of recursion. Finally, after we open for hundreds pages of reference, we almost forget what we asking for!

Thus, we find it will be valuable to have a research log for each user. User information are recorded in our database and users have to login to use the application. Every time they read a paper, there are their own logs to save the record. Even though they go though from one paper to another for hundreds of times, they can still find what they originally look for and can more handily filter what they need.

To exchange ideas?

When it comes to communication, it is not surprised to think of the idea of instant chatting. Chatting or messaging helps for better discussion on a topic, while some views are worth arguing. It seems reasonably, but impractical indeed. Firstly, due to absence of substantial keyword, the application is designed for as less input as possible. Discussing on instance messages requires a certain amount of input data. Moreover, the main advantage of a tablet is the portability. Serious discussion seems not meaningful on the railway journey, or if it does, phone calling would be more practical and efficiency.

We want some features are not redundant but useful. Does no instant messaging mean there is no need towards exchange of ideas? Definitely not. Ideas and views are aroused during the reading of other's writing, and we may give some response to the writer, or share what we think to a specific point. Yes! The comments.

Adding comments onto the paper, when someone has his own view, his can put it down with few sentences. The comments will be saved when anyone reopen the file. Every users can comment and view what others thinking by just one click. Exchange of ideas can hence be achieved, handily.

To minimize workload of managing resources?

Papers are our main characters of the show. We cannot implement an application without papers. We are actually not worried about functionality and the implementations, because once the algorithm and functions work successfully, it seldom have bugs. (Of course further analyze and maintenance is needed to improve the accuracy.)

However, even if the application is precisely implemented with perfect algorithms, the renewal of paper should be carried out everlastingly.

Could the paper library enriches and renews itself automatically? We raise this question and have a draft answer. We may embed a web browser into our application. When the user cannot find what he wants in our existing database, he can open the web browser to look for new information. If he would like to download a new paper, the library would save it add do checking and parsing. If there is duplication, the new file will be abandoned. Otherwise, the paper will be parsed for every data requires and a new entry will be added. With this feature, the maintenance and management by human being can be greatly reduced.

We noticed that there is a http://scholar.google.com/ which is a strong library of scholarly papers. We put this back and will look if we can take this advantage in the Project Phase 2.

3.6. Methods to locate targets: Any searching algorithm helps?

Considering our searching activities within the system, there are several characteristics different from usual web searching. One things worth to note is that keyword list may be occasionally provided in papers. This may affect the “keyword” definition of the searching result. On the other hand, we also try to ensure the keywords would be pointed if the paper itself does not contain any keyword list. As a result, we are going to adopt the principle of Term Frequency - Inverse Document Frequency (TF-IDF) into our searching function.

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is a weight to show the importance of a word to a document in a collection. This ratio is commonly adopted by search engines. The TF-IDF value contains two parts. The Term Frequency in a given document indicates the number of times the looking up key term appears in that document. The Inverse Document Frequency indicates if the key term is common among all other documents in the sample space. The IDF can be calculated by dividing the total number of documents by the number of documents containing the term. Finally, we have TF times IDF and that was the TF-IDF for ONE Document having that specific key term.

$$\text{Term frequency} = \text{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

, where $\text{n}_{i,j}$ means the frequency of the term appears in the documents d_j

$$\text{Inverse document frequency} = \text{idf}(\text{t}, \text{D}) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

, where $|\text{D}|$ *is the* total number of documents in the corpus

, and $|\{d \in D : t \in d\}|$ *means* number of documents where the term t appears

$$\text{TFIDF} = \text{tf} \times \text{idf}(\text{t}, \text{d}, \text{D}) = \text{tf}(\text{t}, \text{d}) \times \text{idf}(\text{t}, \text{D})$$

With higher TF-IDF, we said the importance of the specific key term for this document is higher. We can implement this TF-IDF for deciding key terms in papers.

Stop words

Where calculating TF-IDF, it might be a good idea to build up a dictionary on our own, aiming to record the frequency of each terms in each paper to reduce the processing time of searching. It is anticipated that the dictionary could be very large, however some of words are meaningless to keep. For example, the terms ‘is’, ‘am’, ‘are‘ will not be searched as a key term. These words are called ‘stop words’ and in principle they are filtered by calculating the TF-IDF (Reason: although the word ‘are’ has a high occurrence in one paper, it would also have a high document frequency and hence gives a fair TF-IDF value).

Besides, there are pre-written stop word lists online. There is no definite list of stop words. Many searching engines avoid adopting stop word list because it may hinder the search of phrases.

Stemming

When it comes to searching, it should be noted that some languages, in our case we focus on English only, share same word root for several wordings. Take 'produce' as an example, when it is typed as key term, many other words may be considered as significant to the result – 'product', 'production', 'produced' and 'produces' etc. The process of identifying 'product' and 'production' as 'produce' is called stemming.

There are two basic algorithms to implement stemming. One is lookup algorithm and another is suffix-stripping algorithm. Lookup algorithm relies on a look-up table, which is simple and fast. Exceptions can also easily be handles. However, updating the look-up table puts great workload on maintainers and the table shall be quite large in certain scale.

As for suffix-stripping algorithm, it is implemented by analyzing the suffix rules of general wording formation. It refers to some rules instead of a large table. Applying this algorithm, for example, the 'effectively' would be reduced into 'effective' and then 'effect' eventually. And hence searching for 'effective' would gain the results of 'effectiveness', 'effecting' and 'effected'. In addition, there are further stemming algorithms but there are fundamentally derived from these two algorithms.

3.7. Linkage between papers: How papers relate to each other?

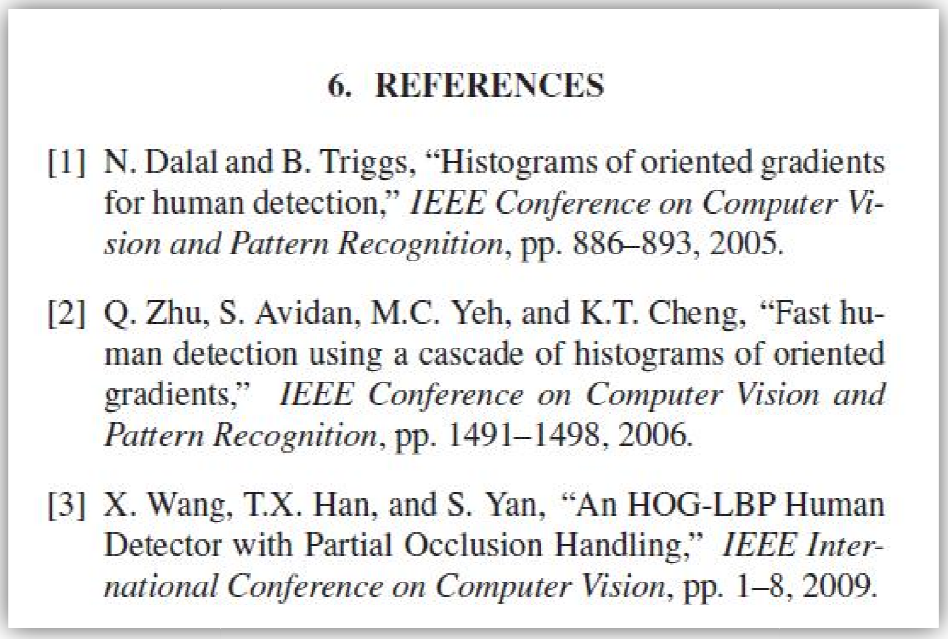
Every paper with references keeps a standard format for each bibliography. In general, each item should contain title, authors' name and year of release. Occasionally, the publication or conference of releasing papers will also be included, and sometimes with place of publication and exact page numbers. To extract the crucial information for the target of referencing, our focus will be on the author parsing and title parsing.

Author

Over 90% of bibliography of the papers puts the names of authors in front of all other information. Every name goes with given name first, sometimes in abbreviation or in full name, and then followed by the surname. If the number of authors is more than 3, the names would be separated by ",". For the case with only 2 authors, and those last 2 authors among 3 or more authors, the name of these two would probably divided by the word "and" (Figure 3.7 and Figure 3.8).

Title

Right after the author, there is always a title. The title would be segregated by "." or ",". For the segregation of ",", since "," is also used for dividing authors' names, further pattern should be analyzed. If the authors' given names are shown in abbreviation, the first word right after the "," without abbreviation indicates the first word of title. If the authors' given names are shown in full name, the title is indicated by double quotation marks (Figure 3.9 and Figure 3.10).



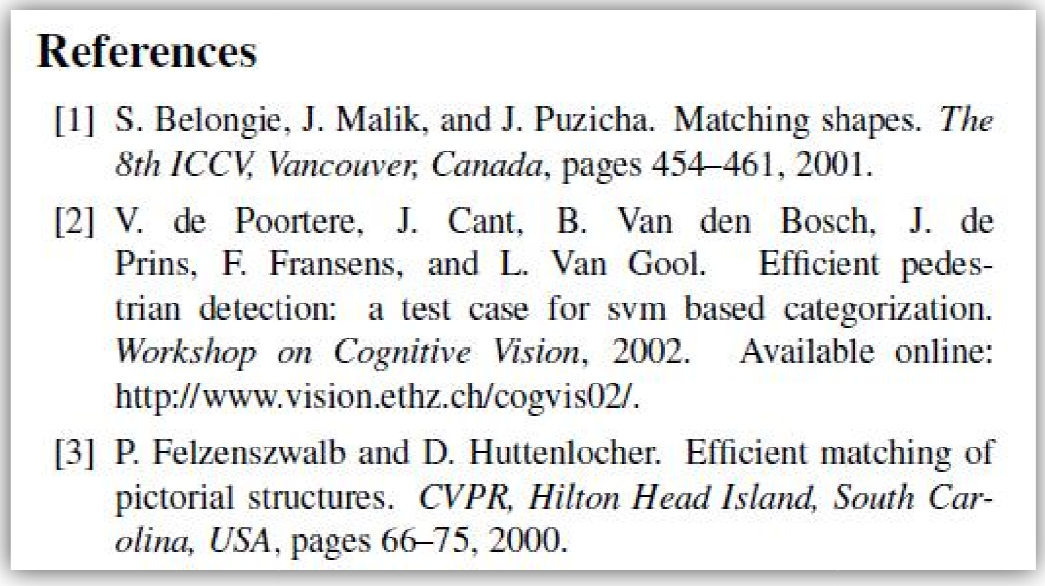
6. REFERENCES

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 886–893, 2005.

[2] Q. Zhu, S. Avidan, M.C. Yeh, and K.T. Cheng, "Fast human detection using a cascade of histograms of oriented gradients," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1491–1498, 2006.

[3] X. Wang, T.X. Han, and S. Yan, "An HOG-LBP Human Detector with Partial Occlusion Handling," *IEEE International Conference on Computer Vision*, pp. 1–8, 2009.

Figure 3.7 Some references in paper "Fast Human Detection Using MI-SVM and a Cascade of HOG-LBP Features"



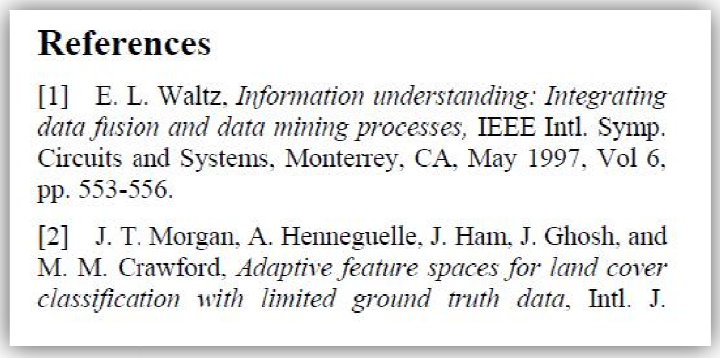
References

[1] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. *The 8th ICCV, Vancouver, Canada*, pages 454–461, 2001.

[2] V. de Poortere, J. Cant, B. Van den Bosch, J. de Prins, F. Fransens, and L. Van Gool. Efficient pedestrian detection: a test case for svm based categorization. *Workshop on Cognitive Vision*, 2002. Available online: http://www.vision.ethz.ch/cogvis02/.

[3] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. *CVPR, Hilton Head Island, South Carolina, USA*, pages 66–75, 2000.

Figure 3.8 Some references in paper "Histograms of Oriented Gradients for Human Detection"

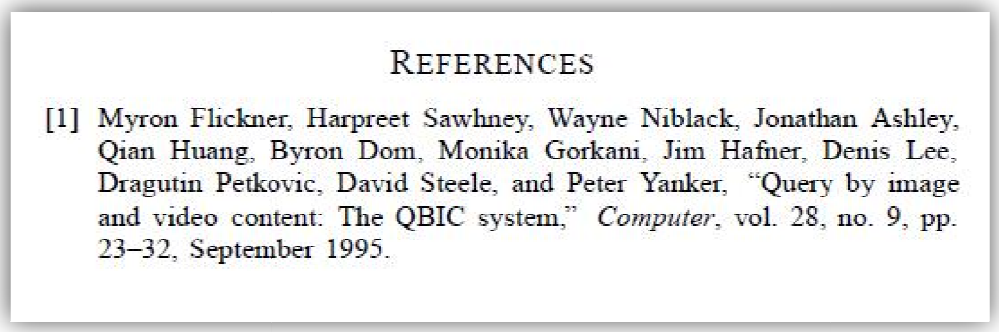


References

[1] E. L. Waltz, *Information understanding: Integrating data fusion and data mining processes,* IEEE Intl. Symp. Circuits and Systems, Monterrey, CA, May 1997, Vol 6, pp. 553-556.

[2] J. T. Morgan, A. Henneguelle, J. Ham, J. Ghosh, and M. M. Crawford, *Adaptive feature spaces for land cover classification with limited ground truth data*, Intl. J.

Figure 3.9 Some references in paper "Mining High-Dimensional Data for Information Fusion: A Database-Centric Approach"



REFERENCES

[1] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker, "Query by image and video content: The QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, September 1995.

Figure 3.10 Some references in paper "Compressed Domain Techniques to Support Information Retrieval Applications for Broadcast News Videos"

3.8. Duplication checking: What makes each paper distinct?

Under most circumstance, title and author names are primary keys to a paper. Sometimes the paper will have different versions. The create day and modify date of the document are not usually trustworthy. The create day and modify day is concerning the FILE but not the PAPER itself. The PAPER itself, indeed, can be saved as different files in different time. As a result, no year and modify date can be chased. There is no any straight-forward suggestion other than comparing both file content directly for duplication checking.

4. Overall Design

4.1. System Architecture

Our system consists of server side (Figure 4.1) and client side (Figure 4.2) for better communication and management, and for better collaboration with the database.

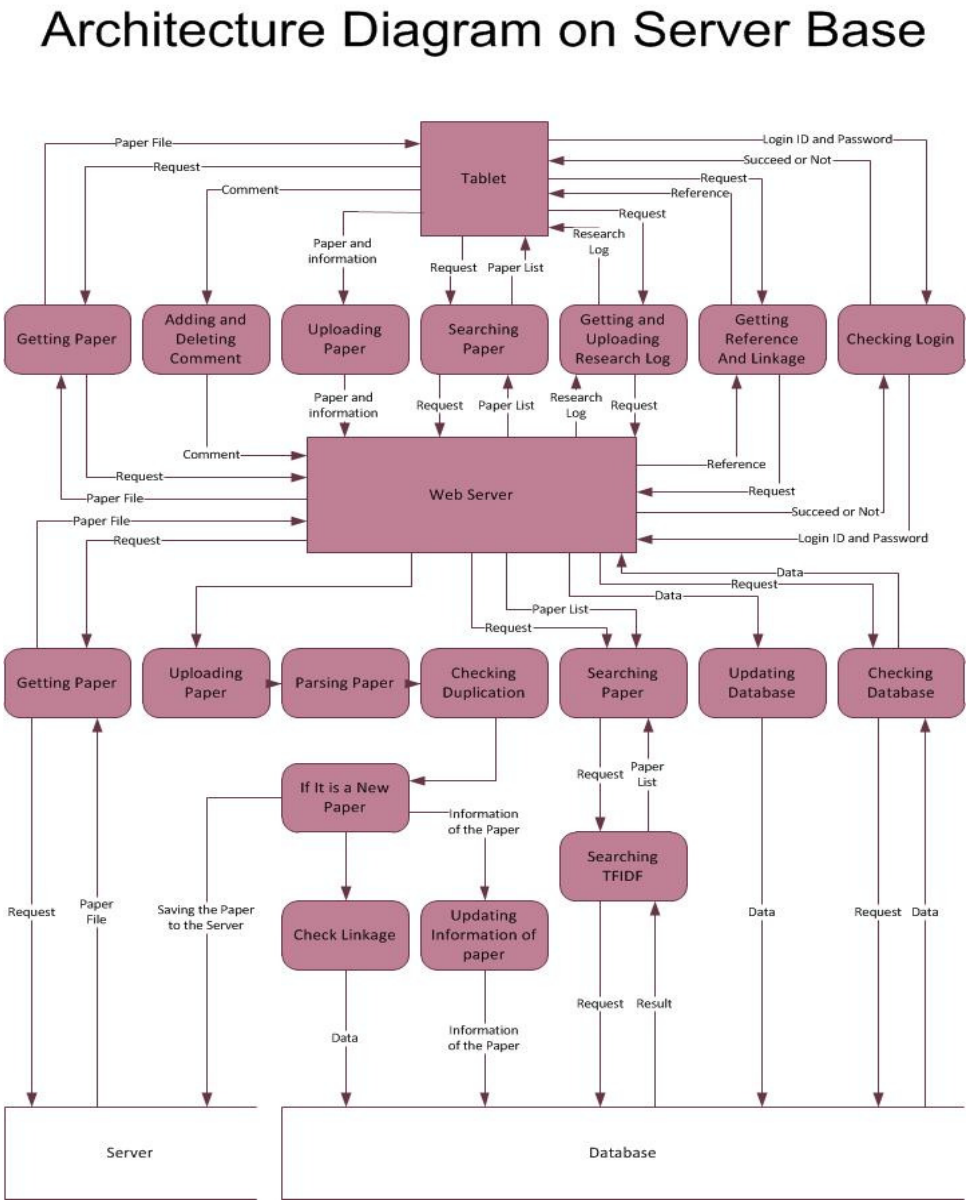


Figure 4.1 Architecture diagram on server base

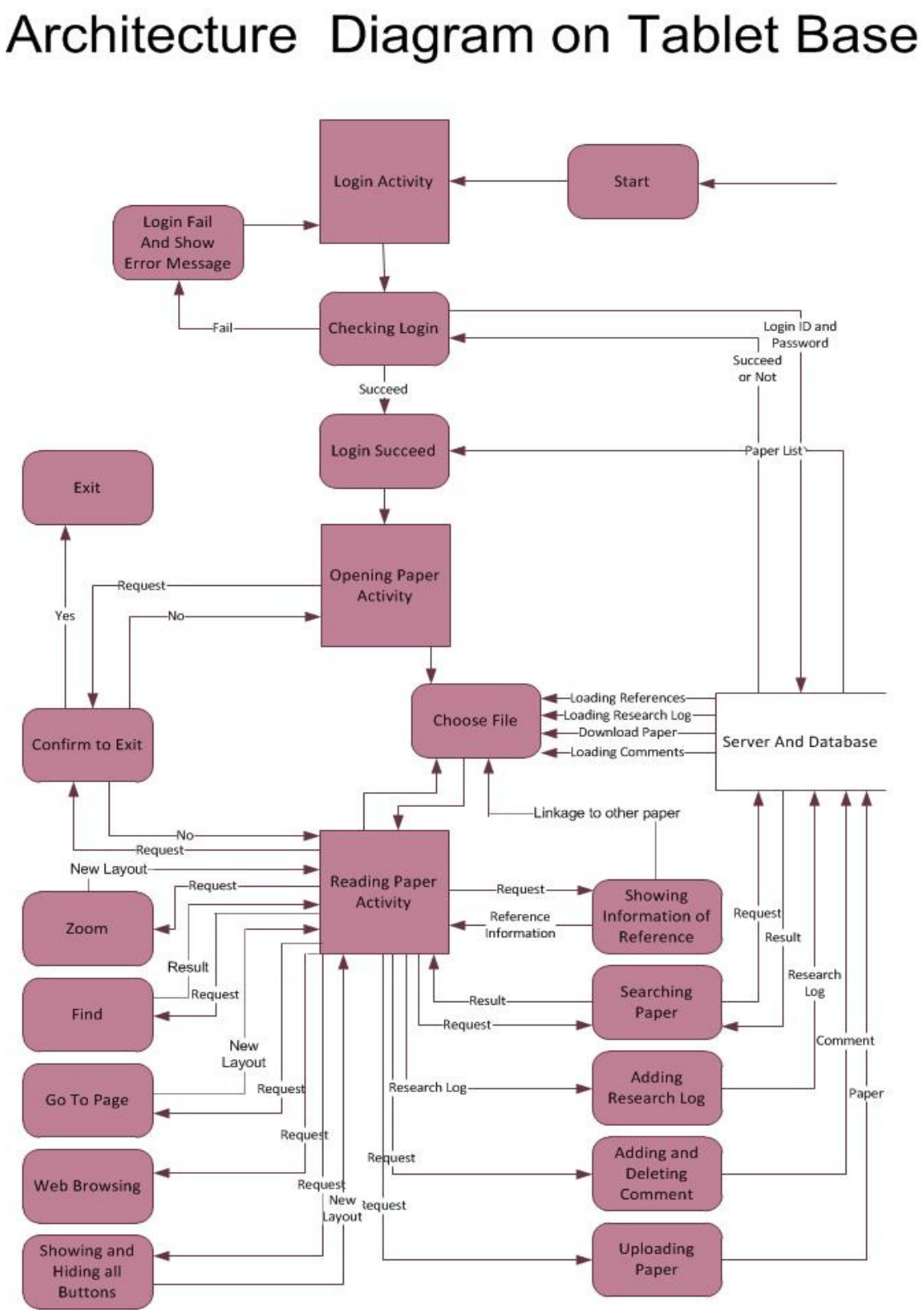


Figure 4.2 Architecture diagram on tablet base

4.2. Module Description

To make the tasks simpler, we have divided our works into three different modules.

4.2.1. Paper Management

Paper management plays a great role in our project. On one hand, well paper management helps researchers and scholars to locate their target resources and information easily. If the library is strong enough, the researchers can save much more time and improve the research efficiency. Just say, how to guarantee each paper is unique? Even when we type some keywords onto anyone of search engine, a list of repetitive results is not surprising. Yet most of the search engine is not clever enough to ease out those duplicated results. Extracting the information cost much time when we try to perform search. Therefore, we would try to alleviate the problem of duplication – try to store only one file despite different file name of same paper, or save the same paper only if the version is different.

On the other, the available paper resources online are vivid. Of course it is impossible to grab all papers from the internet once the application released, yet it is our ultimate goal to keep as much file as possible in our library. The tool may not powerful and intelligent enough to download all files online automatically, but we hope to achieve the task with the help of the users. The application on tablet embeds a web browser and let the users to search on web. When a paper is downloaded, the app will raise a signal and send the link of the paper to the server. After comparing the paper newly downloaded is unique and has not been saved on the server, a new file will be included and the source path is recorded to avoid duplication in the future. The library can hence expanded without heavy maintenance and involving a great amount of manpower.

Moreover, assisting the users to view the paper easily is one of our main objectives. To achieve this, other than show the references in just one click, we would also like to introduce some linkage between papers. While the user takes a look at the reference of some specific points, a hyperlink is provided to link the reference paper and the user can therefore have further investigation on the related topics. If the idea can truly be realized, it can surely minimize much more time for the user to do searching and collecting.

4.2.2. Communication

The concept of exchange of ideas has been discussed previously. We expect no any instant messaging because we hope to avoid inputting great amount of data without a substantial keyword, and we believe the user would relied on other more effective and efficient communication method rather than using our application.

We will let the use to post his comment onto the paper. The comment can be added onto anywhere of the paper. There will be a button or a flag for identification of the existence of the comment. Every other user can view the comment and can give response to the point too. The users who have echoed their ideas before would receive a notice that someone posted response. These stimulate the discussion by a small scale of input data. Since all the comment will be saved, the discussion can be shared to many users and the circulation of ideas can be achieved.

4.2.3. Research Log

Researchers face a huge obstacle that has been noticed for many times before: the huge amount of information flooding into our mind. When we go through for a referenced idea, we flip over the pages for the first term. To understand the meaning of a theory or a term, we shall then refer to one another. We flip the pages over and over, and for many times after we almost forgot what we are reading at first.

The research log shows everything that each user has done. Every single paper that his has read is record and sorted by the opening time. This allows the user to chase back what his is originally looking for.

We would like to make the research as strong as possible. Even the time the user posts a comment on a specific paper would be record. And if the comment posted has been replied by someone else, the user would be told by the research log too.

4.3. Functional Specification

This section will show general functions that the system can perform.

4.3.1. File Exploring

User can click on the file explorer to open the research paper that he needs. According to the interface design, after the user choose the file and the paper will be shown on the right hand side. A searching function will be available for user to search keywords, titles, contents or authors. Besides, for the user to read the papers more convenient, the user can choose to hide or show all the comments and referencing buttons in the paper.

4.3.2. Paper Parsing

Once the paper is added into the server, the paper shall be parsed in detail and all data shall be recorded in the database for better management. All fields will be parsed once to check if it is duplicated with existing papers. If the paper is new, all fields of data will be stored in the database for future uses.

4.3.3. Referencing

The newly added paper will be parsed to check if the reference index available. If yes, a list of references will be parsed into the database. The reference index throughout the whole passage will be highlighted and linked and user can refer to that point by a click. No matter the paper have indexed reference, a checking and linking procedure will be performed on the references in the list. If a piece of reference points to another paper that exists in our database, a linkage will be built for a faster referencing. Else, the user can search that reference paper from the internet for further study.

4.3.4. Library Renewal

A basic paper managing panel should be provided for uploading paper onto the server. The panel can be used by all users and the uploading process is manual. We would like to introduce a browser to be embedded within the application, so as the users can browse papers in the internet and upload papers to our server from his tablet. The application will first check if it is a scholar paper. The paper will be parsed and uploaded to the server, and information of the paper, such as title, author, subject and abstract would be able to be corrected if the original one is wrong. All these are expected to be implemented with manual control.

4.3.5. Posting Comments

User can post comment onto the paper. Different actions can be done, such as viewing, deleting and replying. The important stage is how can those have commented or responded to the specific point can be informed to extend the discussion. This part will be collaborated with the user research log functionality.

4.3.6. User Login

The application requires user to login. The user account would be used for research log, and also uploading new papers.

4.3.7. User Log

Each user has his user log for recording the history of browsing each paper so as to make the process of referencing run easily. The log also contains the history of posting comments and notifies the user if there is anyone responded to the comments he posted, as well as any new papers uploaded.

4.4. Development Platform

Server side

The server is built by php5.3.6 with Ubuntu. Due to the research related to the PDF, there is PDFlib in PHP to read PDF. PDFlib supports only the version newer than PHP 5.0.Xpdf and PDFBox are also need to be installed in the server for parsing the text in the PDF.

As for the pdfmeat, since we still cannot implement the CLI successfully, we would like to implement the project without it in phase1. If the self-implemented result is not satisfied, the pdfmeat will be introduced in phase2.

Client side

Since the Android platform is finally selected by the end of September, the latest version of Android at that time was chosen. Our project will be implemented in the Android 3.1 (the version which Samsung Galaxy tab 10.1 implemented [14]).

5. Detail Design

This chapter describes the design of the whole system in details. The shows all tiny features we would like to implement. In this part, we would state clearly whether the features will be implemented in phase 1 or 2 of the project.

5.1. Module 1 : Paper Management

This module performs four main functions.

5.1.1. File Exploring

The first major concern is on opening files.

Show list of paper (Phase 1)

The user interface (Figure 5.1) is composed by two parts. The sliding door on the left hand side is a file explorer. The list shows all the papers available on the server. In the first phase we will show the file name only. In the second phase we may consider to show the paper with title or author.

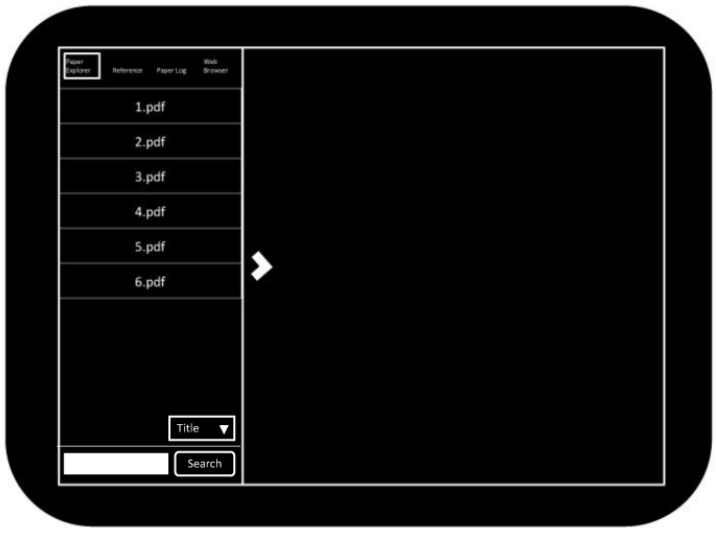


Figure 5.1 Basic UI design on the application

Show paper (Phase 1)

On clicking the file listed on the explorer, the paper will be shown on the right hand side (Figure 5.2). The PDF file should be allowed to zoom in or zoom out. It allows the user to perform searching on the content of the file. Directly going to a certain page is also allowed.

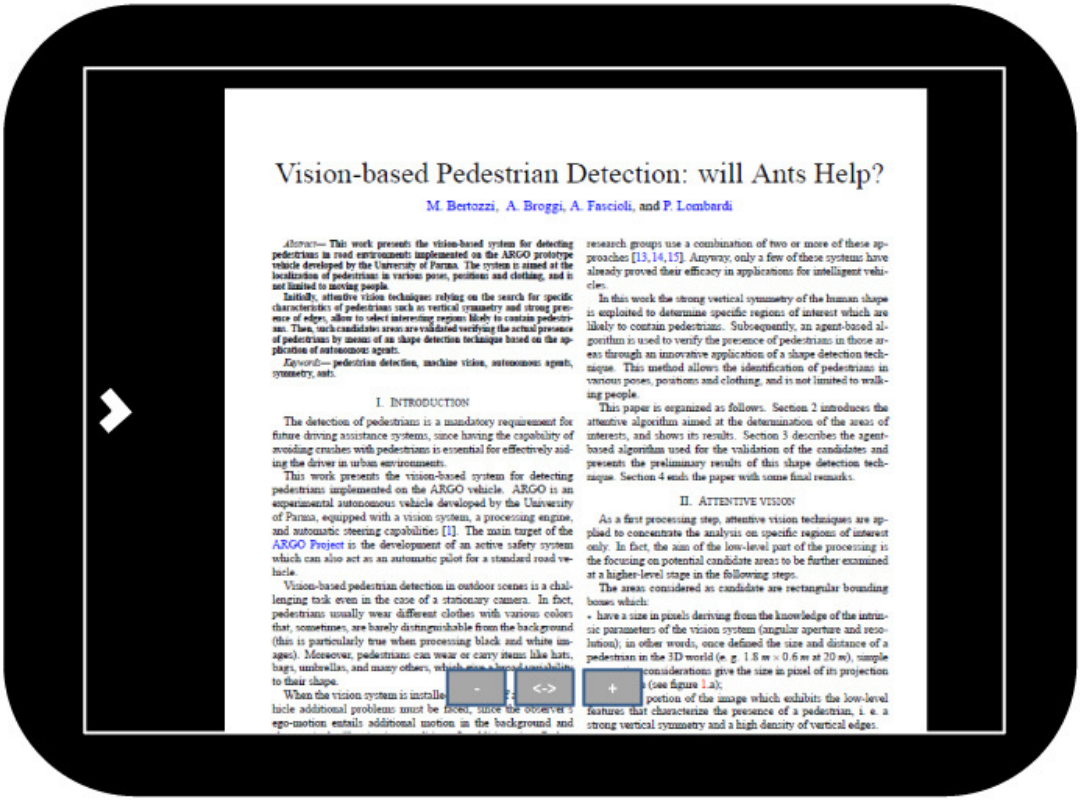


Figure 5.2 UI design of showing a paper

Search for paper (Phase 2)

The searching function should also be available (Figure 5.3). This makes the process of opening the correct paper run faster and more efficient. The searching will echo the keywords in title, authors, key terms and abstracts of the paper. When the list of searching is shown, abstract will follow the title and authors to provide sufficient information to users.

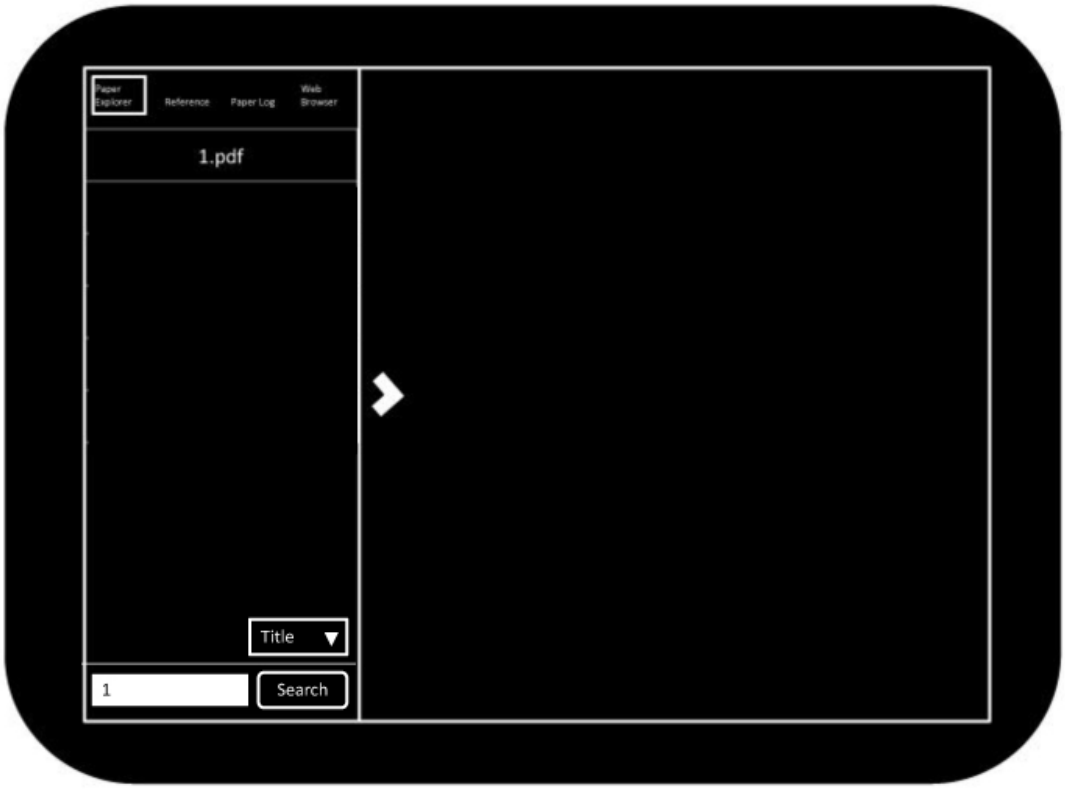


Figure 5.3 UI design of searching papers

Additional feature 1 (Phase 2)

On the first click the abstract of the paper will appear (Figure 5.4). The paper will be shown on the second click.

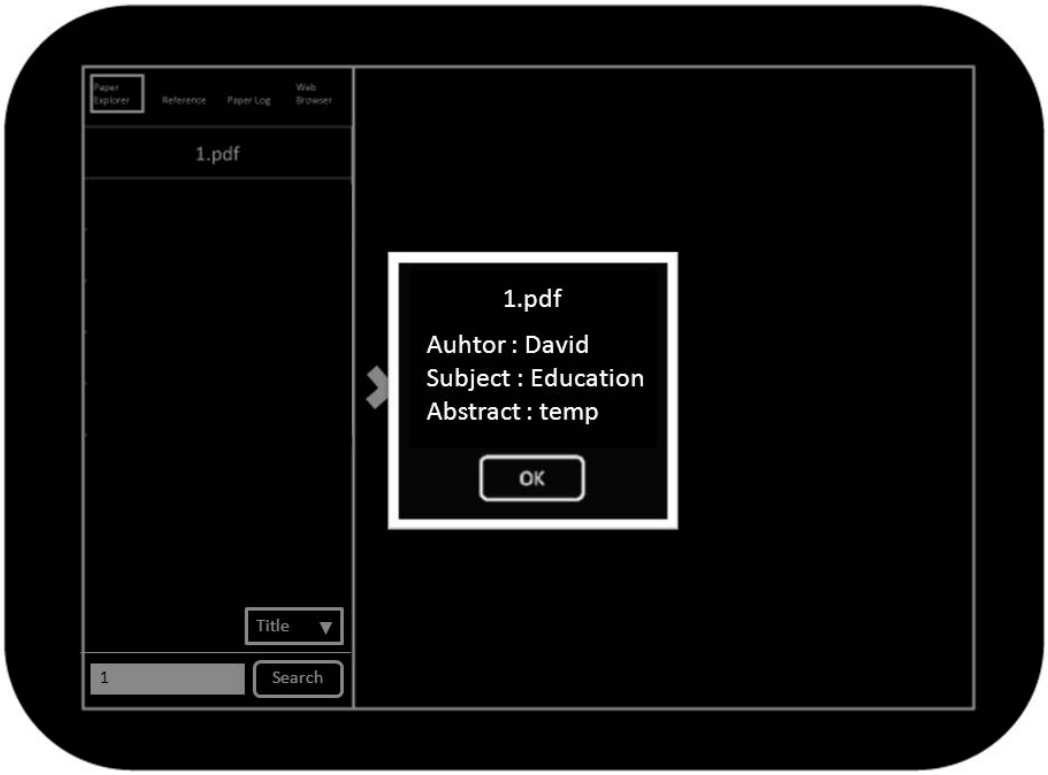


Figure 5.4 Abstract of the paper will be shown

Additional feature 2 (Phase 2)

There will be a button for the users to choose to show the comment and reference or not. One click on the button will hide all the buttons (Figure 5.5), and another click will show all (Figure 5.6).

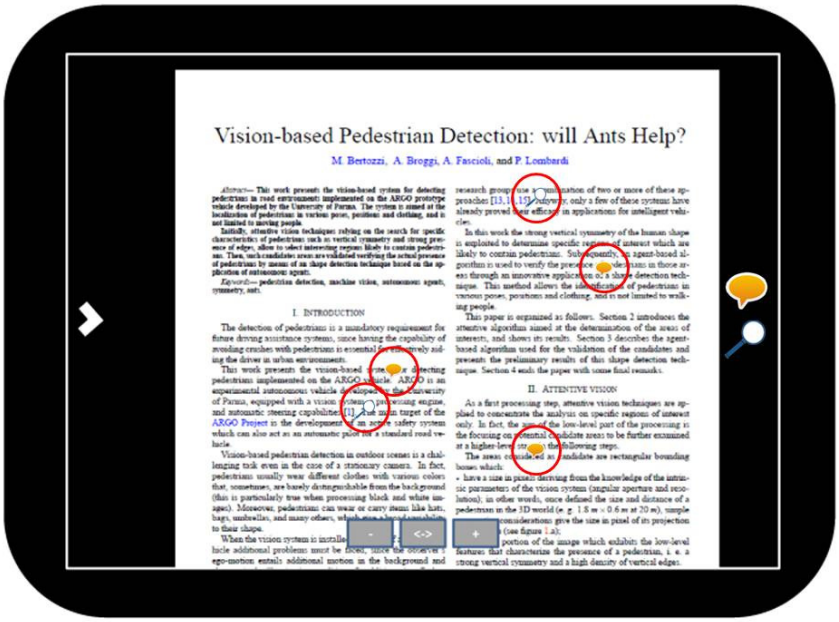


Figure 5.5 Show all Buttons

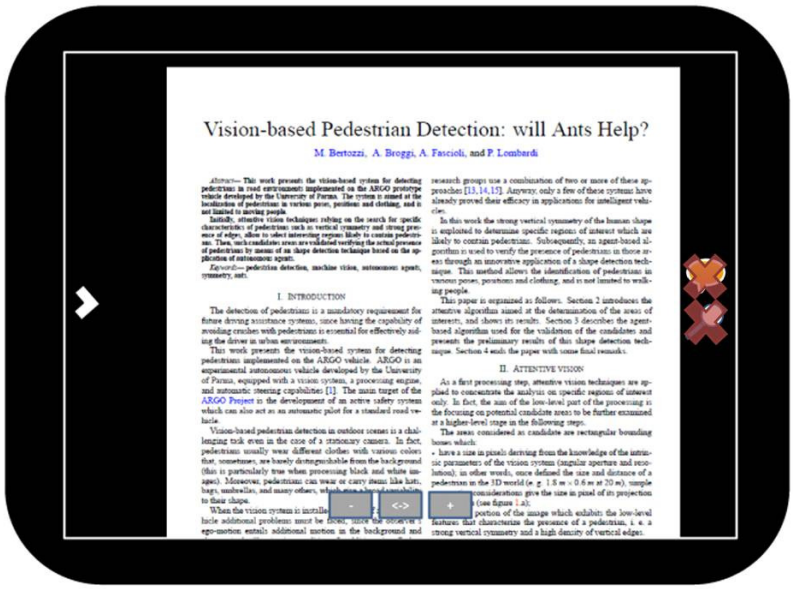


Figure 5.6 Hide all buttons

5.1.2. Paper Parsing

Paper requires parsing and saving data in database.

Get information needed (Phase 1)

Based on the result of our studies, we need several fields of paper: title, authors, abstract, keywords (if any) and references. The title, authors name, abstract and key terms are all help to identify every single paper. The title and author play important roles in display the stream of papers while the abstract and keywords help in searching efficiently. References are parsed for achieving better referencing.

Check duplication (Phase 2)

During the parsing, we would like to compare the original file name and title with the others in the database. If we found the title is duplicated, we will further compare the authors and abstract. If the file is truly duplicated, it will be abandoned. However, we would try to see if the version is different or few modifications are applied. In this case, we would like to store two or more versions for reference.

5.1.3. Referencing

Show the reference (Phase 1)

The indexes appear throughout the paper will be highlighted (Figure 5.7). When one of the indexes is clicked, a box will be prompted and show the piece of reference.

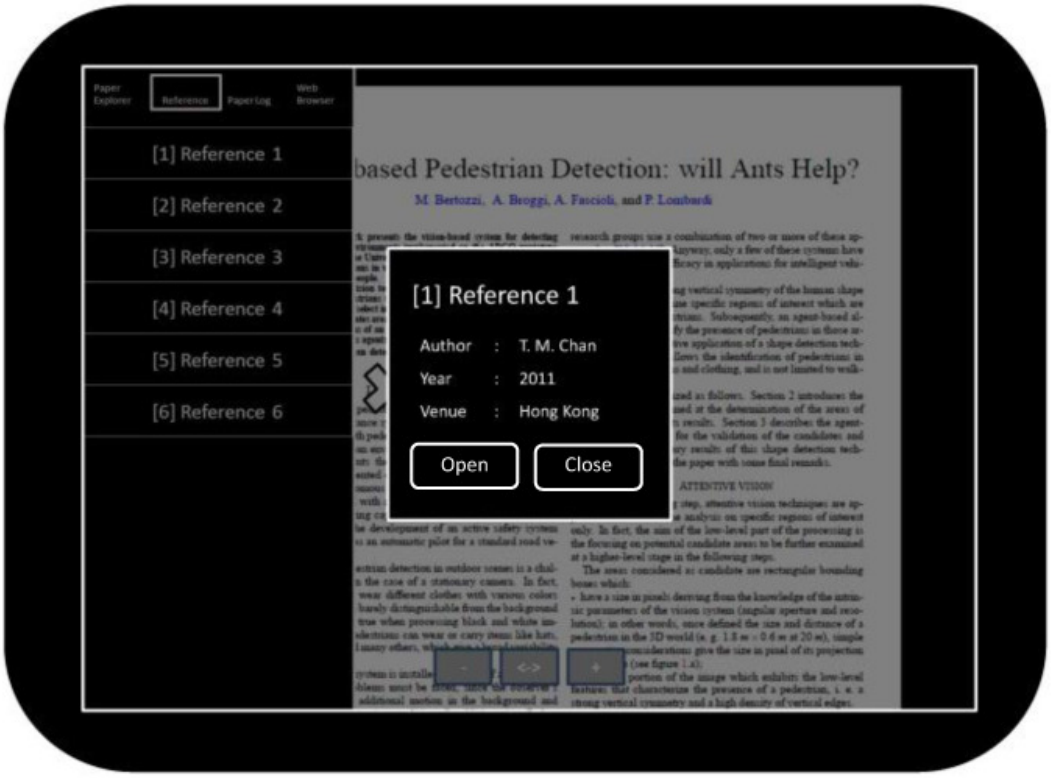


Figure 5.7 UI design of showing reference list

When one of the button of reference is clicked, a box will be prompted and show the piece of reference (Figure 5.8).

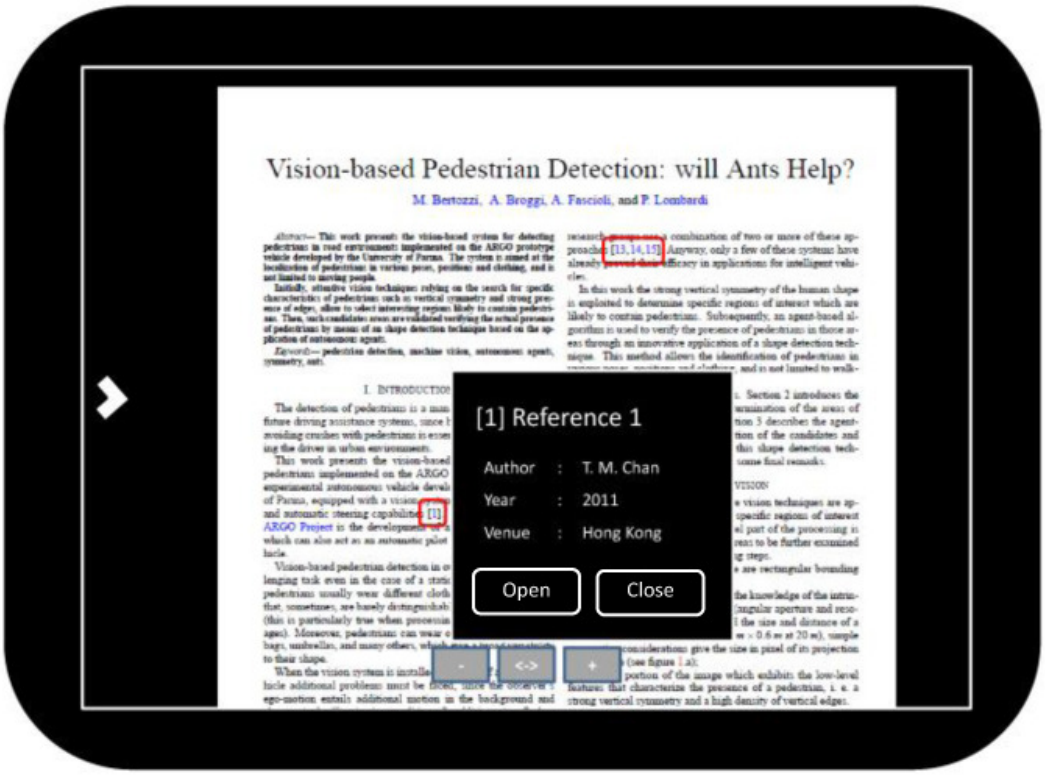


Figure 5.8 UI design of showing reference list

Build linkage to other paper (Phase 2)

If linkage is found during parsing, the result would be recorded in the database. One the linking reference is prompted out, a hyperlink is built and the user can check what it refers to on clicking.

Searching in the Internet (Phase 2)

If linkage cannot be built, a search button will be shown ((Figure 5.9), so the user can search it in the internet for further study (Figure 5.10).

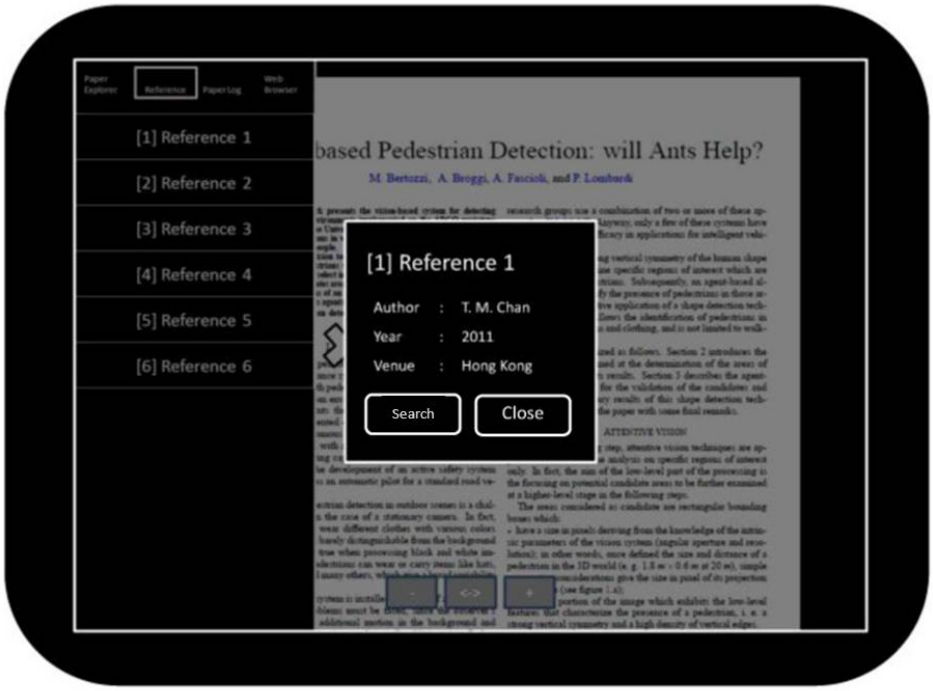


Figure 5.9 UI design of showing search button

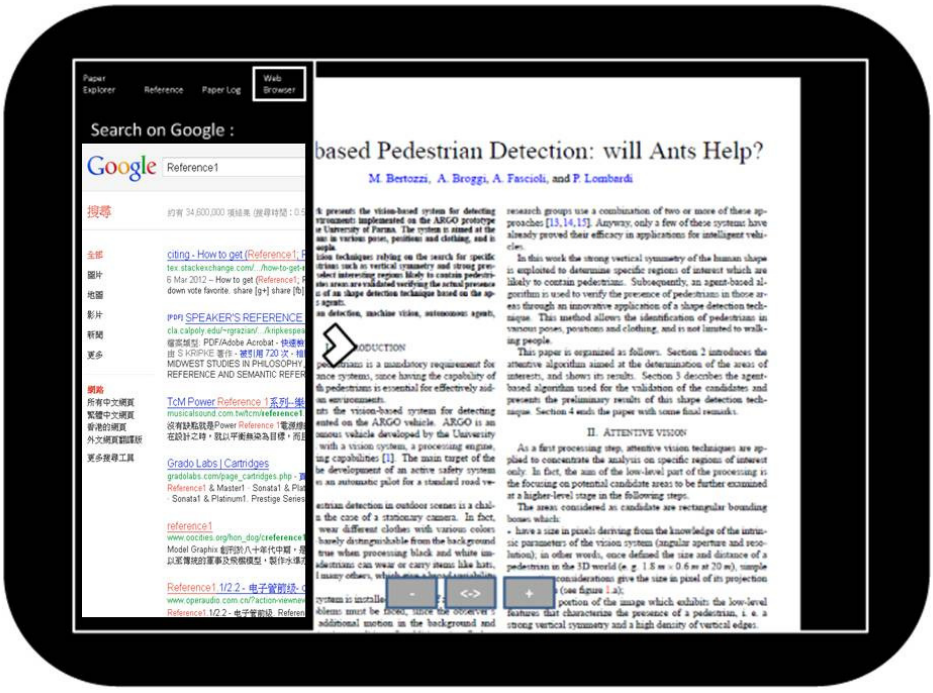


Figure 5.10 UI design of search searching paper in the internet

5.1.4. library renewal

Manage files manually (Phase 1)

A panel is designed for the administrator to perform the paper management handily and efficiently. The panel allows the admin to upload files, to control the parsing procedure manually and to delete files. The panel helps the developer for further enhancement by showing all fields and entries to ease viewing the parsing performance.

Embed a browser (Phase 2)

A browser (Figure 5.11) is suggested to be embedded within the application, so as to make the further search more handily. When downloading PDF is invoked during the searching in the browser, the application can automatically chase the file downloaded.

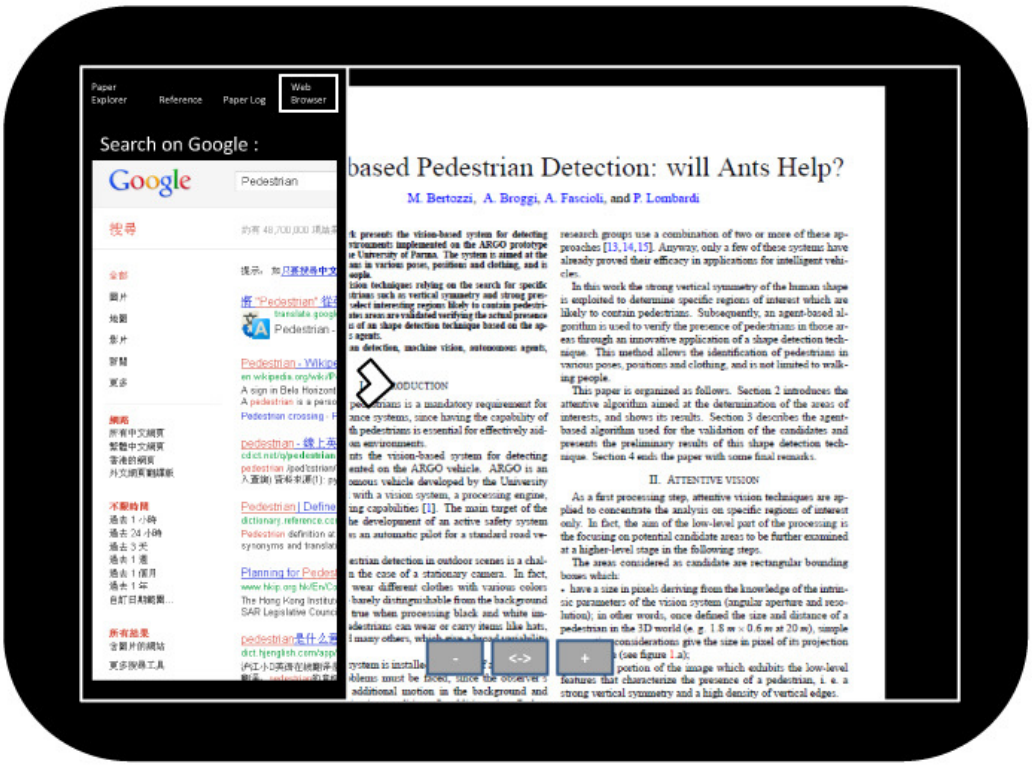


Figure 5.11 UI design of showing web browser

Upload Paper (Phase 2)

When application noticed there is PDF downloading relevant to certain papers, it would first check if the PDF is a scholarly paper. The paper will be grabbed to the server and under ordinary parsing, and hence the paper library can be further extended (Figure 5.12). Besides, the information of the paper can be corrected by the user manually (Figure 5.13).

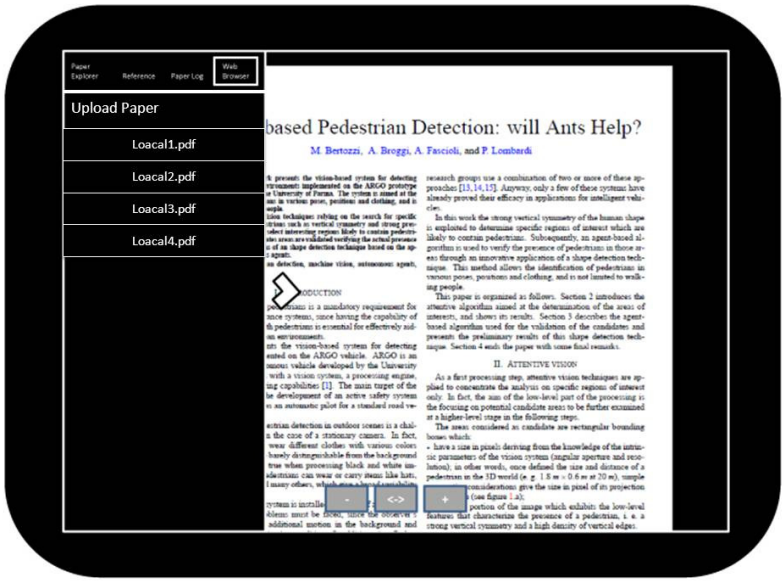


Figure 5.12 UI design of uploading paper

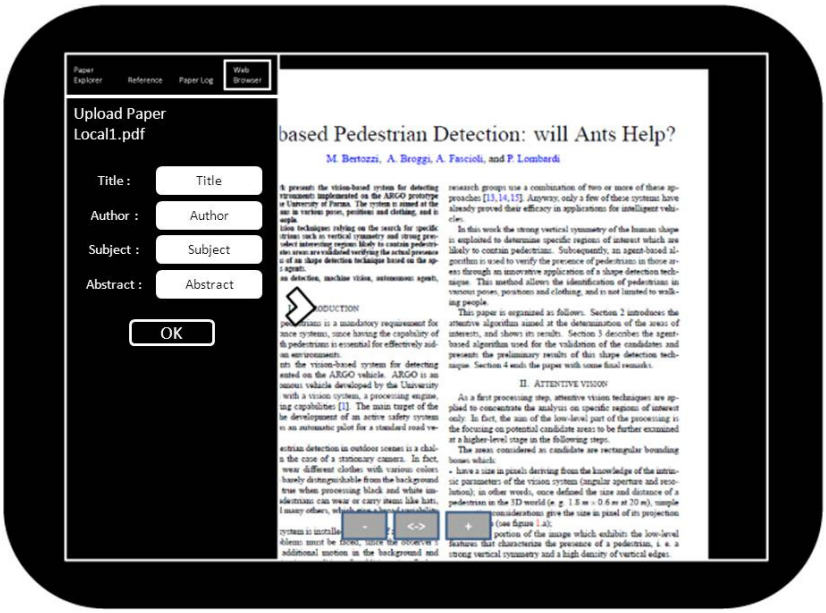


Figure 5.13 UI design of updating information of paper

5.2. Module 2 : Communication

5.2.1 posting comments

Add comments (Phase 1)

Every user can add their comment at anywhere of the PDF file (Figure 5.14). When adding the comments, the user ID and the time will be recorded.

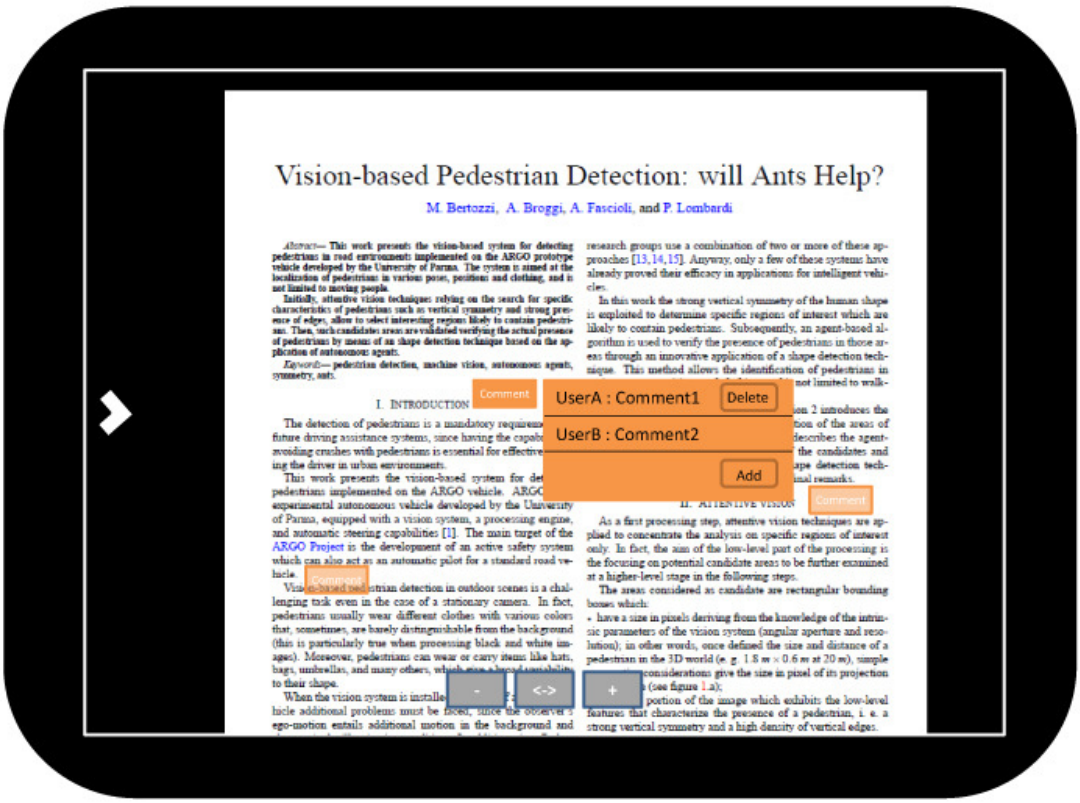


Figure 5.14 UI design of showing comments

Show comments (Phase 1)

When a user added a comment, a flag / a button will be shown on the position he added. When the flag is clicked, a dialogue box will show and all the content of the discussion can be viewed. The discussion will be stored in the database and whenever any user opens the same paper, the comments will stay there and every one can view.

Reply comments (Phase 1)

On the dialogue box showing the details of a discussion, a reply button is available for others to keep the discussion.

Delete comments (Phase 1)

For better management and avoiding misunderstanding in discussion, editing comment is not allowed. However, deleting comment is allowed. If the head topic is deleted, all the replies will be deleted. Yet if the reply is deleted, the topic will still keep.

Notify user of new replies (Phase 2)

The users who have participated in the discussion should all be notified when a new reply responded to the discussion is posted. This could further promote the exchange of ideas. This requires collaboration with the development of research log.

5.3. Module 3 : Research Log

5.3.1. User accounts

Login to system (Phase 1)

Before the user use the application, he should login to the page (Figure 5.15). Unauthorized login is not allowed to use the application. This ensures every user can perform all the features smoothly.

Figure 5.15 UI design of login

5.3.2. Research log

View history of reading (Phase 2)

The reading log (Figure 5.16) will refresh every time the user open a paper to read. The entries of the log will be sorted by the time happened in descending order. For better resources management, about 100 entries will be recorded for each user.

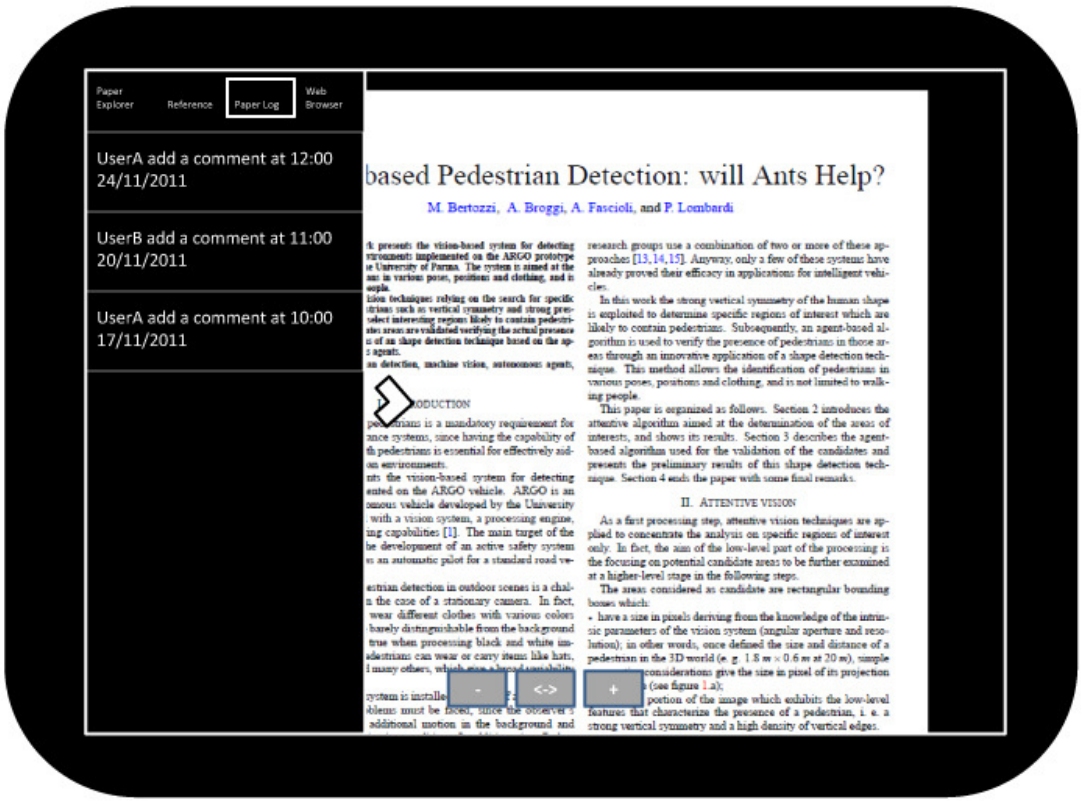


Figure 5.16 UI design of research log

View commenting activities (Phase 2)

The log will show the all the commenting activities that the user performed, including both adding comments and replying. The commenting time, title of paper will be shown.

Show notification of new reply (Phase 2)

While the discussion the user participated has been renewed, the user should be notified by referring the log.

Show notification of recently uploading of paper (Phase 2)

While one user upload a paper, a notification will be shown to all users in the research log.

5.4. Database design

The following diagram shows the whole picture of design of our database (Figure 5.17).

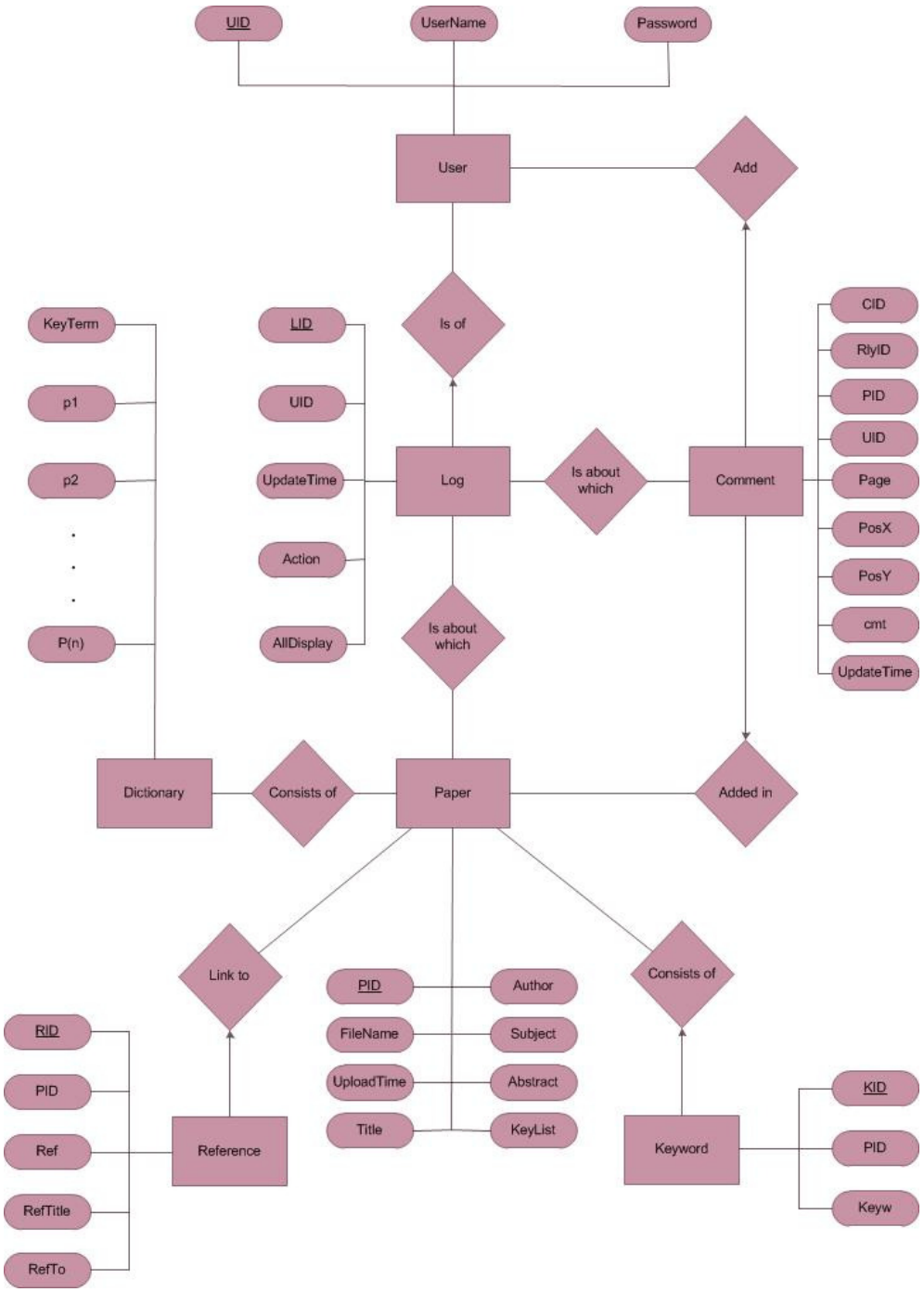


Figure 5.17 ER Diagram

The following diagram shows the detail design of the table of User (Figure 5.18).

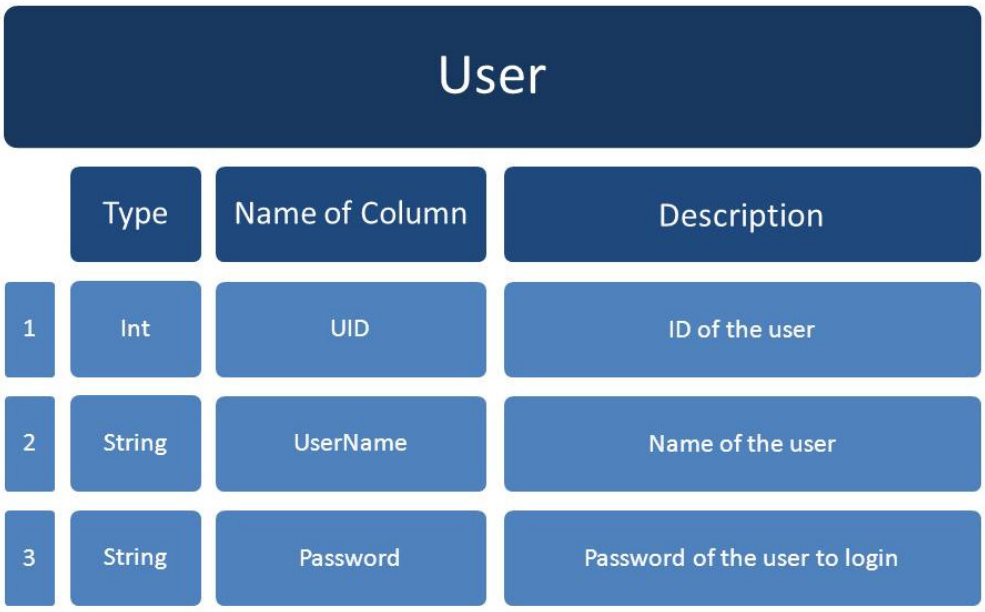


Figure 5.18 Table of User

The following diagram shows the detail design of the table of Log (Figure 5.19).

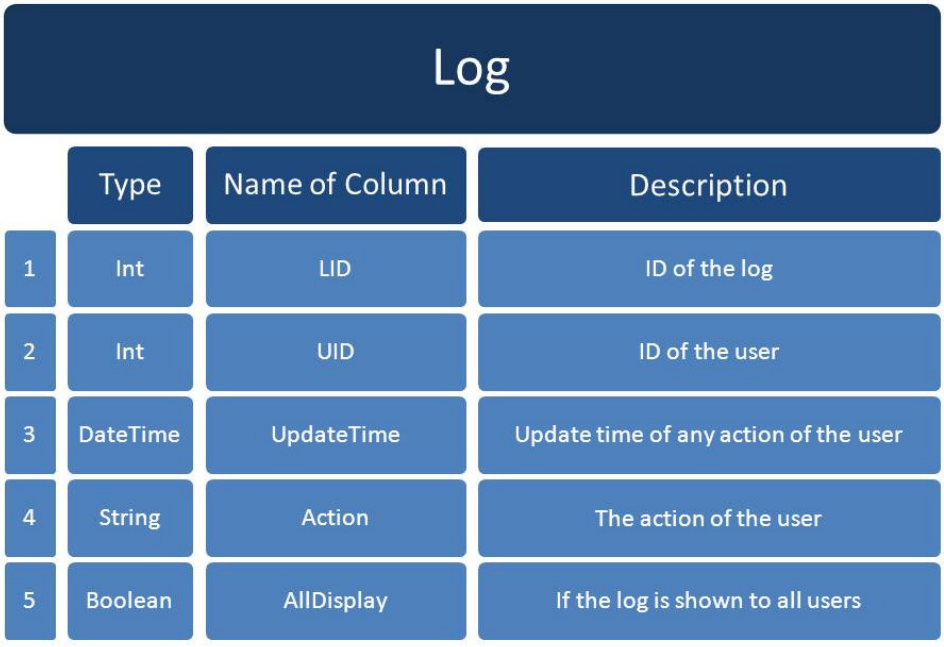


Figure 5.19 Table of Log

The following diagram shows the detail design of the table of Comment (Figure 5.20).

Comment

	Type	Name of Column	Description
1	Int	CID	ID of the comment
2	Int	RlyID	ID of reply in the comment
3	Int	PID	ID of the paper
4	Int	UID	ID of the user
5	Int	Page	Page number of the comment
6	Float	PosX	X-position of the comment on that page
7	Float	PosY	Y-position of the comment on that page
8	String	Cmt	Content of the comment
9	DateTime	UpdateTime	Updated time of the comment

Figure 5.20 Table of Comment

The following diagram shows the detail design of the table of Paper (Figure 5.21).

Paper

	Type	Name of Column	Description
1	Int	PID	ID of the paper
2	String	FileName	File name of the paper
3	DateTime	UploadTime	Upload time of the paper
4	String	Title	Title of the paper
5	String	Author	Author of the paper
6	String	Subject	Subject of the paper
7	String	Abstract	Abstract of the paper
8	String	Keyword	Keyword of the paper

Figure 5.21 Table of Paper

The following diagram shows the detail design of the table of Reference (Figure 5.22).

Reference

	Type	Name of Column	Description
1	Int	PID	ID of the paper
2	Int	RID	ID of the reference
3	String	Ref	Content of the reference
4	String	RefTitle	Title of the reference
5	Int	RefTo	The ID of paper that the reference should link to

Figure 5.22 Table of Reference

The following diagram shows the detail design of the table of Keyword (Figure 5.23).

Keyword

	Type	Name of Column	Description
1	Int	KID	ID of the Keyword
2	Int	PID	ID of the Paper
3	String	Keyw	Content of the keyword

Figure 5.23 Table of Keyword

The following diagram shows the detail design of the table of Dictionary (Figure 5.24).

Dictionary

	Type	Name of Column	Description
1	String	Keyterm	Key term inside the papers
2	int	P1	TFIDF of the keyterm inside paper ID 1
3	int	P2	TFIDF of the keyterm inside paper ID 2
.	.	.	.
.	.	.	.
.	.	.	.
n	int	Pn	TFIDF of the keyterm inside paper ID n

Figure 5.24 Table of Dictionary

6. Implementation

In the chapter, we are going to describe the implementation of features we implemented in project phase 1 only. For every module, we will describe the implementation method for each function in both client side and server side respectively.

6.1. Module 1: Paper Management

6.1.1. File Exploring

Show list of paper (Phase 1)

A) SERVER SIDE –

The server calls functions in PHP to open the directory "paper/" which store all the files. It returns all the files except the current directory path and parent directory path.

```
-  Pseudo code  -

IF (open the directory successfully)
{       WHILE (there is file)
        {       GET every file within the directory
                PUT the file into the json array     }
        CLOSE the directory         }
RETURN the json array
```

Note that the file name is changed into paper ID during the uploading of paper. Original file name with be kept in the database, for the uses of checking duplication in the future.

The expected result will contain a list of file names and will be encoded as a JSON array.

- Value Return - Example -

{"Current Directory":"paper","files":[{"file":"2.pdf"},{"file":"3.pdf"},
{"file":"4.pdf"},{"file":"5.pdf"},{"file":"6.pdf"},{"file":"7.pdf"},{"file":"8.pdf"},
{"file":"9.pdf"},{"file":"10.pdf"},{"file":"11.pdf"}]}

B) CLIENT SIDE –

After login validates, the client device sends request to the server and gets the list of file names. It will show the list in the file explorer in transparent panel on the left hand side one by one.

- Pseudo code -

```
IF (login validate)
{       SEND request to web server for the list of papers
        GET the result from web server
        SHOW the list of papers in the transparent panel        }
```

Although the file name is changed into paper ID, the list will show the title of the papers obtained from the database through the web server (Figure 6.1).

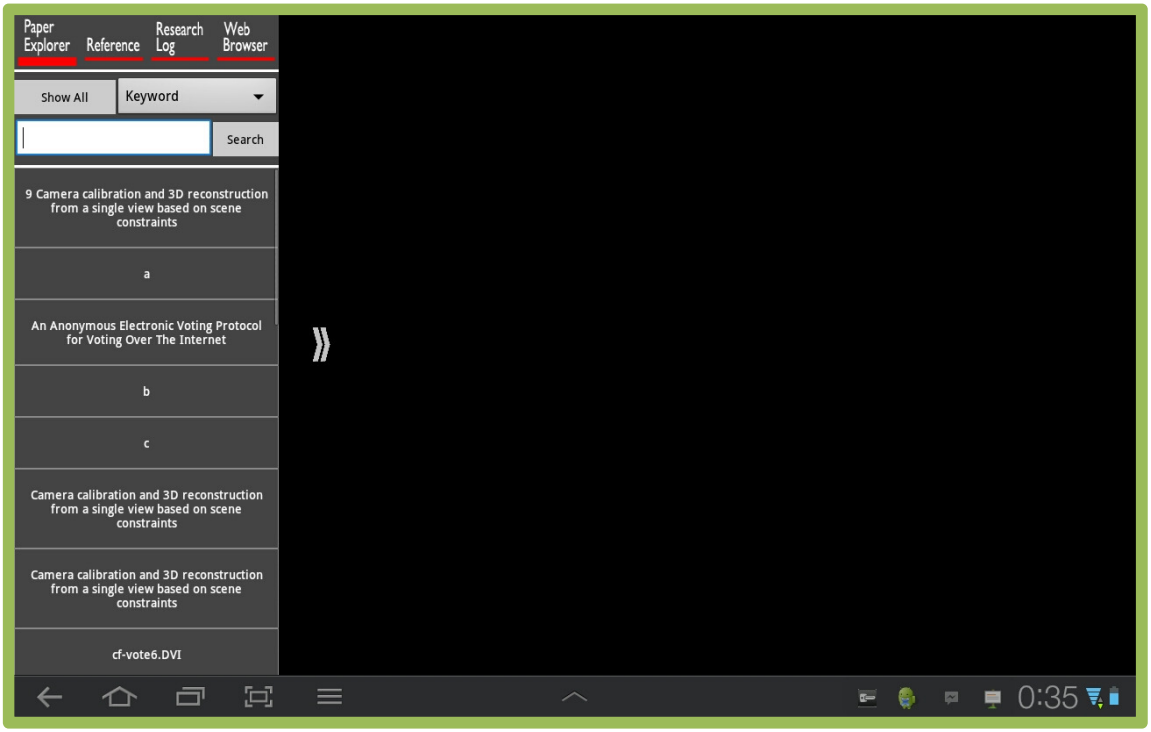


Figure 6.1 Screen capture of the paper list

Show paper (Phase1) and Additional feature 1 (Phase2)

A) SERVER SIDE –

In phase 1, the server saves the PDF files by changing the name to paper ID. This can avoid crashing due to the presence of special characters in file name.

In phase 2, since the paper is parsed right after uploaded, the metadata and abstract have been already inserted into the server database. Therefore, response of paper property and abstract can be rapidly made on request.

B) CLIENT SIDE –

After implementing the additional feature 1 in phase 2, when the user selects a paper, the information of the paper will be shown (Figure 6.2). If the user chooses to open it, the client device will download the paper through web server and save it in a temp folder. Then the client device will show the paper (Figure 6.3).

In phase 1, the application needs to get the data every times when it needs, for example, it needs to re-download the paper list or reference list when the users click on the corresponding buttons. Another visible example is the application needs to calculate the location of the reference buttons every times when the users open the paper. But in this phase, we have improved its efficiency by downloading the paper list and reference list only when the users open a paper. The application will also upload the location of all the reference buttons to the server when the paper is opened first time, so that any user opens the same paper next time, it can get the location of the reference buttons from the server and do not need to recalculate again, it obviously improve the efficiency and the convenience of using the application.

- Pseudo code -

```
IF (select paper)
{       SHOW information of the paper
        IF (choose to open)
        {       SEND request to web server to download the paper
                SEND request to web server for the references and previous comments
                GET the result from web server

                SHOW paper

                SHOW buttons for reference and comments     }       }
```

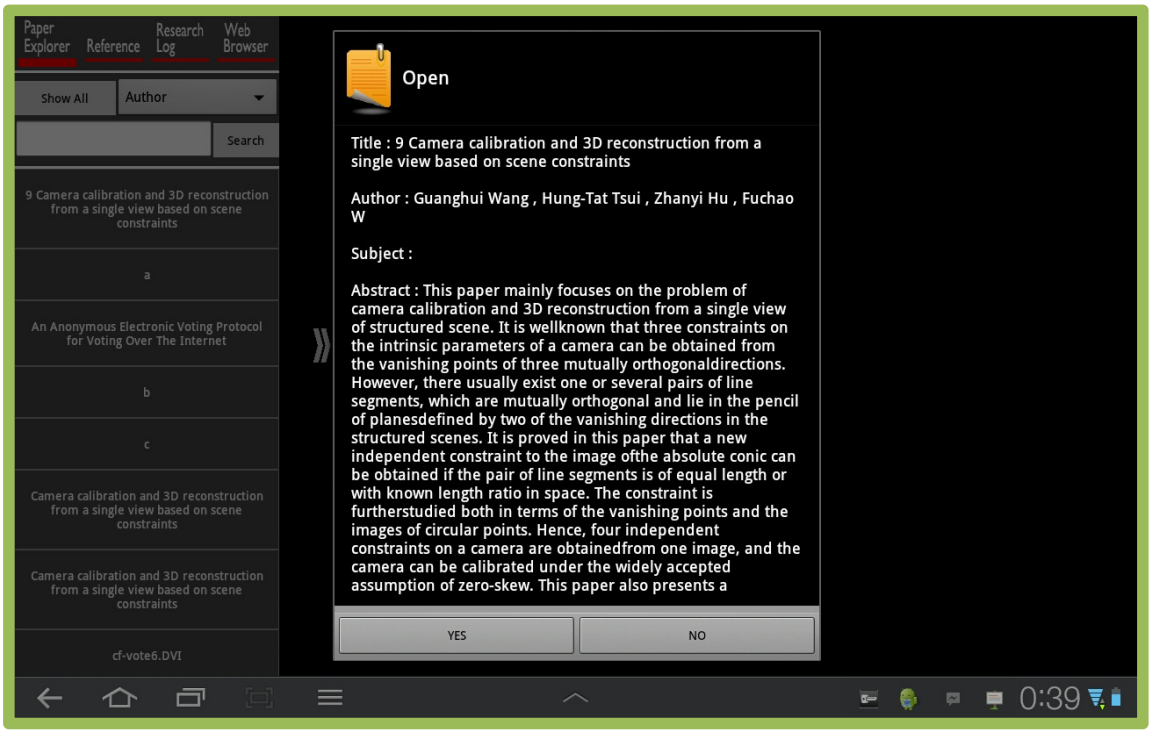


Figure 6.2 Screen capture of showing information of paper before opening

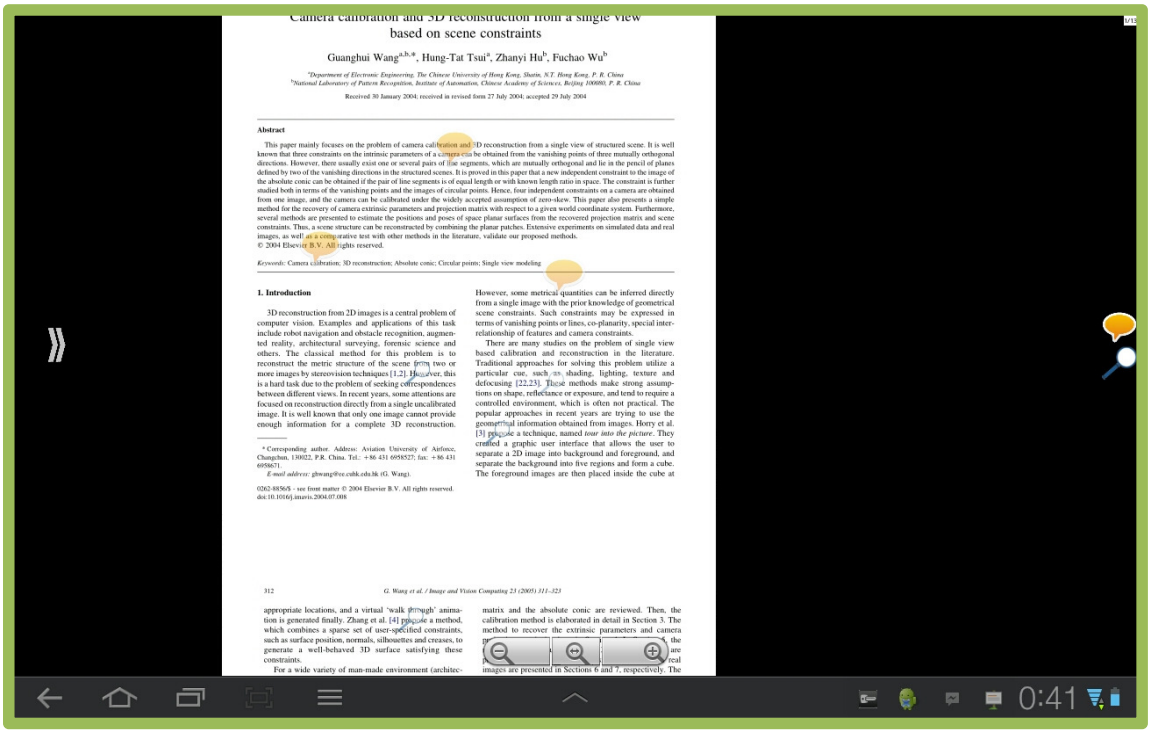


Figure 6.3 Screen capture of showing paper

The application can provide some functions for the users to read the paper more convenient. The three functions, including zooming, finding texts and going to pages, are the built-in functions of the API of PDF viewer we have used. We have studied how they work and are going to make some changes to them, so as to make it more suitable to our design.

For the zooming functions, the user can zoom in, zoom out or zoom to fit the width of paper (Figure 6.4). These zooming functions are implemented by pressing three buttons in the bottom of the screen, or double tapping the screen. For phase 1, the buttons cannot locate correctly after zooming, but in this phase, the location of the buttons will be recalculated after zooming, so that the location would be correct.

- Pseudo code -

```
IF (press button of zooming in or double tap)
{  CALCULATE the new paper size and new buttons' location
   SHOW the new view          }
IF (press button of zooming out)
{  CALCULATE the new paper size and new buttons' location
   SHOW the new view          }
IF (press button of zooming to fit the width)
{  CALCULATE the new paper size and new buttons' location
   SHOW the new view          }
```

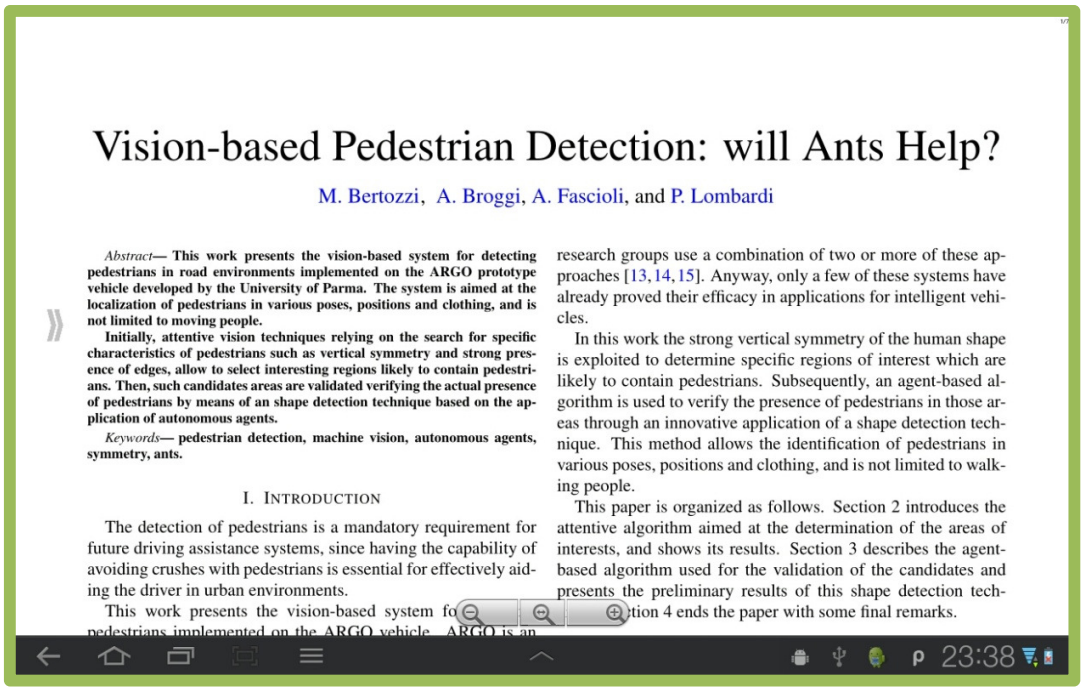


Figure 6.4 Screen capture of zooming

For the function of finding texts (Figure 6.5), the user can find the texts in the paper after pressing the button of finding texts. The texts found by the user will then be bounced three red lines (Figure 6.6). Besides, the user can choose to find the previous and next result. If this texts finding is finished, the user can hide all the result, and the view is returned to the normal view.

- Pseudo code -

```
IF (choose finding text)
{  ASK the user to input the text for finding
   FIND the text in the paper
   SHOW the results in the paper
   WHILE (showing paper with results of finding)
   {       IF (choose previous result of finding)
           {       MOVE the view to the previous result     }
           IF (choose next result of finding)
           {       MOVE the view to the next result }
           IF (choose hiding the results of finding)
           {       HIDE all the results of finding
                   BREAK       }       }       }
```

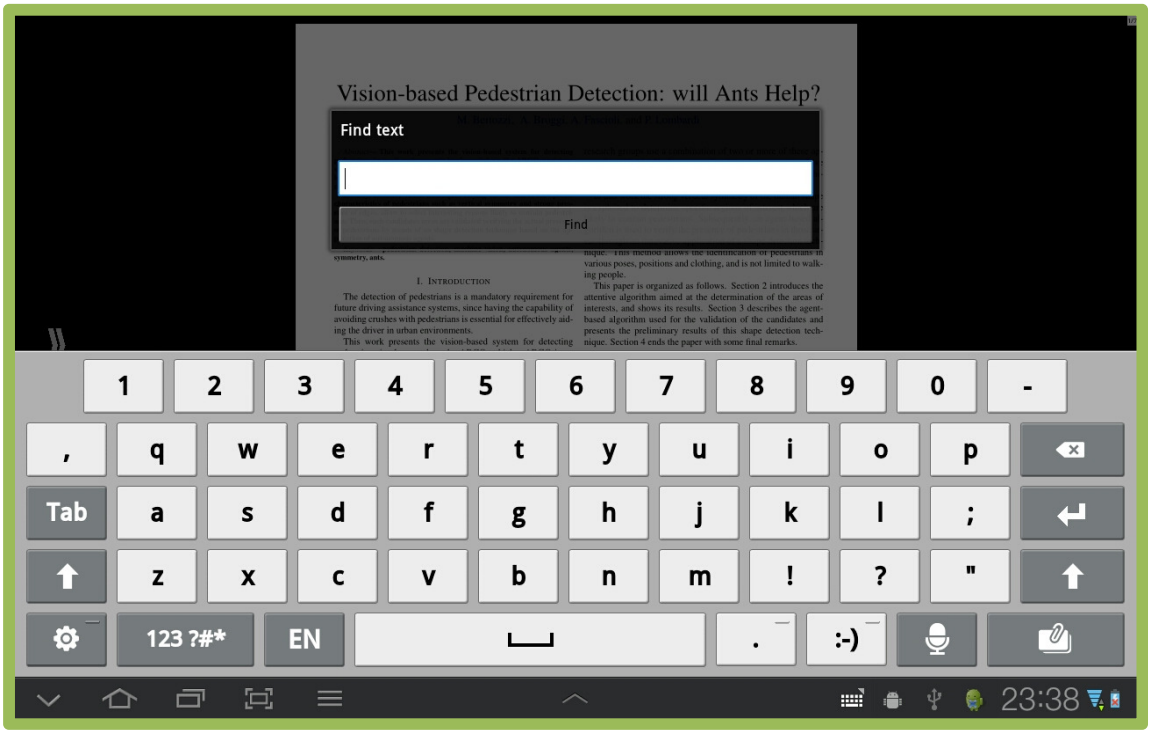


Figure 6.5 Screen capture of finding texts

Figure 6.6 Screen capture of result of finding texts

The last function is going to a certain page (Figure 6.7). The user can go to the first page, last page or a certain page entered.

- Pseudo code -

```
IF (choose going to certain page)
{  ASK the user to input the page number
   SHOW the paper at that page      }
IF (choose going to first page)
{  SHOW the paper at the first page }
IF (choose going to the last page)
{  SHOW the paper at the last page }
```

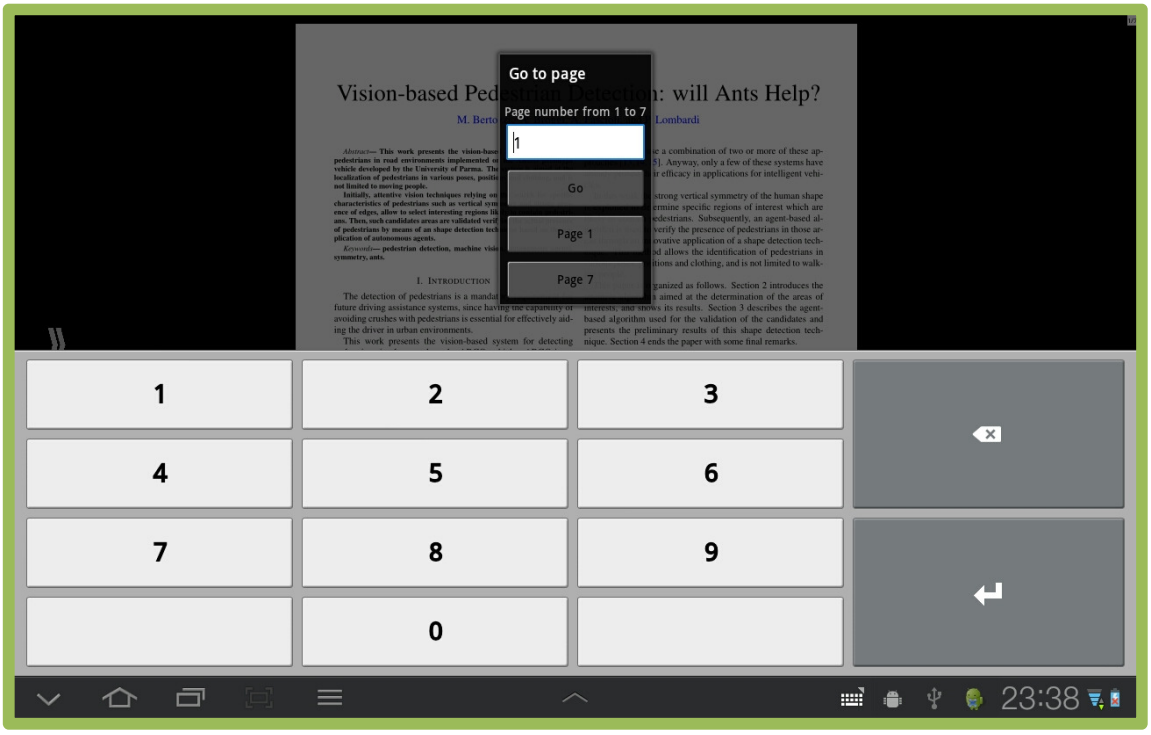


Figure 6.7 Screen capture of result of going to page

Search for paper (Phase 2)

A) SERVER SIDE –

The search of keyword is generally divided to four aspects: title, author, keyword and content.

Content-related searching

To search a term, stemming will be executed. On removing the suffix of the keyword, we look for similar word including the root of the word. For example, when “effective” is searched, it would be reduced into “effect”. All the words containing the root “effect” will be the targets for searching.

To capture the exception of the change in forms of words, such as “success” to “succeed”, we implement a small scale look-up table to handle these exceptions. A final “target term list” is finally generated to perform search in different aspects.

Title

The search for title is simply implemented by locating the terms within the title.

Keyword

The definition of “keyword” here is those keywords provided on the paper. The keyword is compared with the target terms to give results.

Content

The search by content is to analyze the key terms representing the paper besides the keyword list provided (if any). This is implemented by comparing the TF-IDF. The Term Frequencies of all the target keywords are picked from the database (which is pre-done when parsing the paper) and TF-IDF is then generated. The paper with higher TF-IDF value will be returned as result (Figure 6.8 and Figure 6.9).

- Pseudo code -

```
STEM into target_term_list
FOR all paper in repository
{  SUM the target terms count as TF
   CALCULATE TF-IDF        }
RETURN a list of paper with higher TF-IDF
```

Author-related Searching

Most of the recorded author names are in full name in principle. It is because the author names should be identical as the paper printed, on the manual parsing. However, to improve the accuracy of searching, we would compare the keywords to both the full name and short form on the author list (Figure 6.10).

- Pseudo code -

```
COMPARE the keyword and author list

IF (no result return)

{  MAKE keyword abbreviated

   COMPARE the keyword and author list  }
```

B) CLIENT SIDE –

Users can search the papers from the server with the following columns, including titles, keywords, contents and author. When the user search a word, the result of the paper list will be shown, which will show the papers that the title and keyword contains the word (Figure 6.8 and Figure 6.9), as well as the content contains it. For the content one, the one with higher TFIDF will be shown earlier (Figure 6.10). On the other side, if the user searches author, all the papers which are written by that author will be shown (Figure 6.11).

- Pseudo code -

```
IF (Search)
{       IF (Title or Keyword or Content)
        {       SEND request to web server for the result
                GET the result from web server
                IF (Title)
                {       SHOW the list of paper related to title     }
                ELSE IF (Keyword)
                {       SHOW the list of paper related to keyword       }
                ELSE IF (Content)
          {     SHOW the list of paper order by TFIDF  }        }
        ELSE IF (Author)
        {       SEND request to web server for the result
                GET the result from web server
                SHOW the list of paper written by that author    }        }
ELSE IF (show all)
{       SEND request to web server for the list of papers
        GET the result from web server
        SHOW the list of papers in the transparent panel        }
```

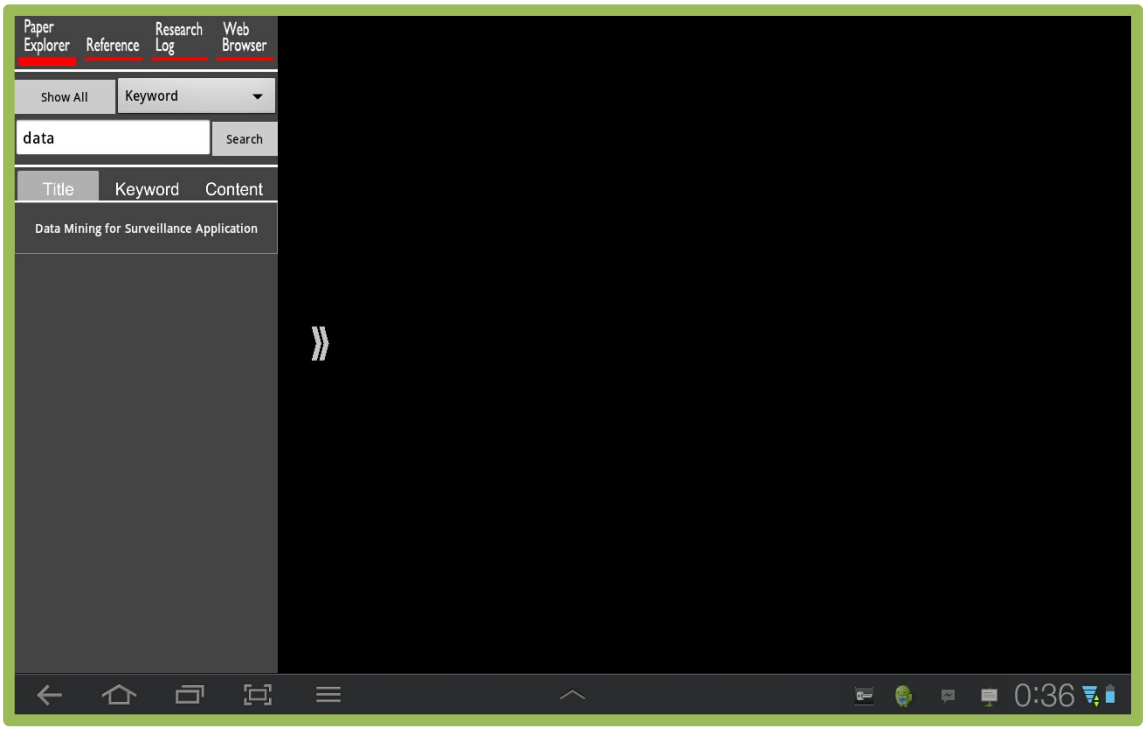


Figure 6.8 Screen capture of searching by title

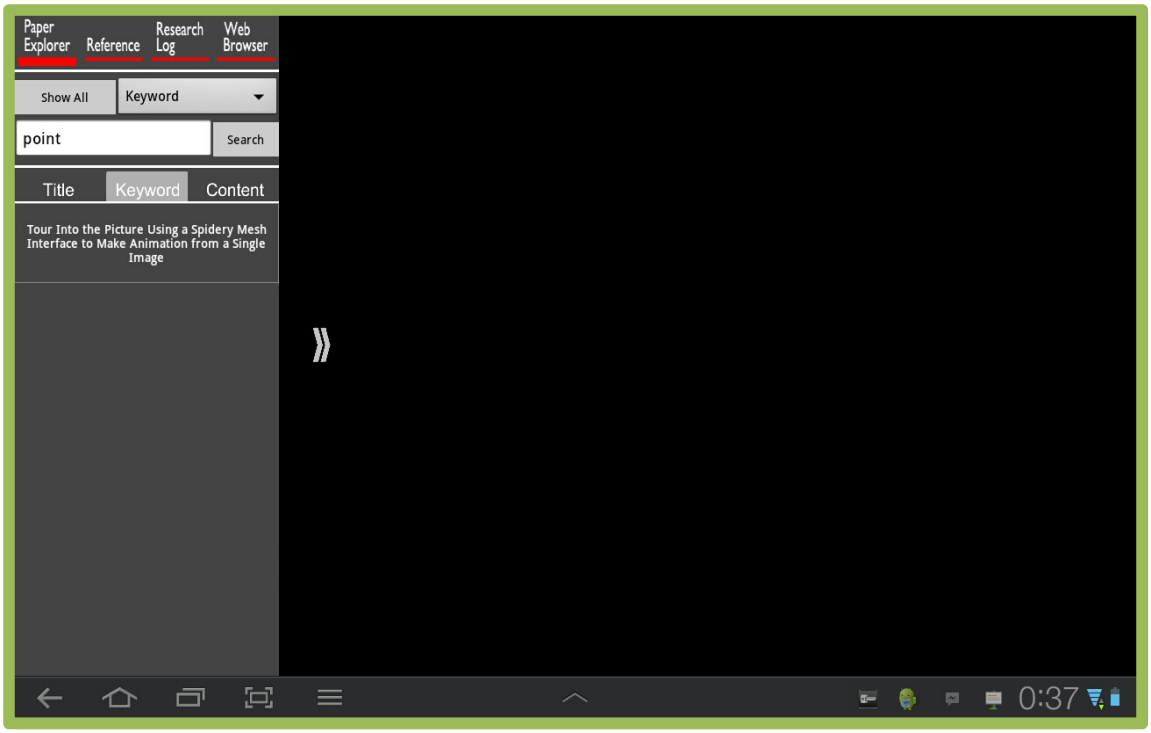


Figure 6.9 Screen capture of searching by keyword

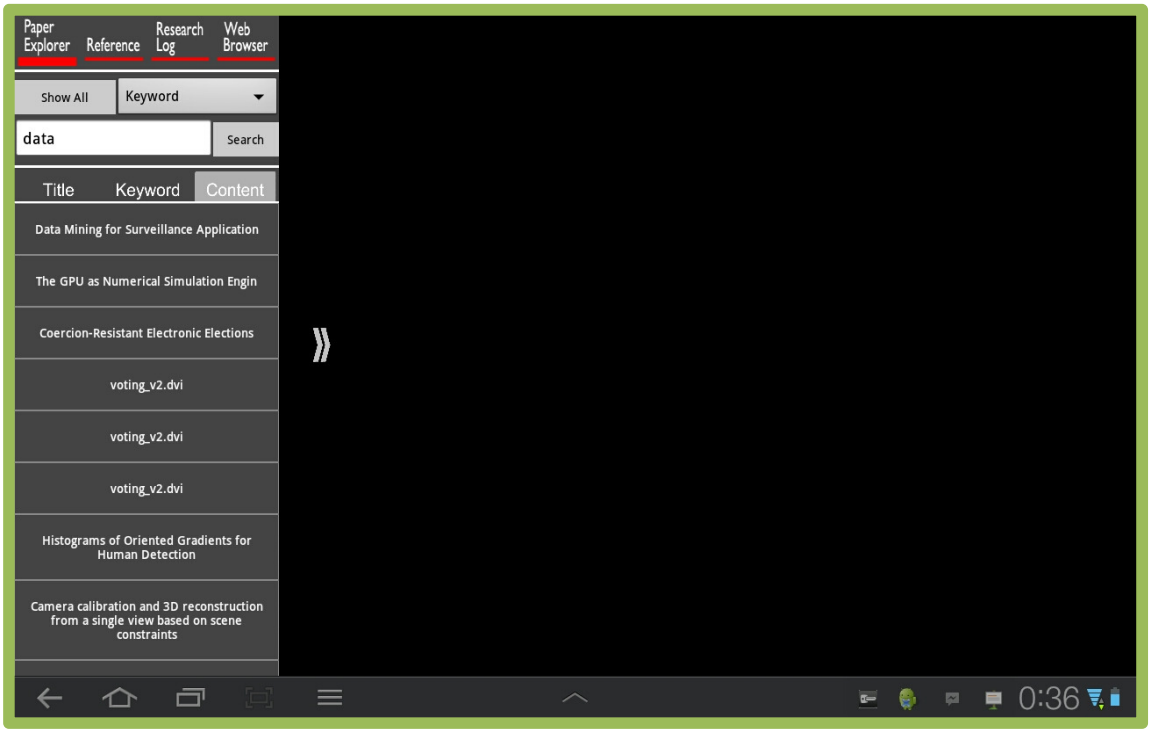


Figure 6.10 Screen capture of searching by content

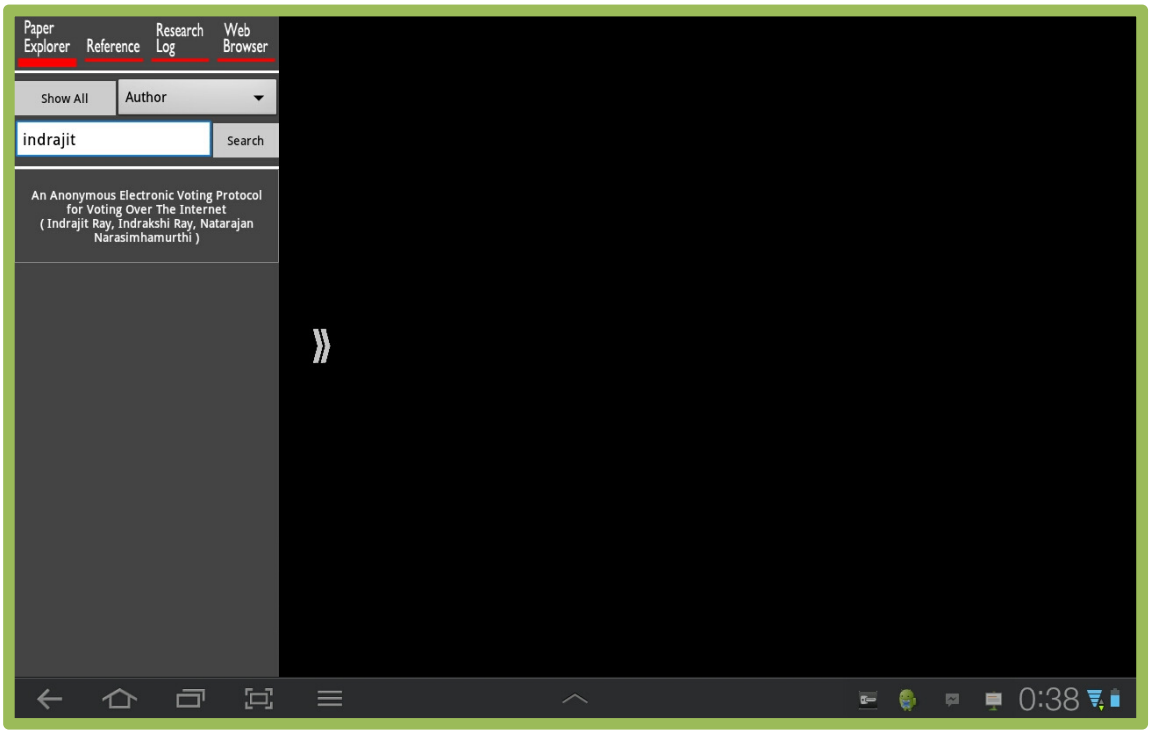


Figure 6.11 Screen capture of searching by author

Additional feature 2 (Phase2)

A) CLIENT SIDE –

The users can also choose to show or hide all the comments and reference buttons so as to convince his viewing. The two buttons are shown in the right hand side, when the user click the “show comments” button, all the comment buttons will be hidden (Figure 6.13), which will be shown after click (Figure 6.12). The reference one is the same as the comment one, which is using another button.

- Pseudo code -

```
IF (Click “show comments” button)
{  IF (showing all comments)
   {       HIDE all comment buttons }
   ELSE
   {       SHOW all comment buttons       }       }
IF (Click “show reference” button)
{  IF (showing all reference)
   {       HIDE all reference buttons }
   ELSE
   {       SHOW all reference buttons       }       }
```

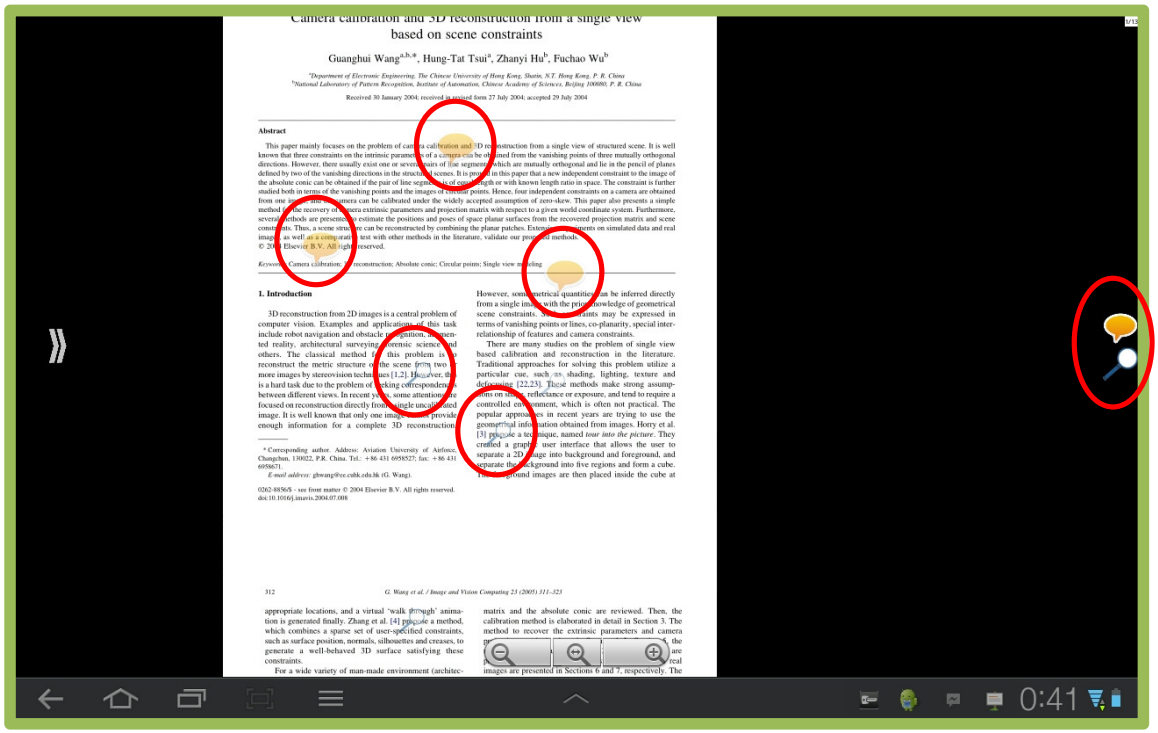


Figure 6.12 Screen capture of showing all buttons

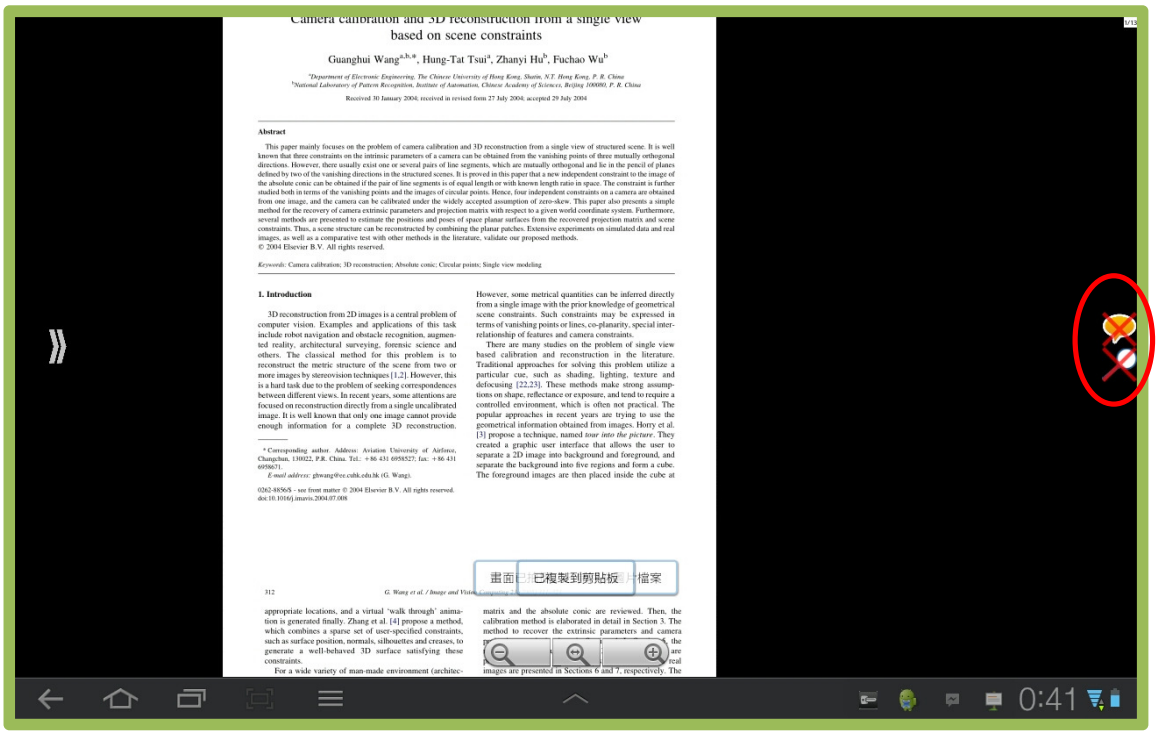


Figure 6.13 Screen capture of hiding all buttons

6.1.2. Paper Parsing

Get information needed (Phase 1 and Phase 2)

A) SERVER SIDE –

To begin with, once the paper is uploaded into the server, the server will call the "pdftotext" program (developed by Xpdf [13]) to print the text elements into the text file. A text file will be generated and will be used to parse the specific field of the paper.

In the section, the result of our survey conducted in chapter 3 should be set as reference. To implement the parsing, we summed up for several methods of parse:

1) Search for the pattern of the target information.
2) Search for the keywords of the target information
3) Search for the start of the target, ie. what is expected to be written before the target.
4) Search for the end of the target, ie. what is expect to appear after the end of the target.

One or more above methods will be used in parsing different fields.

Now we would like to show the implementation one by one.

Title -

Using the result analyzed, there are seldom the "title" term to indicate the title. Also, the titles may not be first line of the paper and they may stay in more than one line. Extracting the first line or some lines are not possible, because it would be more difficult for the computer to identify whether the line following is the authors' name or just some technical proper noun.

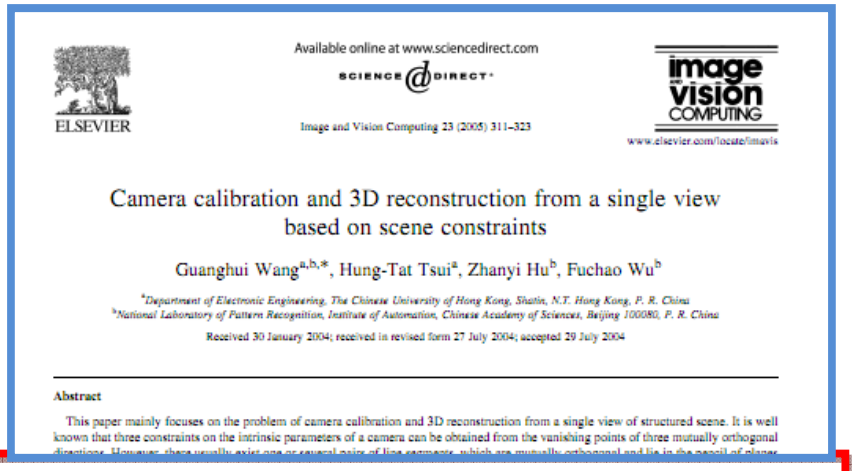
Based on these, we try to get the font size of all the text elements and compare them (implementation method 1 – pattern recognition). We can quite sure that the font size of title is different from the header and the author's information. We found that the title is usually in the biggest size. Moreover, to make sure that the size got is the title's one, few more rules should be set. For example. we compare the size of the fonts on the first page only.

- Pseudo code -

```
WHILE ( is on page one)
{       GET font size of each line
        GET the largest font size
        GET the line number(s) of the largest font appears}
OPEN the txt file
COPY the text in the line(s) of the largest font appears
UPDATE the database with title got
RETURN the title found
```

The title of the paper will be return in expect.

- Value Return - Example -



Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 23 (2005) 311–323
ELSEVIER
image and vision COMPUTING
www.elsevier.com/locate/imavis

Camera calibration and 3D reconstruction from a single view based on scene constraints

Guanghui Wang[a,b,*], Hung-Tat Tsui[a], Zhanyi Hu[b], Fuchao Wu[b]

[a]Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T. Hong Kong, P. R. China
[b]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, P. R. China

Received 30 January 2004; received in revised form 27 July 2004; accepted 29 July 2004

Abstract

This paper mainly focuses on the problem of camera calibration and 3D reconstruction from a single view of structured scene. It is well known that three constraints on the intrinsic parameters of a camera can be obtained from the vanishing points of three mutually orthogonal

```
Image and Vision Computing 23 (2005) 311â€“323 www.elsevier.com/locate/imavis

Camera calibration and 3D reconstruction from a single view based on scene constraints
Guanghui Wanga,b,*, Hung-Tat Tsuia, Zhanyi Hub, Fuchao Wub
b a Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin,
N.T. Hong Kong, P. R. China National Laboratory of Pattern Recognition, Institute of
```

Camera calibration and 3D reconstruction from a single view based on scene constraints

Author -

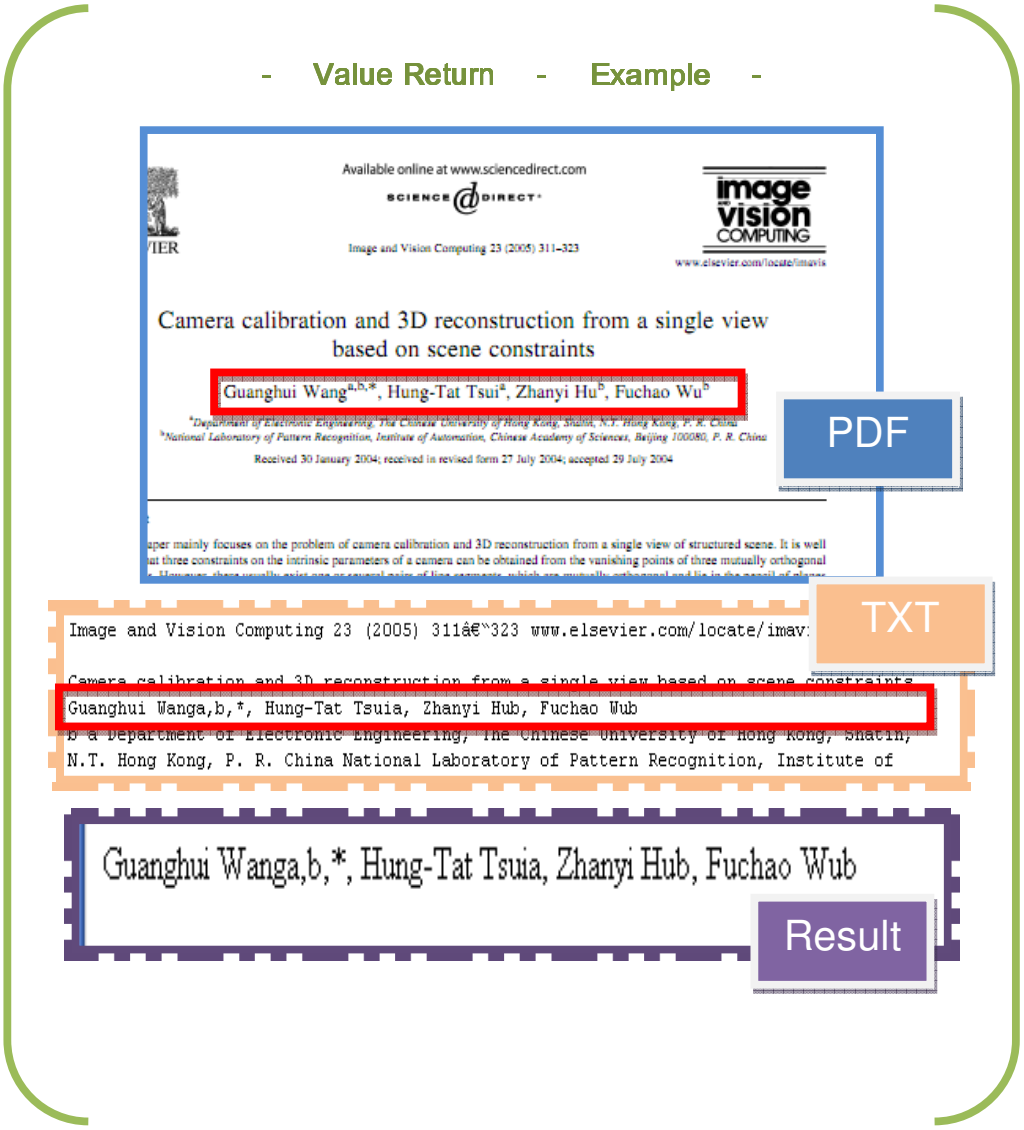
According to the statistics, most of the papers are written by two or more authors. The authors' names are usually grouped in one line right after the title. There are often two to five lines descriptions and contact information follows the authors' names. We found that it is an extremely complex task to search for the next line of authors' names, because there is no any standard and pattern for the description. The description may include the title of the authors or the organization they belong to. Therefore, the implementation method 4 (search for the end) is not feasible. The lack of uniformity is also found in the authors' name, and hence the recognition of pattern for the names is not possible too.

To assist the searching process, we also make a list of short form of authors' name in order to speed up the time of comparison.

Finally, we decided to make the method simple. We would like to simply parse the line after the title (implementation method 3 – search for start) because most of cases the authors' names appear right after the title.

- Pseudo code -

```
WHILE (title is found)
{       GET the next line AS authors }
UPDATE the database with authors got
RETURN authors
```


- Value Return - Example -
Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 23 (2005) 311–323
image vision COMPUTING
www.elsevier.com/locate/imavis
Camera calibration and 3D reconstruction from a single view based on scene constraints
Guanghui Wang[a,b,*], Hung-Tat Tsui[a], Zhanyi Hu[b], Fuchao Wu[b]
Received 30 January 2004; received in revised form 27 July 2004; accepted 29 July 2004
PDF
Image and Vision Computing 23 (2005) 311â€“323 www.elsevier.com/locate/imav
Guanghui Wanga,b,*, Hung-Tat Tsuia, Zhanyi Hub, Fuchao Wub
b a Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T. Hong Kong, P. R. China National Laboratory of Pattern Recognition, Institute of
TXT
Guanghui Wanga,b,*, Hung-Tat Tsuia, Zhanyi Hub, Fuchao Wub
Result

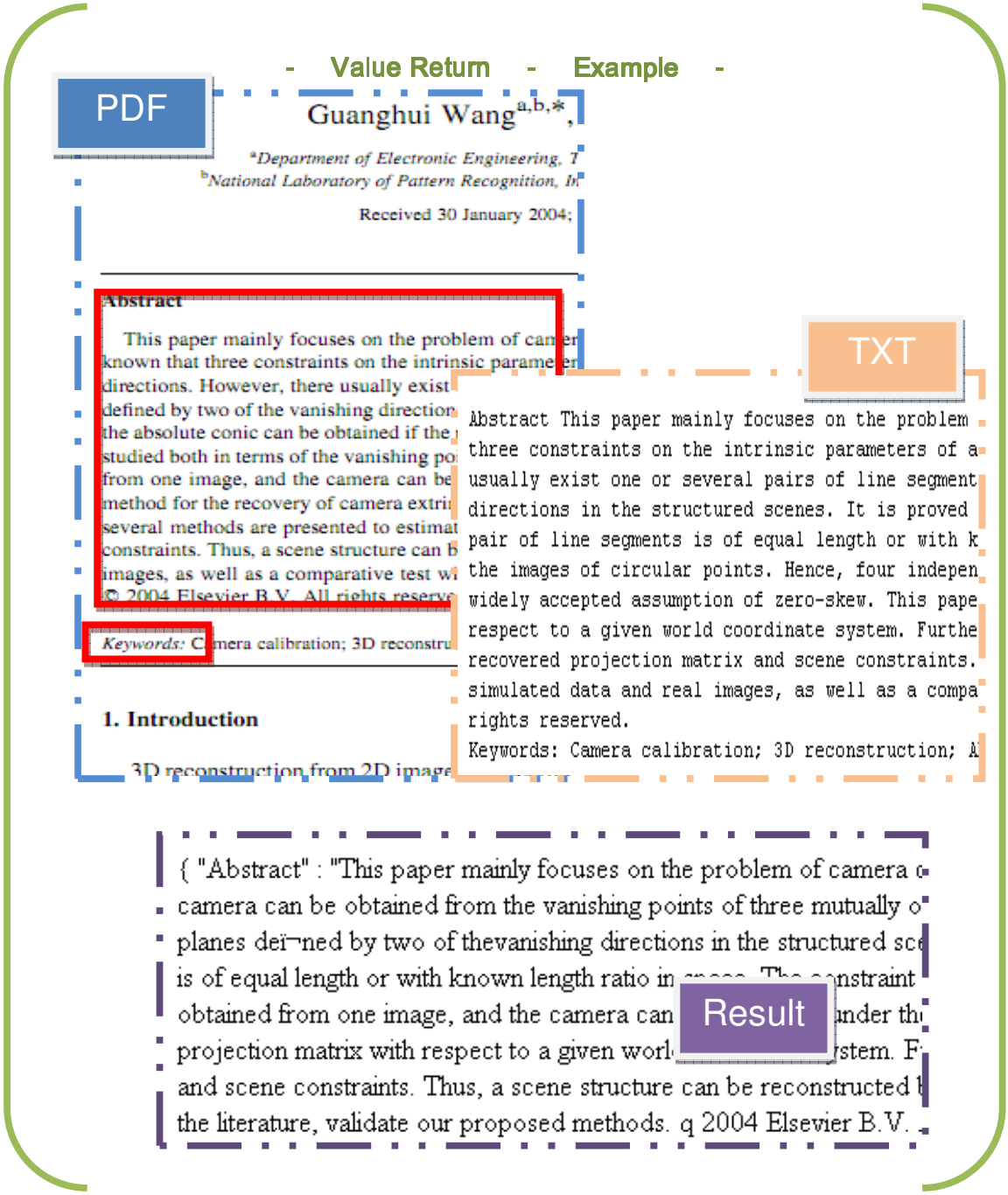
Abstract -

Usually the abstract is written after the appearance of the "Abstract" word, with reference to the survey result. Therefore it is easy to locate where the abstract starts (implementation method 2 – keywords location). However, sometimes the abstract are in more than a paragraph. To make sure the scope of abstract can be extract correctly, we may also check where is the end point of the abstract. Usually the introduction, sometimes the keywords column, follows the abstract. Therefore the scope ends before the line which contains the "keywords" or "introduction/summary" (implementation skill 4 – search for the end).

- Pseudo code -

```
WHILE (a line is parsed)
{       IF ("Abstract" is found)
        {       GET every line AS abstract
                UNTIL "Keywords" or "Introduction" appears } }
UPDATE the database with abstract got
RETURN abstract in a josn
```

Since the abstract may consist of several lines or paragraph, it will be encoded as json array to ensure correct data transmission.



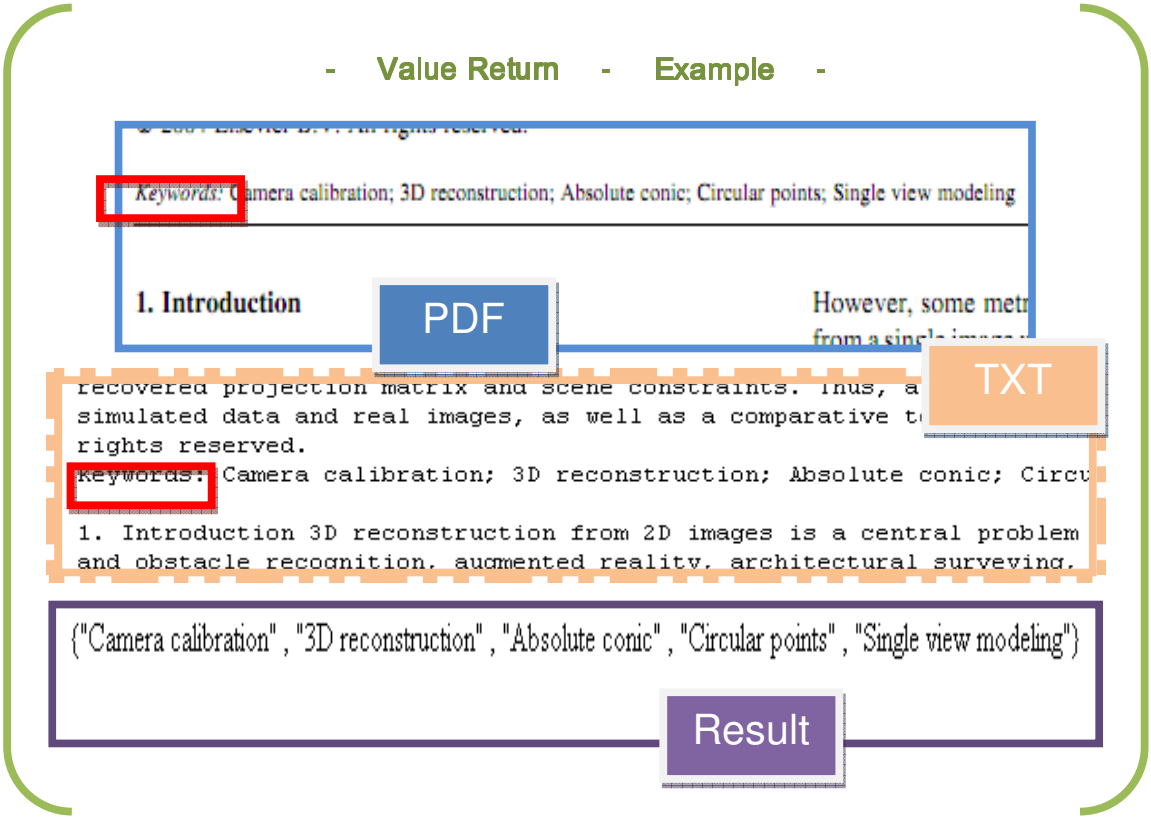
Keywords -

If there is any keyword, most of them appear after several terms indicate the keywords. The terms indicate keywords varies, like “Keywords”, “key terms”, “Index terms” and “General terms”. All of them should be checked. Similar to the case of “Abstract”, the keywords can be found by locating the specific terms and end by the presence of “Introduction” (implementation methods 2 & 4). More to note that, each keyword or key phrase are usually separated by “,” or “;”.

- Pseudo code -

```
WHILE (a line is parsed)
{       IF (“Keywords” is found)
        {       GET Keywords SPLIT BY “,” or “;”
                UNTIL “Introduction” appears } }
UPDATE the database with Keywords got
RETURN Keywords in json
```

Since there will be a list of keywords, it will be encoded as json array to ensure correct data transmission.



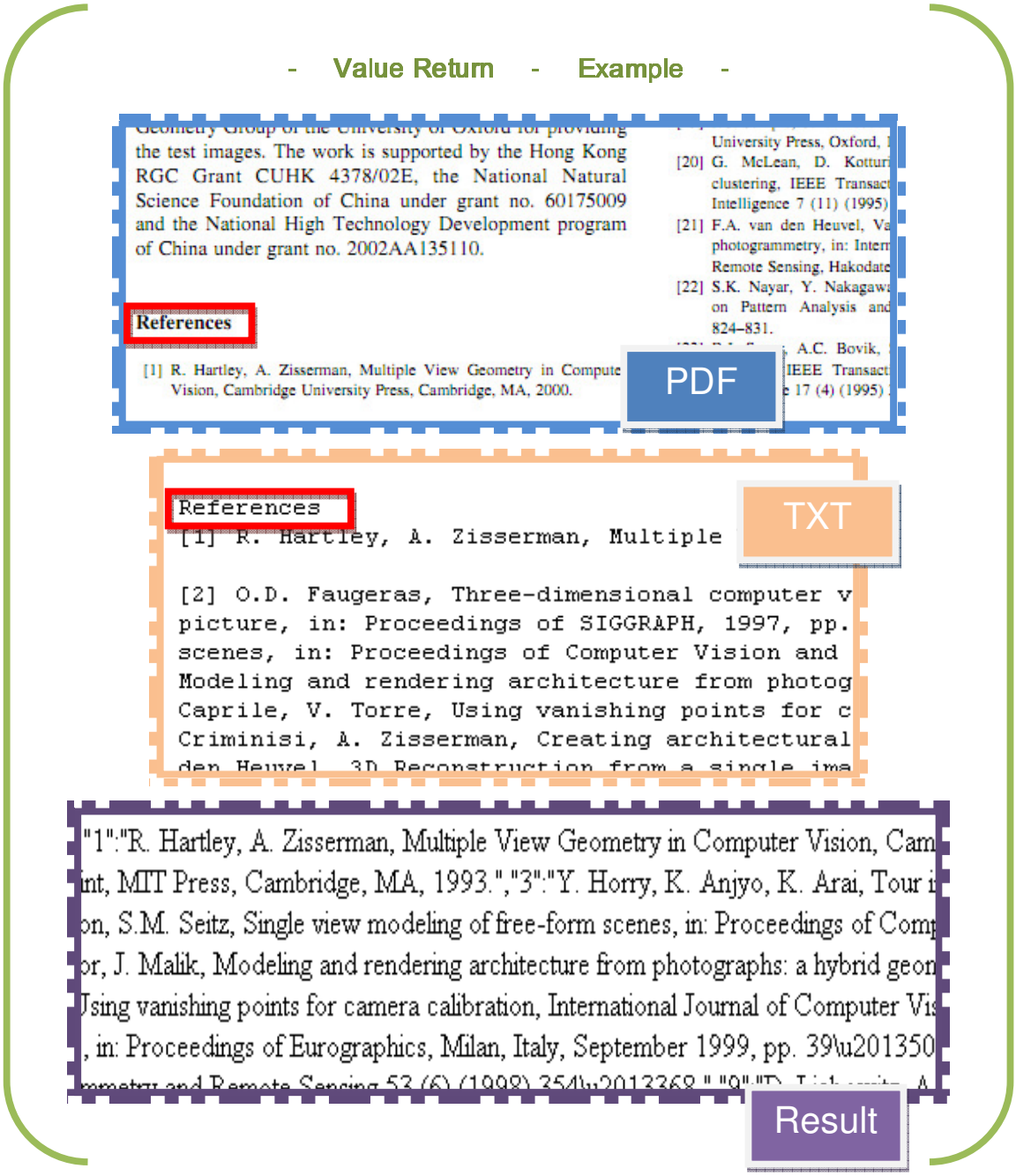
References -

The references appear 100% after the word “REFERENCES” / “BIBLIOGRAPHY”. According to survey, 90% of references have shown in the index form [ ]. Some shows in the form “1. ” while the rest have only the reference items without index. In phase 1, we would like to parse the type of [ ] and those without indexes.

After locating the “References”, we will check if “[“ appears. If “[“ appears and is followed by an index (sometimes may not only digits but also alphabets) and “]”, we record the index with relevant reference item. If there is no “[“ appears within 10 alphabets, we record each line as one reference item.

- Pseudo code -

```
WHILE (a line is parsed)
{       IF (“References” / “Bibliography”is found)
        {       IF (“[“ is found within 10 alphabets]
                {       GET index within [ ]
                        GET relevant reference item }
                ELSE
                {       GET a line AS a reference item } }
UPDATE the database with References got
RETURN References in json
```


- Value Return - Example -
the test images. The work is supported by the Hong Kong RGC Grant CUHK 4378/02E, the National Natural Science Foundation of China under grant no. 60175009 and the National High Technology Development program of China under grant no. 2002AA135110.
References
[1] R. Hartley, A. Zisserman, Multiple View Geometry in Compute Vision, Cambridge University Press, Cambridge, MA, 2000.
PDF
References
[1] R. Hartley, A. Zisserman, Multiple
[2] O.D. Faugeras, Three-dimensional computer v
picture, in: Proceedings of SIGGRAPH, 1997, pp.
scenes, in: Proceedings of Computer Vision and
Modeling and rendering architecture from photog
Caprile, V. Torre, Using vanishing points for c
Criminisi, A. Zisserman, Creating architectural
TXT
"1":"R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cam
int, MIT Press, Cambridge, MA, 1993.","3":"Y. Horry, K. Anjyo, K. Arai, Tour i
on, S.M. Seitz, Single view modeling of free-form scenes, in: Proceedings of Comp
or, J. Malik, Modeling and rendering architecture from photographs: a hybrid geon
Using vanishing points for camera calibration, International Journal of Computer Vis
, in: Proceedings of Eurographics, Milan, Italy, September 1999, pp. 39\u201350
Result

For each item of bibliography, we try to analyze and split the information into pieces in order to build up linkage with the other paper easily. Our main objective is to obtain the authors' name and the title from each item.

As the research result shown, the author names always go first. The author names are split by either "," or "and". If the author names are recorded in short form, we look for the first word after ", / and" without "." as the second character. That would be the first word of the title. If the author names are written in long form, we look for the double quotation mark indicating the start of the title.

- Pseudo code -

```
SPLIT the item into pieces by ","
IF (the author name is short form)
{       IF (the second character of the piece == ".")
        {       GET piece as one author's name }
        ELSE
        {       GET piece as Title  }       }
ELSE
{       WHILE (true)
        {       GET piece as author's name
                IF (double quotation mark is detected)
                {       BREAK       }
                GET the piece as title      }       }
```

Term Frequency -

Term frequency is parsed to achieve the purpose of searching. The whole document would be split into terms and the frequency of appearance is counted. The term counts are then inserted into the database and hence a “Dictionary” is built.

To minimize the size of the database, we adopt a stop word list to filter the terms with low searching value. If the term is not on the stop word list (Figure 6.14), it becomes valid to have term count in the dictionary.

- Pseudo code -

```
SPLIT document into terms
IF (not a stop word)
{       COUNT term frequency
        FOR (each term)
        {       IF (term appears in dictionary)
                {       INSERT term count }
                ELSE
                {       INSERT new term
                        INSERT term count }       }       }
```

able	almost	anyone	available
about	alone	anything	away
above	along	anyway	awfully
abroad	alongside	anyways	back
according	already	anywhere	backward
accordingly	also	apart	backwards
across	although	appear	be
actually	always	appreciate	became
adj	am	appropriate	because
after	amid	are	become
afterwards	amidst	aren't	becomes
again	among	around	becoming
against	amongst	as	been
ago	an	a's	before
ahead	and	aside	beforehand
ain't	another	ask	begin
all	any	asking	behind
allow	anybody	associated	being
allows	anyhow	at	believe

Figure 6.14 Table of stop word list

Check duplication (Phase 2)

A) SERVER SIDE –

When a new file is uploaded, compare the title and the authors of newly-added paper to the existing database. Once duplication is found, a content checking will be performed. If differences are detected, the title of the newly-added paper will be updated as "ver. 2" or above. Otherwise the late coming paper will be dropped.

- Pseudo code -

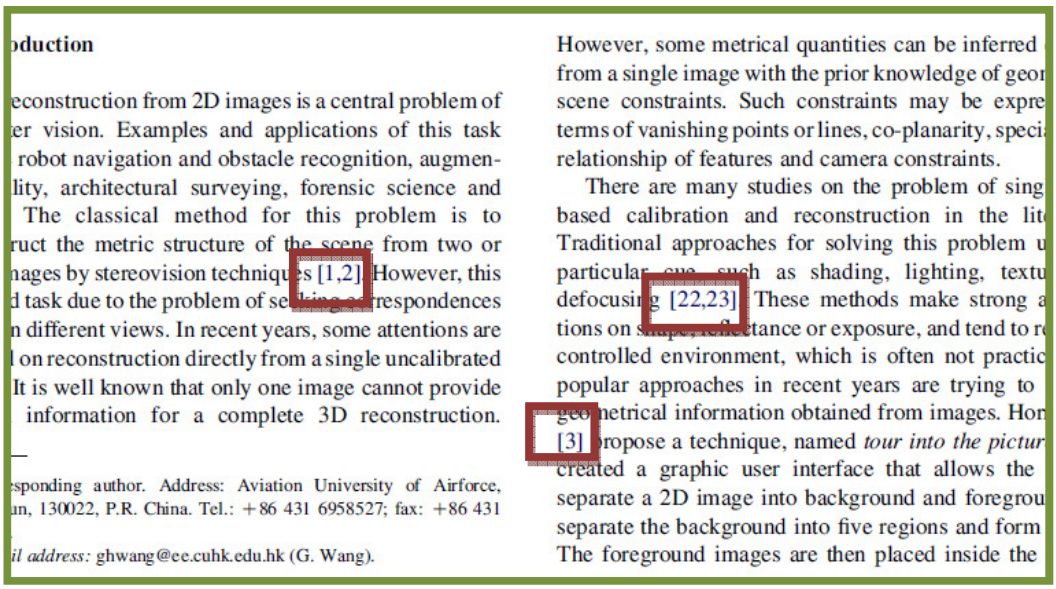
```
IF (is Title duplicated)
{       IF (are authors duplicated)
            {       IF (difference detected)
                    {       UPDATE as new version   }
                    ELSE
                    {       DROP the newly-added file       }       }       }
```

6.1.3. Referencing

Show the reference (Phase 1)

A) SERVER SIDE –

To assist the search of reference indexes and reduce the workload of the client side, the server will parse the whole paper and get a list of reference indexes throughout the passage. For example (Figure 6.15):


oduction

econstruction from 2D images is a central problem of
er vision. Examples and applications of this task
robot navigation and obstacle recognition, augmen-
lity, architectural surveying, forensic science and
The classical method for this problem is to
ruct the metric structure of the scene from two or
nages by stereovision techniques [1,2]. However, this
d task due to the problem of seeking correspondences
n different views. In recent years, some attentions are
l on reconstruction directly from a single uncalibrated
It is well known that only one image cannot provide
information for a complete 3D reconstruction.

sponding author. Address: Aviation University of Airforce,
un, 130022, P.R. China. Tel.: +86 431 6958527; fax: +86 431

il address: ghwang@ee.cuhk.edu.hk (G. Wang).

However, some metrical quantities can be inferred
from a single image with the prior knowledge of geo
scene constraints. Such constraints may be expre
terms of vanishing points or lines, co-planarity, speci
relationship of features and camera constraints.

There are many studies on the problem of sing
based calibration and reconstruction in the lit
Traditional approaches for solving this problem u
particular cue such as shading, lighting, textu
defocusing [22,23]. These methods make strong a
tions on shape, reflectance or exposure, and tend to r
controlled environment, which is often not practic
popular approaches in recent years are trying to
geometrical information obtained from images. Hor
[3] propose a technique, named *tour into the pictur*
created a graphic user interface that allows the
separate a 2D image into background and foregrou
separate the background into five regions and form
The foreground images are then placed inside the

Figure 6.15 Photo showing the references inside the paper

Then the [1,2] ,[22,23] and [3] will be parse into the reference appearance list. The list will be returned to the client directly when client request.

B) CLIENT SIDE –

The client device will first ask the web server to send the list of references to it. The list of references (Figure 6.16) will be shown in the transparent panel in the left hand side of the screen. Furthermore, reference buttons will be shown inside the paper.

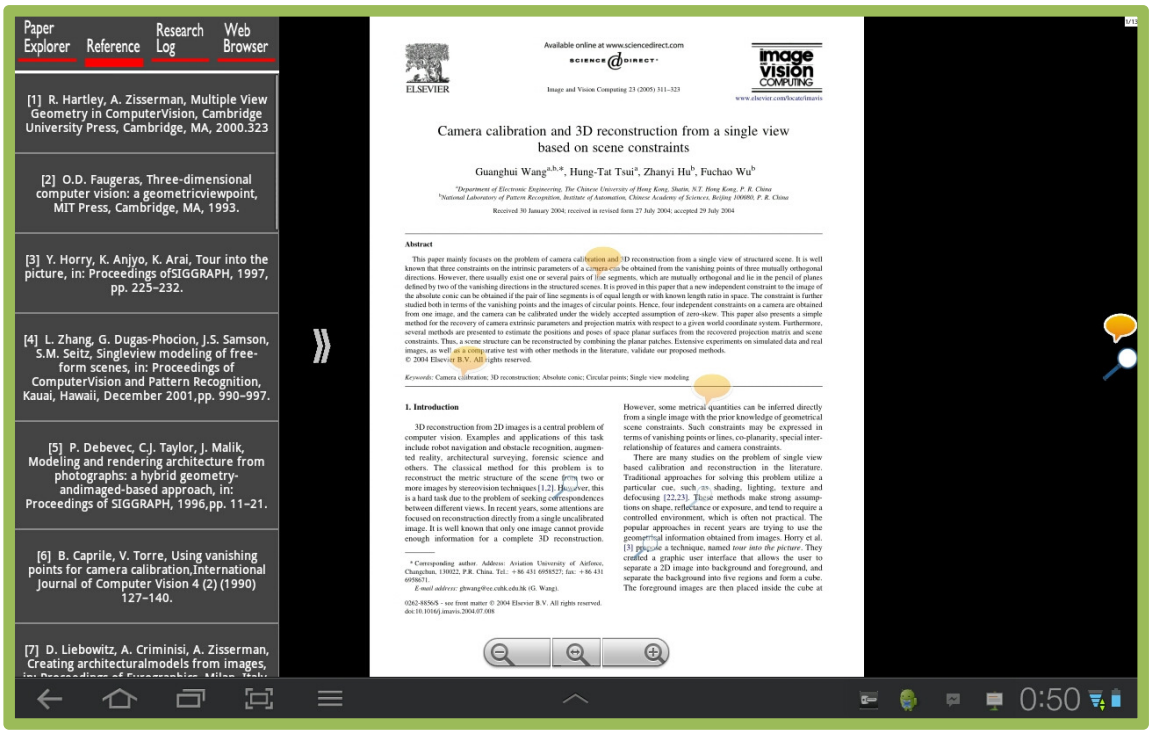


Figure 6.16 Screen capture of reference list

Build linkage to other paper (Phase 2)

A) SERVER SIDE –

For each time a request received for linkage with other paper, server will look for the database to check if records are available for referencing. If no record is found, a search will be invoked. The search on the title and the authors within the bibliography will be performed. Once that target paper can be found in the paper repository, it will be recorded in the database for future use. Linkage than can be built.

If no paper can be found in the paper repository, the searching process will be led to online searching under the user permission (which will be discussed in the section of “library renewal”)

- Pseudo code -

```
CHECK database for record
IF (no record)
{       SEARCH with title and author
        IF (target paper found)
        {       RECORD to database
                RETURN target paper     }       }
```

B) CLIENT SIDE –

The user can click on the buttons of references on the paper to show what the reference is. After clicking, the information of the reference will be shown (Figure 6.17). If the server has this paper, the user can link to it for further reading. Else, the user can search that paper title on google in the web browser inside the client application (Figure 6.18 and Figure 6.19).

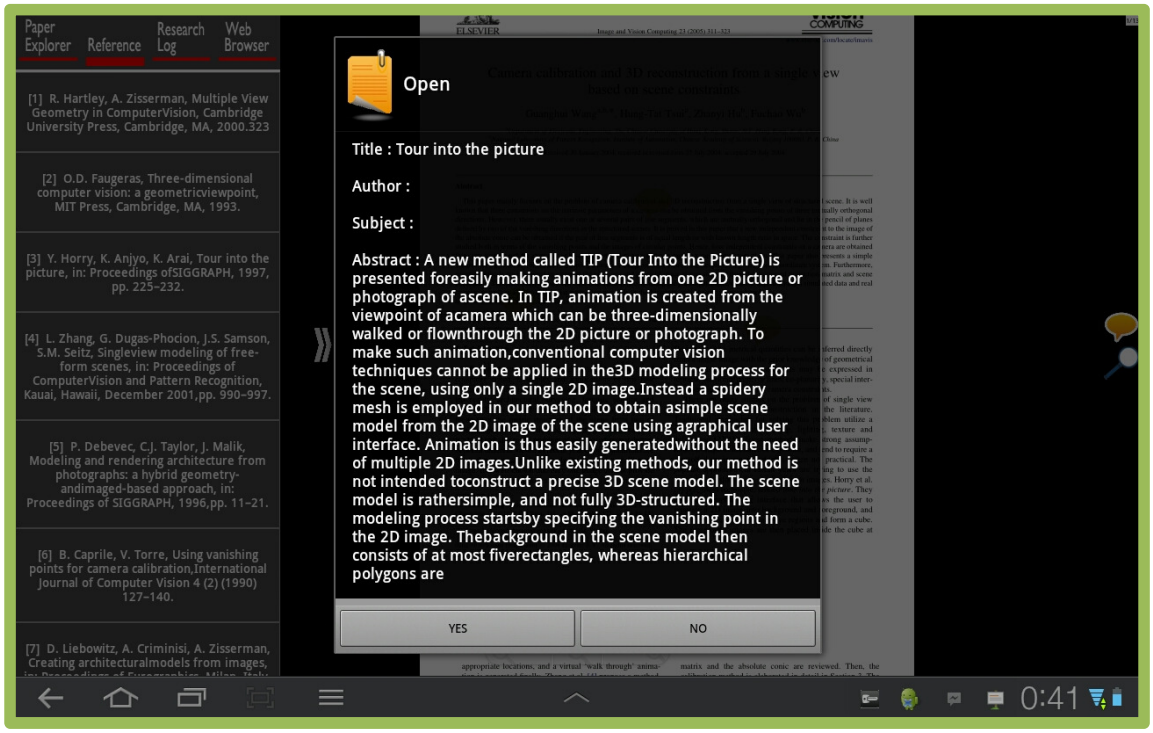


Figure 6.17 Screen capture of showing reference information

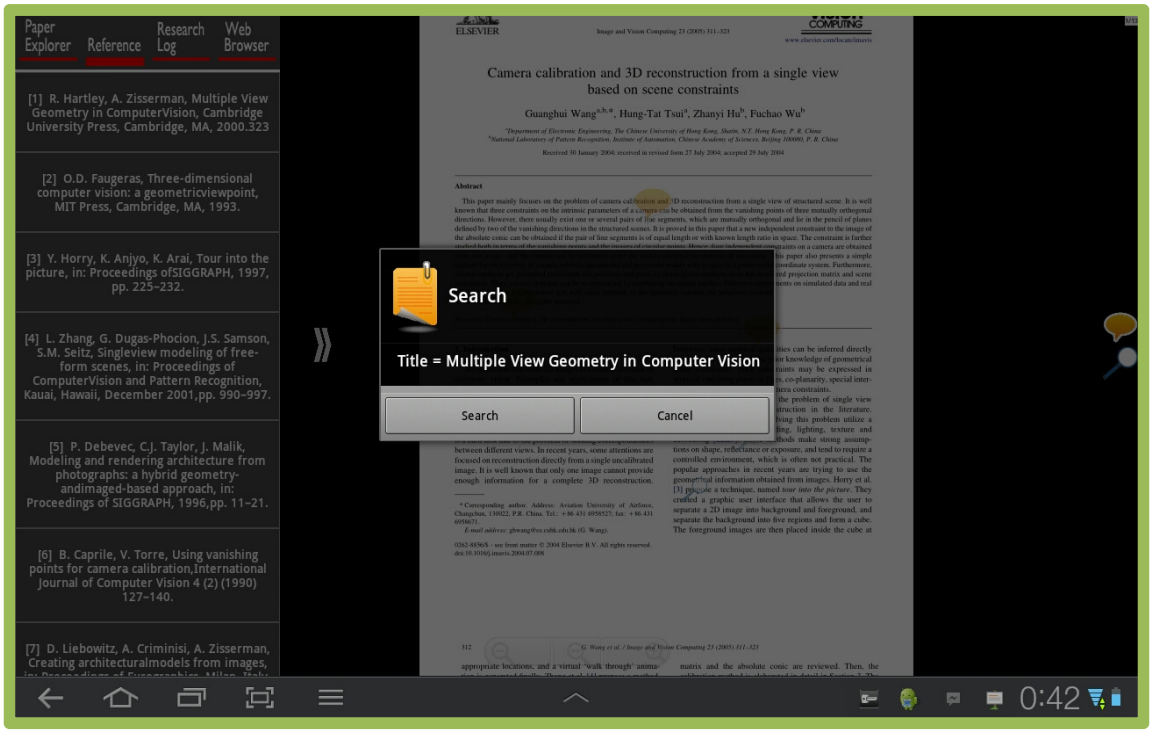


Figure 6.18 Screen capture of suggesting users to search the paper on google

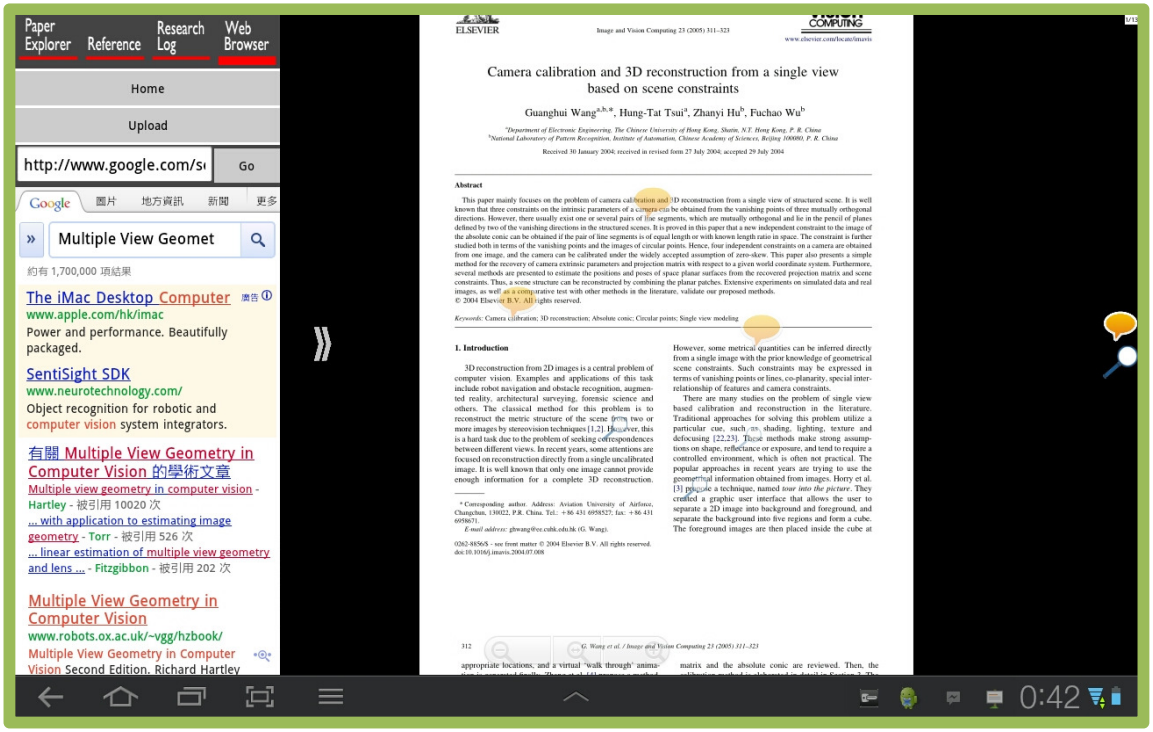


Figure 6.19 Screen capture of searching the paper on google

If the users click the reference buttons inside the paper, the information of the reference paper will also be shown (Figure 6.20). The function have also been implemented.

- Pseudo code -

```
IF (click one of reference buttons)
{  SHOW the information of that reference
   IF (server has the paper)
   {       LINK to the certain paper   }
   ELSE
   {       SEARCH on google inside the application       }       }
```

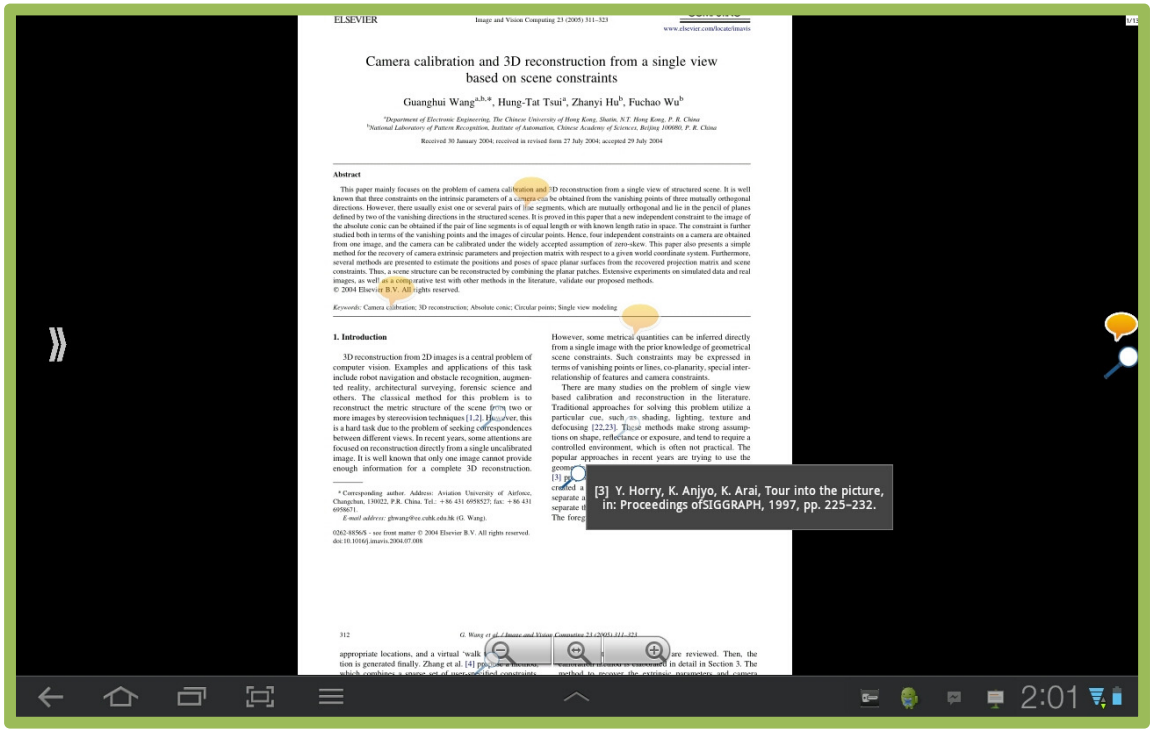


Figure 6.20 Screen capture of clicking on a reference button inside the paper

6.1.4. Library renewal

Manage the library manual (Phase 1 and Phase 2)

A) SERVER SIDE –

During the early development, a viewable managing GUI platform is provided for the administrator to use.

There is a drop box for the admin to drag the PDF file and upload it. Once the file is dropped and successfully uploaded, a new row will be inserted at the top. The txt file will be automatically generated for parsing. The newly uploaded file will be parsed too. On the panel, admin can delete the paper too.

B) CLIENT SIDE –

The users can upload a paper from its device to the web server (Figure 6.21). After uploading the paper, the user can view the new paper and correct the information including title, author, subject and abstract which is parsed by the server (Figure 6.22).

- Pseudo code -

```
IF (choose upload)
{  CHOOSE one paper inside own device
   SHOW the new paper
   CORRECT the information }
```

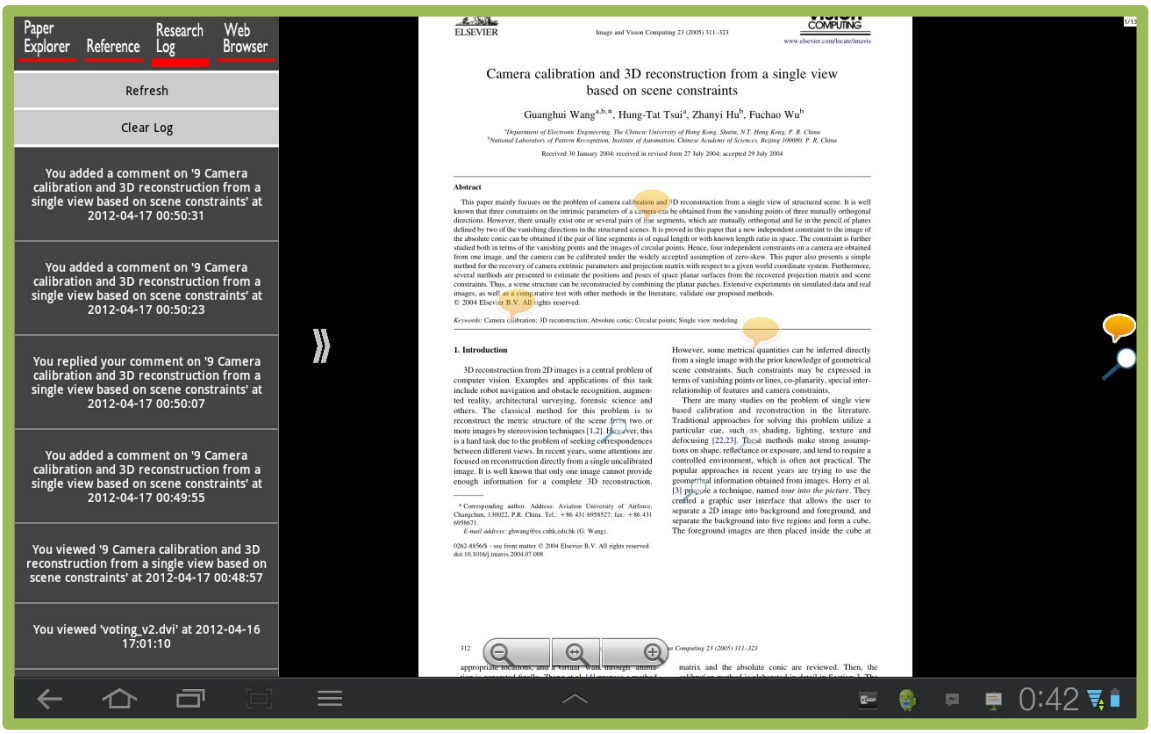


Figure 6.21 Screen capture of choosing one local paper to upload

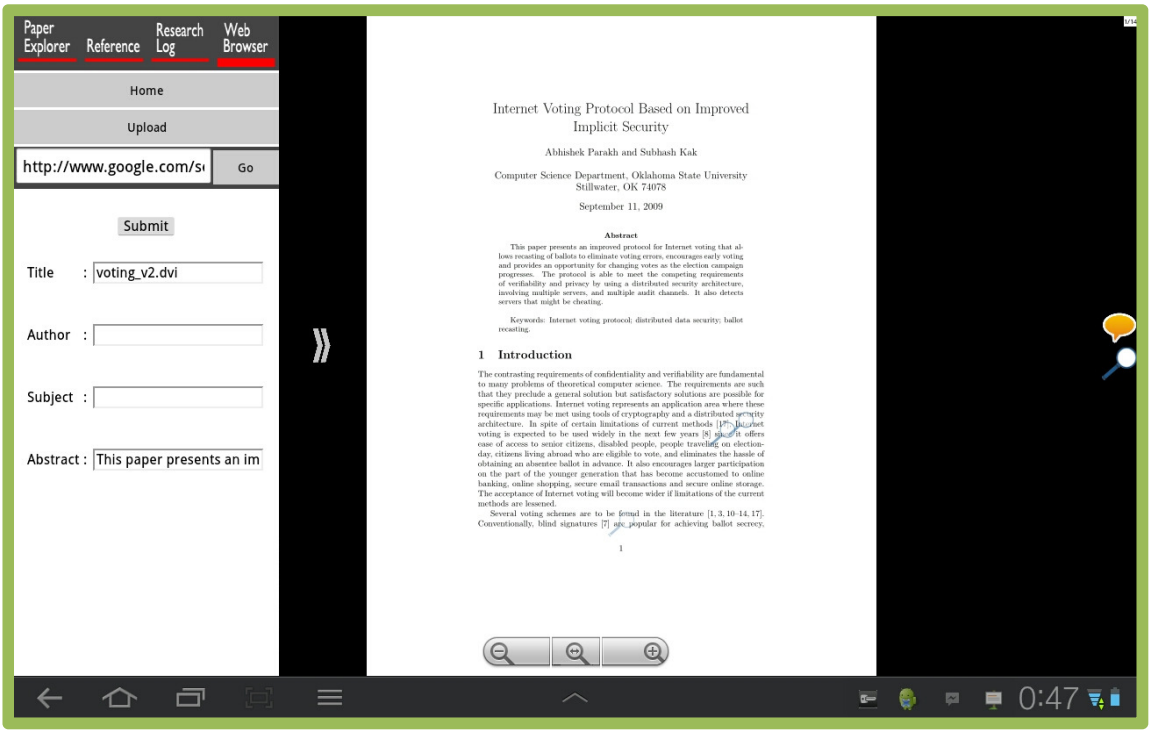


Figure 6.22 Screen capture of correcting the information of the new paepr

Embed a browser (Phase 2)

A) SERVER SIDE –

A framework of webpage is prepared while user is going to search for new papers. When user has successfully found the target paper online, he can just simply upload the paper to the webpage and help to fill in the data. The paper and the data submitted will be inserted into the database.

B) CLIENT SIDE –

A web browser has been embedded into the client application (Figure 6.23) for the users to download paper and upload it to the web server.

- Pseudo code -

SHOW “www.google.com” as home page

CLICK “HOME” to go back to home page

ENTER certain URL will go to that web page

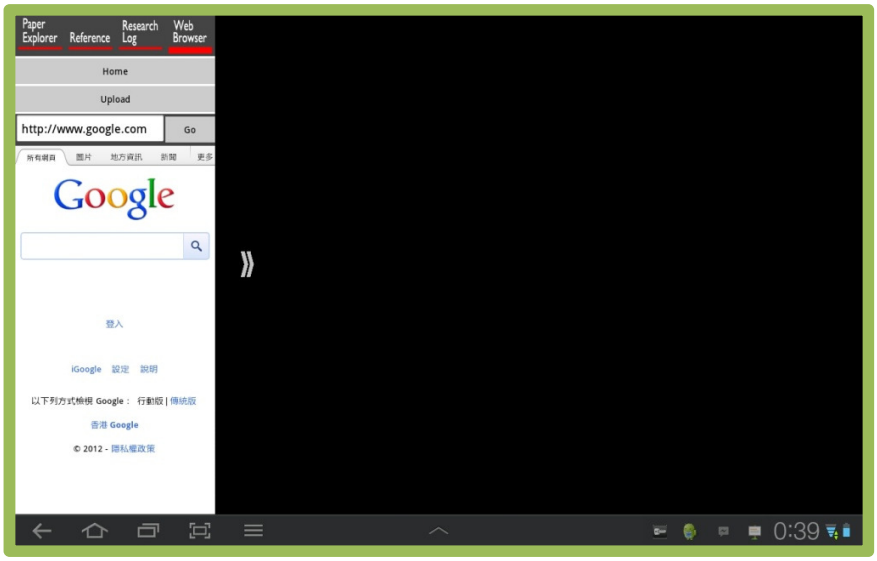


Figure 6.23 Screen capture of clicking on a reference button inside the paper

6.2. Module 2: Communication

6.2.1. Posting comments

Add comments (Phase 1)

A) SERVER SIDE –

On adding the comment, the client sends the user ID, paper ID, page number, positions and the comment to the server. Server will automatically generate a comment ID and the update time for inserting record into database.

B) CLIENT SIDE –

The user can add comments on the paper by long-pressing on the screen (Figure 6.24). The client device will then create a comment button at that location, and ask the user to input the comment. After the user input the comment, the client device will send some information, including location of the comment, content of the comment, paper ID, user ID, etc. , to the web server to update the database.

- Pseudo code -

```
IF (add comment)
{  CHECK the location
   ASK the user to input the comment
     CREATE a new button for comment
   SEND the information to the web server }
```

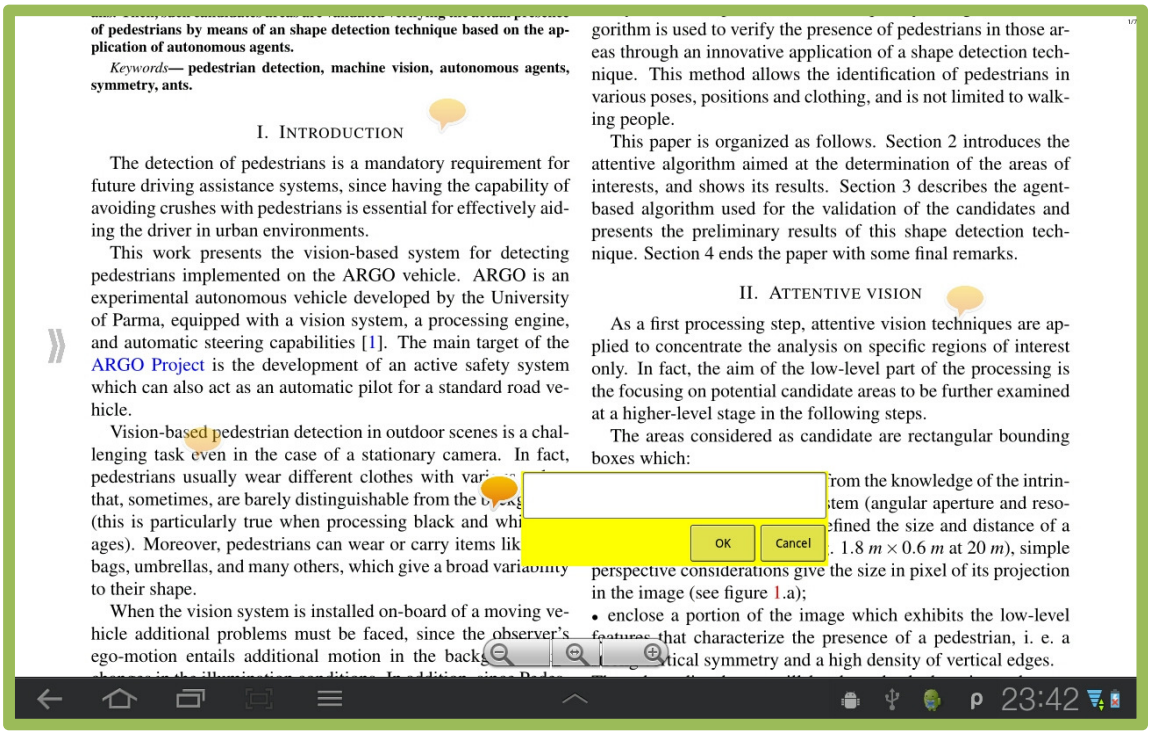


Figure 6.24 Screen capture of adding comment

Show comments (Phase 1)

A) SERVER SIDE –

On deleting the comment, the client sends the paper ID, page number and position as request. Server will check return a list of usernames, comments and comment IDs from database to the client.

B) CLIENT SIDE –

Before showing a paper, the client device will ask the web server for the previous comments of that paper. The client device will then create the comment button at the corresponding location (Figure 6.25), and load the previous comment content. The user can read the comment added by himself or others. Besides, the comment buttons are translucent so that not to block the text of the paper.

- Pseudo code -

```
SHOW paper

CREATE buttons for all previous comments

IF (click on a comment button)

{  SHOW the previous comment contents

   SHOW the button of reply and delete      }
```

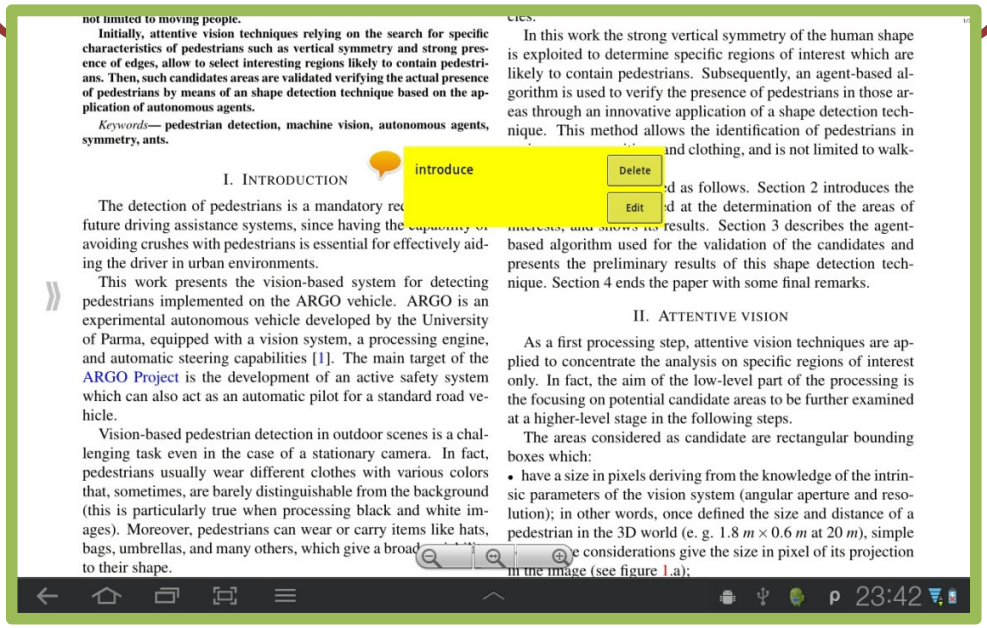


Figure 6.25 Screen capture of showing comment

Reply comments (Phase 1)

A) SERVER SIDE –

The client sends the user ID, paper ID, page number, positions and the comment to the server. By checking the database, server will recognize that there is a discussion invoked on the same point before. It will hence add the comment with an additional entry "ReplyTo", which records the comment ID the new comment replies to.

B) CLIENT SIDE –

User can reply on any comment added by himself or others. After replying, the message will be sent to the web server to update the database.

- Pseudo code -

```
IF (click reply button)
{  ASK user to input the reply message
   SEND the information to the web server }
```

Delete comments (Phase 1)

A) SERVER SIDE –

The client sends the comment ID to the server. With reference to the database, if the CID is the first comment invoking the whole discussion, all the comments of the same discussion will be deleted. Yet, only the comment of that CID will be deleted. Meanwhile, the other comment which have “ReplyTo” points the deleted CID with be redirected to the previous comment that the reply to point to.

B) CLIENT SIDE –

User can delete his own comments and replies, the information will be sent to the web server to update the database.

- Pseudo code -

```
IF (delete comment)
{  SEND the information to the web server }
IF (delete reply)
{  SEND the information to the web server }
```

Notify user of new replies (Phase 2)

A) SERVER SIDE –

The server would update the database which the notification will be shown to everyone who have participated in the discussion.

B) CLIENT SIDE –

The users who have participated in the discussion should all be notified when a new reply responded to the discussion is posted. This could further promote the exchange of ideas. This requires collaboration with the development of research log.

- Pseudo code -

```
IF (reply comment)
{  SEND the information to the web server }
   NOTIFY all the user who have participated in the discussion  }
```

6.3. Module 3 : Research Log

6.3.1. User accounts

Login to system (Phase 1)

A) SERVER SIDE –

When the user logins to system, the client sends the user ID and password to the server. The server will check if the login succeeds, the user name will be returned to the client.

B) CLIENT SIDE –

User is required to login to use this application (Figure 6.26), which is required to input the login ID and password, the client devices will then send these information to the web server to check if the login validates. If it succeeds, the user is able to choose a paper to read and use the application normally, otherwise, an error message about login failure will be shown (Figure 6.27), and the client device asks the user to input the information again.

- Pseudo code -

```
IF (open the application)
{  Ask the user to input login ID and password
   Send the login ID and password to web server
   GET the result from the web server
   IF (validate)
   {      ALLOW the user to user the application  }
   Else
   {      SHOW error message
          ASK the user to login again        }      }
```

Figure 6.26 Screen capture of login

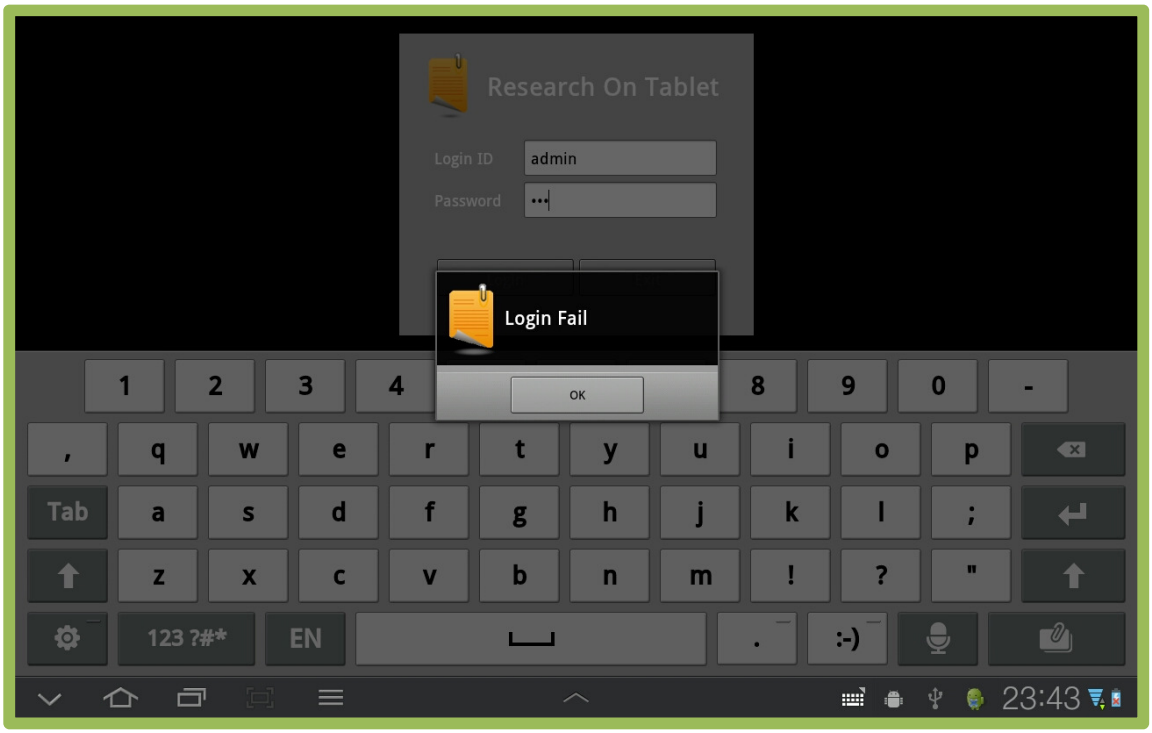


Figure 6.27 Screen capture of login fail

If the user want to exit the application, the client device will ask the user to confirm his exiting (Figure 6.28). If the user confirms to exit, he will logout from the application and the application will close. Otherwise, the client device will return to the previous view and let the user continue to use.

- Pseudo code -

```
IF (choose exit)
{  ASK the user to confirm
   IF (confirm)
   {       EXIT the application}
   ELSE
   {       STAY on that page  }       }
```

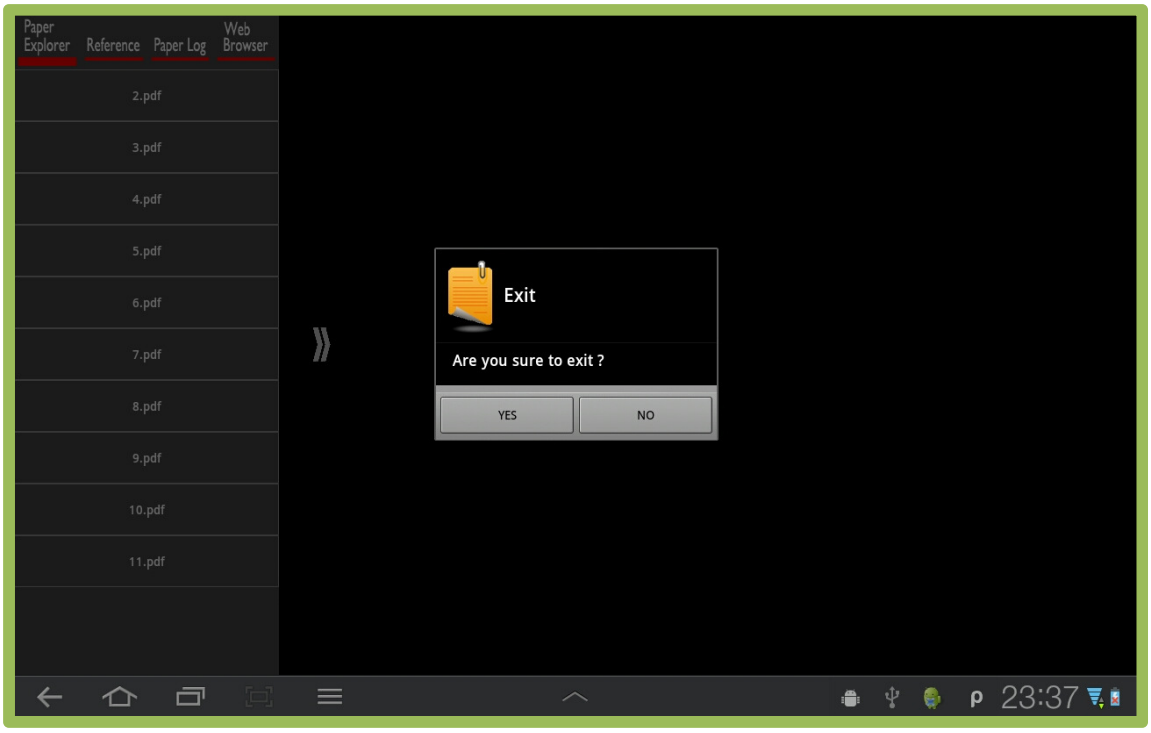


Figure 6.28 Screen capture of exiting

6.3.2. Logs

Showing history of logs (Phase 2)

A) SERVER SIDE –

When server receives request, it will update the database and send back the result to client.

B) CLIENT SIDE –

A list of research logs of the user will be shown inside the panel (Figure 6.29). The user can choose to refresh or clear its research logs (Figure 6.30).

- Pseudo code -

```
SHOW list of research logs

IF (refresh)

{  SEND request to web server to update its logs

   GET the result from web server

   UPDATE the list of research logs }

IF(clear)

{  SEND request to web server to clear its logs

   GET the result from web server

   UPDATE the list of research logs }
```

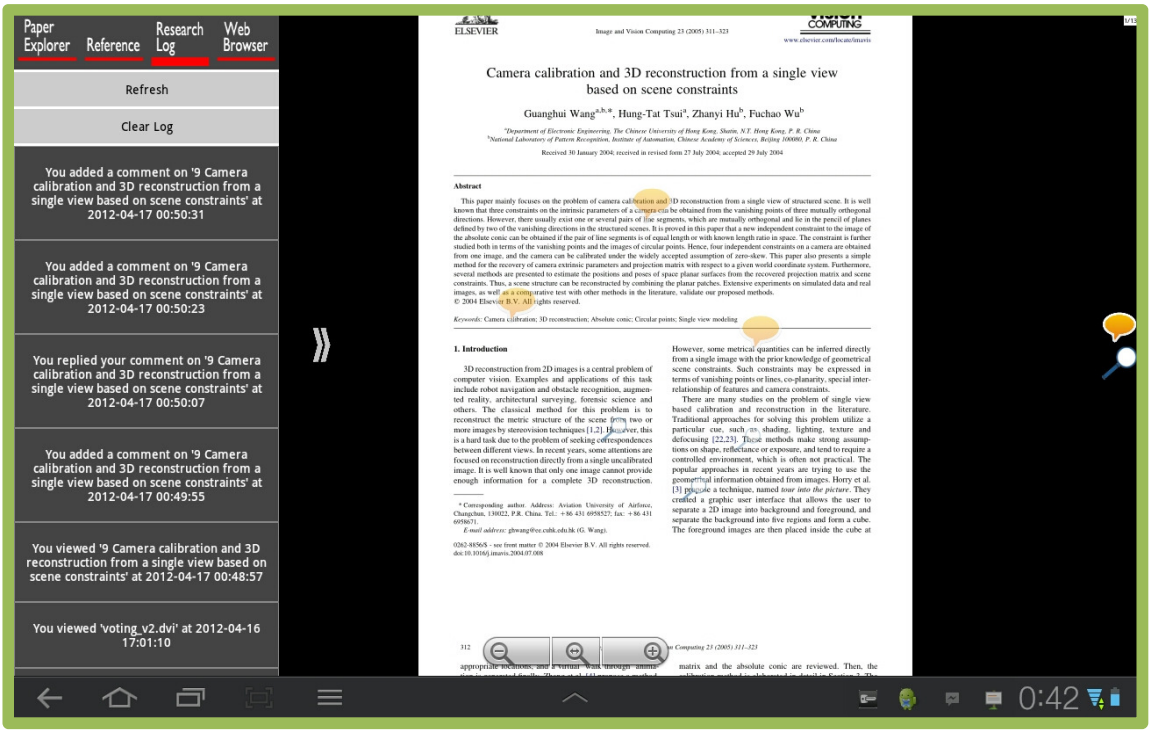


Figure 6.29 Screen capture of list of research logs

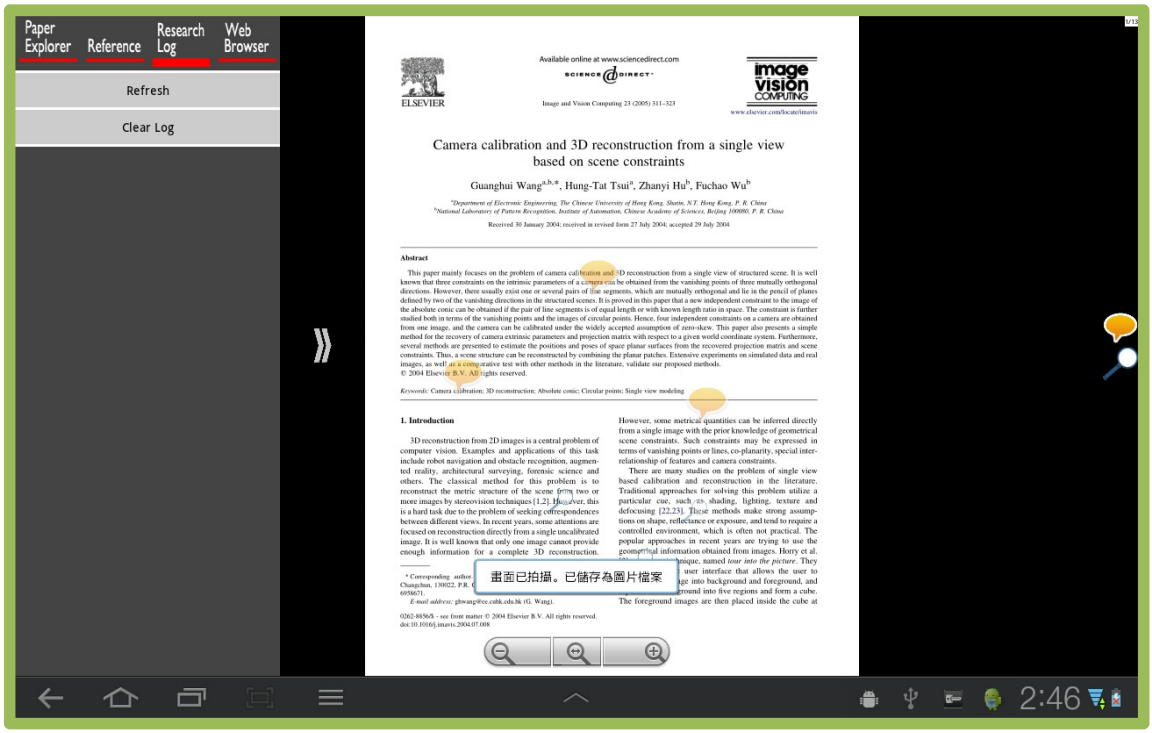


Figure 6.30 Screen capture of clearing logs

View history of reading (Phase 2)

A) SERVER SIDE –

Server will save the history of reading of a user to the database.

B) CLIENT SIDE –

The reading log will refresh every time the user open a paper to read. The entries of the log will be sorted by the time happened in descending order.

- Pseudo code -

```
IF (open a paper)
{ SEND request to web server to update its logs }
```

View commenting activities (Phase 2)

A) SERVER SIDE –

Server will update the history of commenting activities of a user to the database.

B) CLIENT SIDE –

The log will show the all the commenting activities that the user performed, including adding, replying and deleting comments. The commenting time, title of paper will be shown.

- Pseudo code -

```
IF (add OR reply OR delete a comment)
{ SEND request to web server to update its logs }
```

Show notification of new reply (Phase 2)

A) SERVER SIDE –

Server will update the history of replying activities of a user to the owner of the comment to the database.

B) CLIENT SIDE –

While the discussion the user participated has been renewed, the user should be notified by referring the log.

- Pseudo code -

```
IF (someone replies your comment)
{ SEND request to web server to update its logs }
IF (someone replies a comment you have replied)
{ SEND request to web server to update its logs }
IF (someone deletes a comment you have replied)
{ SEND request to web server to update its logs }
```

7. Testing and Evaluation

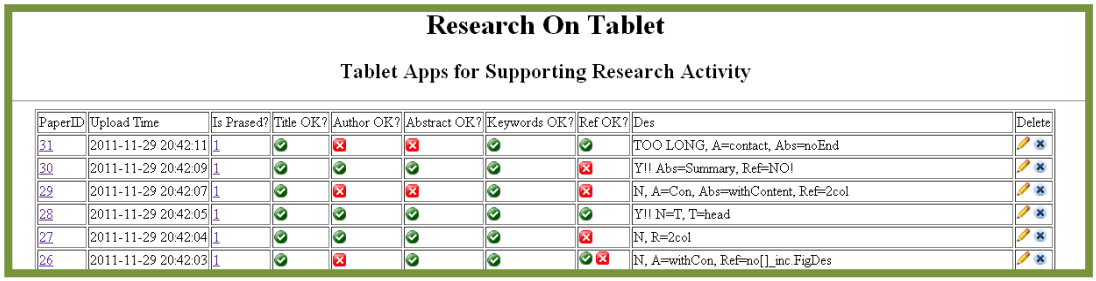
In this section, we are going to show the testing result and evaluation on them. We will test the server and client side separately. Both of them focus on different aspects. For the server, there are two testing on parsing and uploading performance. While for the client, the testing focuses on the functional performance of the application in the tablet devices.

7.1. Server Side

7.1.1. Parsing Performance Testing

A) Testing Methodology

To test the parsing performance, we modified the managing panel designed for the administrator to capture the data easily (Figure 7.1). With the panel, the data call be represented in the table form, so as to make everything clearly.



Research On Tablet

Tablet Apps for Supporting Research Activity

PaperID	Upload Time	Is Prased?	Title OK?	Author OK?	Abstract OK?	Keywords OK?	Ref OK?	Des	Delete
31	2011-11-29 20:42:11	1	✓	✗	✗	✓	✓	TOO LONG, A=contact, Abs=noEnd	
30	2011-11-29 20:42:09	1	✓	✓	✓	✓	✗	Y!! Abs=Summary, Ref=NO!	
29	2011-11-29 20:42:07	1	✓	✗	✗	✓	✗	N, A=Con, Abs=withContent, Ref=2col	
28	2011-11-29 20:42:05	1	✓	✓	✓	✓	✓	Y!! N=T, T=head	
27	2011-11-29 20:42:04	1	✓	✓	✓	✓	✗	N, R=2col	
26	2011-11-29 20:42:03	1	✓	✗	✓	✓	✓ ✗	N, A=withCon, Ref=no[]_inc.FigDes	

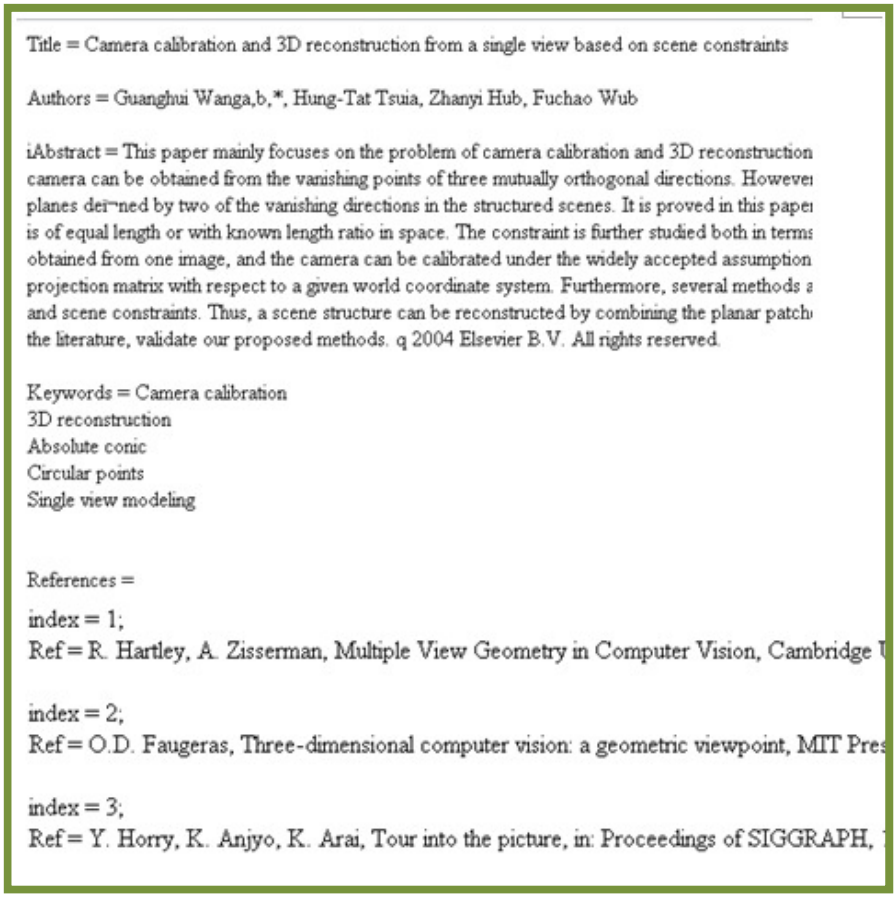
Figure 7.1 Admin Panel on web server

.

There are eight columns designed for the testing.

Data Viewing

The paper file can also be view by clicking the Paper ID, while the "IsPrased" column provides a link to view the parsing results. The page shows all the entries that parsed into the database (Figure 7.2).


Title = Camera calibration and 3D reconstruction from a single view based on scene constraints

Authors = Guanghui Wanga,b,*, Hung-Tat Tsuia, Zhanyi Hub, Fuchao Wub

iAbstract = This paper mainly focuses on the problem of camera calibration and 3D reconstruction
camera can be obtained from the vanishing points of three mutually orthogonal directions. Howeve
planes dei¬ned by two of the vanishing directions in the structured scenes. It is proved in this pape
is of equal length or with known length ratio in space. The constraint is further studied both in terms
obtained from one image, and the camera can be calibrated under the widely accepted assumption
projection matrix with respect to a given world coordinate system. Furthermore, several methods a
and scene constraints. Thus, a scene structure can be reconstructed by combining the planar patch
the literature, validate our proposed methods. q 2004 Elsevier B.V. All rights reserved.

Keywords = Camera calibration
3D reconstruction
Absolute conic
Circular points
Single view modeling

References =

index = 1;
Ref = R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge U

index = 2;
Ref = O.D. Faugeras, Three-dimensional computer vision: a geometric viewpoint, MIT Pres

index = 3;
Ref = Y. Horry, K. Anjyo, K. Arai, Tour into the picture, in: Proceedings of SIGGRAPH,

Figure 7.2 Paper parsing on web server

Result Recording

There are five columns available to mark the tick or cross for each entry. After checking the results, we mark down whether the results given are successful or not.

Remarks Marking

There is a "Des" to show the remarks. We check the results after parsing, then try to find out the reason of failure and mark it down. The remarks can be edited by clicking the pencil icon on the right.

B) Data Description

We have parsed twenty PDF files for testing the parsing performance. The following figure shows our results (Figure 7.3):

PaperID	Upload Time	Is Prased?	Title OK?	Author OK?	Abstract OK?	Keywords OK?	Ref OK?
31	2011-11-29 20:42:11	1	✓	✗	✗	✓	✓
30	2011-11-29 20:42:09	1	✓	✓	✓	✓	✗
29	2011-11-29 20:42:07	1	✓	✗	✗	✓	✗
28	2011-11-29 20:42:05	1	✓	✓	✓	✓	✓
27	2011-11-29 20:42:04	1	✓	✓	✓	✓	✗
26	2011-11-29 20:42:03	1	✓	✗	✓	✓	✗
25	2011-11-29 20:41:58	1	✗	✗	✗	✓	✗
24	2011-11-29 20:41:56	1	✓	✗	✗	✓	✗
23	2011-11-29 20:41:55	1	✗	✗	✗	✓	✗
22	2011-11-29 20:41:54	1	✓	✓	✓	✓	✗
21	2011-11-29 20:41:53	1	✗	✗	✗	✓	✓
20	2011-11-29 20:41:52	1	✗	✗	✓	✓	✓
19	2011-11-29 20:41:50	1	✓	✗	✓	✓	✗
18	2011-11-29 20:41:49	1	✓	✗	✗	✓	✗
17	2011-11-29 20:41:48	1	✓	✗	✓	✓	✓
16	2011-11-29 20:41:47	1	✓	✗	✓	✓	✗
15	2011-11-29 20:41:46	1	✓	✗	✓	✓	✓
14	2011-11-29 20:41:45	1	✓	✗	✓	✓	✗
13	2011-11-29 20:41:43	1	✗	✗	✗	✗	✗
12	2011-11-29 20:41:42	1	✓	✗	✗	✓	✓

Figure 7.3 Parsing result on web server

Percentage of successful TITLE Parsing = 15/20 = 75%
Percentage of successful AUTHOR Parsing = 5/20 = 25%
Percentage of successful ABSTRACT Parsing = 11/20 = 55%
Percentage of successful KEYWORDS Parsing = 3/20 = 85%
Percentage of successful REFENENCE Parsing = 7/20 = 35%

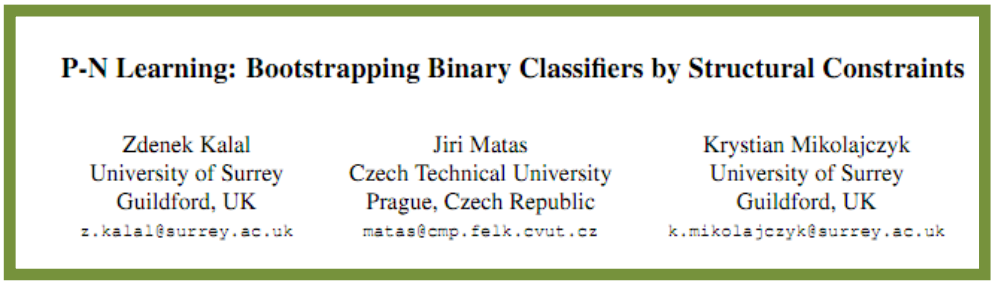
C) Evaluation

Title

The percentage of successful cases (75%) is satisfactory. Throughout the fail cases, most of them resulted in the malfunction of getting font size function in PHP. Within 20 papers, there are totally 4 papers with header goes before the title and only one of them success. The fail case returns the header as the title. Besides, one of failure (PID=25) resulted because of the wrong arrangement of the title and author. The author in this paper goes before the title. The paper with PID=13 fails because the "pdftotext" function does not support the UTF-8 encoding.

Author

The percentage of successful cases (25%) is disappointing. Part of the failure is caused because of the failure of parsing title (since our implementation method of the authors is parsing the "authors" right after the "title".). For the majority of the fail cases, the failures are resulted because of the format of PDF file. Take the following case as examples (Figure 7.4):


P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints

Zdenek Kalal
University of Surrey
Guildford, UK
z.kalal@surrey.ac.uk

Jiri Matas
Czech Technical University
Prague, Czech Republic
matas@cmp.felk.cvut.cz

Krystian Mikolajczyk
University of Surrey
Guildford, UK
k.mikolajczyk@surrey.ac.uk

Figure 7.4 Author format on paper

In this case, the authors are parsed to plain text as (Figure 7.5):

```
P-N Learning: Bootstrapping Binary Classii¬ers by Structural Constraints
Zdenek Kalal University of Surrey Guildford, UK
z.kalal@surrey.ac.uk

Jiri Matas Czech Technical University Prague, Czech Republic
matas@cmp.felk.cvut.cz

Krystian Mikolajczyk University of Surrey Guildford, UK
k.mikolajczyk@surrey.ac.uk
```

Figure 7.5 Parsing author on web server

And this comes with difficulties that have been mentioned in the previous chapters: there has no standard to capture the difference between the authors' name and the descriptions like the authors' titles and contact information.

There is another case with the authors' descriptions. Let's take a look at the following case (Figure 7.6):

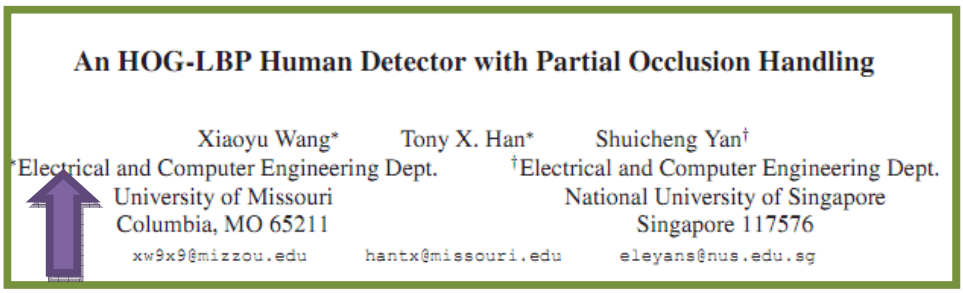


Figure 7.6 Author Descriptions on paper

And we have the following text parsed (Figure 7.7):

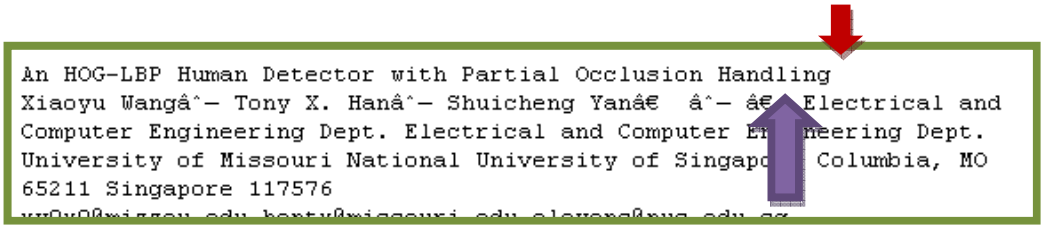


Figure 7.7 Parsing author descriptions on web server

It is clearly that the new line character is not parsed after the end of the last author's name. This results that the authors' names are recorded with their titles.

Abstract

The percentage of successful cases (55%) is just fair. The most significant failure the abstract includes some part of content or the figure caption. The reason that makes the parsing of abstract unsuccessful is the malfunction of the parsing program converting the PDF file to plain text (pdftotext). One of the case is shown as follow (Figure 7.8):

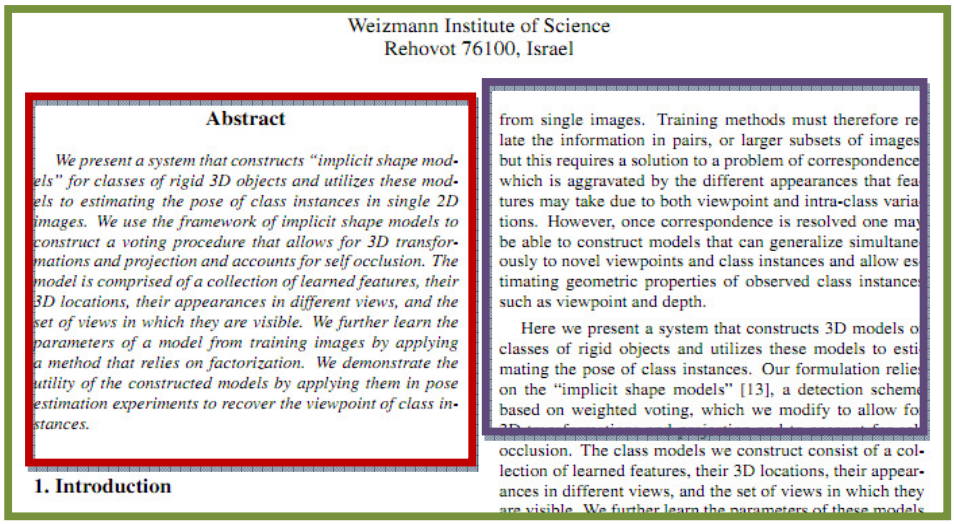


Figure 7.8 Abstract on paper

Let's see the text parsed (Figure 7.9):

Figure 7.9 Parsing abstract on paper

The above case shows that the content of the nest column goes before the "Introduction". This results in wrong parsing of the abstract.

There are still few problems of parsing. One is that some paper in report-like format that contains table of content would make the server cannot locate the end of the scope of abstract. Another problem is that some abstract are stated without the word "Abstract".

Keywords

The percentage of successful cases (95%) is almost satisfied. One reason is that not every paper has the keywords column. Even if the paper has keywords part, the keywords are few that avoid the case happened in the abstract – the probability of wrong parsing into fragments is relatively. The failure case is because of the wrong encoding of the PDF file to text file.

References

The percentage of successful cases (35%) is not satisfied. The reason of failure is quite similar to the cases as shown previously. Most of the papers are divided into two columns for better spacing. Yet, the parser sometimes mistakenly parses the right column first and make the order of the text got mess. Some of the reference items is parsed before the term “reference” and hence results in missing items.

There are also some problems for those reference items stated without the [ ] index. Since the “[ ]“ are missing, we must rely on the new line character to break the reference item into pieces. However the new line characters are not always successfully parsed. This makes parse the reference items fail.

7.1.2. Uploading Performance Testing

When a paper is uploaded, various parsing procedures will be performed. It always takes a long time after submitting the paper. Here an uploading performance test is performed to see if any constrains hinder the pace of uploading.

A) Testing methodology

We took 23 papers in total to record the time of parsing (uploading). The file size, total page and word count of the sample papers are recorded for comparison. A graph is plotted to see if there is any relationship between the factors and the parsing time.

The testing results are shown as below (Figure 7.10, Figure 7.11, Figure 7.12, Figure 7.13 and Figure 7.14):

File Size (KB)	Pages	Word Count	Ref No	Time (s)
151	7	2960	7	262
191	4	2917	9	331
194	5	2697	6	340
200	11	5802	8	380
208	14	5707	24	363
240	6	4590	13	363
246	10	4531	22	350
345	8	5286	26	397
411	12	6132	15	343
446	8	6299	22	446
467	11	7487	16	418
499	6	6358	12	424
635	5	2714	8	434
657	13	8171	22	455
982	4	7858	14	467
1195	8	8729	16	489
1223	10	8288	23	483
1228	8	8876	20	501
1262	6	7543	17	492
1498	11	7545	14	513
1634	8	8994	19	541
1753	26	9047	25	537
2008	4	8971	11	550

Figure 7.10 Table of information of papers we have tested

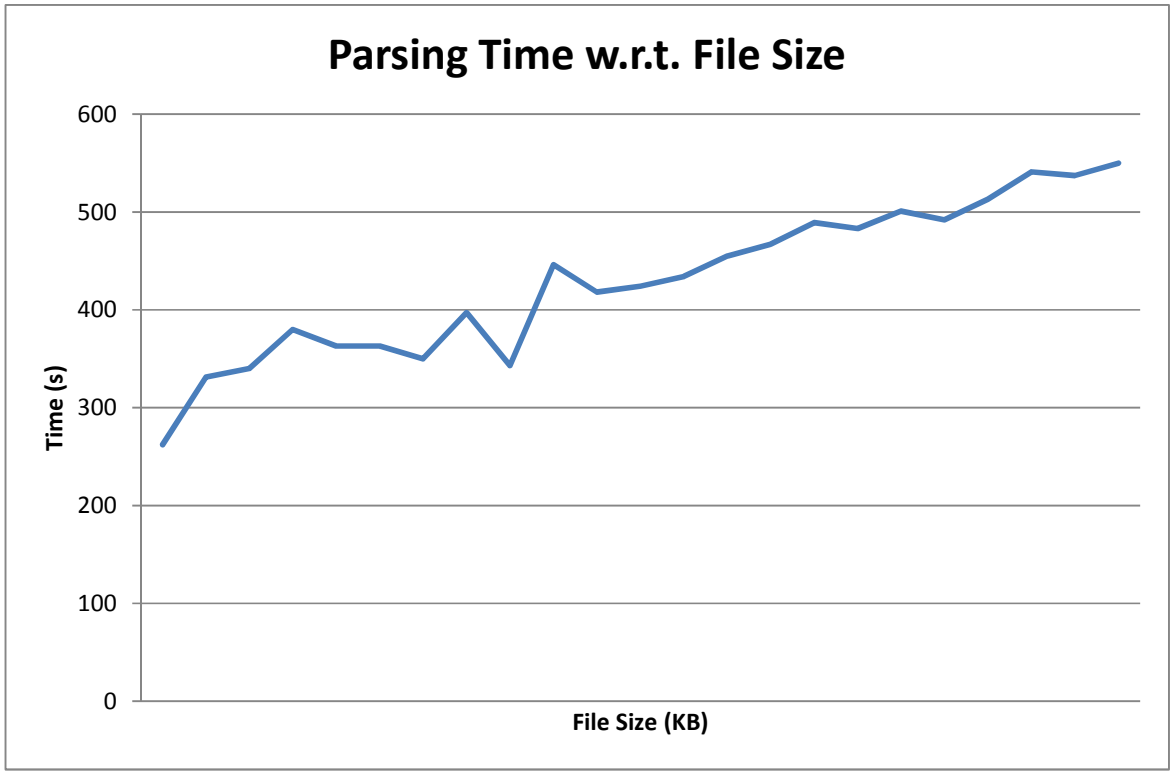


Figure 7.11 Graph of parsing time w.r.t. file size

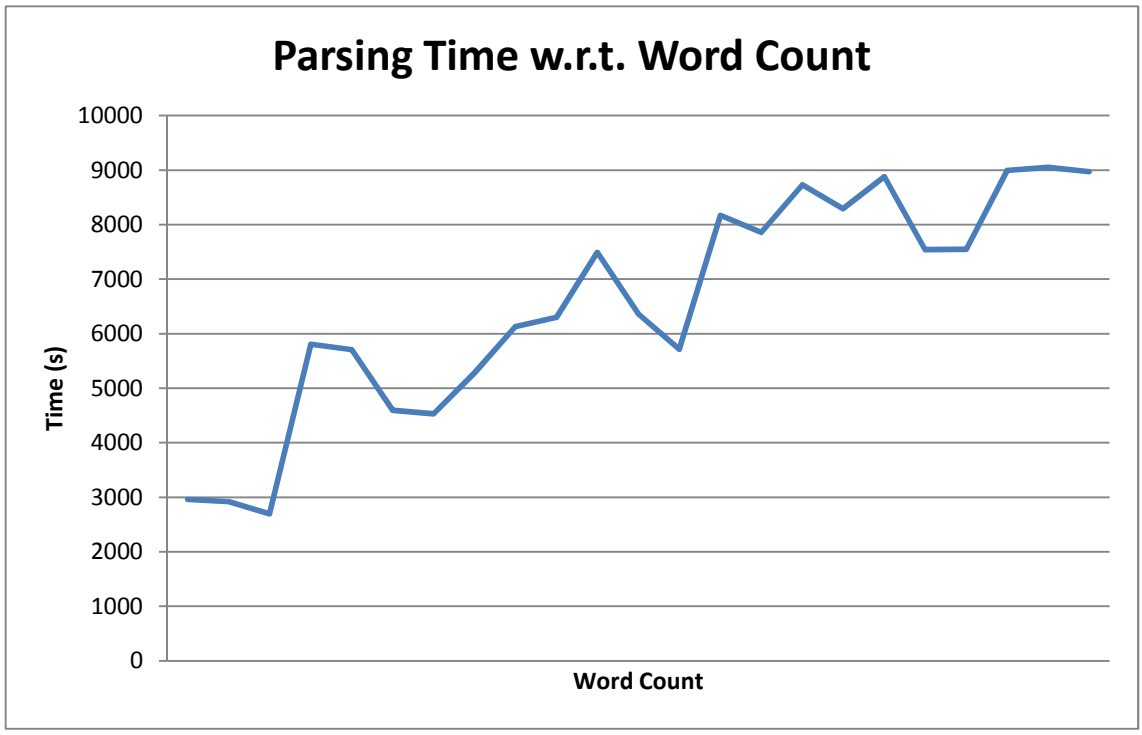


Figure 7.12 Graph of parsing time w.r.t. word count

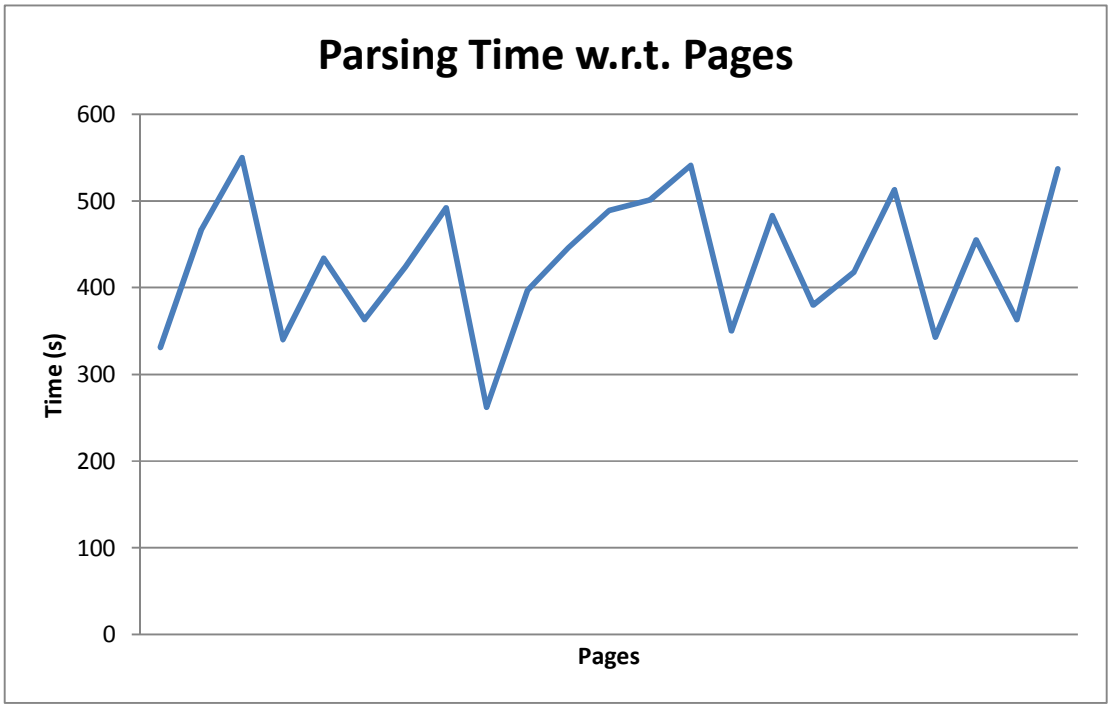


Figure 7.13 Graph of parsing time w.r.t pages

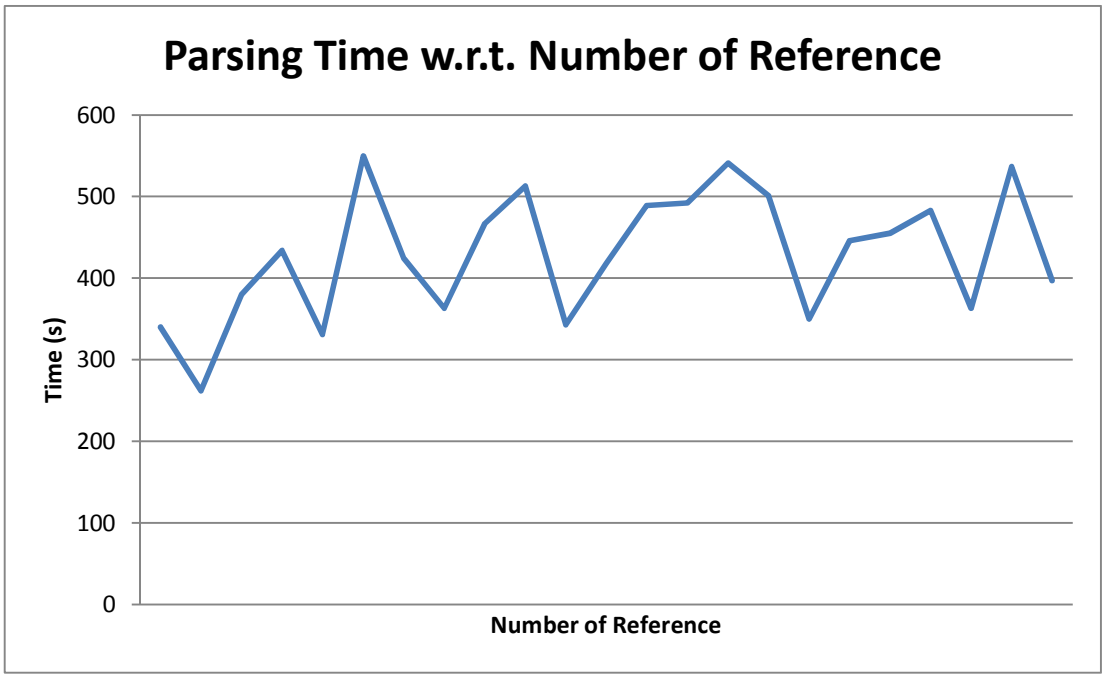


Figure 7.14 Graph of parsing time w.r.t number of reference

The above figures show that only the file size affects the uploading time proportionally. The tendency between word count and uploading time fluctuates quite unexpectedly.

B) Evaluation

During the uploading time, the server is undergoing server procedures. First is uploading file to the server. Then the PDF file would be extracted as plain text file for further parsing of required information.

The phenomenon of direct proportion resulted between file size and uploading time implies the time of file uploading may not a minor factor to the whole uploading time. This may imply harder improvement in uploading time. However, the relationship between word count and file size exists. After all, parsing is one of the main ideas of the system and plays an important role during the upload process. More words could result in longer parsing period, which hinder the whole uploading process,

7.2. Client Side

7.2.1. Evaluation

For application in the client devices, we do testing on every functions, including login, opening and showing a paper, functions about references, functions about comments, zooming, finding texts, going to certain page and exiting the application.

Login

	Case	Expectation	Result
1	Empty Login ID and Password	Error Message	Same
2	Wrong Login ID	Error Message	Same
3	Wrong Password	Error Message	Same
4	Correct Login ID and Password	Login Succeed	Same

Open and Show a Paper

	Case	Expectation	Result
1	Choose one paper	Show the information of the paper	Same
2	Choose to open	Open the paper	Same
3	Choose another paper while reading one	Show the information of the new paper	Same
4	Choose to open the new paper	Close the original one and open the new one	Same
5	Show every part of the paper when moving it	Show every part normally	The last page cannot be read normally, which suddenly flick to the previous page

Search Paper

	Case	Expectation	Result
1	Choose “Keyword” and search	Three tabs will be shown, including “Title”, “Keyword” and “Content”, and also the list of result	Same
2	Choose “Title” after searching	All the result that title contains the words would be shown	Same
3	Choose “Keyword” after searching	All the result that keyword contains the words would be shown	Same
4	Choose “Content” after searching	All the result that the highest TFIDF would be shown	Same
5	Choose “Author” and search	All the result that the paper is written by that author would be shown, including showing the author name	Same
6	Click “Show All” button	Show the list of all the paper in the server	Same

Reference

	Case	Expectation	Result
1	Show the reference list after opening a paper	Show the reference list in the transparent panel	Same
2	Show the reference button after opening a paper	Show the reference button beside every references in the paper	Same
3	Hide all the reference buttons inside the paper	All the reference buttons are hidden	Same
4	Show all the reference buttons inside the paper after hidden	All the reference buttons are shown	Same
5	Show the information of the reference by clicking reference button	Show the information of that reference in an alert box	Same
6	Link to another paper if linkage exists	Close the original one and open the reference paper	Same
7	Search on the Internet in the web browser	Show the search result automatically	The search result exists, but cannot be shown automatically

Research Log

	Case	Expectation	Result
1	Show the log after viewing a paper	A new log about viewing paper will be added	Same
2	Show the log after adding a comment	A new log about adding comment will be added	Same
3	Show the log after replying a comment	A new log about replying comment will be added	Same
4	Show the log after someone else reply your comment	A new log about replying comment will be added	Same
5	Show the log after someone uploading a paper	A new log about uploading paper will be added	Same
6	Click the refresh button	The newest list of research log will be shown	Same
7	Click the clear button	All the previous logs will be cleared	The logs about uploading paper cannot be cleared since it is shown to everyone

Web Browser

	Case	Expectation	Result
1	Search something on google inside the web browser	Show the search result inside the web browser	Same
2	Enter a URL	Show the certain web page	Same
3	Click “Home” button	Show the home page, which is google	Same
4	Click “Upload” button	Show the list of file of the local device	Same
5	Choose a paper to upload	Close the old paper and open the new one, and information of the paper will be shown in the panel automatically	New paper can be opened, but the information cannot be shown automatically, it needs to be opened manually
6	Update the information of the paper	Newly input information will be updated to the server	Same

Comment

	Case	Expectation	Result
1	Show the previous comments after opening a paper	Show the buttons of comments at the correct positions	Same
2	Add new comment within the paper	Ask the user to input the comment, and add a new button	Same
3	Show the comment content by clicking the comment button	Show the comment content besides the button without being out of screen	Same
6	Hide the comment content by clicking the comment button	Hide the comment content, and make the button translucent	Same
7	Reply comment	Show the reply at the bottom of the comment	Same
8	Cancel replying a comment	No new reply will be added	Same
9	Delete a comment	Delete that comment and all its replies, and also the button	Same
10	Delete a reply	Delete that reply	Same
11	Hide all the comment buttons	All the comments buttons are hidden	Same
11	Show all the comment buttons after hidden	All the comments buttons are shown	Same

Zoom

	Case	Expectation	Result
1	Click the zoom in button	The paper is zoomed in, show all the view with the new scale	Same
2	Click the zoom out button	The paper is zoomed out, show all the view with the new scale	Same
3	Click the zoom fit-in-width button	The paper is zoomed to fit the width of the paper, show all the view with the new scale	Same
4	Double click the screen	The paper is zoomed in, show all the view with the new scale	Same

Go to Certain Page

	Case	Expectation	Result
1	Empty String	Error Message	Same
2	A number which is out of bound	Error Message	Same
3	A number which is within the bounding	Move the view to that certain page	Same
4	Choose going to the first page	Move the view to the first page	Same
5	Choose going to the last page	Move the view to the last page	Same

Find Text

	Case	Expectation	Result
1	Empty string	Nothing will be found	The application has an error
2	A text does not exist in the paper	Nothing will be found	Same
3	A text existing in the paper	The texts in the paper will be bounded by three red lines	Same
4	Choose previous result	Move the view to the location of the previous result	Same
5	Choose next result	Move the view to the location of the next result	Same
6	Choose hiding all the result	Return to the normal view without showing the result of finding	Same

Exit the Application

	Case	Expectation	Result
1	Click exit button	Ask the user to confirm	Same
2	Confirm exiting	Exiting the application	Same
3	Cancel exiting	Return to the view before	Same

8. Conclusion and Future Work

8.1. Difficulties and Suggested Solutions

We are going to summarize all the difficulties we have encountered during the whole working period in phase 1.
This part will be divided into the server and client side.

8.1.1. Server Side

Difficulty

During the implementation in phase 2, there are two main difficulties to highlight.

Firstly, it is proved from the result of testing that the larger file size results in longer uploading period. As mentioned before, during the upload period, not just file upload process is performed, but also the procedure of parsing information required. Besides getting the metadata of PDF files, the abstract is also got by the parse. The reference list is also concerned.

The reason makes the problem inevitable is that term frequency of each exact wording is counted. Larger file size, in some sense means more words written, shall take longer time for word counting. This is an unavoidable process if the searching relies on the principle of TF-IDF.

Secondly, the checking of duplication seems not easy. There are two basic algorithm can be implemented. One is doing comparison among all papers when a file is newly added. Doing content comparison ensure the uniqueness of the specific paper. However, this exert great burden on the parsing procedure and deteriorate the problem of long uploading period.

The other approach, which is now adopted in or system, is to compare the title and the author first in order to filter the same files out. This indeed helps reduce much more time for comparison. However, since the parsing metadata process is not credible enough, minor typing errors may cause inaccuracy of comparison. This reduce the credibility of checking duplication

Suggested Solution

We thought several suggestions to the abovementioned problems. Concerning the long uploading period, although it relates to uploading file times, the parsing procedure can be improved. One key factor affecting the parsing time is term frequency parsing. The currently adopted approach is to build up a dictionary to record every exact term count, which force the server to “Read” every word. We believe there shall be alternatives other than reading word by word. Improvement to TF-IDF algorithm may be discussed, or there may be open-source programs available online to do the TF-IDF.

Moreover, since manually parsing is now adopted to ensure the accuracy of parsing, the value of parsing few elements by server is doubtable. As a user to upload the papers, with embedded frame nearby to copy details directly for submission, we thought the importance of parsing abstract and keyword list is greatly reduced. This may be adjusted to see if it can help minimize the parsing time of the paper.

As for the problem of duplication checking, we believe the approach of checking title and author can be sharpened. The server should be taught to be more sensitive to similar wordings in both titles and author names. Further algorithm can be discussed to check the percentage of similarity instead of return if they are exactly the same or not. Such kind of algorithm aims to reduce the error of parsing and hence improve the efficiency and effectiveness of file duplication checking.

8.1.2. Client Side

Difficulty

Because the main purpose of this project is to assist the research activities, helping people to read research papers more convenient, we do not put much effort into how the API designs, but implement more and more ideas and methods to help people reading research papers. However, the API consists of some bugs that affect the reading activities, for example, reading the last pages will cause some trouble of reading, as well as the function of finding texts.

We do not have enough time to achieve both areas, understanding the API and solving those problems, or implementing more and more useful ideas and methods. We choose to develop more functions for the user to do the research works easier, so we have not finished the part of understanding the API and debugging. Although that would cause some troubles to the users, and we think that the new functions would have remedied the problems, and could also bring much more benefits to users.

On the other hand, developing application in Android platform is also a challenge. There are many limits, including controlling the layout, and some important functions like web view. Besides, this project is the first opportunity for us to develop application in Android platform, therefore, our skills in developing Android apps are not mature enough. Lots of useless codes and functions exists in the project, as well as some inefficient functions are used. With the experiences in doing this project, we have great improvement in phase 2.

Suggested Solution

One of our problems is the bugs of the API we have chosen. As we decided to implement more useful functions, we choose not to solve those problems immediately. As most of the functions have been finished, we would suggest solving those problems. Since it needs time to understand the design of the API, if there is enough time, those problems could be solved.

Another suggestion is to develop a new application to view PDF file. Since the API we have chosen does not focus in reading research papers, but just reading PDF file, a newly design application which aims to assist reading research papers would be much more useful for users. With the new reading PDF application, as well as the functions we have developed, the research activities would be much easier and more convenient.

On the other hand, with more experience in developing Android, the more technology we can learn. Therefore, the methods and functions can be improved with higher efficiency and speed, and it can also contain less useless codes. If we can continue to develop and learn Android platform, the application can be much improved.

8.2. Contribution of Work

Item	Steven	Elva
Background Research		
Existing Problems		O
Objectives Indication		O
Research on Relevant Topics		
Studies on research paper	O	O
Discussion on parsing phrases		O
Further discussion for functions	O	O
Investigation of Existing Tools	O	
Studies on searching approach		O
Product Design		
System Architecture Design	O	
Server Design		O
Client Design	O	
Implementation		
Module 1: paper management		O
Module 2: communication	O	
Module 3: Research Log	O	
Report Editing		
Introduction		O
Background		O
Research on relevant topics	O	O
Product Design	O	O
Implementation	O	O
Testing and evaluation	O	O
Difficulties and suggestion		O
Future work		O

8.3. Conclusion

It is nice to cooperate with Elva. Besides, I have gained invaluable guidance suggestions from our supervisor, Prof. Michael Lyu, and great support from Mr. Edward Yau too.

In this year, my work focus on client side, which develops an Android application for reading research papers. In phase 1, I mainly studied how Android application could be developed, how the API for reading PDF files worked, as well as implemented some basic functions. In phase 2, my main duty was implementing the advanced features to the application, and improving the user interface of the application.

Throughout the whole year, I have an in-depth study on how Android application is developed. During the study in the API we had used, I have learnt lots about the layout, design and operation of an Android application, which is very useful for me to develop Android applications in the future.

Besides, since lots of data was needed to be sent through the Android tablet and the web server, I have learnt that how the data can be transmitted by JSON.

This final year project pushes me to be conscious, and it improves my communication skills with my teammate Elva. In addition, it makes me learn much about time management, as well as skills of cooperation with others.

8.4. Future Work

8.4.1. Improve efficiency of uploading paper

To focus back on our raw objectives, the whole Research on Tablet system aims to reduce workload of research resources management. Long waiting is obviously not an efficient way to manage the paper repository, especially when numerous files are waiting to be uploaded. If further improvement is expected, much more effort should be put on enhancing the speed of library renewal.

8.4.2. Sharpen the searching performance

Searching process plays a paramount role especially when the paper repository is huge. Currently, the searching algorithm involves the TF-IDF, stemming and stop word checking to improve the efficiency of searching. Each of approaches can be fine-tuned. For example, the process of stemming can be adjusted. When it comes to searching among scholars, in various occasions we would search for noun phrases, instead of verbs and adjectives. Therefore, ordinary stemming rules seem to be abundant to our searching engine because some of rules may seldom or never be touched.

8.4.3. Improve the performance of the client application

On continuing developing the client application, new methods can be tried to improve the performance and efficiency. For example, using better structure of database or faster methods can speed up the application, which is more convenient to users. Besides, some functions like zooming in and out by pinching the screen could also be implemented to help users to do research works easier.

9. Acknowledgement

We would like to thank the supervisor of our project, Professor Michael Lyu. who gives us valuable advices for our project and remainders to present well. Professor Lyu arranges a meeting with us every week for 1 hour so as to keep our progress up. We are not sure whether the final product can be achieved perfectly, but we feel confident to overcome the obstacles with the help of Professor Lyu.

Besides, we would like to thank Mr. Edward Yan in VIEW Lab. He provides us with facilities and technical support when we are setting up the server and gives us many suggestions on how to make the application more feasible.

10. Reference

[1] KI MAE HEUSSNER and BECKY WORLEY : "Apple iPad: Steve Jobs Unveils the New Apple Tablet ", abcnews http://abcnews.go.com/Technology/apple-ipad-steve-jobs-unveils-apple-tablet/story?id=9667578#.TtDPW7LTpQA

[2] Android application: “gScholarReader” : http://www.androidzoom.com/android_applications/lifestyle/gscholarreader_mdcf.html

[3] Android and iPhone Apps for Selected Research Resources: http://www.lincolnlibraries.org/databases/Android_and_iPhone_Apps_for_Selected_Research_Resources.htm

[4] “Research paper” – WIKIPEDIA : http://en.wikipedia.org/wiki/Research_paper

[5] Behrooz Parhami: "Research Paper Guidelines", University of California, Santa Barbara :http://ece.ucsb.edu/~parhami/rsrch_paper_gdlns.htm

[6] "Writing Research Papers", Rice University: http://www.ruf.rice.edu/~bioslabs/tools/report/reportform.html

[7] "Portable Document Format", WIKIPEDIA : http://en.wikipedia.org/wiki/Portable_Document_Format

[8] Adobe Systems Incorporated: "PDF Referebce" (sixth edition)

[9] "pdfmeat": http://code.google.com/p/pdfmeat/

[10] "PDFlib": http://www.pdflib.com/products/pdflib-family/

[11] "PHP Manual": http://www.php.net/manual/en/function.pdf-set-parameter.php

[12] “pdfbox”: http://pdfbox.apache.org/download.html

[13] “Xpdf”: http://foolabs.com/xpdf/about.html

[14] “SAMSUNG GALAXY Tab 10.1”: http://www.samsung.com/hk/consumer/mobile/mobile-phones/mobile-tablet/GT-P7500FKDTGY/index.idx?pagetype=prd_detail