

ABSTRACT

Title of dissertation: Computer Vision and Image Processing
Techniques for Mobile Applications

Xu Liu

Doctor of Philosophy, 2008

Dissertation directed by: Professor Larry Davis
Department of Computer Science

Camera phones have penetrated every corner of society and have become a focal point for communications. In our research we extend the traditional use of such devices to help bridge the gap between physical and digital worlds. Their combined image acquisition, processing, storage, and communication capabilities in a compact, portable device make them an ideal platform for embedding computer vision and image processing capabilities in the pursuit of new mobile applications. This dissertation is presented as a series of computer vision and image processing techniques together with their applications on the mobile device. We have developed a set of techniques for ego-motion estimation, enhancement, feature extraction, perspective correction, object detection, and document retrieval that serve as a basis for such applications. Our applications include a dynamic video barcode that can transfer significant amounts of information visually, a document retrieval system that can retrieve documents from low resolution snapshots, and a series of applications for

the users with visual disabilities such as a currency reader. Solutions for mobile devices require a fundamentally different approach than traditional vision techniques that run on traditional computers, so we consider user-device interaction and the fact that these algorithms must execute in a resource constrained environment. For each problem we perform both theoretical and empirical analysis in an attempt to optimize performance and usability. The thesis makes contributions related to efficient implementation of image processing and computer vision techniques, analysis of information theory, feature extraction and analysis of low quality images, and device usability.

Computer Vision and Image Processing
Techniques for Mobile Applications

by

Xu Liu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:
Professor Larry Davis (chair)
Doctor David Doermann
Professor Ben Bederson
Professor Ashok Agrawala
Professor Min Wu

© Copyright by
Xu Liu
2008

Dedication

This dissertation is dedicated to my parents and my fiancée.

Acknowledgments

I would like to thank my advisor Dr. David Doeramnn who supported me throughout my Ph.D. study. This dissertation could not be made possible without his help. I am grateful as well to Professor Larry Davis for guiding me through my Ph.D. study, which was one of the most important and inspiring experiences in my life.

The members of my dissertation committee, Ben Bederson, Min Wu, and Ashok Agrawala have generously given their time and expertise to better my work. I thank them for their contribution and comments to my dissertation.

I also thank Dr. Huiping Li who co-authored six papers with me during my Ph.D. study. More importantly he provides the opportunity for me to apply my knowledge and techniques to solve real-world problems under Maryland Technology Enterprize Institute.

Professor David Jacobs has led me into the door of modern computer vision research and provided very important insights of my preliminary exams. Although he was on sabbatical and could not be on my dissertation committee, his advice and patience are highly appreciated.

After all I need to thank my supportive fiancée Qian Guo, who is always there with me, for her love and understanding of me.

Table of Contents

List of Figures	vii
List of Abbreviations	x
1 Introduction	1
1.1 The Evolution of Camera Enabled Mobile Devices	1
1.2 Novel Usage of the Camera	3
1.2.1 Input Extension	3
1.2.2 Cursor and Browsing Control	4
1.2.3 Capture of Visual Information	5
1.2.4 Data Transfer	5
1.2.5 Assistance to People with Visual Impairments	6
1.3 Challenges	7
1.3.1 The Limitations of Camera	7
1.3.1.1 Fixed Focal Length	7
1.3.1.2 Low Resolution and Exposure	9
1.3.1.3 Distortion	10
1.3.1.4 Uneven Lighting and Shadows	12
1.3.2 Computational Power and Battery Life	12
1.3.3 Human Factor Challenges	13
1.3.3.1 The Moving Camera	13
1.3.3.2 User Tolerance	14
1.4 Organization of Dissertation	14
2 Technical Contributions and Related Research	16
2.1 Perspective Correction	16
2.2 Pattern Detection and Recognition	22
2.2.1 Pre-classification and Filtering	23
2.2.2 Fast classification with Random Pixel Pairs	25
2.3 Pixel-wise Manipulation and Enhancement	30
2.3.1 Contrast Enhancement and Adaptive Binarization	30
2.3.2 Rapid Bilinear Interpolation for Binary Images	33
2.4 Ego-motion Estimation	36
2.4.1 Hierarchical Matching	36
2.4.2 Optimizing the Search Space	38
2.4.3 Evaluation	39
2.5 Feature Matching for Content based Image Retrieval	40
2.6 Summary	45

3	VCode - Imaging as an Alternative Mobile Data Channel	46
3.1	Overview	46
3.1.1	Motivation and Related Work	46
3.2	Binary V-Code	50
3.2.1	Binary VCode Encoding	51
3.2.1.1	Data Partitioning and Error Correction	52
3.2.2	Binary VCode Rendering	54
3.2.3	Binary VCode Acquisition	55
3.2.4	Experiments of Binary VCode	56
3.2.4.1	Data Transmission Speed	56
3.2.4.2	Data Capacity in a Single Frame	57
3.2.4.3	Display Frame Rate	60
3.2.4.4	Overall Downloading Bit Rate	63
3.3	Color VCode	63
3.3.1	Analysis	64
3.4	Channel Capacity and Color Selection	65
3.4.1	Two-color Channels	67
3.4.2	Multi-color Channels	68
3.4.3	Color Calibration	74
3.5	Encoding and Rendering	76
3.6	Decoding	77
3.7	Experiments	78
3.7.1	Cell Location Accuracy	79
3.7.2	Color vs. Density	80
3.7.3	Perspective Distortion	82
3.7.4	Throughput	84
3.8	Summary and Discussion	87
3.8.1	Why not use the Design of Existing 2D Barcodes?	88
4	MobileRetriever - Finding a Document with a Snapshot	89
4.1	Introduction	89
4.1.1	Motivation and Related Work	89
4.1.2	Application Scenarios	93
4.1.3	Challenges	95
4.1.4	Related Work	96
4.2	Approach	98
4.3	Indexing	98
4.3.1	Image Captured without Close-up Lens	99
4.3.2	Image Captured with Close-up Lens	101
4.4	Retrieval	106
4.5	Implementation	108
4.5.1	Client	108
4.5.2	Server	108
4.6	Evaluation	109
4.6.1	Retrieval using Token Pairs	111

4.6.1.1	Pages in the Repository	111
4.6.1.2	Pages not in the Repository	113
4.6.2	Token Triplet Verification	114
4.6.2.1	Pages in the Repository	114
4.6.2.2	Pages not in the Repository	116
4.6.3	Usability	116
4.6.3.1	Document Constraint	116
4.6.3.2	Overall Success Rate	118
4.6.3.3	Speed	119
4.7	Summary	120
5	MobileEye Tools - for Persons with Visual Disabilities	122
5.1	Motivation	122
5.2	Existing System and Design Principles	123
5.3	Mobile Pattern Recognition and Currency Reader	124
5.3.1	Initial Design	125
5.3.2	Revised Design	127
5.3.3	Evaluation	129
5.4	Other MobileEye Tools	130
5.4.1	Color Mapper	130
5.4.2	Software Magnifier	131
5.5	Summary	132
6	Conclusion	134
6.1	Summary of Contributions	134
6.2	Future Work	135
	Bibliography	136

List of Figures

1.1	Mobile interaction and desktop interaction	2
1.2	Thin lens equation	8
1.3	Close-up lenses: (a) existing lenses (b) our design	9
1.4	Fisheye distortion	11
2.1	Perspective Correction	17
2.2	Geometrical transformation between the matrix and perspective images	18
2.3	Standard deviation of 6 sub-images at 3 corners of a 20 dollar bill . .	24
2.4	Positive samples: McDonald’s logo	26
2.5	Negative samples: arbitrary scene	27
2.6	Maximize margin between positive and negative samples	29
2.7	Adaptive contrast enhancement method compared with histogram stretching based method	30
2.8	Bilinear Interpolation	33
2.9	Motion predication	39
2.10	Possible triplet matches	43
2.11	Example matches with point correspondences	44
3.1	Various 2D barcodes (a) Interaction oriented 2D barcodes (b) Content oriented 2D barcodes: QRCode, DataMatrix and PDF-417 (c) ColorZip’s ColorCode	47
3.2	Overview of the camera channel	48
3.3	(a) A VCode Frame. (b) An example of the mask used to “encrypt” VCode.	51
3.4	Data Partition. (a) Segment a data file into chunks and frames, and (b) The error correction scheme.	53

3.5	The number of erroneous bits over 100 frames for four settings. (a) 28×35 , (b) 32×40 , (c) 40×50 and (d) 48×60	58
3.6	The relationship between E and EBR	60
3.7	Simulation of binary color channel capacity	68
3.8	Small barcode we used for simulation	69
3.9	Distribution of captured colors, six samples	70
3.10	Eight optimized colors chosen from a graph of 40 vertices	72
3.11	Simulation of channel capacity with k colors for Jamin camera phone	74
3.12	One encoded frame	75
3.13	Data partition and temporal error corrections	76
3.14	Location and registration	77
3.15	Location accuracy test	80
3.16	Perspective distortion	83
3.17	Accuracy v.s. Perspective(K)	85
3.18	Throughput test	86
3.19	Progress of download	87
4.1	Digital Desk(a), VideoPen(b) and MobileRetriever (c)	91
4.2	Overview of MobileRetriever	94
4.3	A typical query image of MobileRetriever	95
4.4	A typical document image captured by a camera phone	99
4.5	An example of Layout Context	100
4.6	Token pairs	103
4.7	Token triplets	104
4.8	Image enhancement	106
4.9	MobileRetriever on Windows Mobile 6	109

4.10	Retrieving using token pairs	111
4.11	The page that matches query may not be ranked first	112
4.12	Queries have no match in repository still have hits of triplet pairs . . .	113
4.13	Correct retrievals are distinguished after token triplet verification. . .	115
4.14	Pages not in repository have low scores after triplets verification. . . .	115
4.15	False positive and false negative rate at different thresholds	117
4.16	Number of words in query vs. success rate	118
4.17	Images failed to retrieve	119
4.18	Time spent on each step of retrieving	120
5.1	Background subtraction and feature area extraction	125
5.2	Normalized feature area and random pixel pair	127
5.3	Standard deviation of 6 sub-images at 3 corners of a 20 dollar bill . .	128
5.4	User evaluation of mobile currency reader	129
5.5	A color design without considering the color deficient user	130
5.6	Color mapper	131
5.7	Image zoomed by software lens	132
5.8	Currency reader for the visually impaired	133

List of Abbreviations

α	alpha
β	beta
VCode	Video bar Code
EBR	Equivalent Bit Rate
BPS	Bits Per Second
FPS	Frames Per Second
OCR	Optical Character Recognition
VGA	Video Graphics Array (640 × 480)
SVGA	Super Video Graphics Array (1024 × 480)
QVGA	Quarter Video Graphics Array (320 × 240)
CDMA	Code Division Multiple Access
GSM	Global System For Mobile Communications
GPRS	General Packet Radio Service
EDGE	Enhanced Data Rates For Gsm Evolution
CMOS	Complementary MetalCoxideCsemiconductor
RISC	Reduced Instruction Set Computer
ARM	Advanced Risc Machine
SDK	Software Development Kit
SIFT	Scale Invariant Feature Transform
SVM	Support Vector Machine
RANSAC	RANdom SAmples Consensus
MAC	Media Access Control
LCD	Liquid crystal display

Chapter 1

Introduction

Integrated image capture, processing, and communication power on a compact, portable, hand-held device is attracting increased interest from computer vision researchers with a goal of applying a diverse collection of vision tasks on the small hand-held device. In this dissertation, we survey the state-of-the-art computer vision/pattern recognition technologies that have been developed on camera phones. We identify potential problems and difficulties with the camera phone and provide solutions to these problems. Furthermore we foresee the trend of vision applications on camera phones and propose solutions to the major technical challenges involving mobile vision and pattern recognition.

1.1 The Evolution of Camera Enabled Mobile Devices

On March 10, 1876, Alexander Graham Bell first transmitted speech electronically, setting in motion the development of telephone systems which have evolved into the devices we enjoy today. Throughout this revolution, voice communication has remained as the primary function of phones despite the introduction of wireless capabilities and, more recently, the view of mobile devices as extending the desktop computer to be used anytime and anywhere. One novel exception is the addition of the camera. Sharp launched its first camera phone product, the J-SH04, in Novem-

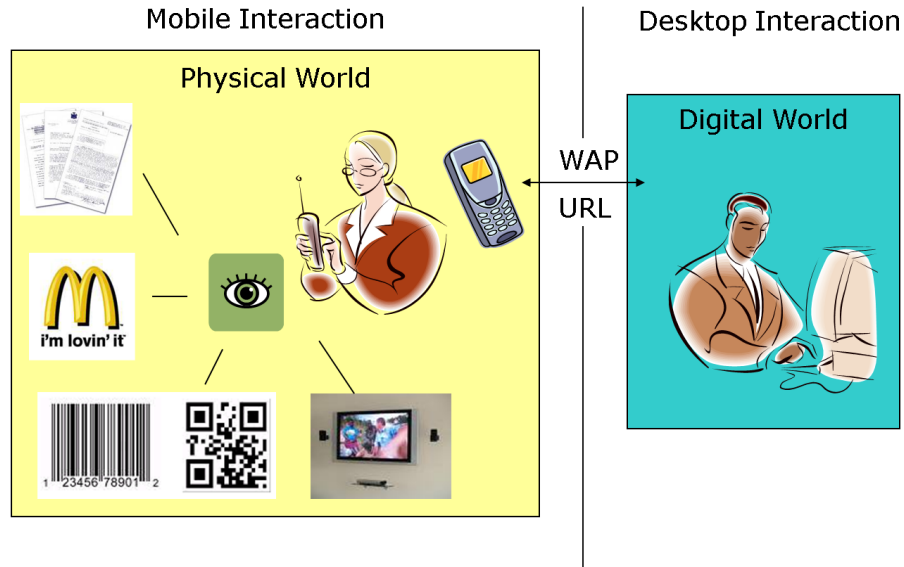


Figure 1.1: Mobile interaction and desktop interaction

ber 2000, with a camera having 0.1M pixel resolution. Cameras have since become an integral “add on” to mobile phones. Based on a Gartner’s report [1], 48% of the mobile phones manufactured in 2006 were embedded with cameras, and this number will increase to 81% in 2010. Moving forward, the camera, originally embedded as an accessory, has the potential to alter significantly the usage of the mobile phone. Powered by increasing computation capabilities and designed primarily for taking pictures, these cameras are being used as ubiquitous input devices [2]. Compared to traditional voice input on these devices, the camera input can gather more information with less effort in a single snapshot. As the proverb says, “*A picture is worth a thousand words.*” More importantly, the embedded camera has created the opportunity for the mobile user to interact with the physical world.

1.2 Novel Usage of the Camera

When using a computer sitting at a desk, the user is often in a “digital mindset” because most of the time he or she is dealing with digital content and can easily neglect the physical world. Using a mobile device, such as a cell phone, can be a different experience. The user typically is not as immersed in the digital world, but is more aware of and interacts with the physical world. The mobile device opens a small window to the digital world so one can receive information but is not completely engaged in the digital world. What we see through this window is a thumb-nailed version of the desktop contents (web, email, or YouTube). Imagine if the mobile device could also “see” what you see in the surrounding physical world and recognize the same objects. Powered by this “mobile vision,” the device can be made aware of its environmental context[3] and can serve as a link between the human, the digital, and the physical worlds. Powered by increasing computation capability, these cameras, are being used as ubiquitous input devices [2, 4] which acquire information from the physical world to for example, to recognize faces[5], road signs [6, 7, 8, 9], text[10, 11], translate documents[12], perform image based search [13, 14, 15, 4, 16, 17, 18]. In our research, we have found several reasons to apply “mobile vision” on the hand-held device.

1.2.1 Input Extension

Input provides the key to access any information gateway (e.g., search engine). We envision the mobile phone acting as a ubiquitous input device [2], but the input

is limited by the small, awkward, slow keyboard. As a hand-held device, its size is limited naturally by users' desire for a compact profile, so the keys remain small and crowded. Limited by the overall phone size, this problem is not easily resolved. The camera provides a great opportunity for input from the physical world. As described in [2] "input" is not confined to typing text and numbers but, in general, can be defined as any information received by the device including ego-motion or visual context used to control the device or trigger events. Unlike traditional keyboard and stylus input, camera input is not limited by the number of keys. More importantly, it may require only a single hand [19]. A combination of camera and keyboard also may increase the input efficiency, including as an extension of traditional device input, for cursor and browsing control, to capture and decode visual information, for data transfer, and to aid users with visual disabilities.

1.2.2 Cursor and Browsing Control

A mobile phone is a pocket-size device with a small display, which makes browsing relatively large scale content (maps, documents or images) difficult. The input device is small as well. In traditional desktop applications, a mouse, track ball or touch screen can control browsing, but the phone usually has only four direction keys (or a four direction joystick). The user has to press the direction key repeatedly to navigate to a desired spot. Unfortunately, the direction key does not capture speed, so all browsing is linear, ignoring important temporal interaction. In our research, we found the camera ego-motion can be used as an ideal tool to

control the browsing, since the ego-motion is omni-directional and continuous. Using motion estimation, the camera phone can be used as a pointing device that controls browsing omni-directionally and continuously. We will present a fast ego-motion estimation algorithm on the camera phone based on hierarchical matching motion prediction in Chapter 2.

1.2.3 Capture of Visual Information

The camera can capture large amount of visual information in one click. This instant input provides a significant advantage compared to linear input methods such as keyboard or handwritten. Combined with the computational and communication capability, the captured information can be digitalized, stored, indexed and shared immediately. The captured information can also serve as the query to a content based information retrieval system. The first attempt at capturing visual information was applied on machine-readable symbologies such as barcodes and, to some extent, on text[10, 11]

1.2.4 Data Transfer

In the ideal world, the data transfer of the phone network should be pervasive, secure, and free. However, most devices require a physical cable or network (data plan) which typically are not free and wireless networks are far from secure. These constraints affect both the content provider and the receiver(phone users). Even if the content provider intends to distribute the information (coupons, ads, ring tones)

for free, the end users may still have to pay for data transfer. Since the phone network is intrinsically bi-directional, the user may not know what information is being transferred while downloading. Furthermore the phone may have private information (contact list, emails), and, in some circumstances, the user would rather download the information passively. We have developed an approach that uses the phone camera as a ubiquitous data channel for passively receiving information. Content is encoded in a visual series of frames that carry information bits, and our software on the camera phone recognize these dynamic patterns in real-time and reconstruct the information to achieve a fee-free passive download. We refer to the encoding as a “V-Code” and discuss it in Chapter 3.

1.2.5 Assistance to People with Visual Impairments

As an electronic eye, the camera phone provides a great opportunity for people with visual impairments see and understand their surroundings better. Compared to optical and electronic image enhancement tools, a camera phone has some unique advantages. First, it is a portable hand-held device that is already carried by a large number of people. Second, new functions can be easily programmed and customized in software with no extra hardware. Third, powered by the device’s general purpose CPU, camera phones may perform sophisticated pattern recognition and image processing algorithms. Fourth, the communication capability of these devices opens a wide range of opportunities to provide access to information and computational resources from the internet. Camera phone based visual assistant

tools have emerged in the field of accessible computing [20, 21, 22, 23].

Realization of these applications are not without serious technical challenges related to imaging, processing and usability. Some unique limitations make the implementation of computer vision applications on the mobile devices fundamentally challenging.

1.3 Challenges

1.3.1 The Limitations of Camera

As the name “camera phone” suggests, the device is first a phone and then a camera. The camera is a feature of the phone and, due to cost considerations and the limited total footprint of the device, the embedded camera does not typically provide as well as a dedicated digital camera. Most devices have a fixed focal length, low resolution and distortion caused by low quality optics, image processing become more difficult, which is especially challenging for applications where we wish to perform analysis of small symbolic content, such as text or barcodes. Nevertheless, addressing these limitations is important to making progress in the field.

1.3.1.1 Fixed Focal Length

In order to fit the optics into a limited size phone, a few manufacturers incorporate focal mechanisms that can adjust the distance between lens and optical sensor, thus limiting the depth of field. For a convex lens, the distance of object p , the distance of image q and the focal length f satisfy the thin lens formula:

$$\frac{1}{p} + \frac{1}{q} = \frac{1}{f}$$

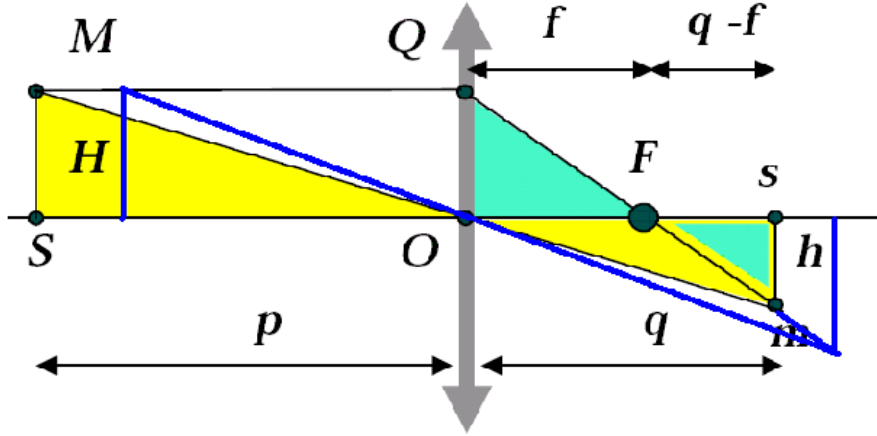


Figure 1.2: Thin lens equation

$1/q + 1/p = 1/f$. When objects are far away relative to the focal length f (which is often less than 1cm) $1/p \approx 0$, and $q \approx f$, the CMOS sensor is therefore usually mounted on the focal plane to capture typical scenes that are on the order of meters from the device. However, when objects are closer to the camera, $1/p$ cannot be approximated by zero and $1/q < 1/f \Rightarrow q > f$. From Figure 1.2 we can see that when an object moves closer to the lens, its image is pushed further back, as the CMOS sensor stays in front of the image, and captures a blurry (de-focused) image. One solution to this problem is to attach a close-up lens (Figure 1.3) over the phone camera lens. We found existing close-up lens (Figure 1.3a) are usually bulky compared to the size of the phone and inconvenient to carry, so we designed one (Figure 1.3b) using two magnetic rings to hold the lens. Our close-up lens can be easily attached to, or removed from, the steel ring mounted on the phone. The attached

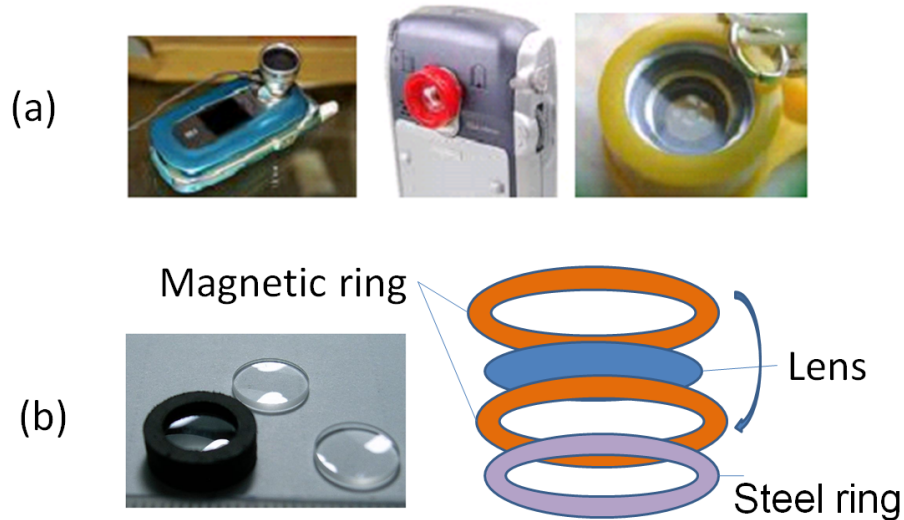


Figure 1.3: Close-up lenses: (a) existing lenses (b) our design

close-up lens decreases the effective focal length of the phone camera, compensates for the increase of $1/p$ and q remains unchanged. An alternative view of a close-up lens is that it converts the rays of light from its focal plane into parallel rays of light. When the parallel light rays go through the original phone camera, they concentrate on the phone camera's focal plane, where the CMOS sensor is mounted. With a close-up lens, the camera takes the best picture of object placed on the focal plane of the close-up lens. For applications which require images taken at a distance D , one should design a close-up lens with focal length D .

1.3.1.2 Low Resolution and Exposure

When Sharp launched its first camera phone product in 2000, the J-SH04 had a camera with 0.1M pixel resolution. Ten years later, some camera phones contain up 100 times the resolution, for example, the Samsung SCH-B600 has a 10M

pixel resolution. Main stream phones, however, still have 1 ~ 3M pixel resolution. Applications are also constrained by the exposure. If one needs to process the images continuously the capture is further constrained. Typically in video mode, 10 ~ 15 QVGA (320x240) resolution images can be captured per second. Some phones may allow a VGA (640x480) preview but with a lower frame rate. The video mode resolution is typically so low for two reasons. The first involves the limited phone screen resolution: VGA remains beyond most of the phone's screen resolution, which means even if a VGA preview was allowed, it would not be necessary for display. Again, the video preview is not designed for image processing! Second derives from the limited bus bandwidth. For a 10fps QVGA 24bits color video mode, $3 \times 240 \times 320 \times 10 = 2\text{M}$ bytes data have to be sent via system BUS from camera to memory in one second, and this is a merely a quarter of VGA mode. This constraint might be relaxed when HD video capture is provided. However, under video mode is no exposure period exists, so the illumination is not well balanced and the image is always darker and blurrier than a with still capture. Although super resolution methods have been proposed to solve this problem in [24, 25], the complexity of the algorithm prevents it from being run on mobile devices. Vision applications must be robust against low image quality to operate under real time video mode.

1.3.1.3 Distortion

Degradation of camera phone image results from both photometric and geometric distortion. A typical distortion observed is the “Fish-eye” effect when the center of an image has better contrast, sharpness and less geometric change than the border area. Figure 1.4(a) shows a closeup of part of a PDF417 barcode captured at approximately 5 cm. Straight lines are bent in the image and when alignment is required, for example for barcode recognition, this distortion might cause problems. Photometric distortion is shown in Figure 1.4(b): the center area is brighter than the border area. This lighting distortion becomes more obvious when capturing objects close to the camera and less so for natural scenes. A shadow map[26] can be computed and subtracted to relieve the lighting distortion but cannot remove it completely. The user can avoid fisheye distortion by having a high resolution camera and positioning the object far away from the camera.

1.3.1.4 Uneven Lighting and Shadows

Uneven lighting and shadows present challenges to many vision tasks. In many applications shadows can be cast by the camera phone itself, such as, when the hand is holding the camera phone. The shadow will disturb illumination sensitive algorithms. Typically an illumination normalization (binarization) step is required before processing an image with shadows.

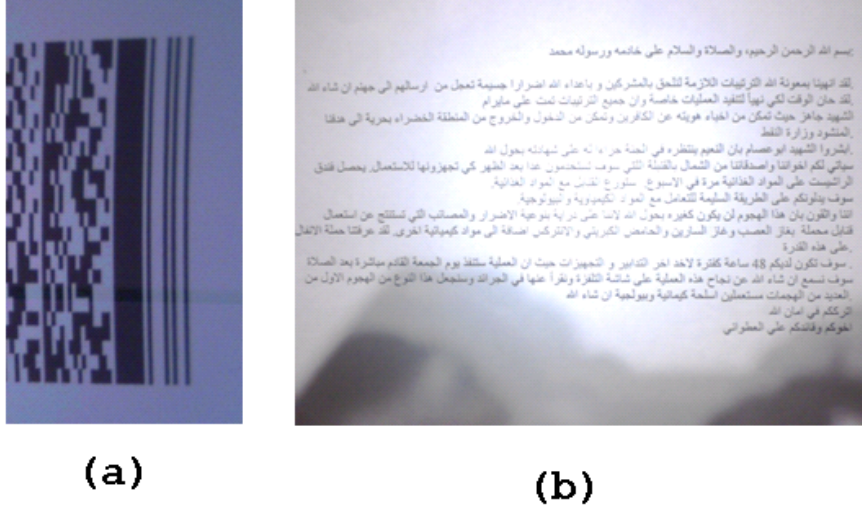


Figure 1.4: Fisheye distortion

(a)Geometrical distortion (b)Photometrical distortion

1.3.2 Computational Power and Battery Life

When we require processing to be performed on the device, the power of the device can also be an issue. Embedded devices often have battery power which limits the computational ability in three ways. First, the architecture of the phone system is not designed to optimize CPU speed. As high performance CPUs use battery more rapidly. Second, generic RISC architectures without a Floating Point Unit, such as the ARM9 series CPU, are often employed. These suffice for general use, but for image processing and geometric vision computation, floating point operations are often hard to avoid. As a compromise, an SDK for the ARM9 contains a software simulated floating point library, but it is extremely slow. Third, multiple tasks may run concurrently. On Symbian phones, the average number of concurrent threads is 100, and a privilege of power is further reserved for essential tasks such as

processing calls. From the software development point of view, we cannot do much about the system and hardware architecture. One possible solution is to improve the algorithm's efficiency, by avoiding floating point operation, using approximated algorithms and a lower image resolution. We will address speed issues for mobile devices in later sections.

1.3.3 Human Factor Challenges

1.3.3.1 The Moving Camera

Unlike a camera mounted on a tripod or surveillance camera mounted on a wall, a camera phone is typically in motion. When capturing a snapshot, users will not hold it with two hands like a digital camera, but operate with one hand: "open preview - select scene - click to capture - finish". Motion blur may occur right at the moment of "capture" and can be further magnified by slow exposure. Although the motion might be small, it could blur the captured image. Currently no general purpose image stabilization (IS) function exists on any camera phone [27, 28] and traditional de-blurring algorithms may be too slow to apply on the device[29]. Solutions have been purposed for stabilization of special targets such as 2D barcodes [30]. When shooting a video, the hand moves more frequently. Even if users intend to hold the camera phone still, they typically will not be able to remain stable for a long period of time (e.g., > 1 minute). As suggested in [19], most mobile phone users prefer to operate the phone with one hand, and we use this as a guideline when we design mobile vision recognition applications. We strive for single hand

operation and a limited number of key presses.

1.3.3.2 User Tolerance

Phone users do not focus on their device in the same way as they do with desktop computers. This may become a problem when vision and or recognition tasks take tens of seconds to complete. A user study [31] shows that the mobile user on average pays no more than 4 seconds of continuous attention to the device, and user attention can be more divergent in daily usage than in lab tests [32]. If a transaction cannot be finished in less than 10 seconds, it is better to put it in the background or run the task remotely on a server. The server side computation currently has data transport problem as it may take minutes to send an SVGA(1024x768) image through today's wireless phone network. When 3G (CDMA) network overtakes 2G (GPRS) and 2.5G (EDGE) more computation may be done remotely and the "real" pervasive computing on the phone will arrive.

We must also attempt to reduce user input, especially for keyboard input since it is inconvenient. The ideal case is to have a "key-press-free" operation after launching an application.

1.4 Organization of Dissertation

The remainder of this dissertation is arranged as follows. We first present an overview of our technical contribution to the literature in Chapter 2. We survey related techniques and discuss why they cannot be directly applied to mobile appli-

cations, and highlight our fundamentally new or improved algorithms for the mobile devices. These techniques will be used as building blocks to promote new mobile vision applications as described in the remainder of the dissertation.

In Chapter 3 we present a novel usage of the mobile camera as a passive secure data channel - the “V-Code”. We address the problem of information uncertainty in the visual communication channel. We introduce a novel spatial-temporal error correction method to encapsulate data being sent through a highly vulnerable channel with both error and data loss. As a theoretical contribution, we estimate the capacity of the camera channel using mutual information theory.

In Chapter 4 we discuss “MobileRetriever,” a system that links a physical document with its digital source using a camera-enabled mobile device. We provide indexing and retrieval solutions for both in-focus and out-of-focus document images and evaluate our system on a database with 100,093-page document images. In our approach, we have two key contributions to the field of content based image retrieval 1) Using layout verification we address the problems of rejecting queries not contained in the database. 2) We estimate the minimum portion of a document page that needs to be captured as query.

In Chapter 5 we present a series of applications, referred as “MobileEye,” that use mobile vision and image processing techniques to help people with visual impairments. We address the problem of real time object recognition when the user cannot take a perfect image for recognition. Finally, we discuss future work and conclude the thesis in Chapter 6.

Chapter 2

Technical Contributions and Related Research

In this chapter we present a series of computer vision and image processing techniques that we have developed for mobile applications. These techniques represent the basic elements required to pursue mobile applications that involve vision and recognition. The techniques being described relate to vision and recognition from the images of planar objects such as documents, signs and symbologies, but they may also be applied to other objects. These techniques span the areas of pattern recognition, geometric vision (e.g., perspective correction), ego-motion estimation, and pixel level image processing (e.g., contrast enhancement and interpolation). In one common feature shared by all the techniques introduced in this chapter, they all have been implemented, tested, and optimized for mobile devices. They serve as building blocks to develop the vision system and applications throughout this thesis and will be referred to in the following chapters. We also survey related research and discuss how each contributes to the field.

2.1 Perspective Correction

Perspective correction is important for normalizing a planar scene captured from an arbitrary view angle. It has wide application in the camera based recognition of symbologies, signs, document images, and other planar objects. These objects

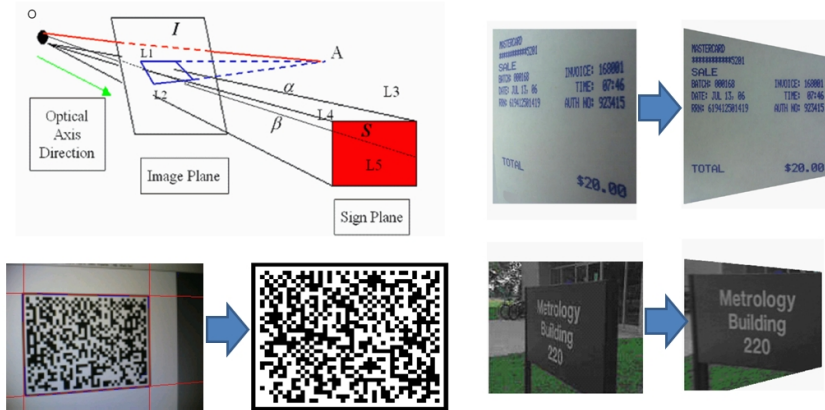


Figure 2.1: Perspective Correction

present the majority of the targets that we will process and recognize with mobile applications. We need to rectify them using perspective correction before performing analysis. In this section, we introduce a fast perspective correction algorithm using four known corners (edges or equivalent) within a captured image. Compared to traditional perspective correction algorithms [33, 34] our new algorithm is fast in that it avoids the floating point computation and avoids explicative computation of the homography.

The classic way of correcting perspective distortion is to calculate the mapping between the ideal, non-perspective image and the real-captured image, which can be described as a plane-to-plane homography matrix \tilde{H} [35, 36]. For any matrix entry (i, j) , \tilde{H} maps homogeneous coordinate $x = (i, j, 1)$ to its image coordinate $X = \tilde{H}x$. Suppose we know n matrix entries $(x_i, y_i, 1)^T$, and their corresponding image points $(X_i, Y_i, 1)^T$, where $i = 1, 2, \dots, n$. The classic way of computing \tilde{H} is using a homogeneous estimation method [33, 34]. First, we reshape matrix \tilde{H} as a

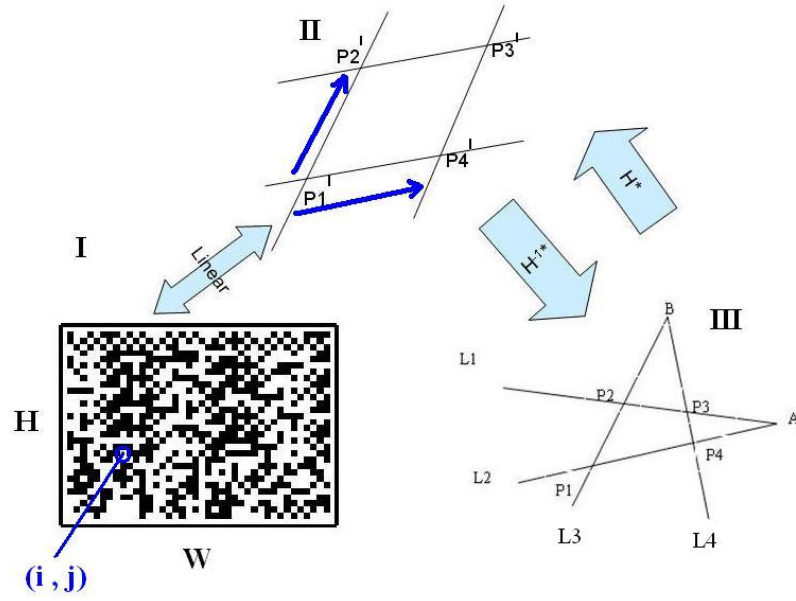


Figure 2.2: Geometrical transformation between the matrix and perspective images

vector $\tilde{h} = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T$, and then solve for

$$M\tilde{h} = 0 \quad (2.1)$$

where M equals

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 & -X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 & -Y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 & -X_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 & -Y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nX_n & -y_nX_n & -X_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_nY_n & -y_nY_n & -Y_n \end{pmatrix} \quad (2.2)$$

When $n > 4$, \tilde{h} can be obtained using a pseudo-inverse as a least-square solution. When $n = 4$, \tilde{h} is the null vector of M , and we have a unique solution for \tilde{h} (assume $|\tilde{h}| = 1$ or $\tilde{h}_{33} = 1$). This means that we only need the coordinates of the four corners (P_1, P_2, P_3, P_4) in Figure 2.2 to compute the \tilde{H} . A typical solution requires LU decomposition with pivoting, and it involves a significant amount of floating point calculation, currently unsupported by mobile phones at the hardware level. Although, the development environments (including Symbian SDK and Windows Mobile SDK) provide a software emulation of IEEE-754 64 bit floating point, it is much slower than integer operations. Other platforms, such as Java(J2ME), provide no floating point capabilities. This motivates us to design a simpler/faster algorithm without floating point operations, and, in the remainder of this section, we map our approach.

As shown in Figure 2.2, we first perform an affine transformation and then a perspective transformation. Suppose the coordinates of four corners in the image plane are P_1, P_2, P_3, P_4 , and the top and bottom boundaries of the bounding box intersect at vanishing point A . Then under homogeneous coordinates $A = L_1 \times L_2 = (P_1 \times P_4) \times (P_2 \times P_3)$. Similarly the left and right boundaries intersect at a vanishing point $B = L_3 \times L_4 = (P_1 \times P_2) \times (P_3 \times P_4)$. A and B are infinite points in the original plane. The third element of A and B under homogeneous coordinates should be 0 in the affine image (Figure 2.2 II). Any homography $H = \begin{pmatrix} \vec{H}_1 \\ \vec{H}_2 \\ \vec{H}_3 \end{pmatrix}$ that maps the perspective image back to the affine image should map A and B to infinity, which

implies

$$\begin{cases} H_3 \bullet A = 0 \\ H_3 \bullet B = 0 \end{cases} \Rightarrow H_3 \sim A \times B \Rightarrow \quad (2.3)$$

$$H_3 \sim ((P_1 \times P_4) \times (P_2 \times P_3)) \times ((P_1 \times P_2) \times (P_3 \times P_4))$$

This suggests that we can calculate H_3 using seven cross products. Any homography H with the third row H_3 computed by (2.3) maps the perspective image (III in Figure 2.2) to an affine image (II). The next task is to define the first and second row of H . We calculate this homography H so that we can quickly tell its pixel coordinate in the image given any matrix coordinate. From the matrix coordinate (I) to the affine image (II), the transformation is linear and can be directly computed by transforming the base of the coordinate system. In the final step, we transform the affine image (II) to the perspective image (III) by computing H^{-1} . Therefore, we choose the first and second row of H so it has a clean inverse. With

$$H = \begin{pmatrix} h_{33} & 0 & 0 \\ 0 & h_{33} & 0 \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \quad (2.4)$$

we have (up to scale)

$$H^{-1} \sim \begin{pmatrix} h_{33} & 0 & 0 \\ 0 & h_{33} & 0 \\ -h_{31} & -h_{32} & h_{33} \end{pmatrix} \quad (2.5)$$

This “inverse” only requires the reverse of two signs in the third row of H . In this way it accelerates the coordinate transformation with numerical stability.

Table 2.1: Speed of Perspective Correction (frames per second)

Platform	Nokia 6680	iMate Jamin
CPU	ARM 9T	TI OMAP850
Classic homogenous estimation	1.2 fps	1.9 fps
Fast perspective Correction	8.2 fps	10.5 fps

Normally the numerical inverse is subject to “division by zero” when H is nearly singular. In summary, we compute the coordinate transformation as follows:

1. Compute H_3 using equation (2.3),
2. Compute H and H^{-1} using equation(2.4) and (2.5),
3. Map P_1, P_2, P_3, P_4 to affine points P'_1, P'_2, P'_3, P'_4 using H , and
4. For any entry (i, j) in the $w - by - h$ matrix compute its affine coordinate $\frac{i}{w}\overrightarrow{P'_1P'_4} + \frac{j}{h}\overrightarrow{P'_1P'_2}$ and use H^{-1} to map this affine coordinate to the image coordinate.

The above algorithm does not require floating point computation. We compare this algorithm to the classic homogenous estimation algorithm which computes the homography using floating point operations and show the results in Table 2.1. The speed was measured by rectifying a 100×100 square sub-image from a 320×240 image and counting the number of frames processed per second.

2.2 Pattern Detection and Recognition

Pattern recognition may serve as an important capability in the pursuit of new mobile applications and will lead us to the goal of making the physical world “clickable”. For example, a recognized logo may lead the user to digital coupons or online ads, or a recognized face may serve as the key to open the mobile device. Pattern recognition can also help people with visual disability to identify objects such as currencies in daily life. In this section, we describe efficient technology for “target based” recognition.

Classic pattern recognition algorithms usually include feature extraction and feature classification as two core components. Popular features such as SIFT[37] or SIFT-like[38] have high repeatability. SVM [39] and neural networks [40] can be trained to achieve high accuracy given enough time and space allowance. However, these classic pattern recognition approaches cannot be ported directly to mobile devices. As we noted in the introduction, implementing pattern recognition on mobile devices has three major challenges. 1) The limited processing power of the device, 2) the fact that the captured scene could contain complex background resulting in false positive that must be eliminated, and 3) the expectation of the user who typically expects instant feedback and requires online (real time) recognition.

These three challenges are related to the speed and efficiency of the algorithm. The algorithm must be efficient enough to fit in the light-weight device and be able to discard images quickly or pixels that are not of interest so more time can be allocated to the image that contains objects to be recognized. Ideally, when an

algorithm is efficient enough that it can run in real time, the recognition can be performed on the video stream of the camera, and the user does not have to hit a key to capture an image. Real time recognition gives a smooth user experience and avoids motion blur caused by “click-to-capture,” but, as noted earlier, it must typically deal with lower quality data.

In our approach, we tackle these changes in two steps to achieve high accuracy, real time recognition on mobile devices. First, we use a very fast pre-classifier to filter the images and areas of an image with low probability of containing the target. This step is inspired by the Viola-Jones face detector [41] and the Speed-Up Robust Feature [42], both of which use box filters to detect objects and features rapidly. Second, we use a set of local pixel pairs to form weak classifiers and use ada-boost [43] to train strong but fast classifiers from these weak classifiers. The idea of using local pixel pairs is inspired by Ojala, et al.’s work on Local Binary Patterns (LBP[44]) and more recent work by Pascal, et al. [45]. The advantage of local pixel pairs lies in their robustness to noise, blur and global lighting changes which are significant challenges for mobile recognition. The details are presented in the next two sub-sections.

2.2.1 Pre-classification and Filtering

To detect an object in an image, an exhaustive search is usually inefficient because most of the areas in the image do not contain the object in which we are interested. A pre-classification which filters these areas is therefore important and

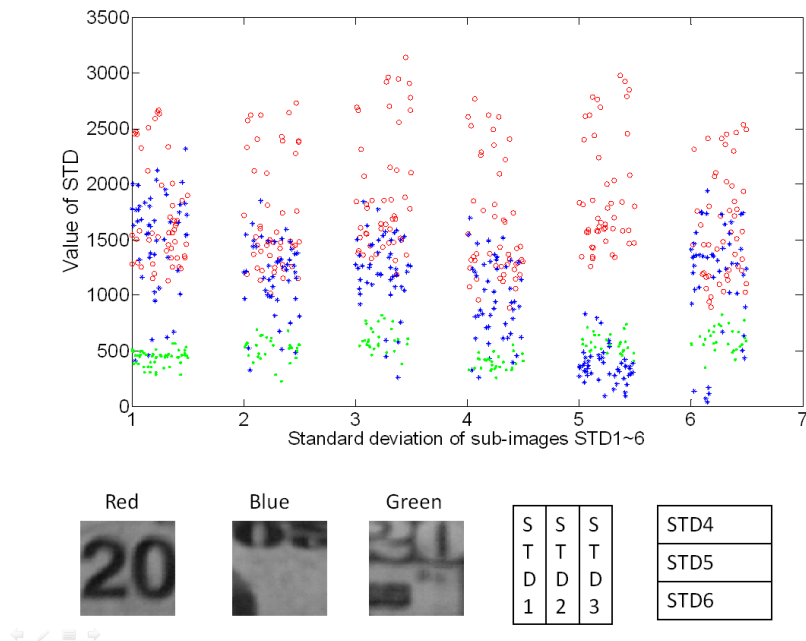


Figure 2.3: Standard deviation of 6 sub-images at 3 corners of a 20 dollar bill

can speed the detection by ten times or more. In our research we found that a box filter computed using an integral image is very efficient and can be applied to mobile devices. In an integral image, at each pixel the value is the sum of all pixels above and to the left of the current position. The sum of the pixels within any rectangle can be computed in four table lookup operations on the integral image in constant time. If we replace the original image with an image with the squared gray scale value at each pixel, we can then compute the standard deviation (second order moment) within any rectangle in $O(1)$ time. Any order of moment can be computed in $O(1)$ time using an integral image. Both the Viola-Jones face detector[41] and SURF[42] benefit from the speed of the box filter and integral image.

We found that the standard deviation (STD) in the sub-image of an object is relatively stable and combination of STDs can be used as a cue to filter non-

interest image areas. In Figure 2.3, we divide the corners of a twenty dollar bill into 6 boxes (3 vertical and 3 horizontal). The STD in each sub-window falls in a relatively stable range and we search only within these ranges for the potential corner patterns to recognize. In each sub-window an STD range may span at most $1/2$ (red) or even less (blue) of possible STD. Assuming the STD in each sub-window is independent and is equally distributed in an arbitrary scene, the box filter can eliminate $1 - (1/2)^6 = 98.4\%$ of the computation by discarding low probability regions. In the currency reader research discussed in Chapter 6, we find the pre-classification can speed the algorithm by 20 times on a camera phone.

2.2.2 Fast classification with Random Pixel Pairs

In this section we introduce a fast classifier based on random pixel pairs. One challenge of pattern recognition on the mobile device is usability. Traditional “capture and recognize” approaches may not be friendly to mobile users. When the recognition fails (because of motion blur, de-focusing, shadows, or any other reason) the user will have to click and try again. Instead, real time recognition is preferred. At the same time, recognition may benefit from the fact that images are taken by a cooperative user. Unlike a mounted camera with the objects moving in the scene, the camera (phone) is moving itself and the user is usually approaching the object he intends to recognize. We assume that the object rests approximately to the same relative position of the camera when being recognized. Under this assumption, we can subtract the background, extract and normalize the object, then perform the

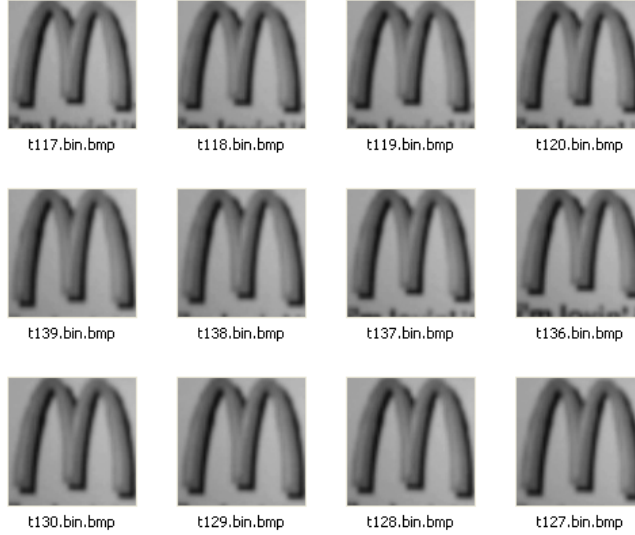


Figure 2.4: Positive samples: McDonald’s logo

recognition under a relative stable setup. Our primary concern comes from the features which are the key subject of this section. The features we seek here must be distinct (not to raise any false alarms), robust (to tolerate weak perspective and lighting variation) and fast to be computed on the phone. When considering feature options, the first consideration may be SIFT [37] key points which outperform most of other features in terms of accuracy. However, the speed of SIFT is a significant challenge for mobile devices. The Gaussian convolutions are too computationally intensive for most of today’s mobile devices. The recent results of efficient and robust feature (point) extraction are all built from simple elements such as box filters and random pixel pairs [45, 42].

Our feature set consists of a large set of binary values of pairwise intensity comparisons. For a $w \times h$ image, there are as many as $C_{w \times h}^2$ different pairs of pixels to compare. Our goal aims to find those pairs that uniquely define the object of

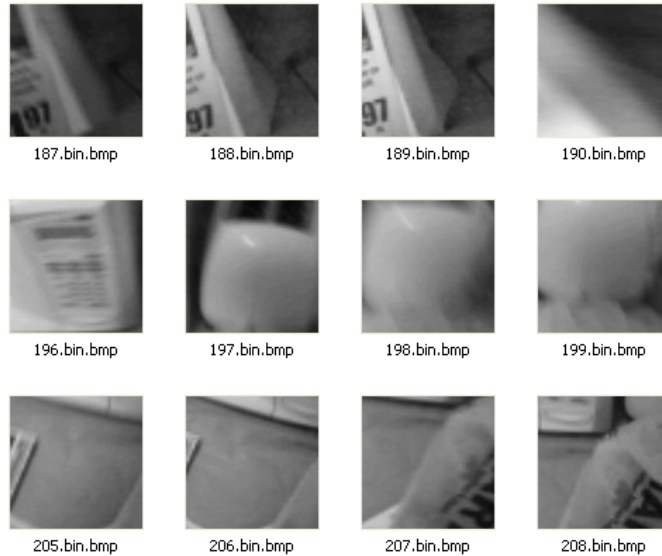


Figure 2.5: Negative samples: arbitrary scene

interest. We achieve this goal with a learning approach.

Consider attempting to recognize a McDonald's logo (Figure 2.4). We first perform background subtraction and normalization, and collect a set of positive samples (Figure 2.4) and an even larger set of negative samples, Figure 2.5. We train our recognizer by selecting discriminating intensity pairs. For each pair of pixels (i, j) , we define $P^+(i, j)$ to be the number of positive samples with greater intensity pixel i than at pixel j , similarly, define $P^-(i, j)$ to be the number of negative samples with greater intensity pixel i than at pixel j . Our goal will then be to choose pairs (i, j) to maximize $P^+(i, j)$ and minimize $P^-(i, j)$. One naive way to achieving this goal is to maximize $P^+(i, j)/P^-(i, j)$, but this will not work because the large collection of negative samples make $P^-(i, j)$ almost random. Nevertheless, numerous pairs satisfy all the positive samples, among which we would like to choose the most distinct ones.

Although the number of hits of a pair (i, j) in the negative samples cannot help us judge whether the choice of (i, j) is good, it can attempt to measure the distance from the closest negative sample to the positive samples. As shown in Figure 2.6, we will maximize the margin between positive samples and the closest negative samples by rating them. A higher score indicates a higher probability of an inlier and lower scores for outliers. After training, we can use a threshold on scores to classify the pattern. Using this criteria, we develop the following algorithm:

Assign an initial score of zero for all negative samples and keep a pointer m that always points to the highest score. This can be done efficiently using a heap.

1. Generate a random pair (i, j) .
2. If (i, j) satisfies a negative sample m , then go back to step 1.
3. If (i, j) does not satisfy the all positive samples, go back to step 1.
4. For all negative samples, increase its score by 1 if it satisfies pair (i, j) and modify pointer m to point to the negative sample with highest score (hence, the score of m is not increased).
5. Go back to 1 until we have n pairs.

Using this algorithm, we collect n discriminating pairs that represent the object. Suppose we also have a score for the positive samples. During each round, all positive samples get increased by 1, while the closet negative sample does not. The gap between positive and negative samples is enlarged. In the recognition phase, we will use this score to judge whether an object is recognized or rejected, and our

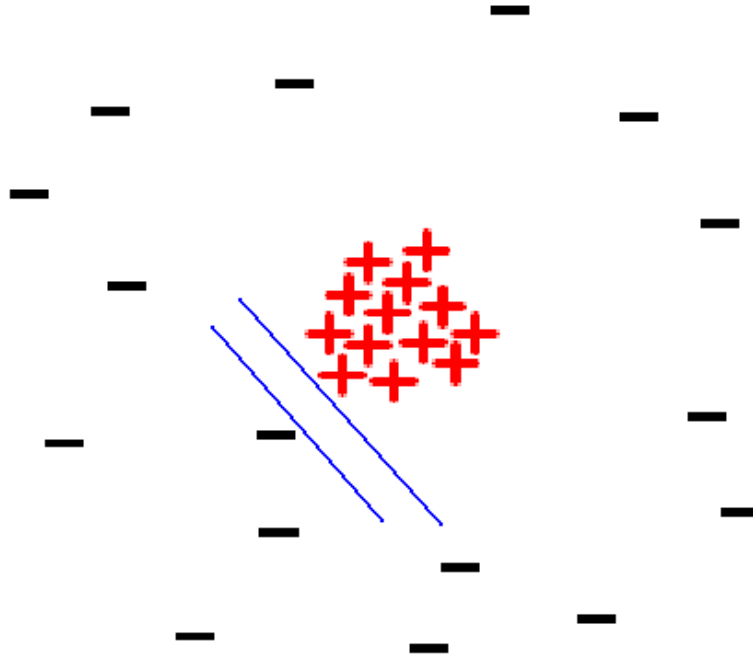


Figure 2.6: Maximize margin between positive and negative samples

threshold will be placed in the middle of the gap. Ideally we would like to see all positive samples with high scores and all negative samples with low scores. However, there might be a negative sample that is similar to the inliers, and our algorithm can spot the most confusing outliers and establish a boundary between the inliers and those outliers. The accuracy of this detection algorithm is further enhanced using the Ada-boost[43] algorithm.

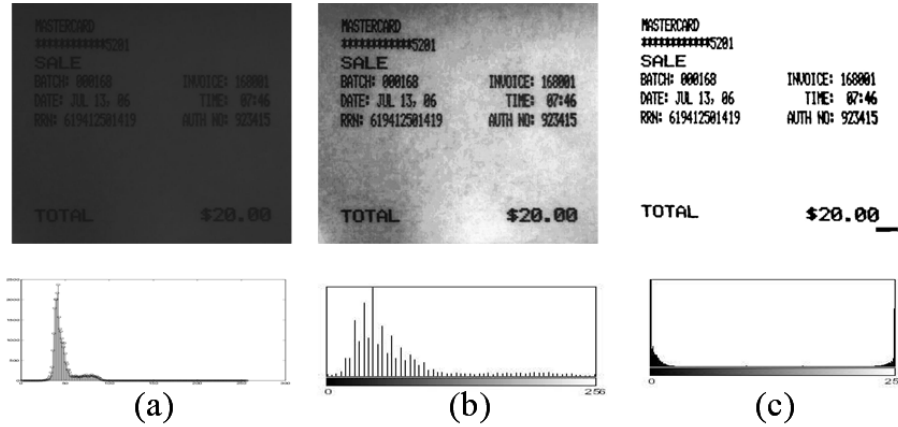


Figure 2.7: Adaptive contrast enhancement method compared with histogram stretching based method

2.3 Pixel-wise Manipulation and Enhancement

2.3.1 Contrast Enhancement and Adaptive Binarization

Ideally, the pixels in an image should use the entire range of intensities for maximum contrast, but under some adverse imaging conditions the majority of the pixel values may lie in a narrow range, making them more difficult to determine. The contrast enhancement technique typically implemented by stretching the range of values where most of the pixels can be distinguished with subtle contrast differences. Contrast enhancement is important for processing the image of documents or document-like objects (signs, symbologies), which are pseudo-binary and dominated by two major colors, usually black and white.

Contrast enhancement can be described mathematically as $s = T(r)$, where r is the original pixel value, T is the transformation, and s is the transformed value. T can be linear or non-linear, depending on the practical imaging conditions. The

principle task here is to make light pixels lighter and dark pixels darker, so the total contrast of an image is maximized. Although histogram stretching is a common and effective approach to general contrast enhancement, it does not necessarily produce the best results for text enhancement. This can be illustrated in Figure 2.7. Figure 2.7a shows the original low-contrast image, and Figure 2.7b presents the contrast enhanced image. Although the contrast is increased, some background pixel values are also stretched, which makes the text difficult to read in some places. A detailed analysis helps us understand the problem better. As we can see in Figure 2.7a, the original histogram has two peaks, with the larger one representing the background pixels and the smaller one representing the text pixels. After histogram stretching, we can see the smaller peak almost disappears because the intensity value corresponding to this peak comes too close to the background. As a result, many of the background pixels combine with the text pixels after histogram stretching. With cluster-based contrast enhancement, text and background pixels form two clusters. When image contrast is high, the distance between two cluster centers is larger, and vice versa. We have developed an innovative clustering based contrast enhancement method, that uses this unique feature of text images for contrast enhancement. Specifically, we first find the two clusters, then enhance the contrast between them. The algorithm consists of the following steps:

Initialization. Choose two initial cluster centers $C1(0)$ and $C2(0)$, as random values between 0 and 255. The convergence can be accelerated if $C1(0)$ and $C2(0)$ are selected as values between the minimum, maximum, and the mean of the image pixel values.

Pixel Clustering. For each pixel in text image $I(i, j)$ at iteration n , calculate the minimum distance:

$$d(i, j) = \arg \min |I(i, j) - C_k(n)|, k = 1, 2 \quad (2.6)$$

The pixels are allocated to the cluster with the minimum distance. In this way, all the pixels are partitioned into two clusters $C1$ and $C2$ based on this distance measure. The error at iteration n is calculated as:

$$e(n) = \frac{1}{M \times N} \sum_{i=0}^M \sum_{j=0}^N d(i, j) \quad (2.7)$$

where $M \times N$ is the size of the image. The iteration stops when $e(n)$ is smaller than a preset threshold or n reaches a preset threshold. In this way, the number of iterations can be limited and the time spent on processing is acceptable.

Updating. Generate the new location of the center by averaging the pixel values in each cluster:

$$C1(n) = \frac{1}{N_{C1}} \sum C1(i, j); C2(n) = \frac{1}{N_{C2}} \sum C2(i, j); \quad (2.8)$$

where N_{C1} and N_{C2} are the number of pixels in $C1$ and $C2$ respectively. The iteration stops when $e(n)$ does not decrease.

Adaptive Stretching. After two cluster centers are determined, we can place one center at a small value (0, for example), and another at a large value (255, for example), and stretch the histogram based on these two centers. Figure 2.7c shows the contrast enhancement result based on our method. We can see the image

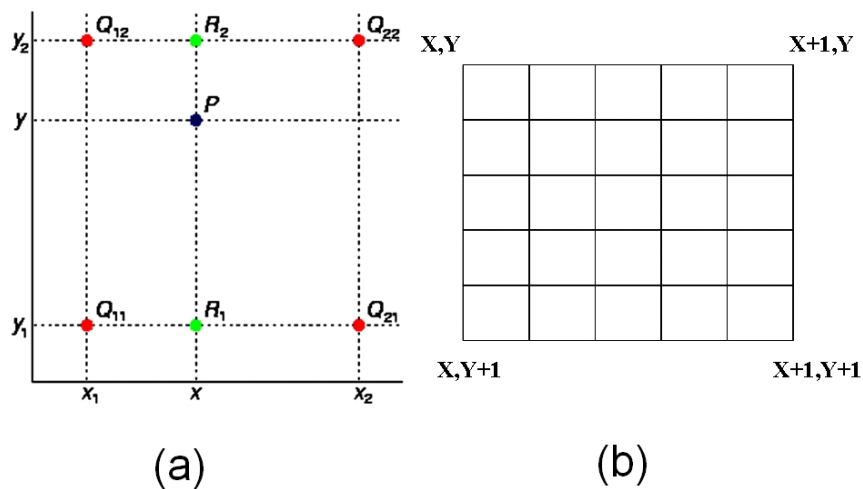


Figure 2.8: Bilinear Interpolation

readability has significantly improved compared with either the original image or the general histogram based method.

2.3.2 Rapid Bilinear Interpolation for Binary Images

As discussed in the previous section, binary images or pseudo-binary images such as documents and signs are one major class of images we are processing for mobile vision applications. When these documents are normalized, rectified or presented to the user, they are usually subject to geometrical transformation (zoom, rotation, affine, or perspective). These transformations are used pervasively so its speed becomes crucial, and we will introduce a rapid bilinear interpolation algorithm for geometrical transformation of binary images.

When an image is geometrically transformed, a pixel in the new image is often projected back to a point with non-integer coordinates in the original image (point P

in Figure 2.8a). We need to estimate the value of point P from its four neighboring points with integer coordinates $(Q_{11}, Q_{12}, Q_{21}, Q_{22})$. To determine the pixel value at point P, the linear interpolation in X direction is first performed:

$$\left. \begin{aligned} f(R1) &\approx \frac{x2 - x}{x2 - x1} f(Q11) + \frac{x - x1}{x2 - x1} f(Q21) \\ f(R2) &\approx \frac{x2 - x}{x2 - x1} f(Q12) + \frac{x - x1}{x2 - x1} f(Q22) \\ \text{where} \\ R1 &= (x, y1) \\ R2 &= (x, y2) \end{aligned} \right\} \quad (2.9)$$

Then interpolation in Y direction is performed:

$$f(P) \approx \frac{y2 - y}{y2 - y1} f(R1) + \frac{y - y1}{y2 - y1} f(R2) \quad (2.10)$$

Substituting 2.9 into 2.10, we have

$$\begin{aligned} f(x, y) &\approx \frac{f(Q11)(x - x2)(y - y2)}{(x1 - x2)(y1 - y2)} + \frac{f(Q21)(x - x1)(y - y2)}{(x1 - x2)(y1 - y2)} \\ &+ \frac{f(Q12)(x - x2)(y - y1)}{(x1 - x2)(y1 - y2)} + \frac{f(Q22)(x - x1)(y - y1)}{(x1 - x2)(y1 - y2)} \end{aligned} \quad (2.11)$$

From this formula we can estimate how many floating point multiplication and addition operations are required to finish the process. We know that $x2 - x1 = 1$ and $y2 - y1 = 1$, so we need to calculate $f(Q11)(x - x2)(y - y2)$, $f(Q21)(x - x1)(y - y2)$, $f(Q12)(x - x2)(y - y1)$ and $f(Q22)(x - x1)(y - y1)$, each of which requires two floating point subtraction and multiplication operations. Therefore, each pixel in the interpolated image will requires $2 \times 4 + 3$ floating point additions (subtractions),

and 2×4 floating point multiplications. This means the interpolation of an image at VGA (640×480) resolution requires $640 \times 480 \times 11 = 3,379,200$ floating point additions, and $640 \times 480 \times 8 = 2,457,600$ floating point multiplications. We tested the interpolation of an image size 320×240 to VGA resolution on a Dell x50v PDA (650MHz XScale CPU, 64MB memory), and the process took almost two minutes. The reason of the slow speed is that mobile devices often use software emulation to process floating point calculations instead of using a specific hardware floating point processor as is typical on a PC.

When the image being transformed is binary, we can use one bit to represent each pixel: black for foreground and white for background, or versa visa. This holds true, for example, with document images. In this case, $[f(Q11), f(Q21), f(Q12), f(Q22)]$ has at most 16 combinations, and we can quantize $x - x1 = 1$ and $y - y1 = 1$ at a small step interval t , as shown in Figure 5b. Each pixel in the interpolated image will correspond to one grid point in Figure 2.8b. Therefore, we only need to pre-calculate the pixel values at each grid point and store them. The smaller t is, the larger the number of grid points is, and the more memory is required to store the values. We set $t = 0.01$. The size of the look-up table that stores all the pixel values is $100 \times 100 \times 16 = 160KB$. After pre-computing the look-up table, the image interpolation becomes a memory-indexing process without any floating point calculations. We tested the approach on a Dell x50v PDA, and it takes approximately 10 milliseconds to interpolate an image at VGA resolution. This is 200 times faster primarily due to the elimination of all floating point calculations, and the use of a look-up table. The cost of this acceleration is 160KB extra memory.

2.4 Ego-motion Estimation

In this section we present a fast motion estimation algorithm for camera-enabled mobile devices. The algorithm runs in real time and the estimated motion can be used to control the cursor and browsing on the mobile devices or for gesture recognition[46, 47, 48]. Traditional ego-motion estimation approaches can be categorized as direct[49] or feature-based[50]. Direct methods are faster but feature based methods may have higher accuracy, and they have both been applied to mobile devices [51, 52]. We do not need very accurate motion to control the cursor position and will rely on user adaptation, so we can use a direct method with a coarser-to-fine matching for acceleration.

2.4.1 Hierarchical Matching

The idea of hierarchical matching originates from MPEG [53] encoding, which encodes the motion of a block of pixels from frame to frame to compress the video data. The situation differs for camera motion. The scene is fixed generally, so the motion is caused by the camera. Consequently, blocks in a frame should move in the same general direction, and we estimate that direction and the magnitude of the motion.

We treat the motion estimation between frames X_0 and X_1 as an optimization problem, where the estimated motion \vec{m} is defined by (2.12):

$$\vec{m} = \arg \min_{\substack{(dx, dy) \\ (x, y)}} \left(\sum |X_0(x, y) - X_1(x + dx, y + dy)| \right) \quad (2.12)$$

To estimate \vec{m} we populate a pyramid of images from coarse to fine and perform

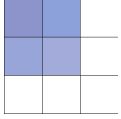
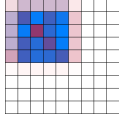
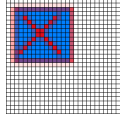
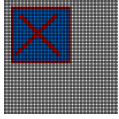
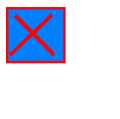
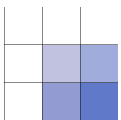
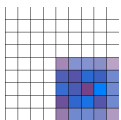
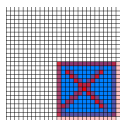
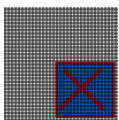
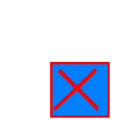
				
				
Layer1 - 3x3	Layer2 - 9x9	Layer3 - 81x81	Layer4 - 243x243	Original image

Table 2.2: synthetic images at 5 resolutions

the estimation on each layer of the pyramid. Each finer level uses the result from the previous coarser level as a rough estimate to limit the search space.

Table 2.4.1 shows the hierarchical matching procedures using synthetic images. From the top level (Layer-1) we estimate the motion to $\overrightarrow{(1, 1)}$ super pixels. When we move to Layer 2, the motion is estimated to $\overrightarrow{(4, 4)}$. As we go down to finer levels, we get a more accurate estimate of the motion based on an estimate of the previous layer. Equation (2.12) is applied to each layer. From a signal processing point of view, we first estimate the motion of the lowest frequency composition of the image, then estimate motion of the higher frequency composition. The motion of the higher frequency composition depends on the motion of lower composition. When we down-sample the image, we convolve the image with a rectangular signal to weaken the high frequency components. In practice, we construct a hierarchical image sequence from the original image, each of which is a smoothed version of the previous one. The pyramid algorithm for MPEG encoding[53] is a mature implementation of this technique. However, for video encoding, the motion focuses on the content inside the

image rather than on the motion of the camera itself. Here we simply use a casual multi grid (e.g. a low pass) filter. One alternative is the algebraic multi grid (AMG). Sharon[54] introduced AMG for image segmentation and proved its efficiency and accuracy, but the computational complexity is far beyond the limitations of today's hand-held devices.

2.4.2 Optimizing the Search Space

The main reason for estimating motion is to enable real time control of the interface. Speed is crucial to real time motion estimation because, when the frame rate is low, matching points may not exist on successive frames. When the device is moved, the motion is generally continuous, with sharp turns or rotation being rare. The trace of the device should be smooth rather than zigzag. The camera's motion can thus be regarded as a Markov random walk because the next movement relies only on the most recent steps. A rough approximation of the device movement is a Markov random walk with a diagonal transition matrix, as the motion tends to repeat. This is the reason a Kalman Filter can be applied for post processing to smooth the trajectory [51].

In our approach, the motion is estimated by hierarchical matching which is essentially a search problem, so the speed is directly related to the dimension of the search space. We have included a branch cut mechanism in our algorithm where the current best match is used as the lower bound. When a motion direction is obviously wrong (worse than the lower bound), we stop matching in that direction.

The earlier we reach the best match, the more incorrect directions can be eliminated.

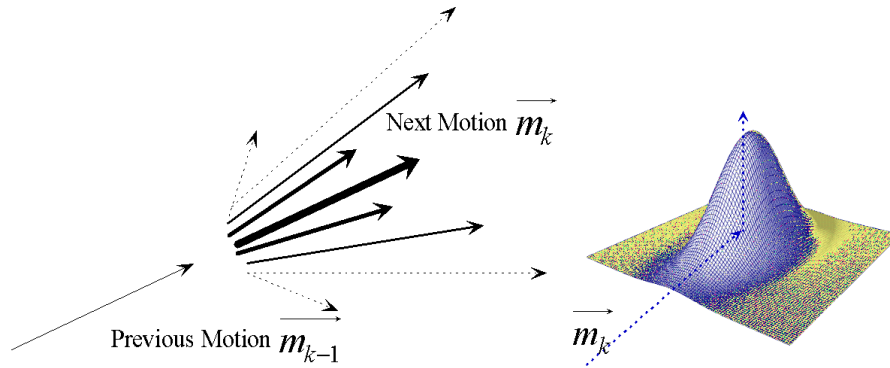


Figure 2.9: Motion predication

Left: next motion predicated by previous motion, bold indicating high possibility.

Right: possibility of next motion modeled as a Gaussian distribution

The next motion vector is predicted by the Gaussian distribution centered at the previous motion vector (Figure 2.9). When searching for the next motion vector on the plane, we first sort the possible motion vectors by their distance from the previous motion vector. Those vectors near the previous motion are checked first because the motion vectors have higher possibility of remaining constant.

2.4.3 Evaluation

Speed is the key metric for evaluating real time processing, so we need to determine what level of real time performance can be achieved. Table 2.3 shows the speed for each layer processed. We can see that the time spent on motion estimation of each frame increases rapidly when motion estimation is conducted beyond layer three. The time complexity varies with the square of the image resolution. Since

Table 2.3: Speed of Perspective Correction (frames per second)

Layer calculated	1	2	3*	4	5
Time spent on estimation for each frame (ms)	19.5	38.3	81.3	714.3	7143

ego-motion estimation runs in real time its speed is critical. If we spend too much time on each frame, the frame rate falls below a given value (typically 10 frames per second), and image sequences captured may have little overlap, making the matching between two successive images impossible. Yet, when the resolution is too low (e.g. 3x3, 5x5 or 9x9), the estimated motion is no longer reliable because of the limited search space. When the estimated motion is not reliable, we observed that the cursor jumps on the screen. As shown in Table 2.3, Layer 3 is the ideal choice, and is stable for maintaining both frame rate and accuracy.

2.5 Feature Matching for Content based Image Retrieval

Content based image retrieval (CBIR[55, 56, 57]) may serve as the key to linking the physical world to massive online content. Using mobile CBIR [58], the retrieved objects (e.g. document) can be further annotated or altered on the device without changing the physical object. We pursue CBIR is one kernel technique, and feature matching between the query and the retrieved object is important for CBIR. For CBIR, feature extraction has been intensively addressed in the literature. Harris-Corners[59], SIFT[37] and SURF[42] are robust feature extraction methods. Although features such as SIFT perform well against geometrical distortion and

lighting change, how to match these extracted feature points was seldom addressed . A “bag-of-words” approach is often applied[60], but “bag-of-words” discards the geometrical relationship between feature points. It therefore is less accurate than the methods using geometrical relationship between feature points, especially when the number of feature points is limited. RANSAC[61] and Soft-Assign[62] are classic feature matching algorithms, RANSAC is a model based algorithm, and it assumes that all the inliers of matches must fit in the model. If the transformation between two point sets is not rigid, the RANSAC algorithm no longer works. The final model of RANSAC also depends on the initial selection of matched points, and it is possible to converge to a local minima. To allow a non-rigid transform, a more elastic method, such as soft assign, might be used, but soft assign offers an iterative method which could take a long time to converge.

We propose a triplet based point matching algorithm, which is robust against projective transforms, deformations and occlusions. Consider three points (A, B, C) on a 2D planar surface with homogeneous coordinates $(X_A, Y_A, 1)$, $(X_B, Y_B, 1)$, and $(X_C, Y_C, 1)$, their orientation is defined as

$$Sign\left(\begin{vmatrix} X_A & Y_A & 1 \\ X_B & Y_B & 1 \\ X_C & Y_C & 1 \end{vmatrix}\right) \quad (2.13)$$

where

$$\text{Sign}(X) = \begin{cases} 1 \cdots X \geq 0 \\ -1 \cdots X < 0 \end{cases}$$

When this surface is bent or viewed from another view angle, these three points appear as A', B' and C' and we have

$$\text{Sign} \begin{pmatrix} X_A & Y_A & 1 \\ X_B & Y_B & 1 \\ X_C & Y_C & 1 \end{pmatrix} \times \text{Sign} \begin{pmatrix} X'_A & Y'_A & 1 \\ X'_B & Y'_B & 1 \\ X'_C & Y'_C & 1 \end{pmatrix} = 1 \quad (2.14)$$

which means the orientation of (A, B, C) is consistent with (A', B', C') . On the contrary, (A, B, C) is inconsistent with (A', B', C') when

$$\text{Sign} \begin{pmatrix} X_A & Y_A & 1 \\ X_B & Y_B & 1 \\ X_C & Y_C & 1 \end{pmatrix} \times \text{Sign} \begin{pmatrix} X'_A & Y'_A & 1 \\ X'_B & Y'_B & 1 \\ X'_C & Y'_C & 1 \end{pmatrix} = -1 \quad (2.15)$$

When a point set S is matched to another point S' , we define the score of this match as

$$\sum_{A,B,C \in S} \left(\text{Sign} \begin{pmatrix} X_A & Y_A & 1 \\ X_B & Y_B & 1 \\ X_C & Y_C & 1 \end{pmatrix} \times \text{Sign} \begin{pmatrix} X'_A & Y'_A & 1 \\ X'_B & Y'_B & 1 \\ X'_C & Y'_C & 1 \end{pmatrix} \right) \quad (2.16)$$

An ideal match from a one point set to another n -point set has a score of C_n^3 when every triplet is consistent with its match. The worst match score is $-C_n^3$ (mirrored).

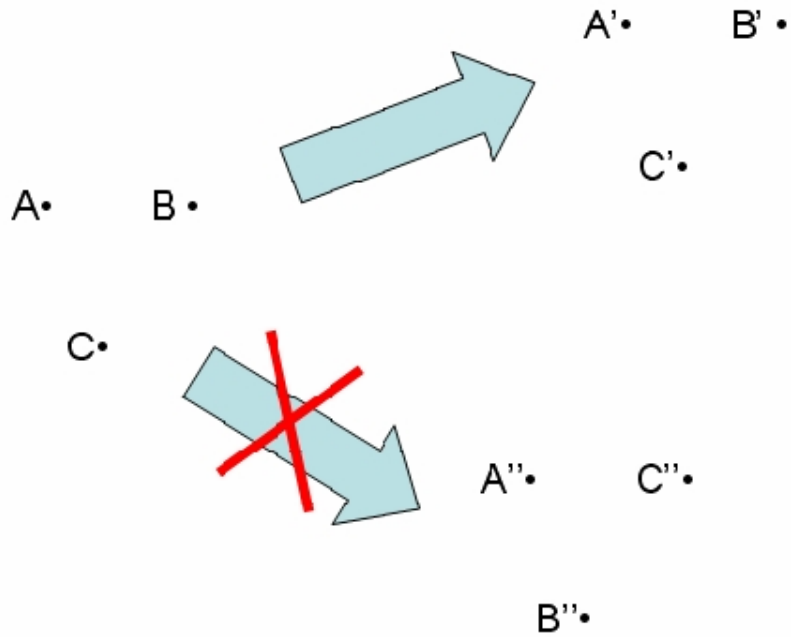


Figure 2.10: Possible triplet matches

To obtain the best match between two sets, a maximum flow or Hungarian algorithm can be used, but such an algorithm has a complexity of $O(v^3)$, where v is the number of vertices of the bi-partition graph (usually $V > 600$ for a single document page). We will apply this verification step to a list of candidate pages, meaning it consumes most of the runtime and must be efficient. We use a greedy algorithm to find an approximate match instead. Consider the two point sets as a bipartite graph and the value of each edge is the Euclidian distance between the layout contexts of its two vertices. We find the edge with the smallest value, match its two vertices, remove this edge together with its vertices, and repeat this procedure m times to find m pairs of point matches. The score of these m matches is between $-C_m^3$ and C_m^3 .

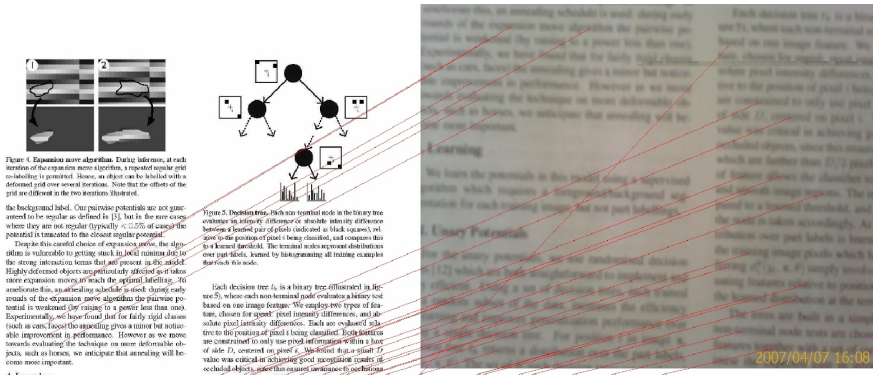


Figure 4: Expansion move algorithm. During inference, at each iteration of the expansion move algorithm, a repeated expansion grid is subdivided if possible. Hence, an object can be labelled with a different grid on several iterations. Note that the effect of the grid are different in the two iterations illustrated.

The background label. Our previous potentials are not guaranteed to be regular as defined in [1], but in our case even when they are not regular (typically ≤ 0.75) of cases the potential is truncated to the closest regular potential.

Despite this careful choice of expansion move, the algorithm is vulnerable to getting stuck in local minima due to the strong interaction terms that are present in the model. Highly detected objects are particularly difficult to get into more expressive moves and reach the optimal labeling. An alternative is an annealing schedule to reach a global minimum in a general case than one. Experimentally, we have found that for fully rigid classes (such as cars), faster annealing gives a minor but noticeable improvement in performance. However, as we move towards evaluating for behavior on more deformable objects, such as faces, we anticipate that annealing will become more important.

4. Learning

We learn the potentials in this model using a supervised algorithm which requires a foreground/background segmentation for each training image, for each label.

4.1. Unary Potential

For the unary potentials, we use the randomized decision trees [12] which go both straightforward to implement and very efficient. Using a set of decision trees, each trained on a random subset of the data, increases the efficiency of learning and improves generalization performance over using a single decision tree. For position x in image x , decision tree t , returns a distribution over the part labels, $\{l_i | p_i, v_i\}$. The set of t for each decision tree are constructed by simply averaging these distributions:

$$\phi_t(p_i, v_i) = \frac{1}{R} \sum_{t=1}^R \phi_t(p_i, v_i) \quad (5)$$

Figure 5: Dependency. This non-trivial node in the binary tree changes its activity depending on whether intensity difference between a limited pair of pixels (indicated in black squares) is large for positive (left) or being identical, and requires that its k -nearest neighbors. The learned unary inverse distribution over part labels, learned by backpropagating all training examples through this node.

Each decision tree t_i is a binary tree constructed in Figure 5, whose each non-terminal node contains a binary test based on one image feature. We employ two types of features, those that open "good intensity" differences, and absolute pixel intensity differences. Each are evaluated relative to the position of pixel being classified. Topologies are constrained to only use pixel and intensity within a box of size 11 , centered on pixel x . Note that at a point x , value was critical in detecting good neighborhood (e.g. occluded object) under the current variance t_i configuration, which is further away from each away, illustrating a process of locality allows the model to detect both image edges and smooth regions. The learned unary distribution is reported to a highest threshold, and the left or right branch of the tree is selected accordingly. At each segment t_i , a change over part labels is learned as the histogram of all the training image nodes which have reached that node during training. The unary potentials are constructed by combining features together with a set of candidate thresholds to maximize the expected gain in information. This process is halted when the best expected gain in information falls below a threshold. The motivation to learn the decision trees is dominated by feature definitions and less about in dependence of the number of labels. This will become more important as we move to more classes and labels in future.

0-7695-2446-2/06 \$20.00 © 2006 IEEE

(a) Successful retrieval, highest score = 1538

2. Cylindrical and spherical panoramas

Cylindrical panoramas are commonly used because of their ease of construction. To build a cylindrical panorama, a sequence of images is taken by a camera mounted on a leveled tripod. If the camera focal length or field of view is known, each perspective image can be warped into cylindrical coordinates. Figure 1 shows two overlapping cylindrical images, where the horizontal line becomes curved.

To build a cylindrical panorama, we map world coordinates (X, Y, Z) to 2D cylindrical screen coordinates (θ, r) using

$$\theta = \tan^{-1}(X/Z), \quad r = \sqrt{Y^2 + Z^2} \quad (1)$$

where θ is the pan angle and r is the radius [18]. Similarly, we can map world coordinates to 2D spherical coordinates (θ, ϕ) using

$$\theta = \tan^{-1}(X/Z), \quad \phi = \tan^{-1}(\sqrt{Y^2 + Z^2}/Y) \quad (2)$$

Once we have warped each input image, constructing the panoramic mosaic becomes a pure translation problem. Ideally, to build a cylindrical or spherical panorama from a horizontal pan sequence, only the unknown pan angle needs to be recovered. In practice, small vertical translations are needed to compensate for vertical jitter and optical axes. Therefore, both a horizontal translation t_x and a vertical translation t_y are estimated for each input image.

To recover the translational motion, we estimate the incremental translation $t_i = (t_x, t_y)$ by minimizing the intensity error between two images:

$$E(t) = \sum_i |I_i(x) - I_{i+1}(x+t)| \quad (3)$$

where $x_i = (x, y)$ and $x_{i+1} = (x', y')$ are corresponding points in the two images, and $A = (A_x, A_y)$ is the global translational motion field which varies for all pixels [2]. After a first order Taylor series expansion, the above equation becomes

$$E(t) \approx \sum_i |g_i(t_x + t_y)| \quad (4)$$

where $g_i = I_x(x_i) - I_y(x_i)$ is the current intensity of color error, and $g_i^T = \nabla I(x_i)$ is the image gradient of I at x_i . This minimization problem has a simple least-squares solution:

$$\left(\sum_i |g_i| \right) t = - \left(\sum_i |g_i| g_i^T \right) \quad (5)$$

Figure 1b shows a portion of a cylindrical panoramic mosaic built using this simple translational alignment technique. To avoid larger initial displacements, we use a hierarchical coarse-to-fine optimization scheme [2]. To reduce discontinuities in intensity and color between the images being composed, we apply a simple bilinear interpolation, i.e., we weight the pixels in each image proportionally to their distance to the edge (the more pixels, the closer to the nearest available pixel) [18]. Once registration is finished, we can clip the mask (and optionally the top and bottom), and write out a single panoramic image.

Creating panoramas in cylindrical or spherical coordinates has several limitations. First, it can only handle the simple case of pure pan/tilt motion. Second, even though it is possible to correct an image to 2D spherical or cylindrical coordinates for a known tilt/roll angle, resampling at north pole and south pole causes big registration errors. Third, it requires knowing the focal length of

equivalent field of view. While focal length can be readily calibrated in the lab [9, 16] containing the focal length of lens by registering two or more images is not very accurate, as we will discuss in section 3.

3. Perspective (8-parameter) panoramas

To overcome these limitations, several authors have suggested using full affine perspective instead of cylinders [12, 9, 18]. The fixed perspective transformation warps an image into another using 8 parameters:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (6)$$

where $x = (x, y, 1)$ and $x' = (x', y', 1)$ are 2D components of two perspective coordinates, with m indicating equality up to scale. This equation can be re-written as

$$x' = \frac{m_{11}x + m_{12}y + m_{13}}{m_{21}x + m_{22}y + m_{23}} \quad (7)$$

$$y' = \frac{m_{21}x + m_{22}y + m_{23}}{m_{21}x + m_{22}y + m_{23}} \quad (8)$$

In projective geometry, only the two parameters m_{11} and m_{21} are arbitrary.

To recover the 8 parameters, we iteratively update the transform matrix using

$$M = (E + D)M \quad (9)$$

where

$$D = \begin{bmatrix} d_1 & d_2 & d_3 \\ d_4 & d_5 & d_6 \\ d_7 & d_8 & d_9 \end{bmatrix} \quad (10)$$

Recomposing image I_1 with the new transformation $x' = (E + D)x$ is the same as warping the resampled image $I_1(x)$ to $I_1(x') = I_1((E + D)x)$:

$$x' = \frac{(1+d_1)x + d_2y + d_3}{d_4x + (1+d_5)y + d_6} \quad (11)$$

$$y' = \frac{d_7x + (1+d_8)y + d_9}{d_4x + (1+d_5)y + d_6} \quad (12)$$

Again, we wish to minimize

$$E(d) = \sum_i |I_1(x'_i) - I_2(x'_i)| \quad (13)$$

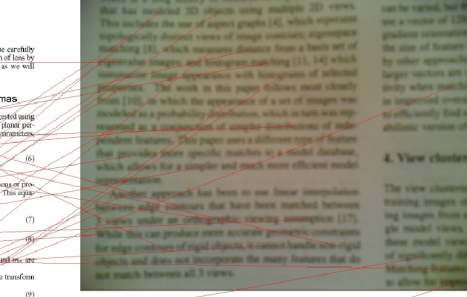
$$= \sum_i |I_1^2(x'_i) - I_2^2(x'_i)| \quad (14)$$

where $d_i = (d_1, \dots, d_9)$ is the incremental update parameter, and $J_d = J_d(x_i)$ where

$$J_d(x) = \frac{\partial x'}{\partial d} = \begin{bmatrix} x & y & 1 & 0 & 0 & -x^2 & -xy \\ 0 & 0 & 0 & x & y & 1 & -xy & -y^2 \end{bmatrix} \quad (15)$$

is the Jacobian of the resampled pair coordinate x' with respect to d . The matrix in the Jacobian correspond to the optical flow induced by the instantaneous motion of a plane in 3D [2]. The least-squares minimization problem (14) can be solved using normal equations analogous to (5).

$$Ad = -b, \quad (16)$$



(b) Failed to retrieve, highest score = 332

Figure 2.11: Example matches with point correspondences

m=30

2.6 Summary

We have introduced basic techniques in this chapter that are useful for pursuing mobile applications that involve vision and image processing. In the following chapters, we will discuss symbology recognition, content based image retrieval, and object recognition that we identify as major computer vision applications on mobile devices. In addition to the theoretical and practical issues of these novel systems and applications, the techniques introduced in this chapter will also be referred to and evaluated throughout the thesis.

Chapter 3

VCode - Imaging as an Alternative Mobile Data Channel

3.1 Overview

We introduce a novel usage of the camera as a visual communication channel, which is especially appropriate for mobile applications. Data is encoded as a series of machine readable dynamic frames, captured by a mobile camera, and reconstructed on the device. We address the problem of information uncertainty in the camera channel because errors may exist within a given frame and frames can be missed during detection. We introduce a novel spatial-temporal error correction method to encapsulate data being sent through this highly vulnerable channel with both error and data loss. To maximize data capacity per pixel, we use color to represent bits so each pixel can carry multiple bits of information. We introduce a color calibration pattern to adapt for different cameras and displays and simultaneously learn the channel property when decoding the captured frames. As a theoretical contribution, we estimate the capacity of the camera channel using mutual information theory and demonstrate that a proper selection of colors for encoding can optimize throughput.

3.1.1 Motivation and Related Work

Cameras on mobile phones have become an integral part of the devices. One of the most popular “camera analysis” applications is to use camera phones to scan

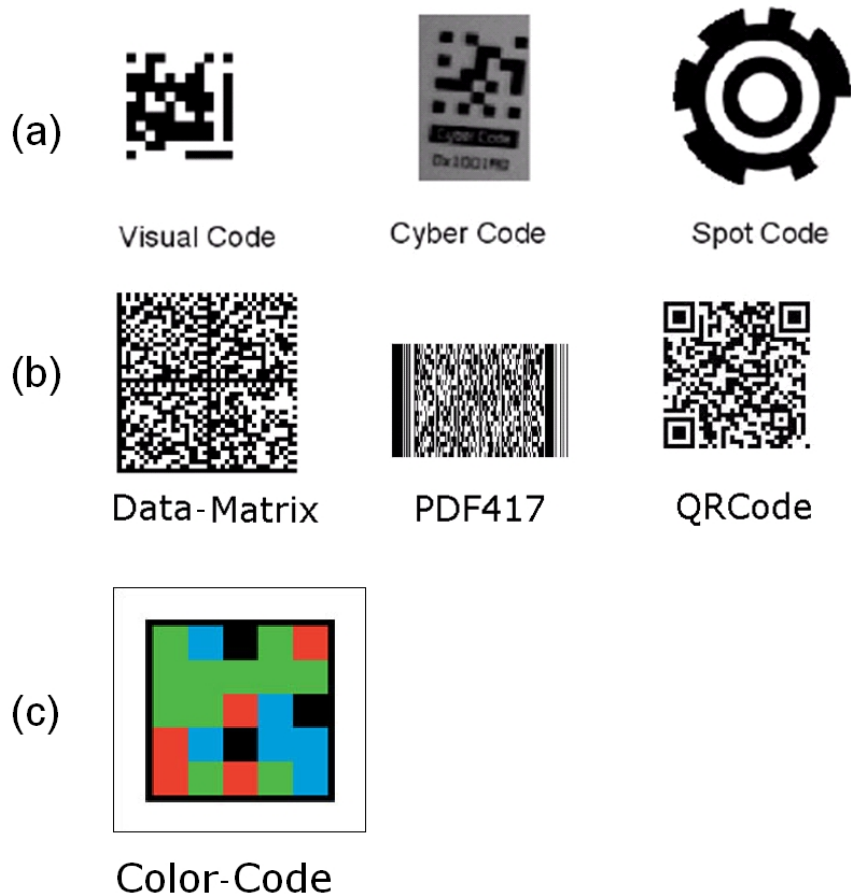


Figure 3.1: Various 2D barcodes (a) Interaction oriented 2D barcodes (b) Content oriented 2D barcodes: QRCode, DataMatrix and PDF-417 (c) ColorZip’s ColorCode 2D barcodes (Figure 3.1) printed on products, papers, advertisements, and business cards [63, 64, 65, 66, 67, 68]. This capability provides users with one click access to online information by decoding various meta-data (i.e., product code, URLs, etc.) instead of typing.

2D barcodes can be functionally categorized as: interaction oriented or content oriented. Interaction oriented 2D barcodes are often used in context aware mobile

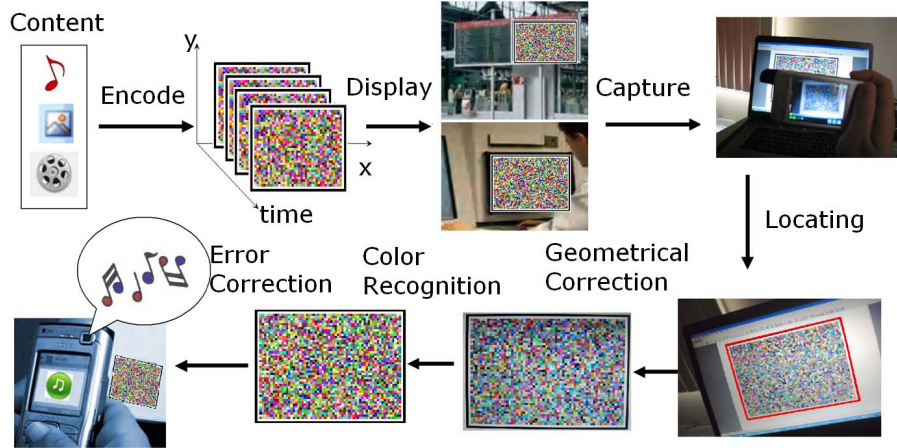


Figure 3.2: Overview of the camera channel

applications, and they only encode a few bytes of information which is either used as an index to online content or reflects context information(Figure3.1 (a)). A “Visual Code” proposed by Rohs [69] stores as many as 83 bits data and can track the motion of the capturing device to facilitate mobile interaction [70, 71]. Scott, et al. used “Spot Codes” for out-of-band Bluetooth device discovery, which bypasses the standard Bluetooth protocol negotiation [72]. Other novel applications allow the mobile device to interact with a display [73, 74].

Content oriented 2D barcodes, on the other hand, are used to deliver rich meta-data to mobile devices. The encoded content could be a URL or contact information. QRCode [75] is one of the most popular content oriented barcodes and is widely used in Japan’s consumer market. Data Matrix and PDF417 are found widely in express mail delivery and identification cards. Limited by its footprint, however, even content oriented 2D barcodes typically hold no more than 200 bytes. Various multimedia data, such as ring tones, theme pictures and public keys, cannot

be delivered to the device because of this limitation.

To break this limitation two categories of approaches has been developed to extend the information capacity of 2D barcodes:

- Using colors allows each pixel to carry more information. This technology has been referred to as “ColorCode” (Figure 3.1(b)) and used in [76, 77].
- Our technology, a dynamic multi-frame 2D barcode which we refer to as V-Code [78, 79, 80]. Theoretically, V-Code can carry an unlimited amount of information as the number of frames increases.

V-Code can allow a camera to operate as an alternative data channel. The basic concept of the camera channel is illustrated in Figure 3.2. The data is encoded into a sequence of images, which are animated on a flat panel display, acquired by the camera, and decoded by software on the device. The camera channel is free, passive, pervasive, and especially attractive to advertisers because it can be integrated into the physical world. Flat panel displays have been installed widely in public areas, so the camera channel has a serious potential for delivering multimedia content (ring tones, pictures, Java games, etc.) to mobile users.

The development of V-Code is motivated by three primary factors. First, the current mobile data access is still sub-optimal. GPRS/CDMA data plans are costly and may not work indoors or underground due to poor signal coverage. Bluetooth and Infrared require extra hardware (adapter) and driver installation. Wireless infrastructures such as WiFi are insecure because the device cannot receive information without emitting information (e.g. MAC address). Potential risks of

broadcasting a MAC address include sniffer and man-in-the-middle attacks. Second, cameras have been widely integrated onto these mobile devices and show promise for visual communications. Their inclusion as part of existing hardware makes them especially attractive. Third, LCD displays have been widely installed in public areas such as airports, train stations and even shuttle buses. Using V-Code, these LCD displays can not only distribute visual information through traditional advertising, but can also deliver digitalized information to the mobile devices.

3.2 Binary V-Code

During the initial stage of V-Code research we designed our own symbology, as shown in Figure 3.3a, to maximize the data capacity of a black and white matrix pattern. Sensors in camera phones are often not square, we designed a rectangle frame for VCode which then had an aspect ratio similar to the captured image and to maximize throughput. As shown in Figure 3.3a, the code area consists of two parts, a rectangular bounding box defining the code's boundary and a data area. The boundary is used as the detection pattern and can be efficiently detected using a fast Hough[81] transform method. The data area consists of black and white cells, each carrying one bit of data with black representing 1 and white representing 0, which is called the binary V-Code. Because the data capacity of binary V-Code is 1 bit /cell, the bit rate achieved in [79] was $\leq 3.3\text{k bps}$ (bits per second).

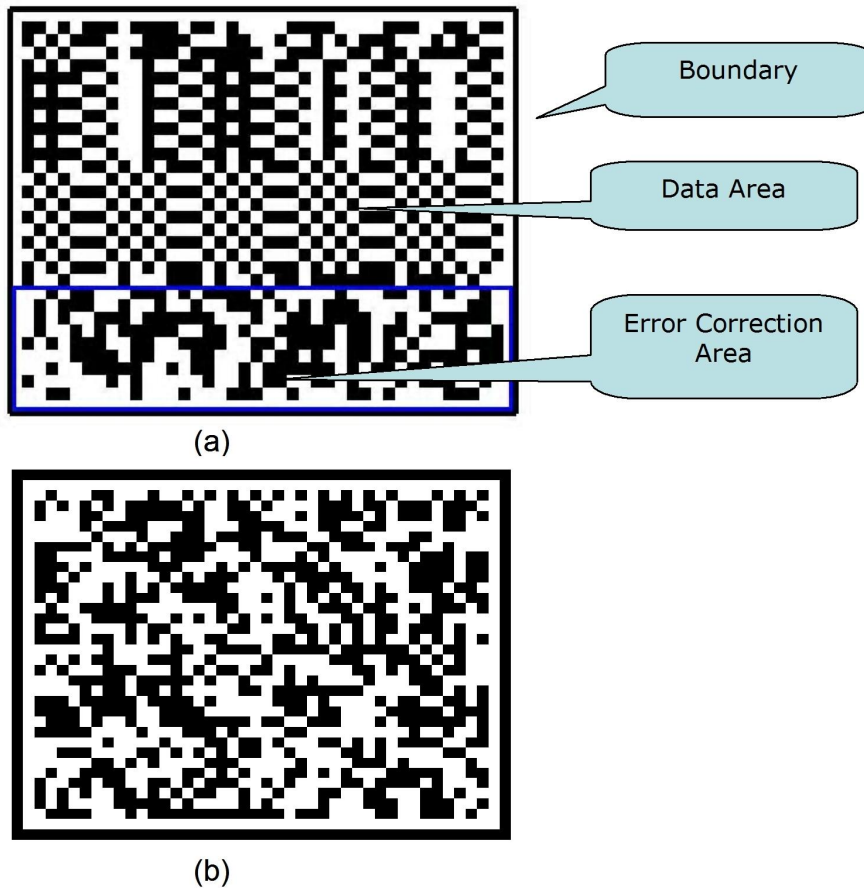


Figure 3.3: (a) A VCode Frame. (b) An example of the mask used to “encrypt” VCode.

3.2.1 Binary VCode Encoding

To encode a data file into a VCode, we first separate the data file into small segments, then encode each segment into an image sequence. While the scheme seems straightforward, the challenge comes in making the encoding robust to the degradation and data loss that are inevitable in the imaging process. The cameras on phones often produce much lower quality images than digital cameras, and we expect users to capture VCode in real environment without constraints in lighting

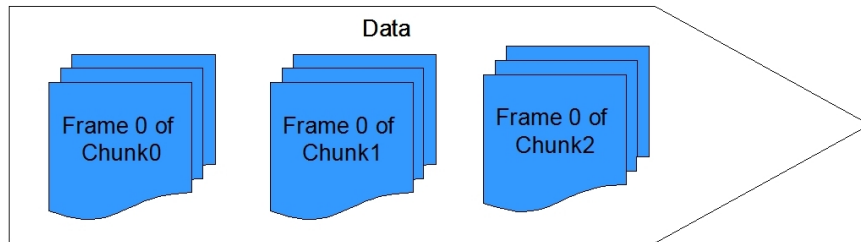
and perspective angles. Our strategy uses state of the art error control in both time and space to make code more robust against these types of degradations.

3.2.1.1 Data Partitioning and Error Correction

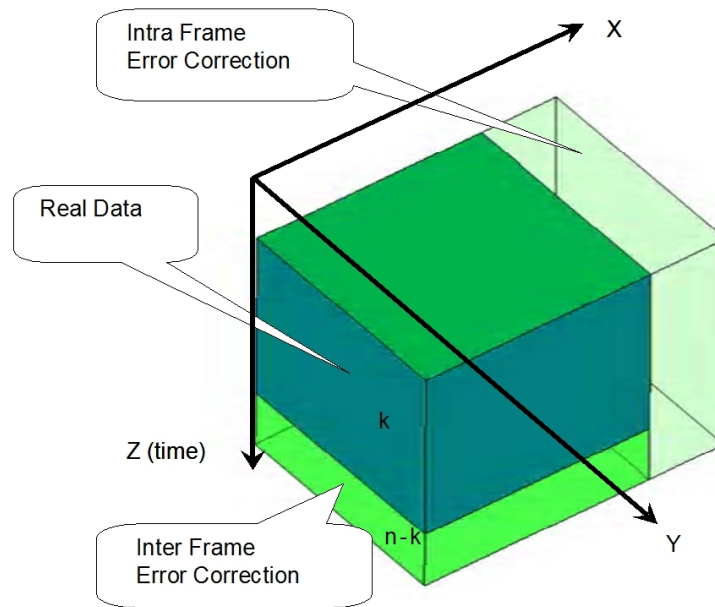
The data being encoded is partitioned in a way that both intra- and inter- error correction bits can be easily inserted. We divide the data into multiple chunks, each of which is divided further into individual frames. This forms a three-layer structure of the data representation, as shown in Figure 3.4.

Figure 3.4b shows the error correction scheme we propose in each chunk. Each data chunk in Figure 3.4a can be visualized as a “Cube,” which consists of three areas: the data area, inter- and intra- frame error correction areas. The data file to be encoded is filled into this “Data Cube” (Figure 3.4b). Therefore we can assign a three-dimensional coordinate to each bit. Specifically, our error correction encoding scheme is described as:

1. Partition data: Split the data into chunks, each of which has the dimension $K \times W \times H$, where K is the frame number, W and H are the width and height for each frame.
2. Correct inter-frame errors: Scan each column along the Z (time) axis of the data cube and add error correction bytes for each column scanned. Since we have K data frames in the “Data Cube,” we add $(N - K)$ frames at the end of each chunk as inter-frame error correction frames. We then can use a (N, K) -Reed-Solomon code [82] to encode each chunk into a $N \times W \times H$ cube. These



(a)



(b)

Figure 3.4: Data Partition. (a) Segment a data file into chunks and frames, and (b) The error correction scheme.

redundancy frames will be dropped if they are not needed.

3. Correct intra frame errors: We add error correction code by padding extra bits to each frame on the $x - y$ plane. Each frame is extended from size $W \times H$ to $W \times (H + R)$.

Each frame consists of three parts: the frame header, the data area, and the error correction area. The frame header contains the frame index, chunk index, the total number of chunks, and a checksum. The frame and chunk indexes provide the position of each frame so it can be placed into the right position after decoding. The checksum is used to check whether the decoded frame and chunk indices are correct. If they are incorrect, the whole frame will be dropped and recovered later by error correction frames. The number of chunks is uniform on all frames and can be used to check if the file is downloaded completely. We encode this information in every frame so users can begin capturing from any frame (the VCode will be displayed in a loop until all data frames are correctly captured and decoded).

3.2.2 Binary VCode Rendering

The rendering converts each frame (including error correction frames) into an image, which can be displayed on flat screens. Rather than using existing 2D barcode symbologies such as QR codes or Data Matrix (which are inherently static), we designed our own symbology, as shown in Figure 3.3a, to maximize the data capacity. Given that the sensors in camera phones are often not square, our design for the frame of a VCode is a rectangle to have a similar aspect ratio to the captured

image. As shown in Figure 3.3a, the code area consists of two parts: a rectangle bounding box defining the boundary of the code and a data area. The boundary can be used as the detection pattern and can be efficiently detected using a new fast Hough transform method. The data area consists of black and white cells, each carrying one bit of data with black representing 1 and white representing 0.

Before a frame is rendered, we use a mask to **xor** each frame. The mask provides encryption to the data because decoding is almost impossible without pre-knowledge of the mask. This allows the data to be downloaded only by users who have the “passcode”. A typical mask is the checker board pattern, as shown in Figure 3.3b.

3.2.3 Binary VCode Acquisition

The device constrains the acquisition size and frame rate. The process, however, must optimize throughput by trading acquisition speed, image resolution, and processing requirements. Ideally, we would choose the highest resolution that remains robust to degradation, yet can be processed at frame rates. Although camera phones often allow users to capture images with different resolutions, from 160×120 to 1600×1200 (2M pixels), our experiments suggest that QVGA resolution is a balance between speed and image quality for current mid-level devices. The acquisition process itself is very simple: Users need only to aim the camera at the VCode to keep the frames at the center of the display. Detection and decoding will occur at frame rate.

3.2.4 Experiments of Binary VCode

One of the direct applications of VCodes is to downloading data through visual communication. Two factors are important from the user's point of view, the data transmission speed and robustness. Our experiments evaluate the performance of these two factors.

3.2.4.1 Data Transmission Speed

The factors directly affecting the data transmission speed are (1) the amount of data encoded in a frame, and (2) the frame rate at which the VCode is displayed and subsequently decoded. Assume the displayed frame rate is P frames/second and D bits are encoded in each frame, then, theoretically, the overall bit rate is $P \times D$ bits per second (bps). Therefore the increase of P and/or D will lead to higher bit rates, but practically, it is much more complex. For example, if more bits are encoded in a frame (increasing D), it will increase the barcode density and decrease the resolution of a single cell unit when the image is captured, possibly leading to more decoding errors. If the frames are displayed too quickly (increasing P), the device may not be fast enough to capture and process them, resulting in missed frames. The experiments we conduct in the following sections result in a quantitative analysis of these factors.

3.2.4.2 Data Capacity in a Single Frame

Currently main stream camera phones can capture a video sequence with a resolution of 320×240 pixels. Although a captured still image may have a Mega- or multi-Mega-pixel resolution, a camera phone needs to capture and process frames continuously. Therefore a video mode is required, which limits D . Although the next generation camera phones may capture HDTV quality video, our analysis is based on the majority of currently available devices.

Like other 2D barcodes, the resolution (the number of pixels) of a unit cell, defined as a black or white square representing one bit information (either 1 or 0), is crucial for decoding. Given the restriction of the frame size (320×240), increasing the number of bits will decrease the resolution of a unit cell in captured images, leading to higher erroneous bits, and correspondingly, more extra bits being required to correct those erroneous ones. As we discussed above, the total number of bits in a frame (N) consists of the data part (D) and the error correction part (E). The actual data $D = N - E$. It is important to find a balance between N and E to achieve the optimal result. To investigate this problem we performed a simulation by generating an all-zero data file and encoding it as a VCode with four different settings of unit cells: 28×35 , 32×40 , 40×50 and, 48×60 . The reason we select an all-zero data file is that zero remains the same after xor operation with the mask defined in Figure 3.3b ($1 \text{ xor } 0 = 1$, $0 \text{ xor } 0 = 0$). After applying the mask, the image looks exactly the same as the mask defined in Figure 3.3b. When the displayed images are captured and decoded, any 1 in the result indicates an erroneous bit.

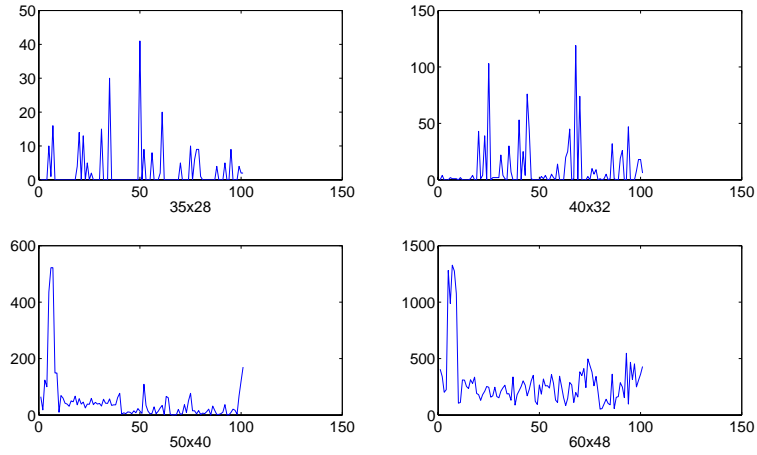


Figure 3.5: The number of erroneous bits over 100 frames for four settings. (a) 28×35 , (b) 32×40 , (c) 40×50 and (d) 48×60

We also use an all-zero data file to eliminate the effect of frame transition (ghost image), which will be discussed in the next section.

Figure 3.5 shows the number of erroneous bits over 100 frames under four different settings. As expected, the larger the value of N , the more erroneous bits that are generated, and the more error correction bytes E are required to correct them. To predict the actual performance of these four settings, we define the “Equivalent Bit Rate” **EBR** as a metric. For F consecutive frames in a VCode, **EBR** is defined as

$$EBR = \frac{TB}{F \times T} \quad (3.1)$$

where TB is the total number of bits that we can decode from F frames, and T is the time spent on decoding a frame. $F = 100$ in this experiments and T depends on the number of unit cells. Given the complexity of sampling N points from an

image and of decoding N -bits data is $\Theta(N)$, we have $T \sim N$:

$$EBR \sim \frac{TB}{F \times N} \quad (3.2)$$

Let $Err(i)$ be the number of erroneous bits on the i_{th} frame and $Data(i)$ be the number of bits we read from the i_{th} frame, which could be either 0 or $N - 8E$, depending on $Err(i)$. If the number of erroneous bits in a frame is too large, the remaining bits will not then be enough to correct them. More specifically, we have:

$$Data(i) = \begin{cases} 0 & \dots Err(i) > E/2 \\ N - 8E & \dots Err(i) \leq E/2 \end{cases} \quad (3.3)$$

Substituting (3.3) into (3.2), we have:

$$EBR \sim \frac{\sum_{Err(i) \leq E/2} (N - 8E)}{F \times N} \quad (3.4)$$

where $i \in 1..F$, as shown in Figure 3.5. For a fixed number of unit cells, the only factor that affects **EBR** is E , the number of error correction bytes. E could be neither too small nor too large. When E is too small, most of the frames with erroneous bytes greater than $E/2$ will be dropped. When E is too large, however, the error correction code will dominate the frame and little data is encoded. Therefore, the purpose of this experiment is to find an optimal E which maximizes the bit rate.

Figure 3.6 shows results that illustrate relations between **EBR** and E for four settings (28×35 , 32×40 , 40×50 and 48×60), respectively. We can see that the largest **EBR** value is located on the red curve, with setting 32×40 and $E \approx 16$. The **EBR** value in the blue curve (setting 28×35) is lower because less

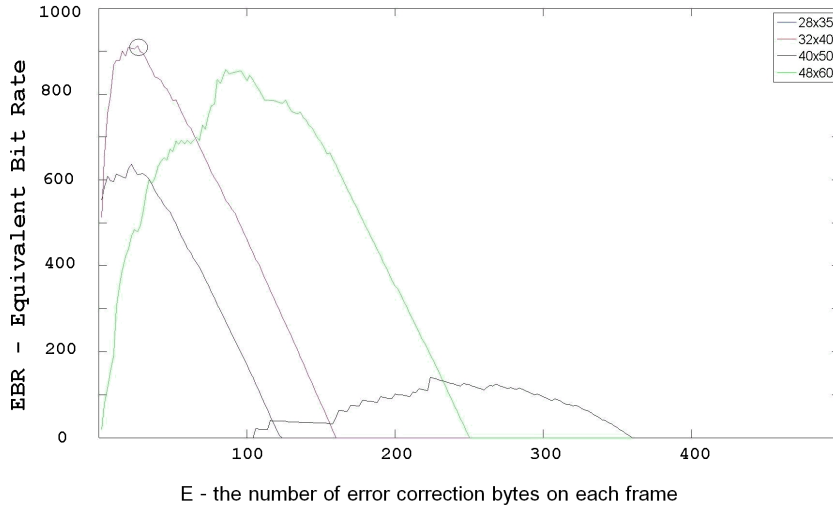


Figure 3.6: The relationship between E and **EBR**.

information is carried in each frame. On the other hand, the highest N (setting 48×60 , corresponding to black curve) actually has very low **EBR** values due to the large number of erroneous bits. Furthermore, it takes longer to decode a higher resolution frame. Our experiments show that the optimal setting is achieved when the number of unit cells is 32×40 with 16 bytes for error correction.

3.2.4.3 Display Frame Rate

The display frame rate generally depends on the speed with which a frame can be captured and processed by camera phones, which is device dependent. A frame cannot be displayed too quickly as camera phones need to have enough time to perform geometric correction, decoding, and error correction. If what is displayed too slowly, however, the camera phone will have to process the same frame repeatedly. Although the duplicate data will be identified and removed, re-decoding decreases

Table 3.1: Throughput (bits/second) vs. Frame Rate (Frames/second)

Frame Rate	20	10	6.6	4
User 1	360	2184	2340	1365
User 2	352	2730	3276	1260
User 3	352	1928	2520	1638
Average	355	2280	2712	1421

Frame Rate (fps)	Average Bit Rate (bps)
20	355
10	2280
6.6	2712
4	1421

the overall bit rate. The ideal situation occurs when camera phones process every frame exactly once. If a frame is dropped, it can be recovered by error correction or be re-captured in the following round because the VCode is displayed in a loop. We tested four different display frame rates using a NOKIA 6680 camera phone as the capture device. The data file selected was a 4KB MIDI ring tone encoded as a VCode containing 60 frames. The VCode was displayed at frame rate of 20, 10, 6.6, 4 frames/second respectively on a 15-inch flat panel computer monitor. For each frame rate we let three users download the file onto the camera phone. The time t used for download is recorded for each run, and the throughput is calculated as $4096 \times 8/t$ bps. The overall results are shown in Table 3.1.

From Table 3.1, we see that when the animation frame rate is very high (20fps) or very low (4fps), the downloading bit rate is low. The optimal result is achieved when the animation frame rate is between 6.6 to 10 fps. To explain these results, we recorded the total number of dropped frames in each run in Table 3.2.

Table 3.2: The number of dropped frame vs. frame rate (frames/second)

Frame Rate	20	10	6.6	4
User 1	622	63	50	130
User 2	646	45	30	145
User 3	675	83	49	100
Average	648	64	43	125

Frame Rate (fps)	Frame Drop (Frames)
20	648
10	64
6.6	43
4	125

From Table 3.2, we see that when the frame rate is high (20fps), the number of dropped frames (over 600) is much higher than that of other settings when finished with the final download. VCode contains only 60 frames, so a large number of dropped frames indicates the VCode has been displayed in a loop for several times before completing the download. There are two reasons for dropping frames: First, the camera phone cannot process a frame within 1/20sec. Second, when frames are displayed quickly, ghost images appear due to the camera’s “visual short term

memory”. When black and white cells flip quickly, they appear as a gray color rather than black or white. When the frame rate is low (5fps), the frame drop rate is also high because the decoding keeps receiving duplicate frames. Therefore, a frame rate between 6.6 and 10 presents a good choice for the device used in this experiment.

3.2.4.4 Overall Downloading Bit Rate

After analyzing specific factors affecting the download speed, we evaluate the overall throughput in a more comprehensive data set. We selected three data files, including a MIDI ring tone, a Java game, and a 3GP video, as our test set. Table 3.3 lists the sizes of these files. We allowed the same three users download these files and recorded the time spent on downloading when the final download is complete. The bit rate is defined as the quotient of a file size over the time spent on downloading. The average bit rates for downloading are also shown in Table 3.3, demonstrating that the bit rate decreases as the file size increases. For comparison, we put the phone on a dock on a desk so both the phone and monitor are static, a configuration we call “dock” mode. In dock mode, the download bit rate is stable, independent of the file size, since no users’ factors are involved and the bit rate is higher (around 3.3 Kbps) than in hand-held mode.

3.3 Color VCode

Naturally the addition of color to the VCode has the potential to increase the VCode’s bit capacity significantly by representing more than one bit per pixel. To

maximize VCode’s capacity, it is desirable to represent as many bits as possible per pixel. This leads to a fundamental question of image information theory, how many bits can be read from a color pixel, subject to camera dependent distortion and environmental noise. We will try to answer this question using mutual information theory. Moreover, we will show that the capacity of the camera channel can be maximized using a proper selection of colors. Despite these theoretical challenges, we will also discuss practical issues, such as how to locate the V-Code accurately in real time and how to guarantee data integrity when a frame drop occurs. Finally, the overall bit rate of the colored V-Code will be tested by downloading multimedia content to the camera phone via the camera channel.

3.3.1 Analysis

The throughput, defined by the bit rate, is a critical measure of the capacity of the data channel. In this section we focus on finding this data channel’s theoretical limit and discussing how we can reach this theoretical limit in practical applications. The issues we need to address include the factors affecting the bit rate, and the

Table 3.3: Comprehensive downloading bit rate test

Media Type	File Size	Hand-held	Dock
Ring tone	4KB	2.67Kbps	3.2Kbps
Game	40KB	2.06Kbps	3.3Kbps
3GP Video	57KB	1.18Kbps	3.3Kbps

camera channel’s throughput and how to maximize it. We start by analyzing the binary V-Code described in Section 3.2 and find the following shortcomings that limit throughput:

1. Only black and white patterns are used, therefore the information capacity at the pixel level was not fully explored.
2. Registration is based on line detection, which may not have enough accuracy for dense codes and may yield a large number of aliases.

In the remainder of this section, we explore how to enhance the camera channel’s throughput by addressing pixel capacity, pattern density, and dropped frames. Using colors to encode bit information, we estimate the pixel capacity by mutual information and show that using a proper selection of distinct colors can maximize the data capacity. To cope with various displays and cameras, a color calibration pattern is frequently displayed for synchronization and the color models are built as the downloading proceeds. We present a pixel level registration and demonstrate its ability to reduce the overall error rate. A temporal error correction code is also used to recover the dropped frames and increase the bit rate. Other methods to enhance the throughput, such as increasing the frame rate (subject to hardware limitation) and data compression, are also suggested.

3.4 Channel Capacity and Color Selection

Research on how images carry information dates to the beginning age of information theory [83]. Sheikh and Bovik [84] use the amount of information carried in

an image to estimate the quality of the image. Digital watermarking is a joint area of research in the fields of information theory and image processing, and Barni, et al. derive an upper bound of information capacity for DCT embedded watermarking [85]. Nakamura, et al.[86] embed watermarks in video and decode them using a camera phone. The general purpose of digital watermarking, however, is to embed a small amount of information without disturbing the visual content. There is little research on transporting information and analyzing its capacity through visual channels.

With the binary V-Code[79], the decoder samples one pixel from the camera-captured image and relays one bit of information by classifying this pixel into black or white. Theoretically, using n colors, we may increase the pixel capacity to $\log_2 n$ bits. Display and imaging systems typically use 24 bits(RGB), so, in theory, we can extract 24-bit information from one pixel of captured image. Unfortunately, noise and degradation of the camera channel causes information loss. Nevertheless, we are still interested in analyzing the capacity per color pixel.

The channel capacity is formally defined in information theory as the maximum mutual information among all distributions of input symbols:

$$I(X;Y) = \sum_{j=0}^{q-1} \sum_{i=1}^{Q-1} P(x_j)P(y_i|x_j) \log \frac{P(y_i|x_j)}{P(y_i)} \quad (3.5)$$

where q is the size of the input symbol set, Q is size of the output symbol set, and in our case, symbol=color. $P(x_j)$ is the probability of color x_j being displayed and $P(y_i)$ is the probability that color y_i is captured by the camera. $P(y_i|x_j)$

is the conditional possibility that color x_j is captured as color y_i . By arranging $P(y_i|x_j)$ in a matrix we get a transition matrix. Mobile contents (pictures, ring tones, applications) are typically compressed before being delivered to the device, so we assume the data being sent via the camera channel is a random bit stream and that every input symbol has equal opportunity to appear. We have $P(x_j) = 1/q$ and $P(y_i) = \sum_{j=0}^{q-1} \frac{P(y_i|x_j)}{q}$ and hence the capacity of the camera channel is fully determined by the transition matrix. In other words, given any input color, if we know the distribution of its captured color, we will be able to determine the channel capacity. Intuitively using a finite set of colors to represent information, we prefer the difference between the captured colors to be large and the variance of the distribution to be small, so the two colors have a lower chance of being confused.

3.4.1 Two-color Channels

To study how the transition matrix affects the channel capacity, we start from the simplest binary channel consisting of two colors, black(0) and white(255). Assuming the captured values rest between 0 and 255 and are normally distributed around means $128 \pm d/2$ with variance v , the capacity of this two-color channel can be determined using the mutual information defined by (3.5). We have the function $b = Capacity(d, v), 0 \leq b \leq 1$ with $d = 1..255, v = 1..100$. Figure 3.7 shows how capacity changes with d and v .

As we can see, the channel capacity approaches 1 as d approaches 255 and v approaches 0. This result shows that higher uncertainty (variation) leads to lower

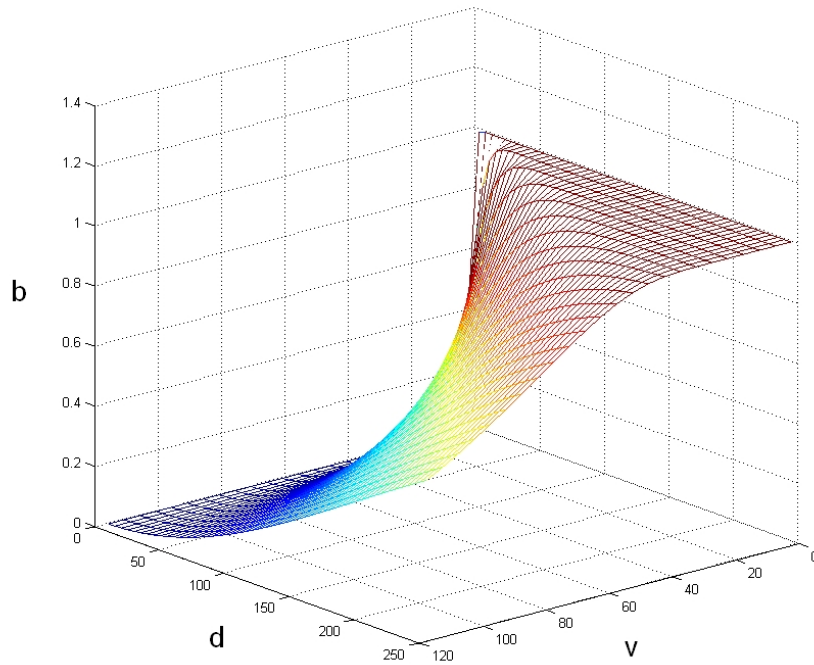


Figure 3.7: Simulation of binary color channel capacity

information capacity, and a steeply peaked distribution leads to larger information capacity. Therefore, we should use a greater distance between colors and a smaller variance to achieve higher capacity.

3.4.2 Multi-color Channels

To study how colors are captured by the camera and measure the information loss, we ran a simulation by capturing colors multiple times. Two thousand random colors are uniformly generated in the 24-bit RGB color space, and each of the sampled colors is embedded into a small 2D barcode (Figure 3.8a) at the bottom-left corner. This small 2D barcode encodes 16 bits (short integer) of information with 16

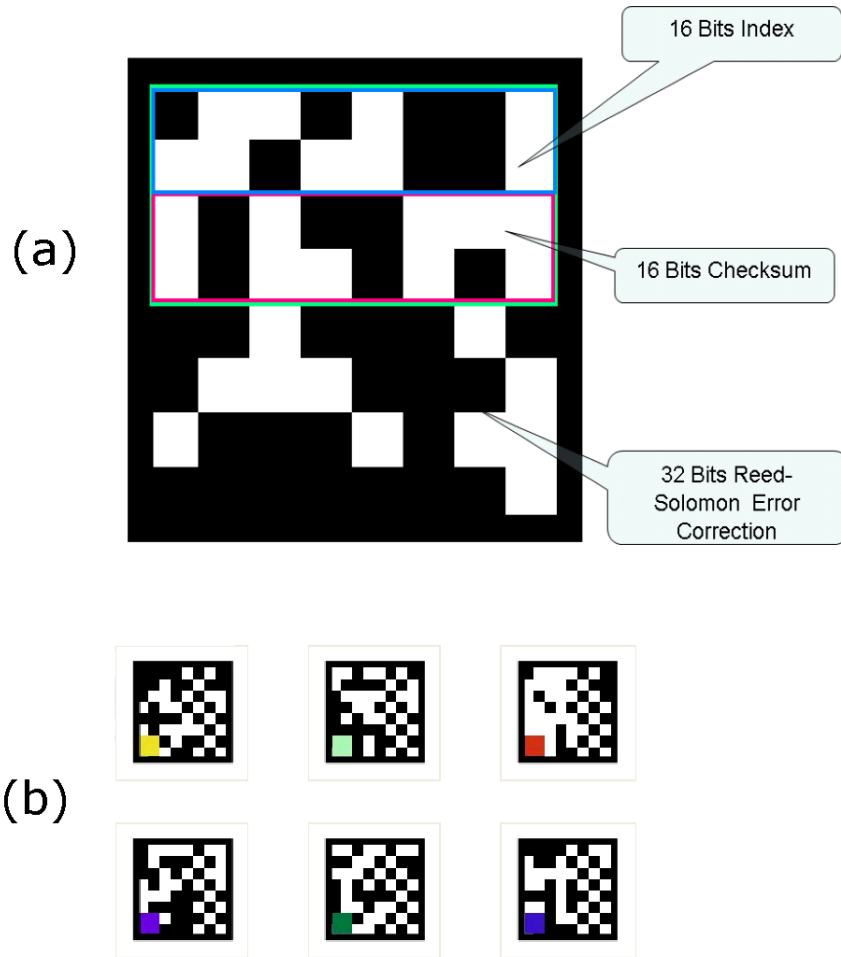


Figure 3.8: Small barcode we used for simulation

bits of checksum data and 32 bits of Reed-Solomon[82] error correction data. We are able to correct 2 bytes (16 bits) of error by using 32 bits(4 bytes) of Reed-Solomon error correction. The encoded short integer is the index to a row of the table that stores the 2000 random colors. We allocate 4-unit cells to embed the color (Figure 3.8b) to avoid the correlation between neighboring black/white pixels. The logged color is sampled from the center pixel of the colored block. Each of these colors is captured 200 times from the same LCD display under the same lighting conditions.

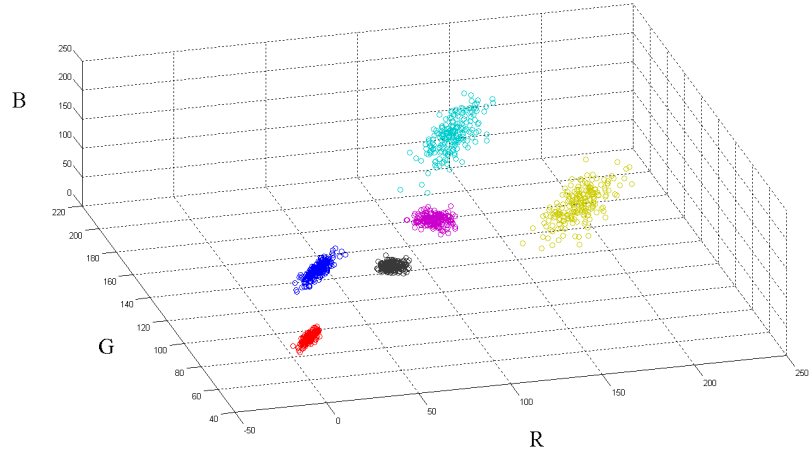


Figure 3.9: Distribution of captured colors, six samples

Figure 3.9 shows the distribution of six sampled colors.

From Figure 3.9, we can see that one color may vary in the camera captured image, but the distribution is, in general, concentrated around the mean with some variance. Assuming Gaussian noise of the color degradation, we model this captured color by a 3D Gaussian distribution. To justify the Gaussian model for modeling the captured color data, we first de-correlate the RGB channels of the 2000 sampled colors using PCA and obtain 6000 1D distributions. We run these 1D distributions through the Jarque-Bera test [87] and 5456 (91%) of them could not be rejected at the 3% significance level. We compute the mean vector and covariance matrix (μ_j, σ_j) for $j = 1..2000$ using maximum likelihood estimation and obtained the probability density function for these 2000 sampled colors. With k colors chosen from these 2000 colors, we quantize the captured color space into $64 \times 64 \times 64 = 262144$ cubes and generate a $k \times 262144$ transition matrix, from which the channel capacity was

computed.

In communication theory, frequency selection for input symbols is a classic problem and usually solved by numerical optimization assuming certain distortion of the channel. But the camera channel does not have a fixed parameter of distortion. For example, some CMOS sensors are insensitive to variation in red, while others are insensitive to variation in green. In addition to the variation caused by one camera, the variation between cameras presents another factor that cannot be neglected. To examine how color varies among cameras, we first define function $P_{cam}(c)$ as the mean value of color c among 200 images obtained using cam , where $cam \in \{\text{N6600, N6680, N7610, N93, E50, PPC6700, iMate Jamin, Qtek S200}\}$. Table 3.4 shows the variation of five colors obtained by these eight cameras:

Table 3.4: Variation of $P_{cam}(c)$ among eight cameras

Color c	σ_R	σ_G	σ_B
Black 000000	5.5	6.1	3.4
Red FF0000	38.4	21.1	14.9
Green 00FF00	18.7	41.8	22.4
Blue 0000FF	12.1	26.6	42.5
White FFFFFFFF	39.8	42.7	35.5

The selected k colors should be as different as possible so that they can be separated from each other by the decoder. We use the 2-norm to measure the difference between two colors. Without loss of generality, the distance between

colors c_1 and c_2 is chosen as the nearest distance between their captured values across the eight different cameras.

$$dist(c_1, c_2) = \min_{cam} (|P_{cam}(c_1) - P_{cam}(c_2)|) \quad (3.6)$$

The actual color distance $|\cdot|$ is normalized in the HSV color space. With the function $dist$ defined above, we construct a non-directional graph where k distinct colors (vertices) are chosen:

$$G_{th}(i, j) = \begin{cases} 0, & dist(c_i, c_j) \geq th \\ 1, & dist(c_i, c_j) < th \end{cases} \quad (3.7)$$

1: connected 0: not connected

The topology of Graph G is a function of threshold th . $G_{th}(i, j) = 1$ means colors i and j cannot be selected at the same time. The maximum number of colors we can choose equals the size of the maximum independent set (MIS) of G : $|MIS(G)|$. $|MIS(G)|$ increases from 1 to 2000 when th decreases from ∞ to 0. To select k colors, we decrease th until $|MIS(G)| \geq k$. Finding the MIS is a NP-complete problem [88], so we use the largest among 1000 maximal independent sets (a maximal independent set is a set that cannot be enlarged) as an approximation. Figure 3.10 shows the MIS of eight vertices (colors) chosen from a graph of 40 vertices with the selected vertices rendered with colors.

To understand how color selection affects the channel capacity, we compare k MIS-selected colors as described above with k randomly selected colors, $k = 1, 2, \dots, 1000$. The blue and red lines in Figure 3.11 show the capacity with k MIS-

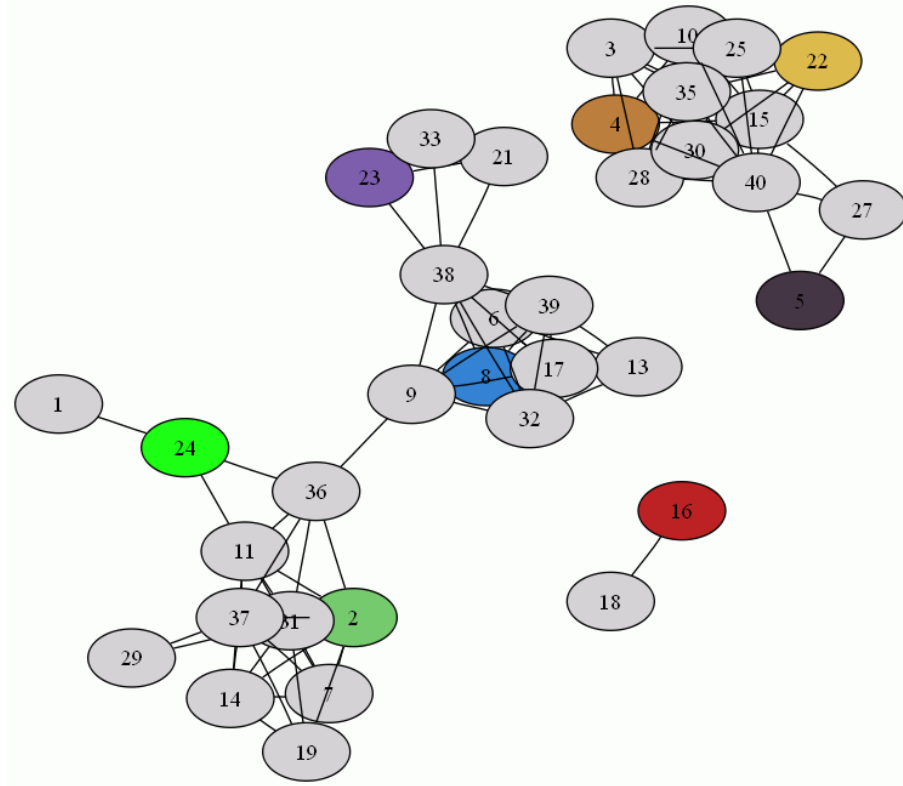


Figure 3.10: Eight optimized colors chosen from a graph of 40 vertices

selected and randomly selected colors, respectively. We can see the capacity of MIS-selected colors is always larger than that of colors randomly selected. When $k > 800$, the results are almost identical because the colors are inseparable even for MIS-selected colors. Figure 3.11 also shows the channel capacity reaches its upper limit (7.2 bits of information per pixel) when $k \approx 300$. When $k > 300$, the channel capacity does not increase with an increase in the number of colors. In practical situations, the capacity is even smaller since noise and lighting variations must also be considered.

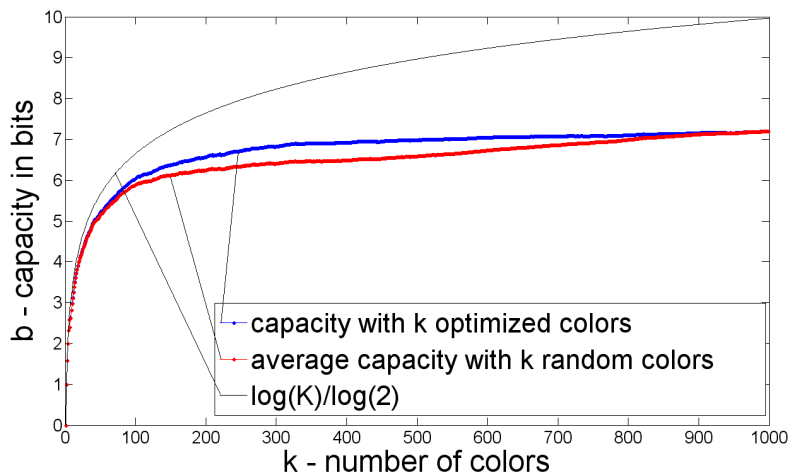


Figure 3.11: Simulation of channel capacity with k colors for Jamin camera phone

3.4.3 Color Calibration

We solve color variation by using color calibration. Tappen, et al. found a number of error sources in digital imaging systems[89]. Our experiment also shows there is an inevitable distortion of RGB values caused by imaging systems, as well as variation between displays. It is difficult, if not impossible, to build a universal color model which can be used to identify colors under any circumstance. To remove the effect of color distortion, we use color calibration[90] to update color models based on the actual environment in which it operates. Our goal is differs slightly from traditional color calibration, because we are more concerned with the bit information represented by the color (i.e. the original index of the distorted color) than restoring the original RGB value.

We place a color calibration pattern (Figure 3.12) with k known colors in the middle of each frame, representing $0..k - 1$ ($\log_2 k$ bits). When starting to download,

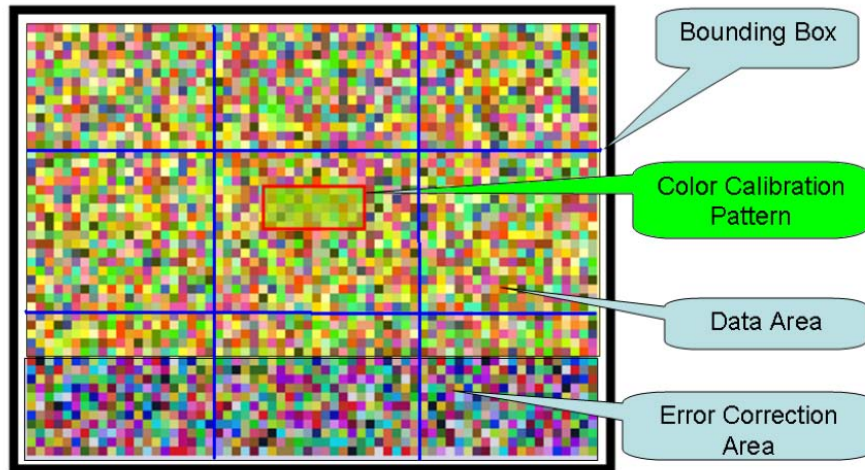


Figure 3.12: One encoded frame

the decoder does not know how the colors are distorted so it actively searches for a calibration pattern at the defined coordinates. The decoder matches every cell to its nearest neighbor in the calibration pattern which provides the bits information of the corresponding cell, and the frame is decoded when a maximal number of cells are matched correctly. Although the calibration pattern occludes a few cells in the frame, the corrupted information can be recovered by intra-frame error correction codes. The calibration pattern is used only for initializing the camera channel, and it is not necessary for every frame. After decoding the frame with the calibration pattern the decoder determines the color of every cells. We divide the entire frame into $3 \times 3 = 9$ zones (Figure 3.12) and build a color model (essentially k mean vectors for k colors) for each zone using the decoded color information. Building nine models instead of one global model allows for variation in color from zone to zone, so colors can be read accurately, which improves the overall bit rate. Experimental

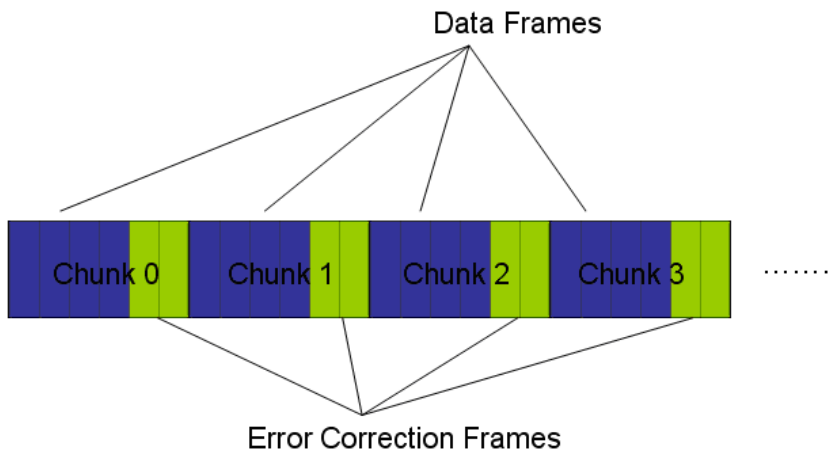


Figure 3.13: Data partition and temporal error corrections

results show that, by adapting the color model at run time, we have a smaller color recognition error rate with different displays and cameras.

3.5 Encoding and Rendering

As discussed in the introduction, one disadvantage of sending information through the camera channel comes when frame drop occurs and the user has to wait until the dropped frame appears again for recovery. We address this problem by adding inter-frame Reed-Solomon [82] error correction, in addition to the intra-frame error correction at the bottom of every frame. We first partition the data into equal sized chunks, each of which will be rendered as several frames. At the end of each chunk, error correction frames are appended (Figure 3.13), so if entire frames are dropped, we can use the error correction frames to recover the dropped frames. One powerful feature of Reed-Solomon error correction is that by using n frames of error correction, so we can recover n dropped frames with the knowledge of which

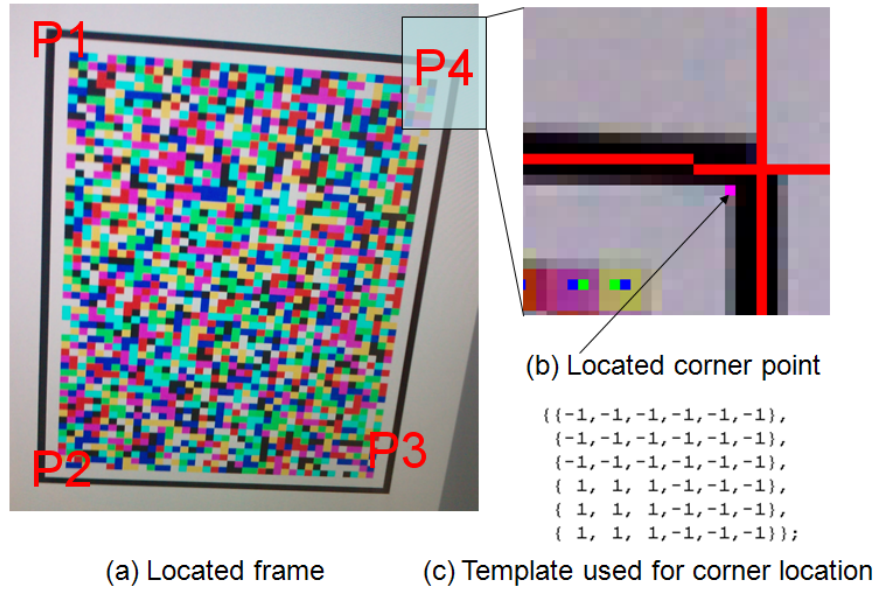


Figure 3.14: Location and registration

frames were dropped. Hence, temporal error correction is efficient.

As shown in Figure 3.12, one rendered frame consists of two parts, 1) a rectangular bounding box defining the boundary of the code and 2) a colored code area. The bounding box is used for registration and locating the color cells. These frames are packed into a GIF89a animation.

3.6 Decoding

The user may capture the animation from an arbitrary angle when using the camera channel, so perspective distortion must be corrected to locate each cell. It is not necessary to correct the entire image, but only to compute the coordinates of every cells in the camera captured image. Given a rectangular frame, four points must be registered to perform the correction.

Registration is necessary to determine the geometry and position of every cell as a critical step for decoding. A Hough transform based registration of the bounding box could be used, but it is not sufficiently precise to guarantee registration. The detected boundary may shift around the center line of the actual boundary, and it is not necessarily parallel to the center line. As a result, the density of cells could not be more than $\approx 40 \times 32$. To improve the accuracy, we use a two-level registration approach to obtain precision at the sub-pixel level. First, we locate the four black boundaries using a Hough transform (Figure 3.14a), and then compute the approximate position of the four corners. In the neighborhood of the approximate corner location, we convolve a template and find the maximum response point (pink pixel in Figure 3.14b). As an example, a template for the upper right corner is shown in Figure 3.14(c). After registration the four corner points from each frame, the cell coordinates can be computed using the fast perspective correction introduced in Chapter 2.

3.7 Experiments

Many factors may affect the throughput of VCode. To successfully decode a V-Code frame, the system needs to accurately locate each color cell and identify its color. The resolution, number of colors, and the camera's view angle all represent important factors. These factors are often correlated with each other, but to evaluate the performance of each quantitatively we first attempt to decouple them.

Using the binary VCode introduced in Section 3.2 as a baseline, we first ex-

perimentally measure the improvement obtained by more accurate registration and by color used to represent bit information. A checkerboard pattern is used to estimate the overall registration error. We then find the optimal balance between the code density and the number of colors. We establish a criterion for evaluating the strength of perspective distortion and measure how perspective affects the accuracy of code decoding. The level of error correction is adjusted accordingly, and a signal bar is added to help users aim the camera at the VCode. Finally we perform a throughput test with the optimized color set and compare it to a random colors set to show the importance of color selection. We use the iMate Jamin (200MHz CPU, 2M pixel camera) camera phone as our platform for testing.

3.7.1 Cell Location Accuracy

To extract color information from the image, each cell must first be located. We use a checkerboard to measure the overall accuracy of cell location. Ideally, what we read from the checkerboard is a bitstream with alternative 0s and 1s. Any bit that deviates from this pattern is counted as an error. We use a binary checkerboard to minimize the error caused by color mismatch. The total number of error bits reflects the accuracy of the geometric location of every cell. As a comparison, we also count the bit error using only the boundary location, and plot the result in Figure 3.15. Compared to bounding box registration, the pixel level registration approximately reduces the location error by 66%.

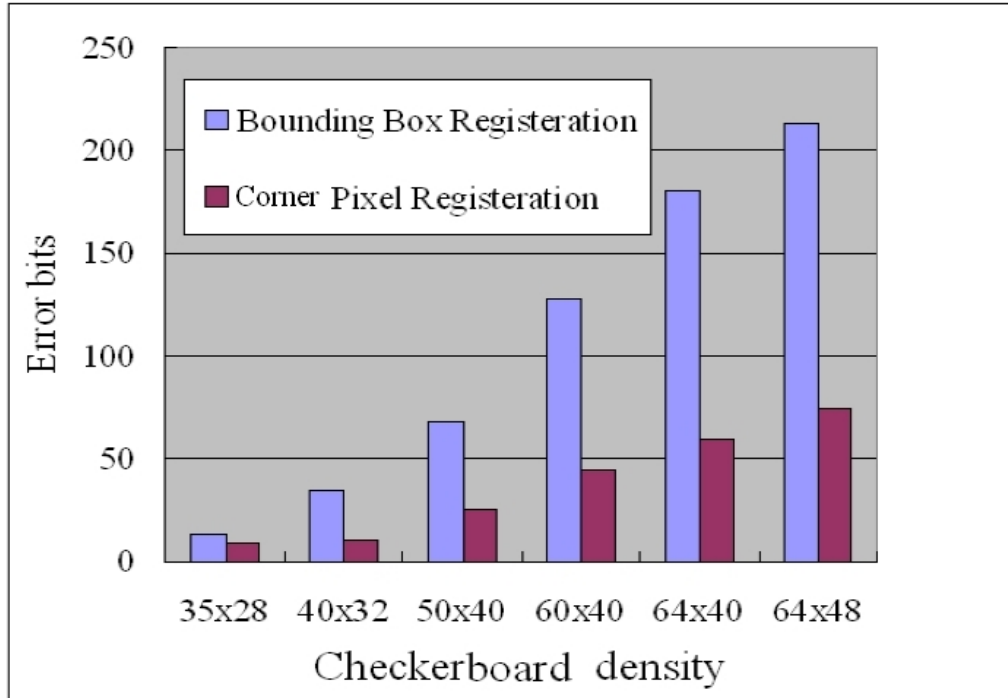


Figure 3.15: Location accuracy test

3.7.2 Color vs. Density

As shown by the simulation in Section 3.4.2, the bandwidth gained by adding more colors has an upper bound. In other words, the camera does not have an unlimited color resolution. Too many colors will create confusion, decreasing the throughput. Nevertheless, the physical resolution of the camera is also limited, e.g. it is impossible for a VGA camera to read a VCode frame denser than 640×480 cells. A color cell has to occupy much more than one pixel to be recognized correctly, so we have to reduce the number of cells in each VCode frame. Moreover, the number of colors is coupled with the density of a VCode, and together they determine the amount of information that can be extracted from each VCode frame. To maximize the throughput of VCode, we have to balance the number of colors and the density

of cells. The optimal combination is found with the following procedure. We use five different density settings 35×28 , 40×32 , 50×40 , 60×40 , and 64×40 and each setting is filled with 2, 4, 8, 16 or 32 optimized colors selected using our color selection method. We have a total of $5 \times 5 = 25$ different combinations, and, for each of the combination, we attempt to recognize a known VCode frame encoded with the corresponding setting 50 times. The number of correctly recognized cells C is recorded. The average of \bar{C} multiplied by the number of bits per cell B (the number of colors = 2^B) is the total number of correctly recognized bits T

$$T = \bar{C} \times B$$

Table 3.5: Resolution vs. Color

bits (B)	35×28	40×32	50×40	60×40	64×40
2	1862	2380	3680	4320	4454
3	2763	3531	5460	6264	6528
4	3600	4660	7204	6880	6752
5	3920	4800	7010	6910	6545

From Table 3.5, we can see that when the code density is low ($35 \times 28 \times 4$ colors) adding colors or increasing the density independently will improve the total number of correctly read bits. However, adding more colors will also cause more error when the code density is high (50×40). Similarly, increasing code density does not always produce a positive result. With more than 16 colors, higher density will, in fact, decrease the number of correct bits. This comes from the correlation

between neighboring cells. When the code density is higher, the aliasing generated in the imaging process starts to mix the color of neighbor pixels together. The captured color of a cell does not reflect the cell itself (as happened in our experiment in Section 3.4.2), but does reflect its neighboring cells. This correlation increases the uncertainty to the camera channel because the color of neighbor cells could be random. The theoretical capacity per pixel computed in Section 3.4.2 was based on the assumption of non-correlated pixels. Practically, we found the optimal setting to be $50 \times 40 \times 16$ colors. Note that our search of the optimal setting was not exhaustive. We have not evaluated the case when the number of colors is not a power of 2. For example, a better combination might be found at 24 colors.

3.7.3 Perspective Distortion

When a user captures a VCode, the camera may be positioned at an arbitrary angle and therefore perspective distortion may exist. Although the distortion is corrected, as described in Section 2.1, it might reduce the bit rate or even stop the data flow if the perspective is too strong (imagine if the code area is flatten to a thin line shape). To measure the strength of perspective distortion, we select four reference points S_1, S_2, S_3, S_4 which form a square shape in the up-front view angle (Figure 3.16). We use the ratio between the shortest edge and the longest edge as a criterion to measure the perspective:

$$K = \frac{\min(|S_1S_2|, |S_2S_3|, |S_3S_4|, |S_4S_1|)}{\max(|S_1S_2|, |S_2S_3|, |S_3S_4|, |S_4S_1|)} \quad (3.8)$$

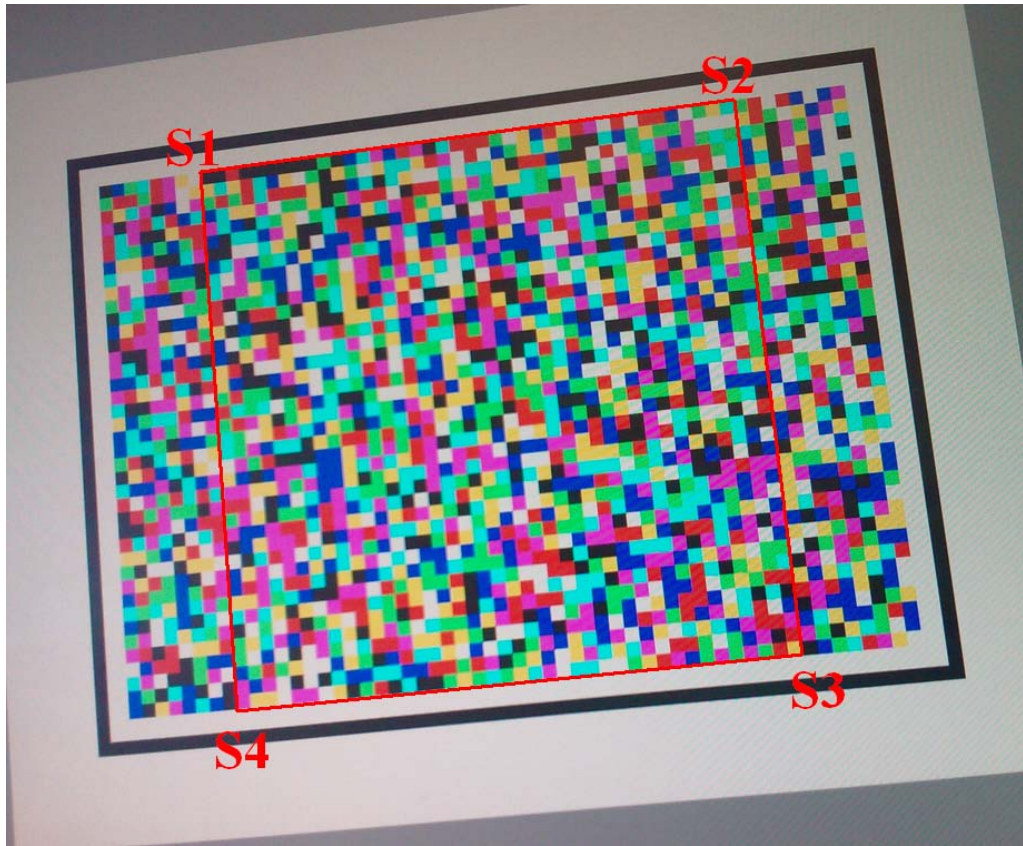


Figure 3.16: Perspective distortion

where $0 \leq K \leq 1$. $K = 1$ indicates no perspective distortion, and a smaller K indicates stronger perspective distortion. To measure how K affects the speed of a download, we use a known VCode as template (Figure 3.16) and capture this template at multiple angles. The number of correctly recognized pixels ($40 \times 50 = 2000$ at most) vs. K is recorded and plotted in Figure 3.17. As we can see, when K is large (> 0.8) the number of correctly recognized pixels is greater than 1800. We use Reed-Solomon error correction encoding meaning the number of erroneous symbols that we can correct is less than $1/3$ of the frame. Consequentially, when $K < 0.68$ (point A) the captured frame is impossible to decode because of too many (> 1333 pixels) errors. In fact, when $K < 0.6$, the perspective distortion is so strong that the correctly recognized pixels are random, resulting the frame as useless. When $K > 0.8$ more than 90% of the pixels can be correctly recognized, and we have 25% of the total frame for error correction codes. This level of error correction can correct 12.5% intra-frame error symbols (leaving some buffer when error is more than 10%). The perspective distortion is an important factor that affects the decoding of VCode, so we overlay a signal bar on our interface to inform the user about the perspective. The signal bar is updated in real time according to K , so the user knows whether the camera is facing the correct angle.

3.7.4 Throughput

The most important feature of a data channel is its bit rate. We measure throughput by recording the time it takes to download a 40KB file (a J2ME game)

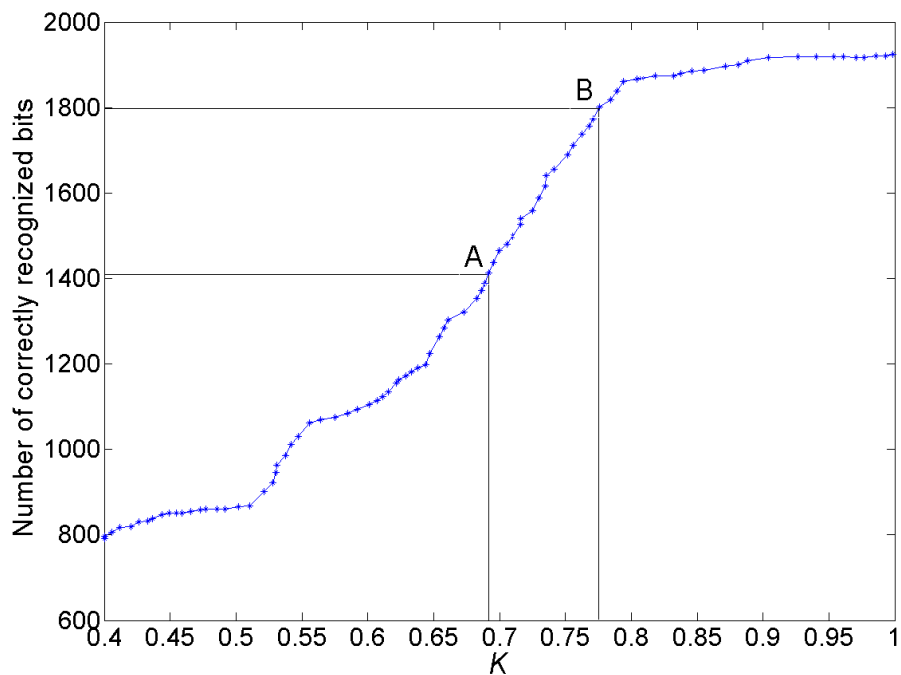


Figure 3.17: Accuracy v.s. Perspective(K)

via the camera channel. To simulate the real application scenario of the camera channel, we asked five users to download content by aiming the camera at the encoded animation displayed on a 15.4 inch LCD. We trained the users on how to use the system before they performed the experiment. For example, we suggested the VCode should be centered in the imaging area and the imaging area should be roughly parallel to the screen. The data was encoded as 50×40 cells with 16 optimized colors. To discover how optimized color selection affects the bit rate, we also encoded the same data, using a random color selection, for every user to download and plot both bit rates in Figure 3.18. The bit rate with the optimized color set is always higher than with the random color set. Note that user 4 actually had a 0 bit rate with random color sets because two of the colors were very similar

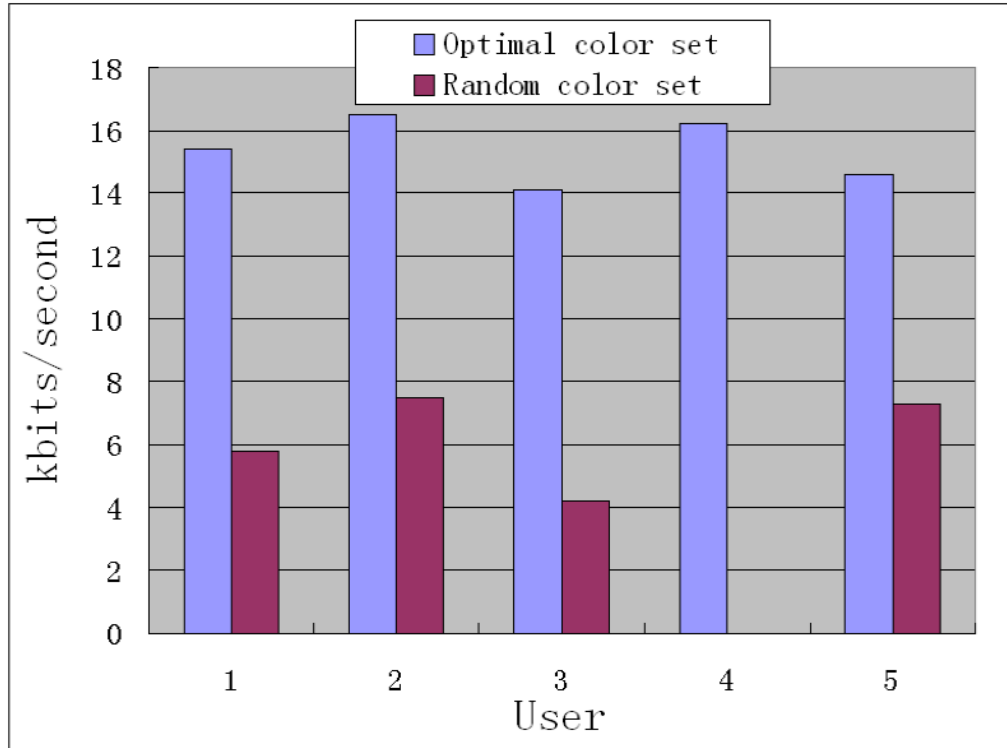


Figure 3.18: Throughput test

and became confused, so nothing could be decoded correctly. Overall, the average bit rate among these five users with optimized color selection is 15.4kbps, which is greater than double speed with the random color selection.

One question asks how well the five users that participated in the experiments represent the general users? To answer this question, we performed a detailed analysis of downloading process for the five users. When each user conducted the experiments, we recorded the time required for each 4KB block to be downloaded and saved it to a log file. For a 40KB file, ten blocks were recorded per user. This data is visualized in Figure 3.19, with each user represented by a different color. We can see that the download speeds follow approximately the same tangent ($\approx 2\text{KB}/\text{second}$) among the five users, illustrating that only the initialization (locating the code area

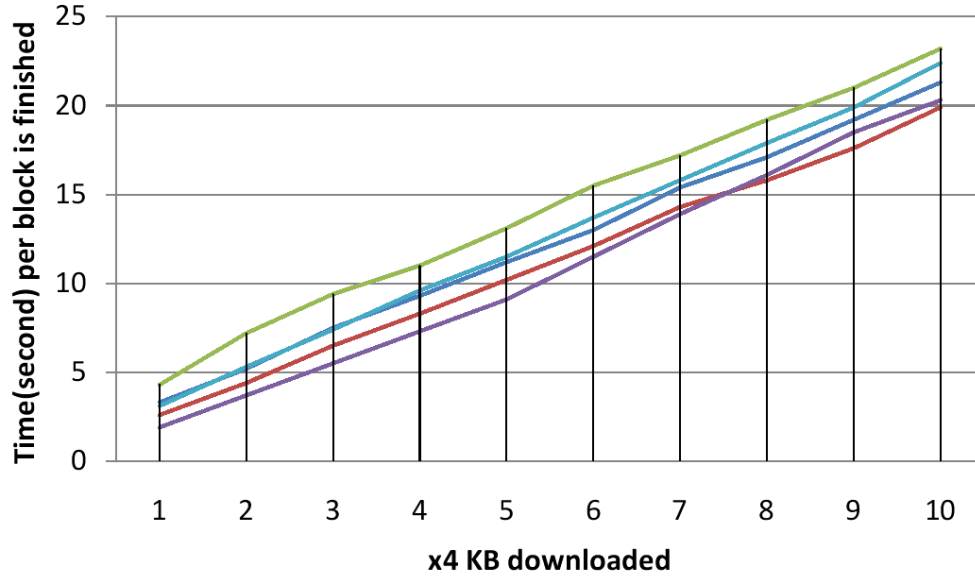


Figure 3.19: Progress of download

in camera) determines the overall bit rate for trained users. We expect to perform additional studies in the future work.

3.8 Summary and Discussion

We have analyzed a novel passive data download application for the camera on mobile devices. We have provided an upper bound of the camera channel's pixel capacity with color degradation based on mutual information theory. We also proposed an optimized color selection algorithm to maximize the channel capacity. With the help of a color calibration pattern we built the color model at run time. The cells' colors are recognized with a high accuracy to extract bit information. A two-level registration and a fast perspective correction are used to correct the geometrical distortion and to locate and read each cell. The current bit rate of

our implementation is 15.4kbps, which outperforms GPRS data channel's bit rate (average 12kbps[91]) of the second-generation mobile networks.

3.8.1 Why not use the Design of Existing 2D Barcodes?

Current 2D barcodes (QRCode, DataMatrix, etc.) are not designed for decoding multiple different codes in rapid succession. They are typically compromised of a large portion of the code as a finer pattern. Decoding a high density 2D barcode requires accurate registration of all the cell units, which is achieved by taking more pictures of the same code (in preview mode). The code is static, so the decoder can process ten frames per second and has to succeed only once, but, in our application, we need to identify as many correctly as possible. Alternatively, we may consider using an animated QR Code to encode data stream, but, if the density of each QR Code is high, frames will be dropped. Given the limited resolution of camera, we would like to maximize the data capacity in each frame, therefore we removed the finder pattern and extended the code to a rectangular shape. Using our code design, a more accurate registration can be performed in real time.

Chapter 4

MobileRetriever - Finding a Document with a Snapshot

4.1 Introduction

Document image retrieval is of special interest as a mobile application because it can be used as a tool to allow mobile users to interact with traditional paper documents. In our research, we will enable document retrieval from a large document repository using only a snapshot on the camera phone. Once retrieved, the user can annotate and manage the digital content, render it in different ways or send it to others. In this way a document with which the user interacts can become accessible in the digital world.

4.1.1 Motivation and Related Work

As the trend toward a completely automated office leads to more documents being created digitally (or digitalized), most of the physical documents that we see today have a corresponding electronic version. Many people, however, still feel more comfortable interacting the traditional paper documents because they are easier to browse and can be more portable. Consider, for example, a newspaper or a weekly periodical: such documents are typically configured for browsing with a front page that allows the reader to identify quickly articles of potential interest [92]. While web or electronic access is growing in popularity, searching is typically the preferred

means of content selection, and this is typically done in front of a computer. We are exposed to printed material continuously throughout the day, and this is of great importance to publishers and advertisers. Publishers recognize the fundamental difference in reaching users with new content via paper vs. electronic content. Users are much more likely to stumble on and read material when browsing hard copy. It is therefore likely that physical and digital documents will coexist for the foreseeable future. Readers can benefit from both forms of documents if we can link them in a meaningful way.

During the past several decades, numerous attempts have been made to link physical documents with their digital representations [93, 94, 95, 96, 97, 26]. Pioneering research was conducted by Pierre Wellner, et al. in the early 1990s [95]. Cameras were mounted above the “Digital Desk” (Figure 4.1a) to read the documents on it and analyze human action. T. Arai, et al.[97] attached a camera to a “VideoPen” (Figure 4.1b) and explicitly used optical character recognition (OCR) to extract a small piece of text and retrieve the text from a “pattern buffer”. Since the “VideoPen” stays in close proximity to the paper, however, only a tiny portion of a few words can be captured, and the “pattern buffer” could store only a limited number of words for matching. The primary application of these technologies was to edit the e-text. In recent years, this technology has been extended to multimedia documents with multi-modal queries [4, 17, 98, 99, 100, 101, 102].

In alternative application, the user retrieves the electronic document so it can be presented to the user in an alternative form - as a summary, translated, on a custom web page, or, perhaps, read and delivered through a mobile device.



(a)



(b)



(c)

Figure 4.1: Digital Desk(a), VideoPen(b) and MobileRetriever (c)

Exbiblio (www.exbiblio.com) is an ambitious project that involves building a key-chain scanner and retrieval from very giga-page document repository. All of these applications provide publishers and advertisers tremendous potential for overcoming problems of reduced revenue caused by the internet, and introduce some interesting research problems for document retrieval from printed material, and access to large document repositories[103].

Historically, document retrieval may be categorized as text based or image based. Text based retrieval is relatively straightforward for known query terms because it is essentially a pattern matching problem, which has been exhaustively studied and is highly parallelizable. Image based methods are more difficult because no off-the-shelf meta-data exists to be used to form a query. Often OCR is used to recognize the text so traditional text based methods can be used, but image quality and articles with similar content can degrade accuracy. Alternatively, unique image features can be extracted for the query and used to index the repository and various approaches has been explored. In [96], Hull proposed a series of distortion-invariant descriptors allowing robust retrieval against re-formatting, re-imaging, and geometric distortion. In [104], Cullen, et al. use texture cues to retrieve documents from a repository. In [105], Tan, et al. measure document similarity by matching partial word images. In [106], Kameshiro, et al. describe the use of an outline of the character shape that tolerates recognition and segmentation errors for document image retrieval. Associating an electronic text with the image then becomes trivial. However, the approaches above assume the query image is from a scanner and rely on the fine structure of texts which are often absent from a camera image.

Using a mobile camera phone as the potential link (Figure 4.1c) has many advantages, where a snapshot provides access to the electronic source. With image acquisition capability, communication capability and portability, camera phones become a natural bridge between the physical and digital worlds. Document retrieval is one such “bridge” application, which we call “MobileRetriever”. The work flow of MobileRetriever is illustrated in Figure 4.2, with the green portion showing the physical world. The user snaps a picture of a piece of an article of interest and the pre-installed software sends the picture (or features extracted from the picture) to the server as a query. If the picture matches a document in the repository, a visual or textual representation will be returned to the user. Sophisticated retrieval and matching algorithms may be applied on the sever side, but all the user has to do is to capture a part of the document.

4.1.2 Application Scenarios

Using mobile document retrieval, we can link virtually any physical document to its digital version and promote many interesting applications. For example, one can tag an article that the user has not finished reading and have it available for access later. Retrieving and downloading a desired article may cost a few cents which could provide alternative revenue to the publishers, benefiting both. Considering related advertising opportunities, this service could even be free. While similar goal can be achieved by attaching pages with “barcodes” [63] and have the user scan the barcode, such solutions have met with resistance because they require the status

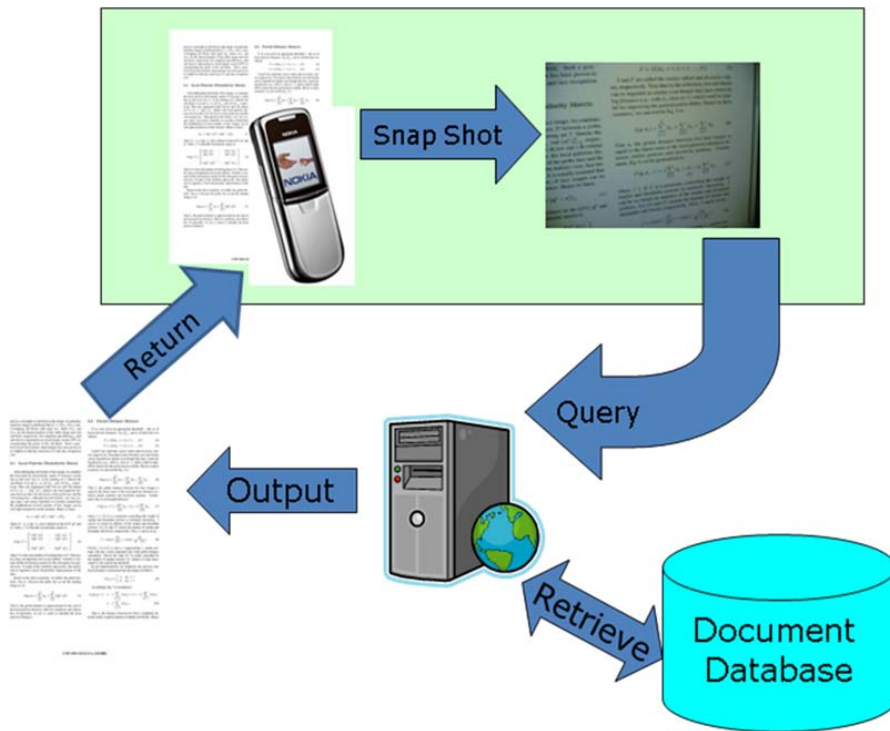


Figure 4.2: Overview of MobileRetriever

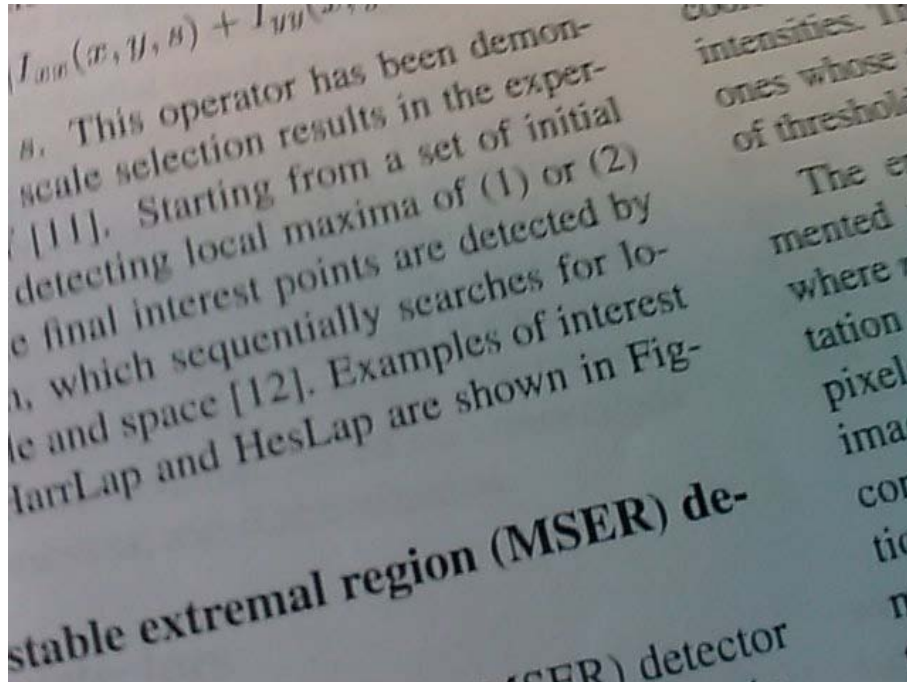


Figure 4.3: A typical query image of MobileRetriever

quo to change. Unlike barcodes, our system knows which part of the page the user is targeting so applications can provide more targeted feedback according to the query, such as keyword explanations or hyperlinked references. For example, other applications include helping the visually impaired user read newspaper articles using retrieval plus text-to-speech (TTS) technology, and translating a retrieved digital document to any language using machine translation.

4.1.3 Challenges

A typical query to the MobileRetriever system appears in Figure 4.3, and it illustrates three major challenges. First, the picture may capture only a small portion of the page because it is impractical to require the user to take a full page snapshot.

Therefore, any retrieval technique that relies on the global structure of whole page may not apply. Second, the picture is taken with a low resolution camera. The contrast and sharpness is largely affected by the environment's lighting. Typically the embedded camera has a fixed focal length that is set for shooting general scenes or portraits, so a document that is captured too close to the camera may not be well focused. Character recognition will have a low accuracy on these images, and therefore, text based retrieval will not be practical. Third, the captured image is subject to perspective distortion (including rotation and translation), and therefore, the retrieval must be based on features that are invariant to these distortions.

Given these challenges, some specific questions need to be answered to build the MobileRetriever system. From a degraded image with perspective distortion, how can one extract stable information which can be used for retrieval? How can one efficiently index large numbers of such documents? Since the captured image covers only a small portion of the page, what can be used to guarantee the accuracy of retrieval? From the user's point of view, if the captured document is not in a repository, at what point shall we reject the query? For evaluation, what is the false negative rate? We will address these questions in the remainder of this paper.

4.1.4 Related Work

Researchers have presented document image retrieval techniques using layout[107], signatures[108], logos[109], and OCR, but these may not be suitable for our MobileRetriever concept. The global structure (layout) may not be captured in the partial

snapshot. Signatures and logos are often absent, and the image quality hardly provide accurate OCR. As stated above, the query we used for retrieval is a low quality image with perspective distortion. It is impractical to use features that rely on the fine structures of the characters or are vulnerable to perspective distortion. Nakai and Kise [110] proposed locally likely arrangement hashing (LLAH), which computes affine invariant from the coordinates of the four nearest neighbors of each word as a hash-code for indexing and retrieval. However, the nearest neighbors may vary as view point changes. Liu, et al. [111] proposed to use SIFT-like feature for document retrieval, but SIFT is computationally intensive on the device and has many false positives due to the rich textures from document images. We used the layout context feature extracted from the camera captured document image[112]. Erol and Hull presented a similar approach but with real time performance on a smaller repository [26]. Layout context can be used to index and retrieve documents efficiently but it is not invariant to strong perspective distortion.

Three important questions have not been answered in the literature in document image retrieval. First, how can we tell whether a query image is in the repository or not? It has not been quantitatively estimated in previous researches how many words should be captured for a success retrieval from a large repository ($> 10^5$ pages). Second, existing document image retrieval systems seldom discuss at what point a query should be rejected if it is not in the repository. Third, since the query covers partial documents, what is the minimum requirement of coverage that can lead to a successful retrieval? In this chapter, we address the these problems and perform evaluation on a document repository with 100,093 pages.

4.2 Approach

For the MobileRetriever, we use “token pairs” defined as words with close proximity in the image. Shape coding is used for indexing, and geometrical invariant “token triplets” are used for verification to prune the false positives. Compared to existing approaches, our shape coding and token-pair-triplet approach is more tractable because the features are extracted from the words and their relative positions. We test its performance on a repository of 100K pages, which is 10 times larger than the latest result reported in [110].

In Section 4.2, we present our solution to index the documents using a query of a partial document image with degradation. We also define the token pairs and triplets. In Section 4.3, we show how they are used for retrieval and verification. In Section 4.4, we briefly describe the system architecture of the MobileRetriever and data collection. Details of our experiments measuring performance and accuracy of retrieval are presented in Section 4.5. We conclude and discuss possible extensions in Section 4.6.

4.3 Indexing

In this section we introduce two sets of indexing techniques for different quality document images. For document image taken by a camera phone without close-up lens we use Layout Context which is solely based on the coordinates of words. For document image taken with close-up lens we use shape coding to smooth OCR errors for indexing and retrieval. Our retrieval algorithm can be coped with both of the

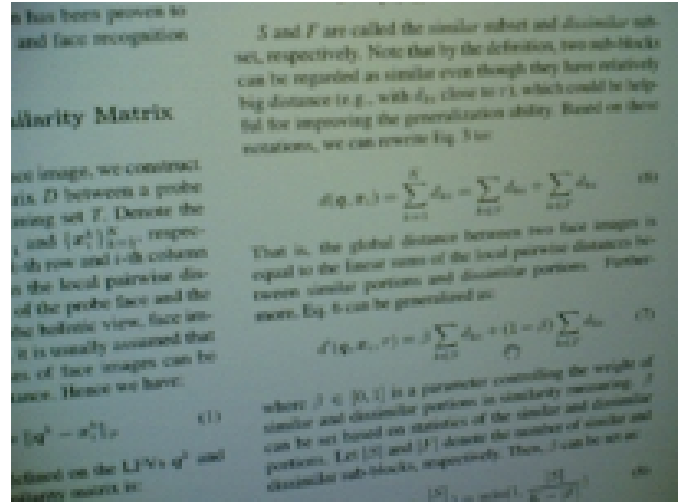


Figure 4.4: A typical document image captured by a camera phone

indexing methods.

4.3.1 Image Captured without Close-up Lens

As we discussed in Chapter 1, camera phone images may not have a high quality and are usually out-of-focus if taken at too close to the source without a macro lens. For a document image at this low quality, traditional features are almost impossible to extract for OCR. A typical example is shown in Figure 4.4. For low quality document image retrieval, Nakai and Kise [113, 110] proposed locally likely arrangement hashing (LLAH), which computes affine invariant from the coordinates of four nearest neighbors of each word as a hash-code for indexing and retrieval. However, the nearest neighbors may vary as view point changes. In this section we introduce “Layout Context”, a robust feature for low quality document images, which is computed from the layout of word boxes and does not rely on the fine structure of the characters.

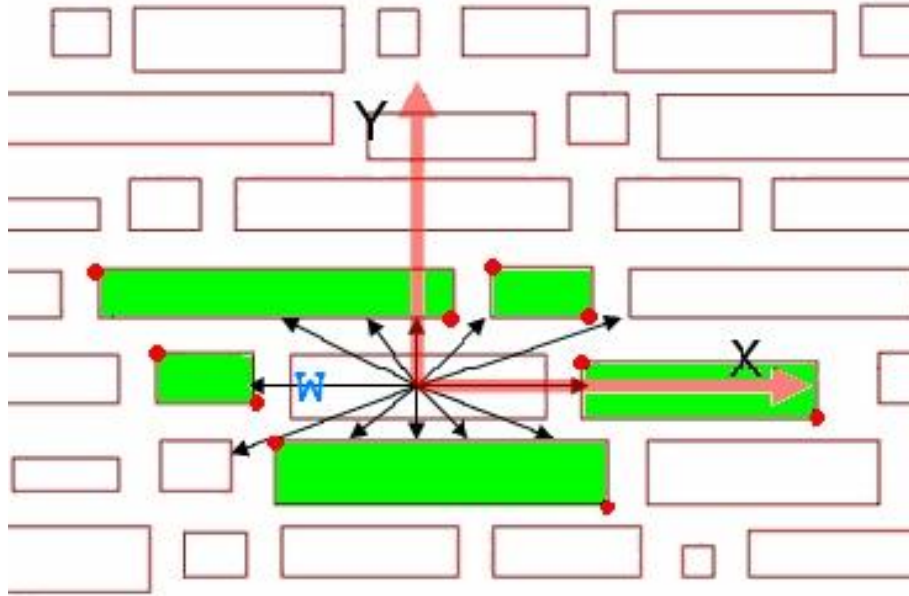


Figure 4.5: An example of Layout Context

We begin with an ideal image with no perspective distortion. In Figure 4.5 each rectangle shows a bounding box of a word. To extract the layout context of a word w in Figure 4.5, we start at the center of the word and look for the most visible n neighbors. Figure 4.5 shows, five green rectangles for $n = 5$. The visibility is defined by the angle of the view, and the top n can be extracted using an efficient computational geometry algorithm with complexity linearly bounded by the total number of nearest m neighbors (it is safe to choose $m = 10n$). The top n visible neighbors are invariant to rotation and the percentage of view angles that a neighbor word occupies will not be effected by rotation. We place the coordinate system origin at the center of w with the X-axis parallel to the baseline of w and define the unit metric using the width of w . Under this coordinate system, the coordinates of the n most visible neighbors are invariant to similarity transformations.

- Translation: the original point always remains at the center of word w .
- Rotation: the X-axis always falls along the direction of the text lines.
- Scale: distortions are normalized by the width of word w . To use width of w as a stable factor of normalization, w must have an aspect ratio greater than a threshold (3 for example). This condition is satisfied by a large portion of words that have more than three characters.

With $n = 5$, a layout context is a vector that occupies $5 \times 2 \times 2 = 20$ bytes of data. When a document image is captured using a camera phone, it undergoes perspective transform, but locally can still be approximated by a similarity transform. For a similarity transform, scale, translation and rotation have to be normalized. We detect the baseline of text lines and rotate them into the horizontal direction. After this, the scaling normalization is the same as for a perfect image.

4.3.2 Image Captured with Close-up Lens

As stated in the challenges, a camera captured document is typically not of sufficient quality for an accurate character recognition. Shape coding was introduced as a document retrieval technique, which is more robust to the degradation of character recognition [114, 115]. The basic idea is to reduce the surface form of glyphs using only robust low level features such as width, height, peaks, and valleys. One way to implement this idea is to represent characters that may be confused with each other with a single representative symbol. For example, letters “c” and “e” are often confused because of their similar appearance, so we can remove letter “e” from

Table 4.1: Reduced Alphabet for Indexing

ceoCDQ0→c	lijJLl1→i	f,t→f	68→6
7Zz→z	9gq→g	BEF→B	ri→n
YVyv→v	li→h	tn,rn,nn→m	cl,el→d
<p>Example:</p> <p>In this paper we describe an image based document re- trieval system</p> <p>→</p> <p>in fhis paper we dcscribc an image bascd dccumcnf rcm- cvai svsfcm</p>			

the alphabet and replace the letter “e” with the letter “c” in the indexed document. Table-1 shows part of the reduced alphabet we use in MobileRetriever. Each group of characters is replaced by a representative symbol. This alphabet is built using the confusion matrix obtained from the OCR software that processes the query image. Having fewer letters in the alphabet may result in some information loss and confusion for some words, e.g., “cat” and “eat,” but with a sufficient number of tokens in the query, accurate retrieval can still be achieved.

Bigrams and trigrams[116] are used intensively in language models. To utilize the geometric relationship of words, we generalize the concept of bigram to a pair of

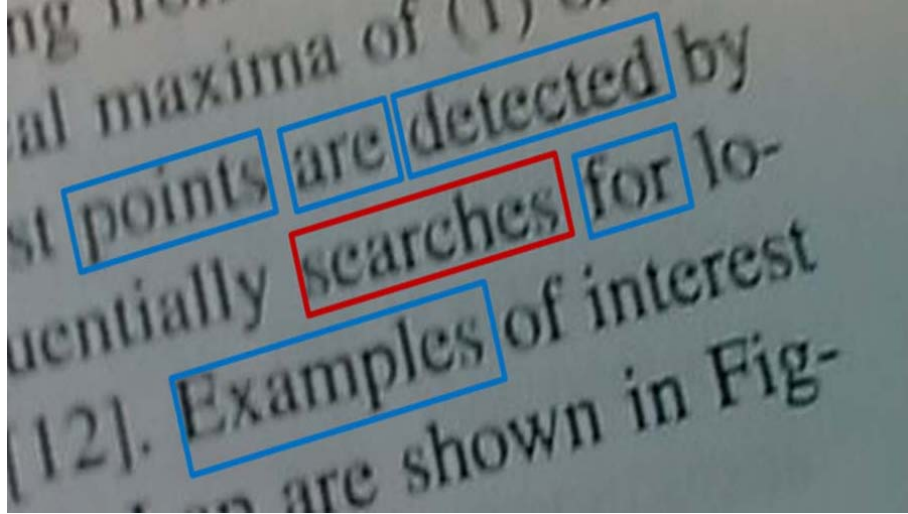


Figure 4.6: Token pairs

tokens that are geometrically close to each other (not necessary in reading order). A token pair is much more unique than a single token because it captures the context.

Figure 4.6 shows five token pairs centered on the word “searches”: searches-detected, searches-Examples, searches-points, searches-are, and searches-for. The K-nearest (we use $K=5$) neighbors are indexed, but only the nearest neighbor is used for retrieval. The nearest neighbor may vary as point of view changes, but, since we index K-nearest neighbors, it will likely be covered by one of the indexed pairs. For a page with N words, there will $K \times N$ token pairs will occur in the index, and the indexed tokens are substituted using the reduced alphabet.

A token triplet presents an even stronger geometric constraint. A token triplet consists of three words in the image and an orientation (clockwise or counterclockwise) of the three words. It is invariant against viewpoint change or even page distortion. As shown in Figure 4.7, the orientation of the triplet (A, B, C) on a page

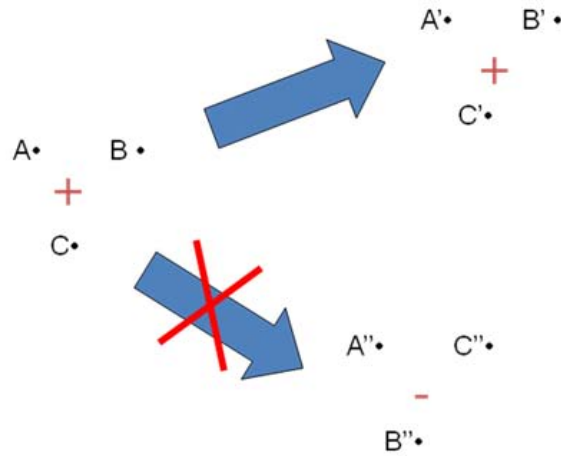
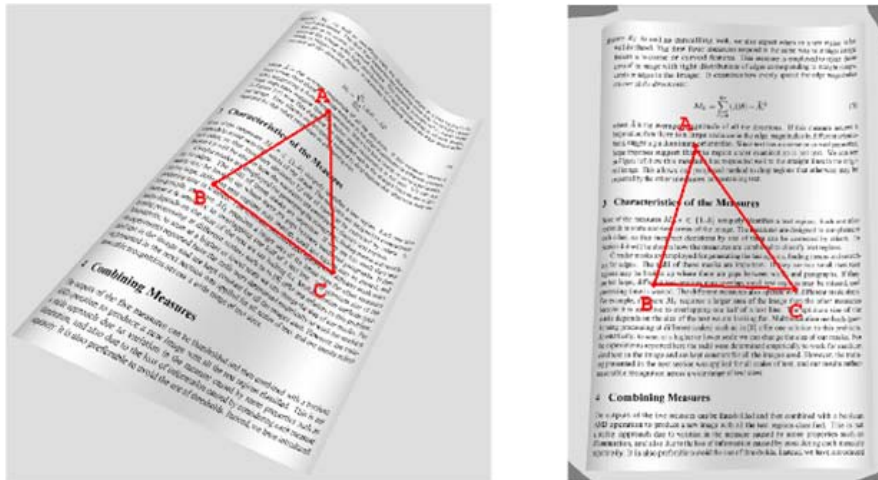


Figure 4.7: Token triplets

is defined as

$$Sign\left(\begin{vmatrix} X_A & Y_A & 1 \\ X_B & Y_B & 1 \\ X_C & Y_C & 1 \end{vmatrix}\right), Sign(T) = \begin{cases} 1 \dots T \geq 0 \\ -1 \dots T < 0 \end{cases} \quad (4.1)$$

When the page is viewed from another angle, these three points appear as A', B' and C' , and we have

$$Sign\left(\begin{vmatrix} X_A & Y_A & 1 \\ X_B & Y_B & 1 \\ X_C & Y_C & 1 \end{vmatrix}\right) \times Sign\left(\begin{vmatrix} X'_A & Y'_A & 1 \\ X'_B & Y'_B & 1 \\ X'_C & Y'_C & 1 \end{vmatrix}\right) = 1 \quad (4.2)$$

The orientation uniquely defines the geometric relationship of these three words (A, B, C) and is robust against perspective distortion or surface bend.

When a point set S in the captured image is matched to another point S' in the indexed page, we define the match score as

$$\sum_{A,B,C \in S} \left(Sign\left(\begin{vmatrix} X_A & Y_A & 1 \\ X_B & Y_B & 1 \\ X_C & Y_C & 1 \end{vmatrix}\right) \times Sign\left(\begin{vmatrix} X'_A & Y'_A & 1 \\ X'_B & Y'_B & 1 \\ X'_C & Y'_C & 1 \end{vmatrix}\right) \right) \quad (4.3)$$

For a page with N words there are $\binom{N}{3}$ token triplets, which is too large for indexing for $N > 200$. Therefore, we use the token triplet for verification instead of indexing. The token triplet offer a strong cue to reject queries that do not reside in the document repository.

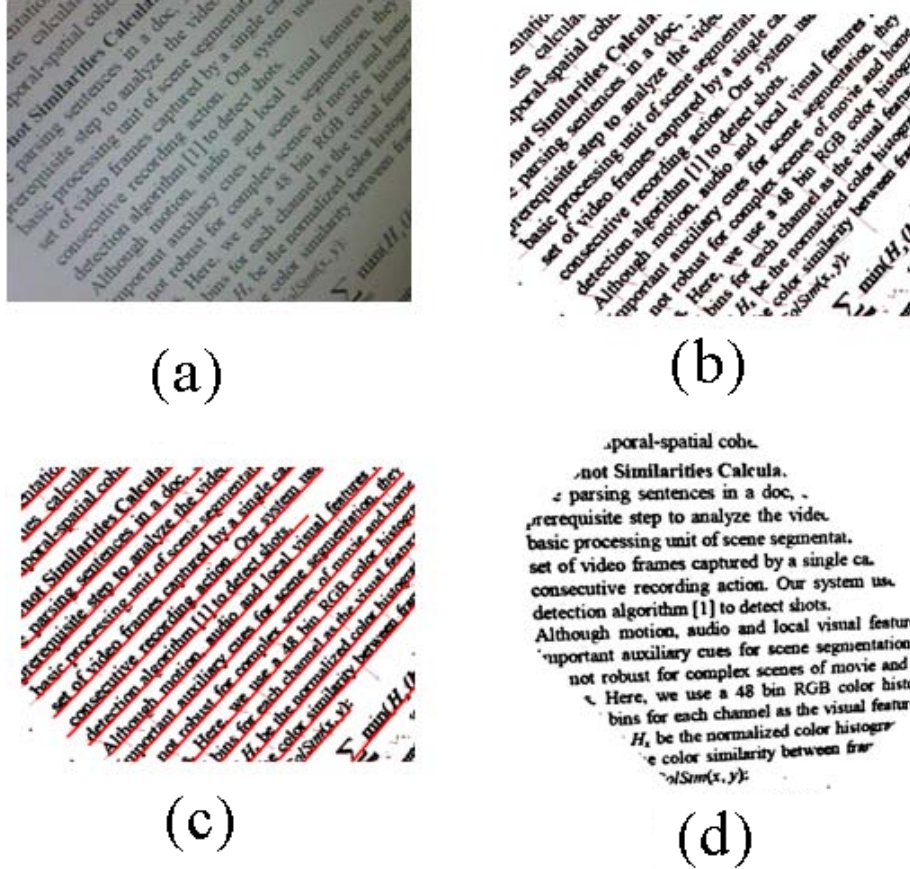


Figure 4.8: Image enhancement

4.4 Retrieval

The input to MobileRetriever is an image captured at an arbitrary viewing angle, as shown in Figure 4.8a. We enhance the input image to facilitate tokenization in two steps. We first enhance the contrast using an adaptive Niblack [117] binarization (Figure 4.8b). We then compute the approximate rotation using the bottom line of each word (Figure 4.8c) with a Hough transform. The captured image is a perspective image, but we are not using the exact position of each word. The image is de-skewed using the estimated rotation. Finally, the enhanced image is sent to

an OCR engine, and the recognized characters are transformed to the reduced alphabet. We extract token pairs with the coordinate of each word, use these pairs to query from the document repository, and rank the pages in the repository according to the hits of token pairs. Those pages with more hits of token pairs in the query are ranked higher. We, however, cannot simply return the highest ranked page as the result for several reasons. First, there can be multiple pages with the same “highest” ranking. Second, the query may not be in the repository. The highest ranked page is simply the one that shares most token pairs with the query. Third, due to recognition errors or incomplete words (on the boundary of the captured image), the highest ranked page may not always be the correct result. We use the initial list as the hypothesis and filter it as follows.

We first collect the H highest ranked pages and verify that the orientation of token triplets are consistent with the query. This step retains the page if it exists in the repository and rejects the query otherwise. We score the page candidates, increasing its score by 1 if a triplet has the same orientation as in the query and decreasing its score by 1 otherwise. If N words are in the query, the highest score a page could obtain is $\binom{N}{3}$ and the lowest score is $-\binom{N}{3}$. Two questions arise, first how many page candidates (H) need to be generated to ensure the one of interest is among the candidates? Second, how many words (N) are required for a successful retrieval? We will briefly describe our implementation and repository, and then experimentally answer these questions in Section 4.5.

4.5 Implementation

4.5.1 Client

Our MobileRetriever client is implemented and will run on any Windows Mobile platform with at least a VGA (640x480) camera. The client uploads the captured image to the server, and image processing and retrieval are performed on the server side. The reason we do not process image on the mobile device is that it does not have enough power for intensive computing. It will take more time to process the image on a device than to upload it to the server, although this may change as more powerful CPUs are installed on devices.

The results of retrieval are shown via an http link that can be loaded on the mobile device. We have not designed a specific user interface for the MobileRetriever because so many different platforms are evolving way rapidly. A browser is a standard component available on almost all the mobile handsets and the appearance will be, in general, consistent. A typical retrieval result is shown in Figure 4.9. It not only shows which page is retrieved but also the specific area of text in the query to demonstrate establishing local proximity.

4.5.2 Server

Our server listens to TCP port 21 and uses standard FTP protocol for receiving the query image. The server creates a unique ID for each request and redirects the client browser to the URL pointing to a dynamic page generated as the response of the request. The response containers either the retrieved page or a rejection

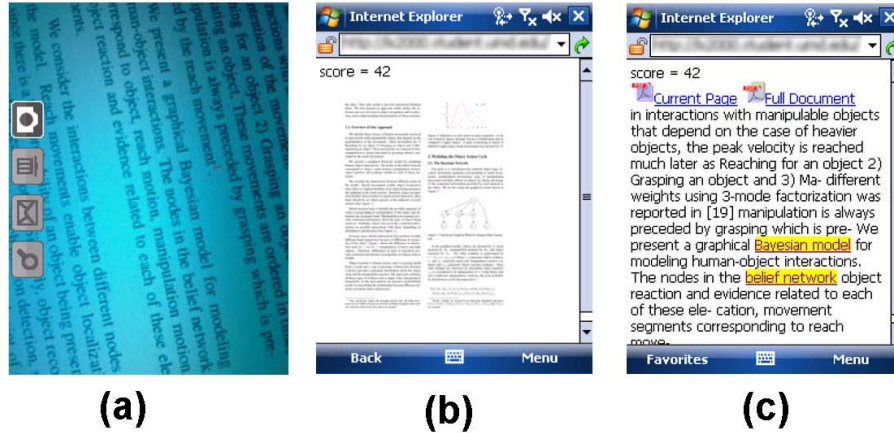


Figure 4.9: MobileRetriever on Windows Mobile 6

(a) snapshot (b) thumbnail view of the full page (c) text and PDF

message.

4.6 Evaluation

We have collected 100,093 pages from the proceedings of the recent computer vision and image processing conferences and will evaluate the accuracy, usability, and speed of MobileRetriever on this data set ¹. The token pair index for these 100,093 pages is 974M bytes. We did not build the index with token triplets, as the token triplets are only computed when verifying the top pages retrieved with token pairs.

We have tested the accuracy, usability, and speed of MobileRetriever on the collected data set with AT&T Tilt - A Windows Mobile 6 Pocket PC device at a camera resolution of 800×600 with auto-focus.

¹A complete list of our collection appears in Table 4.2

Table 4.2: Document Repository

Source	Page	Source	Page	Source	Page
CVPR2007	3741	CVPR2006	4001	CVPR2005	3501
CVPR2004	3397	CVPR2001	1934	ICCV2003	2719
ICCV2005	3376	ICCV2007	3160	ECCV2006	2676
ICASSP 2001	4140	KDD2007	1403	ICME2000	2045
ICME2003	2412	ICPR2002	4024	ICPR 2004	4638
ICPR2006	5660	SPIEPW2001	8381	SPIEPW2005	8975
SPIEPW2006	8959	SPIEPW2006	7593	SPIEPW2008	7460
WACV05	898	Other EBooks	6000		
Total = 100093					

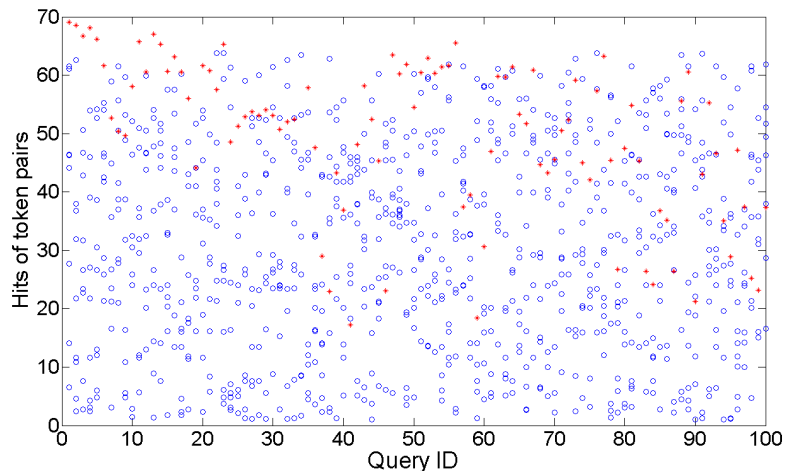


Figure 4.10: Retrieving using token pairs

4.6.1 Retrieval using Token Pairs

4.6.1.1 Pages in the Repository

We first tested the accuracy of retrieval using only the hits of token pairs. We randomly selected ten pages known to be in the repository and captured ten snapshots from each page. These 100 images (ID:1..100) are sent as queries to the MobileRetriever system and rank the pages in repository only according to hits of token pairs. In Figure 4.10, we scatter plot the top ten candidates of each query. The pages that match the queries are marked with red stars. We can see from this plot that the correct page may not rank the as highest among all pages. A histogram of the ranks of the correct pages (Figure 4.11a) shows that, by returning the highest ranked page, we only have an accuracy of approximately 18%. We draw the integral line of Figure 4.11a in Figure 4.11b. Figure 4.11b estimates how many top candidate pages (H) we need to produce to capture the one that matches the query. By taking

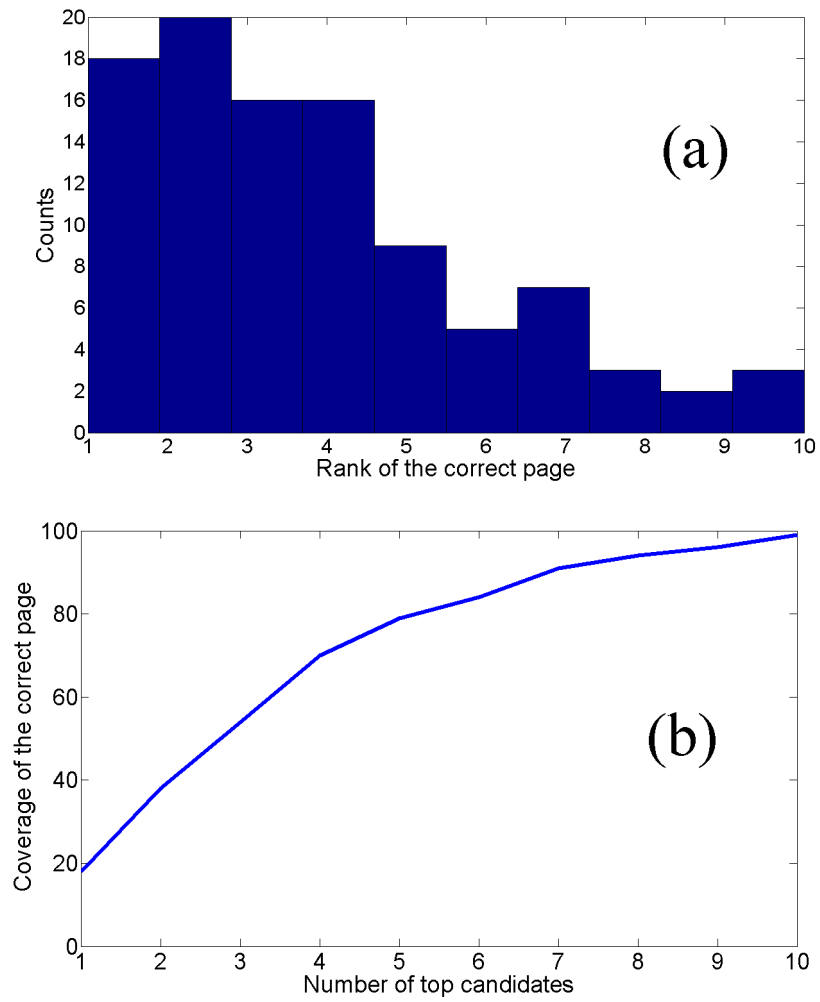


Figure 4.11: The page that matches query may not be ranked first

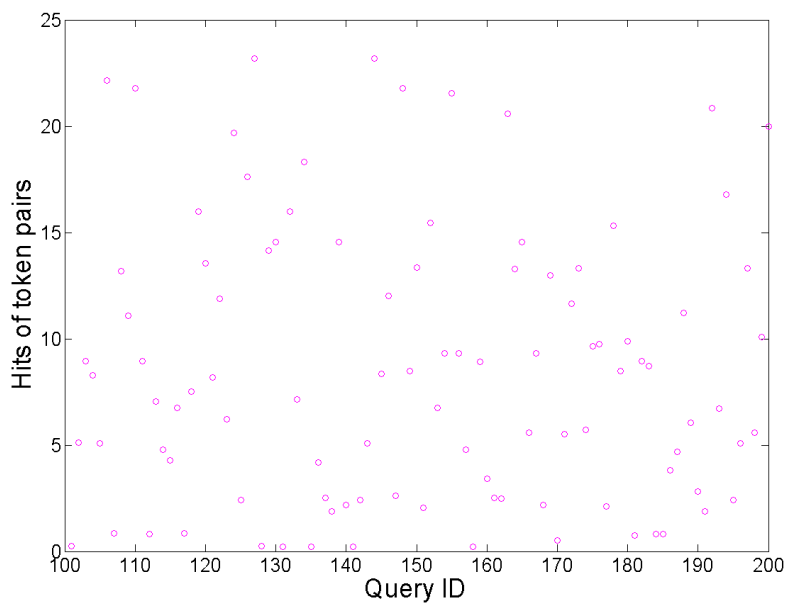


Figure 4.12: Queries have no match in repository still have hits of triplet pairs

the top five candidates we cover 80% of the correct pages among these 100 queries and if we take top ten the coverage is increased to 99%.

4.6.1.2 Pages not in the Repository

One situation that has not been given enough attention in the literature is when the query does not exist in the repository, the so called “reject” case. Similar pages may be returned, but ideally the query should be rejected. Although inexact matches may provide information, for MobileRetriever we would like the user to know whether the exact document is present. This is important, for example, when related information such as translation is required. To test how token pairs work in this situation, we randomly collected ten pages that are not in the repository

and took ten snapshots from each page. We queried the system using these 100 images (ID:101..200) and plotted the topmost candidates for each query in Figure 4.12. Comparing Figure 4.12 (query not in DB) to Figure 4.10 (query in DB), we find that by using only the token pairs we cannot judge whether a query exists in the repository or not. Some of the queries that are not in repository receive more hits (>20) than the queries that actually are in repository (<20).

Testing of token pairs shows that the number of hits of token pairs is insufficient for retrieval because the page that matches the query may not be ranked highest and those queries that are not in repository cannot be rejected using only token pairs. We address this problem using token triplet verification.

4.6.2 Token Triplet Verification

4.6.2.1 Pages in the Repository

After ranking the pages using the hits of token pairs, we take the top N pages and apply token triplets verification. The scores of token triplets verification are computed using equation 4.3 and appear in Figure 4.13 for $N = 10$. Compared to Figure 4.10, these pages that matches the query (red star) are distinguished from other candidates (blue circle). Therefore token triplets verification can choose the correct match from the top pages ranked by hits of token pairs.

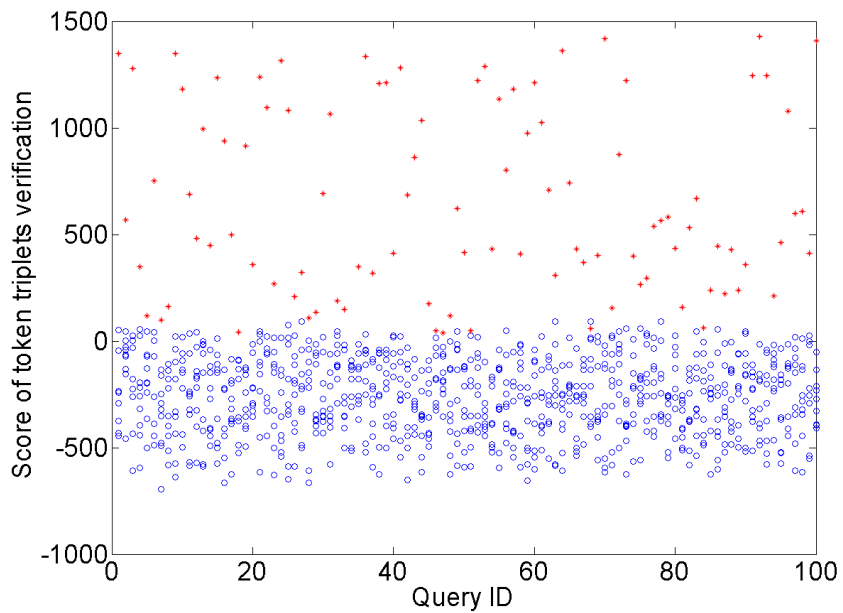


Figure 4.13: Correct retrievals are distinguished after token triplet verification.

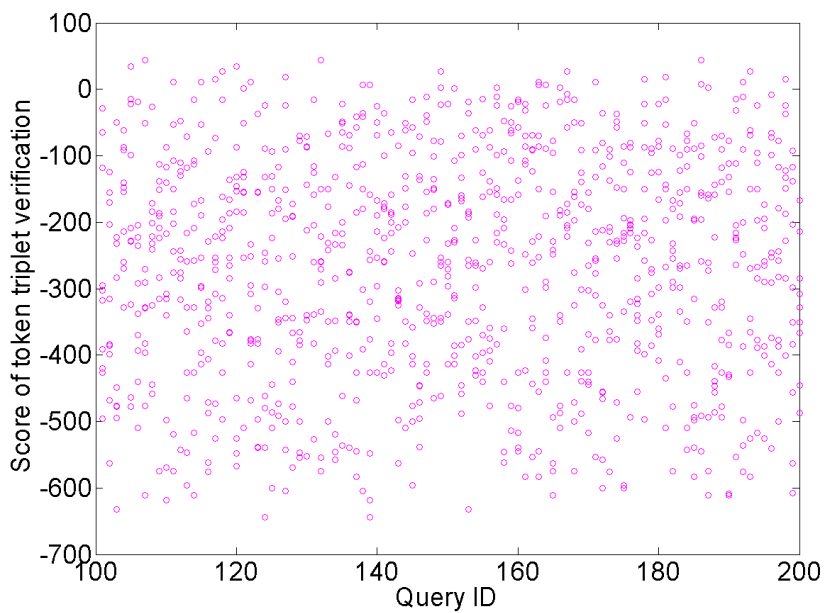


Figure 4.14: Pages not in repository have low scores after triplets verification.

4.6.2.2 Pages not in the Repository

For those queries not in the repository, we also apply token triplets verification of the top M pages. Figure 4.14 shows the score of 100 queries that are known not to be in the repository for $M = 10$. Their scores after triplets verification are, in general, less than zero. Although these queries may have some common token pairs with a page in the repository, more than half of the triplets on average will have reversed orientation, and, therefore, will result in negative score. Hence, using the triplet verification, we solve both the problem of picking the correct page and rejecting the queries that are not in the repository. By synthesizing Figures 4.13 and 4.14, we draw the curve of false positive and false negative rate at different thresholds in Figure 4.15. As the threshold rises, the false positive rate decreases and the false negative rate increases. An optimal choice of threshold is 55, with the false negative rate = 1% and false positive rate = 0%. We normalize the score of triplets verification to [0..100] and return it to the user as the confidence of the retrieval.

4.6.3 Usability

4.6.3.1 Document Constraint

The MobileRetriever system is based on the token pairs and triplets that reflect geometric relationships between words of text. It is important for a query to contain enough words for successful retrieval. It is not possible to estimate the exact number of words from the area of the captured image because typographical variations result

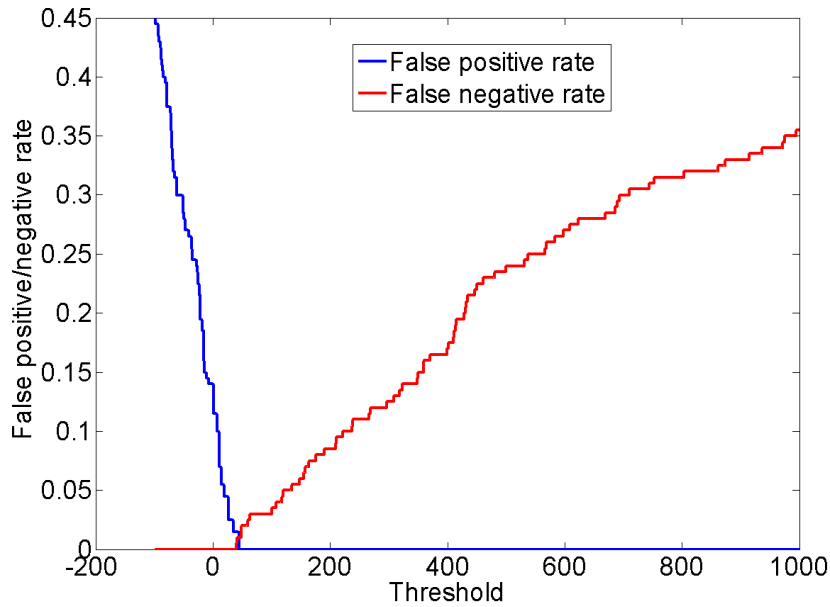


Figure 4.15: False positive and false negative rate at different thresholds

in different word densities. To estimate how many words are required for successful retrieval, we ran a simulation on 100 randomly selected pages from the repository. A random sized rectangle area was cropped from each page as the query. The the rectangle's width was uniformly distributed between 0 and 1/2 page and the aspect ratio of the rectangle set to 4:3, the same as the camera phone image. We recorded how many words were in each query and whether the retrieved page matched the query. A histogram showing both successful retrievals and rejected queries appears in Figure 4.16. According to Figure 4.16, 50 or more words in the query image will lead to a successful retrieval. Based on our statistics, approximately 750 words occur per page in conference proceedings. Considering image distortion and possible recognition errors, a capture of 1/3 page width will lead to a successful retrieval.

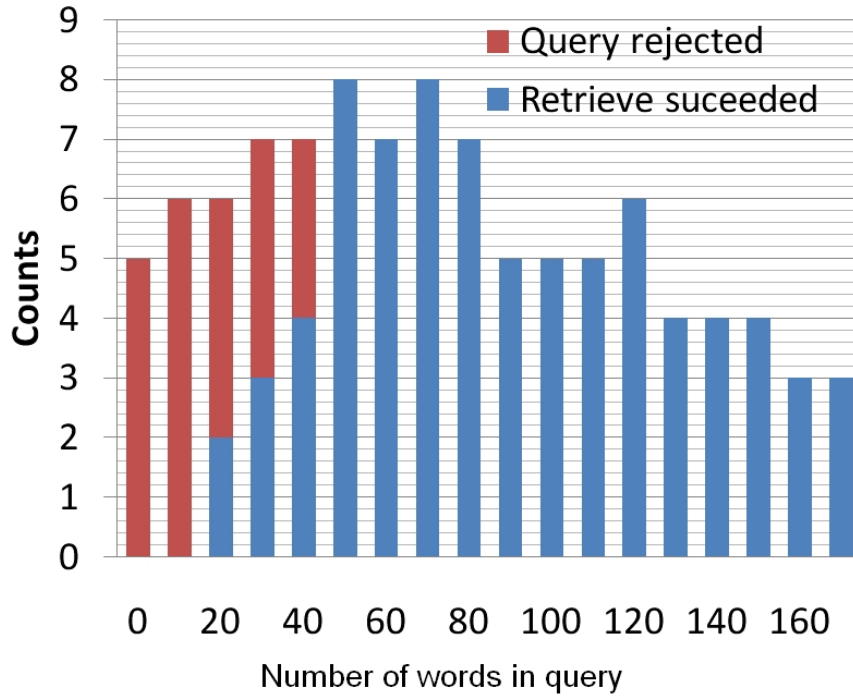


Figure 4.16: Number of words in query vs. success rate

This is not difficult to achieve with a cooperative user.

4.6.3.2 Overall Success Rate

Ultimately the MobileRetriever will be evaluated on overall accuracy. When a user queries the system by taking a picture of a document, what is the chance that the system finds the exact page that matches the query? We asked ten users to choose ten pages from our 100K-page repository and perform a query for each page. We recorded how many pages were correctly retrieved. The only instruction they were given was to capture at least 1/3 page. Among these 100 queries, only four of them are failed due to limited text (Figure 4.17a), perspective distortion (Figure 4.17b) motion blur (Figure 4.17c), or network failure resulting in a success rate of

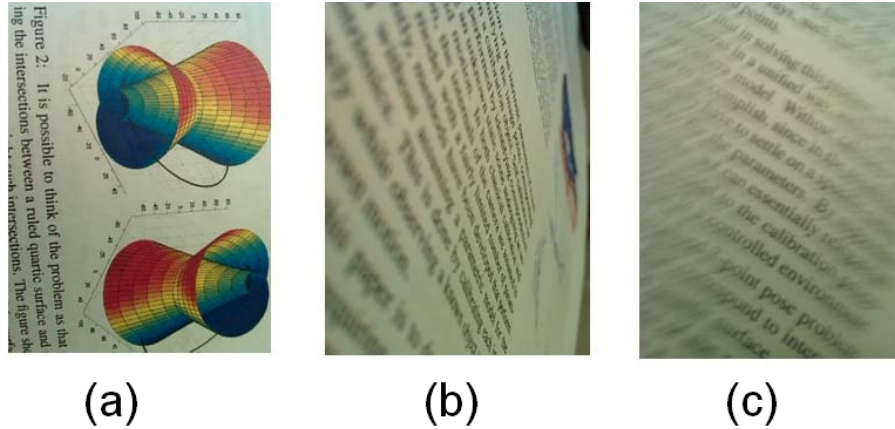


Figure 4.17: Images failed to retrieve

96%.

4.6.3.3 Speed

Unlike a desktop/laptop environment, when users interact with mobile devices, they always expect instant response. The speed of retrieval is a practical issue, so we measure the time spent on each retrieval task. The timer starts when the picture is taken and stops when the result is displayed in the browser. On average each query takes 3.9 seconds. From the server side, we also measure the time spent on each step of retrieval (Figure 4.18). It appears that most of the time is spent on image processing, which is a constant factor. The second most expensive process is sending the image to the server using the 10M bps campus wireless network. The retrieval of the page candidates using token pairs is the least expensive, and the verification is also constant. In general, the time will not grow significantly as the size of the repository grows.

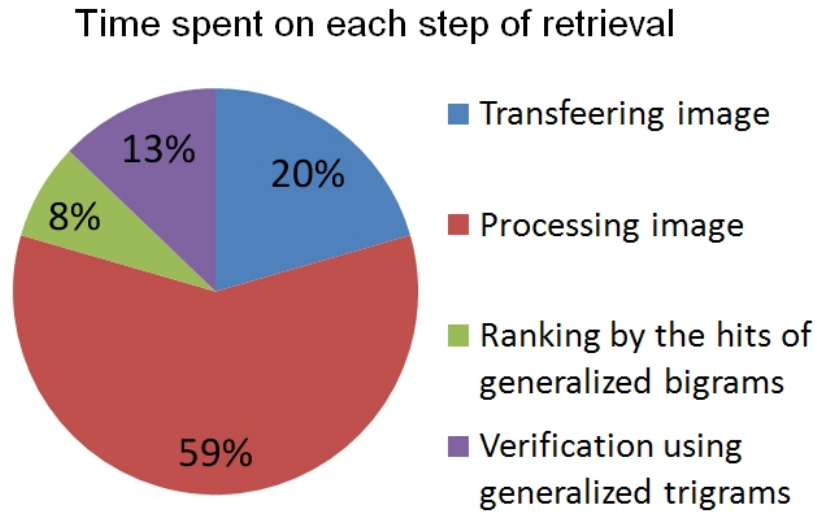


Figure 4.18: Time spent on each step of retrieving

4.7 Summary

In this chapter we have presented a camera based document image retrieval system, the MobileRetriever, which employs the ubiquitous camera phone device as a client. Using the MobileRetriever, the user can retrieve the digital version of a printed document by taking a snapshot of a page, as if every page is given a hyperlink. We proposed token pairs and triplets which retrieve and verify the candidates of retrieval at high speed and accuracy. We evaluated our system on a document repository with 100K pages and showed the system can respond in <4 seconds at a success rate of 96%.

It is worth mentioning that the concept of token pair and triplets is not bound to a given language. Latin scripts have a natural unit of word, but if we apply shape coding to any non-Latin language (such as Chinese or Japanese), and devise

a stable shape code for every character, the Mobile Retriever system can be smoothly extended to these documents. This will provide the next step of our research.

Chapter 5

MobileEye Tools - for Persons with Visual Disabilities

5.1 Motivation

Visual impairments affect a large percentage of population in various ways, including color-deficiency, myopia, low vision, and other more severe visual disabilities [118]. Current estimates suggest approximately ten million blind or visually impaired individuals live in the United States alone. Visual impairment significantly affects quality of life within these populations and most have no effective cure. Devices that provide visual enhancement are in large demand, but are often expensive, bulky, and dedicated to a single task. For example, an optical image enhancer such as a magnifier can provide basic zoom function but cannot enhance the contrast, brightness, color, and other details of an image. Electronic image enhancers (e.g., “Acrobat LCD” and “Sense View Duo”) are powered by digital image processing technology and these programmable devices can be customized for various requirements of the visually impaired users. This device, like others, is a special purpose piece of hardware.

We are exploring a unique opportunity to use a phone’s camera as an electronic eye (a MobileEye) to assist the visually impaired to “see” and understand their surroundings. Compared to optical and electronic image enhancement tools, a camera phone possesses some unique advantages. First, it is a portable hand-held device

already carried by a large number of people, including those with visual disabilities. Second, new functions can be easily programmed and customized in software, with no requirement for extra hardware. Third, the communication capability of these devices opens a wide range of opportunities to provide access to knowledge and computational resources from the internet. By utilizing these advantages of the camera phone, we have built the MobileEye system which consists of three major components:

- A currency reader that helps identify the denominations of U.S. paper currency.
- A color mapper that helps the color deficient user distinguish colors.
- A software magnifier that enhances the detail and contrast of an image to facilitate reading and understanding.

We will focus on the mobile currency reader which was based on the fast pattern recognition technique introduced in Chapter 2. Other functions of the MobileEye system will be covered briefly.

5.2 Existing System and Design Principles

The MobileEye system is designed for persons with visual impairments, it is impractical to require sophisticated operations, which are already difficult for users with normal vision because of the small keypad input [19]. We follow the overall design principle of minimizing user operation, especially with respect to the keypad

hits e.g. the pattern recognizer. A typical mobile pattern recognition system (such as the Kurzweil K1000) requests the user to take a snapshot, then the system attempts to recognize the result. If the image is imperfect, the recognition may fail and the user will have to repeat the process. However, we cannot expect a visually impaired user to perform such tasks, and it is impractical to mandate high quality pictures for recognition. We choose to process the image in real time instead of an “acquisition-process-repeat” loop, which provides a much smoother user experience. Our pattern recognizer operates on the device and processes approximately 10 frames per second, so, the user receives an instant response as the camera acquires new content. This introduces the challenge that we must process the video stream from the camera quickly. We address this problem using a boosted object detector with both high efficiency and accuracy.

5.3 Mobile Pattern Recognition and Currency Reader

Currency, in many countries, is printed on various sizes of paper or with different texture allowing people with visual disabilities can easily distinguish them. In the U.S., however, these user-friendly features are not provided for the visually impaired. The blind community initiated a lawsuit against the Department of the Treasury for discrimination and won the case on May 20, 2008 [119]. Although this situation may be resolved through changes in the engraving and printing process, it will be a highly prolonged and expensive process to replace all currency already in use. Dedicated devices, such as Kurzweil reader [120], have been introduced to help

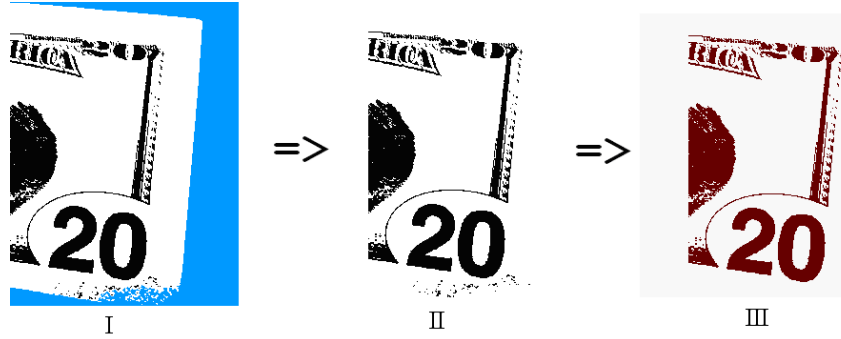


Figure 5.1: Background subtraction and feature area extraction

with reading currency and texts, but they are often bulky and expensive. Novel systems such as iCare[121] have also been developed to assist the visually impaired people with pattern recognition. iCare uses a wearable camera for imaging and a PC for computation. We propose an alternative solution, employ the ubiquitous camera phone[122] to identify different denominations in an instantaneous and inexpensive way. Although we are targeting for reading currency, the designed framework can be extended to recognize other objects as well.

Our currency reader does not require extra or specialized hardware, because our algorithm relies on existing visual features for recognition. It can operate on either side of a bill and the recognition result can be transmitted via the phone speaker or communicated through vibration. Figure 5.1 shows the feature areas used for currency recognition.

5.3.1 Initial Design

To detect and recognize the bill, we first remove irrelevant background. After binarization, black pixels touching the boundary (Figure 5.1-I) of the image are

regarded as background since the bill always has a white or nearly white boundary that separates itself from the background. After background subtraction, some noise (Figure 5.1-II) might still exist. We further refine the location of a bill by running a breadth-first-search(BFS) from the image center to remove the remaining noise. The complexity of this step is linear to the number of pixels in the image. After processing, we know the exact position of the feature area (Figure 5.1-III). We then normalize the area to a rectangle with an aspect ratio of 4:1 for recognition (Figure 5.2).

We collected 1,000 samples of captured images of each side of the most common U.S. bills(\$1,5,10,20,50,100). Each has four potential areas to recognize, two on the front and two on the back. We also collected 10,000 samples of general scenes that are not currency. For each side of a given bill, we use Ada-boost[43] to train a strong classifier from a set of weak classifiers. The weak classifiers must be computationally efficient because hundreds of them must be computed in less than 100 milliseconds.

We define a weak classifier using 32 random pairs of pixels in the image. A random pair of pixels have a relatively stable relationship meaning one pixel is brighter than the other. An example of a random pair is shown in Figure 5.2, where pixel A is brighter than pixel B. The advantage of using pixel pairs is that their relative brightness is not affected by environmental lighting variations. The same relationship may also occur in general scenes, so we select the pairs that appear more frequently in the inliers (currency images) and less frequently in the outliers (non-currency images).

A weak classifier will provide a positive result if more than 2/3 of pairs are

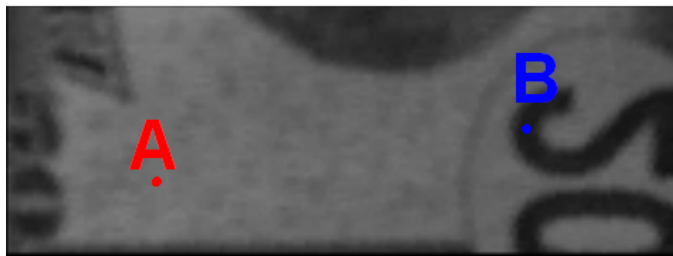


Figure 5.2: Normalized feature area and random pixel pair

satisfied, and negative otherwise. The 10 weak classifiers selected using Ada-boost form a strong classifier that identifies a bill, as long as it appears in the image. To recognize a bill we only need $32 \times 10 = 320$ pair-wise comparisons of pixels. Our system is trained to read \$1,5,10,20,50,100 U.S. bills and can process 10 frames/second on a Windows Mobile (iMate Jamin) phone at a false positive rate $< 10^{-4}$.

5.3.2 Revised Design

Although the initial design of the currency reader satisfies our primary requirements of real time recognition and has a high accuracy, it can be further improved after an experimental study of how its practical use. Users with visual disabilities identified two major disadvantages of the initial design. First, it required the coverage of the entire right hand side of the bill, i.e., the upper right and bottom right side of the bill must be captured at the same time. However, it may be difficult to accomplish such coverage without a great deal of practice. Second, users with visual disabilities liked to fold the bills in different ways to distinguish among denominations, but folding can change the shape of the right hand side of a bill and

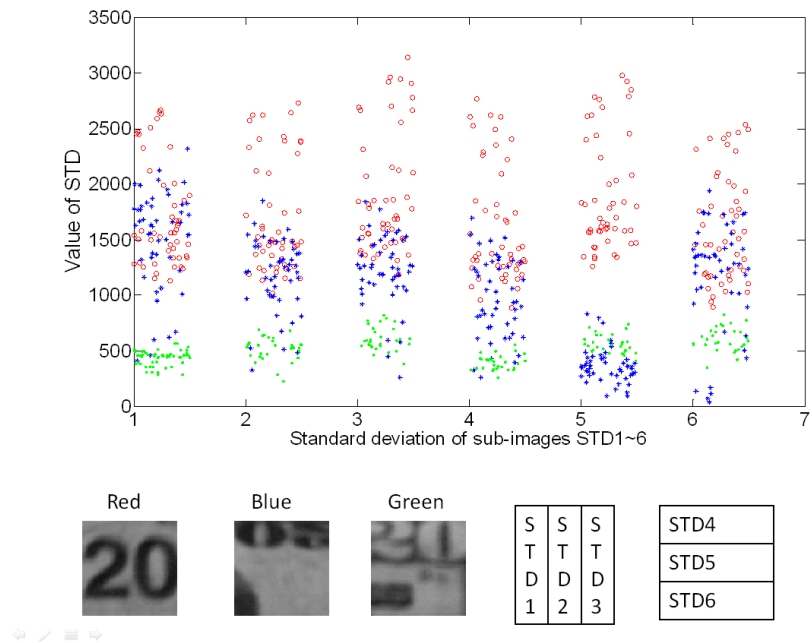


Figure 5.3: Standard deviation of 6 sub-images at 3 corners of a 20 dollar bill

may disturb the recognition.

This suggests the use of a smaller feature area for recognition because it is easier to capture and less likely to be disturbed by folding. We have refined our currency reader to identify a feature area with the number denomination as shown in Figure 5.3. Feature areas are first detected using a fast pre-classifier and then identified using a strong classifier based on random local pixel pairs, as described in Section 2.2.

To meet the requirements of those with visual disabilities, we pay special attention to the details of the user interface. Every operation of the software is guided by a voice message. It requires two key presses to activate the camera to prevent accidental activation. The software automatically exits after being idle for two minutes to save battery power. The user has the option of “forcing” recognition

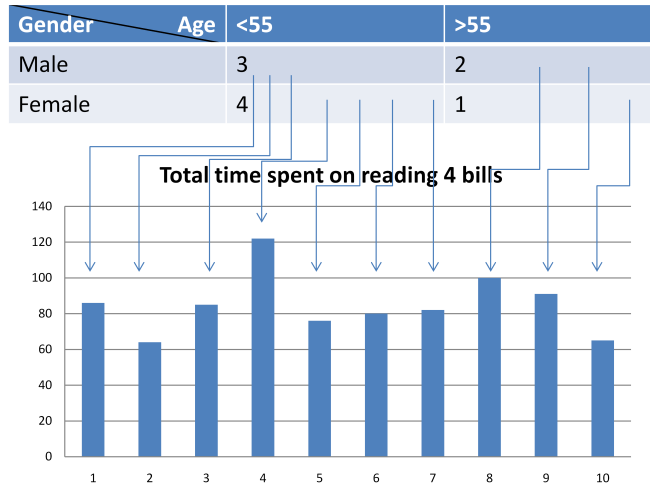


Figure 5.4: User evaluation of mobile currency reader

of a bill by pressing the center button. The software will search for additional scales and positions for the feature area for “forced” recognition.

5.3.3 Evaluation

We have performed user evaluation with the system of the refined design. Ten blind users were asked to identify four bills using the camera phone currency reader. Each user was given a brief two minute introduction on how to use the device’s software, they were asked to continue until all four bills are recognized. The total time (including entering and exiting the program) was recorded to measure the usability of the software. On average, users recognized a bill in 21.3 seconds, as shown in Figure 5.4.

Check iPhone availability at your local Apple Store



Figure 5.5: A color design without considering the color deficient user

5.4 Other MobileEye Tools

5.4.1 Color Mapper

Color deficiency is sometimes referred to as color blindness. In fact, it is very rare that a person is completely “blind” to a color, but deficiency such as red-green color blindness is as high as 8% in Caucasian, 5% in Asian, and 4% in African males. Color deficiency is not a severe visual impairment and is often ignored by visual appearance design. A typical example of poor design relative to red-green color blind is shown in Figure 5.5. Such graphics and figures are misleading to people with color deficiency but widely exist on web sites, slides, and posters. It is not easy to correct them using optical devices, but digital imaging devices can easily adapt. Our color channel mapper processes a camera captured image in real-time by swapping color channels. Depending on the type of color deficiency, we separate one of the two most confusing color channels and exchange it for a third color channel.

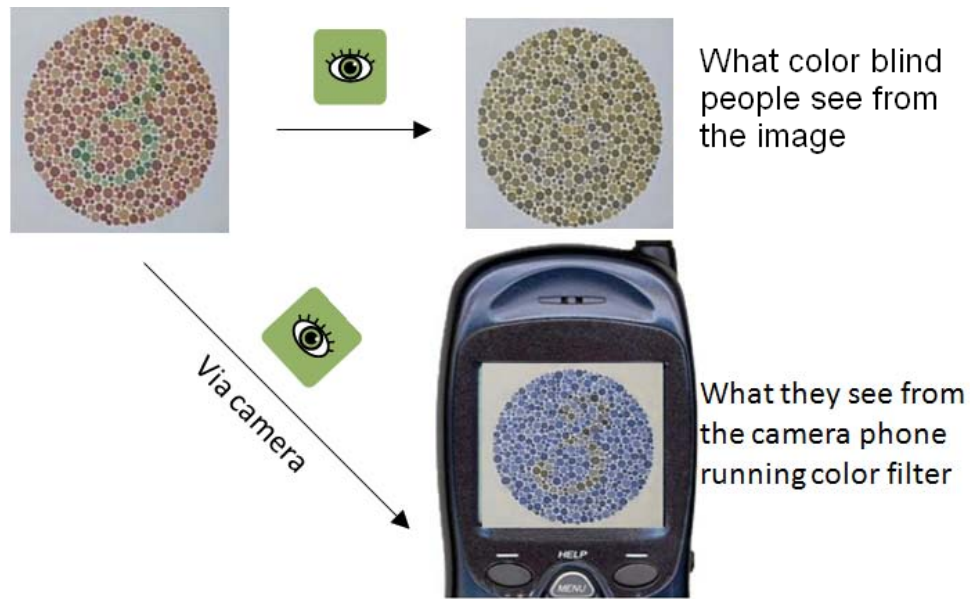


Figure 5.6: Color mapper

For example, the red channel is exchanged with blue for red-green color deficient vision. Figure 5.6 shows how the image is processed.

5.4.2 Software Magnifier

Low vision and myopia are quite common in the elderly population. Most people wear corrective lens or carry a magnifier to help with reading. Our software magnifier performs like a glass magnifier but possesses three distinct advantages. First, the magnification is actually a digital zoom and with not limit. Text can be magnified as long as it fits in the screen. Second, the contrast can be enhanced at the same time, so the text is distinguishable from the background to facilitate reading. Third, by taking a picture of the text, the user may continue reading off line and does not have to hold the paper. Figure 5.7 shows how the software magnifier works.

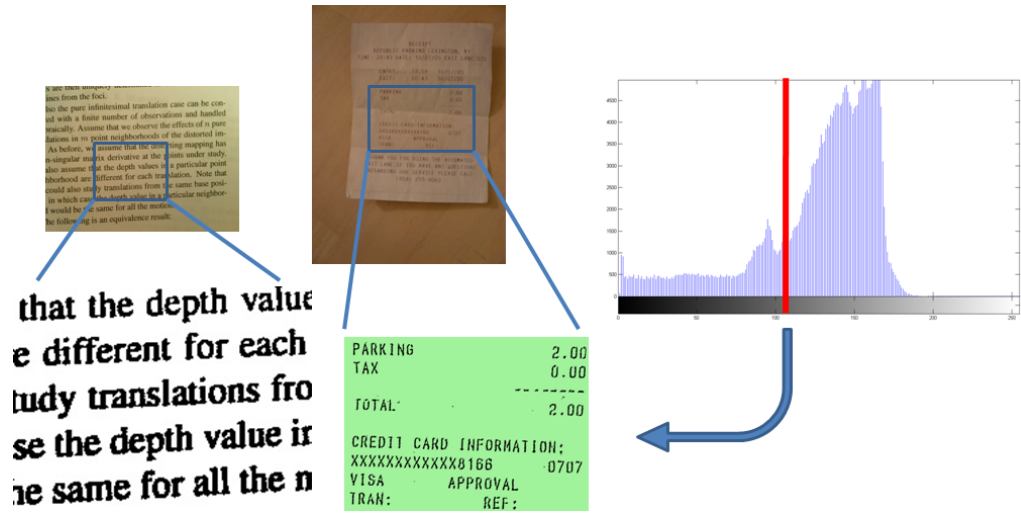


Figure 5.7: Image zoomed by software lens

To perform digital zoom on the image and retain the smoothness, we use bilinear interpolation to enlarge the image. Since floating point support is absent on most mobile architectures (ARM, XScale), we use an integer look-up-table to accelerate and retain the real-time performance.

We enhance the contrast using a two means binarization algorithm that converts the image into black and white. Two means binarization adapts to lighting and can distinguish foreground from background by choosing the threshold at the middle of two peaks from the image’s gray scale histogram. Other types of image enhancement, such as edge enhancement, can also be added programmatically.

5.5 Summary

We have introduced our MobileEye system, which aids individuals with visual impairments to better “see” and understand their surroundings. Our system



Figure 5.8: Currency reader for the visually impaired

requires little effort to operate and can be deployed to a wide varieties of camera enabled mobile devices. We do face one major challenge in reaching the end user. Most handsets together with their software applications are deployed by the wireless service providers. We have put our Color Mapper subsystem online and received 85 downloads and some positive feedback. Our end user with visual disabilities, however, may not have the knowledge and skill to download and install the software themselves. It may require the cooperation of service providers and probable government support to promote the MobileEye system to a larger number of users.

Chapter 6

Conclusion

6.1 Summary of Contributions

In this dissertation we have presented a series of computer vision and image processing techniques designed to operate on mobile devices with low quality imaging. Using these technologies, we have built applications that bridge the gap between the physical and digital worlds. We have developed fast algorithms to perform perspective correction, ego-motion estimation, contrast enhancement, and binary image interpolation. We have eliminated floating point operations in perspective correction and simplified it to the computation of seven cross products, and use a look-up tables to speed interpolation. Our coarser-to-fine image ego-motion estimation runs in real time and can be used to control the cursor and browsing on mobile devices. Besides these techniques, we have answered fundamental questions in two areas: visual communications and document retrieval. We have built a model for color uncertainty of the camera channel. Using mutual information theory, we have estimated the capacity per color pixel from a camera captured frame. We have addressed the problem of rejecting a query that is not in the repository for document image retrieval. We also quantitatively estimated the minimum area required in a query on a 100K page document database.

In general, our research aligns with the concept of “ubiquitous computing”

[123]: people should be able to use computers at any time, anywhere, via any media. We implement this concept on a camera enabled mobile device that allows the user to interact with the physical world via recognition and retrieval techniques.

6.2 Future Work

In our research, we utilize the phone's camera as an alternative input device for bridging the gap between the physical and digital worlds. We can interact with the physical world using the camera. However, increasingly we will be driven to move from the physical world to the mobile world and wish to do so in a seamless way. In our future research we will unify recognition and retrieval on the mobile device. With the ultimate goal of making the physical world "clickable," no matter the source is a barcode, a document, or a TV screen, we will link them to the corresponding digital content. At that time the physical world will have multiple layers of multimedia representation. The camera enabled mobile device will serve as the key to access these layers and numerous novel applications can be built on the mobile devices. We will extend the "MobileRetriever" to non-Latin languages and then to general scenes. The computation is not necessarily hosted on the device, with the help of "cloud computing," we can achieve a transparent access to multimedia representation of the world.

Bibliography

- [1] C. Milanese, A. Liang, N. Mitsuyama, H. Vergne, T. Nguyen, A. Zimmermann, and S. Shen. Forecast: Camera phones, worldwide, 2004-2010. *Gartner Inc. Report No. G00144253*, 2006.
- [2] R. Ballagas, J. Borchers, M. Rohs, and J.G. Sheridan. The smart phone: a ubiquitous input device. *IEEE Pervasive Computing*, 5(1):70, 2006.
- [3] M. Davis, M. Smith, F. Stentiford, A. Bamidele, J. Canny, N. Good, S. King, and R. Janakiraman. Using context and similarity for face and location identification. In *Proceedings of SPIE*, volume 6061, pages 119–127, 2006.
- [4] T. Yeh and T. Darrell. Multimodal question answering for mobile devices. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, pages 405–408. ACM New York, NY, USA, 2008.
- [5] Y. Ijiri, M. Sakuragi, and S. Lao. Security management for mobile devices by face recognition. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (MDM'06)*, page 49, Washington, DC, USA, 2006. IEEE Computer Society.
- [6] X. Chen, J. Yang, J. Zhang, and A. Waibel. Automatic detection of signs with affine transformation. In *WACV '02: Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, page 32, Washington, DC, USA, 2002. IEEE Computer Society.
- [7] A pda-based sign translator. In *ICMI '02: Proceedings of the fourth IEEE International Conference on Multimodal Interfaces*, page 217, Washington, DC, USA, 2002. IEEE Computer Society.
- [8] J. Zhang, X. Chen, J. Yang, and A. Waibel. A PDA-based Sign Translator. *Proceedings of the International Conference on Multimodal Interfaces*, 2002.
- [9] C. Seifert, L. Paletta, A. Jeitler, E. Hodl, J.P. Andreu, P. Luley, and A. Almer. Visual object detection for mobile road sign inventory. *Lecture Notes in Computer Science*, pages 491–495, 2004.
- [10] K. S. Bae, K. K. Kim, Y. G. Chung, and W. P. Yu. Character recognition system for cellular phone with camera. In *COMPSAC '05: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC'05) Volume 1*, pages 539–544, Washington, DC, USA, 2005. IEEE Computer Society.
- [11] M. Koga, R. Mine, T. Kameyama, T. Takahashi, M. Yamazaki, and T. Yamaguchi. Camera-based kanji ocr for mobile-phones: Practical issues. In

- ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pages 635–639, Washington, DC, USA, 2005. IEEE Computer Society.
- [12] E.D. Haritaoglu, I. Haritaoglu, and P.R. Inc. Document recognition and translation with handheld mobile devices. *Proceedings 2005 Symposium on Document Image Understanding Technology*, 2005.
- [13] M. Jia, X. Fan, X. Xie, M. Li, and W.Y. Ma. Photo-to-Search: Using camera phones to inquire of the surrounding world. *Seventh International Conference on Mobile Data Management*, pages 46–46, 2006.
- [14] R. Yeh, C. Liao, S. Klemmer, F. Guimbretièrre, B. Lee, B. Kakaradov, J. Stamberger, and A. Paepcke. ButterflyNet: a mobile capture and access system for field biology research. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 571–580. ACM New York, NY, USA, 2006.
- [15] T. Yeh and K. Tollmar. A picture is worth a thousand keywords: Image-based object search on a mobile platform. In *Conference on Human Factors in Computing Systems*, pages 2025–2028. ACM New York, NY, USA, 2005.
- [16] X. Anguera, N. Oliver, and M. Cherubini. Multimodal and mobile personal image retrieval: A user study. In *SIGIR 2008 Workshop on Mobile Information Retrieval*, page 17.
- [17] X. Xie, L. Lu, M. Jia, H. Li, F. Seide, and W.Y. Ma. Mobile search with multimodal queries. *Proceedings of the IEEE*, 96(4):589–601, 2008.
- [18] C. Gurrin, G.J.F. Jones, H. Lee, N. O’Hare, A.F. Smeaton, and N. Murphy. Mobile access to personal digital photograph archives. In *Proceedings of the seventh international conference on human computer interaction with mobile devices and services*, pages 311–314. ACM New York, NY, USA, 2005.
- [19] A.K. Karlson, B.B. Bederson, and J.L. Contreras-Vidal. Understanding one handed use of mobile devices. *Handbook of Research on User Interface Design and Evaluation for Mobile Technology, Idea Group Reference*, 2007.
- [20] J. Coughlan, R. Manduchi, and H. Shen. Cell phone-based wayfinding for the visually impaired. In *First International Workshop on Mobile Vision, in conjunction with ECCV*, 2006.
- [21] V. Ivanchenko, J. Coughlan, and H. Shen. Crosswatch: A camera phone system for orienting visually impaired pedestrians at traffic intersections. In *11th International Conference on Computers Helping People with Special Needs (ICCHP2008), Linz, Austria, July*. Springer, 2008.

- [22] J. Coughlan and R. Manduchi. Color Targets: Fiducials to help visually impaired people find their way by camera phone. *EURASIP Journal on Image and Video Processing*, 2007.
- [23] K.Y. Chan, R. Manduchi, and J. Coughlan. Accessible spaces: navigating through a marked environment with a camera phone. In *Proceedings of the Ninth International ACM SIGACCESS Conference on Computers and Accessibility*, pages 229–230. ACM Press New York, NY, USA, 2007.
- [24] S.C. Park, M.K. Park, and M.G. Kang. Super-resolution image reconstruction: a technical overview. *Signal Processing Magazine, IEEE*, 20(3):21–36, 2003.
- [25] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(9):1167–1183, 2002.
- [26] B. Erol, E. Antunez, and J. J. Hull. Hotpaper: Multimedia interaction with paper using mobile phones. In *ACM International Conference on Multimedia*, pages 399–408, 2008.
- [27] M. Sorel and J. Flusser. Blind restoration of images blurred by complex camera motion and simultaneous recovery of 3d scene structure. *Proceedings of the Fifth IEEE International Symposium on Signal Processing and Information Technology*, pages 737–742, Dec. 2005.
- [28] E.M. Or and D. Pundik. Hand motion and image stabilization in hand-held devices. *IEEE Transactions on Consumer Electronics*, 53(4):1508–1512, 2007.
- [29] S. Prasad. Statistical-information-based performance criteria for Richardson-Lucy image deblurring. *Journal of the Optical Society of America*, 19(7):1286–1296, 2002.
- [30] C.H. Chu, D.N. Yang, and M.S. Chen. Image stabilization for 2D barcode in handheld devices. In *Proceedings of the 15th International Conference on Multimedia*, pages 697–706. ACM Press New York, NY, USA, 2007.
- [31] A. Oulasvirta, S. Tamminen, V. Roto, and J. Kuorelahti. Interaction in four-second bursts: the fragmented nature of attentional resources in mobile HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 919–928. ACM New York, NY, USA, 2005.
- [32] H.B.L. Duh, G.C.B. Tan, and V.H. Chen. Usability evaluation for mobile device: a comparison of laboratory and field tests. In *Proceedings of the eighth Conference on Human-computer Interaction with Mobile Devices and Services*, pages 181–186. ACM New York, NY, USA, 2006.
- [33] A. Criminisi, I. Reid, and A. Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999.

- [34] A. Bors, I. Pitas, W. Puech, and J. Chassery. Perspective distortion analysis for mosaicing images painted on cylindrical surfaces. In *ICASSP '97: Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97) - Volume 4*, page 3049, Washington, DC, USA, 1997. IEEE Computer Society.
- [35] R.M. Haralick. Determining camera parameters from the perspective projection of a rectangle. *Pattern Recogn.*, 22(3):225–230.
- [36] T. Q. Phong, R. Horaud, A. Yassine, and D. T. Pham. Optimal estimation of object pose from a single perspective view. In *Fourth IEEE International Conference on Computer Vision*, pages 534–539, 1993.
- [37] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [38] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2, 2003.
- [39] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [40] D.F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990.
- [41] P. Viola and M. Jones. Robust real-time face detection. *International Journal on Computer Vision*, 57(2):137–154, 2004.
- [42] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *Proceedings of Ninth European Conference on Computer Vision, Graz, May 7*, 13:404–417, 2006.
- [43] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [44] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.
- [45] M. Özuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8, 2007.
- [46] J. Wang, S. Zhai, and J. Canny. Camera phone based motion sensing: interaction techniques, applications and performance study. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, pages 101–110. ACM New York, NY, USA, 2006.

- [47] A. Haro, K. Mori, V. Setlur, and T. Capin. Mobile camera-based adaptive viewing. In *Proceedings of the Fourth International Conference on Mobile and Ubiquitous Multimedia*, pages 78–83. ACM New York, NY, USA, 2005.
- [48] D. Majoe, S. Schubiger, A. Clay, and S.M. Arisona. SQEAK: A mobile multi platform phone and networks gesture sensor. In *Pervasive Computing and Applications, 2007. ICPCA 2007. 2nd International Conference on*, pages 699–704, 2007.
- [49] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, volume 588, pages 237–252, 1992.
- [50] D. Capel and A. Zisserman. Automated mosaicing with super-resolution zoom. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 885–891, 1998.
- [51] J. Hannuksela, P. Sangi, and J. Heikkila. A vision-based approach for controlling user interfaces of mobile devices. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops - Volume 03*, 2005.
- [52] X. Liu, D. Doermann, and H. Li. Fast camera motion estimation for hand-held devices and applications. In *MUM '05: Proceedings of the fourth international conference on Mobile and ubiquitous multimedia*, pages 103–108, New York, NY, USA, 2005. ACM Press.
- [53] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.
- [54] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, 1, 2000.
- [55] S. K. Chang, B. Perry, and A. Rosenfeld. *Content-Based Multimedia Information Access*. Kluwer Press, 1999.
- [56] Y. Rui, T. S. Huang, and S. Chang. Image retrieval: current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, March 1999.
- [57] R. Datta, J. Li, and J.Z. Wang. Content-based image retrieval: approaches and trends of the new age. In *Proceedings of the Seventh ACM SIGMM International Won Multimedia Information Retrieval*, pages 253–262. ACM Press New York, NY, USA, 2005.
- [58] J.S. Hare and P.H. Lewis. Content-based image retrieval using a mobile device as a novel interface. *Storage and Retrieval Methods and Applications for Multimedia 2005. Proceedings of the SPIE*, 5682:64–75, 2004.

- [59] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, page 50, 1988.
- [60] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
- [61] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [62] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous Pose and Correspondence Determination. *International Journal of Computer Vision*, 59(3):259–284, 2004.
- [63] H. Kato and K.T. Tan. Pervasive 2D Barcodes for Camera Phone Applications. *IEEE Pervasive Computing*, pages 76–85, 2007.
- [64] E. Ohbuchi, H. Hanaizumi, and L. A. Hock. Barcode readers using the camera device in mobile phones. In *CW '04: Proceedings of the 2004 International Conference on Cyberworlds (CW'04)*, pages 260–265, Washington, DC, USA, 2004. IEEE Computer Society.
- [65] A. Otero. A robust software barcode reader using the hough transform. In *ICIIS '99: Proceedings of the 1999 International Conference on Information Intelligence and Systems*, page 313, Washington, DC, USA, 1999. IEEE Computer Society.
- [66] S. Ando and H. Hontani. Automatic visual searching and reading of barcodes in 3d scene. In *IEEE International Conference on Vehicle Electronics*, pages 49–54, 2001.
- [67] H. I. Hahn and J. K. Joung. Implementation of algorithm to decode two-dimensional barcode pdf-417. In *Sixth International Conference on Signal Processing*, volume 2, pages 1791–1794 vol.2, 2002.
- [68] E. Ouaviani, A. Pavan, M. Bottazzi, E. Brunelli, F. Caselli, and M. Guerrero. A common image processing framework for 2d barcode reading. volume 2, pages 652–655, 1999.
- [69] M. Rohs. Real-world interaction with camera phones. *Ubiquitous Computing Systems: Second International Symposium, UCS 2004, Tokyo, Japan, November 8-9, 2004: Revised Selected Papers*, 2005.
- [70] R. Ballagas, M. Rohs, J.G. Sheridan, and J. Borchers. Sweep and point and shoot: phonecam-based interactions for large public displays. In *ACM Conference on Human Factors in Computing Systems*, volume 2, pages 1200–1203, 2005.

- [71] M. Rohs. Visual code widgets for marker-based interaction. *25th IEEE International Conference on Distributed Computing Systems Workshops*, pages 506–513, 2005.
- [72] D. Scott, R. Sharp, A. Madhavapeddy, and E. Upton. Using visual tags to bypass Bluetooth device discovery. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(1):41–53, 2005.
- [73] E. Toye, R. Sharp, A. Madhavapeddy, D. Scott, E. Upton, and A. Blackwell. Interacting with mobile services: an evaluation of camera-phones and visual tags. *Personal and Ubiquitous Computing*, 11(2):97–106, 2007.
- [74] Panu Vartiainen, Suresh Chande, and Kimmo Rämö. Mobile visual interaction: enhancing local communication and collaboration with visual interactions. In *MUM '06: Proceedings of the fifth International Conference on Mobile and Ubiquitous Multimedia*, page 4, New York, NY, USA, 2006. ACM.
- [75] G.A. Fowler. QR codes: In Japan, Billboards take code-crazy ads to new heights. *Wall Street Journal*, 10:2005, 2005.
- [76] C. Cheong, D. Kim, and T. Han. Usability evaluation of designed image code interface for mobile computing environment. *Lecture Notes In Computer Science*, 4551:241, 2007.
- [77] T. Han, C. Cheong, N. Lee, and E. Shin. Machine readable code and method and device of encoding and decoding the same, March 5, 2003. EP Patent 1,287,483.
- [78] X. Liu, H. Li, and D. Doermann. Imaging as an Alternative Data Channel for Camera Phones. In *ACM International Conference Proceeding Series; Proceedings of the fifth International Conference on Mobile and Ubiquitous Multimedia*, page No. 5, December 2006. best paper.
- [79] X. Liu, D. Doermann, and H. Li. VCode pervasive data transfer using video barcode. *IEEE Transactions on Multimedia*, 10(3):361–371, April 2008.
- [80] X. Liu, D. Doermann, and H. Li. A Camera-based Mobile Data Channel: Capacity and Analysis. In *ACM International Conference on Multimedia*, pages 359–368, 2008.
- [81] R.O. Duda and P.E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.
- [82] S.B. Wicker and V.K. Bhargava, editors. *Reed-Solomon Codes and Their Applications*. John Wiley & Sons, Inc., New York, NY, USA, 1999.
- [83] P.B. Fellgett and E.H. Linfoot. On the Assessment of Optical Images. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 247(931):369–407, 1955.

- [84] H.R. Sheikh and A.C. Bovik. Image information and visual quality. *Image Processing, IEEE Transactions on*, 15(2):430–444, 2006.
- [85] M. Barni, F. Bartolini, A. De Rosa, and A. Piva. Capacity of the watermark channel: How many bits can be hidden within a digital image? *SPIE Proceedings*, 3657:437–448, 1999.
- [86] T. Nakamura, S. Yamamoto, R. Kitahara, A. Katayama, T. Yasuno, and N. Sonehara. A fast, robust watermark detection scheme for videos captured on camera phones. *Multimedia and Expo, 2007 IEEE International Conference on*, pages 316–319, 2007.
- [87] C.M. Jarque and A.K. Bera. A test for normality of observations and regression residuals. *International Statistical Review/Revue Internationale de Statistique*, 55(2):163–172, 1987.
- [88] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, 43:85–103, 1972.
- [89] M. E. Wright J. H. Tappan and F. E. Sistler. Error sources in a digital image analysis system. *Computers and Electronics in Agriculture*, 2:109–118, 1987.
- [90] Y.C. Chang and J.F. Reid. RGB calibration for color image analysis in machine vision. *IEEE Transactions on Image Processing*, 5(10):1414–1422, 1996.
- [91] J. Korhonen, O. Aalto, A. Gurtov, and H. Lamanen. Measured performance of GSM, HSCSD and GPRS. *IEEE International Conference on Communications*, 5, 2001.
- [92] D. Doermann, E. Rivlin, and A. Rosenfeld. The function of documents. *International Journal of Computer Vision*, 16:799–814, 1998.
- [93] J. Graham, B. Erol, J.J. Hull, and D.S. Lee. The video paper multimedia playback system. In *Proceedings of the 11th ACM International Conference on Multimedia*, pages 94–95. ACM New York, NY, USA, 2003.
- [94] J.J. Hull, B. Erol, J. Graham, Q. Ke, H. Kishi, J. Moraleda, and D.G. Van Olst. Paper-based augmented reality. *17th International Conference on Artificial Reality and Telexistence*, pages 205–209, 2007.
- [95] P. Wellner. Interacting with paper on the DigitalDesk. *Communications of the ACM*, 36(7):87–96, 1993.
- [96] J.J. Hull. Document image matching and retrieval with multiple distortion-invariant descriptors. *Document Analysis Systems*, pages 379–396, 1995.
- [97] T. Arai, D. Aust, and S.E. Hudson. PaperLink: a technique for hyperlinking from real paper to electronic content. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 327–334, 1997.

- [98] S.R. Klemmer, J. Graham, G.J. Wolff, and J.A. Landay. Books with voices: paper transcripts as a physical interface to oral histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 89–96. ACM Press New York, NY, USA, 2003.
- [99] D. Schmalstieg and D. Wagner. Experiences with Handheld Augmented Reality. In *The Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–13, 2007.
- [100] X. Anguera and N. Oliver. MAMI: multimodal annotations on a camera phone. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 379–382. ACM New York, NY, USA, 2008.
- [101] A. Hwang, S. Ahern, S. King, M. Naaman, R. Nair, and J. Yang. Zurfer: mobile multimedia access in spatial, social and topical context. In *Proceedings of the 15th ACM International Conference on Multimedia*, pages 557–560. ACM Press New York, NY, USA, 2007.
- [102] Y. Zhou, X. Fan, X. Xie, Y. Gong, and W.Y. Ma. Inquiring of the sights from the web via camera mobiles. In *2006 IEEE International Conference on Multimedia and Expo*, pages 661–664, 2006.
- [103] D. Doermann. The indexing and retrieval of document images: a survey. *Computer Vision and Image Understanding*, 70(3):287–298, 1998.
- [104] J.F. Cullen, J.J. Hull, and P.E. Hart. Document image database retrieval and browsing using texture analysis. *Proceedings of the fourth International Conference on Document Analysis and Recognition*, pages 718–721, 1997.
- [105] Y. Lu and C.L. Tan. Information retrieval in document image databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1398–1410, 2004.
- [106] T. Kameshiro, T. Hirano, Y. Okada, and F. Yoda. A document image retrieval method tolerating recognition and segmentation errors of OCR using shape-feature and multiple candidates. *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pages 681–684, 1999.
- [107] P. Herrmann and G. Schlageter. Retrieval of document images using layout knowledge. *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 537–540, 1993.
- [108] S.N. Srihari, S. Shetty, S. Chen, H. Srinivasan, C. Huang, G. Agam, and O. Frieder. Document Image Retrieval using Signatures as Queries. *Proceedings of the Second International Conference on Document Image Analysis for Libraries (DIAL'06)-Volume 00*, pages 198–203, 2006.

- [109] D. Doermann, E. Rivlin, and I. Weiss. Applying algebraic and differential invariants for logo recognition. *Machine Vision and Applications*, 9(2):73–86, 1996.
- [110] T. Nakai, K. Kise, and M. Iwamura. Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. *Lecture Notes in Computer Science (Seventh International Workshop DAS2006)*, 3872:541–552, 2006.
- [111] Q. Liu, P. McEvoy, and C.J. Lai. Mobile camera supported document redirection. In *Proceedings of the 14th ACM international Conference on Multimedia*, pages 791–792, New York, NY, USA, 2006. ACM.
- [112] X. Liu and D. Doermann. Mobileretriever - finding document with a snapshot. *International Workshop on Camera-Based Document Analysis and Recognition*, pages 29–34, September 2007.
- [113] T. Nakai, K. Kise, and M. Iwamura. Hashing with local combinations of feature points and its application to camera-based document image retrieval. *International Workshop on Camera-Based Document Analysis and Recognition*, pages 87–94, 2005.
- [114] C.L. Tan, W. Huang, Z. Yu, and Y. Xu. Imaged document text retrieval without OCR. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(6):838–844, 2002.
- [115] A.F. Smeaton and A.L. Spitz. Using Character Shape Coding for Information Retrieval. *Proceedings of the fourth International Conference on Document Analysis and Recognition*, page 974, 1997.
- [116] P.F. Brown, R.L. Mercer, V.J. Della Pietra, and J.C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.
- [117] W. Niblack. *An Introduction to Digital Image Processing*. 1990.
- [118] R.W. Massof, C.T. Baker Hsu, and F.H. Barnett. Visual disability variables. I,II: the importance and difficulty of activity goals for a sample of low-vision patients. *Archives of Physical Medicine and Rehabilitation*, 86(5):946–953, 2005.
- [119] <http://www.acb.org/press-releases/final-edit-paper-currency-ruling-080520.html>.
- [120] <http://www.knfbreader.com>.
- [121] S. Krishna, G. Little, J. Black, and S. Panchanathan. A wearable face recognition system for individuals with visual impairments. *Proceedings of the seventh International ACM SIGACCESS Conference on Computers and Accessibility*, pages 106–113, 2005.

- [122] T. Kindberg, M. Spasojevic, R. Fleck, and A. Sellen. The ubiquitous camera: An in-depth study of camera phone use. *IEEE Pervasive Computing*, 4(2):42–50, 2005.
- [123] M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, 1993.