5-2013

Eyes-Free Vision-Based Scanning of Aligned Barcodes and Information Extraction from Aligned Nutrition Tables

Aliasgar Kutiyanawala
Utah State University

EYES-FREE VISION-BASED SCANNING OF ALIGNED BARCODES AND INFORMATION EXTRACTION FROM ALIGNED NUTRITION TABLES

by

Aliasgar Kutiyanawala

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Computer Science

Approved:

Dr. Vladimir A. Kulyukin Major Professor	Dr. Dan Watson Committee Member
Dr. Donald H. Cooley Committee Member	Dr. Nicholas Flann Committee Member
Dr. Daniel C. Coster Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2012

ABSTRACT

Eyes-Free Vision-Based Scanning of Aligned Barcodes and Information Extraction from Aligned Nutrition Tables

by

Aliasgar Kutiyanawala, Doctor of Philosophy

Utah State University, 2012

Major Professor: Dr. Vladimir A. Kulyukin
Department: Computer Science

Visually impaired (VI) individuals struggle with grocery shopping and have to rely on either friends, family or grocery store associates for shopping. ShopMobile 2 is a proof-of-concept system that allows VI shoppers to shop independently in a grocery store using only their smartphone. Unlike other assistive shopping systems that use dedicated hardware, this system is a software only solution that relies on fast computer vision algorithms. It consists of three modules - an eyes free barcode scanner, an optical character recognition (OCR) module, and a tele-assistance module. The eyes-free barcode scanner allows VI shoppers to locate and retrieve products by scanning barcodes on shelves and on products. The OCR module allows shoppers to read nutrition facts on products and the tele-assistance module allows them to obtain help from sighted individuals at remote locations. This dissertation discusses, provides implementations of, and presents laboratory and real-world experiments related to all three modules.

(148 pages)

PUBLIC ABSTRACT

Aliasgar Kutiyanawala

Independent grocery shopping is one of the biggest challenges faced by visually impaired (VI) individuals. VI individuals may be able to get to a store on their own by using public transportation or by walking but are unable to shop there independently. Some of the problems that they face after getting to the store include long wait times to get an employee to assist them or getting a store employee who is not familiar with the store layout, gets irritated with long searches, or does not possess the required English skills. These problems ultimately result in VI shoppers having to abandon independent shopping altogether and instead rely on friends and family for their shopping needs.

Assisitive shopping systems can help VI individuals shop independently by helping them in areas such as store navigation, product retrieval, etc. Many assistive shopping systems have been developed but they usually rely on dedicated hardware or instrumenting the store with RFID tags, etc. Unlike these systems, ShopMobile 2 is a software-only solution that only uses fast computer vision algorithms running on a smartphone.

ShopMobile 2 consists of three modules - an eyes free barcode scanner, an optical character recognition (OCR) module, and a tele-assistance module. The eyes-free barcode scanner allows VI shoppers to locate and retrieve products by scanning barcodes on shelves and on products. The OCR module allows shoppers to read nutrition facts on products and the tele-assistance module allows them to obtain help from sighted individuals at remote locations. This dissertation discusses, provides implementations of, and presents laboratory and real-world experiments related to all three modules.

To my parents and my lovely wife

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Vladimir Kulyukin for mentoring me and for inspring me. I would also like to thank my committee members, Dr. Dan Watson, Dr. Donald Cooley, Dr. Daniel Coster, and Dr. Nicholas Flann for their feedback. A special thanks to Mr. Sachin Pavithran who readily participated in a lot of experiments. This dissertation would not be where it is today without his insightful feedback. I would also like to thank all the other participants for their valuable feedback. I would like to thank my friends and labmates John Nicholson, Chaitanya Gharpure and Bharat Chatla for helping me. A special thanks to our department secretaries - Genie and Vicki for making everything so much easier. Last but not the least, I would like to thank my wife who has been so patient with me during the entire process and for being there for me.

Aliasgar Kutiyanawala

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

According to the National Federation for the Blind, there are approximately 1.3 million legally blind individuals in the U.S. and this number is projected to grow to 2.4 million by the year 2030 [1]. Independent grocery shopping is one of the biggest challenges faced by visually impaired (VI) individuals. To understand why independent grocery shopping is difficult, consider that a typical modern supermarket has a median area of 46000 square feet and stocks an average of 38,718 products [2]. VI individuals may be able to get to a store on their own by using public transportation or by walking but are unable to shop there independently. Some of the problems that they face after getting to the store include long wait times to get an employee to assist them or getting a store employee who is not familiar with the store layout, gets irritated with long searches, or does not possess the required English skills. These problems ultimately result in VI shoppers having to abandon independent shopping altogether and instead rely on friends and family for their shopping needs. Peapod [3] and other similar home shopping systems provide grocery shopping alternatives but are not universally available and require shoppers to schedule and wait for deliveries.

Assisitive shopping systems can help VI individuals shop independently by helping them in areas such as store navigation, product retrieval, etc. Many such systems like RoboCart [4–6] , ShopTalk [7–9], GroZi [10, 11], etc. have been developed that aim to help VI individuals shop independently. Chapter 2 presents a detailed description and analysis of these systems. Most of these systems have demonstrated that VI shoppers can use them to shop independently in a grocery store. However, these systems tend to introduce

some problems of their own, chiefly the need for specialized hardware and instrumenting the environment. Systems can employ specialized hardware to solve a problem easily and efficiently. However, this also means that VI shoppers have to purchase, learn to use and maintain this hardware. Most VI individuals already own and operate devices like note takers, wayfinding devices, etc. and requiring them to purchase and maintain additional hardware poses a financial as well as ergonomical burden on them. The second problem is instrumenting the environment with devices such as RFID tags, special barcodes, etc. Supermarkets are wary of installing hardware in their store and raise questions about the responsibility of maintaining the hardware and whether this hardware would be disruptive to normal shopper traffic. Thus, it is best if accessible shopping systems can work with existing store infrastructure and do not make demands of installing additional hardware in the store.

ShopMobile 2 aims to address these two issues with current accessible shopping systems. It aims to prove that computer vision is sufficient for independent blind shopping thus eliminating the need for specialized hardware and instrumenting the environment. The remainder of this chapter is organized as follows: Section 1.2 discusses some of the previous assistive shopping systems developed at the Computer Science Assistive Technology Laboratory (CSATL) of Utah State University and Section 1.3 discusses the insights from these systems. Section 1.4 introduces the ShopMobile 2 system and Section 1.5 discusses a typical use case for the ShopMobile 2 system. Finally, Section 1.6 states the research goals of this dissertation.

1.2 CSATL Assistive Shopping Systems

Three assistive technology systems were developed at the Computer Science Assistive Technology Laboratory (CSATL) of Utah State University. We will briefly describe these systems here. The interested reader is referred to Chapter 2 for a detailed description of these systems.

RoboCart [4–6] was the first assistive shopping system that was developed at CSATL. This system was built using a Pioneer 2DX from ActivMedia Robotics robot as a base

on which a polyvinyl chloride (PVC) structure carrying a shopping basket was securely attached. The sensor suite of the robot included odometers, SONAR sensors, a SICK laser range finder and a RFID reader along with its antenna. The robot also carried a laptop for on-board computations and a numeric keypad for containing input from the user.

The robot used a combination of odometer and laser range finder readings to localize the user in the store and for wayfinding. Passive RIFD tags were placed at various locations in the store and these tags provided additional localization inputs and were used to correct global Markov localizations errors.

In this system, shoppers would use the numeric keypad, which was designed to emulate a cell phone keypad, to browse and choose products. They would then follow the robot by grabbing its PVC handle to arrive in vicinity of the chosen products. After arriving in the vicinity of products, shoppers would use a IT4600 SR wireless barcode scanner to retrieve products from the shelves.

ShopTalk [7–9] was the second assistive shopping system developed at CSATL. ShopTalk was based on the following conceptual formula: *independent blind shopping = verbal route instructions + shelf barcode scans.* It did away with the robot used in RoboCart and instead relied only on barcode scanning to localize the user in the store and to provide directions to target products. Most stores place small Modified Plessey (MSI) type barcodes on shelves beneath products for inventory control purposes. Each MSI barcode can be associated with positional information (aisle number, side of aisle, shelf section number, shelf number and relative position of product within each shelf) and with the product details (name and UPC barcode number of the product). Thus, each MSI barcode now forms a topological landmark in the store and a map of these barcodes can be used to create a data structure known as the *Barcode Connectivity Matrix* (BCM).

In terms of hardware, this system utilized a wireless hand-held barcode scanner connected to an ultraportable OQO computer, which the user carried in a small CamelBak backpack. The system used a numeric keypad mounted on the user's shoulder to receive input and provided output through headphones.

To use this system, the shopper selected products using the keypad and the system provided verbal directions to the target products. If the shopper was lost, she simply scanned any MSI barcode in the store and the system would retrieve her position using the BCM. A new set of directions would then be issued to the user based on her current position and this process would continue until she found the target product and scanned its MSI barcode. She could now reach over and pick the target product and verify that she had picked the correct product by scanning the UPC barcode on the product itself.

ShopMobile 1 [12] was the third assistive shopping system for VI shoppers developed at CSATL. ShopMobile1 aimed to reduce the hardware complexity of ShopTalk by utilizing only a Nokia smartphone and a portable pen-based Baracoda barcode reader. ShopMobile1 used the BCM concept of ShopTalk but users would scan barcodes using the Baracoda wireless barcode reader and would interact with the system using the smartphone.

1.3 Insights from the CSATL Assistive Shopping Systems

In Chapter 2, we propose seven design requirements for assistive shopping systems. These are: mobile product selection **(DR1)**, store navigation **(DR2)**, product search **(DR3)**, product identification **(DR4)**, utilization of existing devices **(DR5)** and minimal environmental adjustments **(DR6)**. A good assistive shopping system must meet all of these design requirements. Let us revisit the assistive shopping systems discussed previously in light of these design requirements.

RoboCart met design requirements **DR1** (mobile product selection), **DR2** (store navigation), **DR3** (product search) and **DR4** (product identification) either completely or partially. However, it did not meet design requirements **DR5** (utilization of existing devices) and **DR6** (minimal environmental adjustments). The biggest drawback with RoboCart was the robot. Even though it was perfectly safe, it was seen as a potential risk to customers. The robot was also expensive and suffered from reliability issues. The other problem with RoboCart was instrumenting the store with RFID tags. Even though the tags were inexpensive and required no maintenance, they had to be placed at strategic locations within the store in order for the robot to function.

ShopTalk aimed to resolve the shortcomings of RoboCart by eliminating the robot and instead relying entirely on barcode scanning. ShopTalk met design requirements **DR2** (store navigation), **DR3** (product search), **DR4** (product identification) and **DR6** (minimal environmental adjustments) either completely or partially. However, it did not meet design requirements **DR1** (mobile product selection) and **DR5** (utilization of existing devices). ShopTalk improved on RoboCart by meeting **DR6** (minimal environmental adjustments). Not meeting **DR1** was not a big issue as RoboCart's method of browsing products using the keypad could be easily incorporated into ShopTalk. However, the main issue with ShopTalk was not meeting **DR5** (utilization of existing devices). Requiring VI shoppers to own, operate and maintain additional devices like the OQO computer and the hand-held barcode scanner not only adds to the cost of ownership but has a significant negative ergonomic impact on them.

ShopMobile 1 improved upon ShopTalk by reducing the hardware complexity of the system to a cell phone and a pen barcode reader. A VI shopper typically uses a guide dog or a cane with one hand and carries a shopping basket or pulls a cart with the other. Also, some shoppers may have small children with them when they go shopping. Thus, shoppers may find it difficult to operate the cell phone and the pen barcode scanner simultaneously while also taking care of the cane, guide dog, cart and children. The other problem with this system was that the Baracoda pen barcode scanner was difficult to use.

Experience with the previous systems have taught us that for a system to be usable in the real world, it is not sufficient for the system to just satisfy requirements like store navigation, product selection, etc. It must also be practical. VI individuals like to remain inconspicuous and do not like to draw attention to themselves. Systems that require shoppers to use robots or wear additional hardware will not be welcomed by the users. Stores do not like systems that disrupt shopper traffic or existing business practices. Systems that require stores to undergo modifications will not be welcomed by supermarkets. The system must also be handy and preferably use hardware that VI individuals already own and know to operate.

1.4 ShopMobile2

ShopMobile2 is the successor to ShopTalk and ShopMobile1. It is based on the premise that computer vision is sufficient for accessible shopping. This system is designed to help the VI shopper shop independently in a real-world grocery store using only a smartphone. It has three major components - an eyes-free barcode scanner, an optical character recognition (OCR) module and a tele-assistance module. The eyes-free barcode scanner allows the VI shopper to scan UPC barcodes on products and MSI barcodes in an *eyes-free* manner. It will be discussed in detail in Chapter 3. The OCR module allows the VI shopper to read barcode labels and nutritional facts on products. This will be discussed in Chapter 6. The tele-assistance module allows the VI shopper to obtain assistance from a caregiver at a remote location by transmitting video from her cellphone. The tele-assistance module is called TeleShop and will be discussed in Chapter 5. ShopMobile2 aims to prove that computer vision algorithms running on a smartphone are sufficient for blind grocery shopping and the smartphone can replace the hardware components needed in ShopTalk and ShopMobile 1.

1.5 A Use Case

Let us consider a typical use case, which describes how ShopMobile 2 can be used for blind grocery shopping. Alice is completely blind. She has good orientation and mobility (O&M) skills, travels independently and can get to a store on her own. She arrives at the store and inputs the desired product on her cell phone.

The cell phone retrieves the location of the product using the BCM and gives her verbal directions, for example, "You will find Thin and Crispy Saltines on Aisle 3, shelf section 4 on the left, shelf 2 and product location 1." Alice finds her way to aisle 3 and enters it. She now scans any MSI barcode on this aisle using the eyes-free barcode scanner and the phone confirms that she is in the correct aisle.

She now finds shelf section 4 on her left by touch and scans a MSI barcode. The system detects that she has scanned a barcode on shelf section 5 instead of shelf section 4 and gives her the following directions, "You have scanned a barcode on shelf section 5. The product is on shelf section 4. Go back one shelf section." She scans MSI barcodes to

find the product and picks it from the shelf. She can now use the phone to scan the UPC barcode on the product to confirm that she has picked the correct product. She can also read the nutritional facts on the product using the OCR module.

Now, let us assume that she wants to shop for a product, which is not present in the BCM. The system cannot give her directions to the product since it does not know the location of the product. Instead of giving up, Alice calls a sighted friend and asks him to help her. She uses the TeleShop module on the phone to transmit video to her friend. She points her phone towards aisle signs and shelves and her friend guides her towards the product. She picks up the product and her friend confirms that she has picked the correct product. Alice can now use the OCR module to read the nutrition facts on the product or her friend can read it out to her.

1.6 Research Goals

The goal of ShopMobile 2 is to allow VI individuals to shop independently using only a smartphone. Both ShopTalk and ShopMobile 1 have demonstrated that VI users can shop independently by scanning barcodes. ShopMobile 2 differs from these two systems by employing only a single smartphone instead of dedicated barcode readers, personal computer, etc. In addition to those two systems, ShopMobile2 allows users to read nutritional facts on products and obtain assistance from sighted caregivers at remote locations.

This dissertation states that computer vision is sufficient for blind grocery shopping and is addressed by the following three systems:

1. A smartphone can be used as an *eyes-free* barcode scanner that can scan both UPC and MSI barcodes. The performance of this system should be comparable to a dedicated hardware barcode scanner

2. A smartphone can be used to read nutritional facts and barcode labels on products using optical character recognition (OCR)

3. A smartphone can be used by VI individuals to obtain assistance from caregivers at remote locations

The eyes-free barcode scanner is the first component of the dissertation. It allows the VI shopper to scan MSI barcodes on shelves and UPC barcodes on products using only a smartphone. Scanning MSI barcodes is comparatively easier since these can be located by touch. Scanning UPC barcodes, on the other hand, is much more difficult since the barcode may be present anywhere on the product. The eyes-free barcode scanner must allow the VI shopper to scan barcodes quickly and efficiently. Since it is meant to replace dedicated barcode scanners, it must also be comparable to them in terms of the time and effort required in scanning barcodes.

The OCR module is the second component of the dissertation. This component should allow the VI shopper to read nutrition facts and barcode labels on the product. The ability to read the nutrition facts gives the VI shopper more independence in terms of choosing products as she is able to make informed decisions. This is also helpful in situations where the VI shopper or any of her family members have allergies to certain ingredients.

The tele-assistance module is the final component of this dissertation. It allows the VI shopper to obtain assistance from sighted caregivers at remote locations.

CHAPTER 2

RELATED WORK

2.1 Introduction

In this chapter, we discuss the various tasks involved in independent blind shopping and postulate a set of design requirements for accessible shopping systems based on these tasks. We review some existing accessible shopping systems and discuss whether they meet these design requirements.

2.2 Design Requirements

We conducted two focus group meetings to identify the accessibility barriers to independent blind shopping. The first focus group consisted of five VI individuals from Logan, UT. Their age ranged from 16 to 47. We met with each individual separately to minimize peer pressure. The second focus group with different participants was conducted during a monthly meeting of the Logan Chapter of the National federation of the Blind (NFB) hosted by the USU Center for Persons with Disabilities (CPD). This group consisted of six people, all of whom held part-time or full-time jobs, used public transportation and walked independently around their neighborhoods. The age ranged between 19 to 51. This meeting lasted three hours.

The written transcripts of both interviews were analyzed using ergonomics for one (EFO), which is an occupational therapy framework proposed by McQuistion [13]. This method fits tasks to individuals with disabilities who must repeatedly accomplish them in specific environments. Interviews or field studies are used to identify *task functions* and to match them with the individual's abilities. Unmatched or partially matched task functions are called *performance gaps*, which must be bridged with solutions called *accommodation*

Table 2.1. Task Functions

Function Name	Function Description
TF1	Shopping list preparation
TF2	Getting to supermarket
TF3	Finding products in store
TF4	Getting to cash registers
TF5	Paying
TF6	Getting to exit
TF7	Getting home

systems. These performance gaps are important because they become *design requirements* for the accommodation systems.

We identified seven task functions that pertain to independent blind shopping [14]. These task functions are given in Table 2.1. The first task function **TF1** is preparing the shopping list. This task function is handled differently if the shopping is *planned* (a complete shopping list is prepared before entering the store), *spontaneous* or *opportunistic* (no shopping list is prepared) and *mixed* (the shopping list is incomplete or has to be modified due to some information received in the store). Planned shopping does not introduce a performance gap since it is assumed that the shopper has access to a PC with a screen reader. Opportunistic and mixed shopping do introduce a performance gap as shoppers are likely to use mobile devices to prepare or modify the shopping list. Eyes-free product selection and browsing user interfaces (UIs) are presently difficult on mobile devices like smartphones. This introduces a performance gap and consequently becomes a design requirement.

The second task function **TF2** is getting to the supermarket. This task function, along with **TF7** (getting to home) pertains to outdoor navigation. This is a separate research area and will not be addressed while formulating the design requirements. The third task function **TF3** is finding products in the store. This task function can be decomposed into three sub-functions: **TF3.1)** store navigation, **TF3.2)** product search and **TF3.3)** product identification (verifying that the correct product has been picked up). Store exploration (product or store browsing) is implied by **TF3.2** and **TF3.3**, but not vice versa. In other words, shoppers cannot search for and retrieve products without browsing but are able to

just browse the store without retrieving products. These three task sub-functions introduce performance gaps and result in three new design requirements - store navigation, product search and product retrieval.

Task functions **TF4** (getting to registers) and **TF6** (getting to the exit) do not introduce new performance gaps since they can be reduced to task function **TF3.1** (store navigation). The fifth task function **TF5** relates to paying. Every participant in the interview mentioned that they handle payments with credit cards and so **TF5** does not introduce any performance gaps. If, however, shoppers prefer to pay with cash, it may introduce new performance gaps since individual bills must be recognized.

Two more design requirements were identified during the interviews. These are utilization of existing devices and degree of environmental adjustment. VI individuals own and operate many devices such as navigation tools, Braille note takers, white canes and guide dogs. In addition to these devices some VI individuals may also have their children with them when they go shopping. Requiring VI shoppers to use special devices for shopping not only adds to the cost of ownership but has a significant negative ergonomic impact on them. The other design requirement is the degree of environmental adjustment. If accessible shopping systems are to work in real supermarkets, we have to take into account the cost that businesses will have to bear to accommodating these systems. For example, during experiments with RoboCart, the management at Lee's marketplace (the store at which these experiments were to be carried out) were concerned with the hardware that must be installed in the store, who would maintain it, and whether it would be disruptive to the shopper traffic. It is clear that if the introduction of the shopping system causes disruptions in the existing business practices, supermarkets will resist or reject its adoption. Thus, it is important that accessible shopping systems cause minimum environmental adjustment with no adjustment being the ideal.

Table 2.2 enumerates all the design requirements of a good accessible shopping system. We will now analyze some existing shopping systems based on these design requirements. Our analysis will be based on the following conventions. We will say that a system S meets

Table 2.2. Design Requirements

Requirement Name	Requirement Description
DR1	Mobile product selection
DR2	Store navigation
DR3	Product search
DR4	Product identification
DR5	Utilization of existing devices
DR6	Minimal environmental adjustments

a design requirement R when S has a hardware or software module specifically designed to meet R. If there is evidence that the designers of the system are aware of R but have not yet implemented it, we will say that S partially meets R. If these is no such evidence, we will say that S does not meet R.

2.3 RoboCart

RoboCart [4–6] was the first assistive shopping system developed at the Computer Science Assistive Technology Laboratory (CSATL) at Utah State University. As shown in Figure 2.1, RoboCart was built on top of a Pioneer 2DX robot to which a polyvinyl chloride (PVC) structure was securely attached. The robot was equipped with a SICK laser range finder, SONAR array, wheel encoders, and a RFID reader with an antenna. A laptop was placed on the PVC structure and was used to control the robot. A shopping basket and a numeric keypad were also attached to the PVC structure.

To shop with RoboCart, shoppers would select products using the numeric keypad, which was modeled after a cell phone keypad. The robot would compute a path to the product's vicinity and shoppers would follow the robot by holding a handle on the PVC structure. Earlier versions of RoboCart had a rope-like tether attached to the PVC structure and shoppers would use this to follow the robot but it was found that this did not provide adequate feedback and was later replaced with the rigid handle. Once shoppers arrived in the vicinity of the product, they would use a hand-held wireless barcode scanner (IT4600 SR) to retrieve items from the shelf and the robot would take them back to the cash registers.

The robot used Markov localization using readings from the laser range finder and wheel encoders to localize in the store. It was found that this method was not very robust to changes in the map (introduced by people and changing store displays) and so passive RFID tags were used to correct errors in Markov localization. These tags were placed at the beginning and end of each aisle and at three different locations within each aisle.

Three studies were carried out using RoboCart. The first study was conducted using two VI participants. The feedback obtained from this study was used to make iterative hardware and software changes to the robot's navigation routines. The second study was a single-subject study conducted over two months where the VI participant was asked to shop in a real grocery store using this system. A total of seven trials were conducted on three separate days for three different sets of products. The participant successfully completed all trials and retrieved all products. The third study was executed with ten VI participants from the Utah Federation of the Blind (NFB) chapter. This study lasted over four months and each participant had to execute fifteen runs on two different days in a real supermarket. All participants were able to complete all runs successfully.

RoboCart proved that VI individuals are able to shop independently in a real grocery store. However, this system was not practical. The two main problems with this system were the robot and RFID tags. The robot is very expensive to purchase and maintain. Also, most stores would not be comfortable with a robot moving about and possibly causing damage. The RFID tags are passive and require no maintenance once installed but there is a initial cost to instrumenting the environment. In terms of design requirements, RoboCart partially meets **DR1** (mobile product selection) since its keypad is emulated as a phone's keypad. It meets **DR2** (store navigation) through robot navigation and partially meets **DR3** (product search). It also meets **DR4** (product identification) as shoppers can scan UPC barcodes on products using the hand-held wireless scanner. It fails in **DR5** (utilization of existing devices) and **DR6** (minimal environmental adjustments) since it requires a robot and instrumenting the environment with RFID tags.

Figure 2.1. RoboCart assistive shopping system.

2.4 ShopTalk

ShopTalk [7–9] was the second assistive shopping system developed at CSATL. It was a wearable system designed to address two major shortcomings of RoboCart - using the robot and instrumenting of the environment with RFID tags. Figure 2.2 shows the hardware used in the system, which consists of a OQO ultraportable computer, a Belkin numeric keypad, a hand-held wireless IT4600 SR barcode scanner with its base station and a USB hub to connect all components. This hardware was carried by the user in a small CamelBak backpack. The numeric keypad was attached to the left or right shoulder strap, depending on whether the shopper was right-handed or left-handed. The shopper received instructions through headphones.

ShopTalk was based on the following conceptual formula: *independent blind shopping* =

verbal route instructions + shelf barcode scans. Most stores place small MSI type barcodes on shelves beneath products for inventory control purposes. ShopTalk uses these barcodes as topological cues for product search and store navigation. A key data structure in ShopTalk is the *barcode connectivity matrix* (BCM). The BCM is a graph like data structure that associates MSI barcodes with aisles, aisle sides, shelf sections (groups of shelves), specific shelves in the shelf section and relative position on shelves. The BCM is used to generate the store navigation and product search and retrieval instructions. Shoppers can scan any shelf barcode in the store and receive information about their position in the store and directions to products.

Two single subject studies were carried out at Lee's marketplace and one single subject study was carried out at Sweet Peas, a smaller independent store. Later, a formal longitudinal study with ten participants was carried out at Lee's marketplace. The product database included 4297 products. In this study, each participant was asked to retrieve the same set of three randomly chosen products five times. It was found that all participants were able to retrieve all the products successfully in each run.

Shoptalk was able to prove that it is possible for VI shoppers to shop independently using barcode scans and verbal route directions. As an accessible system, ShopTalk does not meet **DR1** (mobile product selection) since it assumes that the shopping list has been prepared and stored on the OQO computer. This system meets **DR2** (store navigation) and **DR3** (product search) since it provides verbal route descriptions to the shopper. The system partially meets **DR4** (product identification) as shoppers can find the identity of the product by scanning the UPC barcode on it. However, as observed during the experiments, the scanner is difficult to use on softer packages like potato chips. The system does not meet **DR5** (utilization of existing devices) because it requires shoppers to purchase additional hardware, most of which cannot be utilized for other purposes. The system partially meets **DR6** (minimal environmental adjustment) because it does not require installing and maintaining additional devices but it does assume access to the store's inventory control system. Some stores may not be willing to provide access to their database and in this case

the BCM cannot be computed automatically.

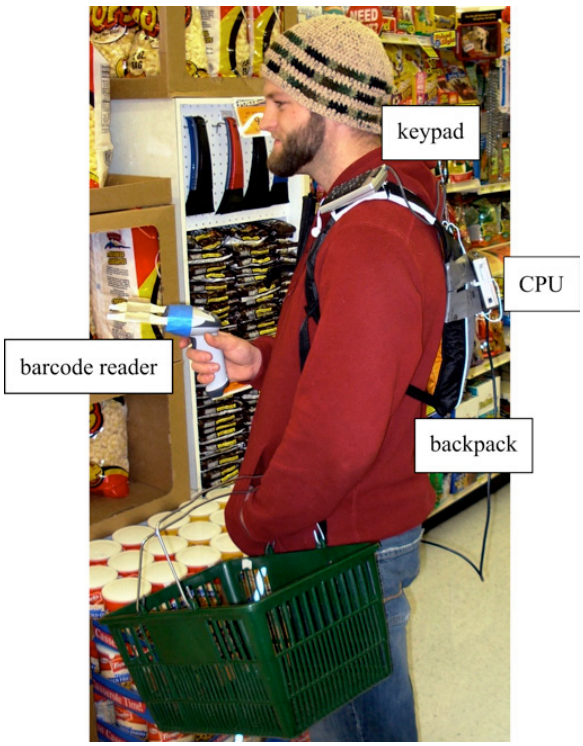


Figure 2.2. Hardware components of ShopTalk.

2.5 GroZi

GroZi [10, 11] is an assistive shopping system for the blind developed by researchers at the University of California at San Diego. This system uses a custom hardware solution called a MoZi box (see Figure 2.3) for object recognition. The MoZi box is equipped with a camera and haptic interfaces. In GroZi, shoppers point the MoZi box at aisle signs. The system captures images from the camera and decodes them using optical character recognition (OCR) and reads out the text using text to speech. Shoppers enter the aisle and point the MoZi box towards products on shelves and move forward. The system captures images of products continuously and compares them with images stored in its database to recognize products. Once products are located, shoppers are guided towards them using voice or haptic interfaces.

GroZi requires two sets of images for each product - *in vitro* images taken under perfect

conditions and *in situ* images taken in natural environments. The Grozi120 [15] database contains an average of 5.6 *in vitro* images and 93.3 *in situ* images for 120 products. Images taken at run-time are compared with the *in vitro* and *in situ* images using different techniques such as color histogramming, SIFT and boosted Haar-like features to recognize products. It was found that SIFT resulted in the best performance (18% precision and a 72% recall rate).

Grozi does not meet **DR1** (product selection) and it is assumed that the shopper has already prepared the shopping list. It partially meets **DR2** (store navigation). As of now, the system assumes that the shopper is capable of navigating on her own but the designers of this system are aware that store navigation is a design requirement and plan to address it in future implementations. The system partially meets **DR3** (product search) since the shopper can point the MoZi box towards the shelves and receive instructions (up, down, left, right, step back and camera blurry). The shopper can also point the MoZi box towards aisle signs and have them read aloud. These instructions should be sufficient for the shopper to retrieve products but this has not been verified in a real supermarket during regular operating hours. This system partially meets **DR4** (product identification). The system uses vision algorithms (e.g., ShelfScanner) that can identify products at roughly two frames per second. However, there is a huge ergonomic cost to this as the shopper has to carry a power laptop on his or her back. This system does not meet **DR5** (utilization of existing devices) since the shopper has to use the MoZi box for shopping. Although, the designers of this system have used off the shelf components to reduce cost and make maintenance simple, these components have to be assembled by hardware professionals.

The two main issues with this system are the image database and the custom hardware. The current database contains an average of 98.9 images per product. A typical grocery store contains about 38,718 products and this would require taking millions of images. Also, since the system requires images to be taken in their natural environments, the *in situ* images would have to be taken separately for each store. It is also unclear if the current performance would scale comparably for such a high number of images. The other problem

with this system is that it relies on custom hardware, which is typically expensive and can only be used for a particular task - shopping, in this case.

Figure 2.3. MoZi box.

2.6 iCare

The iCare shopping system [16] has been developed by researchers at Arizona State University. Figure 2.4 shows the iCare framework. This system utilizes a RFID reader embedded within a glove, a PDA with Bluetooth, Wi-Fi and screen reader. To use this system, shoppers wear the RFID glove and pass their hand over items in the aisle. The RFID reader reads RFID tags on products and transmits this information to the PDA through Bluetooth. The PDA queries a relational database stored on the store's server through WiFi to access information about products and delivers messages like, "passing dairy section," "passing coffee section," etc to the shopper. Shoppers can use this information to navigate to the appropriate section in the store and retrieve products. Shoppers can also get information about the product's price, weight, ingredients and nutritional information when they pick up the products.

As an accessible system iCare does not meet **DR1** (mobile product selection) since they do not seem to mention it in the paper or the references. The system partially meets **DR2** (store navigation) and **DR3** (product search) because the shopper can browse products using the RFID tags on them. The system focuses on product browsing and so it is

unclear if store navigation and product search have been implemented or evaluated in a real supermarket. The system meets **DR4** (product identification) since the shopper can read tags on individual products and get their information from the server. It has been our experience that RFID tags do not work well when placed on metal surfaces and it is unclear if the designers of this system have taken this into consideration. The system partially meets **DR5** (utilization of existing devices) because most components are commercially-off-the-shelf except the hand glove with the RFID reader. The system does not meet **DR6** (minimal environmental adjustment) since it assumes that every item in the store is tagged with a RFID tag. While this may be possible in the future, currently, most products are not individually embedded with RFID tags. This system also requires access to the store's inventory control system and as mentioned previously, most stores will not be comfortable in providing.

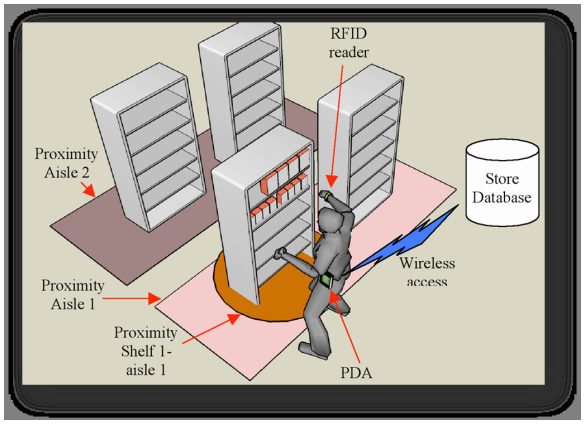


Figure 2.4. iCare framework.

2.7 Trinetra

Trinetra [17–19] is an assistive shopping system designed by Carnegie Mellon University. This system uses a Nokia 6620 smartphone, a Bluetooth wireless headset, Baracoda's IDBlue RFID scanning pen, Baracoda's barcode scanning BaracodaPencil and generic high frequency tags from Texas Instruments. To use this system, shoppers scan either the RFID tag or the UPC barcode on the product using the Baracoda pens. The UPC code or the

RFID tag number is then passed through Bluetooth to the smartphone. The system then checks a local cache on the smartphone to retrieve the product description. In case of a cache hit, the product description is returned to the user. In case of a cache miss, the Trinetra module on the phone communicates with a desktop server over TCP to check for the product entry in a larger cache. If a cache miss occurs at this stage, the system checks either the store inventory control system or a public UPC database on the web to return the product information. The local caches are updated if this information is found.

This system has been evaluated with one VI individual from the development team in a small store. In this study the VI shopper was led to the vicinity of items in a small store and he used the Trinetra system to find the product and pick it. It was reported that he was able to find all products successfully.

This system partially meets **DR1** (mobile product selection) since there is evidence of the shopper using the phone to select products. It is unclear if the shopper selects products from a prepared list or from a large database of products. The system does not meet **DR2** (store navigation) since it assumes that the shopper is able to navigate to the product's location either independently or using someone's assistance. While this assumption may hold true for a small convenience store, it is unlikely to hold in a modern supermarket. The system does not meet **DR3** (product search) in that there does not seem to be any guidance provided to the shopper at run time. The system partially meets **DR4** (product identification) as the shopper can identify the product by scanning the UPC barcode or reading the RFID tag on it. However, more empirical evidence is needed to show that VI shoppers can use the Baracoda pencil to scan barcodes easily as our own experience with the Baracoda pencil indicates otherwise. The system partially meets **DR5** (utilization of existing devices) since this system is the first to advocate using mobile phones for accessible blind shopping. This system does require the Barcoda pencil readers, which many VI individuals do not own and may not be useful in applications other than shopping. If the barcode version of the system is adopted, it partially meets **DR6** (minimal environmental adjustment) as the store has to agree to install and maintain the UPC server. If, on the

Table 2.3. Comparison of Various Assistive Shopping Systems

System	DR1	DR2	DR3	DR4	DR5	DR6	Field Evals
RoboCart	±	+	±	±	-	-	+
ShopTalk	-	+	+	±	-	±	+
GroZi	-	±	±	±	-	?	-
iCare	-	±	±	+	+	-	-
Trinetra	±	-	-	±	+	±	-

other hand, the RFID version of the system is adopted, this system will fail to meet **DR6** because item level RFID tagging does not exist yet and supermarkets will have to undergo major environmental adjustments to accommodate it.

2.8 Summary

Table 2.3 shows a summary of how the various accessible shopping systems described earlier meet the design requirements. In this table, a plus sign signifies that the system met the particular design requirement, a minus sign signifies that the system did not meet the design requirement, a plus/minus sign means that the system partially meets the design requirement and a question mark means that it is unclear whether the system meets the design requirement. The last column specifies whether a give system was evaluated in the field with at least ten VI participants.

CHAPTER 3

EYES-FREE BARCODE SCANNER

3.1 Introduction

Both ShopTalk and ShopMobile 1 proved that VI shoppers can shop independently by scanning barcodes and following verbal directions. However, these systems used dedicated barcode scanners (a wireless hand-held barcode scanner in ShopTalk and a Baracoda pen barcode scanner in ShopMobile 1) to scan barcodes. These dedicated barcode scanners work well but require shoppers to purchase and maintain additional hardware. The aim of ShopMobile 2 is to reduce the hardware complexity of the system. To do this successfully, it must be able to replace a dedicated barcode scanner by allowing the VI shopper to scan UPC barcodes on products and MSI barcodes on shelves quickly and reliably using only the smartphone.

Barcode scanners can be broadly classified into three types - laser-based barcode scanners, pen barcode scanners and CCD barcode scanners. Most dedicated barcode scanners, such as the ones used by supermarkets at checkout stands are laser-based. These barcode scanners use a laser beam to illuminate the barcode and capture the reflected beam using a photo diode. This reflected beam is processed using signal processing algorithms to decode the barcode. Laser-based barcode scanners are fast and easy to use since they illuminate the entire surface of a product. Most laser-based scanners employ a two dimensional beam pattern so that the barcode can be read even if it is rotated. A pen barcode scanner uses the same technique as a laser barcode scanner but employs a single LED for illuminating the barcode. A pen barcode scanner cannot illuminate the entire surface of the barcode and so it has to be dragged across the barcode along a straight line to read it. Using a pen based barcode scanner is not as easy as using a laser based barcode scanner since it requires some

practice to drag it across the barcode at proper speeds [12]. The CCD barcode scanner employs an image sensor typically a charged-coupled device (CCD), to capture an image of the barcode. This image is then decoded to read the barcode. The CCD barcode scanner may employ LEDs or some other light source to illuminate the barcode. Mobile phones also contain a CCD (or CMOS) camera and so applications that use mobile phones to scan barcodes can be classified under CCD barcode scanners.

A lot of research has been performed in the area of decoding barcodes using CCD cameras and mobile phones. Ohbuchi et al. [20] have demonstrated barcode scanning using a camera, mobile application processor, digital signal processor (DSP), and a display device. Many systems [21–24] and applications [25, 26] have been developed for scanning barcodes using mobile phones. These solutions have been developed for sighted users and may not be suitable for VI persons. For example, RedLaser [25] and ZXing [26] are two popular barcode scanning applications for smartphones. These solutions require users to carefully position the phone's camera with respect to the barcode for scanning and cannot decode MSI barcodes. The system described in [27] is developed specifically for VI individuals but this system places colored fiducials next to barcodes for easy localization. This system also uses a custom-made variation of the UPC standard for encoding barcodes. Gallo and Manduchi [28] have proposed an algorithm for decoding barcodes for VI individuals but have not demonstrated it on a cell phone.

In this chapter, we will describe the *eyes-free* barcode scanning solution, which allows VI shoppers to quickly and reliably scan both MSI barcodes on shelves and UPC barcodes on products using only a smartphone.

The eyes-free barcode scanning solution is comprised of four modules - interactive camera alignment module, barcode detection module, barcode localization module and the barcode decoding module. Let us consider a use case, which illustrates how these modules work together. Alice is a completely blind shopper who wants to scan the UPC barcode on a product. She places the phone on the product such that the camera faces the product and aligns it with the product. UPC barcodes are usually located on the bottom side of

boxes and on the sides of cans or bottles. If the product is a box, she finds the bottom side of the box and aligns one edge of the phone with the corresponding edge of the box. If it is a bottle, she aligns the bottom edge of the phone with the bottom edge of the bottle. If the product is a can, she aligns either the top or the bottom edge of the phone with the corresponding edge of the can. Once the phone is aligned, she slowly moves it away from the product. The system automatically detects that she has started moving the phone away from the product and starts a timer, which notifies her to stop moving the phone once it reaches a preset threshold. It is assumed that the user moves the phone with a certain speed. The timer preset value can be adjusted such that the phone is approximately six to eight inches away from the product when it reaches its preset value.

The system now starts the barcode detection module, which takes images continuously in video mode and analyzes each image to detect the presence of a barcode. She now uses the phone like a flashlight - looking for a barcode on the product. If she does not find a barcode, she switches over to a different side in case of a box and repeats the entire procedure or rotates the can or bottle. If a barcode is found, the system starts beeping to let the user know that a barcode has been detected. The system now starts the barcode decoding module, which tries to decode the barcode in the image. If the barcode is decoded successfully, it is read out to the user otherwise the system starts the barcode localization module, which finds the precise location of the barcode in the image and segments it from the image. The barcode decoding module now tries to decode the barcode in the localized image. The system tries to decode the barcode in the original image as well as in the localized image to improve the chances of decoding the barcode. If it is successful, the barcode is read out to the user or else the system tries to decode the barcode in a 90-degree rotated copy of the image (with and without localization) to take care of situations where the barcode may be rotated by 90-degrees in the image. If the barcode is still not decoded, the system checks whether the barcode is cropped along one of the sides and notifies the user accordingly. It then starts the barcode detection module and a new image is captured and analyzed. The interactive camera alignment mode runs during the entire time and helps the

user keep the phone aligned with the product. Figure 3.1 shows the flow of control between the different modules.

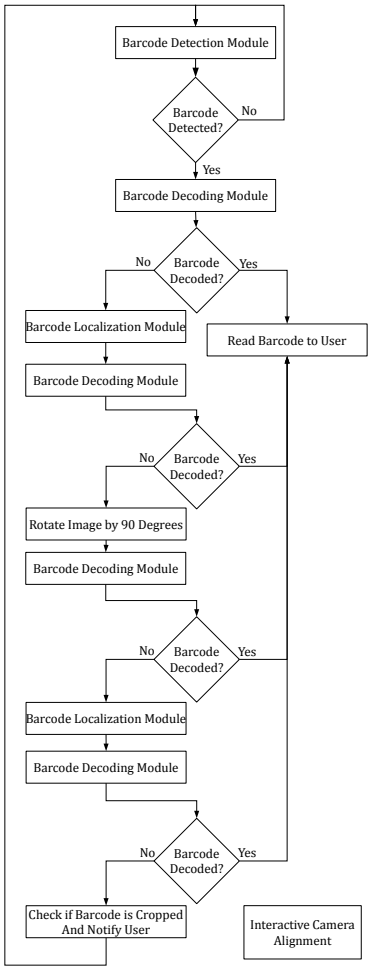


Figure 3.1. Flow of control between the different modules of the eyes-free barcode scanner.

The rest of the chapter is organized as follows. In section 3.3, we describe the interactive camera alignment module. In section 3.4, we describe the barcode detection module and in section 3.5, we describe the barcode localization module. Finally, we describe the barcode decoding module in section 3.6.

3.2 Hardware - Choosing a Smartphone

In ShopMobile 2, a smartphone completely replaces the hardware barcode scanner as well as other components used in ShopTalk and ShopMobile 1. Thus, its choice was very

important and specific hardware and software requirements had to be taken into consideration while making this choice. We identified the following hardware requirements for the phone: a) possess a camera capable of capturing images at a distance of few inches, b) has a led flash, c) has sufficient computing power to run image processing algorithms, and d) has a physical keyboard to allow VI individuals to operate the phone. In terms of software, the requirements were that the phone's operating system must have the requisite API's for camera control, internal sensors, etc.

Image processing algorithms tend to be computationally expensive and so we wanted a high-end smartphone capable of running our software smoothly. At that time, most high-end smartphones were touch screen devices and we were skeptical whether VI individuals would be able to use a touch screen phone since it has few hardware buttons. We contacted three VI people telephonically and asked them if they would be comfortable using touch screen phones for shopping. All the three participants said that they had either used a touch screen phone or knew VI people who were using touch screen phones without any problems. All participants said that they would be comfortable in using the touch screen phone as long as we employed suitable touch interfaces. This caused us to drop the physical keyboard requirement for the phone.

We identified the Google Nexus One and the iPhone as two potential candidates. Both phones had autofocus cameras, sufficiently fast processors and internal sensors. The Google Nexus One phone was available unlocked from Google without any contract whereas the iPhone could only be purchased with a two-year wireless contract from AT&T. The iPhone required developers to purchase a developer license and could only be programmed using a Mac whereas the Nexus One could be programmed for free on either Windows, Mac OS or Linux. In terms of programming language, the Nexus One could be programmed using Java, which we preferred, whereas the iPhone could be programmed using Objective-C. These considerations led us to choose the Nexus One as the smartphone for this project.

The Google Nexus One phone is shown in Figure 3.2. Initially, it ran Android 2.1 but has since been upgraded to Android 2.3. All our software is compatible with Android 2.2+.

This phone can be programmed in Java and can also run native C code using the Java Native Interface (JNI). In terms of hardware, it has a 1 Ghz processor, 512 MB of RAM and a 4 GB SD card. It contains a 5 Megapixel camera with autofocus and led flash for taking pictures. It has the following sensors - GPS, an orientation sensor, an accelerometer and a light sensor. The phone has a 3.7 inch multi-touch screen and small joystick like trackball beneath the screen and both of these can be used for obtaining input from users.

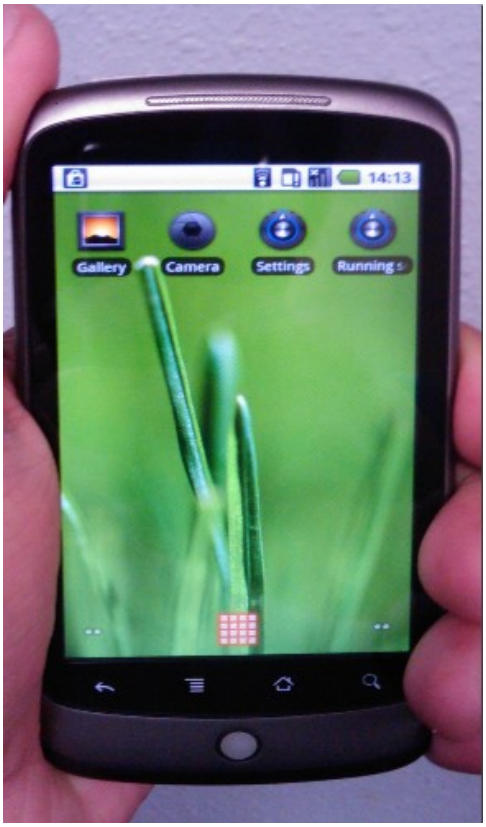


Figure 3.2. Google Nexus One phone.

3.3 Interactive Camera Alignment Module

As described in the use case, the VI shopper starts the barcode scanning process by aligning the phone with respect to the product and the interactive camera alignment module then helps her keep the phone aligned with the product at all times. Let us consider why it is so important to do this. As shown in Figure 3.3, any rigid body can be rotated along three axes - pitch, yaw and roll. Rotating the phone along any one of these axes will cause

the phone (and hence the camera) to be misaligned with the product and introduce skew distortions in the image. Figure 3.4 shows an example of an image and its skew distorted equivalent. Skew distortions are undesirable because they reduce the barcode decoding rate. We first observed this during a pilot experiment where we asked one VI and three blindfolded sighted participants to decode UPC barcodes on ten products. Participants were instructed to align the camera with the product and then slowly move it away from the product. Since there was no interactive camera alignment module to help them keep the phone aligned with the product, participants would inadvertently rotate the phone while moving it away from the product. This introduced skew distortions in the image, which in turn, reduced the barcode decoding rate.

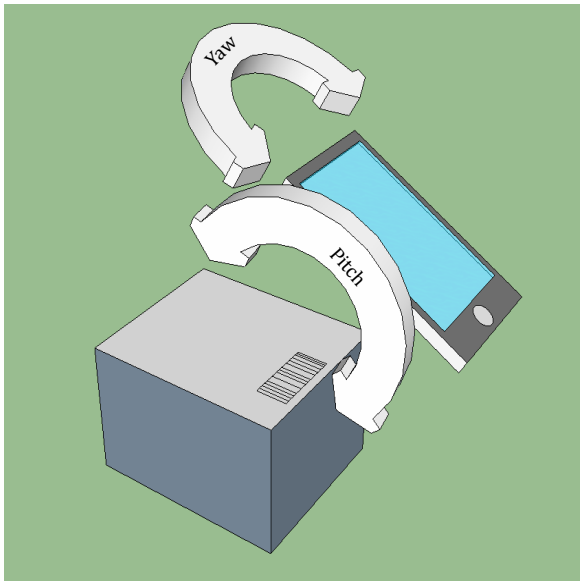


Figure 3.3. Aligning the camera in the pitch and yaw planes.

To increase the barcode decoding rate, we can either make the barcode decoding algorithm robust to skew distortions or minimize skew distortions in the image. Many methods have been proposed to detect and correct skew distortions in images. One such method [29] uses Radon transforms to detect and correct skew distortions. Another method [30] uses the Hough transform to do the same. Incorporating skew detection and correction algorithms in the barcode scanning process would increase its complexity and run time. On the other hand, minimizing skew distortions is simpler, more efficient and is easily implemented by

Figure 3.4. Examples of image (left) and its skew distorted equivalent (right).

the interactive camera alignment module.

The interactive camera alignment module starts automatically as soon as the shopper places the phone on the product. When the phone is in contact with the product, the camera is blocked by the product and so its output is a completely dark image. This is easily detected (average luminance of the image is approximately equal to zero) and the phone's orientation along the pitch, yaw and roll axes are captured using the phone's internal orientation sensor. These readings are averaged to define the absolute pitch, yaw and roll planes. The completely dark image causes the automatic exposure control of the camera to set the sensor gain to a very high value. As soon as the shopper moves the phone away from the product, the camera is no longer blocked and the output is a very bright image due to the high sensor gain. This bright image is also easily detected (average luminance of the image is approximately equal to 255) and a timer is started. The shopper is instructed to stop moving the phone away from the product once the timer reaches a preset value. This preset value can be set depending on how quickly the shopper moved the phone. Once the timer starts, the phone's orientation along the pitch, yaw and roll axes are read continuously. If the phone is misaligned along any plane, specific instructions like "pitch up" or "roll left" are issued to the shopper to help her realign the phone. The interactive camera alignment module runs continuously and stops automatically only when a barcode is decoded or if the user turns it off manually.

3.4 Barcode Detection

The barcode detection algorithm is responsible for detecting the presence of a barcode in an image and notifying the user. Earlier versions of the barcode scanner did not include this module. We realized the necessity of the barcode detection module after performing an experiment in which two VI participants were asked to decode UPC barcodes on ten products. The participants took an average of 83.6 and 93.4 seconds, respectively, to scan a barcode. These scanning times are very high as compared to dedicated hardware based barcode scanners and are therefore unacceptable for our application. Upon analyzing the video of the experiment, we found that participants spent most of their time trying to scan barcodes on surfaces that did not contain a barcode. For example, a box contains six surfaces out of which only one contains a barcode. Since users had no way of quickly determining which surface contained a barcode, they spent equal time on all surfaces until they finally scanned the barcode. The barcode scanning times could be reduced drastically if we could quickly determine the presence of a barcode on a surface. Thus, shoppers could scan a surface and quickly determine if it contained a barcode or else move to a different surface.

Figure 3.5 shows an overview of the barcode detection algorithm. The algorithm works by obtaining an image from the camera and binarizing it into a bi-level image. The binarized image is then divided into subimages and x and y gradients are computed for each subimage. These gradient regions are then classified by a support vector machine (SVM) as barcode or non-barcode regions. Finally, the algorithm looks at the number of barcode and non-barcode regions to decide if the image contains a barcode or not.

3.4.1 Binarizing Images

Typically, the camera produces a color image where each color pixel p_c can be represented by a triad $p^C = (R, G, B)$ corresponding to the colors - red (R), green (G) and blue (B), respectively. Each color is defined by a 8-bit value so $0 \leq R \leq 255$, $0 \leq G \leq 255$ and $0 \leq B \leq 255$. However, all algorithms in our application use either grayscale image Y, where each pixel $0 \leq p^Y \leq 255$ or a bi-level image B, where each pixel $p^B = \{0, 255\}$. We

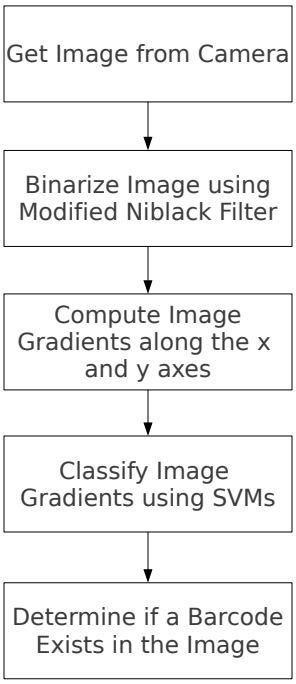


Figure 3.5. Overview of the barcode detection algorithm.

can use the formula $p^Y = 0.3 \times R + 0.59 \times G + 0.11 \times B$ to obtain p^Y from p^C. The Nexus One phone used in our experiments has the ability to capture a grayscale image directly. Thus, for simplicity and efficiency we use $p^Y = G$ to obtain p^Y from p^C.

The grayscale image Y can be converted to the bi-level image B by binarization. To binarize an image, a threshold T is chosen and every pixel p^B in B is set to 255 (white) if its corresponding pixel p^Y in Y is greater than or equal to T or 0 (black) otherwise. Binarization methods can be classified as global methods or local methods depending on the value of the threshold T. In a global binarization method, the value of the threshold T remains the same for each pixel in the image. Conversely, in a local binarization method the value of the threshold T can vary for each pixel in the image. Global binarization methods like [31–33] are simpler but may not be able to handle large variations in illumination over the entire image. Local binarization methods like [34, 35] are able to handle such variations but require computing the value of T for every pixel in the image, which may be

computationally expensive.

We have developed a new binarization method which is loosely based on Niblack's original method. Equation 3.1 shows the method to determine the local threshold $T(x, y)$ for each pixel located at coordinates (x, y) in Y using Niblack's original method. In this equation, $m(x, y)$ and $s(x, y)$ denote the mean and standard deviation values for a $n \times n$ window centered on (x, y) and k is a user defined parameter, which is usually negative.

$$T(x, y) = m(x, y) + k \times s(x, y) \tag{3.1}$$

It is expensive to compute the threshold $T(x, y)$ for each pixel and hence in our modified Niblack method, we divide the image Y in to $n \times n$ pixel subimages and compute a single threshold $T(i, j)$ for each subimage $Y_{i,j} \in Y$. Equation 3.2 shows the method for determining $T(i, j)$. In this equation, $m(i, j)$ and $s(i, j)$ denote the mean and standard deviation values for all the pixels in the subimage $Y_{i,j}$ and k, S & T^c are user defined parameters. For our application, we used $n = 15$, $k = 0$, $S = 12.7$ and $T^c = 127$.

$$T(i, j) = \begin{cases} m(i, j) + k \times s(i, j) & \text{if } s(i, j) \geq S \\ T^c & \text{otherwise} \end{cases} \tag{3.2}$$

It can be noted that since we set $k = 0$, the above equation reduces to $T(i, j) = m(i, j)$ when $s(i, j) \geq S$. The choice of k influences whether the resulting binarized image will be light or dark. Choosing a negative value of k makes the resulting binarized image lighter whereas choosing a positive value of k makes the resulting image darker. For this application, we choose $k = 0$ since we do not want the resulting image to be lighter or darker. However, as will be discussed in Chapter 6, while performing optical character recognition, we choose $k = -0.5$. This results in a lighter binarized image, which makes segmentation easier.

Global methods produce less noise in the binarized image whereas local methods preserve details in the image that would have otherwise been lost due to small variations in illumination. The modified Niblack method combines the best of both methods. It employs

a fixed threshold T^c on subimages that exhibit low standard deviation in the grayscale values of their pixels and an adaptive threshold on subimages that exhibit high standard deviation in the grayscale values of their pixels. A barcode region consists of a large number of alternating black and white lines and hence a subimage containing a barcode will exhibit high standard deviation. This subimage will be thresholded with the adaptive threshold. A subimage containing a constant background will exhibit a low value of standard deviation and will be thresholded with the constant threshold T^c.

Our modified Niblack method is very efficient compared to the original Niblack method. Computing a single threshold takes $O(n^2)$ time. The original method computes a threshold for each pixel in the image and so its running time is $O(whn^2)$, where w and h denote the height and the width of the image, respectively. Our method computes a single threshold for each $n \times n$ subimage. Since there are $\frac{w \times h}{n^2}$ such subimages, the run time of our algorithm is $O(wh)$. Figure 3.6a shows a grayscale image and Figures 3.6b to 3.6d show the corresponding bi-level images obtained using a global threshold, original Niblack method and our modified Niblack method. It can be observed that a lot of detail is lost when binarizing the image using a global threshold. Binarizing the image using Niblack's original method introduces a lot of noise into the image where as our method provides a good balance between noise reduction and detail in the image.

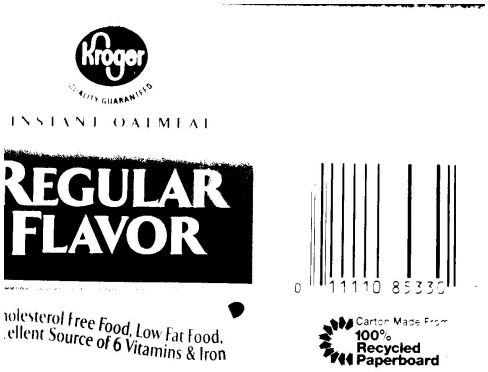
3.4.2 Image Gradients

The next step in the barcode detection process is to compute the image gradients along the x and y axes. A gradient of a continuous function is defined as its derivate along a particular direction. Since an image comprises of discrete pixels, we can compute its gradient using convolution. We can compute the image gradients B^x and B^y for the binarized image B along the x and y axes as follows:

$$B^x = B * G_x \tag{3.3}$$

$$B^y = B * G_y \tag{3.4}$$

(a) Original grayscale image



(b) Binarized image obtained using a global threshold

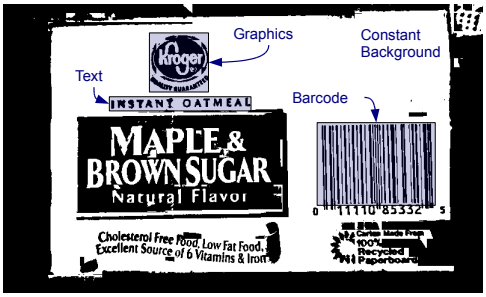
(c) Binarized image obtained using original Niblack method

(d) Binarized image obtained using our modified Niblack method

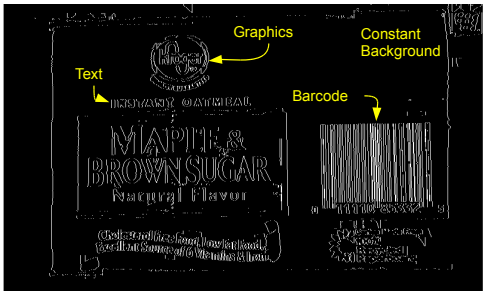
Figure 3.6. Binarization results using various methods

To obtain gradients along both the positive and negative directions along each axes, we define $G_x = [-1, 2, -1]$ and $G_y = [-1, 2, -1]^T$. Figure 3.7 shows the image gradients along the x and y axes for an image containing a barcode. As shown in Figure 3.7, a product package typically consists of four distinct image regions - background, text, graphics and barcode. We are interested in identifying the barcode region and so we must identify some of its properties that will help us in distinguishing it from the other image regions. A barcode consists of a large number of parallel alternating black and white lines in a small region. A barcode with vertical lines exhibits a large gradient along the x axis and a small gradient along the y axis. We can observe that the background region exhibits a small gradient along both the axes and text and graphics exhibit large gradients along both the axes. Thus, we

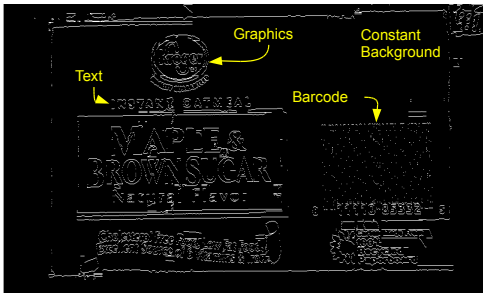
can characterize a barcode region as a region that exhibits a large gradient along one axis and a small gradient along the other.



(a) Original binarized image



(b) Image gradient along x axis



(c) Image gradient along y axis

Figure 3.7. Image gradients along the x and y axes.

To implement this algorithm, the binary image is divided into $n \times n$ pixel subimages $B_{i,j}$. Let the top left and bottom right points for each subimage $B_{i,j}$ be denoted by $P_{TL} = (x_{TL}, y_{TL})$ and $P_{BR} = (x_{BR}, y_{BR})$, respectively. We can now compute the gradients along

the x and y axes, $B^x_{i,j}$ and $B^y_{i,j}$, respectively for each subimage as follows:

$$B^x_{i,j} = \Sigma^{y_{BR}}_{y=y_{TL}} \Sigma^{x_{BR}}_{x=x_{TL}} |2 \times B(x,y) - B(x-1,y) - B(x+1,y)| \tag{3.5}$$

$$B^y_{i,j} = \Sigma^{y_{BR}}_{y=y_{TL}} \Sigma^{x_{BR}}_{x=x_{TL}} |2 \times B(x,y) - B(x,y-1) - B(x,y+1)| \tag{3.6}$$

As described earlier, the values of $B^x_{i,j}$ and $B^y_{i,j}$ vary depending on the type of image in $B_{i,j}$. If the subimage $B_{i,j}$ contains a constant background, the values of both $B^x_{i,j}$ and $B^y_{i,j}$ will be low. If the subimage contains text or graphics the values of both $B^x_{i,j}$ and $B^y_{i,j}$ will be high. However, if the subimage contains a barcode, the values of only one of $B^x_{i,j}$ and $B^y_{i,j}$ will be high and the value of the other will be low depending on the orientation of the barcode lines.

3.4.3 Classification using Support Vector Machines

A Support Vector Machine (SVM) [36,37] is a linear classifier, which is used to classify linearly separable data. Figure 3.8 shows linearly separable data points belonging to two classes P (positive) and N (negative). We can construct a line L (a hyperplane in n-dimensional space) that separates the positive examples from the negative ones. Let the equation of this line be $wx + b = 0$, where w is normal to the hyperplane, $|b|/||w||$ is the perpendicular distance from the hyperplane to the origin and $||w||$ is the Euclidean norm of w. Let d_+ and d_- be the shortest distances from the hyperplane to the closest positive and negative examples. The margin of the hyperplane is defined to be $d = d_+ + d_-$ and a maximum margin hyperplane is defined as one that maximizes this margin. Points lying above this hyperplane can be classified as positive examples and points lying below this hyperplane can be classified as negative examples.

In our case, the data points $\{x_i, y_i\}, x_i \in B^x, y_i \in B^y$ belong to two classes P (barcode regions) and N (non-barcode regions) and we can use SVMs to classify them. To construct the SVMs, we collected over a hundred images and manually classified them into the two classes P and N. We then divided each image into $n \times n$ pixel subimages ($n = 50$) and computed the values of $B^x_{i,j}$ and $B^y_{i,j}$ for each subimage. Consider the graph of points

$\{x_i, y_i\}, x_i \in B^x, y_i \in B^y$ for all the subimages $B_{i,j}$ within each image, which is shown in Figure 3.9. Green colored points belong to P and red colored points belong to N.

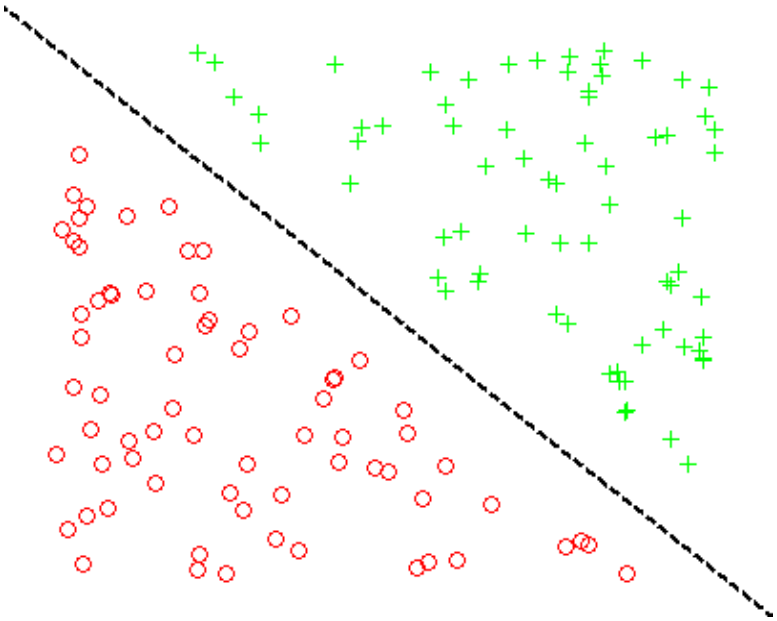


Figure 3.8. Example of a SVM.

It can be observed from the figure that these points are not linearly separable and so in theory, we should not be able to use SVMs to classify them. However, we can observe two regions on the bottom right and top left of the figure that contain only green colored points. These regions represents subimages that contain a barcode and we can construct two SVMs - SVM_1 and SVM_2 (shown by black colored dashed lines in the figure) to classify them. These SVMs can be constructed using the maximum margin hyperplane described earlier. A data point (corresponding to a subimage) is said to contain a barcode if it lies above SVM_1 or below SVM_2. Points lying between the two SVMs are classified as subimages that do not contain a barcode. We can conclude that an image contains a barcode if it has sufficient number of points ($N_p \geq T_N$) that lie either above SVM_1 or below SVM_2, where T_N is some threshold.

Let us represent SVM_1 with the equation $y = mx + c$. From the symmetry between SVM_1 and SVM_2, we can obtain the equation of SVM_2 as $y = (x - c)/m$. We found that the values $m = 0.5$, $c = -15$ and $T_N = 5$ yielded the maximum values of precision equal to

98% and recall equal to 98% for our data set.

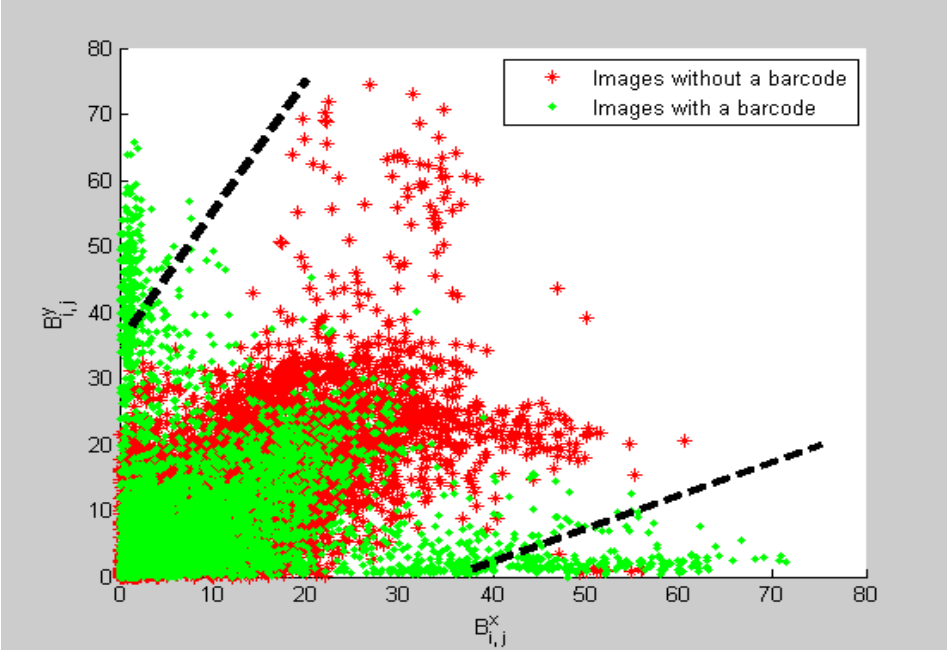


Figure 3.9. Barcode detection SVM.

3.4.4 Detecting if a Barcode is Cropped

It is possible that a barcode is detected by the barcode detection module but cannot be decoded by the barcode decoding module because it is cropped from one of the sides. A barcode can usually be decoded if it is cropped from the top or the bottom since it is redundant along those directions but it cannot be decoded at all if it is cropped from the left or the right. It is important that the user knows that the barcode is cropped and the side along which it is cropped, so that she can position the camera properly. The barcode detection module is responsible for determining this and providing the user with this information.

Let the top left and bottom right points of the subimage $B_{i,j}$ be represented by (x_{TL}, y_{TL}) and (x_{BR}, y_{BR}), respectively. Let N_L, N_R, N_T and N_B denote four counters that keep track of the location of the barcode. These counters will help us in determining if the barcode is cropped from the left, right, top or the bottom of the image. These counters

are all set to zero initially and if subimage $B_{i,j}$ contains a barcode, they are incremented as follows:

$$N_L = N_L + 1 \text{ if } x_{TL} \leq X_L \tag{3.7}$$

$$N_R = N_R + 1 \text{ if } x_{BR} \geq X_R \tag{3.8}$$

$$N_T = N_T + 1 \text{ if } y_{TL} \leq Y_T \tag{3.9}$$

$$N_B = N_B + 1 \text{ if } y_{BR} \geq Y_B \tag{3.10}$$

$$\tag{3.11}$$

where X_L and X_R denote the x-coordinates of two imaginary vertical lines drawn somewhere on the left half and right half of the image, respectively. Similarly, Y_T and Y_B denote the y-coordinates of two imaginary horizontal lines drawn somewhere on the top half and bottom half of the image, respectively. Figure 3.10 shows these lines. A barcode is assumed to be cropped if any of the counters N_L, N_R, N_T and N_B are greater than or equal to some threshold N_C.

For our application, we use $N_C = 1$, $X_L = 0.1 \times w$, $X_R = 0.9 \times w$, $Y_T = 0.1 \times h$ and $Y_B = 0.9 \times h$, where w and h denote the width and the height of the input image, respectively.

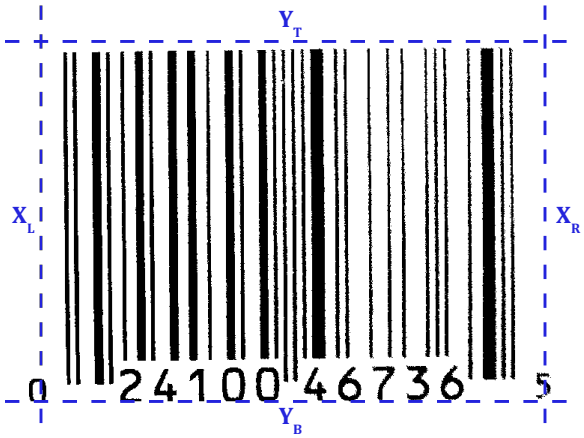


Figure 3.10. Imaginary boundaries on a barcode to detect if it is cropped.

To summarize, the input image is binarized and divided into $n \times n$ pixel subimages. For

each subimage, we compute the values B^x and B^y and increment N_p if the point (B^x, B^y) lies below SVM_1 or above SVM_2. These support vector machines are denoted by the lines $y = 0.5x - 15$ and $y = 2x + 30$, respectively. We declare that the image contains a barcode if $N_p \geq 5$. Finally, if the barcode cannot be decoded, we declare that it is cropped on the left if $N_L \geq 1$, on the right if $N_R \geq 1$, on the top if $N_T \geq 1$ and on the bottom if $N_B \geq 1$ and instruct the user to adjust the camera accordingly.

3.5 Barcode Localization

The barcode localization algorithm is responsible for finding the precise location of the barcode in the image and segmenting it from the image. It is different from the barcode detection algorithm, which just provides an indication of the presence of a barcode in the image. While both algorithms look for barcode regions in the image, the barcode detection algorithm is designed to be very fast, where as the barcode localization algorithm is designed to be precise.

Initially, we used ZXing [26] - an open source barcode scanning application for decoding barcodes. We found that ZXing performed better on images that contained only the barcode as compared to images that contained the barcode alongside other image elements like text and graphics. Thus, we found that the overall barcode scanning rate increased if we localized and segmented the barcode from images before decoding it with ZXing. We also observed a similar increase in performance when we used the barcode localization algorithm with our own barcode decoder. As showing in Figure 3.1, we decode the barcode on the entire image as well on the localized image. This results in a much better scanning performance, which is demonstrated by the experiment described in Section 4.3.

Barcode localization is a well known problem and many methods have been developed for it. Barcode localization methods can be roughly classified into two categories: spatial-domain based methods and frequency-domain based methods. Spatial-domain based methods [24,38–43] use features extracted directly from the image to locate the barcode where as frequency-domain based methods [44–46] use features extracted from the frequency domain (Fourier, Gabor, or Discrete Cosine Transforms) to localize the barcode.

Among the spatial-domain methods, Normand and Viard-Gaudin [38, 39] have developed a method that extracts regions exhibiting high densities of mono-oriented gradients to locate the barcode. Another method described in [40] uses contour analysis to detect lines of the barcode and uses this information to compute a bounding box that encloses the barcode. Neural networks can also be employed to detect barcodes in unconstrained images and the algorithm described in [41] describes such a system. Another method proposed in [24] uses a block-based approach where the image is divided into non-overlapping blocks, which are then binarized and skeletonized. Connected components are extracted from each block and analyzed to localize the barcode. Among the frequency domain methods, the image is processed using a Discrete Cosine Transform (DCT) in [44] to extract features, which are then classified to extract the barcode region. Jain and Chen [45] use Gabor filters and neural networks to localize the barcode. The system described in [46] uses morphological operators like erosion and dilation to extract key regions of interest from the image to localize the barcode. Fiducials can also be used to locate the barcode easily. An example of such a system can be found in [27].

3.5.1 Alternating Frequency and Vertical Continuity

As mentioned earlier, an image typically contains elements like graphics, text, and background alongside a barcode. To localize the barcode in the image, we have to differentiate it from these image elements. In Section 3.4, while discussing barcode detection, we used the image gradients G_x and G_y to differentiate the barcode from these elements. However, this method, though accurate in detecting the presence of a barcode, is not accurate enough to give the precise location of the barcode in the image.

Let us consider two properties of a barcode - *alternating frequency* and *vertical continuity*, which will help us in characterizing it. Alternating frequency is defined as number of zero-to-one and one-to-zero transitions in a one pixel wide binarized bit string representation of an image. In a bit string representation, black pixels are denoted by 0 and a white pixels are denoted by 1. Vertical continuity is defined as the continuity of black and white lines along the y-axis. Consider two one-pixel wide lines A and B placed over the barcode as

shown in Figure 3.11. The alternating frequency of a line can be found out by counting the number of zero-to one and one-to-zero transitions in the bit strings. The vertical continuity can be estimated as the length of the longest common subsequence between the two bit strings.

Both the UPC as well as MSI barcodes contain a large number of alternating black and white parallel lines. If we assume the barcode lines to be vertically oriented, we can observe that this region will exhibit large alternating frequency (due to the alternating black and white lines) and large vertical continuity (due to the lines being vertical and parallel to each other). We can thus define the barcode as a region that exhibits high alternating frequency and high vertical continuity.

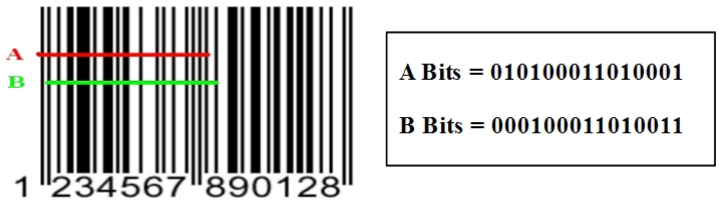


Figure 3.11. One pixel wide lines on barcode.

3.5.2 Barcode Localization Algorithm

Figure 3.12 shows an overview of the barcode localization algorithm. The first step is to downscale the image to a manageable size (say 320 pixels by 240 pixels). The second step (passing the image through a line filter) runs in $O(wh)$ time, where w is the width of the image and h is the height of the image. Thus, downscaling the image prior to processing it with a line filter makes the algorithm run faster without a big loss of precision.

The second step of the process is to pass the image through a line filter. The line filter is designed to let vertical lines in the image pass through and filter out everything else. Figure 3.13 shows an example of an image and its line filtered output. Since we assume that the barcode lines are always vertically oriented, they will pass through the filter along with other vertical lines in the image and everything else will be filtered out. In reality, the interactive camera alignment mode ensures that the barcode lines are oriented

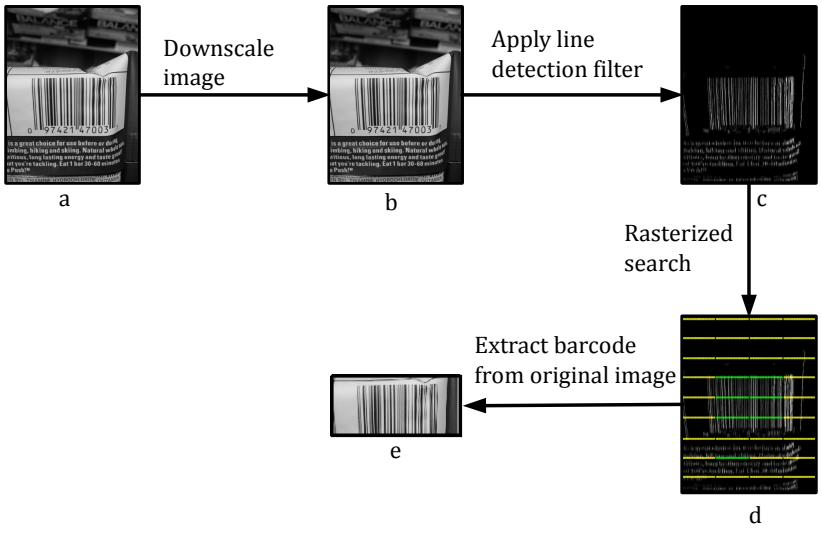


Figure 3.12. Overview of the barcode localization algorithm.

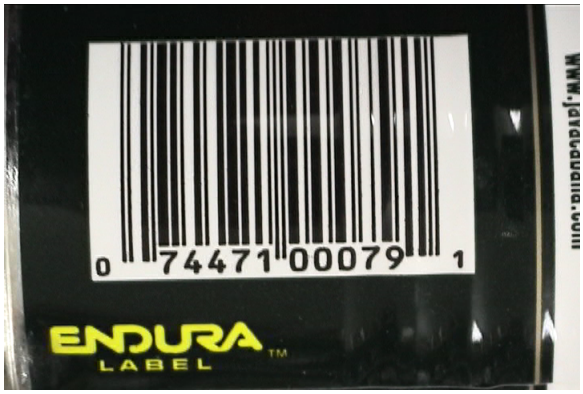
either vertically or horizontally but not along any other angle. To keep the assumption about barcode lines always being oriented vertically valid, the original image as well as a 90 degree rotated copy of the image are processed through the barcode localization and decoding algorithms.

Earlier, we used a Gabor filter as the line filter. A Gabor filter is a band pass filter for unidimension signals. It is a type of linear filter that is defined as a product of a Gaussian signal with a complex sinusoid. It us defined as:

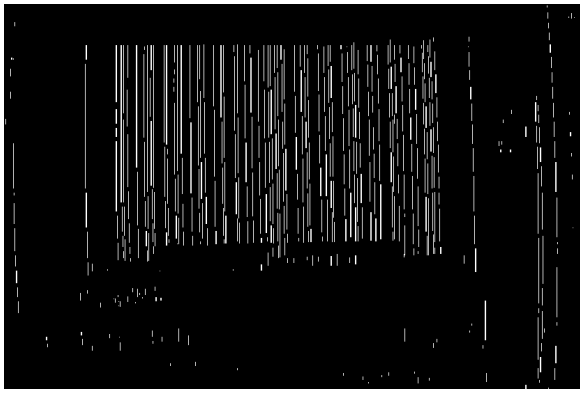
$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = exp(\frac{-x'^2 + \gamma^2 y'^2}{2\sigma^2})cos(2\pi\frac{x'}{\gamma} + \psi)) \tag{3.12}$$

where $x' = xcos\theta + ysin\theta$ and $y' = -xsin\theta + ycos\theta$.

Here, λ represents the wavelength of the cosine factor, θ represents the orientation of the normal to the parallel stripes of a Gabor function, ψ is the phase offset, σ is the sigma of the Gaussian envelope and γ us the spatial aspect ratio and specifies the ellipcity of the support of the Gabor function. For our application, we used $\sigma = 5$, $\gamma = 1$, $\psi = 0$ and $\lambda = 42.25$. Since we assume the barcode lines to be vertical, $\theta = 0$.



(a) Original Image



(b) Line Filtered Image

Figure 3.13. Line filter output of an image

To obtain the Gabor filtered image, we first apply a Canny edge detector to the original image and then apply the Gabor filter on the output of the Canny edge detector. The output of the Gabor filter is then converted to a binary image by applying a simple thresholding approach. We use 0.9 as a threshold and so any Gabor filtered responses whose values are bigger than 0.9 will be set as 255's (white). The other smaller responses will be set as 0's (black).

Figure 3.15 shows an example of an image and its Gabor filtered output. It can be observed that the Gabor filter only passes vertical lines in the image and filters out everything else. The Gabor filter approach produces good quality line filtered outputs but it is computationally expensive and so we use another method to produce a line filtered

-1	2	-1
-2	4	-2
-3	6	-3
-4	8	-4
-5	10	-5
-6	12	-6
-7	14	-7
-6	12	-6
-5	10	-5
-4	8	-4
-3	6	-3
-2	4	-2
-1	2	-1

Figure 3.14. Line filter kernel.

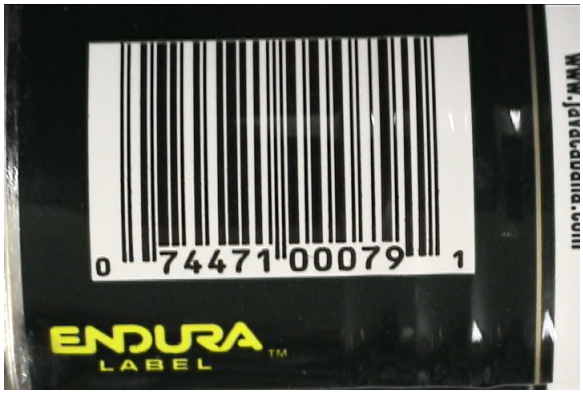
image. Our line filter is derived from [47]. This filter is obtained by convolution of a 13×3 kernel shown in Figure 3.14 through the entire image. A generalized $m \times n$ version of this kernel [48] is described below:

$$f[i][j] = \begin{cases} (\frac{m+1}{2} - |i|) \times \frac{n^2-1}{4} & \text{if } j = 0 \\ -(\frac{m+1}{2} - |i|) \times (\frac{n+1}{2} - |j|) & \text{if } j \neq 0, \end{cases}$$

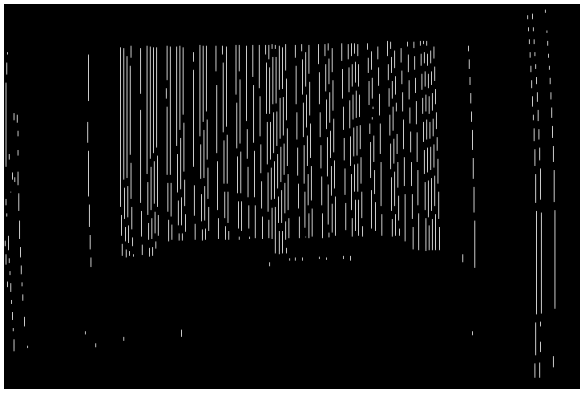
$$\text{where } -m/2 \leq i \leq m/2 \text{ and } -n/2 \leq j \leq n/2.$$

Cell phone processors are optimized for integer arithmetic and since our kernel uses only integers as opposed to floating point values used in the Gabor filter, it runs much faster.

The third step in the barcode localization process is to identify regions exhibiting high alternating frequency and high vertical continuity in the line filtered image. To perform this step two parallel horizontal lines A and B of width l_w are placed on the line filtered image at positions $p_a = (x, y)$ and $p_b = (x, y + y_o)$, where y_o is the vertical offset between the two lines. The values of alternating frequency $a(x, y)$ and vertical continuity $v(x, y)$ are computed for the two lines and stored. The x and y values of p_a and p_b are incremented



(a) Original Image



(b) Gabor Filtered Image

Figure 3.15. Gabor filter output of an image

and the entire image is scanned in a rasterized fashion. A 2-D histograms H is created to represent areas having alternating frequency and vertical continuity. For each position $p(x, y)$ of the lines, the histogram is updated as follows:

$$m = x/w \times M \tag{3.13}$$

$$n = y/w \times N \tag{3.14}$$

$$H(m, n) = H(m, n) + 1 \text{ if } a(x, y) \geq A_T \text{ and } v(x, y) \geq V_T \tag{3.15}$$

where, M & N are the number of bins in H along the x & y axes, respectively, w & h are the height and the width of the line filtered image and A_T & V_T are two thresholds.

Once the histogram is updated, it is thresholded to create a new histogram H_T as follows:

$$H_t(m,n) = \begin{cases} 1 & \text{if } H(m,n) \geq T \\ 0 & \text{if } H(m,n) < T \end{cases}$$

where, T is some threshold.

This step is performed to reduce noise in the histogram. This thresholded histogram is now scanned to identify connected components of ones. These components represent areas having high alternating frequency and vertical continuity, which most likely represents a barcode.

The final step of the localization process is to segment the barcode region from the original image. Let $p_{TL} = (x_{TL}, y_{TL})$ and $p_{BR} = (x_{BR}, y_{BR})$ represent the top-left and bottom-right corners of the connected component in the histogram. Let N be the scale factor by which the original image was scaled down in the first step. The top-left corner $p^B_{TL} = (x^b_{TL}, y^b_{TL})$ and bottom-right corner $p^B_{BR} = (x^b_{BR}, y^b_{BR})$ of the barcode region in the original image can be computed as follows:

$$x^b_{TL} = x_{TL}/M \times w \times N \tag{3.16}$$

$$y^b_{TL} = y_{TL}/M \times h \times N \tag{3.17}$$

$$x^b_{BR} = x_{BR}/M \times w \times N \tag{3.18}$$

$$y^b_{BR} = y_{BR}/M \times h \times N \tag{3.19}$$

where, w and h represent the height and the width of the line-filtered image.

3.6 Barcode Decoding

The barcode decoding module is responsible for converting the barcode image to a string of characters, which represent the barcode. Barcode decoding is a well known problem and many solutions are developed for it. Most barcode decoding algorithms, including our own, are scanline based [28, 49] but other solutions based on Hough transform [50] and neural networks [51] exist. A scanline represents a one pixel wide slice of the image. It is analogous to drawing a one pixel wide line on the image and extracting the pixels that lie on that line. We employ two scanlines S^I and S^B, which are obtained from the original image I and the binarized image B, respectively. Since most linear barcodes are encoded by alternating black and white lines of varying widths, we introduce two new notations *line widths (LW)* and *line colors (LC)*, which will help us in decoding the barcode. LW is used to denote the width of each line in the barcode and LC is used to denote the colors (black or white) of those lines. Before decoding, both S^I and S^B are converted to their respective line widths and line colors notations.

3.6.1 Line Widths and Line Colors Notations

Let us define two notations - *Line Widths* (LW) and *Line Colors* (LC), which can be used to represent linear barcodes. Line widths represent the widths of each line in the barcode and line colors represent their respective colors (0 for black and 1 for white). Each barcode can now be represented using the pair (LW, LC). Consider a scanline S on a part of a linear barcode shown in Figure 3.16. For simplicity, this figure assumes that the scanline starts and ends on the barcode. In reality, the scanline will also contain other parts of the image like graphics and text. If we assume the width of the thinnest line to be one, we can represent this barcode as (LW, LC), where $LW = 1111221113211$ and $LC = 1010101010101$.

To improve our chances of decoding the barcode, we employ two scanlines S^I and S^B obtained from the grayscale and binarized images, respectively. Both these scanlines have to be converted to their respective line widths and line colors notations for decoding the barcode.

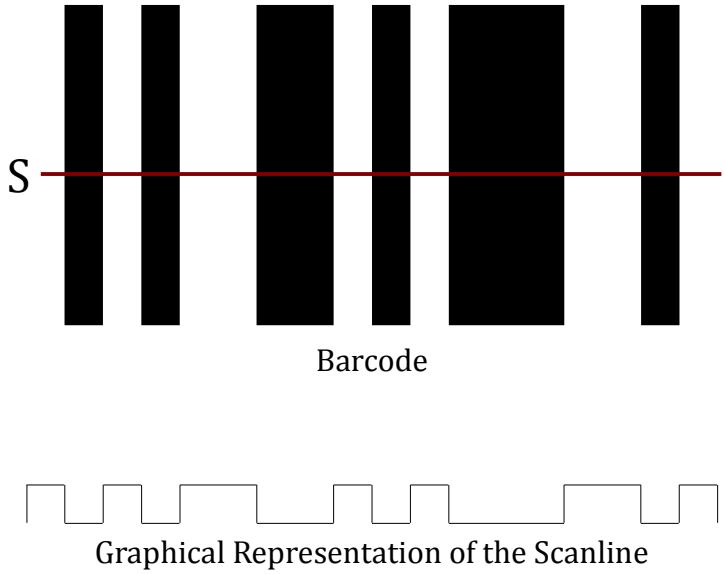


Figure 3.16. Line widths and line colors representation for a linear barcode.

Let us first look at how S^I is converted to its equivalent line widths LW^I and line colors LC^I notations. Consider Figure 3.17, which shows a graphical representation of the scanline. The scanline S^I can be represented as a collection of points $\{s_0, s_1, \ldots, s_{n-1}\}$, where $s_j = (j, i_j)$ is the j^{th} point on the curve corresponding to the j^{th} pixel in the scanline and i_j is its intensity or grayscale value. Let $m = \text{mean}(i_0, i_1 \ldots, i_{n-1})$ represent the mean intensity of the scanline and let the line M having the equation $y = m$ represent the mean intensity graphically. We have to compute the values of the line widths $LW^I = \{lw_0, lw_1, \ldots, lw_{k-1}\}$ and line colors $LC^I = \{lc_0, lc_1, \ldots, lc_{k-1}\}$, where lw_j and lc_j are the width and the color of the j^{th} barcode line, respectively.

Let us consider an example where we will derive the value of lw_0 and lc_0. Let $P = \{p_0, p_1, \ldots, p_k\}$, where $p_j = (x_j, y_j)$, denote the set of all the intersection points of S^I with M. As shown in Figure 3.17, the value of lw_0 can be computed as $lw_0 = x_1 - x_0$. It would be difficult to compute the exact locations of p_0 and p_1 (this is true for all the intersection points). However, as shown in Figure 3.17, we can approximate the location of p_0 with the

intersection of line L_0 and M and the location of p_1 with the intersection of L_1 and M. Here, line L_0 is a line that connects a pair of two points (s_1 and s_2) that lie on the scanline and are on either side of M. Similarly, line L_1 connects the next pair of such points (s_9 and s_{10}). Since s_2 lies above s_1 ($i_2 > i_1$), we can set $lc_0 = 1$.

In general, we first have to find a pair of points, $s_j = (j, i_j)$ and $s_{j+1} = (j+1, i_{j+1})$, such that they lie on either side of M. This can be performed by sequentially going through S^I and finding a pair of points such that $i_j < m$ & $i_{j+1} > m$ or $i_j > m$ & $i_{j+1} < m$. The equation of the line L can now be computed as follows:

$$y = (i_{j+1} - i_j)(x - j) + i_j \tag{3.20}$$

The equation of line M is $y = m$. The intersection point $p_k = (x_k, y_k)$ of L and M can be computed as follows:

$$x_k = \frac{m - i_j}{i_{j+1} - i_j} + j \tag{3.21}$$

$$y_k = m \tag{3.22}$$

The width lw_k of the k^{th} barcode line can be computed as follows:

$$w_k = x_{k+1} - x_k \tag{3.23}$$

The value of lc_j is obtained as follows:

$$lc_j = 1 \text{ if } i_{j+1} > i_j \text{ and } 0 \text{ otherwise} \tag{3.24}$$

To obtain LW^B, groups of black and white pixels in S^B are replaced with their run lengths. LC^B represents the corresponding colors (black or white) of elements in LW^B. For example, if $S^B = \{1,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0\}$, $LW^B = \{5,4,4,3\}$ and $LC^B = \{1,0,1,0\}$.

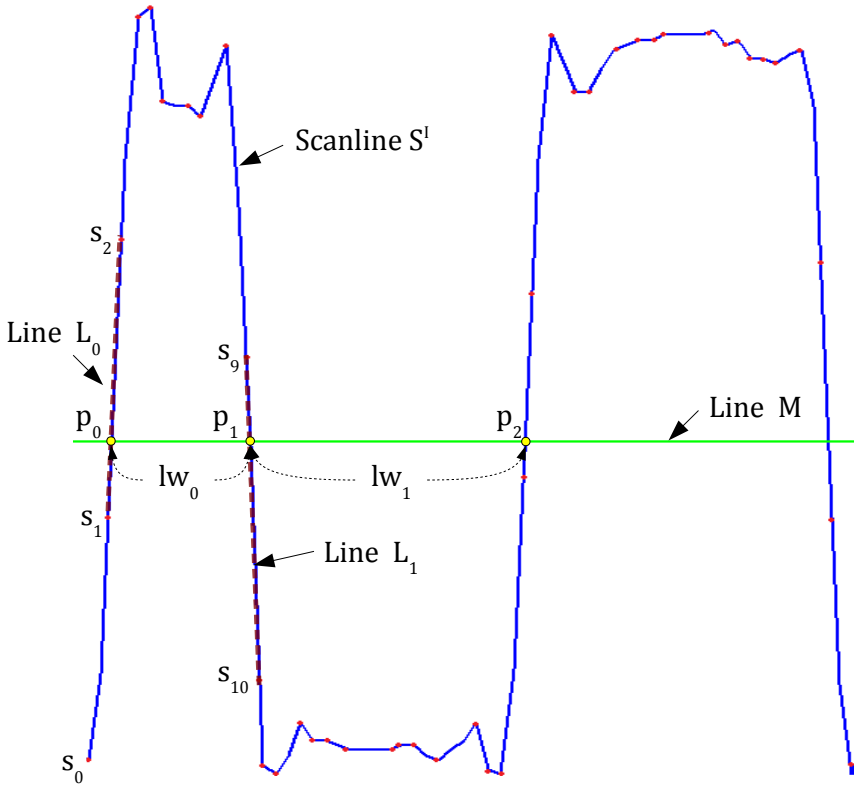


Figure 3.17. Intensity curve and mean intensity.

In practice, LW^I and LC^I are generated and decoded first. If a valid barcode is not found, LW^I and LC^I are reversed and decoded. This step ensures that a barcode will be decoded even if it is rotated by 180 degrees. If we are still unable to find a barcode, LW^B and LC^B are generated and decoded, both originally and after reversal. As will be discussed later in section 4.3.1, LW^I and LC^I have a greater probability of decoding a barcode as compared to LW^B and LC^B and so are given preference over LW^B and LC^B.

The above procedure shows how a single scanline is decoded in the image. In practice, multiple scanlines horizontal and vertical are generated until a valid barcode is obtained.

The barcode decoding process uses only the line widths and line colors format and henceforth all discussion will assume that the scanlines are converted to this format.

3.6.2 Decoding UPC Barcodes

UPC (Universal Product Code) is the most widely barcode format for labeling products in North America. There are several versions of UPC barcodes - UPC-A, UPC-B, UPC-C, UPC-D, UPC-5 and Zero Compressed UPC [52]. UPC-A is the most common version of the UPC barcode and henceforth all discussion will be related only to this version of the UPC barcode.

A UPC barcode is a type of linear or 1-D barcode, which is encoded by alternating black and white lines of varying widths. The UPC barcode consists of the following components - a start pattern (S), two groups of six digits (D) separated by a middle pattern (M) and an end pattern (E). These components are arranged as: $SDDDDDDMDDDDDDE$. Let the width of the thinnest line in the barcode be w and let the thinnest black and white lines be represented by B and W, respectively. The start and end patterns are encoded as BWB. Each digit is encoded by four lines, whose combined width is $7w$. The middle pattern is encoded as $WBWBW$. Thus, each UPC barcode contains a total of 3 (start pattern) + 4 (per digit) $\times$ 12 (digits) + 5 (middle pattern) + 3 (end pattern) = 59 lines and has a combined width of $3w$ (start pattern) + $7w$ (per digit) $\times$ 12 (digits) + $5w$ (middle pattern) + $3w$ (end pattern) = $95w$. Table 3.1 shows the encodings for the three patterns and for the digits $0 - 9$ in the line widths and line colors format.

As shown in Figure 3.18, the scanline may contain image elements like text, graphics and background along with the barcode. This is true even if the barcode is localized since the scanline will at least contain some background region along with the barcode. The first step of the decoding process is to identify the start and the end of the UPC barcode within the scanline. Every UPC barcode contains a start pattern and an end pattern. We can locate the start and the end of the barcode by locating these patterns. Let the start and end pattern indices be denoted by s and e, respectively. These can be computed as follows:

Figure 3.18. Scanline on a product.

$$s = argmax(lw_{i-1} - std(lw_i, lw_{i+1}, lw_{i+2})) \text{ and } c_i = 0 \tag{3.25}$$

$$e = argmax(lw_{i+3} - std(lw_i, lw_{i+1}, lw_{i+2})) \text{ and } c_i = 0 \tag{3.26}$$

The start pattern is encoded as three alternating black and white lines having unity width. Thus, if s represents the start pattern index, then lw_s, lw_{s+1} and lw_{s+2},will be approximately equal and the value of $std(lw_s, lw_{s+1}, lw_{s+2}))$ will be low. Also, the value of lw_{s-1} will be high because it represents white space to the left of the barcode. The value of $(lw_{s-1} - std(lw_s, lw_{s+1}, lw_{s+2}))$, will be maximum in LW and this can be used to locate the start pattern index. A similar analysis can be applied to derive the value of the end pattern index.

Since e represents the index of the end pattern, there are a total of $59 - 3 = 56$ (subtracting 3 for the end pattern) lines between s and e. Thus, further decoding of the

Table 3.1. UPC Barcode Symbology

Digit/Pattern	**Template of line widths**
Start Pattern	111
End Pattern	111
Middle Pattern	11111
Digit 0	3211
Digit 1	2221
Digit 2	2122
Digit 3	1411
Digit 4	1132
Digit 5	1231
Digit 6	1114
Digit 7	1312
Digit 8	1213
Digit 9	3112

barcode is proceeded only if $e - s = 56$. If we assume each line to be one pixel wide, the sum of the widths for all the lines in the UPC code is 3 (start pattern) + 3 (end pattern) + 5 (middle pattern) + 7 (per digit) $\times$ 12 (digits) = 95. The set of line widths LW can now be normalized by dividing it by 95. The index i of the j^{th} digit d_j in the line widths can be calculated as follows:

$$i = \begin{cases} s + (j-1) \times 4, \text{ if } j \leq 6 \text{ and} \\ s + (j-1) \times 4 + 5, \text{ if } j > 6 \text{ (add 5 lines for the middle pattern)} \end{cases} \tag{3.27}$$

Since each digit is encoded by four lines, the value of the d_j can now be found out as:

$$d_j = argmin \sqrt{(\Sigma(lw_{i+j} - T^k_{i+m})^2)} \tag{3.28}$$

where, $0 \leq m < 4$ and T^k represents the template of line widths for the j^{th} digit as described in Table 3.1. The twelfth digit of the barcode represents the checksum of the preceding eleven digits and can be used to verify if the barcode was correctly decoded or not.

Table 3.2. MSI Barcode Symbology

Digit/Pattern	**Template of line widths**
Start Pattern	21
End Pattern	121
Digit 0	12121212
Digit 1	12121221
Digit 2	12122112
Digit 3	12122121
Digit 4	12211212
Digit 5	12211221
Digit 6	12212112
Digit 7	12212121
Digit 8	21121212
Digit 9	21121221

3.6.3 Decoding MSI Barcodes

A MSI barcode [53] is a variable length barcode that is arranged as follows: $SDD...DE$, where S represents the start pattern, D represents a digit and E represents the end pattern. Table 3.2 shows the encoding of the start and end patterns as well as of digits 0 to 9 in the line widths format. The total number of lines in the barcode is equal to 2 (start pattern) + 3 (end pattern) + 8 (per digit) $\times n$ digits $= 8n + 5$.

To decode the barcode from the line widths LW, we have to find the start pattern index (s) and the end pattern index (e) within LW. The start pattern index and the end pattern index can be found as follows:

$$s = argmax(lw_{i-1} - \sqrt{(\Sigma(lw_{i+j} - T_j^S)^2)}), \text{ where } c_i = 0 \text{ and } 0 \leq j < 2 \tag{3.29}$$

$$e = argmax(lw_{i+3} - \sqrt{(\Sigma(lw_{i+j} - T_j^E)^2)}), \text{ where } c_i = 0 \text{ and } 0 \leq j < 3 \tag{3.30}$$

In the above equation, T^S and T^E represent the templates of line widths for the start and end patterns as described in Table 3.2.

Since e represents the index of the end pattern, there are a total of $8n + 5 - 3$ (subtracting 3 for the end pattern) lines between s and e. Thus, further decoding of the barcode is proceeded only if $e - s = 8n + 2$. If we assume that each line in the barcode is one

pixel wide, the total width of each digit is 12 pixels. Thus, the total width of the barcode is 3 (start pattern) + 4 (end pattern) + 12 $\times n$ (for n digits in the barcode) = $12n + 7$ pixels.. The set of line widths W can now be normalized by dividing it by $12n + 7$ (assuming we know the value of n). The index i of the j^{th} digit d_j in the width string can be calculated as: $i = s + (i - 1) \times 8$.

The value of the j^{th} digit d_j can now be found out as follows:

$$d_j = argmin(\sqrt{\Sigma(w_{i+k} - T_k^m)^2}) \tag{3.31}$$

where, $0 \leq k < 8$ (each digit is represented by eight lines), $0 < m <\leq 9$ (for digits 0 to 9) and T^m represents the template of line widths for the m^{th} digit as described in Table 3.2. The last digit of the barcode represents the checksum of the preceding eleven digits and can be used to verify if the barcode was correctly decoded or not. A modulo 10 checksum is most commonly used and it can be computed by the Luhn algorithm [54].

CHAPTER 4

EYES-FREE BARCODE SCANNER EXPERIMENTS

4.1 Introduction

The aim of ShopMobile 2 is to provide a hand-held eyes-free replacement for a traditional barcode scanner. We performed several experiments to evaluate the performance of our barcode scanner both as a whole and in terms of its component parts. This chapter describes all such experiments and discusses their results. There is an inherent risk associated with retrieving products from shelves as they may slip and cause injury to the participant. ShopMobile 2 currently does not have a system in place that can warn shoppers of potentially dangerous products like large and heavy items. We try to minimize the risk of injury by excluding certain types of products or handing them to participants whenever possible.

The remainder of this chapter is organized as follows: Section 4.2 describes the experiment performed to evaluate the barcode detection module. Section 4.3 describes the experiments performed to evaluate the barcode localization module and finally, Section 4.4 describes the experiments performed to evaluate the barcode scanner as a whole.

4.2 Barcode Detection Experiments

The barcode detection module detects the presence of a barcode in an image. The motivation for this module was an experiment (see Section 4.6 for details) that was performed at the Smith Kettlewell Eye Research Institute. In this experiment, two VI individuals were asked to scan barcodes on ten products. We found that the participants spent a lot of time trying to scan barcodes on surfaces that did not contain a barcode. This motivated us to develop the barcode detection module, which allows a VI user to quickly scan a surface and determine if it contains a barcode or not. The interested reader is directed to Section 3.4

Table 4.1. Results of Software Experiments on the Barcode Detection Module

	Images with a barcode	**Images without a barcode**
Barcode detected	50	1
Barcode not detected	1	72

for more information on this module.

We performed two experiments to evaluate the performance of the barcode detection module. The first experiment a simulation and the second experiment was a real-world test performed with three blindfolded sighted people. In the first experiment, a total of 124 images of products were taken in a grocery store out of which 51 images contained a barcode and 73 did not. These images were manually classified into two sets - images containing a barcode and images without a barcode and the barcode detection algorithm was run on both the sets. The results of this test are shown in 4.1.

The values of precision and recall can be computed as follows:

$$precision = \frac{tp}{tp + fp} \tag{4.1}$$

$$recall = \frac{tp}{tp + fn} \tag{4.2}$$

where, tp, fp and fn denote the number of true positive, false positive and false negative results, respectively. Since we have $tp = 50$, $fp = 1$, $fn = 1$, we can compute precision $= 50/(50 + 1) = 98\%$ and recall $= 50/(50 + 1) = 98\%$.

The second experiment was designed to measure the amount of time it took a person to find (detect and not decode) a barcode on an actual product. Three blindfolded sighted participants were asked to find the UPC barcode on ten products (eight boxes, one can and one bottle) using the phone. All three participants were graduate students from the Computer Science Department at Utah State University.

Each participant was explained that they had to find UPC barcodes on the products using the phone. They were then blindfolded and trained on using the system. Specifically, they were taught to align the phone with the product and then keep the phone aligned with

the product at all times using the camera alignment module. They were then taught how to move the phone away from the product such that the phone was approximately six to eight inches from the product and how to move the phone parallel to the product for finding the UPC barcode. They were asked to stop if they found a barcode (phone beeped) or switch to a different side if the current side did not contain a barcode (the phone did not beep) and repeat the procedure. Each participant was given a set of products (different from the test set) to practice. During this time, we observed how they were using the system and corrected them if they made mistakes. There was no time limit on how long each participant could practice. Once they were comfortable with the system, they were asked to perform the experiment using the test products. The entire experiment was recorded using a video camera and the results were analyzed later using the recordings. After the experiment, each participant was asked some open ended questions and their responses were noted.

Table 4.2 shows the amount of time each participant (Alice, Bob, and Carl - names changed to protect privacy) took to find the barcode on the ten products. It can be observed that 28 out of 30 barcodes were detected correctly. Alice was not able to detect a barcode on the bottle and there was one false positive in case of Bob. The mean values for detecting a barcode for Alice, Bob, and Carl are 31, 32.11, and 26.1 seconds, respectively and the median values for the three are 20 seconds, 33 seconds, and 13 seconds, respectively. It can be observed that all three participants spent a lot of time detecting the barcode on product 9 (Saltine Crackers). The barcode on this product was located on the side instead of the bottom and so the participants took some time in scanning the other surfaces that did not contain the barcode.

To scan a barcode, the participant had to align the phone with one of the sides of the product and move it approximately six to eight inches away from the product. The final distance between the phone and the product is extremely important. If the phone is too close to the product, the camera will not be able to focus on the product and the barcode will not be detected. Similarly if the phone is too far from the product, the barcode will occupy a small area in the image and will not be detected. We observed that Alice was not

Table 4.2. Times Taken to Find a Barcode Using the Barcode Detection Module

	Alice	**Bob**	**Carl**
Product 1	14	32	13
Product 2	16	13	34
Product 3	29	45	10
Product 4	-	39	33
Product 5	37	FP	9
Product 6	61	49	18
Product 7	20	13	13
Product 8	15	12	6
Product 9	78	53	115
Product 10	9	33	10

able to position the phone properly with respect to the product and that caused her to take additional time in detecting the barcode. Training and practice plays a very important part in this and we believe that participants get better at positioning the phone with practice.

4.3 Barcode Localization Experiments

Our experience with the barcode scanner led us to believe that the performance of the barcode detection module improves if it is presented with a localized barcode image in addition to the entire image. This experiment was designed to evaluate the contribution of the barcode localization module.

We scanned UPC barcodes on 68 randomly chosen products (boxes, cans and bottles) in a real store. The barcode scanning algorithm shown in Figure 3.1 was modified such that for all images, the barcode decoding module ran on the original image as well as the localized image irrespective of whether the barcode was decoded on the original image or not. Figure 4.1 shows the results of this experiment. It can be observed that the barcode decoding stage was able to decode barcodes on 32 out of 68 products using the original image. It was able to decode 36 out of 68 images using just the localized image from the barcode localization module and 22 out of 64 images using both the methods. It can be observed that the addition of the barcode localization module increased the barcode decoding rate by 25.92%.

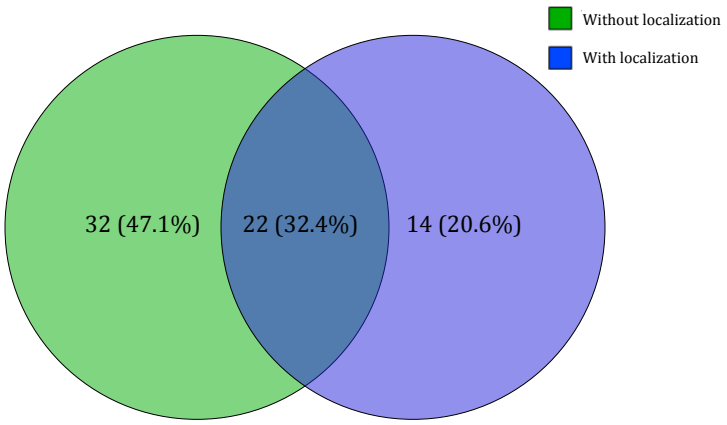


Figure 4.1. Contribution of the barcode localization module to barcode scanning.

4.3.1 Barcode Decoding Experiments

The barcode decoding module is designed to work with grayscale S^I as well as two-level binary S^B scanlines. The motivation behind this is to take advantage of the extra information present in grayscale scanlines and the reduced overall luminosity difference in the two-level binary scanlines (see Section 3.6.1 for details). Our assumption is that employing these two different types of scanlines would increase the barcode detection rate as compared to employing just one type of scanline. To test this assumption, we decoded UPC barcodes on 67 products in a real grocery store. For each barcode, the phone automatically logged whether the barcode was decoded using S^I, S^B or both.

Figure 4.2 shows the results of this experiment. It can be observed that S^I was responsible for decoding 53 barcodes out of 67 (79%) and S^B was responsible for decoding 26 barcodes out of 67 (38%). Both S^I and S^B were responsible for decoding 12 barcodes out of 67 (18%). Thus, it can be observed that employing two different types of scanlines increases the barcode decoding performance by at least 21%. We can also observe that S^I is able to decode more barcodes than S^B. This could be because the extra grayscale information present in S^I allows us to estimate the line widths with sub-pixel accuracy, which is not possible with S^B. The sub-pixel accuracy results in better precision and thus S^I is more robust to errors introduced by JPEG artifacts in the image.

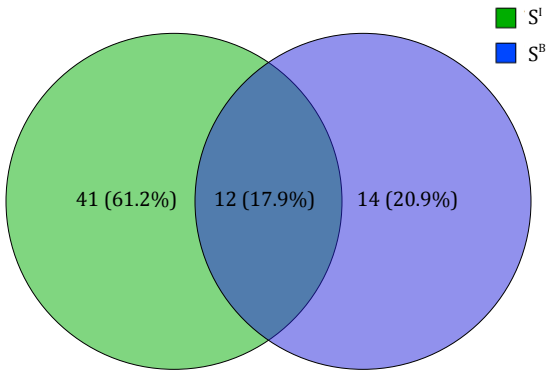


Figure 4.2. Number of barcodes decoded by S^I and S^B.

4.4 Barcode Scanning Experiments

Previous experiments were responsible for evaluating the contributions of specific modules and techniques in the barcode scanner. This section describes four experiments that test the barcode scanner as a whole.

4.5 Barcode Scanning Experiment - I

In this experiment, we assembled and stocked three plastic shelves with twelve empty boxes of real-world products to simulate a grocery aisle. The assembled shelf can be viewed in Figure 4.3. We placed MSI barcodes on shelves beneath products and entered their information into the Barcode Connectivity Matrix (BCM). This experiment was performed with an early version of the barcode scanner which operated in manual mode, where users had to press a button on the screen to capture the image, which was localized and decoded. The barcode scanner did not contain the interactive camera alignment module or the barcode detection module and used ZXing for barcode decoding.

Participants were given a 10-minute training session in which they were shown how to use the system. Specifically, they were shown how to align the phone with the product and then move it away before taking a picture. In case of MSI barcodes, participants were shown how to locate barcodes haptically, align the phone with the barcode and then move it away before taking a picture. Participants were also shown how to retrieve products

Figure 4.3. Simulating a grocery aisle using plastic shelves stocked with empty boxes of real products.

using the system and verify that they had picked the correct products. Each participant was then asked to retrieve and verify ten randomly selected products. For each participant, the system logged the image associated with each scan and the result of the scan.

All participants were able to retrieve and verify all products successfully. Figure 4.4 shows the mean number of scans taken by each participant to retrieve and verify products and Figure 4.5 shows the median number of scans taken by each participant to retrieve and verify products. Since participants had to push a button on the screen each time they wanted to scan a barcode, the results are shown in terms of the number of scans and not in terms of the actual time taken. In general, the median values for product retrieval are lower than the median values for product verification. It can be observed that for participant

5, the median value for product retrieval is zero. For every product, the system would announce its name and location on the shelf, for example, "Now looking for Chamomile Tea on Shelf Two, Position Three." Participants had the option of using the BCM to find products or just locating them haptically. After a few tries, participant 5 chose to just locate products haptically instead of using the BCM. Thus, his median number of scans for retrieving products is zero.

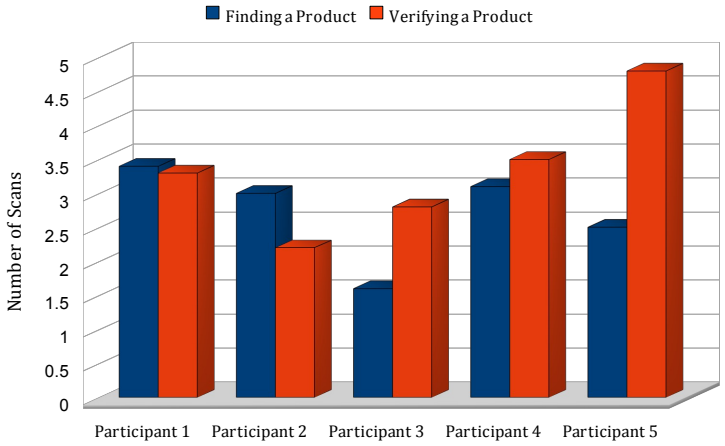


Figure 4.4. Mean number of scans required to retrieve and verify products.

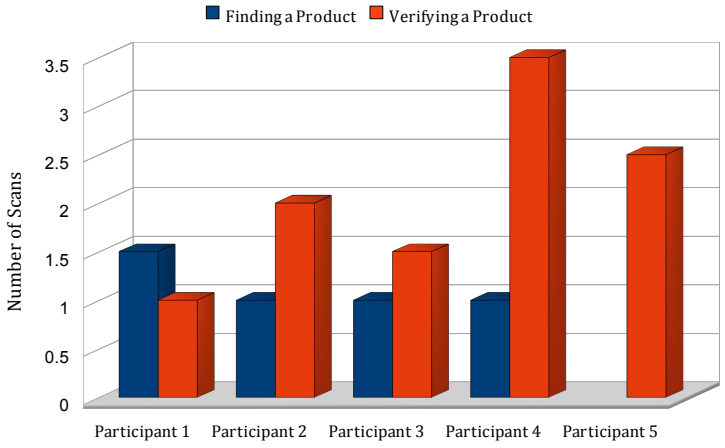


Figure 4.5. Median number of scans required to retrieve and verify products.

In our informal post-experiment interviews, we found that participants found our interface simple but too basic to use. The interface consisted of a single button encompassing the entire touch screen of the Google Nexus One phone. The barcode decoding procedure

was initiated either by finger tapping on any part of the screen or by tapping on the phones trackball, a small joystick-like hardware component below the touch screen. All participants preferred tapping the screen to tapping the trackball, which resulted in many accidental images with no barcodes present in them, which negatively affected the barcode localization and decoding performance. The VI participant found it difficult to keep the phone aligned with the product while moving it away from the product and suggested that we incorporate some automatic alignment scheme. The feedback obtained from this study caused us to abolish the manual scanning mode and develop the video mode for scanning, where images are captured, localized and decoded automatically. The VI participant said that he found it difficult to keep the phone aligned with product while moving it away from the product. This caused us to develop the interactive camera alignment module, which helps the user in keeping the phone aligned with the product at all times.

4.6 Barcode Scanning Experiment - II

The second barcode scanning experiment was carried out with two participants at the Smith Kettlewell Eye Research Institute in San Fransisco, CA. Both the participants were completely blind staff electrical engineers. One had a cellphone and the other never owned or used one. This version of the barcode scanner operated in video mode, contained the interactive camera alignment module but did not contain the barcode detection module.

Both the participants were trained on using the system using two sample products (a plastic bottle, and a juice box). The training session lasted for approximately ten minutes for the first participant and eight minutes for the second. Once participants were comfortable in using the system, they were asked to scan UPC barcodes on ten products (a cereal box, a small tea box, two small juice bottles, a small milk carton, a Pringles tube, a toothpaste box, a larger juice bottle, a small water bottle, and a yogurt cup). The detection time for each product was recorded on the phone. When the participant could not detect the barcode for over five minutes, the detection was considered a failure and the participant was given the next product. The order of the products was randomized for each participant.

Table 4.3. Results of Barcode Scanning Experiment II

Product	**Participant 1**		**Participant 2**	
	Time Taken	**False Positives**	**Time Taken**	**False Positives**
Cereal box	33	0	40	0
Tea box	140	0	100	1
Small juice bottle 1	15	0	Fail	0
Small juice bottle 2	142	0	151	0
Milk carton	56	0	51	1
Pringles	37	0	111	0
Toothpaste	184	0	Fail	0
Large juice bottle	125	0	140	0
Water bottle	87	0	121	0
Yogurt cup	17	0	41	0

Table 4.3 shows the times taken by each participant to scan barcodes on all the products. It can be observed that participant 1 was able to scan all barcodes successfully and participant 2 was not able to scan two barcodes and had two false positives. The average barcode scanning times are 83.6 seconds and 93.4 seconds for participant 1 and 2, respectively.

Our aim is to provide the user with a barcode scanner that is comparable to a hardware barcode scanner but the barcode scanning times mentioned earlier are too large and the failure rate is too high. Upon analyzing the video, we found that the participants spent most of their time trying to scan barcodes on surfaces that did not contain a barcode. We conducted informal post-experiment interviews and participants mentioned that it would be beneficial to know which surface of the product contained a barcode so that they would not waste time on other surfaces. This caused us to develop the barcode detection module, which can quickly analyze a given image to determine the presence of a barcode and alert the user. In this experiment, the system would grab images continuously in video mode and try to localize and decode the barcode in each image. Both these processes are computationally expensive and this slowed down the system considerably. After the barcode detection module was developed, each image frame had to be processed only by the barcode detection module and the barcode localization and decoding modules ran only if a

barcode was detected. This caused the performance of the system to increase considerably. The other problem with the system was that the interactive camera alignment mode was manual, that is users had to manually turn it on to use it. The participants would just forget about turning it on and so they were not able to benefit from this module. Thus, in the final version of the barcode scanner, the interactive camera alignment module starts automatically when the user moves the phone away from the product. Thus, they are able to benefit from this module and make fewer alignment errors.

4.7 Barcode Scanning Experiment - III

This experiment was performed with the final version of the barcode scanning software, which included grabbing images in video mode, automatically starting the interactive camera alignment module, the barcode detection module as well as the barcode localization and decoding modules. This experiment was performed with one completely blind participant in our lab. The participant was explained the purpose of the experiment and trained on using the system. Once the participant was confident in using the system, he was asked to scan UPC barcodes on ten products. The entire experiment was recorded and the video was analyzed later on to determine the times taken to scan barcodes for each product.

Table 4.4 shows the times taken to scan UPC barcodes on ten products. It can be observed that the participant was able to scan UPC barcodes on all the products successfully. The mean and median times for scanning were 51.7 seconds and 28.5 seconds, respectively. It can be observed that the participant took a long time to scan product 3, which was a can. One probable reason for this is that since the barcode was placed on a curved surface, the barcode image had skew distortions and this made it difficult to decode the barcode. In situations like this, it is recommended that the shopper uses the teleassistance module.

4.8 Barcode Scanning Experiment - IV

The aim of this experiment was to investigate the times taken by a VI individual to scan UPC barcodes on actual products in a real-world supermarket. This experiment was performed at Fresh Market Supermarket in Logan, UT during regular business hours

Table 4.4. Results of Barcode Scanning Experiment III

Products	Scanning Times
Product 1	20
Product 2	29
Product 3	239
Product 4	15
Product 5	31
Product 6	75
Product 7	35
Product 8	21
Product 9	24
Product 10	28

and the subject chosen for this experiment was a completely blind employee of Utah State University. The subject was briefly trained on using the system and then asked to scan UPC barcodes on fifty products.

All products were selected prior to the experiment (five products per aisle across ten aisles) by a sighted individual who was not involved with this project. A product was selected if it had a twelve digit UPC barcode and if it was possible to hold it in hand while scanning. Thus, certain items like produce (no barcode), gum sticks (that contained only a six digit barcode) and sugar sacks (too heavy) were excluded from this experiment. The UPC barcodes on these products were not scanned during the selection process and thus we had no a-priori knowledge of how long it would take to scan a barcode on a particular product. Three types of products were included in this experiment - boxes, cans and bottles and the selection process was performed in a way so the number of boxes, cans and bottles would be approximately the same.

The subject was taken to an aisle and was given a product from this aisle. He would then align the phone with the product and scan the barcode. The subject continued to scan the barcode on the product until it was read out aloud by the system or he felt that he would not able to scan the barcode and wanted to switch to a different product. We recorded this experiment on video and analyzed the barcode scanning times for each product. We measured the barcode detection time, which was computed as the time between placing the

phone on the product and the first beep and the total scanning time, which was computed as the time between placing the phone on the product and the start of the barcode utterance.

The participant was able to scan barcodes on 46 out of 50 products. The subsequent analysis discusses the times required to detect and scan barcodes only on the 46 products that were successfully scanned. Figures 4.6 and 4.7 show the time required to detect and scan barcodes, respectively. It can be observed that the participant took an average of 45 seconds to detect the barcode and 54 seconds to scan it on the product. The median times for both detecting barcodes and scanning them are lower at 26 seconds and 40 seconds, respectively. The maximum time to scan a barcode was 3 minutes and 24 seconds. The product in question was a box and the participant was not able to locate the barcode on it. We eventually gave him a hint of the side that contained the barcode and he was able to scan it successfully. We would like to take this opportunity to reiterate that such situations are possible and the shopper is better off in relying on the tele-assistance module in such cases.

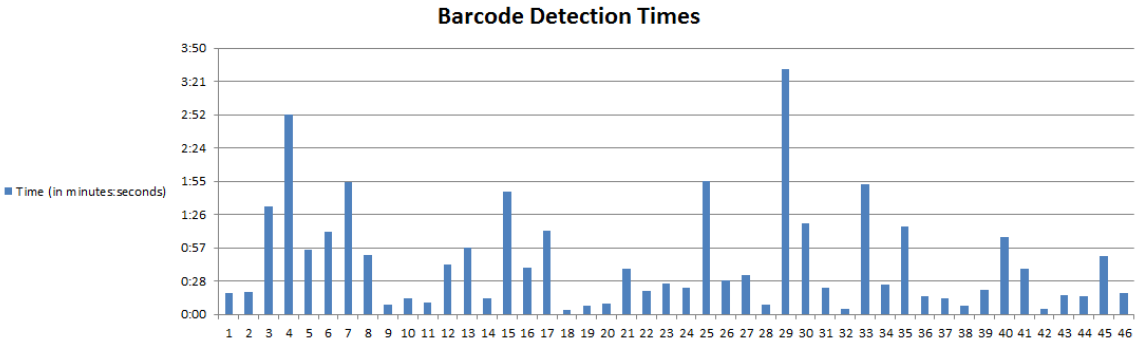


Figure 4.6. Barcode detection times.

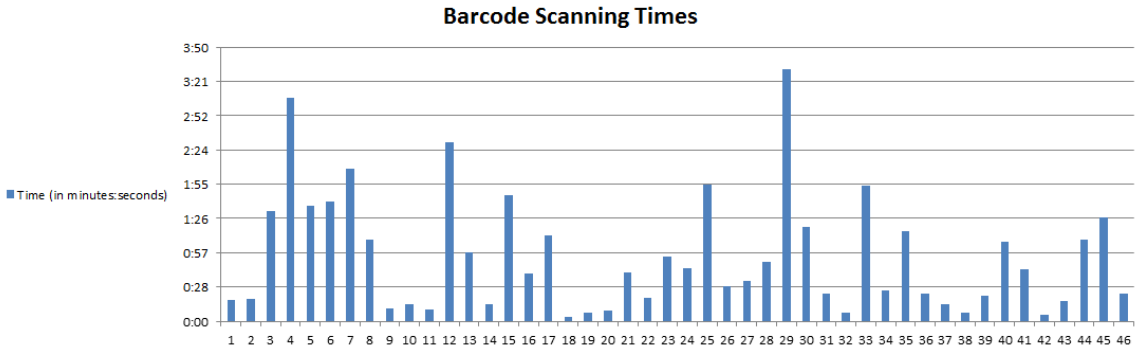


Figure 4.7. Barcode scanning times.

CHAPTER 5

TELESHOP

5.1 Introduction

TeleShop is the teleassistance module of the ShopMobile2 project. It allows VI individuals to transmit audio and video of their surrounding environment to sighted caregivers at remote locations. While shopping, the VI shopper may encounter situations where both the barcode scanner and the OCR engine fail or malfunction. TeleShop provides a backup in such cases by allowing the VI shopper to obtain help from their sighted caregivers. TeleShop is not limited to shopping alone. It can be used in navigation, in hotels and at any other places where the VI individual would benefit from sighted guidance.

TeleShop is the teleassistance module of the ShopMobile2 project. TeleShop allows VI users to obtain assistance from remote sighted caregivers by transmitting images and voice from their smartphones to the guides' computers or phones. TeleShop provides a backup in situations when the barcode scanner and OCR engine fail or malfunction. There is research evidence that having sighted guidance reduces the psychological stress on VI individuals [55]. TeleShop can provide the equivalence of sighted guidance without requiring the guide to be physically present.

Human navigation can be classified in to two categories - *micro-navigation* and *macro-navigation* [56]. Micro-navigation involves tasks in immediate vicinity of the traveler like obstacle avoidance where as macro-navigation involves tasks outside of the immediate perceptible environment. Some examples of macro-navigation tasks include planning a path between two points and looking for landmarks & waypoints. VI travelers perform both tasks continuously.

VI travelers typically use long canes or guide dogs to handle micro-navigation tasks.

However, sophisticated devices such as sonar canes and optical systems such as the Tom Pouce [57] or the TeleTact [57] system may also be used. Long canes can detect obstacles in front of the traveler from the ground up to waist height but are unable to detect overhanging obstacles or obstacles at head height. Sonar based systems cannot detect small obstacles while optical based systems do not perform well in areas glass surfaces.

GPS based systems [58,59] are broadly used to assist VI travelers with macro-navigation. However, since GPS solutions do not work well indoors, some researchers resort to other methods, such as RFID [60] for indoor navigation. Vision based systems can also be used for indoor navigation. The system described in [27] places fiducials next to barcodes, which can be decoded with a cell phone camera. Another vision based system is Google Goggles [61]. Using this system, the VI traveler can capture an image using her cell phone and Google Goggles can automatically decode text from it or match it with other images in its database. While this approach may be the right way to go in the long term, the system is currently not too reliable.

The term *teleassistance* covers a wide range of technologies to enable VI individuals to transmit video and voice to remote locations to obtain assistance which is typically given through voice. The systems developed by Bujacz et al. [62] and by Garaj et al. [63] are but two examples of such systems. The system developed by Bujacz et al. uses two notebook computers - one is carried by the VI traveler in a backpack and the other used by the sighted guide. The VI traveler transmits video through a USB camera mounted on the chest and connected to the computer. A earphone and microphone headset are used for communicating with the guide. The authors conducted indoor navigation trials and found that VI travelers walked faster, at a steadier pace, and were able to navigate easily when assisted by remote guides. The system developed by Garaj et. al. uses a GPS receiver in addition to the camera and notebook computer. Communication is established by using two GSM cell phones - one for voice and one for transmitting GPS data and a UHF link for transmitting video. The sighted guide can view the VI traveler's position on a map obtained from a GIS database in addition to the images from the camera. They conducted

an outdoor trial and tested both the micro-navigation and macro-navigation functionality of the system. They found that mobility levels for VI travelers increased when they were aided by sighted guides as compared to traveling unguided.

5.2 TeleShop

The TeleShop module of ShopMobile 2 consists of a server running on the VI shopper's smartphone and a client running on the caregiver's computer. As shown in Figure 5.2, images from the phone's camera are continuously transmitted by the server to the client and subsequently displayed on the GUI shown in Figure 5.1. The client allows the user to start, stop, and pause the incoming image stream and to change image parameters like resolution and quality. The pause option allows the caregiver to hold the current image on the screen when she wants to read something in the image. Changing the image parameters allows the caregiver to choose between the level of detail in the image and the smoothness of the incoming image stream. Images of high resolution and quality provide very good detail but may cause the resulting video stream to be choppy. On the other hand, images of lower resolution and quality result in a smoother video stream but do not provide much detail. The remote guide is given the option to choose the settings that suit her best.

All communication occurs over UDP. The VI shopper inputs the IP address and port number of the client to the server, which uses it to transmit images to the client. The client can retrieve the IP address and port number of the server from the incoming packets and uses it to transmit image parameters to the server. The client's information was input on the server because the client's IP address stays the same whereas the server's IP address can change if it is on a 3G network. TeleShop can operate with WiFi or 3G.

5.3 Laboratory Study

Two laboratory studies TeleShop were conducted. The first study was done with two sighted students, Alice and Bob. The second study was done with a married couple: a completely blind person (Carl) and his wife (Diana). All names have been changed to protect privacy. For both studies, we stocked four plastic shelves with empty boxes, cans,

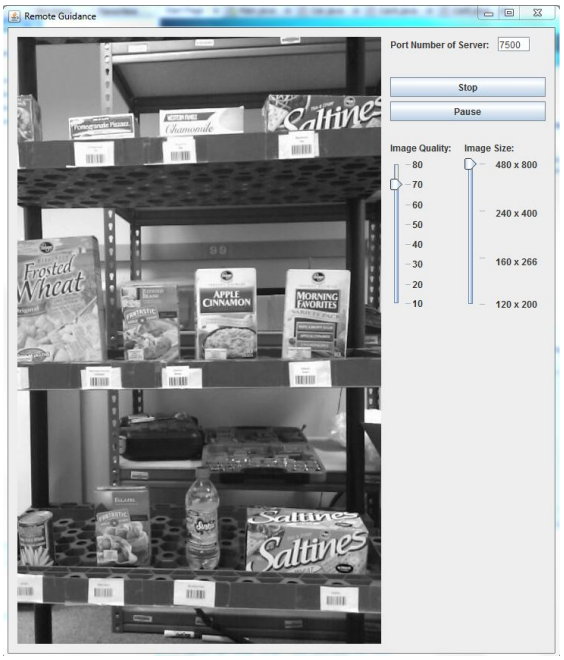


Figure 5.1. Screenshot of the client.

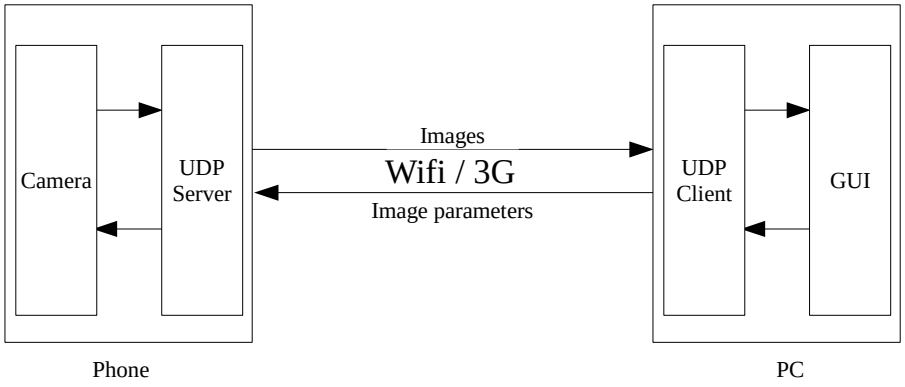


Figure 5.2. Overview of communication between the server and the client.

and bottles to simulate an aisle in a grocery store. In both studies, a Google Nexus One smartphone ran the TeleShop server and transmitted images and voice over WiFi to the remote guide's laptop with the client software in a different room.

In the first study, we blindfolded Bob so that he could assume the role of a VI shopper, and Alice assumed the role of the sighted guide. Alice was trained to use the client GUI,

and Bob was trained to use the cell phone. A voice link was established between the two by making a regular call. Once both of them were comfortable with the system, Alice was given a list of nine products (three sets containing three products each), which she had to help Bob shop for. Bob used the smartphone to transmit images of the shelf and Alice helped him pick the target products. When a target product was found, Alice would help Bob align the product with the camera so that she could read the nutrition facts from the product's package. She would then read out the nutritional facts on the product to Bob before moving on to the next product on the list. The second laboratory with Diana and Carl used the same training and settings.

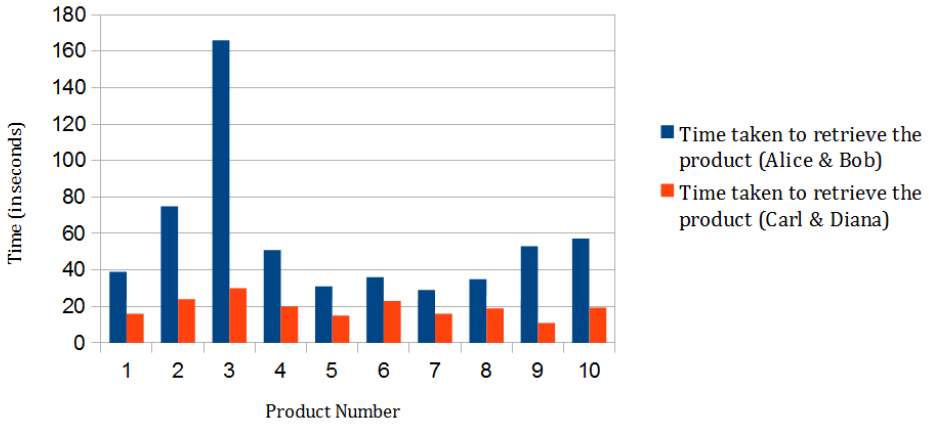


Figure 5.3. Times taken to retrieve products.

Both teams were able to retrieve and read nutrition facts from all the nine products successfully. Figure 5.3 shows the times taken to retrieve products and Figure 5.4 shows the times taken to read the nutrition facts of each product for both teams. It must be noted that product six did not have any nutrition facts on it and so the times taken to read its nutrition facts are zeros. Alice and Bob took an average of 57.22 and 86.5 seconds to retrieve a product from the shelf and to read its nutrition facts, respectively. The corresponding times for Carl and Diana were 19.33 and 74.8 seconds, respectively [64]. The times taken to read the nutrition facts are greater than the times taken to retrieve products. To read the

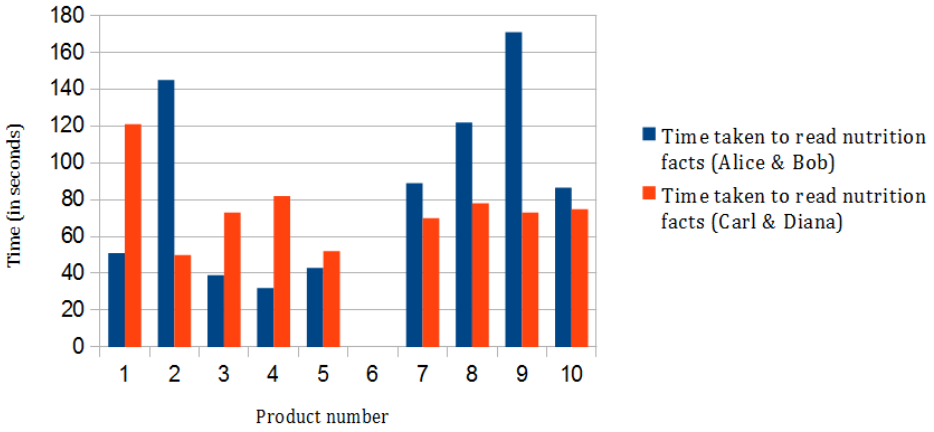


Figure 5.4. Times taken to read nutrition facts on products.

nutrition facts, the VI shopper had to align the product so that its nutrition table faced the camera, which took considerable time for both teams. It was observed that communication between the VI shopper and the sighted guide were key for quick retrieval and alignment of products. This may be the reason why Carl and Diana, being a married couple, were able to retrieve products and read nutrition facts faster than Alice and Bob. It was also observed that Alice did not change the resolution and quality settings at all whereas Diana changed it several times.

During the post-experiment informal interviews, Alice said that she was comfortable with the default resolution and size settings and did not need to change them. Both teams also said that they were comfortable with the system and did not have any problems with it. Diana suggested that allowing her to rotate the paused image would help with reading the nutrition facts. When asked about using this system in real-life, Carl said that he would find this system very helpful. He mentioned that when he travels, he uses Skype from his laptop to video call his wife to get information about the layout of his hotel rooms. The TeleShop module would allow him to get the same assistance more easily.

CHAPTER 6

OPTICAL CHARACTER RECOGNITION

6.1 Introduction

Optical Character Recognition (OCR) can be defined as the process of converting images of printed or handwritten text into machine encoded text. This process allows books and documents to be read digitally and finds many applications in the fields of information retrieval, text mining, etc. It also allows visually impaired (VI) people to read text that would otherwise be inaccessible to them.

The history of OCR can be traced as far back as the Nipkow disk [65], which was invented by P. Nipkow of Poland in the early part of the twentieth century. This machine used a system of holes drilled on a rotating disk to scan images, which could then be transmitted across distances. The first OCR systems were developed by Emmanuel Goldberg and Edmund Fournier D'Albe between the years of 1912 and 1914. Goldberg developed and patented a machine in 1912 that was able to read characters and convert them into standard telegraphic code. This machine laid the foundations of OCR by proving that printed characters could be converted to an encoded format. Fournier D'Albe is credited for developing a device known as the Optophone, which was specifically targeted towards the blind. This hand-held device produced a series of audible tones when moved across a printed page. The Optophone matched each character with a specific tone and VI people were able to read the page by moving the device across it and mentally mapping the audio output to the characters. This system was not practical since it required a lot of concentration and skill but it laid the basis for a working OCR system for the blind. Two engineers at the Radio Corporation of America (RCA) improved this system by developing a much smaller 'electric pencil' similar to the Optophone.

OCR technology is now very mature and many reliable systems have been developed for desktop machines. However, its implementation on cell phones is relatively new. Modern cell phones possess all the technologies required for creating a hand-held OCR system but lack in processing power and the quality of the scanned image. The built-in camera is great for capturing images on the go but the quality of the image is not comparable with the quality of dedicated scanning equipment. Many researchers have focused on porting existing OCR technologies and even developing new technologies for cell phone use. Examples of traditional OCR readers include [66,67] and examples of technologies that are better suited for cell phones include business card readers [68,69] and language translators [70,71]. Many systems have also been developed specifically for VI people. Examples of such systems include signage readers [72], systems that can recognize banknotes [73], etc.

The ability to carry a truly hand-held OCR system opens up many exciting possibilities for VI people. We aim to venture in one such area - using OCR technology to help VI individuals read nutrition facts on grocery products. The format of the nutrition table is fairly standard in the United States. It consists of categories like 'Calories', 'Calories from Fat', 'Carbohydrates', etc. Each of these categories have a numerical value associated with them. Our aim is to develop a proof-of-concept prototype relating to many key areas in OCR technology that can help VI individuals read nutrition facts on grocery products. As per our research, this specific application has not be explored previously and will allow VI individuals to compare between different products and choose the one that bests suits their needs. This system can also be used to warn VI shoppers about specific products that pose a health risk (due to peanut allergies, for example) by reading the ingredient list.

The OCR process can be broken down into three main subprocesses - *Preprocessing*, *Segmentation* and *Classification*. Preprocessing is concerned with improving the overall quality of the image. The next subprocess is called Segmentation and is concerned with segmenting (or removing) relevant portions of the image from the whole image. The nutrition facts table is the relevant portion of the image in our case but it cannot be processed directly. Thus, the segmentation process is further broken down to nutrition table seg-

mentation, line segmentation, word segmentation and finally, character segmentation. The output of segmentation is fed to the Feature Extraction and Classification subprocess, which is responsible for converting the individual word and letter images to machine encoded text. Finally, a fourth (optional) subprocess called 'Error Detection and Correction' may also be employed, which as its name implies, detects and corrects spelling and other mistakes from the classification stage.

Figure 6.1 shows an overview of the entire process. It also indicates if each subprocess was implemented in MATLAB, Android or both. In general, the entire development was done in MATLAB to save time and some subprocesses were then ported on the Android phone for verification. The nutrition facts table segmentation is the most important (and the most difficult) subprocess of all the segmentation subprocesses and so it was ported on Android for verification. Similarly, the neural network used for character recognition is the most important and difficult subprocess of the two Classification subprocesses and so it was also implemented on Android. MATLAB provides a plethora of 'out-of-the-box' libraries for image processing but we have been very careful to avoid using something that cannot be replicated on an Android device.

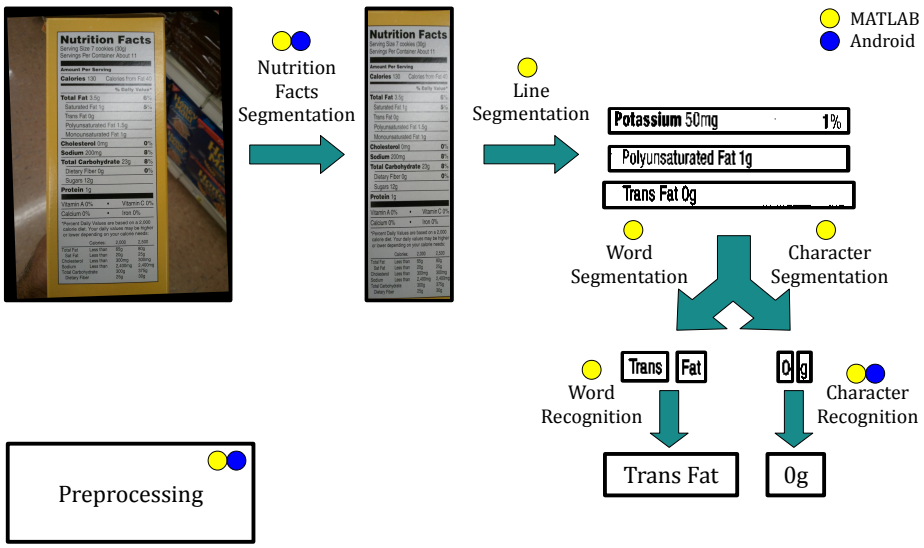


Figure 6.1. Overview of the OCR process.

This remainder of this chapter will be organized as follows. Section 6.2 discusses some

pre-processing techniques used in the OCR process. Section 6.3 discusses the process of segmenting the nutrition table from the image and then segmenting individual lines and words from it. Finally, Section 6.4 addresses the feature extraction and classification part of the OCR process.

6.2 Preprocessing

"Garbage In, Garbage Out" is a very popular phrase in computing. If the input to an computer system is absolute "garbage", one cannot expect anything more than garbage in return. However, in the field of OCR, it is quite possible that data which can be easily read by humans appears as garbage to the computer. This is due to the fact that almost all input images are degraded by noise, shadows, highlights or skew errors. Humans are very adept at detecting such noise and removing it from consideration but unfortunately, the same cannot be said for OCR systems. Thus, the first step in the OCR process is to pre-process the image and clean it. The preprocessing step is responsible for binarizing the image and removing noise from it. This stage is also responsible for ensuring that the characters in the image are vertically aligned and are minimally distorted. Let us now examine some of the common pre-processing steps. We will also indicate if these steps are being used in our application and the specific techniques used to implement them.

6.2.1 Binarizing Images

The image from the camera is a grayscale image Y, where each pixel in the image has a value from 0 to 255. This image has to be converted to a two-level binary image B where each pixel has a value of either 0 or 255. Binarization is defined as the process that converts a grayscale image to a two-level binary image. Binarization is performed using a very simple concept known as thresholding, where every pixel p from the input image Y is compared against a threshold τ and the corresponding output pixel is set to 0 if $p < \tau$ or 255 otherwise. Binarization techniques can be broadly classified as global threshold methods and local threshold methods depending on how the obtain the value of τ. Global thresholding methods assume a single threshold for the entire image whereas local

thresholding methods compute thresholds for each pixel in the image or a block of pixels in the image. We will now discuss both these methods briefly.

6.2.1.1 Global Threshold

Using a global threshold is the easiest way to binarize an image. In this method, we use a global threshold τ and for every pixel Y_p in the input image Y, we obtain the corresponding pixel B_p in the two-level binary image B as follows:

$$B_p = \begin{cases} 0 & \text{if } Y_p < \tau \\ 255 & \text{otherwise} \end{cases} \tag{6.1}$$

The threshold τ can be fixed for all images (e.g. $\tau = 127$) or it may be computed for each individual image (e.g. $\tau = mean(Y)$).

Otsu's binarization method [33] is a global thresholding method that is very popular for binarizing images with well-defined foreground and background regions. This method searches for a threshold that maximizes inter-class variance and minimizes intra-class variance between the two classes (foreground and background).

The advantage of the global thresholding methods is that they are very simple and inexpensive. However, for images where the difference in foreground and background intensities is less or where foreground and background intensities overlap each other, these methods do not achieve good results.

6.2.1.2 Local Threshold

Instead of using a single threshold for the entire image, local thresholding methods compute local thresholds $\tau_{x,y}$ for each pixel $p = Y_{x,y}$ or a group of pixels in the image. Niblack's binarization method [34] is an example of a local thresholding method. However, as shown in Section 3.4.1, this method produces noisy images and so we use a modified version of Niblack's original method for binarizing the input image. In this method, the image is divided into $n \times n$ pixel subimages and a threshold $\tau_{i,j}$ is computed for every subimage $Y_{i,j}$ in Y. This threshold is computed using the following equation:

$$\tau_{i,j} = \begin{cases} m_{i,j} + k \times s_{i,j} & \text{if } s_{i,j} \geq S \\ \tau^c & \text{otherwise} \end{cases} \tag{6.2}$$

where, $m_{i,j}$ and $s_{i,j}$ are the mean and standard deviation values for the subimage $Y_{i,j}$. S is the standard deviation threshold that decides if the threshold value $\tau_{i,j}$ is computed using the mean and standard deviation values or is assigned the fixed threshold τ^c and k is a user-defined parameter that controls the brightness of the resulting image. In our application, we set $S = 12.7$, $k = 0.5$ and $\tau^c = 127$, respectively.

6.2.2 Noise Filtering

The authors in [74] define noise as undesired random degradations in images, which may occur during capture, transmission, and processing. These degradations manifest themselves as either additive noise or subtractive noise. Figure 6.2 shows an example of an image (top) and its noisy counterpart (bottom). It can be observed that additive noise takes the form of foreground pixels and subtractive noise takes the form of background pixels. Noise is undesirable because it can cause segmentation as well as classification errors. Additive noise can join adjoining characters together whereas subtractive noise can cause a single character to appear as two (or more) distinct characters.

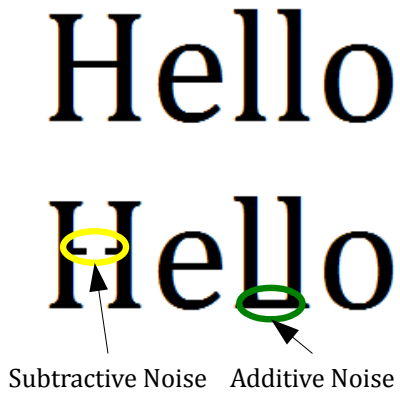


Figure 6.2. Additive and subtractive noise.

Jain [75] classifies image noise as Gaussian noise, Rayleigh noise, Gamma noise, Exponential noise, Salt and Pepper noise, Uniform noise, and Sinusoidal noise. Gaussian noise and Salt and Pepper noise are the two most commonly encountered forms of noise encountered in images. In the case of Gaussian noise, the noisy pixel has a grayscale value that is a function of the Gaussian distribution whereas in the case of Salt and Pepper noise, the noisy pixel takes on one of two values ('salt', which is lighter or 'pepper', which is darker). Median filters [76–78] and kFill algorithms [79, 80] are two very common techniques used to remove salt and pepper noise from images. A study [81] comparing different types of filtering algorithms for removing Gaussian noise in images found that the Wiener filter [82] performed the best.

We did not find the need to use noise filtering algorithms as they would increase computational load. Also our use of a blurring filter in the word segmentation process (see Section 6.3.4.1) and template matching for the classification process (see Section 6.4.3) eliminate the need for noise filtering in both the segmentation and classification stages.

6.2.3 Thinning

Thinning is defined as the process by which characters are reduced to *skeletons* - a set of thin lines (one pixel thick), attached one to another in a few connection points that preserve the topological and geometric properties of its originating object [83]. The challenge in this process is to retain the original shape of the character. Apart from OCR, thinning algorithms are employed in various other applications from medicine [84] to fingerprint analysis [85]. In case of OCR, thinning algorithms enable us to analyze characters from a topographical perspective where each character can be represented in terms of features like end points, junction points and connections among components [86].

Thinning algorithms work by examining foreground pixels in the image and then deleting them until only a skeleton remains. This process is done in multiple passes and each pass peels away a boundary layer of the image. We can classify thinning algorithms as either sequential or parallel depending on the way pixels are deleted. In sequential thinning algorithms [87, 88], a boundary pixel is marked for deletion depending on the result of

the preceding pixel whereas in parallel thinning algorithms [89, 90], each pixel is examined independently of other pixels.

We use template matching techniques for the classification stage, which uses pre-defined templates of each character/word to classify images. This process does not require images to be thinned and so we do not use thinning in the pre-processing stage.

6.2.4 Normalization

An image can suffer from four basic forms of distortion: translation, rotation, scaling and skew [91]. The image normalization process is responsible for transforming an image to its normal form that is invariant to these distortions. The need for normalization is most evident for systems that use template matching for the classification stage. Normalization techniques are able to transform the input image so that both the input image and the template can be compared in a meaningful way. Techniques such as moment invariants [92] can easily rectify translation, rotation and scaling distortions. Skew detection and correction is a more complex problem but many algorithms have been developed for it [93, 94].

The camera alignment module 3.3 ensures that the camera is always aligned with the product. This in turn, ensures that the input image will not suffer from rotation and skew distortions. The input image can still suffer from translation and scaling distortions and so we normalize the segmented character/word image to remove these distortions. This is a two step process. First, we remove whitespace from around the character/word image to ensure that the character/word fully occupies the image. This process removes all translation distortions from the image. We then scale the image to match the size of the template and this process gets rid of the scaling distortion. Figure 6.3 shows an example of removing these distortions.

6.3 Segmentation

Segmentation is defined as a process by which an image is decomposed into one or more subimages each of them containing regions of interest. The regions of interest in a typical OCR situation are subimages that contain individual characters, which are then classified

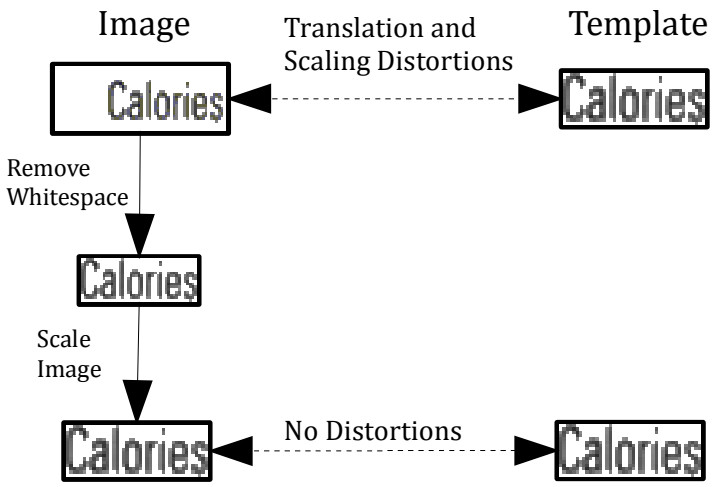


Figure 6.3. Removing translation and scaling distortions from an image.

into machine encoded characters. In our application, the image may contain text within the nutrition table as well as outside of it. Since we are primarily interested in the contents of the nutrition table, the first step is to segment the nutrition table from the image and then segment characters and words within the nutrition table. Let us first discuss a couple of key concepts that will help us in segmentation.

6.3.1 Horizontal and Vertical Projections

The horizontal projection HP of an image is defined as the running count of foreground pixels for each row of the image [95]. Similarly, the vertical projection VP of the image is defined as the running count of foreground pixels for each column of the image [95]. Foreground pixels, as opposed to background pixels, are defined as pixels that contain information of interest. These pixels are usually black-colored pixels but there are certain cases in which they may be white-colored. We will assume that foreground pixels are black unless specifically mentioned otherwise. Figure 6.4 shows the horizontal and vertical projections for an image. Equation 6.6 shows the equation by which these projections can be obtained for a $m \times n$ pixel image I (assuming black-colored foreground pixels):

$$f = HP(I) \tag{6.3}$$

$$f_y = \Sigma_{x=0}^{n}(255 - I(x, y)) \tag{6.4}$$

$$g = VP(I) \tag{6.5}$$

$$g_x = \Sigma_{y=0}^{m}(255 - I(x, y)) \tag{6.6}$$

where, $I(x, y) = 0$, for a black pixel and $I(x, y) = 255$ for a white pixel. The above equation can be modified for white-colored foreground pixels as follows:

$$f = HP(I) \tag{6.7}$$

$$f_y = \Sigma_{x=0}^{n} I(x, y) \tag{6.8}$$

$$g = VP(I) \tag{6.9}$$

$$g_x = \Sigma_{y=0}^{m} I(x, y) \tag{6.10}$$

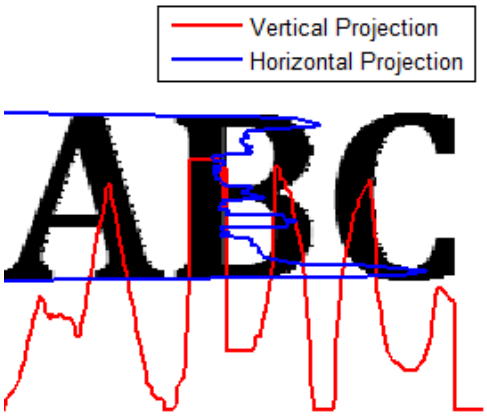


Figure 6.4. Horizontal and vertical projections.

6.3.2 Nutrition Table Segmentation

Figure 6.5 shows an image containing a complete nutrition table that has its sides

aligned with the sides of the image. In reality, it is possible that the image does not contain a nutrition table, contains a nutrition table that is cropped along one or more sides, or contains a nutrition table that is rotated. However, in this chapter, we will assume that the nutrition table exists in the image, is complete (not cropped along any side) and that its sides are aligned with the sides of the image. We will also assume that the nutrition table is is not warped (like on a can). The layout of the nutrition table is somewhat standardized and it contains items like, "Serving Size," "Amount Per Serving," "Calories," etc. Let these items be referred to as *nutritional categories*. We can observe that these items are separated by horizontal lines and the nutrition table is usually bounded by a rectangular box.

Figure 6.5. Nutrition tables.

The nutrition table segmentation process is comprised of three stages. The first stage will determine the approximate location of the nutrition table along the horizontal axis. The second stage will determine the exact location of the nutrition table along the horizontal axis and finally, the third stage will determine the exact location of the nutrition table along the vertical axis.

The first stage uses the horizontal lines within the nutrition table to determine the approximate location (x'_{start}, x'_{end}) of the nutrition table along the x (or horizontal) axis. Consider a horizontal line detection kernel $HLDK$ having dimensions $3 \times m$ as shown in Equation 6.11. Here, m is a user-defined parameter, which is used to determine the length of horizontal lines in the image and its value is set to 50 for our application. This kernel is used to detect large horizontal lines in the image. Figure 6.6 shows the result when a two-level binary version of the image shown in Figure 6.5 is convoluted with the line detection kernel. Let this resulting horizontal line image be denoted by HLI. We can observe that HLI has a high response for areas containing large horizontal lines and a low response for other areas. We can also observe that the area containing the nutrition table has a high response in HLI since it contains a lot of large horizontal lines that separate the various nutritional categories.

$$HLDK(i,j) = \begin{cases} 1 & \text{if } i = 1 \\ -m & \text{otherwise} \end{cases} \tag{6.11}$$

The projection in Figure 6.7 assumes that the foreground pixels are white-colored. Consider a threshold τ_H, which will be used to find the approximate location of the nutrition table. This threshold is set to the mean value of $VP(HLI)$ and shown using the red colored line in Figure 6.7. We can observe in Figure 6.7 that the area containing the nutrition table exhibits values of $VP(HLI)$ that are higher than the threshold and other areas exhibit values of $VP(HLI)$ that are lower than the threshold. We can use this observation to determine the approximate location of the nutrition table as follows:

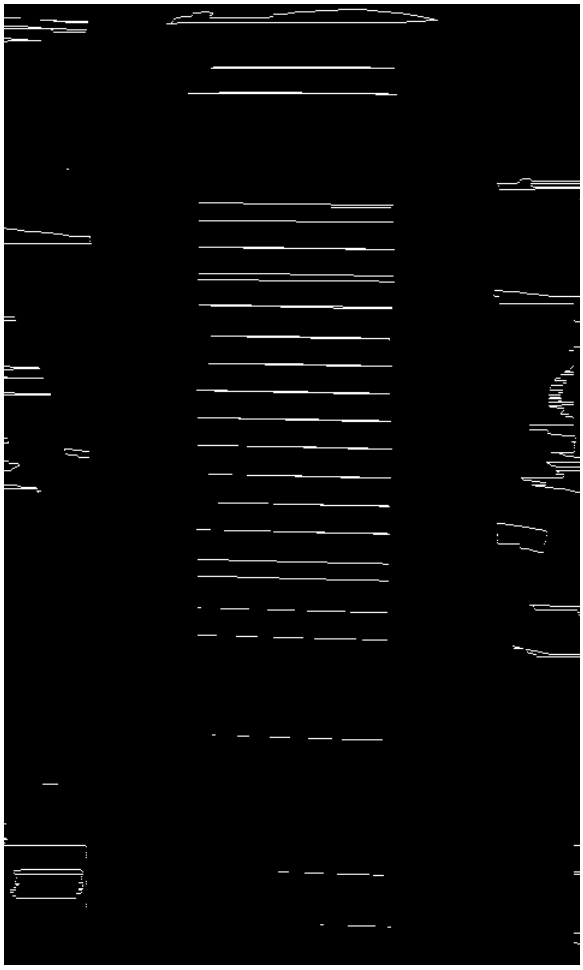


Figure 6.6. Horizontal line filtered image.

$$f = VP(HLI) \tag{6.12}$$

$$x'_{start} = \text{maximize } i | f(i) >= \tau_H \tag{6.13}$$

$$x'_{end} = \text{minimize } i | f(i) >= \tau_H \text{ and} \tag{6.14}$$

$$x'_{start} < x'_{end} \tag{6.15}$$

The horizontal lines separating the nutritional categories do not extend fully to the sides of the nutrition table and there is a small gap between these lines and the vertical

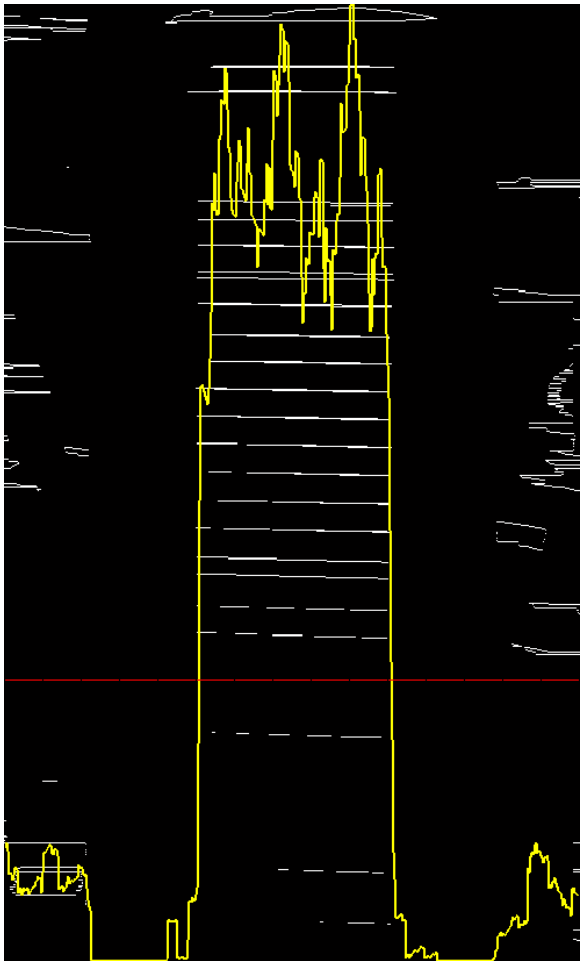


Figure 6.7. Vertical projection of horizontal line filtered image.

lines that enclose the nutrition table. If we segment the nutrition table based solely on the horizontal lines, the nutrition table will be cropped along the sides. Thus, it is essential to add the second stage in the nutrition table segmentation process, which closely examines the area around the approximate location (x'_{start}, x'_{end}) to obtain the exact location (x_{start}, x_{end}) of the nutrition table.

Consider another line detection kernel ($VLDK$) that detects large vertical lines in the image. This kernel is of size $m \times 3$, where m is a user-defined parameter, which is set to 50 in our application. This kernel is shown in Equation 6.16. Figure 6.8 shows the result when a two-level binary version of the image shown in Figure 6.5 is convoluted with the line

detection kernel. Let this resulting image be denoted by VLI. We can observe that VLI has a high response for areas containing large vertical lines and a low response for other areas.

$$VLDK(i,j) = \begin{cases} 1 & \text{if } j = 1 \\ -m & \text{otherwise} \end{cases} \tag{6.16}$$

Figure 6.8. Vertical line filtered image.

Consider the vertical projection of the vertical line filtered image, which is shown in Figure 6.9. Let it be denoted by $VP(VLI)$. As shown in Figure 6.5, the nutrition table is bounded by a box. The vertical lines of this box show up as large spikes in $VP(VLI)$. Consider a threshold τ_V, which is the mean of all positive values of $VP(VLI)$. The exact

location of the nutrition table is obtained as follows:

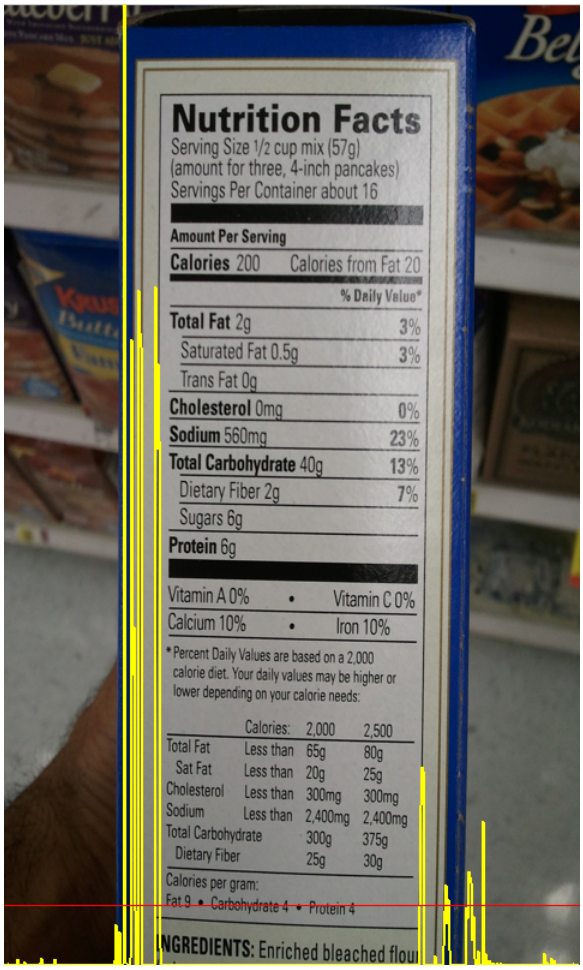


Figure 6.9. Vertical projection of vertical line filtered image.

$$f = VP(VLI) \tag{6.17}$$

$$x_{start} = \text{minimize } (x'_{start} - i) | f(i) >= \tau_V \text{ and } i \leq x'_{start} \tag{6.18}$$

$$x_{end} = \text{minimize } (i - x'_{end}) | f(i) >= \tau_V \text{ and } i \geq x'_{end} \tag{6.19}$$

The third stage of the nutrition table segmentation process determines the exact location of the nutrition table along the y (or vertical) axis. Consider the horizontal projection of the horizontal line filtered image $HP(HLI)$ as shown in Figure 6.10. This projection

is obtained differently when compared to the previous two projections. In this case, we consider only the areas between x_{start} and x_{end} in HLI while computing $HP(HLI)$. Since we know the exact location of the nutrition table along the horizontal axis there is no reason to consider the other areas while computing $HP(HLI)$. We compute the value of the threshold τ_V as follows:

$$f = HP(HLI) \tag{6.20}$$

$$\tau_V = \text{ mean } f(i)|f(i) > 0 \tag{6.21}$$

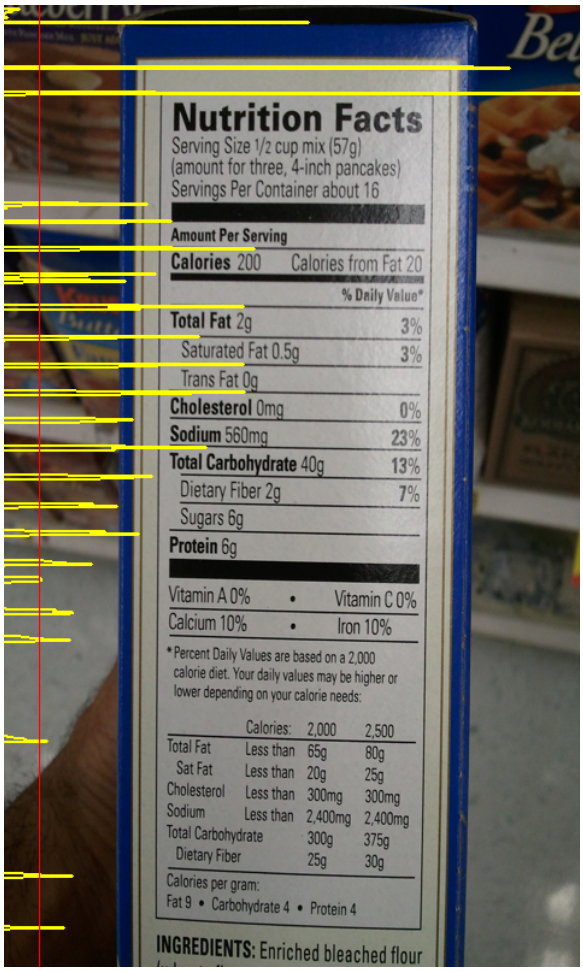


Figure 6.10. Horizontal projection of horizontal line filtered image.

The threshold τ_V is shown by a red colored vertical line in Figure 6.10. The exact location of the nutrition table along the vertical axis (y_{start}, y_{end}) can be determined as follows:

$$f = HP(HLI) \tag{6.22}$$

$$y_{start} = \text{minimize } i | f(i) >= \tau_V \tag{6.23}$$

$$y_{end} = \text{maximize } i | f(i) >= \tau_V \tag{6.24}$$

The exact location of the nutrition table is now determined and it can be cropped from the original image for further processing.

6.3.2.1 Nutrition Table Segmentation Experiments

This section discusses an experiment performed to evaluate the performance of the nutrition facts segmentation process. We captured images of nutrition facts on 45 products using the Nexus One cell phone. All images were captured in the University store to simulate actual conditions. The pictures were captured after viewing them on the screen, which means the nutrition table exists in the image, is not rotated and is not cropped along any of the sides. These images were then processed using the nutrition facts segmentation process to obtain the segmented images. We also manually marked the four corners of the nutrition table and stored their coordinates as ground truth for comparison.

We then compared the location of the segmented nutrition table with the ground truth to evaluate the performance of the segmentation process. Figures 6.11 through 6.14 show the resulting error histograms for the starting and ending positions of the nutrition table along the X and Y axes, respectively. A positive error denotes that the segmented image contains some part of the background image and a negative error denotes that the resulting segmented image has lost some areas of interest. In general, it is better to have a positive error than a negative one because a negative error implies that some information is lost that may result in classification errors.

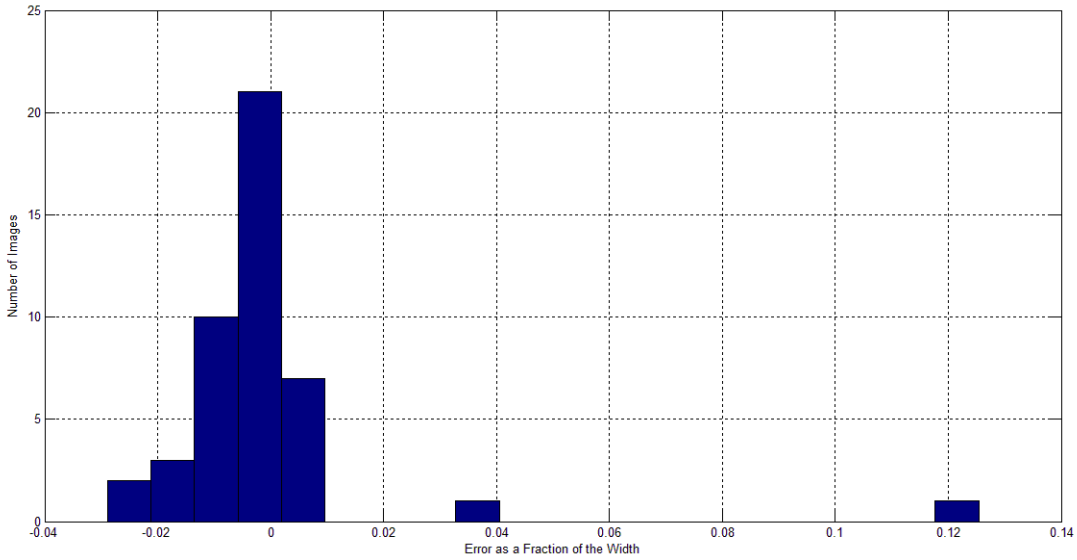


Figure 6.11. Histogram of errors in the starting position of the nutrition table along the X axis.

It can be observed that the performance of the system is very good along the X axis with mean errors of approximately 1%. The notable exception is the case where the error for the starting and ending positions was 12.5% and 14%, respectively. Figure 6.15 shows the image that caused this error. It can be observed that this image lacks a black colored bounding box that is usually present in the nutrition table. Thus, the system would have obtained a good coarse location of the nutrition table but was not able to obtain the exact location along the X axis due to the absence of the vertical lines of the bounding box.

We can also observe that the performance of the system is not as good along the vertical axis with average errors of 5% and 7% for the starting and ending positions, respectively. Since most errors are positive errors, this is not a problem as the line segmentation process will get rid of areas extraneous to the nutrition table. Most of these errors occur because the segmentation process picks up the top edge of the box (see Figure 6.15) as the start of the nutrition table. However, each segmented nutrition table contained the area between the two thick black colored lines, which is the main body of the nutrition table.

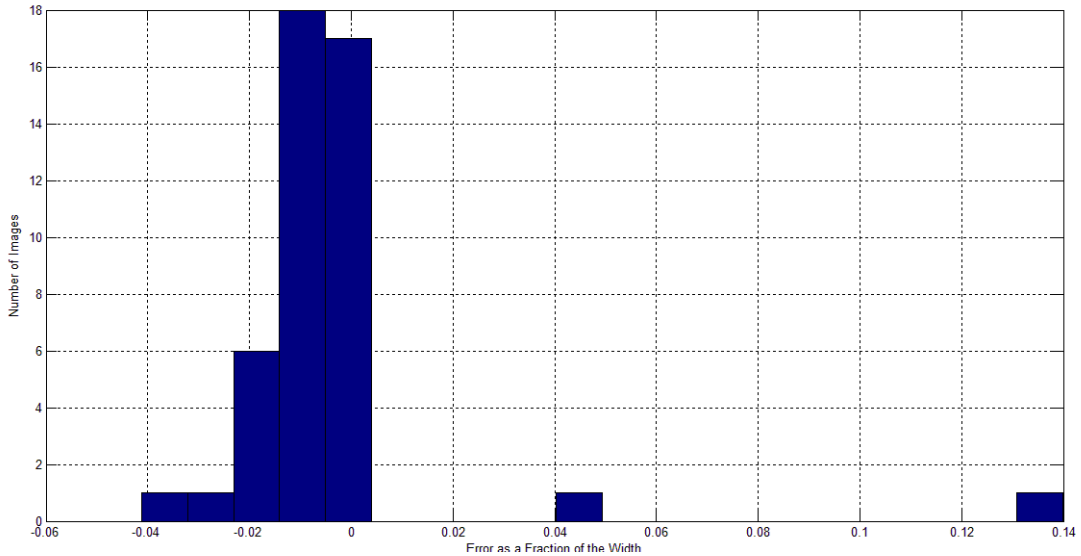


Figure 6.12. Histogram of errors in the ending position of the nutrition table along the X axis.

6.3.3 Line Segmentation

Let us refer to the part of the image containing each nutritional category as a 'line' within the nutrition table. Each of these lines must be segmented from the nutrition table before the words in them can be recognized. We can observe in Figure 6.5 that each of these lines are separated by black colored horizontal lines. In order to reduce confusion, these lines shall always be referred to as 'black colored lines' and the part of the image containing the nutritional categories shall be referred to only as 'lines'.

The first step in the line segmentation process is to detect the black colored lines within the table. Let N denote the binarized segmented nutrition table image and let N_i denote the i^{th} row within this image. To detect the black colored lines, we shall define a function BL such that $BL(i)$ denotes the probability of N_i containing a black colored line. Let l_j denote the length of the j^{th} black colored line segment within N_i that has a length greater than some threshold τ_L and let m_i denote the total number of such black colored line segments. The following equation shows how this function is defined:

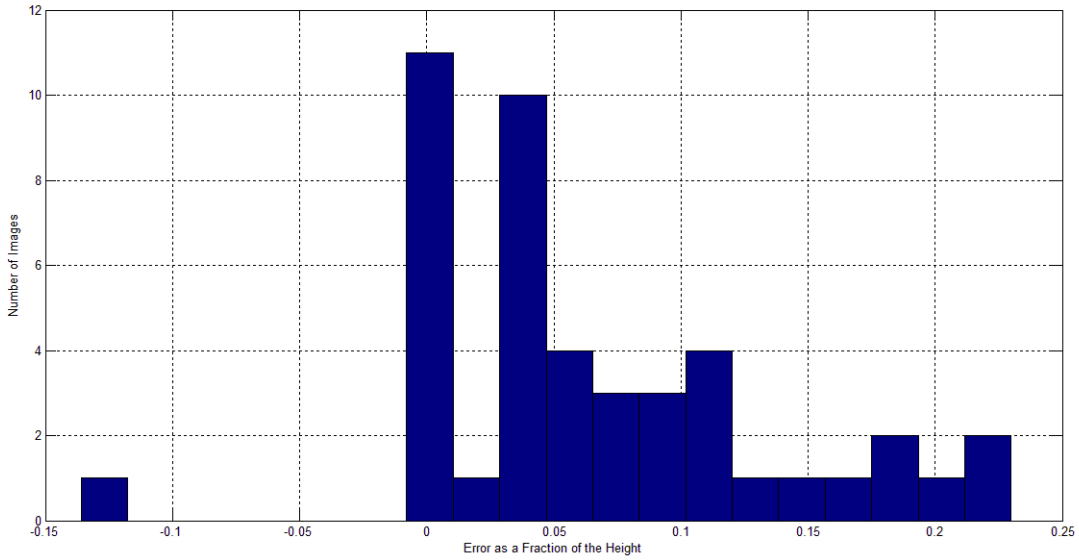


Figure 6.13. Histogram of errors in the starting position of the nutrition table along the Y axis.

$$BL(i) = \text{geometric mean } (l_0, l_1, \ldots, l_m) \tag{6.25}$$

We chose to use the geometric mean instead of the arithmetic mean because the geometric mean is more indicative of the central tendency of a set of numbers as compared to the arithmetic mean. Consider Figure 6.16, which shows two lines - A and B each containing black colored line segments. Both these lines contain 50% black colored pixels and hence the arithmetic mean of both these lines is equal to 0.5. The geometric mean of line A can be computed as $GM(A) = \sqrt[8]{(0.0625)^8} = 0.0625$ and similarly the geometric mean of line B can be computed as $GM(B) = \sqrt[2]{(0.25)^2} = 0.25$. It can be observed that the geometric mean of line B is much higher than that of line A and is indicative of the length of line segments present in it. Since we are interested in detecting large black colored lines in the nutrition table, we use geometric mean to compute the function BL.

The next step in the line segmentation process is to normalize the values of BL by dividing the value of BL for each row by the maximum value of BL for the entire image. This step ensures that every value of BL is between zero and one. We now compute a

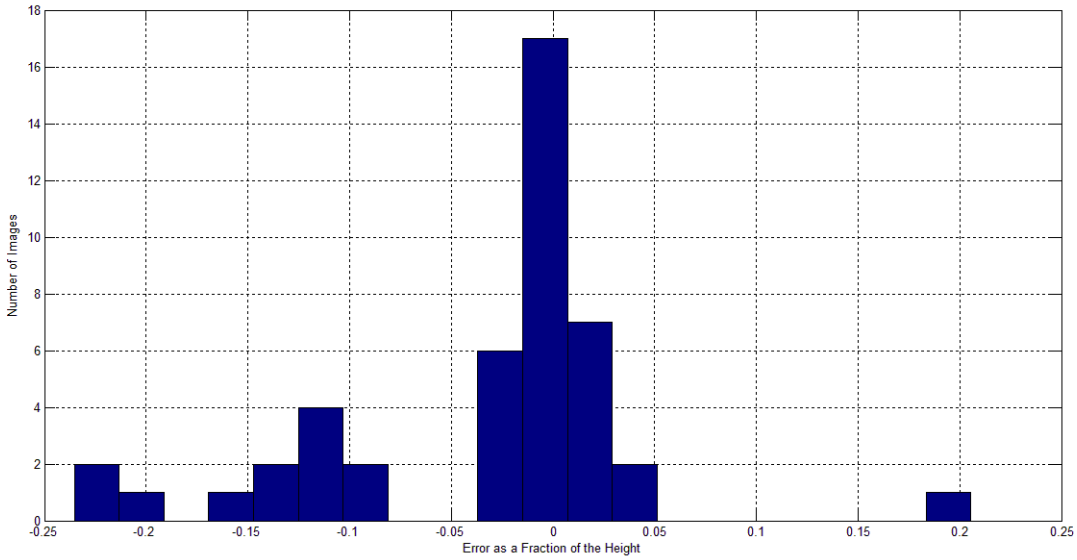


Figure 6.14. Histogram of errors in the ending position of the nutrition table along the Y axis.

threshold τ as the mean of all positive values of BL. Using this threshold we can identify the starting (s) and ending (e) coordinates of every line along the y axis as follows:

$$s = i | BL(i-1) >= \tau \text{ and } BL(i) < \tau \tag{6.26}$$

$$e = j | BL(j+1) >= \tau \text{ and } BL(j) < \tau \tag{6.27}$$

We can use the threshold T to identify black colored lines. The i^{th} row N_i is assumed to be a black colored line if $N_i >= \tau$. Since each line is enclosed by a black colored line above and below it, the starting coordinate s is defined as the coordinate i that is just below a black colored line and the ending coordinate e is defined as the coordinate j that is just above a black colored line. Once these coordinates are identified, the line can be segmented from either the binarized or the grayscale nutrition table image.

6.3.4 Word and Character Segmentation

The segmentation process has a very significant impact on the overall result of OCR

Figure 6.15. Cause of error in segmentation.

and many errors can be directly attributed to it [96]. The authors in [97] define character segmentation as an operation that seeks to decompose an image of a sequence of characters into subimages of individual symbols. This process can also be extended to include word segmentation where entire words are segmented instead of characters. Character segmentation is much more popular than word segmentation because it allows virtually unlimited words to be recognized.

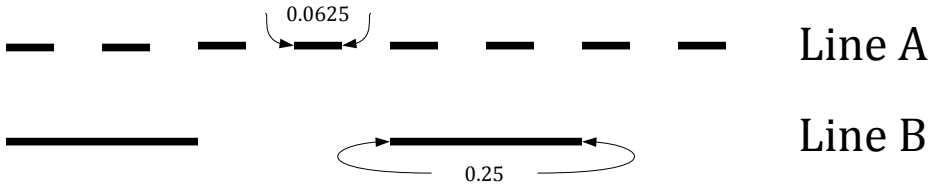


Figure 6.16. Differences between arithmetic and geometric mean.

Word segmentation, on the other hand, is relatively simpler but places a restriction on the number of words that can be recognized since the system needs a way to classify them. This approach is better suited in cases with a limited lexicon. We use both word and character segmentation in our application. Word segmentation is used to segment words like 'Calories', 'Fat', etc and character segmentation is used to segment the numerical values of those categories.

6.3.4.1 Word Segmentation

The word segmentation process is responsible for identifying and segmenting words from the line segmented in the previous step. The line segment may contain some noise due to the remnants of black colored lines from the nutrition table. The presence of these lines may cause segmentation errors and so it is important to eliminate them. One approach is to identify regions that belong to the characters and then segment words based on information from those regions. Let us first identify a region called *middle zone*, which belongs to the torso of characters.

We can define four imaginary lines - *topline*, *midline*, *baseline* and *beardline* for every letter in the Latin alphabet [98]. The baseline is defined as the line on which the letters rest. The midline is defined as the line connecting the top of letters like 'a', 'c', 'e', etc. and the top of the torso of letters like 'b', 'd', etc. The topline is defined as the line connecting the ascenders of the letters like 'b', 'd', etc. and finally, the beardline is defined as the line connecting the descenders of letters like 'g', 'j', etc. Figure 6.17 shows the topline, midline,

baseline and beardline for letters in a word.

The topline and the beardline touch only a few letters in the alphabet while the midline and the baseline touch most of the letters in the alphabet. We can now extract the middle zone, which is the area between the midline and the baseline to detect words in a line. Consider Figure 6.18, which shows a horizontal projection ($HP(word)$) of the black colored pixels in a word. Let $\frac{\delta}{\delta y}HP(word)$ denote the gradient along the y axis. Figure 6.19 shows this gradient. It can be observed that the gradient has minimum value at the baseline and maximum value at the midline. The baseline connects the bottom of the letters and so it connects a lot of black colored pixels that have white colored pixels below them. This causes the gradient $\frac{\delta}{\delta y}HP(word)$ to be minimum at that point. A similar justification can be used for the midline as it connects the top of letters (black colored pixels that have white colored pixels above them). We can derive the locations of the midline and the baseline as follows:

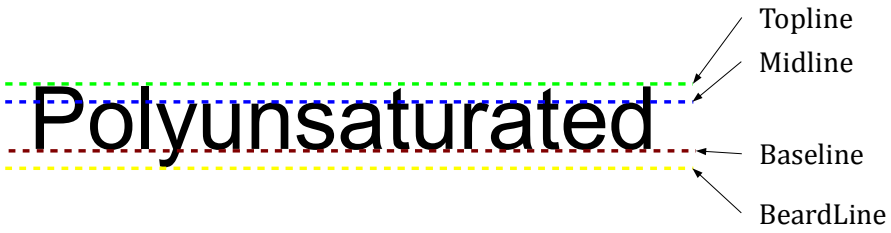


Figure 6.17. Topline, midline, and baseline for letters in a word.

Figure 6.18. Horizontal projection of letters in a word.

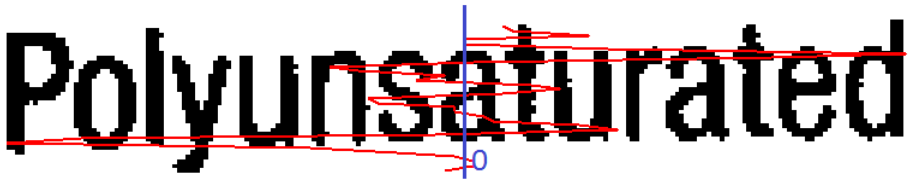


Figure 6.19. Vertical gradient of horizontal projection of letters in a word.

$$f = \frac{\delta}{\delta y} HP(word) \tag{6.28}$$

$$baseline = i | f(i) \text{ is minimum} \tag{6.29}$$

$$midline = i | f(i-1) \text{ is maximum} \tag{6.30}$$

Figure 6.20. Part of word between baseline and midline.

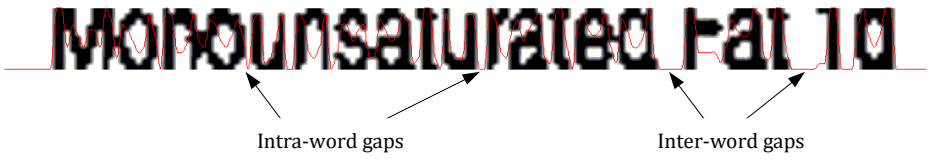


Figure 6.21. Vertical projection of a line image.

Let L_{middle} define the portion of the line image that lies between the baseline and the midline. As shown in Figure 6.20, this portion of the image will contain black pixels that belong to words within the line and exclude pixels belonging to noise. Let $VP(L_{middle})$ denote the vertical projection of this portion of the image, which is shown in Figure 6.21.

It can be observed that the value of the vertical projection is zero for spaces between letters (*intra-word gap*) and words (*inter-word gap*). To segment words from the image, we have to segment the line using the inter-word gaps and thus it is important to distinguish between the two types of gaps. The authors in [99] discuss and evaluate eight different

algorithms and metrics to separate words based on these gaps. We chose to go with a simpler solution of applying a smoothening (or blurring) filter to the entire line image. A similar approach is used in [100] for line segmentation. This filter is defined as follows:

$$SF = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{6.31}$$

Let SL define the result of convolution of SF and the inverted image (black pixels are replaced by white and vice versa) of L_{middle}. The reason we invert the image is because SF has a higher response for white colored pixels as compared to black colored pixels. Figure 6.22 shows an example of SL as well as its vertical line projection $VP(SL)$. It can be observed that SL contains no intra-word gaps and thus our problem is greatly simplified. The words can now be segmented from the line image using the intra-word gaps. To do this, we examine the values of $VP(SL)$ and segment areas where the value of $VP(SL)$ is above a certain threshold (equal to 3 in our case). This threshold is usually very small and is selected so as to minimize the number of false positives.

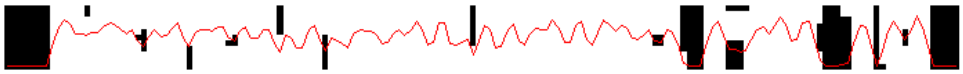


Figure 6.22. Smoothened line image and its vertical projection.

6.3.4.2 Experiments to Evaluate the Midline and Baseline Detection Algorithm

We performed two experiments to verify the performance of the midline and baseline detection algorithm. In the first experiment, we took fifty line images from segmented nutrition table images. The nutrition table images as well as line images were segmented automatically from images of real-world products using algorithms discussed previously. We discarded line images that contained either no text, multiple lines of text or curved text. We zoomed each line for better viewing and manually marked the midline and baseline in

the first fifty line images. We then compared these manually marked midlines and baselines against the midlines and baselines detected by our algorithm. Figures 6.23 and 6.24 show the error in computing the midline and baseline for these images. It can be observed that the error in midline and baseline for most images is within one pixel but there are some exceptions where the error is as high as three pixels. Figure 6.25 shows an example of an image where the error in computing the midline was three pixels. The blue colored line shows the midline detected by the algorithm and the green colored lines show the actual midline and baseline. In this case, our algorithm mistook the topline for the midline. This happens only in the case when line containing many letters with ascenders. We would like to point out that mistaking the topline for a midline will not affect the word segmentation process.

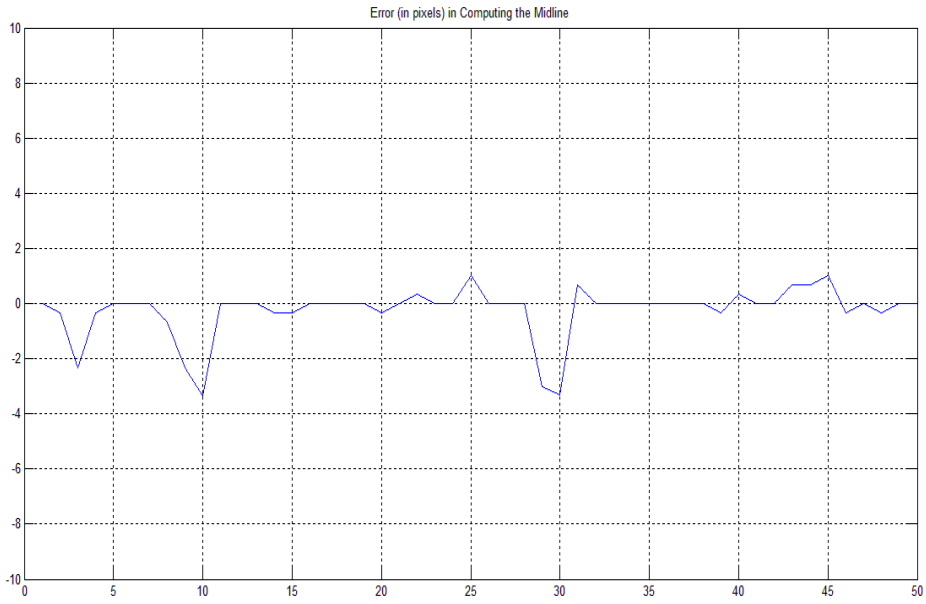


Figure 6.23. Error in detecting the midline.

We wanted to know the performance of our algorithm in a more general case (like a book) and so we segmented lines from a page of Shakespeare's Comedy of Errors. These lines were segmented automatically and no lines were excluded from the experiment. We manually identified midlines and the baselines for each line and compared them with mid-

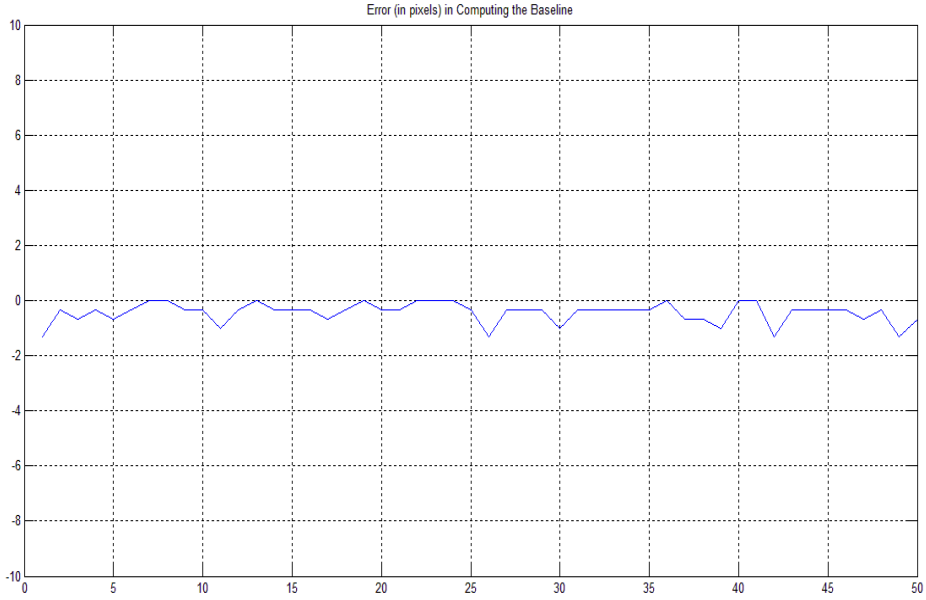


Figure 6.24. Error in detecting the baseline.

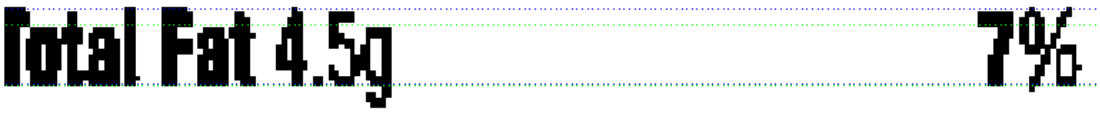


Figure 6.25. Example of topline being mistaken for the midline.

lines and baselines detected by the algorithm. Figures 6.26 and 6.27 show the error in computing the midline and baseline for these images. It can be observed that the error for all the lines is within one pixel and this proves that our algorithm works even better for the general case.

6.3.4.3 Character Segmentation

This section deals with segmenting the characters from the line image and is useful for recognizing the values associated with items like 'Calories', 'Carbohydrates', etc. The previous step should have isolated the actual numerical value as a 'word'. This step will

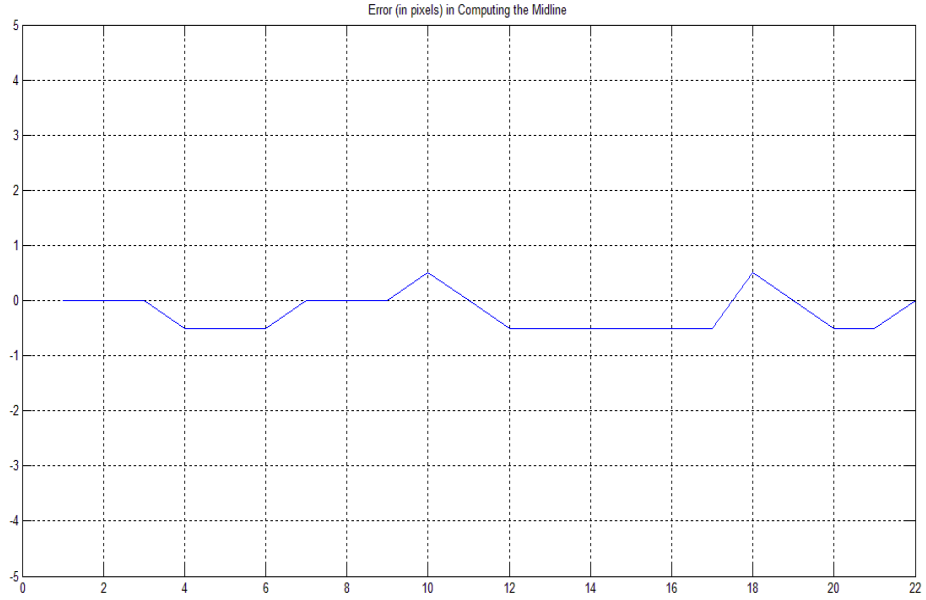


Figure 6.26. Error in detecting the midline.

break the numerical value into individual components. For example, the value 35 will be broken down into the numbers 3 and 5. These numbers can then be recognized using the technique discussed later in Section 6.4.2.

The character segmentation method works similar to the word segmentation method discussed in the previous section. We compute the vertical projection VP of the number N in each line. Figure 6.28 shows a number and its vertical projection. We can use a small threshold τ (again set to 3 for our application) to segment parts of the image that exhibit values of VP greater than τ to obtain the digits.

The observant reader may have noticed a problem with our segmentation system, namely how do we know that a particular subimage is a 'number' and needs to be further segmented? The solution to this question is to segment and classify words from each line and then use a rule-based method to determine if the next segmented subimage is a word or a number. For example, if the line is found to have words 'Calories', 'from' and 'Fat' in succession, the probability that the next subimage is a number is very high.

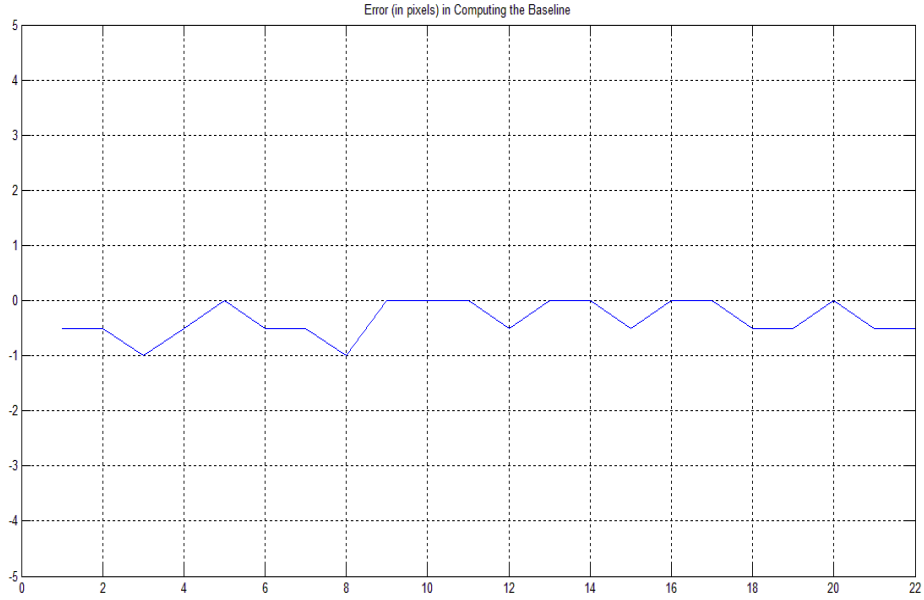


Figure 6.27. Error in detecting the baseline.

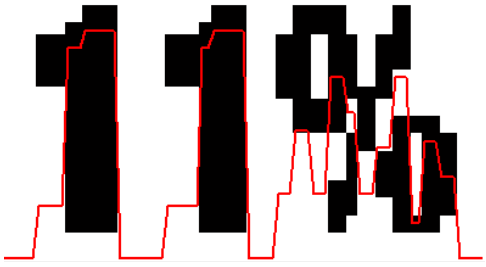


Figure 6.28. A number and its vertical projection.

6.4 Feature Extraction and Classification

The classification problem can be defined as that of matching patterns with classes. In most OCR applications, the pattern is a feature vector that describe the segmented character image and the classes are letters in the target language. Since the input to the classifier is a feature vector, the classification step is preceded by a feature extraction step. Let us look at some common feature extraction techniques.

6.4.1 Common Feature Extraction Techniques

At the very basic level, each color (red, green, blue) of each pixel in the input image can be viewed as a point in the feature space. Thus, an image containing $m \times n$ pixels, will have a total of $3mn$ points in the feature space. It is apparent that such a large feature vector contains a lot of useless information that will be impossible to manage and only make the classification harder. Devijver and Kittler [101] defined feature extraction as the problem of "extracting from the raw data the information which is most relevant for classification purposes, in the sense of minimizing the within-class pattern variability while enhancing the between-class pattern variability."

6.4.1.1 Zoning

Zoning is one of the simplest feature extraction methods. In this method, a $m \times n$ grid is superimposed on the input image and the resulting feature is a $m \times n$ vector. The i^{th} feature point in the vector is given a value of 1 if the number of foreground pixels in the i^{th} zone is greater than some threshold (τ) or 0 otherwise. Figure 6.29 shows an example of zoning. The feature vector in this case is: 0110011011111111. Bosker [102] describes a commercial OCR application using zoning.

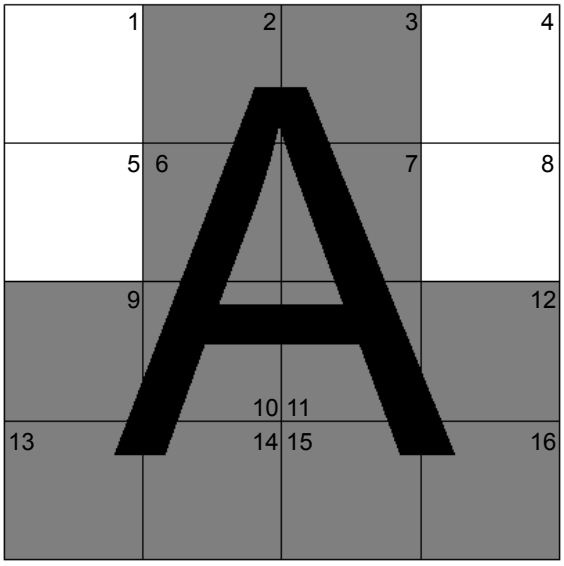


Figure 6.29. An example of zoning.

6.4.1.2 Horizontal and Vertical Projections

In this method, an input image is represented by its horizontal and vertical projections (refer to Section 6.3.1 for an explanation on horizontal and vertical projections). These continuous projections along the X and Y axes are discretized into two vectors by using two bins of size m and n, respectively. The resultant feature vector is of size $m + n$ and represents the input image. This method has the advantage of being simple but is not able to uniquely represent distinct images. For example, consider Figure 6.30 where two distinct input images A and B have the same projections along the X and Y axes.

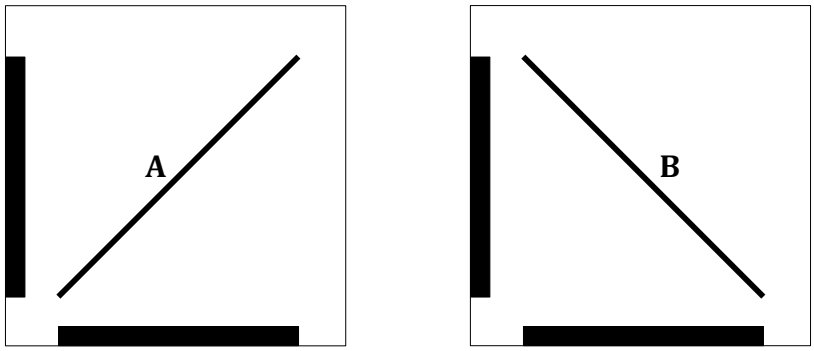


Figure 6.30. Two distinct images with the same projections.

A variant of this technique has been successfully applied in [103] for recognizing Devanagari characters. Horizontal and vertical projections are also used as part of the feature set in for font recognition [104].

6.4.1.3 Crossings and Distances

In these methods, a set of vectors along various directions are superimposed over the input image. To obtain the feature vector using the crossings methods, we compute the number of times the character image crosses each vector. Figure 6.31 shows an example of a feature vector obtained using the crossing method. To obtain the feature vector using the distances method, we compute the distance of the intersection of the image with each

vector. If the image crosses the vector multiple times we consider only the distance of the first intersection between the image and the vector. This technique is used in [105] to obtain 'transition vectors', which represent characters using the number of foreground to background pixel transition along certain predefined lines.

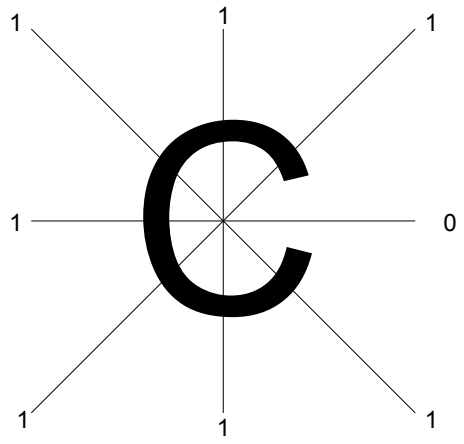


Figure 6.31. Feature vector using the crossings method.

6.4.2 Recognizing Digits using a Probabilistic Neural Network

We use a probabilistic neural network (PNN) [106, 107] to classify digits, which were segmented in the previous step. A PNN is based on a Bayesian classifier and is used to solve a diverse group of classification problems like cancer classification [108], radar/target identification [109], stock trends [110], etc. A PNN offers significant advantages over a backpropagation algorithm such as rapid training, guarantee of convergence to a Bayesian classifier, addition and deletion of data from the training set and an output indicating the amount of evidence on which it bases its decision.

To understand how a PNN works, let us first look at how a Bayesian classifier works. Consider a box that contains plastic P and wooden W balls. Given a ball at random, we have to determine if it is a plastic ball or a wooden ball by measuring its mass m. Let h_P and h_W determine the fraction (*a priori* probabilities) of plastic and wooden balls,

respectively. Let $f_P(m)$ and $f_W(m)$ denote the probabilities that a given ball of mass m is made of plastic and wood, respectively. Given a ball of mass m, we can now compute the probability that it is made of plastic $p_P(m)$ and wood $p_W(m)$ as:

$$p_P(m) = h_P \times f_P(m) \tag{6.32}$$

$$p_W(m) = h_W \times f_W(m) \tag{6.33}$$

We can now classify the ball as plastic if $p_P > p_W$ or wood if $p_P > p_W$.

The previous example works if we know the values of h_P, h_W, f_P and f_W. It is easy to determine the values of h_P and h_W since these can be computed as the fractions of plastic and wooden balls in the box. Computing the values of f_P and f_W is a little trickier. The values of f_P and f_W can be estimated by maintaining a history of the mass of plastic and wooden balls. Let $B = \{b_0 \ldots b_n\}$ represent n bins representing mass $\{m_0 \ldots m_n\}$. Let B_P and B_W denote the bins for plastic and wooden balls, respectively. These bins can be populated over time by measuring the mass of each ball and placing them in the appropriate bins. If the number of bins is sufficiently large (tending to infinity in the ideal case) and given enough samples (again tending to infinity in the ideal case), the histogram of these bins will represent the *probability density function* or *pdf* of the mass. Figure 6.32 shows an example of the probability density function. The probability that a plastic (or wooden) ball has mass between a and b can now be calculated by finding the area under the curve between the points a and b through integration.

The previous example assumes that we have an infinite amount of data to calculate the probability density functions. However, in reality, there is a limited amount of data available to us and the *pdf* has to be approximated. Parzen windows is a technique developed by Parzen that allows us to approximate the *pdf* with a relatively small amount of data. In essence, for each n-dimensional training vector, we compute n-dimensional Gaussians centered on that sample. The sum of the Gaussians for all samples will then approximate the *pdf* of the set. One advantage of using Parzen windows is that we do not need to

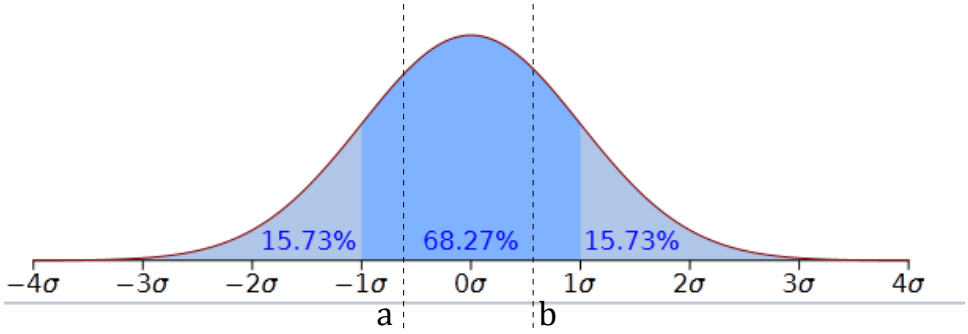


Figure 6.32. Probability density function.

calculate the full *pdf* and instead the value of the *pdf* for some class A at point X can be computed as:

$$f_a(X) = \begin{pmatrix} 1/(2*\pi)^{(p/2)}\sigma^p \\ 1/n_a \Sigma_{i=1}^{n_a}(-(X-Y_{ai})^t(X-Y_{ai})/2\sigma^2) \end{pmatrix} \tag{6.34}$$

where, $f_a(X)$ is the value of the *pdf* of class A at point X, i is the training vector number, p is the number of components in the training vector, σ is the smoothing variable, n_a is the number of training vectors in class A, X is the test vector to be classified, Y_{a_i} is the i^{th} training vector from class A and t is the vector transpose.

This technique provides us with a simple and effective way to compute the value of *pdf* at any point X. As the number of training samples increases, the estimated *pdf* approaches the true value. The value of σ determines the size (or spread) of the Gaussian. A small value of σ will cause the Gaussian to be peaked and the classifier converges to a nearest neighbor classifier. On the other hand, a large value of σ will cause the Gaussian to be flat and the classifier converges to a linearly separable classifier. We have set $\sigma = 25$ for our application.

A probabilistic neural network (PNN) uses this technique to quickly classify a test sample using a limited amount of test data. Figure 6.33 shows a high-level view of the PNN, which contains four layers - a distribution layer, a pattern layer, a summation layer and a decision later. The distribution layer serves merely as a connection point and does

not perform any computation. It connects the n-dimensional input vector X to the nodes in the pattern layer. Each node in the pattern layer corresponds to a sample in the training data set. Figure 6.34 shows one such neuron in the pattern layer. The input weights for this neuron represent the value of the training sample. Specifically, the j^{th} input weight w_j is equal to the value of the training sample along the j^{th} dimension. The neuron sums up the weighted inputs and applies the non-linear function $f(\bullet)$ described in equation 6.34 to produce the output Z. Each node in the pattern layer thus produces an output Z_{ci}, where c represents the class of the associated training vector and i represents the pattern layer neuron computing that class.

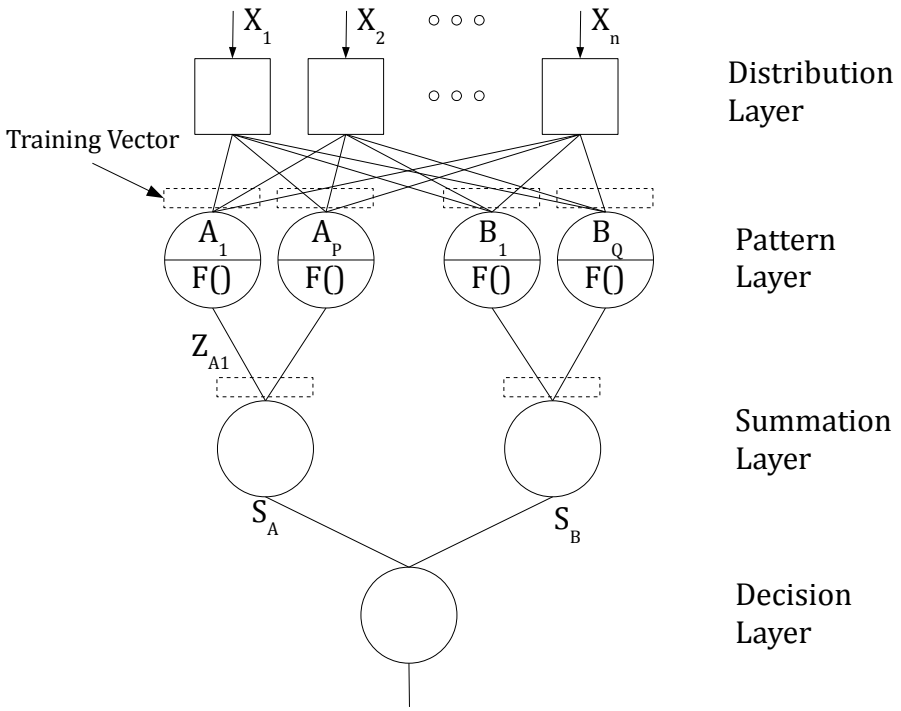


Figure 6.33. Probabilistic neural network.

The summation layer is responsible for summing up the outputs of neurons for each class c and contains a node for each class in the training sample. In other words, if our training sample contains two classes with ten samples in the first class and twenty samples in the other, the summation layer will have two nodes - one with ten inputs and the other with

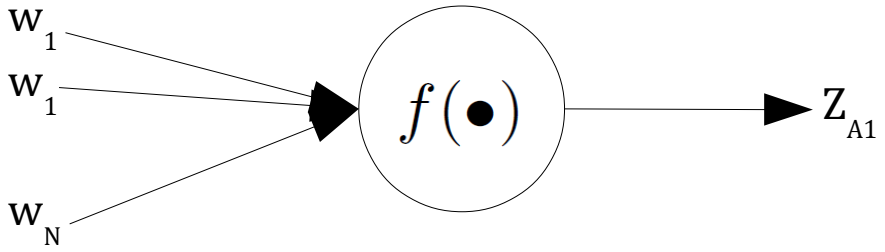


Figure 6.34. Pattern layer neuron.

twenty. The number of inputs to the summation layer are equal to the number of training samples. It is preferable that the training sample is of a small size for PNNs running on mobile devices. The summation layer passes the results of the summation to the decision layer, which contains a single node. This node has one input for each summation node (or each class c) and only one output. The output of the decision layer is the classification of the input sample. It chooses this classification by looking at the outputs of the summation nodes for each class and picking the one with the maximum value.

6.4.2.1 Experiments on Classifying Digits using the PNN

We collected images of barcodes on products sold in our campus store and manually segmented digits from them. These digits were then manually classified into ten bins numbered zero through nine. The digits were then binarized and cropped programatically. The entire sample contained a total of 308 digits. We randomly split the entire sample into a training sample (approximately 60%) and a testing sample (approximately 40%). The training sample was used to train the neural network and the testing sample was used to test the accuracy of the neural network. We ran the experiment ten times, with different randomly generated training and testing set each time. The following table summarizes the results of that experiment.

The overall accuracy was 97.66% for the entire experiment.

Table 6.1. Results of Classifying Digits using the PNN

Experiment Number	**Images Classified Correctly**	**Images Classified Incorrectly**	**Accuracy**
1	117	2	98.32
2	119	4	96.75
3	112	2	98.25
4	120	2	98.36
5	117	4	96.69
6	119	3	97.54
7	115	3	97.46
8	117	2	98.32
9	116	2	98.31
10	115	4	96.64

6.4.3 Recognizing Words in the Image

As mentioned previously, this technique is not as popular as character recognition methods chiefly because word recognition assumes a predefined and finite lexicon of words. The number of characters in a language are few when compared to the total number of words. For example, the English language contains 26 (52 - if we include both uppercase and lowercase) letters and 10 digits but well over 200,000 words [111]. It would be highly improbable, if not impossible, to generate a library containing feature vectors for all the words in a particular language.

We classify the word images segmented in the previous step but there are systems that can recognize words without segmentation [112, 113]. These systems typically match templates of words across the entire image. The first part of this section discusses feature extraction and the second part discusses classification.

6.4.3.1 Feature Extraction

We use a hybrid feature vector consisting of two simple feature vectors to represent the word image. This vector is represented as $F = [R, I]$, where R represents the height-to-width ratio of the image and I represents the grayscale image pixels. Each image in the training sample is scaled down so that the height of each image is ten pixels. We chose to

keep the height of every image the same because the width of the image can vary greatly depending on the number of characters in the word but the height of each character remains the same. This scaling is performed using a bilinear interpolation method, where the output pixel value is a weighted average of the neighboring 2×2 pixels.

6.4.3.2 Classification

We use template matching to classify words in the image. This is a popular technique and is used in many OCR systems [114, 115]. Let $F_w = [R_w, I_w]$ denote the feature vector that represents the word image that has to be classified. Let $C = \{C_1, C_2, \ldots, C_m\}$ represent the m different classes that the word image can be classified into. These classes represent the different words found in the nutrition table like 'Calories', 'Fat', etc. Table 6.2 shows the 24 different classes (or words) that we have identified for our application. For template matching, we also assume a library of precomputed feature vectors $F = \{F_1, F_2 \ldots,_n\}$, which are generated from images that are representative of the different classes in C. In general, $n \gg m$. Table 6.2 also shows the average height and width (in pixels) and total number of feature vectors for word images belonging to each of the 24 classes in our application.

We use a two-stage process to classify F_w. In the first stage, we compare the image ratio of F_w with the image ratios of all the features vectors from the library F. If R_w and R_i are comparable, we compute the error e_i between I_w and I_i. In our application, we conclude that R_w is comparable to R_i if either R_w is 70% of R_w or R_w is 70% of R_i. The error e_i is computed as the root mean squared error of I_w and I_i if the two ratios are comparable and is set to ∞ otherwise. It should be noted that I_w is scaled using bilinear interpolation to match the dimensions of I_i for each image in the library. Let F_c represent the vector for which the error e_c is minimum. We then classify the word image by assiging it to the same class as F_c.

6.4.3.3 Experiments on Word Classification

We took pictures of nutrition tables on real products and then segmented words using the segmentation method described previously. We then hand picked 465 images belonging

Table 6.2. Word Templates

Name	Average Height	Average Width	Number of Samples
Amount	10	32.7143	7
Calcium	10	31	6
Calories	10	30.875	24
Carbohydrate	10	45.3846	13
Cholesterol	10	44.8571	14
Daily	10	19	7
Dietary	10	21.4615	13
Fat	10	11.6944	72
Fiber	10	18.875	16
From	10	16.8182	11
Insoluble	10	39	1
Monounsaturated	10	60.1111	9
Per	10	13.3333	3
Polyunsaturated	10	48.3333	9
Potassium	10	41.7143	7
Protein	10	29.6364	11
Saturated	10	34	12
Serving	10	26.1667	6
Sodium	10	30.7692	13
Soluble	10	30	1
Sugars	10	22.1538	13
Total	10	20.4167	24
Trans	10	18.5714	7
Vitamin	10	27.4706	17

to 22 classes (or words) from these segmented images. The images were hand picked to ensure that the results of classification were not affected by the quality of segmentation. The selection process was not rigid. Only images that contained no words, partial words or more than one words were discarded. This set of images was then split randomly into a training set for generating the library F and a testing set. The training set contained 280 images and the testing set contained 185 images. We ran the word classification method on the testing set and found that 169 images out of 185 were correctly classified by the system and only 16 images were incorrectly classified.

CHAPTER 7

FUTURE WORK

This dissertation comprises of three components - the eyes-free barcode scanner, the tele-assistance module and the optical character recognition module. This chapter discusses the future work and improvements that can be applied towards each of these three components.

7.1 Eyes-Free Barcode Scanner

The eyes-free barcode scanner is explained in detail in Chapter 3. It is a software-only solution that uses the phone's camera and internal sensors to allow VI individuals to scan UPC barcodes on products and MSI barcodes on shelves. This module has been through various iterations that paint a picture of how it has evolved over time. The first version of the barcode scanner required shoppers to press a button on the screen to scan a barcode and did not include help the user in keeping the camera aligned with the product. The next version incorporated automatic image capture using video mode and this was later improved by adding the automatic camera alignment module. The final version incorporated the barcode detection module that allowed shoppers to quickly determine if a barcode was within the camera's field of view.

Even though the current version of the barcode scanner is a tremendous improvement over the first, there are still areas of improvement. One of the biggest challenges is to determine the correct distance between the camera and the product. If the camera is too close to the product, the barcode is not fully contained within the camera's field of view and cannot be decoded. If the camera is too far away, the barcode occupies only a small portion of the image and cannot be decoded. currently, we train the user in moving the camera so that it is only 6-8 inches away from the product but it would be nice if this process can

be automated. This will allow the barcode scanner to detect if the camera is positioned at the correct distance and let the user know when to stop moving the camera away from the product.

The second biggest improvement is to make the barcode detection and decoding process independent of the camera's orientation to the barcode. Currently, the system can detect and decode barcodes that are either vertically or horizontally aligned with the image. Allowing the system to detect and decode barcodes that are rotated arbitrarily will reduce the burden of keeping the camera aligned in the yaw plane.

The final improvement would be to improve the performance of the system. Most current smartphones have dual-core and even quad-core processors. The system should take advantage of these extra cores by parallelizing the barcode detection and decoding processes. This will result in an increase in the throughput of the system in terms of number of frames decoded per second.

7.2 TeleShop

The TeleShop system is a tele-assistance module, which is described in Chapter 5. This system allows the VI shopper to get help from sighted individuals by transmitting a video of her current surroundings. This system is intended as a fail-safe system, which can be used when the eyes-free barcode system or the OCR system fail. This system is not limited to shopping and can be used for other purposes too.

The current implementation uses a series of images to create the video feed. This system should be replaced by a real-time streaming protocol like RTSP. This will reduce the bandwidth requirement and increase performance.

The second improvement to the system can be made by removing the need for a client on the PC. In this case, the VI user's cell phone acts as a server, transmitting a true video feed, which can be accessed anywhere by anyone on the web. This would enable sighted individuals to access the video using a cell phone, tablets or PC. This method would need to implement security policies to restrict the video feed to only a select group of users.

7.3 Optical Character Recognition System

The OCR system, which is described in detail in Chapter 6, is used to read nutrition fact labels on products. This system is presented as a proof-of-concept prototype and thus has the most room for improvement among all the three components of this dissertation. The biggest area of improvement is to implement all the different components in Android and integrate them. The current implementation is implemented in MATLAB and only some components are implemented in Android. Even though we have been very careful not to use MATLAB libraries that do not have an Android counterpart, it would be best to implement all the components in Android and integrate them.

The current system assumes that the nutrition table is present in the image, is not rotated and is not cropped. The next area of improvement is to detect if the nutrition table is present in the image, if it is cropped and then provide instructions the VI shopper so that she can capture the nutrition table in its entirety. This will allow VI shoppers to use the system easily without having to spend a lot of time running the process on images that do not contain the nutrition table. This will also allow them to use the system without worrying about aligning the camera with the product.

The current system also assumes that the nutrition table contains black colored characters on a light colored background but this may not be true for all products. The system should be modified so that it can correctly read light-colored characters on a dark background.

Currently, all experiments are performed in software using images of real products. The system has to be validated by performing experiments with a VI individual in a real grocery store. This will expose problems in the user interface and user experience along with other problems relating to lighting, etc.

CHAPTER 8

CONCLUSION

Independent grocery shopping ranks amongst the highest in the list of prerequisites for visually impaired (VI) people to achieve true independence. It is something that can greatly improve the quality of life of VI individuals. Many systems have been proposed and developed to solve this problem. We have developed three such systems - RoboCart, ShopTalk and ShopMobile 1 in our laboratory to solve this problem. Each of these three systems have been successful but at the expense of instrumenting the environment or using specialized hardware. ShopMobile 2 combines the best of all three systems and removes the need of instrumenting the environment and using specialized hardware. It is a software only system that relies on a smartphone, something that has become ubiquitous in our current lifestyle.

The first component of this dissertation is an *eyes-free* barcode scanner that can be used to replace the dedicated barcode scanners used in RoboCart, ShopTalk and ShopMobile 1. We have developed an *eyes-free* barcode scanner that uses the smartphone's camera to detect and decode both UPC barcodes on products and MSI barcodes on shelves. This is a software-only solution that uses fast computer vision algorithms to quickly scan barcodes. These algorithms are modularized and can easily be used to extend the scanning functionality to other types of barcodes. We have verified the viability of our scanning solution as a replacement for dedicated hardware barcode scanners by performing several experiments in a laboratory setting as well as in a real grocery store.

The second component of this dissertation is a tele-assistance system that can be used by VI individuals to get help from sighted individuals. We admit that there are shortcomings in our system and that there are cases where the best solution is to just ask someone else for help. The tele-assistance module allows the VI individual to get help from a sighted

individual by sharing a video of her surroundings. The sighted individual can view the video using an application that allows her to change the resolution and quality of the incoming stream as well as to freeze a frame for viewing. These controls provide additional functionalities that are missing in current video sharing applications like Skype.

The final component of this dissertation is an optical character recognition (OCR) module which is a proof-of-concept prototype that allows VI individuals to read the contents of the nutrition facts table. Existing assistive shopping systems allow VI individuals to browse and select products but do not provide any indication of the product's ingredients or other nutritional information. Thus, VI shoppers have no way to distinguish between products based on their nutritional information and are restricted in their choices between different brands of the same type of product. This information, or the lack thereof, becomes even more important in case of dietary restrictions or allergies. Our system describes and implements all the necessary components that allow a VI shopper to read nutrition facts on a product.

REFERENCES

[1] National Federation for the Blind. (2011, July) Blindness statistics. Retrieved July 14, 2011. [Online]. Available: http://www.nfb.org/nfb/blindness_statistics.asp

[2] Food Marketing Institute. (2011, July) Supermarket facts. Retrieved July 14, 2011.

[3] Peapod LLC. (2011, Aug) Peapod. Retrieved Aug 30, 2011.

[4] V. A. Kulyukin and C. Gharpure, "Ergonomics-for-one in a robotic shopping cart for the blind," in *Proc. of the 1st ACM SIGCHI/SIGART Conf. on Human-robot interaction*, ser. HRI '06. New York, NY, USA: ACM, 2006, pp. 142–149. [Online]. Available: http://doi.acm.org/10.1145/1121241.1121267

[5] V. Kulyukin, C. Gharpure, and C. Pentico, "Robots as interfaces to haptic and locomotor spaces," in *Proc. of the ACM/IEEE international Conf. on Human-robot interaction*, ser. HRI '07. New York, NY, USA: ACM, 2007, pp. 325–331. [Online]. Available: http://doi.acm.org/10.1145/1228716.1228760

[6] C. Gharpure and V. Kulyukin, "Robot-assisted shopping for the blind: issues in spatial cognition and product selection," *Intelligent Service Robotics*, vol. 1, pp. 237–251, 2008, 10.1007/s11370-008-0020-9. [Online]. Available: http://dx.doi.org/10.1007/s11370-008-0020-9

[7] J. Nicholson and V. Kulyukin, "Shoptalk: Independent blind shopping = verbal route directions + barcode scans." in *Proc. of the 30-th Annual Conf. of the Rehabilitation Engineering and Assistive Technology Society of North America*, 2007.

[8] V. Kulyukin, J. Nicholson, and D. Coster, "Shoptalk: toward independent shopping by people with visual impairments," in *Proc. of the 10th international ACM SIGACCESS*

Conf. on Computers and Accessibility, ser. Assets '08. New York, NY, USA: ACM, 2008, pp. 241–242. [Online]. Available: http://doi.acm.org/10.1145/1414471.1414518

[9] J. Nicholson, V. Kulyukin, and D. Coster, "Shoptalk: Independent blind shopping through verbal route directions and barcode scans." *The Open Rehabilitation Journal*, vol. 2, 2009.

[10] M. Merler, C. Galleguillos, and S. Belongie, "Recognizing groceries in situ using in vitro training data," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[11] T. Winlock, E. Christiansen, and S. Belongie, "Toward real-time grocery detection for the visually impaired," in *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, 2010, pp. 49–56.

[12] K. Janaswami, "A Mobile Shopping Aid For Visually Impaired Individuals," Master's thesis, Utah State University, Logan, UT, 2010.

[13] L. McQuistion, "Rehabilitation engineering: Ergonomics for one," *Ergonomics in Design*, January 1993.

[14] V. Kulyukin and A. Kutiyanawala, "Accessible shopping systems for blind and visually impaired individuals: Design requirements and the state of the art," *The Open Rehabilitation Journal*, vol. 2, pp. 158–168, 2010.

[15] UCSD. (2011, July) Grozi120 database. Retrieved July 6, 2011. [Online]. Available: http://grozi.calit2.net/

[16] S. Krishna, V. Balasubramanian, N. Chatapuram Krishnan, C. Juillard, T. Hedgpeth, and S. Panchanathan, "A wearable wireless rfid system for accessible shopping environments," *3rd International ICST Conf. on Body Area Networks*, May 2008.

[17] P. E. Lanigan, A. M. Paulos, A. W. Williams, D. Rossi, and P. Narasimhan, "Trinetra: Assistive technologies for grocery shopping for the blind," in *Proc. of the 10th IEEE*

International Symposium on Wearable Computers (ISWC). Los Alamitos, CA, USA: IEEE Computer Society, October 2006, pp. 147–148.

[18] P. E. Lanigan, A. M. Paulos, A. W. Williams, and P. Narasimhan, "Trinetra: Assistive technologies for the blind," Carnegie Mellon CyLab, Pittsburgh, PA 15213, Tech. Rep. CMU-CYLAB-06-006, May 2006.

[19] P. E. Lanigan, A. M. Paulos, A. W. Williams, D. Rossi, and P. Narasimhan, "Trinetra: Assistive technologies for grocery shopping for the blind," in *International IEEE-BAIS Symposium on Research on Assistive Technologies*. Los Alamitos, CA, USA: IEEE Computer Society, April 2007, pp. 29–36.

[20] E. Ohbuchi, H. Hanaizumi, and L. Hock, "Barcode readers using the camera device in mobile phones," in *International Conf. on Cyberworlds*, November 2004, pp. 260–265.

[21] M. Rohs, "Real-world interaction with camera-phones," in *In 2nd International Symposium on Ubiquitous Computing Systems*. Springer, 2004, pp. 74–89.

[22] R. Adelmann, "Toolkit for bar code recognition and resolving on camera," in *Informatik 2006 workshop on Mobile and Embedded Interactive Systems, Phones Jump Starting the Internet of Things*, 2006.

[23] J. McCune, A. Perrig, and M. Reiter, "Seeing-is-believing: using camera phones for human-verifiable authentication," in *IEEE Symposium on Security and Privacy*, May 2005, pp. 110– 124.

[24] D. Chai and F. Hock, "Locating and decoding ean-13 barcodes from images captured by digital cameras," in *Fifth International Conf. on Information, Communications, and Signal Processing*, 2005, pp. 1595–1599.

[25] RedLaser. (2011, May) Redlaser. Retrieved May 19, 2011. [Online]. Available: http://redlaser.com/

[26] ZXing. (2011, May) Zxing. Retrieved May 19, 2011. [Online]. Available: http://code.google.com/p/zxing/

[27] J. Coughlan, R. Manduchi, and H. Shen, "Cell phone-based wayfinding for the visually impaired," in *1st International Workshop on Mobile Vision*, Graz, Austria, 2006.

[28] O. Gallo and R. Manduchi, "Reading challenging barcodes with cameras," in *Workshop on Applications of Computer Vision*, December 2009, pp. 1–6.

[29] R. Kapoor, D. Bagai, and T. S. Kamal, "A new algorithm for skew detection and correction," *Pattern Recognition*, vol. 25, pp. 1215–1229, August 2004. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2004.03.020

[30] A. Amin and S. Fischer, "A document skew detection method using the hough transform," *Pattern Analysis & Applications*, vol. 3, pp. 243–253, 2000, 10.1007/s100440070009. [Online]. Available: http://dx.doi.org/10.1007/s100440070009

[31] A. S. Abutableb, "Automatic thresholding of gray-level pictures using two-dimensional entropy," *Computer Vision, Graphics, and Image Processing*, vol. 47, pp. 22–32, July 1989. [Online]. Available: http://portal.acm.org/citation.cfm?id=65331.65333

[32] J. N. Kapur, P. Sahoo, and A. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 3, pp. 273– 285, 1985. [Online]. Available: http://www.sciencedirect.com/science/article/B7GXG-4KF75CN-1/2/8cd070b7e175a896d48e037ffc348d21

[33] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, 1979.

[34] W. Niblack, *An introduction to digital image processing.* Birkeroed, Denmark: Strandberg Publishing Company, 1985.

[35] S. Yanowitz and A. Bruckstein, "A new method for image segmentation," in *9th International Conf. on Pattern Recognition*, vol. 1, no. 1, November 1988, pp. 270–275.

[36] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998. [Online]. Available: http://www.springerlink.com/index/Q87856173126771Q.pdf

[37] M. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *Intelligent Systems and their Applications, IEEE*, vol. 13, no. 4, pp. 18–28, July 1998.

[38] N. Normand and C. Viard-Gaudin, "A two-dimensional bar code reader," in *Proc. of the 12th IAPR International Conf. on Pattern Recognition*, vol. 3, October 1994, pp. 201–203.

[39] C. Viard-Gaudin, N. Normand, and D. Barba, "A bar code location algorithm using a two-dimensional approach," in *Proc. of the Second International Conf. on Document Analysis and Recognition*, vol. 2, October 1993, pp. 45–48.

[40] S.-J. Liu, H.-Y. Liao, L.-H. Chen, H.-R. Tyan, and J.-W. Hsieh, "Camera-based bar code recognition system using neural net," in *Proc. of 1993 International Joint Conf. on Neural Networks*, vol. 2, October 1993, pp. 1301–1305.

[41] R. Howlett, S. Berthier, and G. Awcock, "Determining the location of industrial barcodes using neural networks," in *Sixth International Conf. on Image Processing and its Applications*, vol. 2, July 1997, pp. 511–515.

[42] S. Arnould, G. Awcock, and R. Thomas, "Remote bar-code localisation using mathematical morphology," in *Seventh International Conf. on Image Processing and Its Applications*, vol. 2, 1999, pp. 642–646.

[43] S. Ando and H. Hontani, "Automatic visual searching and reading of barcodes in 3-d scene," in *Proc. of the IEEE International Vehicle Electronics Conf.*, 2001, pp. 49–54.

[44] A. Tropf and D. Chai, "Locating 1-d bar codes in dct-domain," in *IEEE International Conf. on Acoustics, Speech, and Signal Processing*, vol. 2, May 2006, pp. 14–19.

[45] A. Jain and Y. Chen, "Bar code localization using texture analysis," in *Proc. of the Second International Conf. on Document Analysis and Recognition*, no. 1, October 1993, pp. 41–44.

[46] Z. Bai, Z. Yang, J. Wu, and Y. Chen, "Region localization based on rotational invariant feature and improved self organized map," in *3rd International Conf. on Intelligent System and Knowledge Engineering*, vol. 1, November 2008, pp. 703–706.

[47] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. (2011, May) Line detection. Retrieved May 19, 2011. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/linedet.htm

[48] V. Kulyukin and A. Kutiyanawala, "Eyes-free barcode localization and decoding for visually impaired mobile phone users," in *Proc. of the 2010 International Conf. on Image Processing, Computer Vision, & Pattern Recognition*, 2010.

[49] E. Tekin and J. Coughlan, "An algorithm enabling blind users to find and read barcodes," in *Workshop on Applications of Computer Vision*, December 2009, pp. 1–8.

[50] R. Muniz, L. Junco, and A. Otero, "A robust software barcode reader using the hough transform," in *Proc. of the International Conf. on Information Intelligence and Systems*, 1999, pp. 313–319.

[51] H.-Y. Liao, S.-J. Liu, L.-H. Chen, and H.-R. Tyan, "A bar-code recognition system using backpropagation neural networks," *Engineering Applications of Artificial Intelligence*, vol. 8, no. 1, pp. 81– 90, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/B6V2M-4031N2N-11/2/1edf6479d9936c234293e5e9b116d4a4

[52] Wikipedia. (2011, May) Upc barcode. Retrieved May 19, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Universal%20Product%20Code

[53] ——. (2011, May) Msi barcode. Retrieved May 19, 2011. [Online]. Available: http://en.wikipedia.org/wiki/MSI%20Barcode

[54] ——. (2011, May) Luhn algorithm. Retrieved May 19, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Luhn%20algorithm

[55] P. Peake and J. Leonard, "The use of heart rate as an index of stress in blind pedestrians." *Ergonomics*, vol. 14, no. 2, pp. 189–204, 1971.

[56] V. Garaj, P. Jirawimut, P. P. anf F Cecelja, and W. Balachandran, "A system for remote sighted guidance of visually impaired pedestrians." *The British Journal of Visual Impairment*, vol. 21, no. 2, 2003.

[57] R. Farcy, R. Leroux, A. Jucha, R. Damaschini, C. Gregoire, and A. Zoghagi, "Electronic travel aids and electronic orientation aids for blind people: technical, rehabilitation and everyday life points of view," in *Conf. on Assistive Technology for Vision and Hearing Impairment*, vol. 1, 2006.

[58] L. Ran, S. Helal, and S. Moore, "Drishti: an integrated indoor/outdoor blind navigation system and service," in *Proc. of the Second IEEE Annual Conf. on Pervasive Computing and Communications*, 2004, pp. 23–30. [Online]. Available: http://dx.doi.org/10.1109/PERCOM.2004.1276842

[59] H. Makino, I. Ishii, and M. Nakashizuka, "Development of navigation system for the blind using gps and mobile phone combination," in *Proc. of the 18th Annual International Conf. on Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine*, vol. 2, October 1996, pp. 506–507 vol.2.

[60] S. Chumkamon, P. Tuvaphanthaphiphat, and P. Keeratiwintakorn, "A blind navigation system using rfid for indoor environments," in *5th International Conf. on Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology*, vol. 2, May 2008, pp. 765–768.

[61] Google Inc. (2011, July) Google goggles. Retrieved July 29, 2011. [Online]. Available: http://www.google.com/support/mobile/bin/answer.py?hl=en\&answer=166331

[62] M. Bujacz, P. Baranski, M. Moranski, P. Strumillo, and A. Materka, "Remote guidance for the blind: A proposed teleassistance system and navigation trials," in *Conf. on Human System Interactions*, May 2008, pp. 888–892.

[63] V. Garaj, Z. Hunaiti, and W. Balachandran, "Using remote vision: the effects of video image frame rate on visual object recognition performance," *Transactions on Systems, Man, and Cybernatics*, vol. 40, pp. 698–707, July 2010. [Online]. Available: http://dx.doi.org/10.1109/TSMCA.2009.2036938

[64] V. Kulyukin and A. Kutiyanawala. (2011, July) Remote guidance in assistive shopping. r&d video. Retrieved July 14, 2011. [Online]. Available: http://www.youtube.com/user/csatlusu\#p/u/0/vmWFhOtmhuI.

[65] H. Schantz, *The history of OCR, optical character recognition.* Recognition Technologies Users Association, 1982. [Online]. Available: http://books.google.com/books?id=r1ewAAAAIAAJ

[66] K. S. Bae, K. K. Kim, Y. G. Chung, and W. P. Yu, "Character recognition system for cellular phone with camera," in *Proc. of the 29th Annual International Computer Software and Applications Conf.*, ser. COMPSAC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 539–544. [Online]. Available: http://dx.doi.org/10.1109/COMPSAC.2005.55

[67] M. Laine and O. Nevalainen, "A standalone ocr system for mobile cameraphones," in *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, September 2006, pp. 1–5.

[68] X.-P. Luo, J. Li, and L.-X. Zhen, "Design and implementation of a card reader based on build-in camera," in *17th International Conf. on Proc. of the Pattern Recognition*,

ser. ICPR '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 417–420. [Online]. Available: http://dx.doi.org/10.1109/ICPR.2004.289

[69] A. F. Mollah, S. Basu, M. Nasipuri, and D. K. Basu, "Text/graphics separation for business card images for mobile devices," *CoRR*, vol. abs/1004.0766, 2010.

[70] A. Canedo-Rodriguez, S. Kim, J. Kim, and Y. Blanco-Fernandez, "English to spanish translation of signboard images from mobile phone camera," in *IEEE Southeastcon*, March 2009, pp. 356–361.

[71] M. Hsueh, "Interactive text recognition and translation on a mobile device," Master's thesis, EECS Department, University of California, Berkeley, May 2011. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-57.html

[72] I. Hairuman and O.-M. Foong, "Ocr signage recognition with skew and slant correction for visually impaired people," in *11th International Conf. on Hybrid Intelligent Systems (HIS)*, December 2011, pp. 306–310.

[73] V. Gaudissart, S. Ferreira, C. Thillou, and B. Gosselin, "Sypole: mobile reading assistant for blind people," in *Proc. of European Signal Processing Conf., EUSIPCO*, 2005.

[74] A. O. Akyüz and E. Reinhard, "Noise reduction in high dynamic range imaging," *Journal of Visual Communication and Image Representation*, vol. 18, no. 5, pp. 366–376, October 2007. [Online]. Available: http://dx.doi.org/10.1016/j.jvcir.2007.04.001

[75] A. K. Jain, *Fundamentals of digital image processing.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[76] S. J. Ko and Y. H. Lee, "Center weighted median filters and their applications to image enhancement," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 9, pp. 984–993, September 1991.

[77] W. Zhou and D. Zhang, "Progressive switching median filter for the removal of impulse noise from highly corrupted images," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 1, pp. 78–80, January 1999.

[78] K. Toh, H. Ibrahim, and M. Mahyuddin, "Salt-and-pepper noise detection and reduction using fuzzy switching median filter," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1956–1961, November 2008.

[79] L. O'Gorman, "Image and document processing techniques for the rightpages electronic library system," in *11th IAPR International Conf. on Pattern Recognition*, vol. 2, no. 11, August 1992, pp. 260–263.

[80] H. Al-Khaffaf, A. Talib, and R. Salam, "Removing salt-and-pepper noise from binary images of engineering drawings," in *19th International Conf. on Pattern Recognition*, vol. E92.D, no. 4, December 2008, pp. 689–704.

[81] B. Singh, Mridula, V. Chand, A. Mittal, and D. Ghosh, "A comparative study of different approaches of noise removal for document images," in *Proc. of the International Conf. on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011*, ser. Advances in Intelligent and Soft Computing, K. Deep, A. Nagar, M. Pant, and J. C. Bansal, Eds. Springer India, 2012, vol. 130, pp. 847–854. [Online]. Available: http://dx.doi.org/10.1007/978-81-322-0487-9_80

[82] M. Kazubek, "Wavelet domain image denoising by thresholding and wiener filtering," *IEEE Signal Processing Letters*, vol. 10, no. 11, pp. 324–326, November 2003.

[83] T. SteinHerz, N. Intrator, and R. Ehud, "A special skeletonization algorithm for cursive words," in *7th International Workshop on Frontiers in Handwriting Recognition*, 2000, pp. 529–534.

[84] K. Palágyi and A. Kuba, "A thinning algorithm to extract medial lines from 3d medical images," in *Proc. of the 15th International Conf. on Information Processing*

in Medical Imaging, ser. IPMI '97. London, UK, UK: Springer-Verlag, 1997, pp. 411–416. [Online]. Available: http://dl.acm.org/citation.cfm?id=645595.660567

[85] Y. He, J. Tian, X. Luo, and T. Zhang, "Image enhancement and minutiae matching in fingerprint verification," *Pattern Recognition*, vol. 24, no. 9-10, pp. 1349–1360, Jun. 2003. [Online]. Available: http://dx.doi.org/10.1016/S0167-8655(02)00376-8

[86] L. Lam, S.-W. Lee, and C. Suen, "Thinning methodologies-a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 9, pp. 869–885, September 1992.

[87] P. Kwok, "A thinning algorithm by contour generation," *ACM Communications*, vol. 31, no. 11, pp. 1314–1324, Nov. 1988. [Online]. Available: http://doi.acm.org/10.1145/50087.50092

[88] P. Kardos, G. Németh, and K. Palágyi, "An order–independent sequential thinning algorithm," in *Proc. of the 13th International Workshop on Combinatorial Image Analysis*, ser. IWCIA '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 162–175. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10210-3_13

[89] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *ACM Communications*, vol. 27, no. 3, pp. 236–239, March 1984. [Online]. Available: http://doi.acm.org/10.1145/357994.358023

[90] C. M. Holt, A. Stewart, M. Clint, and R. H. Perrott, "An improved parallel thinning algorithm," *ACM Communications*, vol. 30, no. 2, pp. 156–160, February 1987. [Online]. Available: http://doi.acm.org/10.1145/12527.12531

[91] S.-C. Pei and C.-N. Lin, "Image normalization for pattern recognition," *Image and Vision Computing*, vol. 13, no. 10, pp. 711– 723, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/026288569598753G

[92] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, February 1962.

[93] Y. Nakano, Y. Shima, H. Fujisawa, J. Higashino, and M. Fujinawa, "An algorithm for the skew normalization of document image," in *Proc. of the 10th International Conf. on Pattern Recognition*, vol. 2, June 1990, pp. 8–13.

[94] J. Liu, C.-M. Lee, and R.-B. Shu, "An efficient method for the skew normalization of a document image," in *11th IAPR International Conf. on Pattern Recognition*, August 1992, pp. 122–125.

[95] H. Al-Yousefi and S. Udpa, "Recognition of arabic characters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 8, pp. 853–857, August 1992.

[96] Y. Lu, "Machine printed character segmentation an overview," *Pattern Recognition*, vol. 28, no. 1, pp. 67–80, 1995.

[97] R. Casey and E. Lecolinet, "Strategies in character segmentation: a survey," in *Proc. of the Third International Conf. on Document Analysis and Recognition*, vol. 2, August 1995, pp. 1028–1033 vol.2.

[98] J. McClurg-Genevese. (2005, August) The elements of design. [Online]. Available: http://www.digital-web.com/articles/elements_of_design

[99] G. Seni and E. Cohen, "External word segmentation of off-line handwritten text lines," *Pattern Recognition*, vol. 27, no. 1, pp. 41–52, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0031320394900167

[100] R. Manmatha and N. Srimal, "Scale space technique for word segmentation in handwritten documents," in *Proc. of the Second International Conf. on Scale-Space Theories in Computer Vision*, ser. SCALE-SPACE '99. London, UK, UK: Springer-Verlag, 1999, pp. 22–33. [Online]. Available: http://dl.acm.org/citation.cfm?id=647082.715226

[101] P. Devyver and J. Kittler, *Pattern Recognition: A Statistical Approach.* Prentice-Hall, 1982. [Online]. Available: http://books.google.com/books?id=Em9QAAAAMAAJ

[102] M. Bokser, "Omnidocument technologies," *Proc. of the IEEE*, vol. 80, no. 7, pp. 1066–1078, 1992.

[103] V. Bansal and R. Sinha, "A complete ocr for printed hindi text in devanagari script," in *Proc. of the Sixth International Conf. on Document Analysis and Recognition*, 2001, pp. 800–804.

[104] A. Zramdini and R. Ingold, "Optical font recognition using typographical features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 877–882, August 1998.

[105] M. Mohamed and P. Gader, "Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 5, pp. 548–554, May 1996.

[106] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/089360809090049Q

[107] P. D. Wasserman, *Advanced Methods in Neural Computing*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[108] D. P. Berrar, C. S. Downes, W. Dubitzky, D. P. Berrar, C. S. Downes, and W. Dubitzky, "Multiclass cancer classification using gene expression profiling and probabilistic neural networks," in *In Proc. of the Pacific Symposium on Biocomputing (PSB*, 2003, pp. 5–16.

[109] Q. Zhao and Z. Bao, "Radar target recognition using a radial basis function neural network," *Neural Networks*, vol. 9, no. 4, pp. 709–720, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0893608096000883

[110] E. W. Saad, S. Member, D. V. Prokhorov, D. C. Wunsch, and S. Member, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," *IEEE Transactions on Neural Networks*, pp. 1456–1470, 1998.

[111] Oxford Dictionary. (2012). [Online]. Available: http://oxforddictionaries.com/words/how-many-words-are-there-in-the-english-language

[112] J. Rocha and T. Pavlidis, "New method for word recognition without segmentation," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series, vol. 1906, April 1993, pp. 74–80.

[113] C. Chen and J. DeCurtins, "Word recognition in a segmentation-free approach to ocr," in *Proc. of the Second International Conf. on Document Analysis and Recognition*, October 1993, pp. 573–576.

[114] B. Chaudhuri and U. Pal, "An ocr system to read two indian language scripts: Bangla and devnagari (hindi)," in *Proc. of the Fourth International Conf. on Document Analysis and Recognition*, vol. 2, August 1997, pp. 1011–1015.

[115] Y. Xu and G. Nagy, "Prototype extraction and adaptive ocr," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 12, pp. 1280–1296, December 1999.