Vision Based Extraction of Nutrition Information from Skewed Nutrition Labels

Tanwir Zaman
Utah State University

VISION BASED EXTRACTION OF NUTRITION INFORMATION FROM SKEWED NUTRITION LABELS

by

Tanwir Zaman

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

of

DOCTOR OF PHILOSOPHY

in

Computer Science

Approved:

Dr. Vladimir Kulyukin Major Professor	Dr. Nicholas Flann Committee Member
Dr. Xiaojun Qi Committee Member	Dr. Haitao Wang Committee Member
Dr. David Paper Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2016

ABSTRACT

Vision Based Extraction Of Nutrition Information

From Skewed Nutrition Labels

by

Tanwir Zaman, Doctor of Philosophy

Utah State University, 2016

Major Professor: Dr. Vladimir Kulyukin
Department: Computer Science

An important component of a healthy diet is the comprehension and retention of nutritional information and understanding of how different food items and nutritional constituents affect our bodies. In the U.S. and many other countries, nutritional information is primarily conveyed to consumers through nutrition labels (NLs) which can be found in all packaged food products. However, sometimes it becomes really challenging to utilize all this information available in these NLs even for consumers who are health conscious as they might not be familiar with nutritional terms or find it difficult to integrate nutritional data collection into their daily activities due to lack of time, motivation, or training. So it is essential to automate this data collection and interpretation process by integrating Computer Vision based algorithms to extract nutritional information from NLs because it improves the user's ability to engage in continuous nutritional data collection and analysis. To make nutritional data collection more manageable and enjoyable for the users, we present a Proactive NUTrition Management System (PNUTS). PNUTS seeks to shift current research and clinical practices in nutrition management toward persuasion, automated nutritional information processing, and context-sensitive nutrition decision support.

PNUTS consists of two modules, firstly a barcode scanning module which runs on smart phones and is capable of vision-based localization of One Dimensional (1D) Universal Product Code (UPC) and International Article Number (EAN) barcodes with relaxed pitch, roll, and yaw camera alignment constraints. The algorithm localizes barcodes in images by computing Dominant Orientations of Gradients (DOGs) of image segments and grouping smaller segments with similar DOGs into larger connected components. Connected components that pass given morphological criteria are marked as potential barcodes. The algorithm is implemented in a distributed, cloud-based system. The system's front end is a smartphone application that runs on Android smartphones with Android 4.2 or higher. The system's back end is deployed on a five node Linux cluster where images are processed. The algorithm was evaluated on a corpus of 7,545 images extracted from 506 videos of bags, bottles, boxes, and cans in a supermarket. The DOG algorithm was coupled to our in-place scanner for 1D UPC and EAN barcodes. The scanner receives from the DOG algorithm the rectangular planar dimensions of a connected component and the component's dominant gradient orientation angle referred to as the skew angle. The scanner draws several scan lines at that skew angle within the component to recognize the barcode in place without any rotations. The scanner coupled to the localizer was tested on the same corpus of 7,545 images. Laboratory experiments indicate that the system can localize and scan barcodes of any orientation in the yaw plane, of up to 73.28 degrees in the pitch plane, and of up to 55.5 degrees in the roll plane. The videos have been made public for all interested research communities to replicate our findings or to use them in their own research. The front end Android application is available for free download at Google Play under the title of NutriGlass. This module is also coupled to a comprehensive NL database from which nutritional information can be retrieved on demand. Currently our NL database consists of more than 230,000 products.

The second module of PNUTS is an algorithm whose objective is to determine the text skew angle of an NL image without constraining the angle's magnitude. The horizontal, vertical, and diagonal matrices of the (Two Dimensional) 2D Haar Wavelet Transform are used to identify 2D points with significant intensity changes. The set of points is bounded with a minimum area rectangle whose rotation angle is the text's skew. The algorithm's performance is compared with the performance of five text skew detection algorithms on 1001 U.S. nutrition label images and 2200 single- and multi-column document images in multiple languages. To ensure the reproducibility of the reported results, the source code of the algorithm and the image data have been made publicly available. If the skew angle is estimated correctly, optical character recognition (OCR) techniques can be used to extract nutrition information.

(110 pages)

PUBLIC ABSTRACT

Vision Based Extraction Of Nutrition Information

From Skewed Nutrition Labels

by

Tanwir Zaman, Doctor of Philosophy

Utah State University, 2016

Major Professor: Dr. Vladimir Kulyukin
Department: Computer Science

Vision-based extraction of nutritional information from nutrition labels (NLs) available on most product packages is critical to proactive nutrition management, because it improves the user's ability to engage in continuous nutritional data collection and analysis. However, even users who are health conscious find it difficult to keep track of their nutrition intake due to lack of time, motivation, or training. In order to make nutrition management more proactive we present a Proactive NUTrition Management System (PNUTS), which aims to make nutrition management more user friendly and proactive using computer vision techniques running on smartphones which are ubiquitous and powerful computers at the same time. There are essentially two modules in PNUTS. First of all, a skewed barcode scanning module capable of reading barcodes irrespective of the camera alignment and a second module which can detect the skew angle of text in nutrition labels and eventually read the text using optical character recognition.

ACKNOWLEDGMENTS

I am deeply indebted to my major professor, Dr. Vladimir Kulyukin, for his advice, guidance, and patience throughout my graduate level education. His recommendations and suggestions have been invaluable to me.

I would also like to thank the members of my committee, Dr. Nicholas Flann, Dr. Xiaojun Qi, Dr. Haitao Wang and Dr. David Paper for all their help and suggestions on this dissertation.

Finally, I appreciate the continuous support and encouragement of my beloved parents throughout the duration of my academic pursuits.

Tanwir Zaman

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

The U.S. Department of Agriculture estimates that U.S. residents have increased their caloric intake by 530 calories per day since 1970 [1]. According to the World Health Organization (www.who.int), obesity causes such diseases as diabetes, kidney failures, and strokes and predicts that these diseases will be a major cause of death worldwide [2]. Mismanaged diets are estimated to account for 30-35 percent of cancer cases. While there is no permanent cure for diabetes or cancer as of now, many nutritionists and dieticians consider proactive nutrition management to be a key factor in reducing and controlling cancer, diabetes, and other illnesses related to or caused by mismanaged diets. In this regard, a proper understanding of nutrition labels (NLs) is essential to ensure eating a healthy, balanced diet. Many products sold worldwide have nutrition labels (NLs). In the U.S., the display of nutrition information is mandated by the Nutrition Education and Labeling Act (NLEA) of 1990 [3]. Similar initiatives and acts (e.g., EU FLABEL [4]) exist in other countries. These labels provide information on the amounts of thirteen core nutrients and calories in an amount of food, along with a daily value indicator to help people make informed decisions over food choices [5]. This data is presented in the form of a standardized table. Familiarity with the terms of the NLs allows a consumer to make a better decision while shopping for packaged food products and comparing one product with another. However, sometimes the data in such labels may be difficult to interpret by the common user who may find it difficult to locate and to comprehend nutritional terms on many products [6].

Technology can play a key role in this comprehension process and provide the consumers the ability to choose what is right for them by making the nutrition information more suited to user interpretation. In a grocery store environment a readily available means of technology is the mobile phone which remains severely under-utilized. The immense processing power of such devices can be put to use to extract nutritional information from NLs available on most product packages, using computer vision based techniques to implement a proactive nutrition management system which will improve the user's ability to engage in continuous nutritional data collection and analysis.

Modern nutrition management systems assume that patients understand how to collect nutritional data and can be triggered into data collection with digital prompts (e.g., email or SMS) [7]. Such systems often underperform, because many patients find it difficult to integrate nutritional data collection into their daily activities due to lack of time, motivation, or training. Eventually they turn off or ignore digital stimuli [8]. To overcome these challenges, we have begun to develop a Persuasive NUTrition Management System (PNUTS). PNUTS seeks to shift current research and clinical practices in nutrition management toward persuasion, automated nutritional information extraction and processing, and context-sensitive nutrition decision support. PNUTS is based on a nutrition management approach inspired by the Fogg Behavior Model (FBM) [8], which states that motivation alone is insufficient to stimulate target behaviors. Even a motivated user must have both the *ability* to execute a behavior and a *trigger* to engage in that behavior at an appropriate place and time. Many nutrition management system designers assume that consumers and patients are either more skilled than they actually are or that they can be trained to obtain the required skills. Since training is difficult and time consuming, a more promising path is to make target behaviors easier and more intuitive to execute. PNUTS utilizes the relative advantages of mobile and cloud computing to improve nutrition information comprehension and automate real-time vision-based NL analysis and nutrition intake recording [9, 10].

In PNUTS, there are two kinds of nutrition information extraction techniques: 1) Utilizing barcode localization and scanning to extract nutrition information from online product information databases and 2) Extraction of nutrition information from localized NLs using real time OCR.

1.2 Barcode Scanning

Two algorithms for in-place vision-based skewed barcode scanning were designed and developed which do not require any smartphone camera alignment. The first algorithm localizes skewed barcodes in captured frames by computing dominant orientations of gradients (DOGs) of image segments and collecting smaller segments with similar DOGs into larger connected components. The second algorithm localizes skewed barcodes by growing edge alignment trees (EATs) on binary images with detected edges. Since our experiments showed that the DOG algorithm outperformed the EAT algorithm [10], the current version of PNUTS uses the DOG algorithm for barcode localization. The localized barcodes are then scanned without any image rotation. Our current barcode scanning algorithm handles both UPC and EAN formats. Unlike other barcode scanning solutions (e.g., https://github.com/zxing/zxing, http://redlaser.com), our algorithm does not require the user to align the smartphone's camera with the barcode and can detect skewed or aligned barcodes anywhere in the image. Recognized barcodes are used to retrieve NLs from a database of NLs. More than 230,000 packed food products have been indexed and a database of NLs has been created after crawling and screen scraping public websites. The NLs are referenced using the product barcode. A smartphone app has been launched on Google play [11] where a user can scan a barcode and get searchable nutrition label, provided the product is present in our database. The problem with this arrangement is that of a constant need to update the database and reliance upon commercial web sites which currently permit crawlers but may prohibit them in the future.

There is a plethora of similar nutrition related smartphone apps available nowadays that seem to help consumers become more alert about their diet and keep track of their daily nutrition intake. Most of these apps allow the users to scan the aligned product barcodes and retrieve the nutrition information. [http://www.fooducate.com/]. However, the primary problem with these nutrition apps is that they too rely on pre-populated databases of food products and may not have updated information.

It is hard to keep adding newer food products to a database as there is a huge if not unlimited supply of products available in supermarkets. The only solution for the consumer would be to not rely on pre-populated databases and scan the information directly from product packages and extract as much information as possible in real time to maintain a record of food products consumed, thereby keeping track of daily food intake and maintain a healthy lifestyle.

1.3 Nutrition Label Scanning

Vision based scanning of NLs utilizes the smartphone cameras to capture images of NLs for subsequent processing and extraction of nutrition information. This includes localizing the NL in the image and using an OCR engine to read the text data from the NL. Varying degrees of skew may be introduced into the images while capturing especially using handheld cameras or smartphones. Skew angle is the angle that the text lines in the digital image makes with the horizontal direction. Text in such cases is rotated or distorted and degrades the performance of further processing and may seriously affect the performance of subsequent stages of segmentation and recognition, since the contemporary OCR systems cannot handle rotated text and perform well only in recognizing texts that are linearly aligned. While the horizontally aligned text is easily detected and recognized, skewed text poses a challenge to recognition. In most existing OCR systems, a skew correction process is often performed prior to recognition, should a need arise. Most skew estimation techniques deal with small skew angles less than 15 degrees but perform poorly for images, which contain text lines that are oriented in arbitrary directions.

In our previous research, we developed a vision-based localization algorithm for horizontally or vertically aligned nutrition labels (NLs) on smartphones [12]. Our next NL processing algorithm [13] improved on the algorithm proposed in [12] in that it handled not only aligned NLs but also NLs skewed up to 35-40 degrees from the vertical axis of the captured frame. A limitation of that algorithm was its inability to handle arbitrary text skew angles. In this dissertation an algorithm is presented whose objective is to determine the text skew angle of an NL in the image without constraining the angle's magnitude.

1.3.1 Text Skew Angle Detection

An algorithm for text skew angle detection has been designed that utilizes 2D Haar Wavelet Transform (2DHWT) and is called Text Skew Angle Wavelets (TSAW). The algorithm takes an NL image and applies several iterations of the 2DHWT to downsample the image and to compute horizontal, vertical, and diagonal change matrices. The horizontal, vertical, and diagonal matrices are used to identify a set of 2D points with significant intensity changes. These points form a singularity point set in the 2D plane. The convex hull algorithm [14] is applied to this set to enclose it with a minimum area rectangle. The text's skew is the enclosing rectangle's rotation angle relative to the absolute north.

1.3.2 Optical Character Recognition

If the skew angle is estimated correctly, that information can be used in two ways:

1. The image can be rotated accordingly so that the standard optical character recognition (OCR) techniques can be used to extract nutrition information.
2. An OCR engine can be designed that can read skewed text. Currently there are no such

OCR engines available.

The working of such an engine requires the following steps:

a. Angular Text Row Segmentation

b. Skeletonization

c. Character Segmentation .

d. Zone Vector Matching

Angular Text Row Segmentation(ATRS) involves the segmentation of the text rows of the NL that contain nutrition information about a particular nutrient. This is done to improve the performance of the OCR engine, so that it can process the NL text in a line by line basis. This might also speed up the process with the use of parallelization by delegating the OCR task to multiple engines running in parallel on different machines.

Skeletonization is defined as the process by which characters are reduced to skeletons, a set of thin lines (one pixel thick), attached to one another in a few connection points that preserve the topological and geometric properties of its originating object. This is done in order to standardize the shapes and sizes of the fonts in the NL and thereby remove variability. The challenge is to preserve the original shape of the character.

Character Segmentation is primarily done in order to extract the individual characters in a text row. This allows the OCR engine to match one character at a time with pre computed templates. The process involves the determination of the character boundaries invariant of rotation and skew. Segmentation of individual letters is achieved through a formalism that is known as histogram analysis at an angle.

Zone Vector Matching (ZVM) is a computer vision technique used for matching images with templates. The entire image is divided into several sections called zones and a specific statistic is computed for each. This statistic might be anything that uniquely identifies a zone, such as the number of pixels, the number of edges or some other feature. This generates a one dimensional vector. If we want to match two images, we can compute two zone vectors, one for each image and then match them by computing their cosine similarity index. In the OCR engine, the zone vector of a segmented character is matched against the zone vectors of a set of templates which need to be pre computed and normalized to the same resolution as the segmented character.

1.4 Research Scope

The primary purpose of this research is to test the hypothesis that nutrition information can be extracted using vision based techniques, from images of packaged food products, which were captured using a handheld smartphone camera in a grocery store environment with varying illumination and with or without any proper alignment of the camera with the NL. This arrangement might introduce significant skewness in the NL alignment which severely affects a standard OCR engine's capability of reading text from such an image. Before the research work presented in this dissertation no technique had been proposed that could scan skewed barcodes or read text from a skewed NL. This dissertation presents an algorithm that can scan a 1D barcode skewed in the yaw or pitch plane and rotated to any degree. It also presents a framework that can be used to detect the skew angle of an NL without any constraints on the rotation magnitude and subsequently extract nutrition information from it by using a novel OCR engine capable of reading skewed text.

The research was conducted in three phases. The first was the development of an application to scan skewed barcodes using simple computer vision based techniques executing in real time on smartphones, and use it to retrieve NL data from a pre populated database of packaged food products indexed by barcodes. The second phase was the development of an algorithm for estimating the angle of skew in a text document image without putting any restrictions on the angle of skew. The third and final phase was to implement an OCR engine that can read the textual information from a skewed or aligned NL image.

The major contribution of this work is the development of a system which can localize and detect the text from real world NL images which may be rotated by any angle in the two dimensional plane. During the development of this system various other contributions were made, including a first of its kind skewed barcode scanner and an OCR engine capable of reading skewed text.

The rest of the dissertation is organized as follows. In chapter 2, we give some background information and discuss related work. In chapter 3, we present the Dominant Orientation of Gradients (DOG) algorithm for skewed barcode localization and scanning. In chapter 4, we present the Text Skew Angle detection algorithm. In chapter 5 we describe the skewed OCR engine and its comparison with a standard open source OCR engine. In chapter 6, we present our conclusions and outline several directions for our future work.

CHAPTER 2

RELATED WORK

2.1 Introduction

In this chapter we discuss the background and related work. We divide the chapter into two sections viz. related work on barcode detection and scanning and text skew angle detection and reading.

2.2 Barcode Localization and Scanning

The use of smartphones to detect barcodes has been the focus of many research and development efforts for a long time. Given the ubiquity and ever increasing computing power of smartphones, they have emerged as a preferred device for many researchers to implement and test new techniques to localize and scan barcodes. Open source and commercial smartphone applications, such as RedLaser (redlaser.com) and ZXing (code.google.com/p/zxing), have been developed. However, these systems require the barcodes to be aligned with camera.

Tekin and Coughlan [15,16] have designed a vision - based algorithm to guide visually impaired smartphone users to center target barcodes in the camera frame via audio instructions and cues. However, the smartphone cameras must be aligned with barcode surfaces and the users must undergo training before they can use the mobile application in which the algorithm is implemented. Image analysis and pattern recognition methods are used by Wachenfeld et al. [17] to design a vision-based algorithm that detects barcodes on smart phones. The algorithm overcomes typical distortions, such as inhomogeneous illumination, reflections, or blurriness due to camera movement. However, a barcode is assumed to be present in the image. Nor does the algorithm appear to address the localization and scanning of barcodes misaligned with the surface in the pitch, roll, and yaw planes.

Adelmann et al. [18] have developed a randomized vision-based algorithm for scanning barcodes on mobile phones. The algorithm relies on the fact that, if multiple scanlines are drawn across the barcode in various arbitrary orientations, one of them might cover the whole length of the barcode and result in successful barcode scans. This recognition scheme does not appear to handle distorted or misaligned images.

Lin et al. [19] developed an automatic barcode detection and recognition algorithm for multiple and rotation invariant barcode decoding. However, the system requires custom hardware. In particular, the proposed system is implemented and optimized on a DM6437 DSP EVM board, a custom embedded system built specifically for barcode scanning.

Gallo and Manduchi [20] present an algorithm for 1D barcode reading in blurred, noisy, and low resolution images. However, the algorithm detects barcodes only if they are slanted by less than 45 degrees in the yaw plane. The researchers appear to make no claims on the ability of their algorithm to handle barcodes misaligned in the pitch and roll planes.

Peng et al. [21] present a smartphone application that helps blind users locate EAN barcodes and expiration dates on product packages. It is claimed that, once barcodes are localized, existing barcode decoding techniques and OCR algorithms can be utilized to obtain the required information. The system provides voice feedback to guide the user to point the camera to the barcode of the product, and then guide the user the point the camera to the expiration date for OCR. The system requires user training and does not appear to handle misaligned barcodes. A common weakness of many barcode scanners, both open source and commercial is the camera alignment requirement: the smartphone camera must be aligned with a target barcode to obtain at least one complete scanline for successful barcode recognition. This requirement is acceptable for sighted users but presents a serious accessibility barrier to visually impaired shoppers or to shoppers who may not have good dexterity. Skewed barcode scanning is also beneficial for sighted smartphone users, because it may make barcode scanning faster because the camera alignment requirement no longer needs to be satisfied.

2.3 Text Skew Angle Detection

A variety of algorithms have been developed to determine the text skew angle. They can be classified according to the approaches that they take to solve the problem.

The first category of algorithms typically use horizontal or vertical projection profiles. A horizontal projection profile is a 1-dimensional array whose size is equal to the number of rows in the image. Similarly, a vertical projection profile is a 1-dimensional array whose size is equal to the number of columns in the image. Each location in a projection profile stores a count of the number of black pixels associated with text in the corresponding row or column of the image. Projections can be thought of as1-dimensional histograms. A horizontal projection histogram is computed by rotating the input image through a range of angles and calculating black pixels in the appropriate bins. All projection profiles for all rotation angles are compared with each other to determine which one maximizes a given criterion function.

The concept of projection profiles was pioneered [22] and subsequently patented by Postl [23]. Postl's algorithm uses the horizontal projection profile for text skew angle detection. The algorithm calculates the horizontal projection profiles for angles between 0 and 180 degrees in small increments, e.g., 5 degrees. The algorithm uses the sum of squared differences between adjacent elements of the projection profile as the criterion function and chooses the profile that maximizes that value.

Hull [24] proposes a text skew angle detection algorithm similar to Postl's. Hull's algorithm is more efficient, because it rotates individual pixels instead of rotating entire images. Specifically, the coordinates of every black pixel are rotated to save temporary storage and thereby to reduce the computation that would be required for a brute force implementation.

Bloomberg et al. [25] also use projection profiles to determine the text skew angle. Their algorithm differs from Postl's and Hull's algorithms in that the images are down sampled before the projection profiles are calculated in order to reduce computational costs. The criterion function used to estimate the text skew angle is the variance of the number of black pixels in a scan line.

Kanai and Bagdanov [26] present another text skew angle estimation algorithm based on projection profiles. The algorithm extracts fiducial points and uses them as points of reference in the image by decoding the lowest resolution layer of the JBIG compressed image. The JBIG standard consists of two techniques, a progressive encoding method and a lossless compression method for the lowest resolution layer. These points are projected along parallel lines into an accumulator array. The text skew angle is computed as the angle of projection within a search interval that maximizes alignment of the fiducial points. This algorithm detects a skew angle in the limited range from ±5 degrees to ±45 degrees.

Papandreou and Gatos [27] use vertical projections for text skew detection with the criterion function being the sum of squares of the projection elements. This method is claimed to be resistant to noise and image warping and to work best for the languages where most characters include at least one vertical line, which is true for Latin-based languages. In a more recent publication [28], Papandreou et al. report using minimum bounding box areas of combined horizontal and vertical projection profiles to determine document text skews. They claim that this approach is more resistant to noise and image warp, has no range restrictions on text skews, and is well suited for printed documents.

Li, Shen, and Sun [29] combine projection profiles with wavelet decomposition. Document images are divided into sub-images with the discrete wavelet transform (DWT). The matrix with the absolute values of the horizontal sub-band coefficients is rotated through a range of angles. A step size of 2 degrees is used to compute an initial estimate of the skew angle α. A finer search is then executed from $\alpha - 1$ to $\alpha + 1$ with a step of 0.5 degrees. The algorithm is evaluated on a data set with skews from 0 to ±15 degrees.

Shivakumara et. al. [30] propose a document skew angle estimation approach based on linear regression. They use linear regression formula in order to estimate a skew angle for each text line segment of a text document. The part of the text line is extracted using static and dynamic thresholds from the projection profiles. This method is based on the assumption that there is space between text lines. The method loses accuracy for the documents having skew angle greater than 30 degrees and appears to work best for printed documents with well separated lines.

Second category of algorithms use texture based approaches to estimate document skew angles. Algorithms in this category compute discriminative features on blocks of text using image filters to determine patterns that are unique to the language or the script.

Chaudhury et. al [31] proposed using a frequency domain representation of projection profiles of horizontal text lines. Busch et. al [32] present an extensive evaluation of a broad number of texture features, including projection profiles, Gabor and wavelet features and gray-level co-occurrence matrices for detecting the script. This category of approaches has the drawbacks of requiring large and aligned homogeneous regions of text in one script, and of the features in question often being neither very discriminative nor reliable to compute in the presence of noisy or skewed text.

The third category comprises of algorithms which implement connected component based methods. These utilize shape and stroke characteristics of individual connected components.

Hochberg et. al [33] proposed using script-specific templates by clustering frequently occurring character or word shapes. Spitz et. al [34] construct shape codes that capture the concavities of characters, and use them to first classify them as Latin-based or Han-based, and then within those categories using other shape-features.

Ma et al [35] use Gabor-filters with a nearest-neighbor classifier to determine script and font-type at the word-level. Several hybrid variants of local and global approaches have also been suggested [36].

CHAPTER 3

VISION BASED LOCALIZATION OF SKEWED BARCODES

3.1 Introduction

One way to improve the comprehension and retention of nutritional information by consumers is to use computer vision to scan barcodes in order to retrieve NLs from databases. A common weakness of many 1D barcode scanners, both free and commercial, is the camera alignment requirement: the smartphone camera must be horizontally or vertically aligned with barcodes to obtain at least one complete scanline for successful barcode recognition. Another weakness of the current mobile smartphone scanners is lack of coupling of barcode scanning to comprehensive NL databases from which nutritional information can be retrieved on demand. In this chapter, we will describe an algorithm for in-place vision-based skewed barcode scanning that no longer requires the smartphone camera alignment. The algorithm is in-place in that it performs no rotation of input images to align localized barcodes for scanning. The algorithm is cloud-based, because image processing is done in the cloud. The algorithm is implemented in a distributed, cloud-based system. The system's front end is a smartphone application that runs on Android 4.3 or higher. The system's back end is currently deployed on a four node Linux cluster used for image recognition and nutritional data storage.

3.2 Barcode Localization Algorithm - Dominant Orientation of Gradients

The algorithm is based on the observation that barcodes characteristically exhibit closely spaced aligned edges with the same angle, which sets them apart from text and graphics. Let I be an RGB image and let f be a linear relative luminance function computed from a pixel's RGB components:

$$f(R,G,B) = 0.2126R + 0.7152G + 0.0722B. \tag{3.1}$$

The gradient of *f* and the gradient's orientation θ can then be computed as follows:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]; \theta = \tan^{-1}\left(\frac{\partial f}{\partial x} \Big/ \frac{\partial f}{\partial y}\right). \tag{3.2}$$

Let *M* be an *n* x *n* mask, *n* > 0, convolved with *I*. Let the dominant orientation of gradients of *M*, *DOG(M)*, be the most frequent discrete gradient orientation of all pixels covered by *M*. Let *(c, r)* be the column and row coordinates of the top left pixel of *M*. The regional gradient orientation table of *M*, *RGOT(c, r)*, is a map of discrete gradient orientations to their frequencies in the region of *I* covered by *M*. The global gradient orientation table (*GGOT*) of *I* is a map of the top left coordinates of image regions covered by *M* to their RGOTs. In our implementation, both GGOTs and RGOTs are implemented as hash tables. Figure 3.1 shows the logical organization of an image's GGOT. Each GGOT maps *(c, r)* 2-tuples to RGOT tables that, in turn, map discrete gradient orientations (i.e., GO1, GO2, …, GOk in Figure 3.1) to their frequencies (i.e., FREQ1, FREQ2, …, FREQk in Figure 3.1) in the corresponding image regions.

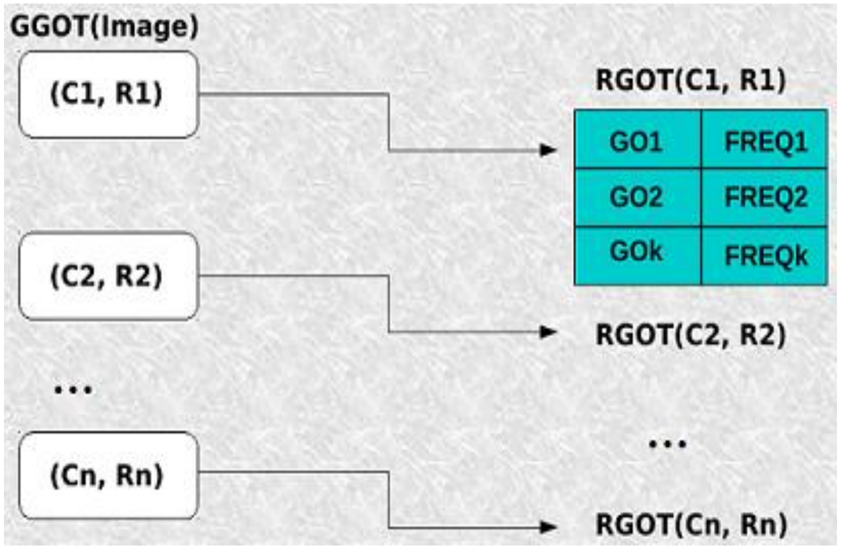

Figure 3.1. Logical structure of global gradient orientation table.

Each RGOT represents the region whose top left coordinates are specified by the corresponding *(c, r)* 2-tuple and whose size is the size of *M*. Each RGOT is subsequently converted into a single real number called the most frequent gradient orientation. This number, denoted by *DOG(M)*, is the region's dominant orientation of gradients, also known as its DOG. Consider an example of a barcode skewed in the yaw plane in Figure 3.2. Figure 3.3 gives the DOGs for a 20 x 20 mask convolved with the image in Figure 3. Each green square is a 20 x 20 image region. The top number in each square is the region's DOG, in degrees, whereas the bottom number is the frequency of that particular DOG in the region, i.e., how many pixels in that region has this gradient value. If no gradient orientation clears a given frequency count threshold, both numbers are set to 0. Figure 3.4 displays the DOGs for the 50 x 50 mask convolved with the image in Figure 3.2. It should be noted that Figures 3.3 and 3.4 show a typical tendency that, as

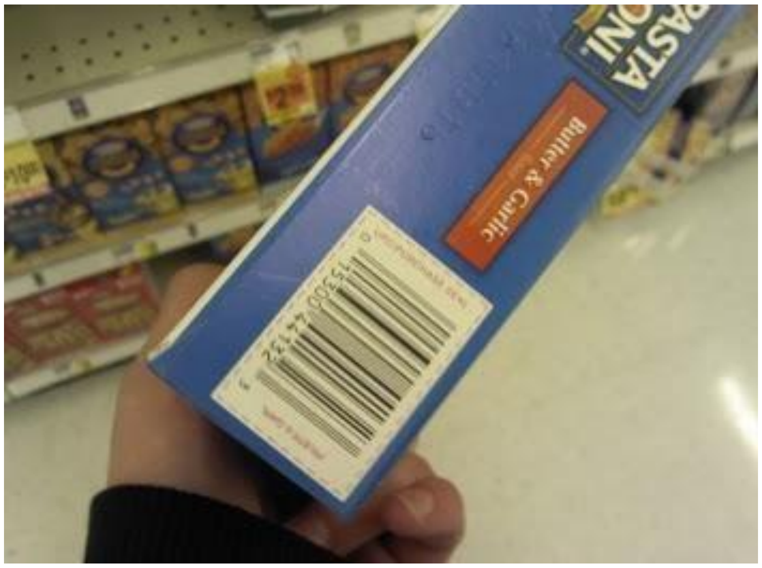

Figure 3.2. UPC-A barcode skewed in the yaw plane.

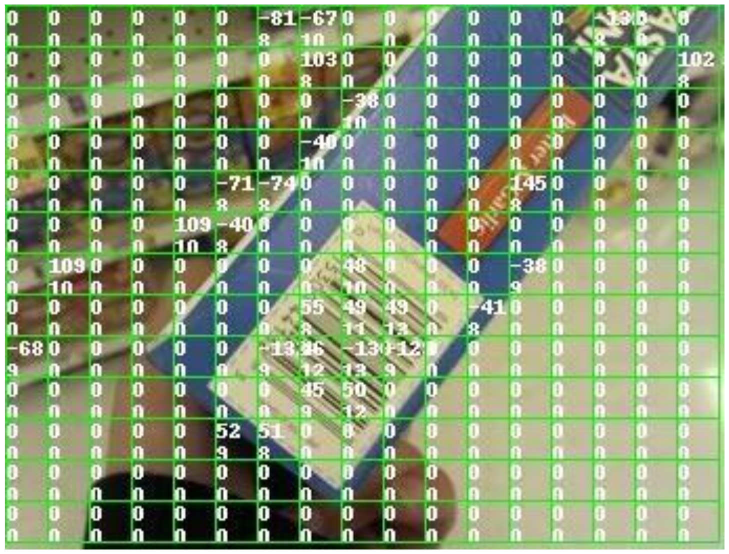

Figure 3.3 GGOT for a 20 x 20 mask.

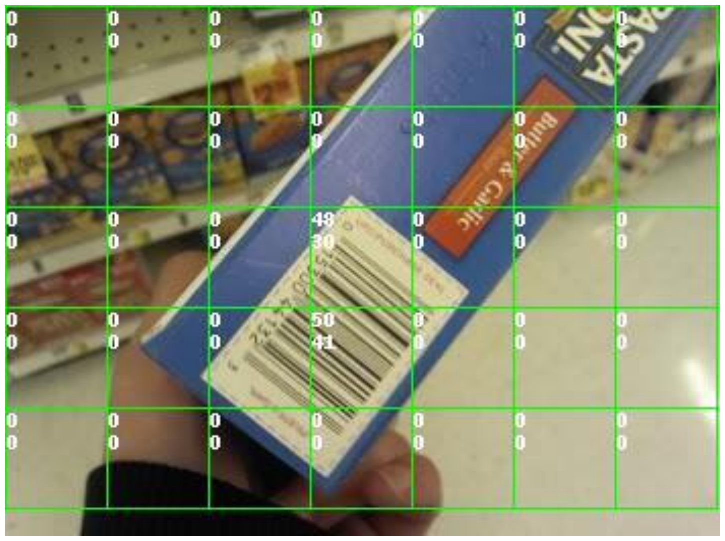

Figure 3.4 GGOT for a 50 x 50 mask.

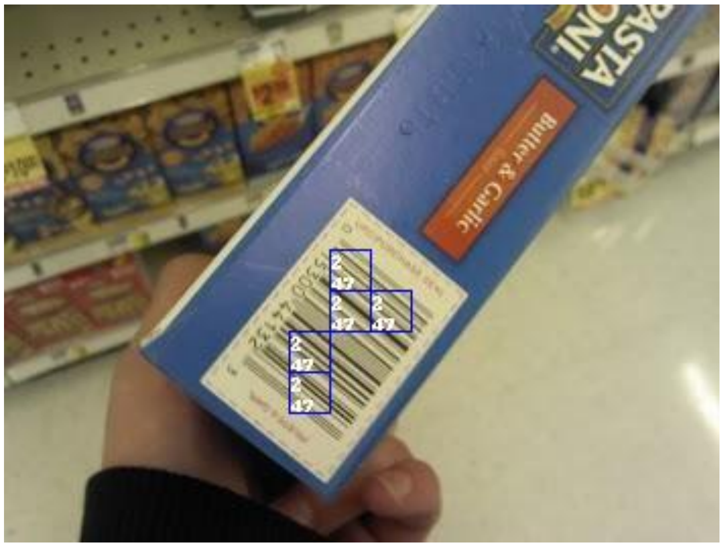

Figure 3.5 D-neighborhood found in GGOT in Figure 3.3

the size of the mask increases, fewer image regions are expected to clear the DOG threshold if the latter is set as a ratio of pixels with specific gradient values over the total number of the region’s pixels in the image.

3.3 D-Neighborhoods

Let an RGOT 3-tuple (c_k, r_k, DOG_k) consist of the coordinates of the top left corner, (c_k, r_k), of the sub image covered by an *n* x *n* mask *M* whose dominant gradient orientation is DOG_k. We define DOG-neighborhood (D-neighborhood) is a non-empty set of RGOT 3-tuples (c_k, r_k, DOG_k) such that for any such 3-tuple (c_k, r_k, DOG_k) there exists at least one other 3-tuple (c_j, r_j, DOG_j) such that $(c_j, r_j, DOG_j) \neq (c_k, r_k, DOG_k)$ and $sim((c_j, r_j, DOG_j),(c_k, r_k, DOG_k)) = True,$ where *sim* is a Boolean similarity metric. Such similarity metrics define various morphological criteria for D-neighborhoods. In our implementation, the similarity metric returns true when the square regions specified by the top left coordinates (i.e., (c_k, r_k) and (c_j, r_j)) and the mask size *n* are horizontal, vertical, or diagonal neighbors and the absolute difference of their DOGs does not exceed a small threshold. An image may have several D-neighborhoods. The D-neighborhoods are computed simultaneously with the computation of the image's GGOT. As each RGOT 3-tuple becomes available during the computation of RGOTs, it is placed into another hash table for D-neighborhoods. The computed D-neighborhoods are filtered by the ratio of the total area of their component RGOTs to the image area. For example, Figure 3.5 shows RGOTs marked as blue rectangles that are grouped into a D-neighborhood by the similarity metric defined above, because they are horizontal, vertical, and diagonal neighbors and the absolute difference of their DOGs does not exceed a small threshold. This resultant D-neighborhood is shown in Figure 3.6. This neighborhood is computed in parallel with the computation of the GGOT in Figure 3.3.

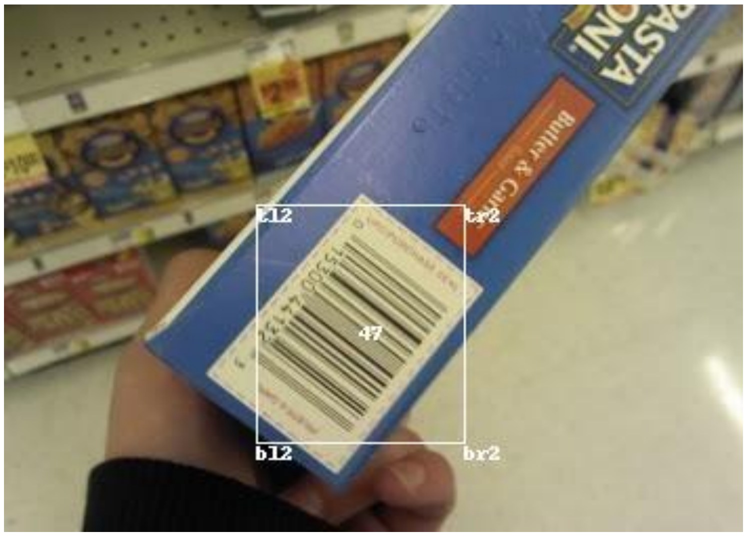

Figure 3.6. D-Neighborhood detected in Figure 3.5

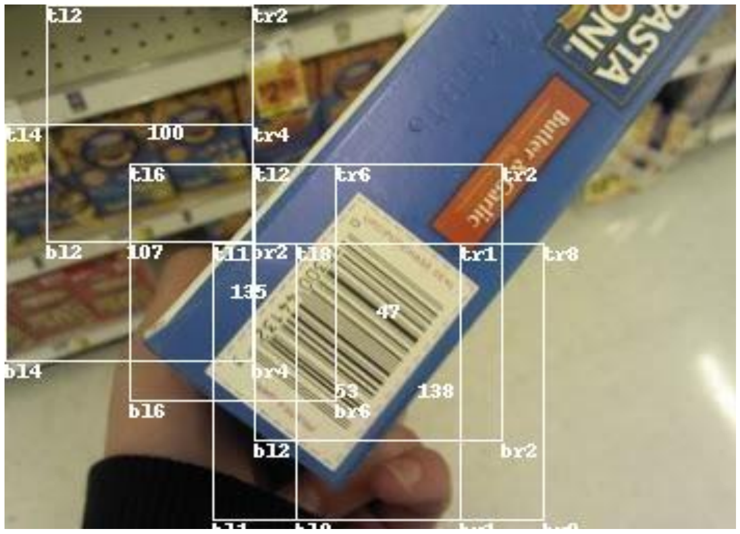

Figure 3.7. Multiple D-Neighborhoods

Detected D-neighborhoods are enclosed by minimal rectangles that contain all of their RGOT 3-tuples, as shown in Figure 3.6, where the number in the center of the white rectangle denotes the neighborhood's DOG. A minimal rectangle is the smallest rectangle that encloses all RGOTs of the same connected component. All detected D-neighborhoods are barcode region candidates. There can be multiple D-neighborhoods detected in an image. For example, Figure 3.7 shows all detected D-neighborhoods when the threshold is set to 0.01, which is too low. Figure 3.6 exemplifies an interesting and recurring fact that multiple D-neighborhoods tend to intersect over a barcode.

The DOG algorithm is given in Appendix A. Its asymptotic complexity is O(k^2), where k is the number of masks that can be placed on the image. This is because, in the worst case, each RGOT constitutes its own D-neighborhood, which makes each subsequent call to the function *FindNeighbourhoodForRGOT()*, which finds the home D-neighborhood for each newly computed RGOT, to unsuccessfully inspect all the D-neighborhoods computed so far. A similar worst-case scenario happens when there is one D-neighborhood that absorbs all computed RGOT, which takes place when the similarity metric is too permissive. Both of these scenarios, while theoretically possible, rarely occur in practice.

3.4 Linux Cluster

We built a Linux cluster out of five nodes for cloud-based computer vision and data storage. Each node is a PC with an Intel Core i5-650 3.2 GHz dual-core processor that supports 64-bit computing. The processors have 3MB of cache memory. The nodes are equipped with 6GB DDR3 SDRAM and have Intel integrated GMA 4500 Dynamic Video Memory Technology 5.0. All nodes have 320 GB of hard disk space. Ubuntu 12.04 LTS was installed on each node. We installed JDK 7 in each node. We used JBoss (http://www.jboss.org) to build and configure the cluster and the Apache mod_cluster module (http://www.jboss.org/mod_cluster) to configure the cluster for load balancing. The cluster has one master node and four slaves. The master node is the domain controller that runs mod_cluster and httpd. All nodes are part of a local area network and have hi-speed Internet connectivity. The JBoss Application Server (JBoss AS) is a free open-source Java EE-based application server. In addition to providing a full implementation of a Java application server, it also implements the Java EE part of Java. The JBoss AS is maintained by jboss.org, a community that provides free support for the server. JBoss is licensed under the GNU Lesser General Public License (LGPL). The Apache mod_cluster module is an httpd-based load balancer. The module is implemented with httpd as a set of modules for httpd with mod_proxy enabled. This module uses a communication channel to send requests from httpd to a set of designated application server nodes. An additional communication channel is established between the server nodes and httpd. The nodes use the additional channel to transmit server-side load balance factors and lifecycle events back to httpd via a custom set of HTTP methods collectively referred to as the Mod-Cluster Management Protocol (MCMP). The mod_cluster module provides dynamic configuration of httpd workers. The proxy's configuration is on the application servers. The application server sends lifecycle events to the proxies, which enables the proxies to auto-configure themselves. The mod_cluster module provides accurate load metrics, because the load balance factors are calculated by the application servers, not the proxies. All nodes in our cluster run JBoss AS 7. Jboss AS 7.1.1 is the version of the application server installed on the cluster. Apache httpd runs on the master node with the mod_cluster-1.2.0 module enabled. The Jboss AS 7.1.1 on the master and the slaves are discovered by httpd. A Java servlet for image recognition is deployed on the master node as a web archive file. The servlet's URL is hardcoded in every front end smartphone. The servlet receives images uploaded with HTTP POST requests, recognizes barcodes, and sends an HTML response back to front end smartphones. No data caching is done on the servlet or the front end smartphones.

3.5 Skewed Barcode Localization Experiments

3.5.1 Experimental Setup

The DOG algorithm was tested on images extracted from 506 video recordings of common grocery products. Each video recorded one specific product from various sides. The videos had a 1280 x 720 resolution and were recorded on an Android 4.2.2 Galaxy Nexus smartphone in a supermarket in Logan, Utah. All videos were recorded by a user who held a grocery product in one hand and a smartphone in the other. The videos covered four different categories of products: bags, bottles, boxes, and cans. The average video duration is fifteen seconds. There were 130 box videos, 127 bag videos, 125 box videos, and 124 can videos. Images were extracted from each video at the rate of 1 frame per second, which resulted in a total of 7,545 images, of which 1950 images were boxes, 1905 images were bags, 1875 images were bottles, and 1860 images were cans. These images were used in the experiments and the output of the algorithm, i.e., enclosed barcode regions (see Figures 3.6 and 3.7), were manually evaluated by two human evaluators independently.

A frame was classified as a *complete true positive* if there was a D-neighborhood with at least one straight line across all bars of a localized barcode. A frame was classified as a *partial true positive* if there was a D-neighborhood where a straight line could be drawn across some, but not all, bars of a barcode. An image was classified as a *false positive* if there was a D-neighborhood that covered an image area with no barcode and no D-neighborhood detected in the same image covered a barcode either partially or completely. For example, in Figure 3.7, the D-neighborhood, with a DOG of 100, in the upper left corner of the image, covers an area with no barcode. However, the entire frame in Figure 3.7 is classified as a complete true positive, because there is another D-neighborhood, with a DOG of 47, in the center of the frame that covers a barcode completely. A frame was classified as a *false negative* when it contained a barcode but no D-neighborhoods covered that barcode either completely or partially and no D-neighborhood covered an area with no barcode, because in the latter case, the frame was classified as a false positive. A frame was classified as a true negative when the frame contained no barcode and could not be classified as a false positive.

3.5.2 DOG Localization Experiments

The DOG algorithm was implemented in Java with OpenCV2.4 bindings for Android 4.2 and ran on Galaxy Nexus and Samsung Galaxy S2. In our previous research [10], the best threshold values for each mask size were determined. These values are given in Table 3.1. We used these values to run the experiments for each category of products. The performance analysis for the DOG algorithm is presented in the pie charts in Figures 3.8-3.11 for each category of products.

Table 3.1**:** Optimal mask sizes and threshold.

Product Type	**Mask Size**	**Threshold**
Bag	20 x 20	0.02
Bottle	40 x 40	0.02
Box	20 x 20	0.02
Can	20 x 20	0.01

As can be seen in Figures 3.8-3.11, the DOG algorithm produces very few false positives or false negatives and performs well even on unfocussed and blurry images. The large percentages of true negatives show that the algorithm is conservative. This is done by design, because it is more important, to avoid false positives. Moreover, at a rate of two frames per second, eventually there will be a frame where a barcode is successfully and quickly localized. The algorithm produces very few false negatives, which indicates,

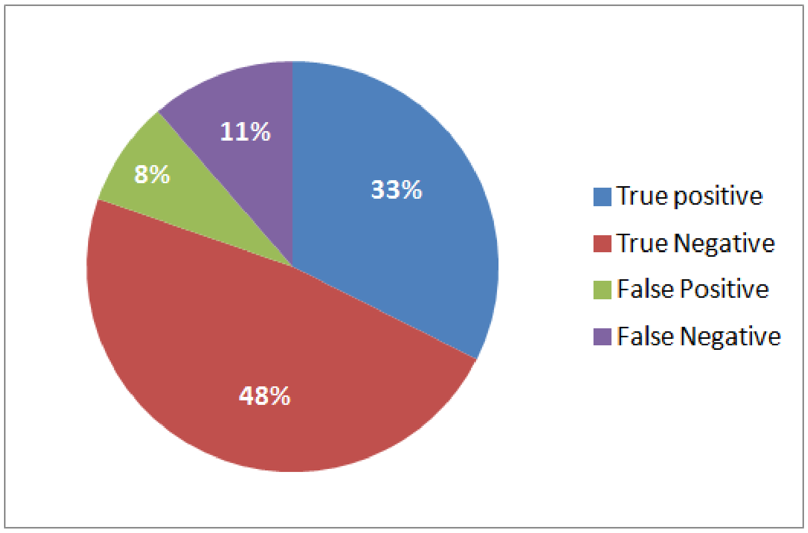

Figure 3.8 DOG performance on bags

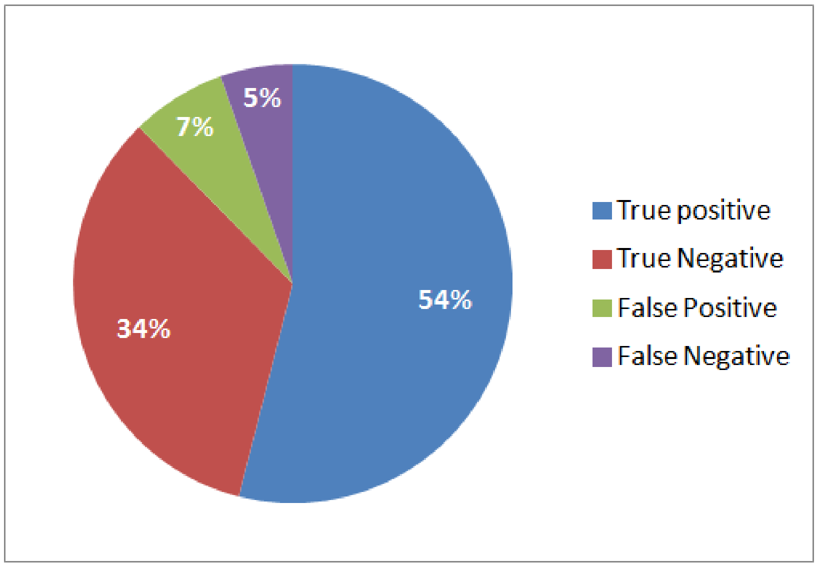

Figure 3.9 DOG performance on bottles

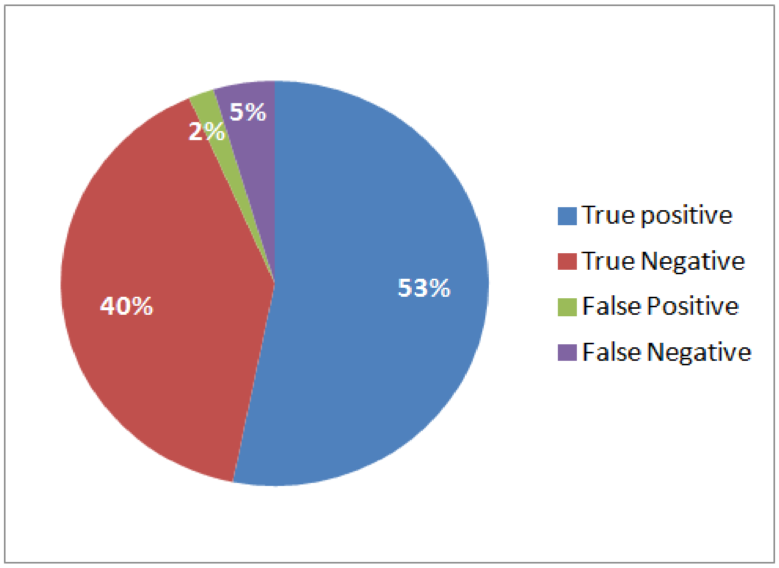

Figure 3.10 DOG performance on boxes

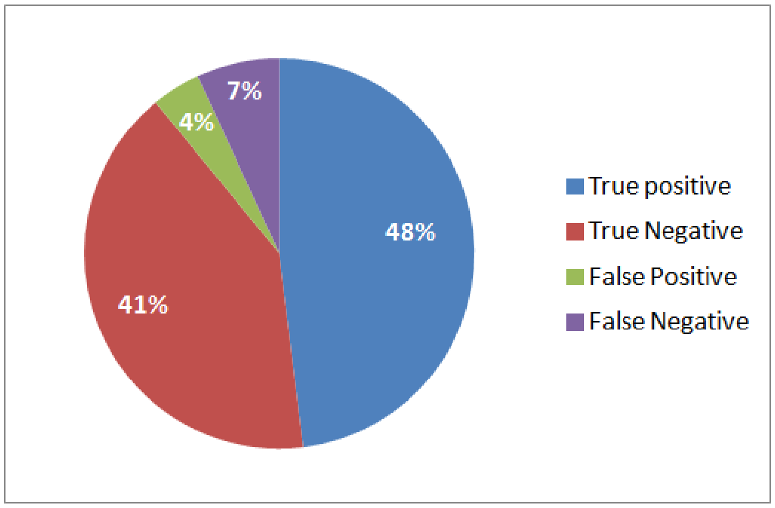

Figure 3.11 DOG performance on cans

that, if a frame contains a barcode, it will likely be localized. Figure 3.12 gives the DOG precision, recall, accuracy, and specificity values for different categories of products. The graph shows that the algorithm produced the best results for boxes, which can be attributed to the clear edges and smooth surfaces of boxes that result in images without major distortions. The largest percentages of false positives were on bags (8 percent) and bottles (7 percent). Many surfaces of these two product categories had shiny materials that produced multiple glares and reflections.

3.6 Skewed Barcode Scanning Algorithm

Our one dimensional (1D) algorithm for UPC and EAN barcode scanning works on frames with localized barcodes. In Figure 3.13, the output of the DOG localization algorithm is shown with a blue rectangle around the localized barcode. As was discussed above, the barcodes are localized in captured frames by computing dominant orientations of gradients (DOGs) of image segments and collected into larger connected components on the basis of their DOG similarity and geometrical proximity.

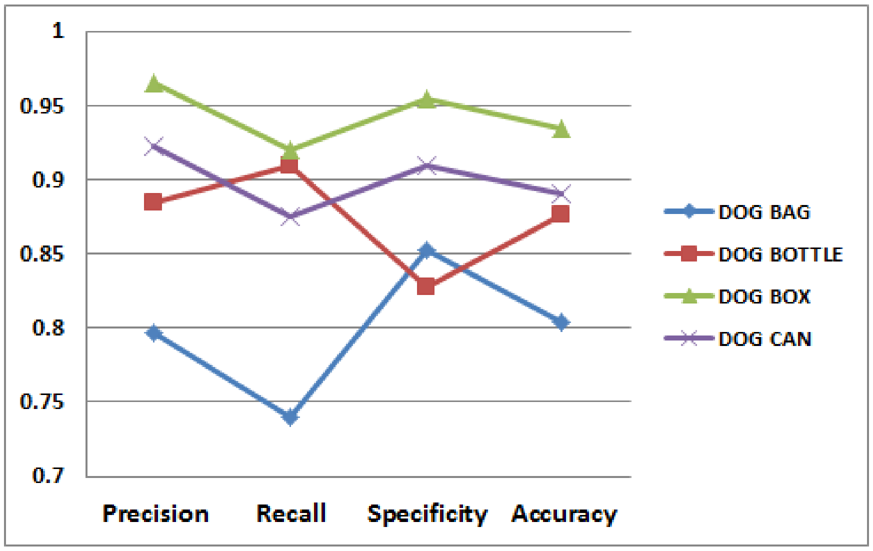

Figure 3.12 DOG precision, recall, specificity, and accuracy

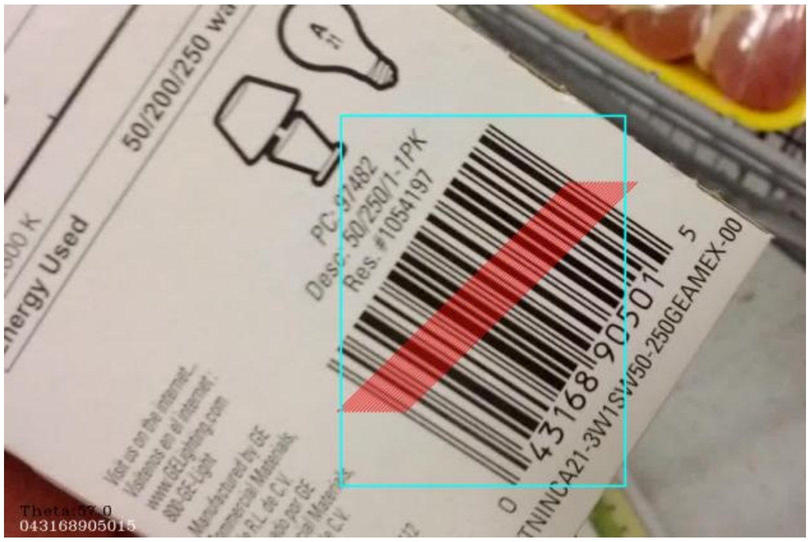

Figure 3.13 Barcode localization with DOG

Figure 3.14 shows the control flow of the 1D barcode scanning algorithm. The algorithm takes as input an image captured from the smartphone camera's video stream. This image is processed by the DOG algorithm. If the barcode is not localized, another frame is grabbed from the video stream. If the DOG algorithm localizes a barcode, as shown in Figure 3.13, the coordinates of the detected region is passed to the line grower component. The line grower component selects the center of the localized region, which is always a rectangle, and starts growing scanlines. For an example of how the line growing component works, consider Figure 3.15. The horizontal and vertical white lines intersect in the center of the localized region. The skew angle of the localized barcode, computed by the DOG algorithm, is 120 degrees. The line that passes the localized region's center at the skew angle detected by the DOG algorithm is referred to as the skew line. In Figure 3.15, the skew line is shown as a solid black line running from north-west to south-east. After the center of the region and the skew angle are determined, the line growing module begins to grow scanlines orthogonal to the skew line. A scanline is

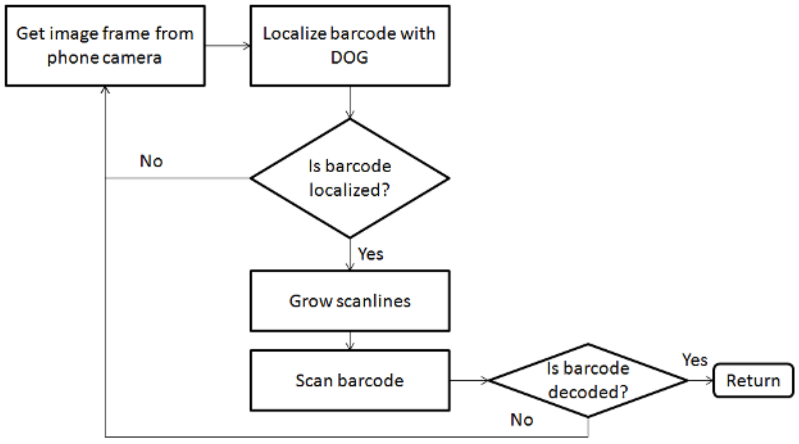

Figure 3.14 Flowchart for barcode localization

grown on both sides of the skew line. In Figure 3.15, the upper half of the scanline is shown as a red arrow and the lower half of the scanline is shown as a blue arrow. Each half-line is extended until it reaches the portion of the image where the barcode lines are no longer detectable. A five pixel buffer region is added after the scanline's end to improve subsequent scanning. The number of scanlines grown on both sides of the skew line is controlled through an input parameter. In the current implementation of the algorithm, the value of this parameter is set to 10. The scanlines are arrays of luminosity values for each pixel in their growth path. It should be noted that the scanlines are grown and scanned in place without any image or line rotation. For each grown scanline, the

Line Widths (LW) for the barcode are then computed by finding two points that are on the intensity curve but lie on the opposite sides of the mean intensity. By modeling the

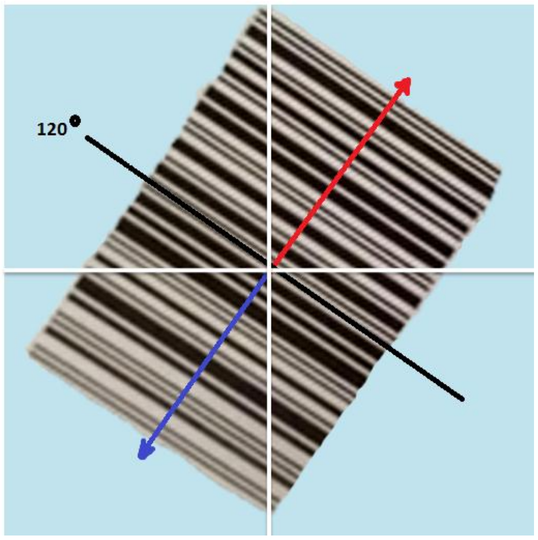

Figure 3.15 Growing a scanline orthogonal to skew angle

curve between these points as a straight line the intersection points are obtained between the intensity curve and the mean intensity.

Figure 3.16 shows a sequence of images that gives a visual demonstration of how the algorithm works on a captured frame. The top image in Figure 3.16 is a frame captured from the smartphone camera’s video stream. The second image from the top in Figure 3.16 shows the result of the clustering stage of the DOG algorithm that clusters small subimages with similar dominant gradient orientations and close geometric proximity. The third image shows a localized barcode enclosed in a white rectangle. The bottom image in Figure 3.16 shows ten scanlines, one of which results in a successful barcode scan.

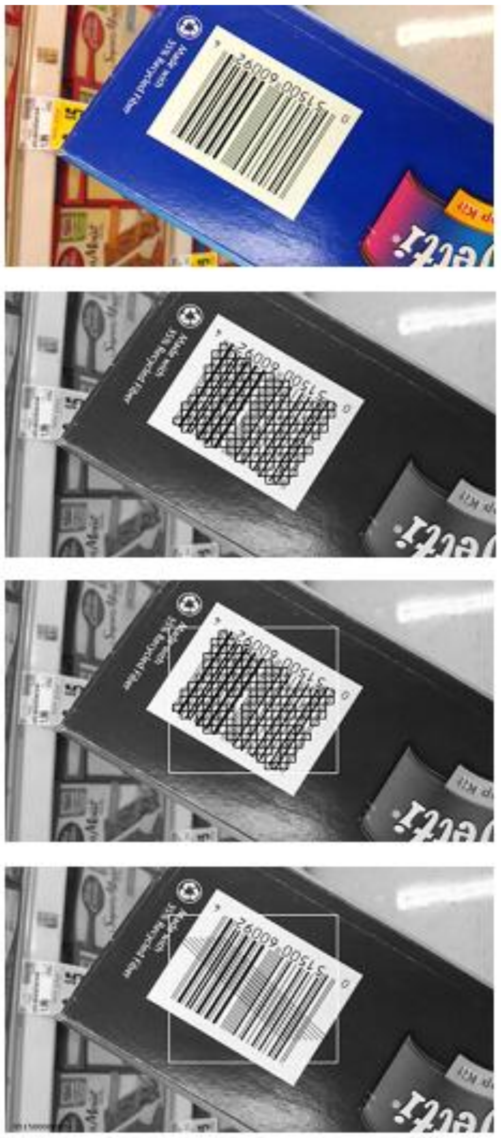

Figure 3.16 1D Barcode scanning steps

3.7 Skewed Barcode Scanning Experiments

Our 1D algorithm for UPC and EAN barcode scanning works on frames with localized barcodes. In Figure 3.13, the output of the DOG localization algorithm is shown with a blue rectangle around the localized barcode. As was discussed above, the barcodes are localized in captured frames by computing dominant orientations of gradients (DOGs) of image segments and collected into larger connected components on the basis of their DOG similarity and geometrical proximity.

3.7.1 Experiments in the supermarket

We conducted our first set of barcode scanning experiments in a local supermarket to assess the feasibility of our system. A user, who was not part of this research project, was given a Galaxy Nexus 4 smartphone with an AT&T 4G connection. Our front end application was installed on the smartphone. The user was asked to scan ten products of his choice in each of the four categories: box, can, bottle, and bag. The user was told that he can choose any products to scan so long as each product was in one of the above four categories. A research assistant accompanied the user and recorded the scan times for each product. Each scan time started from the moment the user began scanning and ended when the response was received from the server. Figure 3.17 denotes the average scan times in seconds for each category. The cans showed the longest scanning average due to glares and reflections.

Bags showed the second longest scanning average due to some crumpled barcodes. As we discovered during these experiments, another cause for the slower scan times on individual products in each product category is the availability of Internet connectivity at various locations in the supermarket. During the experiments in the supermarket, we noticed that at some areas of the supermarket the Internet connection did not exist, which caused delays in barcode scanning. For several products, a 10 or 15-step change in location within a supermarket resulted in a successful barcode scan.

3.7.2 Impact of blurred images

The second set of barcode scanning experiments was conducted to estimate the impact of blurriness on skewed barcode localization and scanning. These experiments were conducted on the same set of 506 videos of boxes, bags, bottles, and cans that we used for our barcode localization experiments. The average video duration is fifteen seconds. There are 130 box videos, 127 bag videos, 125 box videos, and 124 can videos. Images were extracted from the videos at the rate of 1 frame per second, which resulted in a total of 7,545 images, of which 1950 images were boxes, 1905 images were bags, 1875 images were bottles, and 1860 images were cans.

Each frame was automatically classified as blurred or sharp by our blur detection algorithm using Haar wavelet transforms [37, 38]. Each frame was also manually classified as having a barcode or not and labeled with the type of grocery product: bag, bottle, box, can. There were a total of sixteen categories. Figure 3.18 shows the results of the experiments. Images that contained barcodes for all four product categories had no false positives. In each product category, the sharp images had a significantly better true positive percentage than the blurred images. A comparison of the bar charts in Figure

3.18 reveals that the true positive percentage of the sharp images is more than double that of the blurry ones.

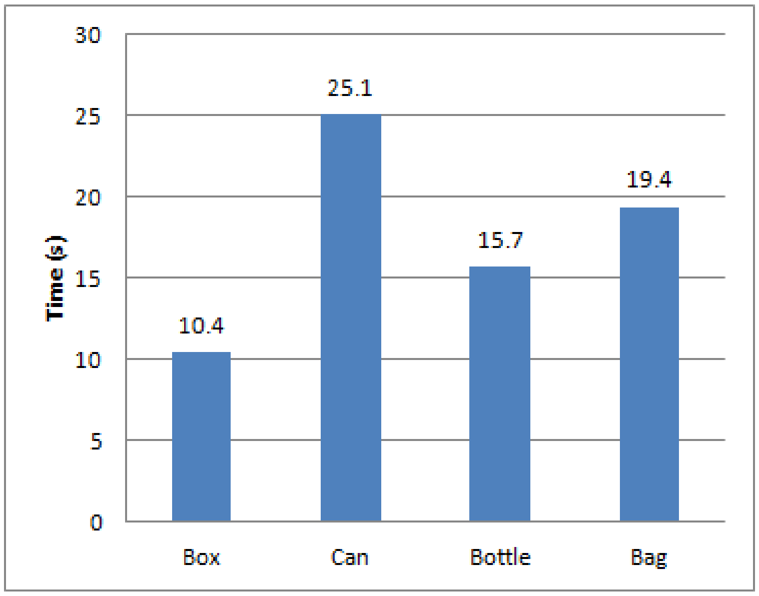

Figure 3.17 Average barcode scan times

Images without any barcode for all categories produced 100% accurate results with all true negatives, irrespective of the blurriness. In other words, the algorithm is highly specific in that it does not detect barcodes in images that do not contain them. Another observation on Figure 3.18 is that the algorithm showed its best performance on boxes. The algorithm’s performance on bags, bottles, and cans was worse because of some crumpled, curved, or shiny surfaces. These surfaces caused many light reflections, which hindered performance of both barcode localization and barcode scanning. The percentages of the skewed barcode localization and scanning were better on boxes due to smoother surfaces. Quite expectedly, the sharpness of images made a positive difference in that the scanning algorithm performed much better on sharp images in each product category. Specifically, on sharp images, the algorithm performed best on boxes with a detection rate of 54.41%, followed by bags at 44%, cans at 42.55%, and bottles at 32.22%.

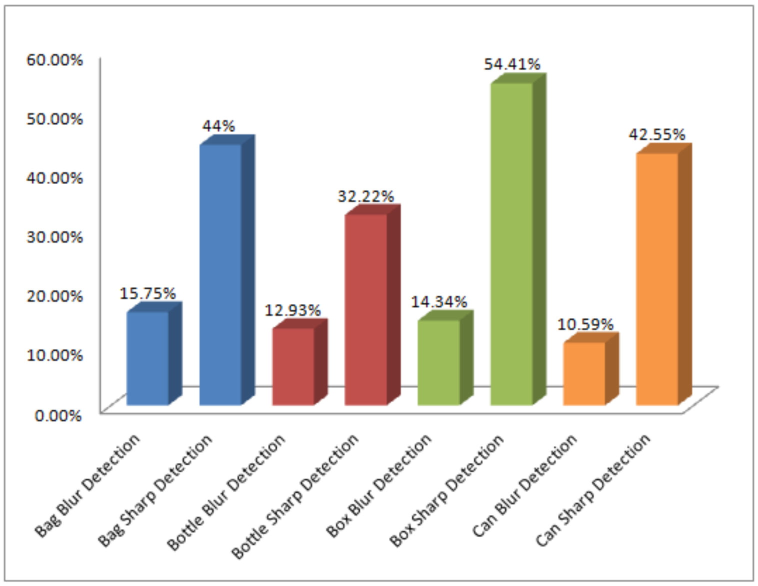

Figure 3.18 Barcode detection rates on blurred and non-blurred images

3.8 Robustness and Speed of Linux Cluster

The third set of experiments was conducted to assess the robustness and speed of our Linux cluster for image processing. After all classifications were completed (blurred vs. sharp; barcode vs. no barcode; type of grocery product), the classified frames were stored in the smartphone's memory card. An Android service was implemented and installed on a Galaxy Nexus 4 smartphone. The service took one frame at a time and sent it to the node cluster via an http POST request over a local Wi-Fi network with a download speed of 72.31 Mbps and an upload speed of 29.64 Mbps.

The service recorded the start time before uploading each image and the finish time once a response was received from the cluster. The difference between the finish and start times was logged as a total request-response time. The service was run with one image from each of the sixteen categories described in Section 3.5.2 for 3000 times, and the average request-response time for each session was calculated.

Each image sent by the service was processed on the cluster as follows. The DOG localization algorithm was executed and, if a barcode was successfully localized, the barcode was scanned in place within the localized region with ten scanlines. The detection result was sent back to the smartphone and recorded on the smartphone's memory card. Figure 3.19 gives the graph of the node cluster's request-response times.

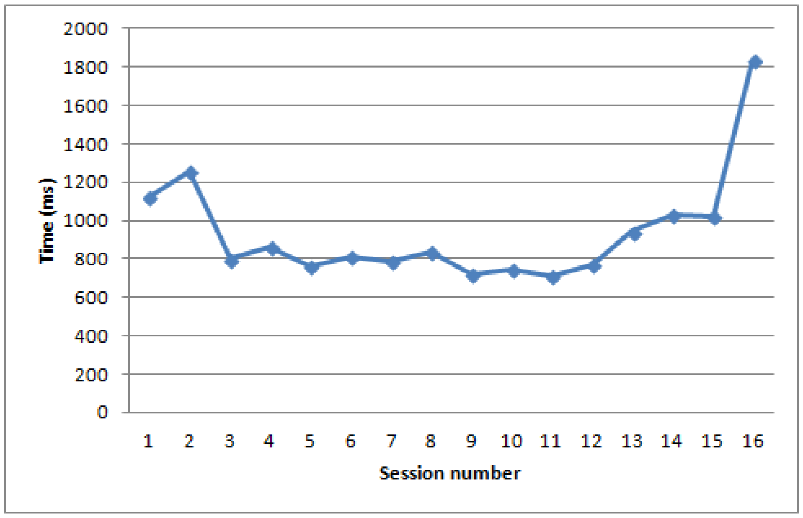

Figure 3.19 Average request - response times

The lowest average request-response time was 712 milliseconds; the highest average was 1813 milliseconds.

3.9 Discussion

An algorithm was presented for vision-based localization of 1D UPC and EAN barcodes with relaxed roll, pitch, and yaw camera alignment constraints. The algorithm (DOG) localizes barcodes in images by computing dominant orientations of gradients of image segments and grouping smaller segments with similar dominant gradient orientations into larger connected components. Connected components that pass specific morphological criteria are marked as potential barcodes and enclosed with minimal rectangular areas. The algorithm was implemented in a distributed, cloud-based system. The system's front end is a smartphone application that runs on Android smartphones with Android 4.2 or higher. The system's back end was deployed on a five node Linux cluster where images are processed. The algorithm was evaluated on a sample of 506 videos of bags, boxes, bottles, and cans in a supermarket. All videos were recorded with an Android 4.2 Google Galaxy Nexus smartphone. The videos have been made public for all interested research communities to replicate our findings or to use them in their own research [39]. The front end Android application is available for free download at Google Play under the title of NutriGlass [40].

The DOG algorithm is designed to be conservative in that it rejects the frames on the slightest chance that it does not contain any barcode. While this increases false negatives, it keeps both true negatives and false positives close to zero. The DOG algorithm was subsequently coupled to our 1D UPC and EAN barcode scanner. The scanner receives a localized barcode region from the DOG algorithm along with the region's skew angles and uses a maximum of ten scanlines drawn at the skew angle to scan the barcode in place without any rotation of the scanlines or the localized barcode region.

After the DOG algorithm was coupled to our 1D barcode scanner, three sets of barcode scanning experiments were conducted with the system. The first set of barcode scanning experiments was conducted in a local supermarket to assess the feasibility of our system by a user with a Galaxy Nexus 4 smartphone with an AT&T 4G connection. The user was asked to scan ten products of his choice in each of the four categories: box, can, bottle, and bag. The cans showed the longest scanning average due to glares and reflections. Bags showed the second longest scanning average due to some crumpled barcodes. Another cause for the slower scan times on individual products was the availability of Internet connectivity at various locations in the supermarket. At some areas of the supermarket the Internet connection did not exist, which caused delays in barcode scanning.

The second set of barcode scanning experiments was conducted to estimate the impact of blurriness on skewed barcode localization and scanning. These experiments were conducted on the same set of 506 videos of boxes, bags, bottles, and cans. Images were extracted from the videos at the rate of 1 frame per second, which resulted in 1950 box images, 1905 bag images, 1875 bottle images, and 1860 can images. Images for all four product categories had no false positives. In each product category, the sharp images had a significantly better true positive percentage than the blurred images. The true positive percentage of the sharp images was more than double that of the blurry ones. Images without any barcode for all categories produced 100% accurate results with all true negatives, irrespective of the blurriness. The sharpness of images made a positive difference in that the scanning algorithm performed much better on sharp images in each product category.

The third set of experiments was conducted to assess the robustness and speed of our five Linux cluster for image processing. An Android service was implemented and installed on a Galaxy Nexus 4 smartphone. The service took one frame at a time and sent it to the node cluster via an http POST request over a local Wi-Fi network with a download speed of 72.31 Mbps and an upload speed of 29.64 Mbps. Sixteen sessions were conducted during each of which 3,000 images were sent to the cluster. The cluster did not experience any failures. The lowest average request-response time was 712 milliseconds; the highest average was 1813 milliseconds.

One limitation of the current front end implementation is that it does not compute the blurriness of the captured frame before sending it to the back end node cluster where barcode localization and scanning are performed. As the experiments described in Section 3.5.2 indicate, the scanning results are substantially higher on sharp images than on blurred images. This limitation points to a potential improvement that we plan to implement in the future. When a frame is captured, its blurriness coefficient can be computed on the smartphone and, if it is high, the frame should not even be sent to the cluster. This improvement will reduce the load on the cluster and increase its responsiveness.

Another approach to handling blurred inputs is to improve camera focus and stability, both of which are outside the scope of our research agenda, because it is, technically speaking, a hardware problem. It is likely to work better in later models of smartphones. The current implementation on the Android 4.2 platform attempts to force the camera focus at the image center through the existing API. Over time, as device cameras improve and more devices run newer versions of Android, this limitation will likely have a smaller impact on the system’s performance.

CHAPTER 4

TEXT SKEW ANGLE DETECTION

4.1 Introduction

Vision-based extraction of nutritional information from nutrition labels (NLs) available on most food product packages is critical to proactive nutrition management, because it improves the user's ability to engage in continuous nutritional data collection and analysis. Computer vision can play a key role in the food selection process by providing consumers with real time text analysis of NLs, which will likely engage consumers in proactive nutrition management [9]. We have previously developed a vision-based localization algorithm for horizontally or vertically aligned NLs on smartphones [9]. The algorithm was subsequently modified to process not only aligned NLs but also slightly skewed ones. A limitation of the algorithm was its inability to handle arbitrary text skews [13]. In this chapter, we address this limitation by proposing an algorithm for text skew detection without any constraints on the magnitude of the skew. The proposed algorithm works not only on NLs but also on single- and multi-column printed text images. The algorithm is called TSAW (Text Skew Angle Wavelets) and is implemented in Java. To ensure the reproducibility and veracity of the results reported in this article, we have made our source code publicly available [42]. TSAW takes printed text images and downsamples them with several iterations of the 2D Haar Wavelet Transform (2D HWT). The HL, LH, and HH matrices are used to identify 2D points with significant intensity changes. These points form a *singularity point set* in the 2D plane. The convex hull algorithm [14] is applied to this set to enclose it with a minimum area rectangle. The text's skew is the enclosing rectangle's rotation angle relative to the absolute north.

4.2 Haar Wavelet Transform

In TSAW, images are down-sampled with several iterations of 2D Haar Wavelet Transform (HWT) to obtain the HL, LH, and HH matrices. The HWT is a discrete wavelet transform (DWT) applicable to $l^2(Z)$ signals. The recurrences for forward 1D HWT implemented in TSAW are given in (4.1) and are formally developed in [43].

$$\begin{aligned} d_{j-1}^{(1)}[n] &= s_j[2n+1] - s_j[2n] \\ s_{j-1}^{(1)}[n] &= s_j[2n] + \frac{1}{2} d_{j-1}^{(1)}[n] \\ s_{j-1}[n] &= \sqrt{2}\, s_{j-1}^{(1)}[n] \\ d_{j-1}[n] &= d_{j-1}^{(1)}[n]/\sqrt{2} \end{aligned} \tag{4.1}$$

The s and d values are the values of the low and high pass filters, respectively, recursively computed from the previous scale. Unlike more sophisticated DWTs, e.g., Daubechies D4 [44, 45], HWT does not have the boundary problem when the computation of the low and high pass filter values at the current scale requires samples and wavelets outside of the boundaries of $l^2(Z)$ signals.

In TSAW, the generalization of 1D HWT to 2D is based on the tensor products of basic wavelets in the first dimension with basic wavelets in the second dimension, as given in (4.2). The formal treatment of this generalization is developed in [43].

$$\varphi_{[0,1[}(r) = \begin{cases} 1 \ if \ 0 \leq r < 1, \\ 0 \ \ otherwise. \end{cases}$$

$$\psi_{[0,1[}(r) = \begin{cases} 1 \ if \ 0 \leq r < \frac{1}{2}, \\ -1 \ if \frac{1}{2} \leq r < 1, \\ 0 \quad otherwise. \end{cases} \tag{4.2}$$

$$\begin{aligned} \Phi_{0,0}^{(0)}(x,y) &= \left(\varphi_{[0,1[} \times \varphi_{[0,1[}\right)(x,y) \\ \Psi_{0,0}^{h,(0)}(x,y) &= \left(\varphi_{[0,1[} \times \psi_{[0,1[}\right)(x,y) \\ \Psi_{0,0}^{v,(0)}(x,y) &= \left(\psi_{[0,1[} \times \varphi_{[0,1[}\right)(x,y) \\ \Psi_{0,0}^{d,(0)}(x,y) &= \left(\psi_{[0,1[} \times \psi_{[0,1[}\right)(x,y) \end{aligned} \tag{4.3}$$

Given two functions, f_1 and f_2, of one argument, their tensor product is $(f_1 \times f_2)(x,y) = f_1(x) \cdot f_2(x)$. The 2D wavelets for 2D HWT are tensor products of $\varphi_{[0,1[}(r)$ and $\psi_{[0,1[}(r)$ defined in (4.3), where superscripts *h*, *v*, and *d* denote the horizontal, vertical, and diagonal wavelets, respectively. In 2D $l^2(Z)$ signals, e.g., images, the horizontal wavelets reflect horizontal (left to right) changes, the vertical wavelets reflect vertical (top to bottom) changes, and the diagonal changes reflect the changes between the two main diagonals.

$$\begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} 11 & 9 \\ 7 & 5 \end{bmatrix} \qquad (4.4)$$

In practice, 2D HWT is computed by applying 1D HWT to each row and then to each column. As an example, suppose there is a 2 × 2 pixel image, defined in (4.4), where $s_{r,c}$ denotes a pixel in row r and column c. Applying 1D HWT to each row results in the 2 × 2 matrix in (4.5). 1D HWT is then applied to each column of the matrix in (4.5), which results in the matrix in (4.6) whose coefficients encode the data in the original matrix in (4.4) in terms of the four tensor wavelets $\Phi_{0,0}^{(0)}(x,y)$, $\Psi_{0,0}^{h,(0)}(x,y)$, $\Psi_{0,0}^{v,(0)}(x,y)$, and $\Psi_{0,0}^{d,(0)}(x,y)$ in (4.7). This decomposition operation can be represented in terms of matrices in (4.8).

$$\begin{bmatrix} \frac{11+9}{2} & \frac{11-9}{2} \\ \frac{7+5}{2} & \frac{7-5}{2} \end{bmatrix} = \begin{bmatrix} 10 & 1 \\ 6 & 1 \end{bmatrix} \qquad (4.5)$$

$$\begin{bmatrix} \frac{10+6}{2} & \frac{1+1}{2} \\ \frac{10-6}{2} & \frac{1-1}{2} \end{bmatrix} = \begin{bmatrix} 8 & 1 \\ 2 & 0 \end{bmatrix} \tag{4.6}$$

$$\begin{bmatrix} 11 & 9 \\ 7 & 5 \end{bmatrix} = 8\Phi_{0,0}^{(0)}(x,y) + 1\Psi_{0,0}^{h,(0)}(x,y) + 2\Psi_{0,0}^{v,(0)}(x,y) + 0\Psi_{0,0}^{d,(0)}(x,y) \tag{4.7}$$

$$\begin{bmatrix} 11 & 9 \\ 7 & 5 \end{bmatrix} = 8\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + 1\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix} + 2\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} + 0\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{4.8}$$

The value 8 in the upper-left corner of the matrix in (4.6) is the average value of the original matrix in (4.4): (11+9+7+5)/4= 8. The value 1 in the upper right-hand corner of (4.6) is the horizontal change in the data in (4.4) from the left average, (11+7)/2=9, to the right average, (9+5)/2=7, which is equal to $1 \cdot \Psi_{0,0}^{h,(0)}(x,y) = 1 \cdot -2$. The value 2 in the bottom-left corner in (4.6) is the vertical change in the original data in (4.4) from the upper average, (11+9)/2=10, to the lower average, (7+5)/2=6, which is equal to $2 \cdot$

$\Psi^{v,(0)}_{0,0}(x,y) = 2 \cdot -2$=−4. The value 0 in the bottom-right corner of (4.6) is the change in the original data in (4.4) from the average along the first diagonal (from the top left corner to the bottom right corner), (11+5)/2=8, to the average along the second diagonal (from the top right corner to the bottom left corner), (9+7)/2=8, which is equal to $0 \cdot \Psi^{d,(0)}_{0,0}(x,y)$.

4.3 Text Skew Detection

TSAW was originally designed to work on NL images taken with a smartphone camera. TSAW receives as input images with NL texts, as shown in Figure 4.1 (left). In the current publicly available implementation [42], the default input image size is $2^{10} \times 2^{10}$. 2D HWT is applied to the input image for two iterations to compute the horizontal, vertical, and diagonal changes and store them in the three matrices: HC (horizontal change), VC (vertical change), and DC (diagonal change), as shown in Figure

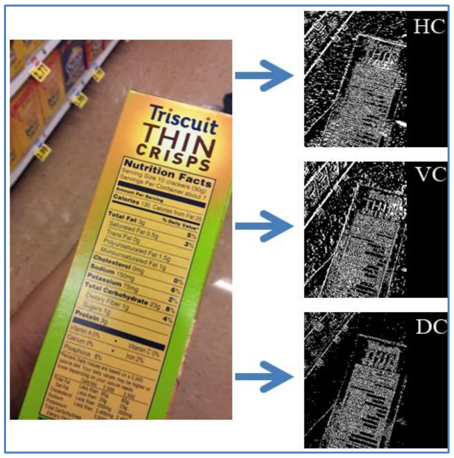

Figure 4.1 Horizontal, vertical, and diagonal changes

4.1 (right). Since 2D HWT is applied twice, the three change matrices are $2^8 \times 2^8$. Thus, the original image is downsampled from 1024×1024 to 256×256. The number of iterations is one of the input parameters in the algorithm and can be easily adjusted if necessary.

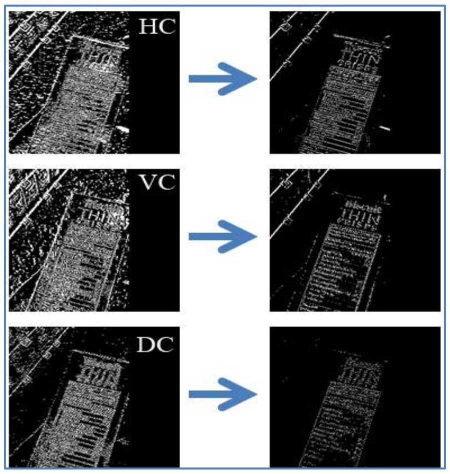

Figure 4.2 Binarization of HC, VC, and DC matrices

$$S[i,j] = \begin{cases} 255 & \text{if } \alpha HC[i,j] + \beta VC[i,j] + \gamma DC[i,j] \geq \theta \\ 0 & \text{if } \alpha HC[i,j] + \beta VC[i,j] + \gamma DC[i,j] < \theta \end{cases} \tag{4.9}$$

Each change matrix is binarized to set each pixel to $v_1 = 0$ or $v_2 = 255$, as shown in Figure 4.2. The binarized matrices are combined into a 256×256 matrix $S[i,j]$ defined in (4.9), where $\alpha + \beta + \gamma = 1$ and $\theta \in R$ is a threshold. The parameters α, β, and γ control the relative contributions of the horizontal, vertical, and diagonal changes, respectively. A sample $S[i,j]$ matrix is shown in Figure 4.3 (right) with

$\theta = 255$. The pixels of $S[i,j]$ whose value is 255 indicate 2D points with significant intensity changes. The DC wavelets were observed to detect the presence of text better than the HC or VC wavelets. This may be due to the fact that printed text has more diagonal edges than horizontal or vertical ones as compared to other objects in the image such as lines or graphics. Consequently, in the current implementation of TSAW the following parameter values are used: $\alpha=\beta=0.2$ and $\gamma=0.6$. The convex hull algorithm [14] is used to find a minimum area rectangle bounding the singularity point set defined by $S[i,j]$, as shown in Figure 4.4 (right). The text skew is computed as the rotation angle of this rectangle relative to the true north of 90 degrees.

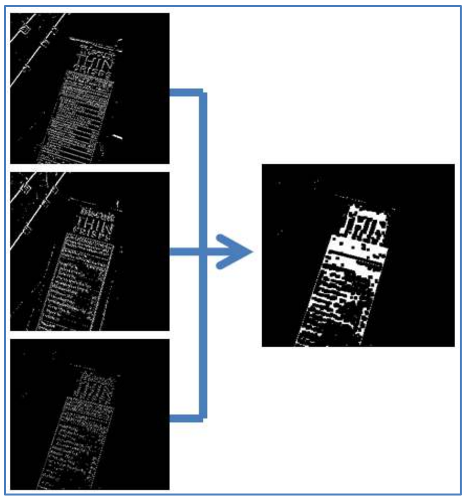

Figure 4.3 Combining wavelet matrices into result matrix

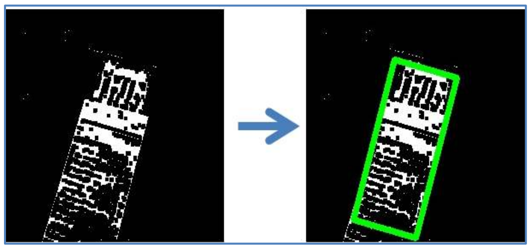

Figure 4.4 Using the minimum area rectangle for text skew angle computation

Appendix B gives the TSAW pseudocode. The algorithm takes as input a 2D image *Img* of size $N \times N$, where $N = 2^i, i > 1$. If the size of the image is not equal to an integral power of 2, as required by 2D HWT, the image is padded with 0's. The third argument, *NITER*, specifies the number of iterations for 2D HWT. On line 2, 2D HWT is applied to *Img* for *NITER* iterations, which in the current implementation is equal to 2. The procedure 2DHWT returns an array of four n x n matrices $AVRG$, *HC*, *VC*, and *DC*. The first matrix contains the averages while *HC*, *VC*, and *DC* record horizontal, vertical and diagonal wavelet coefficients.

On line 4, the matrices *HC*, *VC*, and *DC* are binarized in place. Lines 5-14 give the pseudocode for the **Binarize** procedure. Figure 4.2 shows an example of how this procedure works. On line 5, the procedure **FindSkew** is called. The pseudocode for this procedure is shown in lines 15-26. **FindSkew** takes three $n \times n$ matrices *HC*, *VC*, and *DC* and the α, β, γ parameter used in computing $S[i, j]$.

On line 16, the matrix $S[i, j]$ is initialized. On lines 17-24, the $S[r, c]$ values are computed from the $HC[r, c]$, $VC[r, c]$, and $DC[r, c]$ values, as defined in (4.9). On line 26, the algorithm first calls the procedure **FindMinAreaRectangle** that uses the convex hull algorithm to find a minimal area rectangle around the points with significant intensity changes, i.e., $S[r,c]$ = 255. and then calls the procedure **FindRotationAngle** that returns the value of the text skew as the rotation angle of this rectangle relative to the truth north of 90 degrees.

4.4 Text Skew Angle Detection Experiments

The performance of TSAW was compared with the algorithms by Postl [22] (Algo1), by Hull [24] (Algo2), by Li, Shen, and Sun [29] (Algo3), and by Papandreou and Gatos [27] (Algo4). Since we were unable to obtain the source code of these algorithms in a performance sensitive imperative programming language (C/C++ or Java), we implemented them in Java with JDK 1.7, the same JDK version we used to implement TSAW. Our implementations of these algorithms are publicly available [46].

The first experiment was designed to evaluate text skew detection in the context of vision-based nutrition information extraction where the ultimate objective is to extract NL information in real time from NL images taken with smartphones. Toward that end, 1001 NL frames of common grocery products were extracted, at a rate of 1 frame per second, from 1280 x 720 HD videos of common grocery packages with an average duration of 15 seconds. The videos were recorded on an Android 4.3 Galaxy Nexus smartphone in two supermarkets in Logan, UT. The videos include four different categories of products: bags, boxes, bottles, and cans. Our image blur detection algorithm [37,38] was used to remove all blurred images from the frames extracted from the videos. Thus, 1001 NL images of common grocery products are the images classified as sharp by our blur detection algorithm. This data set is henceforth referred to as DS1.

The text skew ground truth was obtained from two human volunteers who used an open source protractor [47] to estimate the text skew manually for each image in DS1. To facilitate the replication of our results, we have made DS1 and the ground truth estimates publicly available [48]. All five algorithms were executed on DS1. The computed text skews were recorded for each image. The text skews were compared with the ground truth via box plots.

In the second experiment, all the algorithms were applied to a public data set (henceforth referred to as DS2) of 2200 scanned document images [49]. DS2 includes figures, tables, diagrams, block diagrams, architectural plans, electrical circuits in multiple languages from newspapers, journals, books, dictionaries, etc. The images are rotated from -5 to +5 degrees with a step of 1. The text skews were logged for each algorithm and image. The text skews were compared with the ground truth via box plots. In box plots, better methods have narrower boxes centered at 0. Wider boxes indicate greater variability between the estimated and actual values. Median lines far away from 0 suggest method biases.

The performance of TSAW was also compared with a more recent algorithm by Papandreou et al. [28]. The researchers evaluated their algorithm on DS2 by computing the average error deviation (AED) and the percentage of correct estimations (CEs) given in (4.10) and (4.11), where N is the number of images in the dataset and $E(j)$ is the error in skew angle estimation for the j-th image. The AED of 0 and percentage CE of 100 are ideal. The AED and CE values were computed for all algorithms on DS1 and DS2.

$$\text{AED} = \frac{\sum_{j=1}^{N} E(j)}{N} \qquad (4.10)$$

$$\text{CE}=\frac{\sum_{j=1}^{N} K(j)}{N}*100 \text{ , where } K(\text{j}) = \begin{cases} 1 \; if \; E(j) = 0 \\ 0 \; otherwise \end{cases} \qquad (4.11)$$

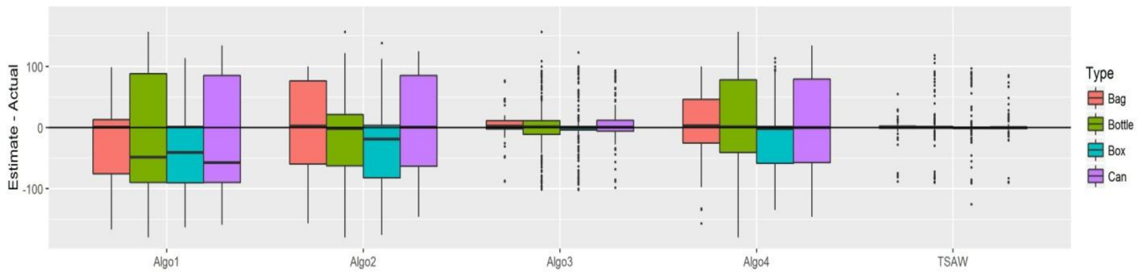

Figure 4.5 Box plots on DS1

4.5 Results

In Figure 4.5, the box plots are given for each algorithm on DS1.The vertical axis denotes the difference between the text skews computed by each algorithm and product type. The box plots indicate that TSAW has the narrowest boxes and median errors close to 0 in all image categories, which suggests that this algorithm is less error prone and more consistent than the other four algorithms on DS1. Algo3 is a close second with the median errors close to 0, however the boxes are wider than TSAW's. Algo1 has a negative bias for cans, boxes and bottles. Algo2 also has a negative bias for boxes and wider spreads than TSAW in all image categories. While Algo4 has median errors at 0 in all four image categories, it has wider spreads than either Algo3 or TSAW.

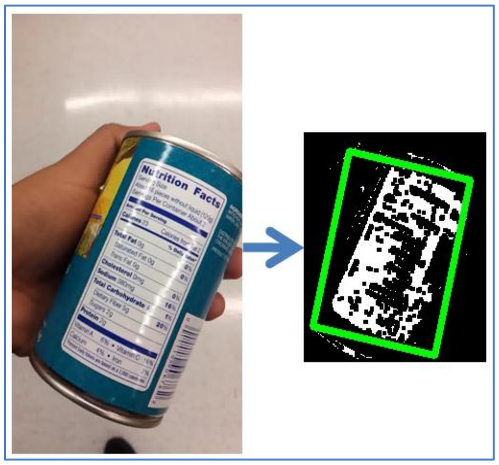

Figure 4.6 Text Skew Angle detection error for TSAW

The main causes of failure for DS1 were light reflections and irregular product shapes. For example, Figure 4.6 shows an image of a can on which all algorithms had deviations. The ground truth text skew on the image in Figure 4.6 is 66.29 degrees. The skew angles estimated by Algo1, Algo2, Algo3, Algo4 and TSAW on the image in Figure 4.6 were 60, 0, 90, 120, 77.52, respectively. For TSAW, the light reflections both above and inside the NL caused the point outliers and the subsequent error in the minimum area rectangle identification.

All the algorithms performed well on DS2. Boxplots for DS2 are not included because most of the errors are close to 0 and outliers dominate the plots for all the methods. Tables 4.1 and 4.2 show the AED and CE statistics computed for DS1 and DS2.

In Table 4.2, Algo5 refers to a more recent version of the text skew algorithm by Papandreou et al. [28]. Recall that the ideal values are 0 and 100, respectively.

Table 4.1. AED and CE statistics on DS1

Algorithm	**AED**	**CE**
Algo1	67.85	3.50
Algo2	52.96	4.80
Algo3	21.44	9.59
Algo4	45.69	5.89
TSAW	8.60	24.98

Table 4.2. AED and CE statistics on DS2

Algorithm	**AED**	**CE**
Algo1	4.20	55.36
Algo2	6.76	46.09
Algo3	6.33	51.59
Algo4	8.28	43.18
Algo5	0.06	74.50
TSAW	6.11	51.18

TSAW has the lowest AED and highest CE on DS1. On DS2 Algo5 has the lowest AED and highest CE values. AED and CE values on DS2 for TSAW are comparable to the other algorithms. The results of the experiments on DS2 show that although TSAW was originally designed to work in real time on NL images it performs on par with the algorithms designed to detect text skew in standard text documents.

4.6 Discussion

A text skew detection algorithm, called TSAW, is presented that does not place any restrictions on text skew magnitudes. Although TSAW is originally designed to work with nutrition label images captured with handheld mobile phone cameras, it can also detect text skews in printed document images which have significantly better lighting and exposure than NL images. TSAW down-samples text images through several iterations of 2D HWT. The HL, LH, and HH matrices are used to identify 2D points with significant intensity changes. The convex hull algorithm [14] encloses these points it with a minimum area rectangle. The text's skew is the enclosing rectangle's rotation angle relative to the true north.

The performance of TSAW was evaluated on two data sets and compared with the performance of the algorithms by Postl [22, 23] (Algo1), by Hull [24] (Algo2), by Li et al. [29] (Algo3) and by Papandreou et al. [27] (Algo4) and its more recent version [28] (Algo5). The first data set (DS1) consisted of 1001 images of NL extracted from videos captured in grocery stores with a smartphone camera. The second data set (DS2) consisted of 2200 scanned document images of single- and multi-column documents.

On DS1, TSAW was found to be the most accurate with a median error of 4.62 as compared to 68.85, 20.92, 9.71, and 17.5 for Algo1, Algo2, Algo3, and Algo4, respectively. All the algorithms performed well on images from DS2, which indicates that even though TSAW is originally designed for NL images it performs on par with the algorithms specifically designed to detect text skews in document images.

CHAPTER 5

OPTICAL CHARACTER RECOGNITION

5.1 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. The history of OCR can be traced as far back as the Nipkow disk [50], 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. 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. This system was not practical since it required a lot of concentration and skill but it laid the basis for a working OCR system. 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. Varying degrees of skew may be introduced into the images while capturing especially using handheld cameras or smartphones. Skew angle is the angle that the text lines in the digital image makes with the horizontal direction. Text in such cases is rotated or distorted and degrades the performance of further processing and may seriously affect the performance of subsequent stages of segmentation and recognition, since the contemporary OCR systems cannot handle rotated text and perform well only in recognizing texts that are linearly aligned. While the horizontally aligned text is easily detected and recognized, skewed text poses a challenge to recognition. In most existing OCR systems, a skew correction process is often performed prior to recognition, should a need arise. Most skew estimation techniques deal with small skew angles less than 5 degrees but perform poorly for images, which contain text lines that are oriented in arbitrary directions.

In this chapter we present an OCR engine that can read skewed text. As far as we are concerned currently there are no such OCR engines available.

5.2 Preprocessing

"Garbage In, Garbage Out" is a very popular phrase in computing. If the input to a 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.

5.2.1 Binarization

The image from the camera is a colored or 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 we 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.

5.2.1.1 Thresholding

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:

$$Bp = \begin{cases} 0, \; if \; Y_p < \tau \\ 255, \; otherwise \end{cases} \qquad (5.1)$$

The threshold τ can be fixed for all images (e.g. τ = 127) or it may be computed for each individual image (e.g. τ = mean(Y)). Otsu's binarization method [51] 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.

5.2.2 Noise Filtering

The authors in [3] 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 5.1and 5.2 show 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.

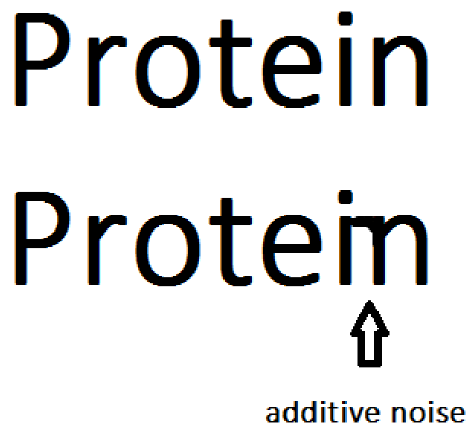

Figure 5.1 Additive noise

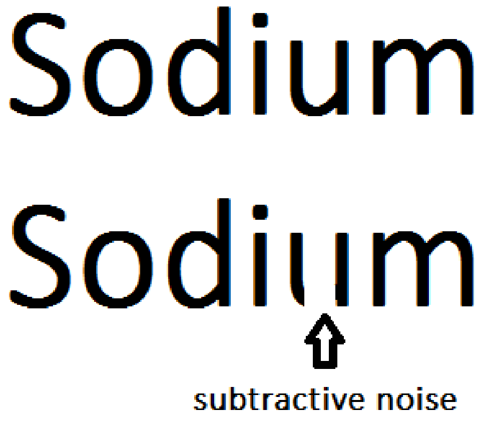

Figure 5.2 Subtractive noise

Jain [53] 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 [54,55,56] and kFill algorithms [57,58] are two very common techniques used to remove salt and pepper noise from images. A study [59] comparing different types of filtering algorithms for removing Gaussian noise in images found that the Wiener filter [60] 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 template matching for the classification process eliminates the need for noise filtering in both the segmentation and classification stages.

5.3 Normalization

An image can suffer from four basic forms of distortion: translation, rotation, scaling and skew [88]. 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 [39] can easily rectify translation, rotation and scaling distortions. The text skew angle detection step 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.

5.4 Skew OCR Engine

The rotation of an entire NL image is an expensive task in terms of computation. It might also introduce noise in an image or remove the extremities of the image altogether. In order to overcome these problems we present an OCR engine that can read skewed text. As far as we are concerned currently there are no such OCR engines available. The development of an OCR engine capable of reading skewed text requires the following steps:

5.4.1 Angular text row segmentation

Text segmentation [41] is an inherent part of an OCR system irrespective of the domain of application. Contemporary OCR systems contain a segmentation module where the text lines, words and ultimately the characters must be segmented properly for its successful recognition. Text segmentation, in general, incorporates line segmentation, word segmentation and character segmentation from a document image. It is the process through which the text component within an image is isolated from the background. Nutrition labels have multiple rows containing information about each nutritional category. Each of these rows must be segmented from the nutrition label before the characters in them can be recognized. We can observe in Figure 5.3 that each of these lines are separated by black colored lines.

The first step in the line segmentation process is to detect the black colored lines within the table and segment the individual rows in the NL by detecting these separator lines in the NL. This can be done by designing a line filter which can scan the localized NL from top-left to bottom-right at an angle determined by the text skew angle detection algorithm. Let N denote the localized nutrition table image, rotated at angle θ and let N_i denote the *i*th row within this image. Let l_j denote the length of the *j*th black colored line segment within Ni that has a length greater than some threshold 'λ'. The *i*th row N_i is assumed to be a black colored line if ($l_j >= \lambda$). 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 row can be segmented from an NL image as shown in Figure 5.4.

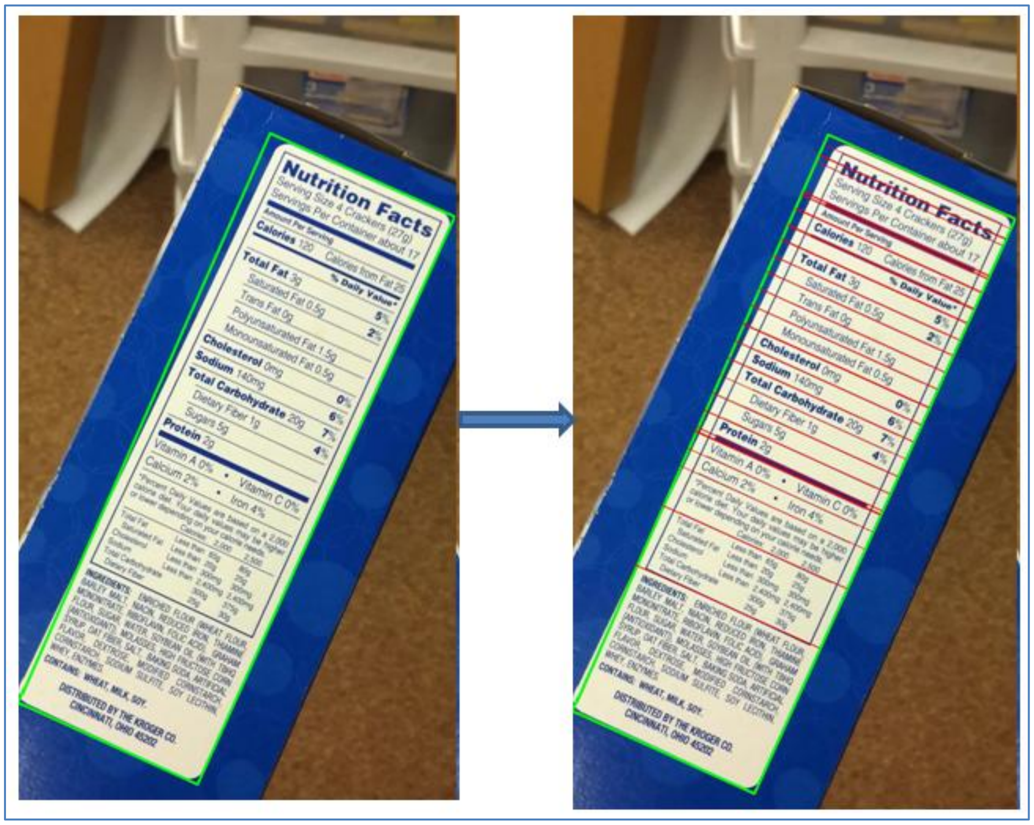

Figure 5.3 Identification of black separator lines in the NL to identify text rows

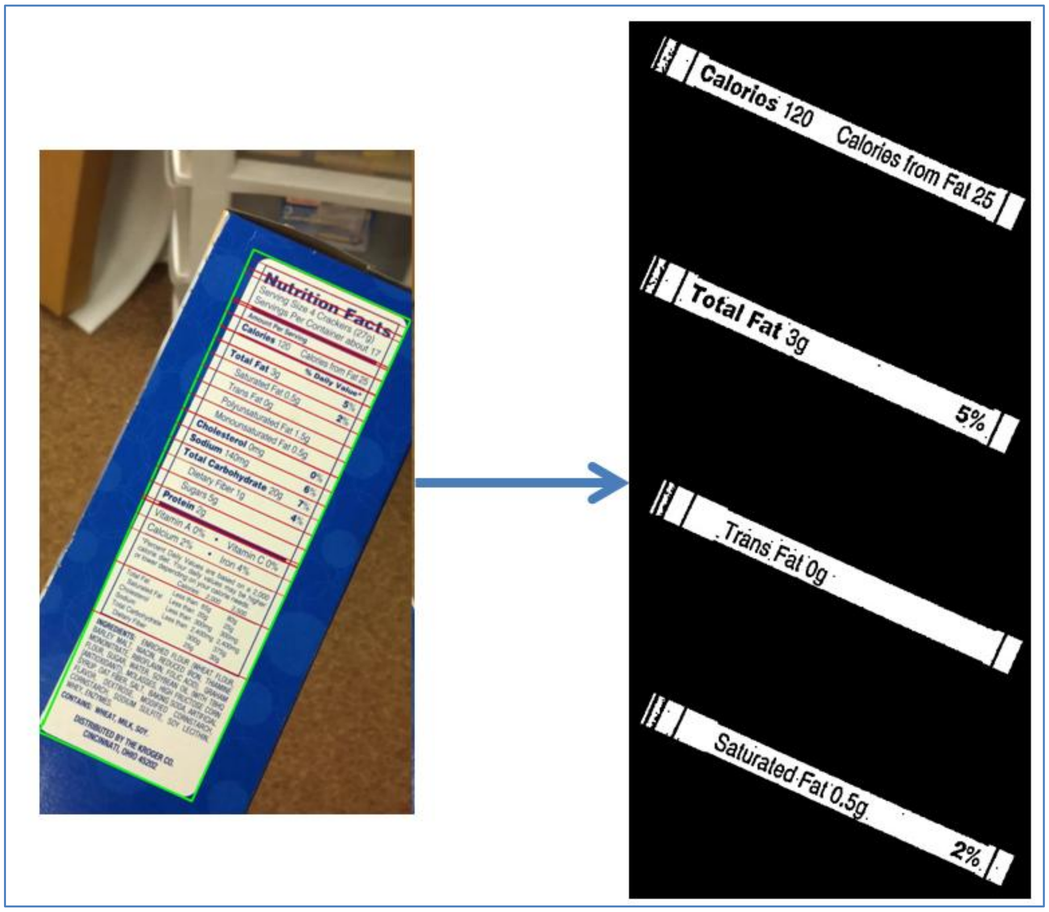

Figure 5.4 Angular text row segmentation

5.4.2 Creating skeletons of characters from the detected text row

Skeletonization is defined as the process by which characters are reduced to skeletons, a set of thin lines (one pixel thick), attached to one another in a few connection points that preserve the topological and geometric properties of its originating object [61]. This is done in order to standardize the shapes and sizes of the fonts in the NL and thereby remove variability. The challenge in this process is to retain the original shape of the character. Apart from OCR, skeleton algorithms are employed in various other applications from medicine [62] to fingerprint analysis [63]. In case of OCR, skeleton 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. A skeleton must preserve the structure of the shape but all redundant pixels should be removed. Figure 5.5. shows a skeleton of the letter "B":

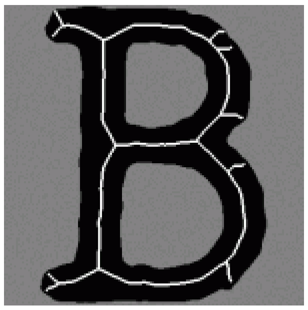

Figure 5.5 Skeleton of letter 'B'.

Skeletonization is the process by which characters are reduced to a set of one pixel thick lines, attached to one another in a few connection points that preserve the topological and geometric properties of its originating object.

Skeletonization algorithms work by examining foreground pixels in the image and then deleting them until only a skeleton remains. This is done by eroding and dilating the image. This process is done in multiple passes and each pass peels away a boundary layer of the image. We can classify skeletonizing algorithms as either sequential or parallel depending on the way pixels are deleted. In sequential algorithms [64], a boundary pixel is marked for deletion depending on the result of the preceding pixel whereas in parallel algorithms [65], each pixel is examined independently of other pixels.

5.4.3 Character Segmentation

Character segmentation has long been a critical area of the OCR process. It is used to isolate individual characters in a word to process them separately. The purpose of our character segmentation technique is, to identify the characters present in the text rows produced after line or row segmentation step as shown in Figure 5.6, by determining their boundaries invariant of rotation and skew. Segmentation of individual letters is achieved through a formalism that we can refer to as histogram analysis at an angle. We scan the text at an angle identified by the skew detection algorithm and generate histograms which are angular projections of the characters as shown in Figure 5.7. Then we try to identify

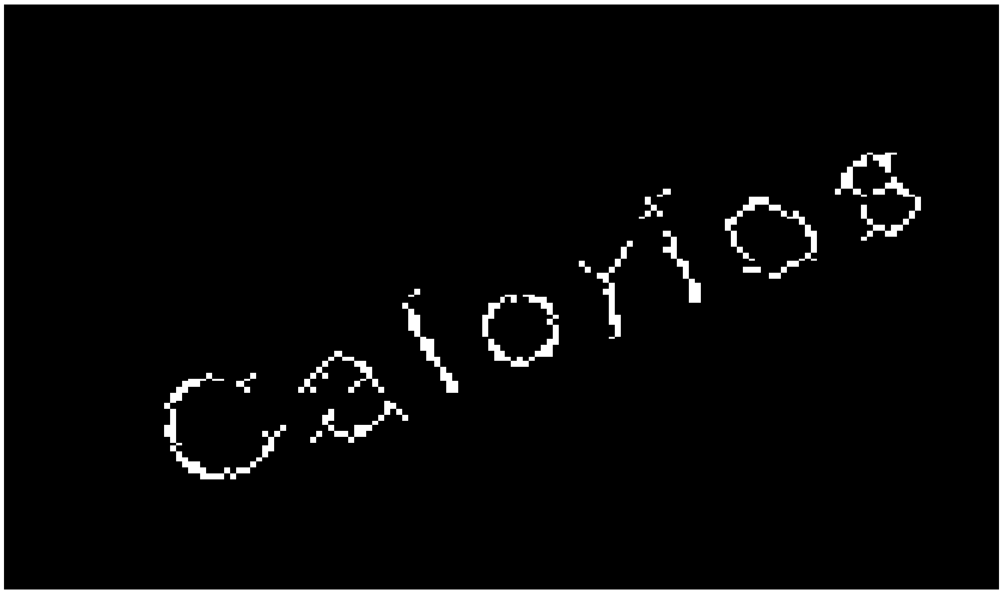

Figure 5.6 A cropped word from a text row produced after line segmentation step

the gaps between the characters as they have a zero angular projection. Figure 5.8 shows the process of identification of these gap regions as the boundary of the character. We can use a small threshold to segment parts of the image that exhibit values of angular projection greater than this threshold to isolate the characters from words for subsequent matching and identification.

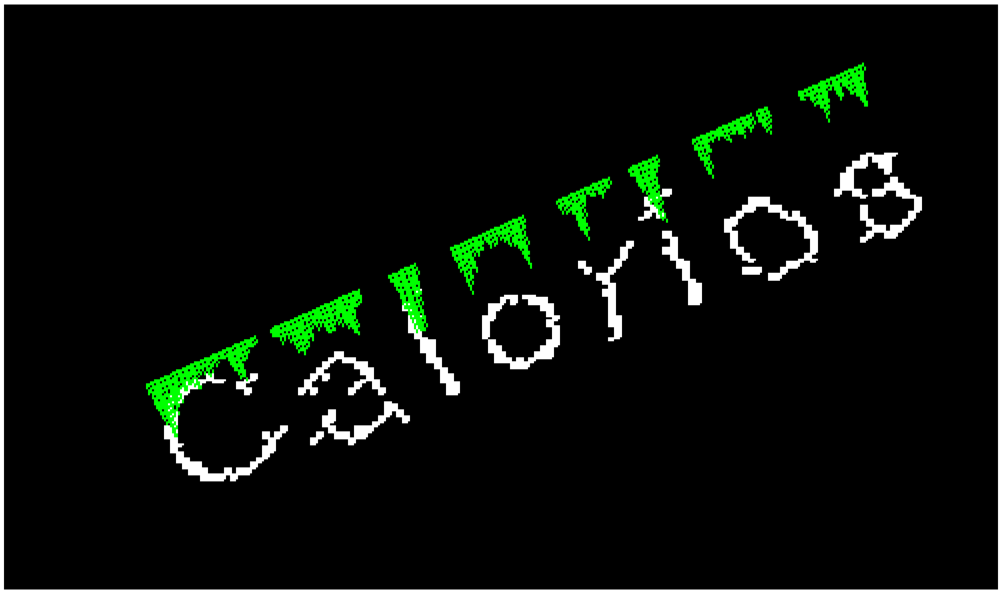

Figure 5.7 Angular projections of characters

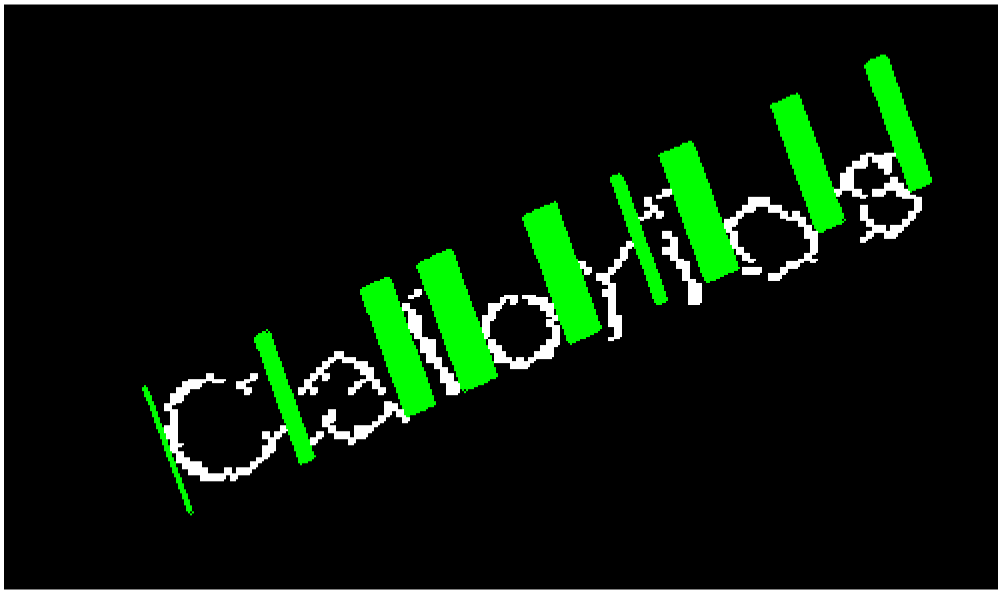

Figure 5.8 Gap identification from angular projections

5.4.4 Zone Vector Matching

Zone matching is a technique used in computer vision for image matching. An image is divided into several sub-images, called zones. In each zone, a specific statistic is

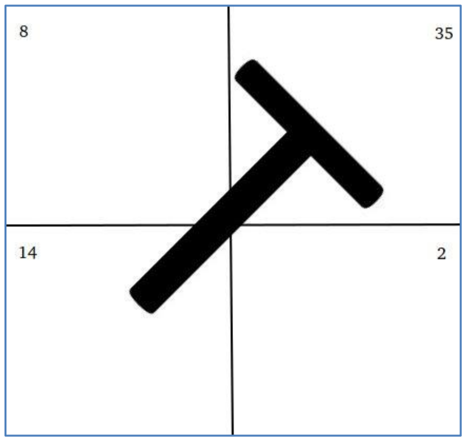

Figure 5.9 Zone vector calculation of a skewed character

computed. For example, that statistic can be the number of horizontal lines or the number of pixels of a specific color. Consider the image below. The image is divided into 4 zones. Suppose that statistic that we are interested in is the count of black pixels in each zone. Moving clockwise from the top left zone (zone 1), there are 8 black pixels in zone 1, 35 black pixels in zone 2, 2 black pixels in zone 3, and 14 black pixels in zone 4. The zone vector for the letter in Figure 5.9 is [8,35,2,14]. If we want to match two images, img1 and img2, we can compute two zone vectors, zv1 and zv2, and then match them by computing their similarity index. One similarity index is cosine similarity that can be computed by the following formula, where A and B are two vectors (sequences of numbers):

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \qquad (5.2)$$

To do zone matching, we compute two zone vectors and then apply a similarity metric, e.g. cosine similarity, that returns a number indicating how similar the two bitmaps are. If we have two character bitmaps and we want to compute how similar they are we would use a similar technique. In OCR, such bitmaps can be obtained through character segmentation when a text image is segmented into individual characters. In order to match the identified characters with a template using zone vector matching , first we would need to scale the size of the characters to those of the templates , create skeletons of the characters and then calculate the zone vector in order to take care of different shapes and sizes of the font. The Skeleton of the template is then matched with the skeleton of the scaled character.

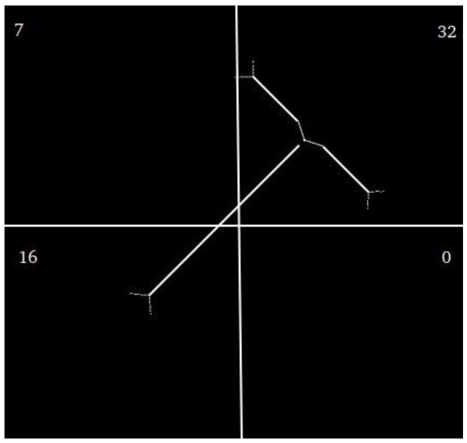

Figure 5.10 The zone vector of the skeleton of a character

As we are dealing with possibly rotated characters we need to create a dictionary of templates at compile time for all letters in the English alphabet (a-z, A-Z) and digits (0-9) rotated at various angles starting at 0 degree to 180 degree with a step of 1 degree. After the skew angle is estimated and the characters are segmented we can try to match the characters with their respective templates rotated at the particular angle using zone vector matching.

5.4.5 Word Matching using Levenshtein Edit Distance

We use zone vector template matching to classify the words in the image. We match the individual letters in the word with the templates and once a word boundary is reached we use Levenshtein Edit Distance formula to match a a given set of words found in a typical Nutrition label. This technique works for our OCR engine due to the fact that we are dealing with a small vocabulary of words instead of the entire English language dictionary. Table 5.1 shows the set of words we have identified for our application.

Table 5.1 Nutrition Label words

Name
Amount
Calcium
Calories
Carbohydrate
Cholesterol
Daily
Dietary
Fat
Fiber
From
Insoluble
Monounsaturated
Per
Polyunsaturated
Potassium
Protein
Saturated
Serving
Sodium
Soluble
Sugars
Total
Trans
Vitamin

5.5 OCR Experiments

The skew OCR engine was tested on a set of 291 images of textchunks of nutrition data obtained after localizing an NL and running the text segmentation algorithm. Each line contained some nutrition related data with both a nutrient text and numbers denoting the amount of the nutrient. The text chunks were rotated by the skew angle of the NL in order to compare the performance of our OCR engine with an open source OCR engine called Tesseract [https://github.com/tesseract-ocr/]. The ground truth values for the text was obtained from six human evaluators who were asked to look at the text segment and note down the readable text in it.

There were three possibilities in the text classification.

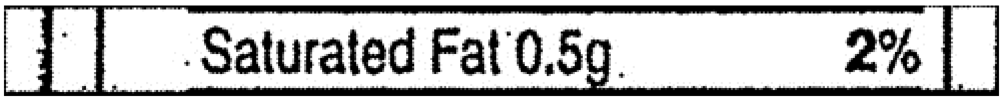
Saturated Fat 0.5g 2%

Figure 5.11 Single line of text

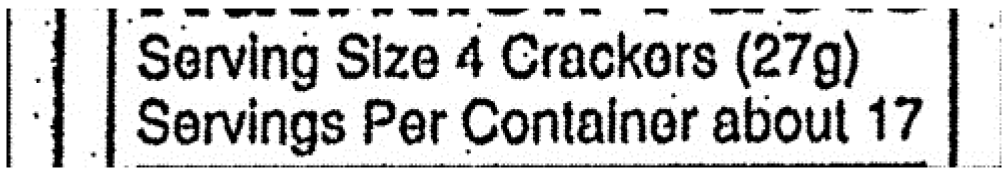
Serving Size 4 Crackers (27g)
Servings Per Container about 17

Figure 5.12 Multi line text

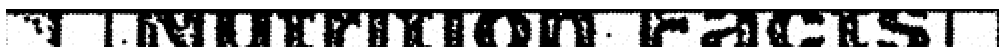

Figure 5.13 Illegible text due to incorrect segmentation

The human evaluators were asked to note down the text from the segments with only a single line of legible text. All the images were then run through the Skew OCR engine and Tesseract. The output from both the OCR engines was evaluated in terms of number of correct identifications with respect to the ground truth obtained from the human evaluators. Out of the 291 images Tesseract was able to read 186 NL segments correctly, whereas the Skew OCR engine was able to read 236 text segments correctly. The skew OCR engine was also tested on 307 skewed text chunks without any rotations as shown in Figure 5.13. The ground truth values were obtained from human evaluators. Out of the 307 skewed text chunks the skew OCR engine was able to read the text of 178 images correctly.

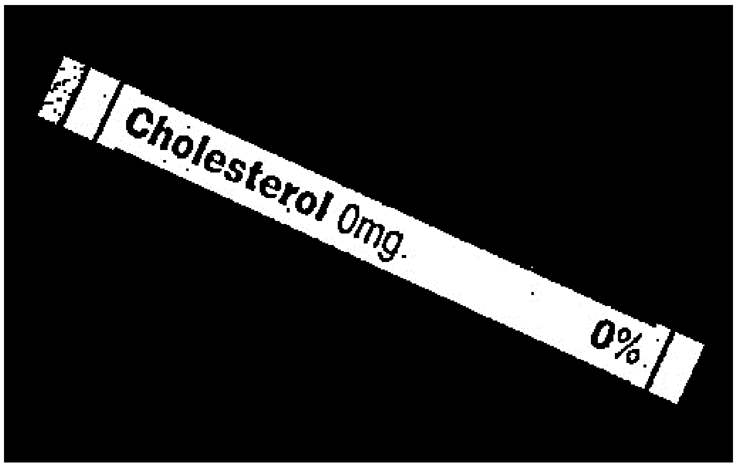

Figure 5.14 Skewed text chunk

CHAPTER 6

CONCLUSION

A proper understanding of nutrition labels (NLs) is essential to ensure eating a healthy, balanced diet. These labels provide information on the amounts of 13 core nutrients and calories in an amount of food, along with a % Daily Value indicator to help people make informed decisions over food choices. This data is presented in the form of a standardized table. Familiarity with the terms of the NLs allows a consumer to make a better decision while shopping for packaged food products and comparing one product with another. However, many consumers find it difficult to interpret the nutritional information on products and feel less motivated to keep track of their nutrient consumption. One way to improve the comprehension and retention of nutritional information by consumers is to use computer vision algorithms that can run on a smartphone. This may include the scanning of barcodes on packed food products to determine the product type and obtain nutritional information about the product if it is available online or scanning the Nutrition Label itself and extracting the nutritional data from it.

The primary challenge in scanning barcodes with a smartphone is the alignment of the camera with the product. We have developed an algorithm called DOG to solve this problem. The first component of this dissertation is the skewed barcode recognition algorithm. The algorithm is developed for in-place vision-based skewed barcode scanning that does not require the smartphone camera alignment. The algorithm is in-place in that it performs no rotation of input images to align localized barcodes for scanning. The algorithm is implemented in a distributed, cloud-based system. The system's front end is a smartphone application that runs on Android 4.3 or higher. The system's back end is currently deployed on a four node Linux cluster used for image recognition and nutritional data storage.

The second component of this dissertation is a text skew angle detection algorithm called TSAW that is used to estimate the skew angle of the NL in an image. The algorithm takes a nutrition label image and applies several iterations of the 2D Haar Wavelet Transform (2D HWT) to downsample the image and to compute the horizontal, vertical, and diagonal change matrices. The values of these matrices are binarized and combined into a result set of 2D change points. The convex hull algorithm is applied to this set to find a minimum area rectangle containing all text pixels. The text skew angle is computed as the rotation angle of the minimum area rectangle found by the convex hull algorithm.

The third and the final component of this dissertation is the skew OCR engine which is a proof of concept prototype that allows users to read the contents of a skewed nutrition label. As far as we are concerned, currently there are no such OCR engines available. We have compared the performance of this OCR engine with one of the most popular open source OCR engines available and found our engine to perform better.

Our future work will focus on the overall advancement of PNUTS. The biggest area of improvement is to integrate all the different components in a single executable project and make an Android user interface for easy testing. This includes the upgradation of the current skew OCR engine and increasing its text recognition rates.

The skew OCR engine is presented as a proof-of-concept prototype and thus has a significant room for improvement. 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 or shiny surfaces.

Another possibility would be the replacement of the zone vector matching algorithm with other types of template matching algorithms such as Hausdorff distance or connected component analysis which might be better suited for varying font shapes and sizes and give better results in terms of absolute character matches.

The system currently uses Levenshtein Edit distance as the spell correction module. We would like to evaluate to performance other spell checking algorithms. This might involve the integration of a context based spell checking into the skew OCR engine so that the relative position of the words in the NL can be utilized to guess the words correctly and thereby improve the recognition rates. We would also like to run more thorough experiments with larger data sets to test the performance of the skew OCR engine.

REFERENCES

[1] United States Department of Agriculture, Economic Research Service data: http://www.usda.gov/factbook/chapter2.pdf

[2] World Health Organization. "Annual World Health Statistics." Avail. at http://www.who.int/gho/publications/world_health_statistics/en/.

[3] Nutrition Labeling and Education Action of 1990. Avail. at http://en.wikipedia.org/wiki/Nutrition_Labeling_and_Education_Act_of_1990.

[4] Food Labeling to Advance Better Education for Life. Avail. at www.flabel.org/en.

[5] S. Sinclair, D. Hammond, and S. Goodman, "Sociodemographic differences in the comprehension of nutritional labels on food products." Journal of Nutr. Educ. Behav. 45(6), pp. 767–72, 2013.

[6] D.J. Graham, R.W. Jeffery, "Location, location, location: eye tracking evidence that consumers preferentially view prominently positioned nutrition information," *J. Am. Diet. Assoc.,* vol. 111, pp. 1704–1711, 2011.

[7] E. Arsand, N. Tatara, G. Ostengen, and G. Hartvigsen, "Mobile Phone-Based Self-Management Tools for Type 2 Diabetes: The Few Touch Application," *Journal of Diabetes Science and Technology*, vol. 4(2), pp. 328-336, March 2010

[8] B.J. Fogg, "A behavior model for persuasive design," in *Proc. 4th International Conference on Persuasive Technology*, Article 40, ACM, New York, USA, 2009.

[9] V. Kulyukin, T. Zaman, and S. Andhavarapu, " Effective Use of Nutrition Labels on Smarphones", In proceedings of the 15th International Conference on Internet Computing and Big Data (ICOMP 2014), pp. 93-99, July 21-24,2014, Las Vegas, NV,USA,CSREA Press, ISBN-1-60132-227-1.

[10] V. Kulyukin and T. Zaman, "Vision-based localization of skewed upc barcodes on smartphones," in *Proc. International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV 2013)*, pp. 344-350, July 2013, CSREA Press, Las Vegas, NV, USA, ISBN 1-60132-252-6.

[11] Nutriglass app for reading skewed barcodes and retrieving NLs: https://play.google.com/store/apps/details?id=org.vkedco.mobappdev.nutriglass&hl=en

[12] V. Kulyukin, A. Kutiyanawala, T. Zaman, and S. Clyde, “Vision-based localization & text chunking of nutrition fact tables on android smartphones.” In *Proc. of the International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV 2013),* pp. 314-320, CSREA Press, Las Vegas, NV, USA, ISBN 1-60132-252-6.

[13] V. Kulyukin and C. Blay, “An algoritm for mobile vision-based localization of skewed nutrition labels that maximizes specificity.” In *Proc. of the 18th International Conference on Image Processing and Pattern Recognition* (IPCV 2014), pp. 3-9, July 21-24, 2014, Las Vegas, NV, USA, CSREA Press, ISBN: 1-60132-280-1.

[14] H. Freeman, and R. Shapira, "Determining the minimum area encasing rectangle for an arbitrary closed curve." Comm. ACM, pp.409-413, 1975.

[15] E. Tekin, and J. Coughlan, “An algorithm enabling blind users to find and read barcodes,” in Proc. of Workshop on Applications of Computer Vision, pp. 1-8, Snowbird, UT, December, 2009. IEEE Computer Society.

[16] E. Tekin and J. Coughlan, “A mobile phone application enabling visually impaired users to find and read product barcodes,” in Proc. of the 12th international conference on Computers helping people with special needs, pp. 290-295, Vienna, Austria, 2010. Springer-Verlag, Berlin, Heidelberg.

[17] S. Wachenfeld, S. Terlunen, and J. Xiaoyi, “Robust recognition of 1-D barcodes using camera phones,” in Proc. of the 19th International Conference on Pattern Recognition, pp. 1-4, Tampa, FL, 2008. IEEE Computer Society.

[18] R. Adelmann, M. Langheinrich, and C. Floerkemeier, “A Toolkit for barcode recognition and Resolving on Camera Phones - Jump Starting the Internet of Things,” in Proc. of workshop on mobile and embedded information systems (MEIS’06) at informatik, Dresden, Germany, 2006.

[19] D.T. Lin, M.C. Lin, and K.Y. Huang, “Real-time automatic recognition of omnidirectional multiple barcodes and DSP implementation.” Appl. Mach. Vision, 22, vol. 2, pp. 409-419, 2011.

[20] O. Gallo and R. Manduchi, “Reading 1D barcodes with mobile phones using deformable templates.” IEEE Transactions on Pattern Analysis and Machine Intelligence, 33, vol. 9, pp. 1834-1843, 2011.

[21] E. Peng, P. Peursum, and L. Li, “Product barcode and expiry date detection for the visually impaired using a smartphone,” in Proc. of international conference on digital image computing techniques and applications, pp. 3-5, Perth Western Australia, Australia, 2012. Curtin University.

[22] W. Postl, "Detection of linear oblique structures and skew scan in digitized documents." In *Proc*. of *International Conference on Pattern Recognition*, pp. 687-689, 1986.

[23] W. Postl, " Method for automatic correction of character skew in the acquisition of a text original in the form of digital scan results." US Patent 4,723,297,1988. https://www.google.com/patents/US4723297.

[24] J.J. Hull, "Document image skew detection: survey and annotated bibliography.WS" In J.J. Hull, S.L. Taylor (eds.), *Document Analysis Systems* II*,* World Scientific Publishing Co., 1997*,* pp. 40-64.

[25] D.S. Bloomberg, G.E. Kopec, and L. Dasari, “Measuring document image skew and orientation.” *Document Recognition* II (SPIE vol. 2422), San Jose, CA, February 6-7, 1995, pp. 302-316.

[26] J. Kanai and A.D. Bagdanov, "Projection profile based skew estimation algorithm for JBIG compressed images." *International Journal on Document Analysis and Recognition*, 1(1), pp.43-51, 1998.

[27] A. Papandreou and B. Gatos, “A novel skew detection technique based on vertical projections.” In *Proc. of International Conference on Document Analysis and Recognition* (ICDAR), pp. 384-388, Sept. 18-21, 2011, Beijing, China.

[28] A. Papandreou, B. Gatos, S.J. Perantonis, and I. Gerardis, "Efficient skew detection of printed document images based on novel combination of enhanced profiles." *Int. J. Doc. Anal. Recognit.* 17(4), pp. 433-454, 2014.

[29] S.T. Li, Q.H. Shen, and J. Sun, "Skew detection using wavelet decomposition and projection profile analysis." *Pattern Recognition Letters*, 28(5), pp. 555–562, 2007.

[30] P. Shivakumara, G. Hemantha Kumar, D.S. Guru, and P. Nagabhushan, "Skew estimation of binary document images using static and dynamic thresholds useful for

document image mosaicing." *In Proc. of National Workshop on IT Services and Applications* (WITSA 2003), pp.51-55, Feb 27–28, New Delhi, India, 2003.

[31] S. Chaudhury and R. Sheth, "Trainable Script Identification Strategies for Indian Languages", Proc. 5th IEEE Intl. Conf. on Document Analysis and Recognition (ICDAR), pp. 657-680, 1999.

[32] A. Busch, W. Boles, and S. Sridharan, "Texture for Script Identification", IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI), 27 (11), pp. 1720-1732, 2005.

[33] J. Hochberg, P. Kelly, T. Thomas, and L. Kerns, "Automatic Script Identification From Document Images Using Cluster-Based Templates", IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), pp. 176-181, 1997.

[34] A.L. Spitz, " Determination of the Script and Language Content of Document Images", IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI), pp. 235-245, 1997.

[35] H. Ma and D. Doermann," Gabor filter based multi-class classifier for scanned document images", Proc. 7th IEEE Intl. Conf. on Document Analysis and Recognition (ICDAR), pp. 968-972, 2003.

[36] L.J Zhou, Y. Lu, and C.L. Tan, "Bangla/English Script Identification Based on Analysis of Connected Component Profiles", 7th IAPR Workshop on Document Analysis Systems (DAS), pp. 243-254, 2006

[37] V. Kulyukin and S. Andhavarapu, "Image blur detection with 2D haar wavelet transform and its effect on skewed barcode scanning." In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition* (IPCV 2015), pp. 125-131. Las Vegas, NV, USA, CSREA Press. ISBN: 1-60132-404-9.

[38] Java source code of image blur detection algorithm described in reference [21]. https://github.com/saratkiran/BlurDetection.

[39] T. Zaman and V. Kulyukin, "Videos of common bags, bottles, boxes, and cans." Available from https://www.dropbox.com/sh/q6u70wcg1luxwdh/LPtUBdwdY1.

[40] NutriGlass: An android application for scanning skewed barcodes. Available from https://play.google.com/store/apps/details?id=org.vkedco.mobappdev.nutriglass.

[41] S. Yanowitz, and A. Bruckstein, "A new method for image segmentation." In Pattern Recognition, 1988., 9th International Conference on (November 1988), vol. 1, pp. 270-275.

[42] Java source code of the TSAW algorithm: https://github.com/tanwirzaman/HaarTextSkewDetection

[43] Y. Nievergelt, "*Wavelets made easy*." Birkhauser, Boston, USA, 2001.

[44] A. Jensen and A. la Cour-Harbo, "*Ripples in mathematics*: *the discrete wavelet transform*." Springer-Verlag, New York, 2001.

[45] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps." *J Fourier Anal. App*. 4(3): 245-267, 1998.

[46] Java source code of text skew dection algorithms: https://github.com/tanwirzaman/TextSkewDetectionAlgorithms/.

[47] Open source onscreen protractor program. http://sourceforge.net/projects/osprotractor/.

[48] Dataset1 (DS1): Online annotated database for NL images. https://usu.box.com/s/9zk660t5h1g0dmw4pjj1x1yp6r7zovp3.

[49] Dataset2 (DS2) : Online database of text documents. https://www.iit.demokritos.gr/~alexpap/dataset_A.rar.

[50] H. Schantz, "The history of OCR, optical character recognition." Recognition Technologies Users Association, 1982.

[51] N. Otsu, "A threshold selection method from gray-level histograms." IEEE Transactions on Systems, Man, and Cybernetics 9 (1979), 62-66.

[52] A.O. Akyuz and E. Reinhard, "Noise reduction in high dynamic range imaging." Journal of Visual Communication and Image Representation 18, 5 (October 2007), 366-376.

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

[54] S.J. Ko, and Y.H. Lee, "Center weighted median filters and their applications to image enhancement." Circuits and Systems, IEEE Transactions on 38, 9 (September 1991), 984-993.

[55] K. Toh, H. Ibrahim, and M. Mahyuddin, "Salt-and-pepper noise detection and reduction using fuzzy switching median filter." Consumer Electronics, IEEE Transactions on 54, 4 (November 2008), 1956-1961.

[56] W. Zhou and D. Zhang, "Progressive switching median filter for the removal of impulse noise from highly corrupted images." Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on 46, 1 (January 1999), 78-80.

[57] L. O'Gorman, "Image and document processing techniques for the rightpages electronic library system." In Pattern Recognition, 1992. Vol.II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on (August 1992), vol. 2, pp. 260-263.

[58] H. Al-Khaffaf, A. Talib, and R. Salam, "Removing salt-and-pepper noise from binary images of engineering drawings." In Pattern Recognition, 2008. ICPR 2008. 19th International Conference on (December 2008), vol. E92.D, pp. 689-704.

[59] B. Singh, Mridula, V. Chand, A. Mittal, and D. Ghosh, "A comparative study of different approaches of noise removal for document images." In Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011, K. Deep, A. Nagar, M. Pant, and J. C. Bansal, Eds., vol. 130 of Advances in Intelligent and Soft Computing. Springer Berlin / Heidelberg, 2012, pp. 847-854.

[60] M. Kazubek, "Wavelet domain image denoising by thresholding and wiener filtering." Signal Processing Letters, IEEE 10, 11 (November 2003), 324-326.

[61] 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.

[62] K. Palagyi and A. Kuba, "A thinning algorithm to extract medial lines from 3d medical images." In Proceedings of the 15th International Conference on Information Processing in Medical Imaging (London, UK, UK, 1997), IPMI '97, Springer-Verlag, pp. 411-416.

[63] Y. He, J. Tian, X. Luo, and T. Zhang, "Image enhancement and minutiae matching in fingerprint verification." Pattern Recogn. Lett. 24, 9-10 (June 2003), 1349-1360.

[64] P. Kardos, G. Nemeth, and K. Palagyi, "An order independent sequential thinning algorithm." In Proceedings of the 13th International Workshop on Combinatorial ImageAnalysis (Berlin, Heidelberg, 2009), IWCIA '09, Springer-Verlag, pp. 162-175

[65] T.Y. Zhang and C.Y. Suen, "A fast parallel algorithm for thinning digital patterns." Communications of ACM 27, 3 (March 1984), pp. 236-239

APPENDICES

APPENDIX A

DOMINANT ORIENTATION OF GRADIENTS PSEUDOCODE

```
1. FUNCTION ComputeDOGs(Image, MaskSize)
2.  ThetaThresh = 360; MagnThresh = 20.0;
3.  FreqThresh = 0.02; ListOfNeighborhoods = [];
4.  GGOT = new HashTable();
5.  Foreach mask of MaskSize in Image Do
6.    SubImage = subimage currently covered by mask;
7.    RGOT = ComputeRGOT(SubImage, ThetaThresh, MagnThresh);
8.    GGOT[coordinates of masks' top left corner] = RGOT;
9.    If RGOT ≠ NULL Then
10.     RGOT.row = mask.row;
11.     RGOT.column = mask.column;
12.     If (RGOT(freq)*1.0/(SubImage.cols * subImage.rows)>=FreqThresh)
13.        Neighbourhood=FindNeighbourhoodForRGOT(RGOT, ListOfNeighborhoods);
14.        If ( Neighbourhood ≠ NULL ) Then Neighbourhood.add(RGOT);
15.        Else
16.          NewNeighbourhood=Neighbourhood(RGOT.dtheta, ListOfNeighborhoods.size+1);
17.          NewNeighbourhood.add(RGOT)
18.          ListOfNeighborhoods.add(newNeighborhood);
19.        EndIf
20.     EndIf
21.   EndIf
22. EndForeach
```

```
1. FUNCTION ComputeRGOT(Image, THETA_THRESH, MAGN_THRESH)
2.  Height = Image.height; Width = Image.width;
3.  RGOT = new HashTable();
4.  For row = 1 to Height Do
5.    For column = 1 to Width Do
6.      DX = Image(row, column+1)[0]-Image(row, column-1)[0];
7.      DY = Imaget(row -1, column)[0]- Image(row +1, column)[0];
8.      GradientMagn = sqrt(DX^2+DY^2);
9.      GradTheta = arctan(DY/DY)*180/PI;
10.     If (|GradTheta|≤THETA_THRESH) AND (|GradMagn|≥MAGN_THRESH)) Then
11.         If (RGOT contains GradTheta) Then
12.              RGOT[GradTheta] += 1;
13.         Else
14.             RGOT[GradTheta] = 1;
15.         EndIf
16,     EndIf
17.   EndFor
18. EndFor
19. Return RGOT;

1. FUNCTION FindNeighbourhoodForRGOT(RGOT, ListOfNeighborhoods)
2.  ThetaThresh = 5.0;
3.  Foreach neighborhood in LisOfNeighborhoods Do
4.     If (|neighborhood.dtheta - RGOT.theta| < ThetaThresh) Then
5.        If (HasNeighborMmask(neighborhood, RGOT)) Then
6.            Return neighborhood;
7.        EndIf
8.     EndIf
9. EndForeach
```

```
1. FUNCTION HasNeighborMask(neighborhood, RGOT)
2,   Foreach RGOTMember in Neighborhood.members Do
3.       If (  RGOT.row = RGOTMember.row  ) Then
4.          If ( | RGOT.column – RGOTMember.column | = maskSize  ) Then
5.           Return True;
6.          EndIf
7.       EndIf
8.       If ( RGOT.column = RGOTMember.column ) Then
9.          If ( | RGOT.row – RGOTMember.row | = maskSize  ) Then
10.          Return True;
11.           EndIf
12.       EndIf
13.       If ( |RGOT.column – RGOTMember.column| =  maskSize  ) Then
14.           If ( |RGOT.row – RGOTMember.row| = maskSize  ) Then
15.          Return True;
16.           EndIf
17.      EndIf
18. EndForeach
```

APPENDIX B

TEXT SKEW ANGLE WAVELETS (TSAW) PSEUDOCODE

```
1.  FUNCTION DetectTextSkewAngle(Img, N, NITER)
2.    [AVRG, HC, VC, DC] = 2DHWT(Img, NITER);
3.    $n = N/2^{\mathrm{NITER}}$;
4.    Binarize(HC, $n$); Binarize(VC, $n$); Binarize(DC, $n$);
5.    FindSkewAngle(HC,VC,DC, $n$);
5.  FUNCTION Binarize(Matrix, n, θ=5, v1=255, v2=0)
6.    For $r = 1$ to $n$
7.      For $c = 1$ to $n$
8.        If $Matrix[r,c] > \theta$ Then
9.          $Matrix[r,c]$ = v1;
10.       Else
11.         $Matrix[r,c]$ = v2;
12.       End If
13.     End For
14.   End For

15. FUNCTION FindSkew($HC$, $VC$, $DC$, $n$, $\alpha, \beta, \gamma$, θ)
16.   Initialize a new $n \times n$ $S[i,j]$ matrix with 0's;
17.   For $r = 1$ to $n$ Do
18.     For $c = 1$ to $n$ Do
19.     If $\alpha HC[r,c] + \beta VC[r,c] + \gamma DC[r,c] \geq \theta$ Then
20.         $S[r,c] = 255$;
21.       Else
22.         $S[r,c] = 0$;
23.     End If
24.     End For
25.   End For
26.   return FindRotationAngle(FindMinAreaRectangle($S$));
```

CURRICULUM VITAE

Tanwir Zaman

CAREER OBJECTIVE:

To obtain a position in an economically competitive high-tech corporation which requires technical expertise, leadership, and communication skills. Special areas of interest: computer vision, machine learning, distributed systems.

EDUCATION:

Ph.D., Computer Science. Utah State University, Logan, UT, 2016.

B.Tech., Information Technology. West Bengal University of Technology, Kolkata, India, 2008.

INDUSTRY EXPERIENCE:

Software Engineer, Cognizant Technology Solutions, 2009 to 2010

Research Assistant, Utah State University, 2011 to 2016

Software Engineer Intern, Intermountain Healthcare, 2015

TEACHING EXPERIENCE:

Graduate Instructor. CS 3430 Python and Perl Programming, Utah State University. 2014 to 2016.

Teaching assistant at Utah State University:

CS 3200: Mobile Apps development, 2012 to 2015

CS 6890: Android programming, Spring 2013

CS 5200: Distributed Network Programming, Fall 2012

PUBLICATIONS:

1. T. Zaman, V. Kulyukin. Text Skew Angle Detetion in Vision-Based Scanning of Nutrition Labels. In Ed. H. R. Arabnia, L. Deligiannidis, F. G. Tinetti. *Proceedings of the 19th International Conference on Image Processing, Computer Vision, & Pattern Recognition* (IPCV 2015), pp. 139-144, July 27-30, 2015, Las Vegas, NV, USA, CSREA Press, ISBN: 1-60132-404-9

2. V. Kulyukin & T. Zaman. Vision-Based Localization and Scanning of 1D UPC and EAN Barcodes with Relaxed Pitch, Roll, and Yaw Camera A lignment Constraints. *International Journal of Image Processing* (IJIP), Volume (8) : Issue (5) : 2014, pp. 355-383.

3. Kulyukin, V., Zaman, T., and Andhavarapu, S. Effective Use of Nutrition Labels on Smarphones. In *Proceedings of the 15th International Conference on Internet Computing and Big Data* (*ICOMP 2014*), pp. 93 - 99, July 21-24, 2014, Las Vegas, NV, USA, CSREA Press, ISBN: 1-60132-227-1.

4. Kulyukin, V. and Zaman, T. An Algorithm for In-Place Vision-Based Skewed 1D Barcode Scanning in the Cloud. In *Proceedings of the 18th International Conference on Image Processing and Pattern Recognition* (*IPCV 2014*), pp. 36-42, July 21-24, Las Vegas, NV, USA, CSREA Press, ISBN: 1-60132-280-1.

5. Kulyukin, V. and Zaman T. (2013). Vision-Based Localization of Skewed UPC Barcodes on Smartphones. In *Proceedings of the International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV 2013)*, pp. 344-350, pp. 314-320, ISBN 1-60132-252-6, CSREA Press, Las Vegas, NV, USA.

6. Kulyukin, V., Kutiyanawala, A., Zaman, T., & Clyde, S. (2013). Vision-Based Localization & Text Chunking of Nutrition Fact Tables on Android Smartphones. In *Proceedings of the International Conference on Image Processing, Computer Vision, & Pattern Recognition (IPCV 2013),* pp. 314-320, ISBN 1-60132-252-6, CSREA Press, Las Vegas, NV, USA.

7. Kulyukin, V., Zaman, T., Andhavarapu, A., and Kutiyanawala, A. (2012). Eyesight Sharing in Blind Grocery Shopping: Remote P2P Caregiving through Cloud Computing. In *Proceedings of the 13-th International Conference on Computers Helping People with Special Needs* (ICCHP 2012), K. Miesenberger et al. (Eds.), ICCHP 2012, Part II, Springer Lecture Notes on Computer Science (LNCS) 7383, pp. 75-82, July 11-13, 2012, Linz, Austria.