8-2015

Image Blur Detection with Two-Dimensional Haar Wavelet Transform

Sarat Kiran Andhavarapu
Utah State University

IMAGE BLUR DETECTION WITH TWO-DIMENSIONAL HAAR WAVELET TRANSFORM

by

Sarat Kiran Andhavarapu

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

of

MASTER OF SCIENCE

in

Computer Science

Approved:

_____	_____
Dr. Vladimir Kulyukin Major Professor	Dr. Curtis Dyreson Committee Member
_____	_____
Dr. Nicholas Flann Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2015

ABSTRACT

Image Blur Detection with Two-Dimensional Haar Wavelet Transform

by

Sarat Kiran Andhavarapu, Master of Science

Utah State University, 2015

Major Professor: Dr. Vladimir Kulyukin
Department: Computer Science

An algorithm is presented for image blur detection with the use of Two-Dimensional Haar Wavelet transform (2D HWT). The algorithm classifies an image as blurred or sharp by splitting the image into *N* x *N* tiles, applying several iterations of the 2D HWT to each tile, and grouping horizontally, vertically, and diagonally connected tiles with pronounced changes into tile clusters. Images with large tile clusters are classified as sharp. Images with small tile clusters are classified blurred. If necessary, the blur extent can be estimated as the ratio of the total area of the connected tile clusters and the area of the image. When evaluated on a sample of five hundred images, the algorithm performed on par or better than two other blur detection algorithms found in the literature. The effect of blur detection on skewed barcode scanning is investigated by integrating the presented blur detection algorithm into a skewed barcode scanning algorithm. The experimental results show that blur detection had a positive effect on skewed barcode scanning rates and OCR rates.

(46 pages)

PUBLIC ABSTRACT

Image Blur Detection with Two-Dimensional Haar Wavelet Transform

Sarat Kiran Andhavarapu

Efficient detection of image blur and its extent is an open research problem in computer vision. Image blur has a negative impact on image quality. Blur is introduced into images due to various factors including limited contrast, improper exposure time or unstable device handling. Toward this end, an algorithm is presented for image blur detection with the use of Two-Dimensional Haar Wavelet transform (2D HWT). The algorithm is experimentally compared with two other image blur detection algorithms frequently cited in the literature. When evaluated over a sample of images, the algorithm performed on par or better than the two other blur detection algorithms.

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 for both me and this project.

I would also like to thank the members of my committee, Professor Nicholas Flann and Professor Curtis Dyreson. Their suggestions, support, and hard work have been invaluable and deeply appreciated.

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

Sarat Kiran Andhavarapu

CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

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 [1]. Similar initiatives or legislative acts (e.g., EU FLABEL [2]) exist in other countries. Unfortunately, even highly motivated consumers, who look for NLs to make healthy food choices, sometimes find it difficult to locate and to comprehend nutrition information on many products [3]. Recent investigations of NL use by consumers have used digital cameras to track consumers' eye movements to better understand how consumers locate and understand NLs [4].

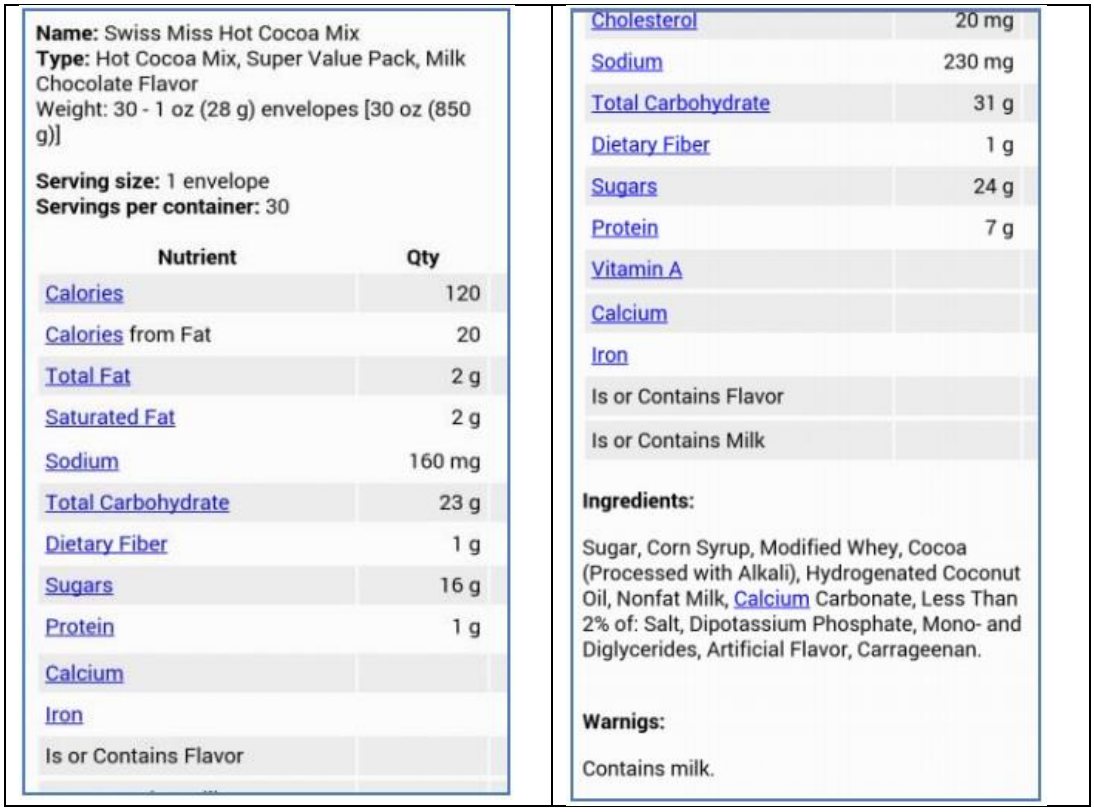
In a previous research project of the USU Computer Science Assistive Technology Laboratory, a system was implemented for effective use of nutrition label use on smartphones [5]. The system's front end is implemented as a smartphone application. The application runs on the Google Nexus 4 smartphone with Android 4.3 or 4.4. The system's back end is currently a four node Linux cluster used for image recognition and data storage.

The front end smartphone sends captured frames to the back end cluster across a wireless data channel (e.g., 3G/4G/Wi-Fi) where barcodes, both skewed and aligned, are recognized [6]. Corresponding NLs are retrieved from a cloud database, where they are stored as HTML documents, and sent across the wireless data channel back to the

smartphone where the HTML documents are displayed on the touchscreen. Wikipedia links to important nutrition terms are embedded for better comprehension.

A nutrition label crawler [5] was implemented to get the nutrition table information from various websites. The current version of the crawler scrapes three public web sites dedicated to nutrition: www.directionsforme.org, www.smithsfoodanddrug.com, www.digit-eyes.com. Each URL is parsed with the Python BeautifulSoup library (http://www.crummy.com/software/BeautifulSoup/) implemented on top of the popular Python XML and HTML parsers LXML (lxml.de) and HTML5LIB (http://code.google.com/p/html5lib/) . Each URL is parsed to obtain the NL, ingredients, warnings, and categories. When this information is extracted, a new HTML document is generated that contains not only the extracted information but also embedded Wikipedia links to all nutrition terms used in the tabular part of the NL. A path to this HTML document is saved in a database under a specific barcode. The NL database currently includes 200,000 products compiled from public web sites by the crawler.

When a barcode is recognized at the back end in an image sent to it from the smartphone, its HTML document (if there is one) is sent back to the smartphone and displayed on the touchscreen, as shown in Figure 1.1. Consumers can use standard touch gestures (e.g., zoom in/out, swipe) for manipulating the label's surface size or browsing its contents. For example, Figure 1.1 (left) shows the lower part of the NL displayed in Figure 1.1 (right) after the user does a down swipe on the touchscreen. When the user clicks on an embedded link, a Wiki page for that nutrient is displayed, as shown in Figure 1.2.



Name: Swiss Miss Hot Cocoa Mix
Type: Hot Cocoa Mix, Super Value Pack, Milk Chocolate Flavor
Weight: 30 - 1 oz (28 g) envelopes [30 oz (850 g)]

Serving size: 1 envelope
Servings per container: 30

Nutrient	Qty
Calories	120
Calories from Fat	20
Total Fat	2 g
Saturated Fat	2 g
Sodium	160 mg
Total Carbohydrate	23 g
Dietary Fiber	1 g
Sugars	16 g
Protein	1 g
Calcium	
Iron	
Is or Contains Flavor	

Figure 1.1 Upper part of NL (left); Lower part of NL (right)

The nutrition label crawler was a temporary solution. Some sites may contain inaccurate or obsolete nutrition information. Web site scraping is unreliable in the long term in that a site that currently permits robots may prohibit them in the future. Consequently, the USU CSATL currently has ongoing projects to automate NL scanning with computer vision and OCR [6].

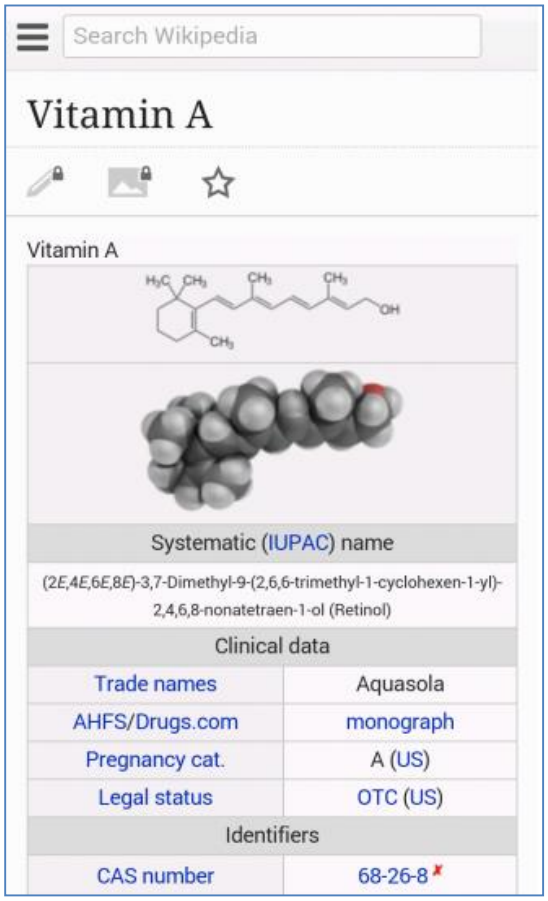


Figure 1.2 Wiki page for Nutrient A

While processing an image for subsequent skewed barcode scanning or OCR, the blurriness of an image plays a major role on the accuracy of specific barcode scanning or OCR algorithms. For example, in [7, 8] an algorithm is developed for in-place vision-based skewed barcode scanning with relaxed pitch, roll, and yaw camera alignment constraints. The skewed barcode scanning experiments were conducted on a set of 506 video recordings of common grocery products. Experiments with the algorithm showed that the scanning results were substantially higher on sharp images than on blurred ones.

A limitation of that algorithm was that it did not filter the blurred frames out of the barcode localization and scanning process.

The same limitation was experimentally discovered in another algorithm that was developed for mobile vision-based localization of skewed nutrition labels (NLs) on grocery packages that maximizes specificity, i.e., the percentage of true negative matches out of all possible negative matches [9]. The NL localization algorithm works on frames captured from the smartphone camera's video stream and localizes NLs skewed up to 35-40 degrees in either direction from the vertical axis of the captured frame. The NL localization algorithm uses three image processing methods: edge detection, line detection, and corner detection. Experimentally it is discovered that the majority of false negative matches were caused by blurred images.

Both the Canny edge detector [10] and dilate-erode corner detector [11] used in the algorithm require rapid and contrasting changes to identify key points and lines of interest. These data cannot be readily retrieved from blurred images, which results in run-time barcode scanning and NL localization failures. Consequently, effective image blur detection methods will likely improve both skewed barcode scanning and NL localization rates.

The remainder of our report is organized as follows. In Chapter 2, we present and analyze related work. In Chapter 3, we outline the details of our image blur detection algorithm. In Chapter 4, we present our experiments with the blur detection algorithm and experimentally compare the algorithm's performance with two other blur detection algorithms found in the literature [12, 13]. In Chapter 5, the results of the experiments are

discussed. Chapter 6 summarizes our findings, presents our conclusions, and outlines some research venues we would like to pursue in the future.

CHAPTER 2

RELATED WORK

In this chapter, we review several existing algorithms for blur detection frequently cited in the literature.

Mallat and Hwang [14] mathematically prove that signals carry information via irregular structures and singularities. In particular, they show that the local maxima of the wavelet transform detect the locations of irregularities. For example, the 2D wavelet transform maxima indicate the locations of edges in images. The Fourier analysis [15], which has been traditionally used in physics and mathematics to investigate irregularities, is not always suitable to detecting the spatial distribution of such irregularities.

According to Tong et al. [13], image blur detection methods can be broadly classified as direct or indirect. Figure 2.1 shows different edges present in an image. Indirect methods characterize image blur as a linear function $I_B = B \cdot I_O + N$, where I_O is the original image, B is an unknown image blur function, N is a noise function, and I_B is the resulting image after the introduction of the blur and noise.

Indirect methods consider B unknown and use various techniques to estimate it. Rooms et al. [16] propose a wavelet-based method to estimate the blur of an image by looking at the sharpness of the sharpest edges in the image. The Lipschitz exponents [17] are computed for the sharpest edges and a relation between the variance of a Gaussian

point spread function and the magnitude of the Lipschitz exponent is shown to be dependent on the blur present in the image and not on the image contents.

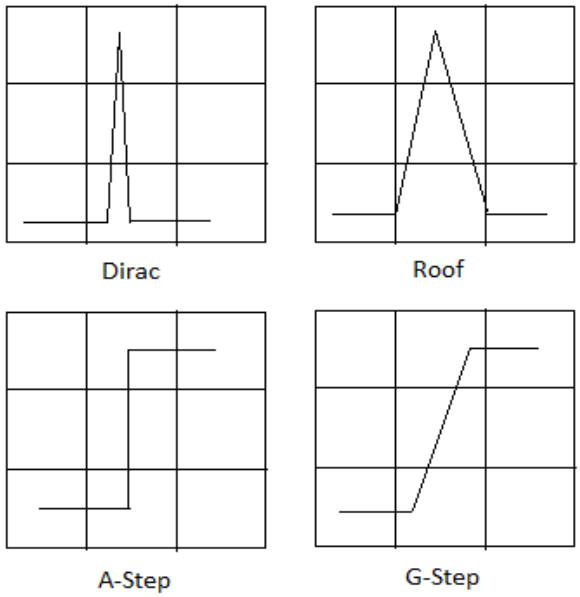


Figure 2.1 Edge classification

Venkatakrishnan et al. [18] show that the wavelet transform modulus maxima (WTMM) detect all the singularities of a function and describe strategies to measure their regularity and propose an algorithm for characterizing singularities of irregular signals. The researchers present a method for measuring the Lipschitz exponents that uses the area between the straight line satisfying specific properties and the curve of the WTMM in a finite scale interval in the log-log plot of scales versus WTMM as the objective function.

Pavlovic and Tekalp [19] propose a formulation of the maximum likelihood (ML) blur identification based on parametric modeling of the blur in the continuous spatial coordinates. Unlike ML blur identification methods based on discrete spatial domain blur

models, their method finds the ML estimate of the extent and the parameters of arbitrary point spread functions that admit a closed form parametric description in the continuous coordinates. Experiments show significant results for the cases of 1D uniform motion blur, 2D out-of-focus blur, and 2D truncated Gaussian blur at different signal-to-noise ratios.

Panchapakesan et al. [20] present an indirect method for image blur identification from vector quantizer encoder distortion. The method takes a set of training images from all candidate blur functions. These sets are used to train vector quantizer encoders. The a-priori unknown blur function is identified from a blurred image by choosing among the candidate vector quantizer encoders the encoder with the lowest distortion. The researchers investigated two training methods: the generalized Lloyd algorithm and a non-iterative discrete cosine transform (DCT)-based approach.

Direct methods estimate blur extent on the basis of some distinctive features directly found in images such as edges, corners, or discrete cosine transform (DCT) coefficients. Marichal et al. [21] estimate image blur based on the histogram computation of non-zero DCT coefficients computed from MPEG or JPEG compressed images. The proposed method takes into account the DCT information from the entire image. A key assumption is that any edge type will likely cross some 8 x 8 blocks at least once in the image. The camera and motion blur is estimated through the globalization among all DCT blocks.

Tong et al. [13] propose a direct method similar to the one proposed in this paper in that it also uses the 2D HWT. Their method is based on the assumption that the

introduction of blur has different effects on the four main types of edges shown in Figure 1: Dirac, A-Step, G-Step, and Roof. It is claimed that in blurred images the Dirac and A-Step edges disappear while G-Step and Roof edges lose their sharpness. The method classifies an image as blurred on the basis of the presence or absence of Dirac and A-Step edges and estimates the blur extent as the percentage of G-Step and Roof edges present in the image.

The algorithm presented in this report is also based on the 2D HWT. However, it does not extract any explicit features such as edges or corners from the image. Instead, it uses the 2D HWT to detect square tiles with pronounced changes and combines those tiles into larger segments without explicitly computing the causes of those changes. This algorithm continues our investigation of vision-based barcode and nutrition label scanning on mobile phones with relaxed pitch, yaw, and roll constraints [7, 8]. As we have previously reported, most false negatives in our experiments were caused by blurred images. While newer models of smartphones will likely have improved camera stability and focus capabilities, software techniques to detect blurred images can still make vision-based skewed barcode scanning and nutrition information extraction more reliable and efficient. Since our barcode scanning and nutrition information extraction algorithms are cloud-based, eliminating blurred images from processing will likely improve the network throughput and decrease data plan consumption rates.

CHAPTER 3

BLUR DETECTION ALGORITHM

3.1 Overview

In this chapter, we describe our blur detection algorithm based on the 2D HWT. The chapter consists of two parts. The first part briefly reviews the formal foundations of the 2D HWT [12]. The second part describes our algorithm. Images are split into *N x N* tiles, by applying several iterations of the 2D HWT on each tile and grouping horizontally, vertically, and diagonally connected tiles with pronounced changes into title clusters. Images with large title clusters are classified as sharp. Images with small clusters are classified as blur.

3.2 2D Haar Transform

Our implementation of the 2D HWT is based on the approach taken in [12] where the transition from 1D Haar wavelets to 2D Haar wavelets is based on the products of basic wavelets in the first dimension with basic wavelets in the second dimension. For a pair of functions f_1 and f_2 their tensor product is defined as$(f_1 \times f_2)(x, y) = f_1(x) \cdot f_2(x)$. Two 1D basic wavelet functions are defined as follows:

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

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

The 2D Haar wavelets are defined as tensor products of $\varphi_{[0,1[}(r)$ and $\psi_{[0,1[}(r)$: $\Phi_{0,0}^{(0)}(x,y) = \left(\varphi_{[0,1[} \times \varphi_{[0,1[}\right)(x,y)$, $\Psi_{0,0}^{h,(0)}(x,y) = \left(\varphi_{[0,1[} \times \psi_{[0,1[}\right)(x,y)$, $\Psi_{0,0}^{v,(0)}(x,y) = \left(\psi_{[0,1[} \times \varphi_{[0,1[}\right)(x,y)$, $\Psi_{0,0}^{d,(0)}(x,y) = \left(\psi_{[0,1[} \times \psi_{[0,1[}\right)(x,y)$. The superscripts h, v, and d indicate the correspondence of these wavelets with horizontal, vertical, and diagonal changes, respectively. The horizontal wavelets detect horizontal (left to right) changes in 2D data, the vertical wavelets detect vertical (top to bottom) changes in 2D data, and the diagonal changes detect diagonal changes in 2D data.

In practice, the basic 2D HWT is computed by applying a 1D wavelet transform of each row and then a 1D wavelet transform of each column. Suppose we have a 2 x 2 pixel image

$$\begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} 11 & 9 \\ 7 & 5 \end{bmatrix}.$$

Applying a 1D wavelet transform to each row results in the following 2 x 2 matrix:

$$\begin{bmatrix} \frac{s_{0,0}+s_{0,1}}{2} & \frac{s_{0,0}-s_{0,1}}{2} \\ \frac{s_{1,0}+s_{1,1}}{2} & \frac{s_{1,0}-s_{1,1}}{2} \end{bmatrix} = \begin{bmatrix} \frac{11+9}{2} & \frac{11-9}{2} \\ \frac{7+5}{2} & \frac{7-5}{2} \end{bmatrix} = \begin{bmatrix} 10 & 1 \\ 6 & 1 \end{bmatrix}.$$

Applying a 1D wavelet transform to each new column is fetches us the result 2 x 2 matrix:

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

The coefficients in the result matrix obtained after the application of the 1D transform to the columns express the original data in terms of the four tensor product wavelets $\Phi_{0,0}^{(0)}(x,y)$, $\Psi_{0,0}^{h,(0)}(x,y)$, $\Psi_{0,0}^{v,(0)}(x,y)$, and $\Psi_{0,0}^{d,(0)}(x,y)$:

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

$$2\Psi_{0,0}^{v,(0)}(x,y) + 0\Psi_{0,0}^{d,(0)}(x,y).$$

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

(7+5)/2 = 6, which is equal to $2 \cdot \Psi_{0,0}^{v,(0)}(x,y) = 2 \cdot -4$. The value 0 in the bottom-right corner is the change in the original data 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 $0 \cdot \Psi_{0,0}^{d,(0)}(x,y)$. The decomposition operation can be represented in terms of matrices.

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

3.3 Blur Detection Algorithm

The first stage of the blur detection algorithm is to find square tiles (windows) that have pronounced horizontal, vertical, or diagonal changes. A captured frame is divided into *N* x *N* windows, called tiles, where $N = 2^k, k \in Z$. Figure 3.1 shows square tiles of size $N = 64$. The border pixels at the right and bottom margins are discarded when captured frames are not evenly divisible by *N*. A candidate tile must have a pronounced change along at least one of the three directions: horizontal, vertical, or diagonal. Whether a change is pronounced or not is determined through a threshold.

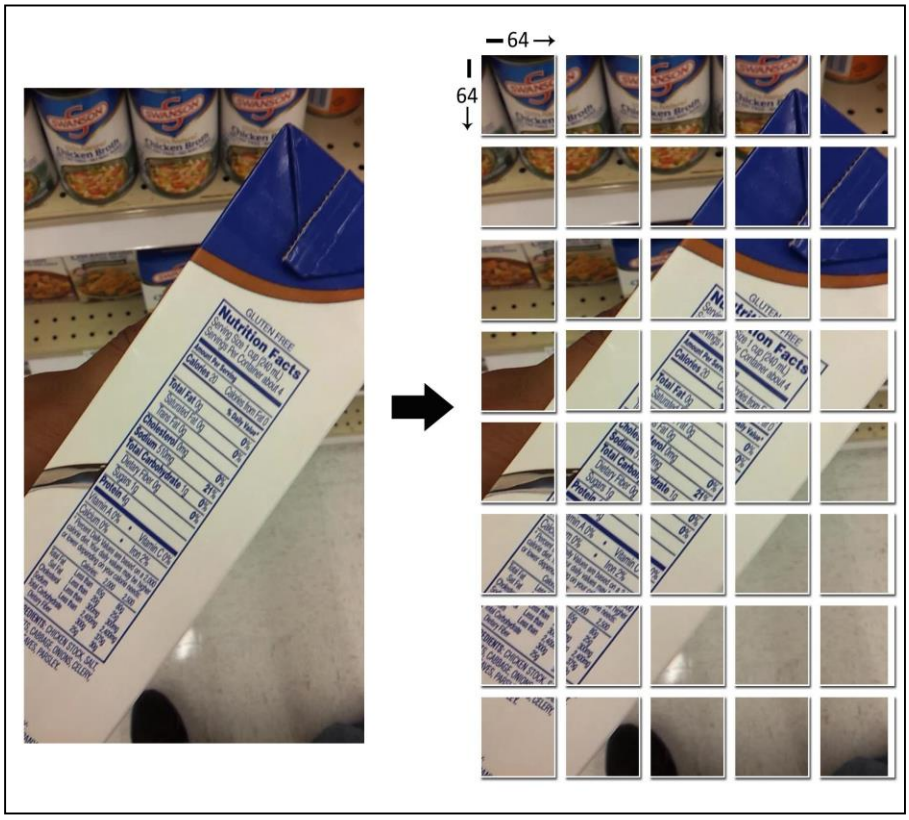


Figure 3.1 Tile splitting

Each tile is processed by two iterations of the 2D HWT [22]. Figure 3.2 shows two 2D HWT iterations applied to the image shown in the upper left corner. The number of iterations and the tile size are parameters and can be made either smaller or larger. The values reported in this paper were experimentally found to work well in our domain of vision-based skewed barcode scanning and nutrition information extraction [7, 8, 9].

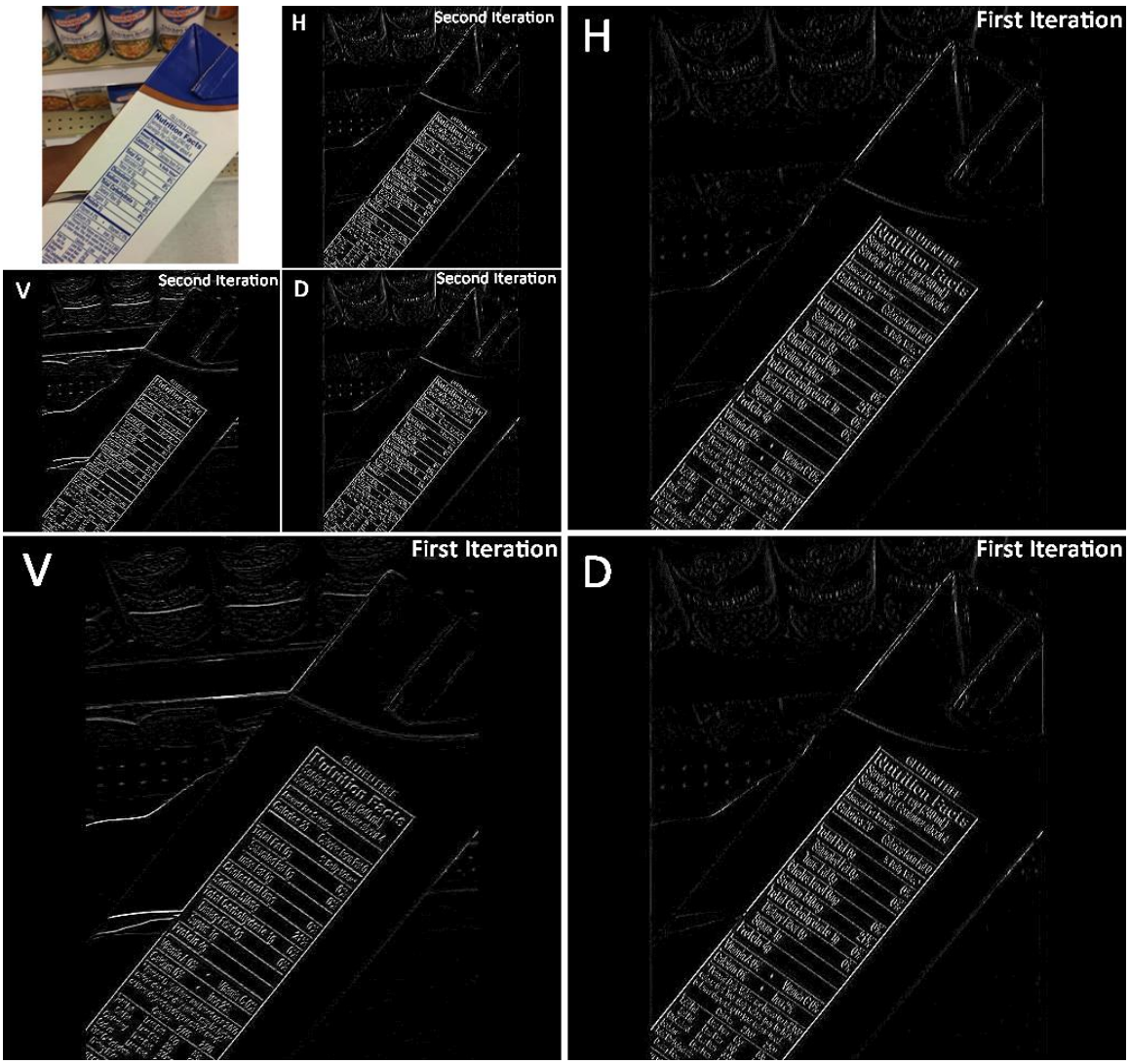


Figure 3.2 Two Iterations of 2D HWT

Let *HC* be the horizontal change between the left half of the tile and the right half of the tile. Let *VC* be the vertical change between the upper half of the tile and the lower half of the tile. Let *DC* be the change between the first diagonal (top left to bottom right) and the second diagonal (top right to bottom left). If at least one of the values *HC*, *VC*, and *DC* is above the corresponding thresholds HC_θ, VC_θ, and DC_θ, respectively, the tile

is marked as having a pronounced change. Figure 3.3 (left) shows the tiles with pronounced changes marked as squares.

After the tiles with pronounced changes are found, the depth-first search (DFS) is used to combine them into tile clusters. The DFS starts with an unmarked tile with a pronounced horizontal, vertical, or diagonal change and proceeds by connecting to its immediate horizontal, vertical, and diagonal tile neighbors if they have pronounced changes. If such tiles are found, they are marked with the same cluster number and the search continues recursively. The search stops when no other tiles can be added to the current cluster. The algorithm then continues to look for another unmarked tile to which it can apply the DFS. If no such tile is found, the algorithm terminates. Figure 3.3 (right) shows five DFS-found tile clusters.

After the tile clusters are found, two cluster-related rules are used to classify a whole image as sharp or blurred. The first rule is the percentage of the total area of the image covered by the clusters. The second rule uses the number of the tiles in each cluster to discard small clusters.

The first rule captures the intuition that in a sharp image there are many tiles with pronounced changes, as shown in Figure 3.3 (right). The second rule discards small clusters whose size, i.e., the number of tiles in the cluster, is below a given threshold.

Figure 3.3 Square tiles with pronounced changes (left); Tile clusters found by DFS (right)

The second rule can be viewed as cluster weeding. In the current implementation, this rule has a threshold of five. In other words, any cluster with fewer than five tiles is rejected by the second rule and does not contribute to the area of the image with pronounced changes. Thus, in Figure 3.3 (right), only two clusters are left after the application of the second rule: the cluster with 18 tiles and the cluster with 6 tiles. Since both clusters are large, the image is classified as sharp by the first rule. The three singletons are discarded. On the other hand, all clusters found in the blurred image of Figure 3.4 and shown in Figure 3.5 are small, and are discarded by the second rule. Therefore, the image is classified as blurred.

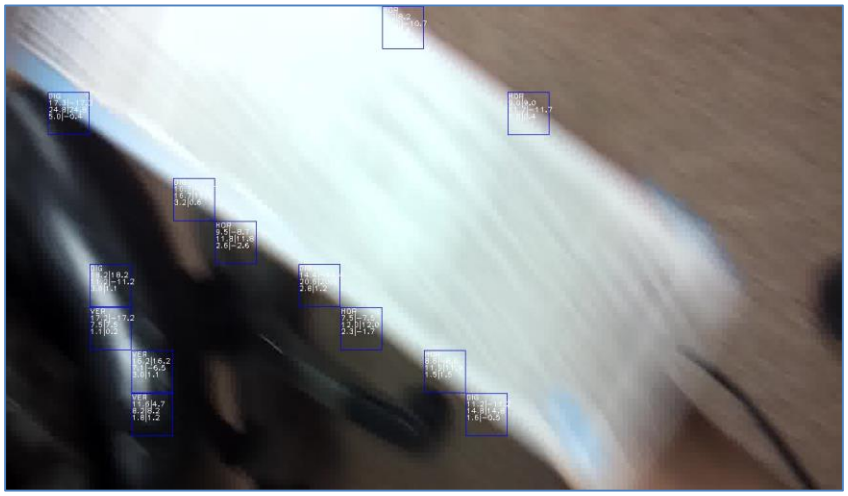


Figure 3.4 Tiles with pronounced changes in a blurred image

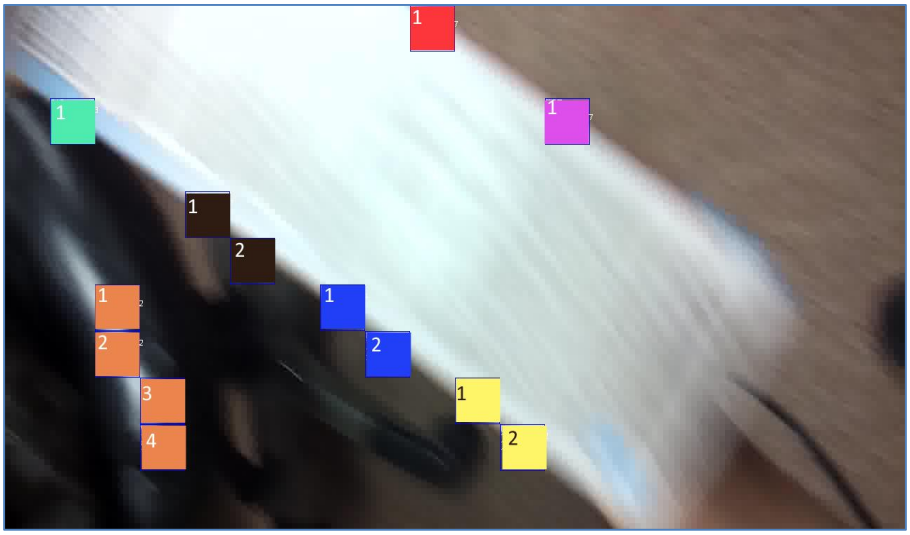


Figure 3.5 DFS-found tile clusters

Figure 3.6 gives the pseudo code of our blur detection algorithm. The first argument to the **DetectBlur** function is the image, the second argument is the size of the square tile into which the image is split, as shown in Figure 3.1. In our experiments presented in the next section, *N*=64. The next argument, *NITER*, is the number of iterations of the **2DHWT** function runs in each square tile in line 4. In our current implementation, *NITER*=4. Our Java source of the **2DHWT** function is available at [16].

This function returns an array of four matrices: *AVRG*, *HC*, *VC*, and *DC*. The matrix *AVRG* contains the average numbers after all iterations and the matrices *HC*, *VC*, and *DC* contain the horizontal, vertical, and diagonal wavelet coefficients, respectively.

```
1. DetectBlur(Img, N, NITER, HC_θ, VC_θ, DC_θ, CSZ, A)
2. {
3.   FOR each N x N tile T in image Img {
4.       [AVRG, HC, VC, DC] = 2DHWT(T, NITER);
5.       AH=Avrg(HC); AV=Avrg(VC); AD=Avrg(DC);
6.       IF (AH > HC_θ or AV > VC_θ or AD > DC_θ)
7.          Mark T as having pronounced change;
8.   }
9.   FOR each N x N tile T in image Img {
10.       IF ( T is not in any tile cluster )
11.         Run DFS(T) to find and mark all tiles in the
12.         same cluster;
13.  }
14.  TotalArea = 0;
15.  FOR each tile cluster TC {
16.      IF ( TC's size  >  CSZ ) {
17.          ClusterArea = TC's size * N x N;
18.          TotalArea += ClusterArea;
19.      }
20.  }
21.  IF ( TotalArea/Area(Img)  < A )  Return True;
22.  ELSE Return False;
23. }
```

Figure 3.6 pseudo code of the algorithm

. If at least one of the averages of the *HC*, *VC*, or *DC* matrices after all *NITER* iterations is above a corresponding threshold, which is computed in lines 5 and 6, then the appropriate tile is marked as having pronounced change. In lines 9-13, the DFS is used to find all tile clusters in the image, as shown in Figure 3.3 (right).

In lines 14-20, the two rules described above to compute the total area occupied by the clusters greater than the value of the cluster threshold parameter *CSZ*. In lines 21-

22, if the percentage of the total area occupied by the clusters is smaller than the threshold value specified by the last parameter *A*, the image is classified as blurred. If need be, the algorithm can be modified to return the blur extent as the ratio of the total area occupied by the large tile clusters and the total area of the image. The smaller the ratio, the more blur exists in the image.

The outlined algorithm is based on the assumption that in blurred images square tiles with pronounced changes do not form large clusters but scatter across the image as singletons or form clusters whose combined area is small relative to the size of the image.

For example, Figure 3.4 is a blurred image where 64 x 64 tiles with pronounced changes are marked. Figure 3.5 shows the tile clusters found by the DFS. As can be seen in Figure 3.5, most clusters are either singletons or are of size 2. The largest cluster in the bottom left of the image is of size 4.

CHAPTER 4

EXPERIMENTS

4.1 Overview

This chapter describes the experiments we conducted to compare our algorithm with two other blur detection algorithms frequently cited in the literature. A sample of five hundred images that contained both sharp and blur images was taken on a mobile phone. The ground truth was obtained by three human evaluators who classified each image as blurred or sharp, as explained in the next section. This human evaluation resulted in 167 images classified as blurred and 333 classified as sharp [23]. All three algorithms were run on these images and compared with the human classified results to verify the accuracy of each algorithm.

4.2 Experimental Procedure

We took five hundred random RGB images from a set of 506 video recordings of common grocery products that we made publicly available in our previous field investigations of skewed barcode scanning [24]. The videos have a 1280 x 720 resolution and an average duration of 15 seconds. All videos were recorded on an Android 4.2.2 Galaxy Nexus smartphone in a supermarket in Logan, UT. All videos were taken by an

operator who held a grocery product in one hand and a smartphone in the other. The videos covered four different categories of products: bags, boxes, bottles, and cans.

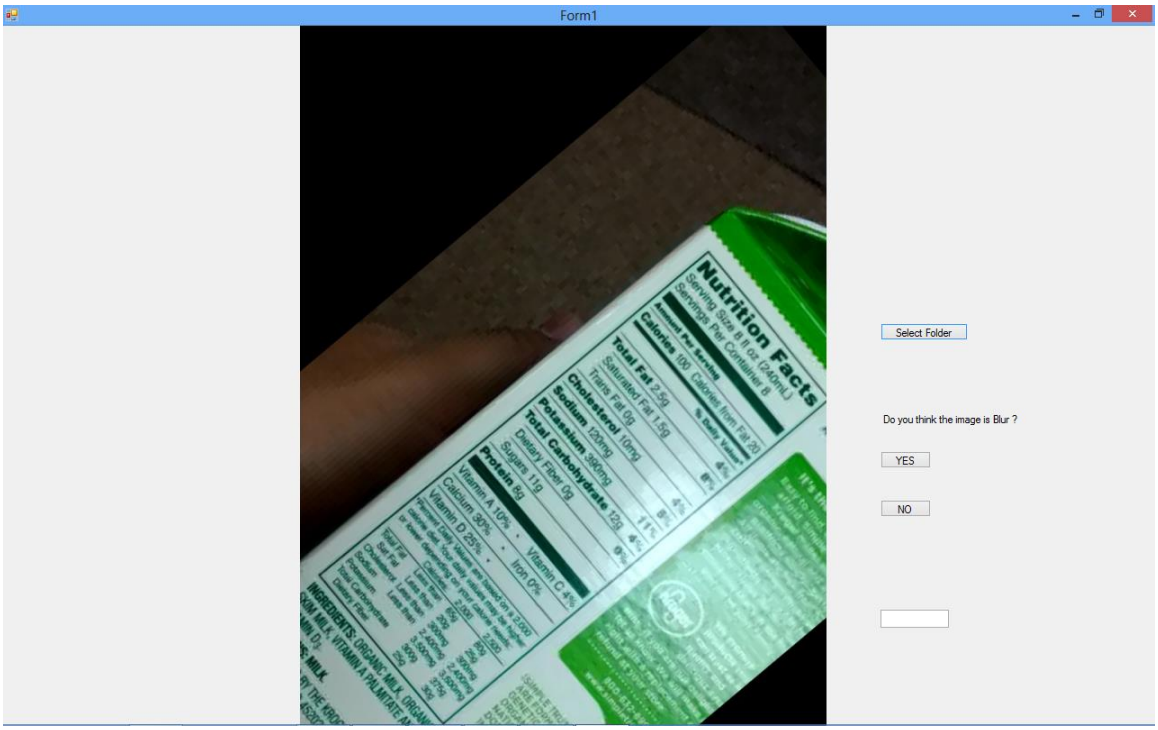


Figure 4.1 Experimental setup for volunteers to classify image as blur or sharp

Three human volunteers were recruited to classify each of the five hundred images as blurred or sharp. Figure 4.1 shows the tool used by the volunteers to classify an image into either blur or sharp. The above shown tool uses a simple interface, image is displayed and the volunteer clicks the button YES, if he/she thinks that the image is blur or clicks the button NO, if he/she thinks the image is sharp. Clicking either of the buttons brings up a new image. This process is repeated till the end of five hundred images. Three volunteers classified each of five hundred images as blurred or sharp. An image was classified as blurred if at least two volunteers classified it as blurred. It was otherwise

classified as sharp. The human evaluation resulted in 167 blurred images and 333 sharp images. These results were used as the ground truth.

We compared our algorithm with two other image blur detection algorithms frequently cited in the literature [13, 25]. Since we could not find the source code of [13] publicly available online, we implemented it ourselves in Python. Our Python source code is available at [26]. We found a MATLAB implementation of the other algorithm at [27] and used it for the experiments.

Table 4.1 gives the numbers of true and false positives for all three algorithms. The columns TPB and FPB give the numbers of true and false positives, respectively, for the blurred images. The columns TPS and FPS give the numbers of true and false positives, respectively, for the sharp images. The row *Algorithm 1* gives the statistics for our algorithm implemented in Java. The rows *Algorithm 2* and *Algorithm 3* give the statistics for the Python implementation of [13] and the MATLAB implementation of [25], respectively.

Table 4.1 True and false positives

Algorithm	TPB	FPB	TPS	FPS
Algorithm 1	163	4	254	79
Algorithm 2	167	0	183	150
Algorithm 3	81	86	268	65

To compare the performance numbers of each algorithm with the ground truth, we used the relative difference percentage, which is a unit less measure that compares two quantities while taking into account their magnitudes. Table 4.2 gives the relative

difference percentages computed as $|x-y|/\max(|x|,|y|)\cdot 100$, where y is the humanly estimated number of blurred or sharp images, i.e., the ground truth, and x is the number of sharp or blurred images found by a given algorithm. For example, for *Algorithm 1*, the first relative difference is computed as $|163-167|/\max(|163|,|167|)\cdot 100=2.39$, where 163 is the number of blurred images found by *Algorithm 1* and 167 is the number of blurred images found by the human evaluators.

Table 4.2 Relative differences

Algorithm	Blurred	Sharp
Algorithm 1	2.39	23.72
Algorithm 2	0.00	45.05
Algorithm 3	51.50	19.52

Table 4.3 Effect of blur on barcode scanning I

Sample	Sharp	Blurred	Barcode in sharp	Barcodes in blurred
1	15	15	12	1
2	13	17	11	0
3	16	14	12	0

4.3 Effect of Image Blur on Skewed Barcode Scanning

We investigated the effect of image blur on skewed barcode scanning. We chose three random samples of 30 images from the same set of 500 images classified by the three human evaluators. In each sample, 15 images were classified as blurred and 15 as sharp. We integrated our blur detection algorithm into our cloud-based barcode scanning

algorithm and estimated the effect of accurate image blur detection on skewed barcode scanning. Tables 4.3 and 4.4 give the results of our experiments.

In Table 4.3, the first column gives the sample numbers. The column *Sharp* gives the number of sharp images classified as sharp by our algorithm. The column *Blurred* gives the number of images classified as blurred by our algorithm. The column *Barcode in sharp* gives the number of barcodes correctly scanned in the images classified as sharp. The column *Barcodes in blurred* gives the number of barcodes correctly scanned in the images classified by our algorithm as blurred. Thus, in sample 1, all blurred and sharp images were classified accurately. However, in the 15 sharp images, the barcode scanner accurately scanned 12 barcodes whereas in the 15 blurred images, the barcode scanner accurately scanned only 1 barcode.

In sample 2, 13 out of 15 images were accurately classified as sharp with 2 false negatives and 17 images were classified as blurred with 2 false positives. In 11 images classified as sharp, the barcodes were accurately scanned. No barcodes were accurately scanned in the images classified as blurred.

In sample 3, 16 images were classified as sharp with 1 false positive and 14 images were classified as blurred with 1 false negative. Barcodes were successfully scanned in 12 images classified as sharp. No barcodes were scanned in the images classified as blurred.

Table 4.4 Effect of blur on barcode scanning II

Sample	Blurred	Sharp	Total	Gain
1	1/15	12/15	13/30	0.37
2	0/17	11/13	11/30	0.50
3	0/16	12/14	12/30	0.46

Table 4.4 gives the results of the effect of blur detection on skewed barcode scanning. The first column gives the numbers of the random samples. The second column records the ratio of accurately scanned barcodes in the images classified as blurred. The third column records the ratio of accurately scanned barcodes in the images classified as sharp. The fourth column gives the ratio of recognized barcodes in all images. The fifth column gives the gain measured as the difference between the ratio of the accurately recognized barcodes only in the sharp images and the ratio of the accurately recognized barcodes in all images, which estimates the effect of blur detection on barcode scanning. Thus, in sample 1, we increase the barcode scanning rate by 37 percent if we eliminate images classified as blurred from barcode scanning. In sample 2, if blurred images are eliminated from barcode scanning, we gain 50 percent in barcode scanning rates. In sample 3, the gain is 46 percent.

4.4 Effect of Image Blur On OCR Rates

We investigated the effect of image blur on OCR rates. We chose three random samples of 16 images classified by the three human evaluators. In each sample, 8 images were classified as blurred and 8 as sharp. We integrated our blur detection algorithm into open source OCR engine, Tesseract and estimated the effect of accurate image blur

detection on OCR rates. Figure 4.2 shows a sharp NL image on the left and the Tesseract output on the right. Figure 4.3 shows a blurred NL image on the left and the Tesseract output on the right. Tables 4.5 give the results of our experiments.

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

Amount Per Serving

Calories 120 Calories from Fat 25

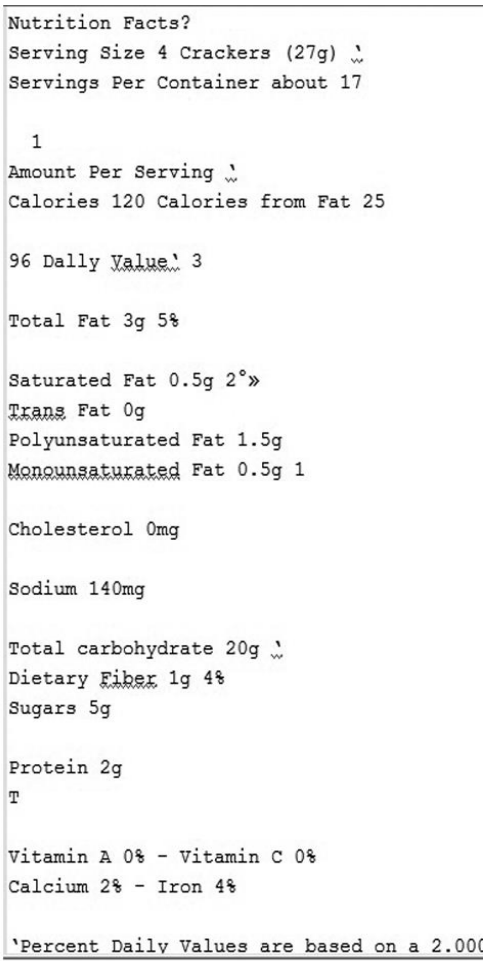
% Daily Value*

Total Fat 3g **5%**
Saturated Fat 0.5g **2%**
Trans Fat 0g
Polyunsaturated Fat 1.5g
Monounsaturated Fat 0.5g
Cholesterol 0mg **0%**
Sodium 140mg **6%**
Total Carbohydrate 20g **7%**
Dietary Fiber 1g **4%**
Sugars 5g
Protein 2g

Vitamin A 0% • Vitamin C 0%
Calcium 2% • Iron 4%

*Percent Daily Values are based on a 2,000 calorie diet. Your daily values may be higher or lower depending on your calorie needs.

	Calories:	2,000	2,500
Total Fat	Less than	65g	80g
Saturated Fat	Less than	20g	25g
Cholesterol	Less than	300mg	300mg
Sodium	Less than	2,400mg	2,400mg
Total Carbohydrate		300g	375g
Dietary Fiber		25g	30g



```
Nutrition Facts?
Serving Size 4 Crackers (27g) ‘
Servings Per Container about 17

  1
Amount Per Serving ‘
Calories 120 Calories from Fat 25

96 Dally Value‘ 3

Total Fat 3g 5%

Saturated Fat 0.5g 2°»
Trans Fat 0g
Polyunsaturated Fat 1.5g
Monounsaturated Fat 0.5g 1

Cholesterol 0mg

Sodium 140mg

Total carbohydrate 20g ‘
Dietary Fiber 1g 4%
Sugars 5g

Protein 2g
T

Vitamin A 0% - Vitamin C 0%
Calcium 2% - Iron 4%

‘Percent Daily Values are based on a 2.00(
```

Figure 4.2 OCR Detection for sharp images by Tesseract.

Table 4.5 gives the results of the effect of blur detection on OCR rates. The first column gives the numbers of the random samples. The second column records the ratio of accurately recognized words in the images classified as sharp. The third column records the ratio of accurately recognized words in the images classified as blurred. The fourth column gives the ratio of recognized words in all images.

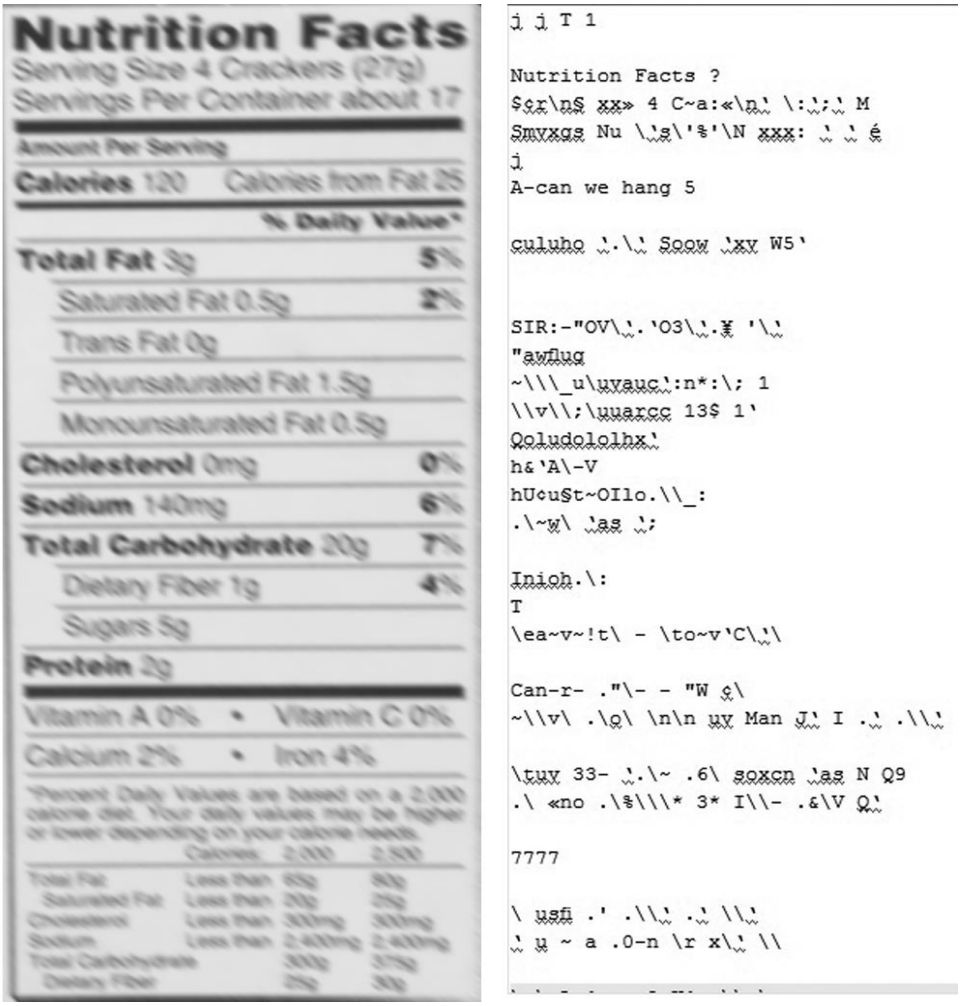


Figure 4.3 OCR Detection for blur images by Tesseract.

The fifth column gives the gain measured as the difference between the ratio of the accurately recognized words only in the sharp images and the ratio of the accurately recognized words in all images, which estimates the effect of blur detection on OCR rates.

In sample 1, we increase the OCR rate by 37 percent if we eliminate images classified as blurred from barcode scanning.

Table 4.5 Effect of blur on OCR rates.

Sample	Sharp Image	Blur Image	Total	Gain
1	258/267	21/267	279/534	0.45
2	218/288	8/288	226/576	0.39
3	213/321	9/321	222/642	0.32

In sample 2, if blurred images are eliminated from OCR scanning, we gain 50 percent in OCR rates. In sample 3, the gain is 46 percent.

CHAPTER 5

RESULTS

In discussing the results of the experiments, we will again refer to our algorithm as *Algorithm 1*, to the algorithm by Tong et al. [13] as *Algorithm 2*, and to the algorithm by [25] as *Algorithm 3*. The experiments indicate (see Table 4.1) that, on our sample of images, in classifying images as blurred. *Algorithm 1* performs as well as Algorithm 2 and outperforms *Algorithm 3*. In classifying images as sharp, *Algorithm 1* performs as well as *Algorithm 3* and outperforms *Algorithm 2*.

Table 4.2 confirms the observations recorded in Table 4.1. In image blur detection, there is almost no difference between *Algorithm 1* and *Algorithm 2* in that these algorithms do not deviate from the ground truth provided by the human evaluators on blurred images. On the other hand, *Algorithm 3* shows a significant deviation from the ground truth on blurred images. On the other hand, in classifying images as sharp, *Algorithm 1* and *Algorithm 2* deviate from the ground truth by approximately 20 points while *Algorithm 2* deviates from the ground truth by 45 points.

Tables 4.3 and 4.4 indicate that image blur detection has a pronounced positive effect on skewed barcode scanning. In all three random samples, the barcoding recognition gain was above thirty percent. While we ran these experiments only with our

barcode scanning algorithm, we expect similar gains with other vision-based barcode scanning algorithms.

Tables 4.5 indicates, as should be expected, that image blur detection has a pronounced positive effect on OCR rates. In all three random samples, the barcoding recognition gain was above thirty percent. While we ran these experiments only with open source OCR engine, Tesseract, we expect similar gains with other OCR engines.

Our experiments indicate that direct methods provide a viable alternative to indirect methods. While indirect methods may be more accurate, they tend to be more computationally expensive due to complex matrix manipulations. Direct methods may not be as precise as their indirect counterparts. However, they compensate for it by increased efficiency, which makes them more suitable for mobile and wearable platforms.

Another observation that we would like to make is that in working with our samples of images we could not observe the blur effect on the edges observed by Tong et al. [13] in some images. The edge blur effect observed by these researchers is that the injection of blur in the images causes the Dirac and A-Step edges disappear or turn into Roof and G-Step edges, respectively and the G-Step and Roof edges to lose their sharpness.

Instead of using the 2D HWT to detect edge types, the algorithm proposed in this paper is based on the assumption that in blurred images square tiles with pronounced changes do not form larger clusters but scatter across the image as singletons or form clusters that are small in size relative to the overall size of the image.

Our approach is theoretically rooted in the research by Mallat and Hwang [14] who show that the 2D HWT can detect the location of irregularities in 2D images. In our algorithm, the 2D HWT is used to detect the location of changes via square tiles without explicitly identifying the causes of the detected changes, e.g., edges or corners.

CHAPTER 6

CONCLUSIONS

We have presented an algorithm for direct image blur detection with the 2D Haar Wavelet transform (2D HWT). The algorithm classifies an image as blurred or sharp by splitting it into N x N tiles, applying four iterations of the 2D HWT to each tile, and grouping the horizontally, vertically, and diagonally connected tiles with pronounced changes into tile clusters. Images with large tile clusters are classified as sharp. Images with small tile clusters are classified as blurred. If necessary, the blur extent can be estimated as the ratio of the total area of the large tile clusters and the area of the whole image.

Our experiments on a sample of 500 images indicate that our algorithm either performs on par or outperforms two other blur detection algorithms found in the literature. The experiments also indicate that image blur detection has a pronounced positive effect on skewed barcode scanning, OCR rates. One possible implication of the research presented in this paper is that it may be possible to estimate blurriness in the images without explicitly computing the explicit features in the image that caused the blurriness (e.g., edges) or using involved methods to find the best fitting blur function.

In our future work, we plan to investigate the effect of parallel processing when input images are divided into rows that are processed concurrently. In our previous work,

we proposed to a greedy spellchecking algorithm to correct OCR errors during nutrition label scanning on smartphones [6]. The proposed algorithm, called *skip trie matching*, uses a dictionary of strings stored in the trie data structure to correct run-time OCR errors by skipping misrecognized characters while going down specific trie paths.

REFERENCES

[1] “Nutrition labeling and education action of 1990” [Online]. Available: http://en.Wikipedia.org/wiki/Nutrition_Labeling_and_Education_Act_of_1990

[2] “Food labelling to advance better education for life” [Online]. Available: www.flabel.org/en

[3] Graham, D. J., Orquin, J. L., and Visshers, V. H. M. "Eye tracking and nutritional label use: a review of the literature and recommendations for label enhancement." *Food Policy,* vol. 32, pp. 378-382, 2012.

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

[5] Kulyukin, V., Zaman, T., and Andhavarapu, S. “Effective Use of Nutrition Labels on Smartphones,” *in 15th Int. Conf. Internet Computing and Big Data* (ICOMP 2014), July 21-24, 2014, Las Vegas, NV, USA, CSREA Press, pp. 93 – 99.

[6] Kulyukin, V., Vanka, A., Wang, W. “Skip trie matching: a greedy algorithm for real-time OCR error correction on smartphones.” *Int. J. Digital Information and Wireless Commun.,* vol. 3, no. 3, pp. 56-65, 2013.

[7] Kulyukin, V. and Zaman, T. “Vision-based localization and scanning of 1D UPC and EAN barcodes with relaxed pitch, roll, and yaw camera alignment constraints.” *Int. J. of Image Proc.*, vol. 8, no. 5, 2014, pp. 355-383.

[8] Kulyukin, V. and Zaman, T. “An Algorithm for in-place vision-based skewed 1D barcode scanning in the cloud,” *in Proc. 18th Int. Conf. Image Processing and Pattern Recognition*, July 21-24, Las Vegas, NV, USA, CSREA Press, pp. 36-42.

[9] Kulyukin, V. and Blay, C. “An Algoritm for mobile vision-based localization of skewed nutrition labels that maximizes specificity,” *in Proc. 18th Int. Conf. Image Processing and Pattern Recognition*, July 21-24, 2014, Las Vegas, NV, USA, CSREA Press, pp. 3-9.

[10] Canny, J.F. “A Computational approach to edge detection.” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 8, 1986, pp. 679-698.

[11] Laganiere, R. *OpenCV 2 Computer Vision Application Programming Cookbook*, Packt Publishing Ltd, 2011.

[12] Nievergelt, Y. *Wavelets Made Easy*, Boston, 2000.

[13] Tong, H., Li, M., Zhang, H., and Zhang, C. "Blur detection for digital images using wavelet transform," *in Proc. IEEE Int. Conf. on Multimedia and Expo*, June 2004, vol.1, pp. 27-30, doi: 10.1109/ICME.2004.1394114.

[14] Mallat, S. and Hwang, W. L. “Singularity detection and processing with wavelets.” *IEEE Trans. Information Theory*, vol. 38, no. 2, March 1992, pp. 617-643.

[15] Smith, J.O. *Mathematics of the Discrete Fourier Transform with Audio Applications*, 2nd ed., W3K Publishing, 2007.

[16] Rooms, F., Pizurica, A., Philips, W. “Estimating image blur in the wavelet domain,” *in Proc. IEEE Int. Conf. on Acoustics and Signal Processing*, vol. 4, 2002, pp. 4190-4195.

[17] Wanqing, S., Qing, L., Yuming, W. “Tool wear detection using Lipschitz exponent and harmonic wavelet.” *Mathematical Problems in Engineering*, August 2013, Article ID 489261, [Online]. Available: http://dx.doi.org/10.1155/2013/489261.

[18] Venkatakrishnan, P., Sangeetha, S., and Sundar, M. “Measurement of Lipschitz exponent using wavelet transform modulus maxima.” *Int. J. of Scientific & Engineering Res.*, vol. 3, no. 6, pp. 1-4, June, 2012.

[19] Pavlovic, G., and Tekalp, M. “Maximum likelihood parametric blur identification based on a continuous spatial domain model.” *IEEE Trans. Image Processing*, vol. 1, no. 4, Oct. 1992, pp. 496-504.

[20] Panchapakesan, K., Sheppard, D.G., Marcellin, M.W., and Hunt, B.R. “Blur identification from vector quantizer encoder distortion,” *in Proc. 1998 Int. Conf. Image Processing*, Chicago, IL, pp. 751-755, 4-7 Oct. 1998.

[21] Marichal, X., Ma, W., and Zhang, H.J. “Blur determination in the compressed domain using DCT information,” *in Proc. IEEE Int. Conf. Image Processing*, Oct. 1999, vol. 2, pp. 386–390.

[22] “Java implementation of the 2DHWT procedure” [Online]. Available: https://github.com/VKEDCO/java/tree/master/haar.

[23] “Five hundred random RGB images from a set of 506 video recordings of common grocery products” [Online]. Available: https://app.box.com/s/n4s2ve0dajz5gkzqx9vpm1f6fzhw5upz

[24] “Mobile supermarket barcode videos of grocery packages” [Online]. Available: https://www.dropbox.com/sh/q6u70wcg1luxwdh/LPtUBdwdY1.

[25] Cretea,F., Dolmierea, T., Ladreta, P., Nicolas, M. “The Blur effect: perception and estimation with a new no-reference perceptual blur metric,” *in Proc. SPIE 6492, Human Vision and Electronic Imaging XII*, 64920I, February 12, 2007, doi:10.1117/12.702790.

[26] “Python implementation of the blur detection algorithm proposed in reference [13]” [Online]. Available: https://github.com/VKEDCO/PYPL/blob/master/haar_blur.

[27] “MATLAB implementation of blur detection algorithm proposed in reference [8]” [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/24676-image-blur-metric.ss