The Histogram of Partitioned Localized Image Textures

A Thesis

Presented to

The Faculty of the Graduate School

At the University of Missouri

In Partial Fulfillment

Of the Requirements for the Degree

Masters of Science

By

BREWSTER, ERIC B. (MU-STUDENT)

James M. Keller Ph.D., Thesis Supervisor

May 2017

The undersigned, appointed by the dean of the Graduate School,

have examined the Thesis entitled

THE HISTOGRAM OF PARTITIONED LOCALIZED IMAGE TEXTURES

Presented by Brewster, Eric B. (MU-Student)

A candidate for the degree of

Masters of Science

And hereby certify that, in their opinion, it is worthy of acceptance.

James M. Keller

Dominic K. Ho

Mihail Popescu

DEDICATION

I would like to dedicate this thesis to Violet Brewster a loving and supportive grandmother. Thank you for always being there when I needed it and believing I could accomplish anything I put my mind to. You will always live in my heart.

ACKNOWLEDGEMENTS

I would like to express my gratitude towards my advisor, Dr. James Keller, for encouraging me to further my education and attain a master's degree. He pushed me to begin looking outside the box and push the bounds of my knowledge to help me grow as a researcher and person. Without his support, this thesis would not be possible. I would also like to thank Dr. Dominic Ho and Dr. Mihail Popescu for their advice in the development of this new texture feature as well as their participation in the thesis committee. Furthermore, I would like to thank Andrew Buck and Pooparat Plodpradista for allowing me to bounce ideas off them as well as provide ideas to solve challenging design problems. Without them, the results may not have been so promising. Thank you to Night Vision for providing some of the date used in this thesis under the Army Research Office grant number 57940-EV to support the U. S. Army RDECOM CERDEC NVESD. Finally, I would like to give my sincere thanks to my friends and family for always believing in me and pushing me to do my best.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In the field of machine learning and pattern recognition, texture has been a prominent area of research. Humans are uniquely equipped to distinguish texture; however, computers are more equipped to automate the process. Computers accomplish this by taking images and extracting meaningful features that describe their texture. Some of these features are the Haralick texture features, local binary pattern (LBP), and the local direction pattern (LDP). Using the local directional pattern as an example, we propose a new texture feature called the histogram of partitioned localized image textures (HoPLIT). This feature utilizes a set of filters, not necessarily directional, and generates filter response vectors at every pixel location. These response vectors can be thought of as words in a document, which causes one to think of the bag-of-words model. Using the bag-of-words model, a codebook is created by partitioning a subset of response vectors from the entire data set. The partitions are represented by their mean texture and thus a word in the codebook. The mean textures now represent the keywords within the document, i.e. image. A histogram descriptor for an image is the frequency of pixels that belong to each partition. This feature is applied to a texture classification and segmentation problem as well as object detection. Within each problem domain, the HoPLIT feature is compared to the Haralick texture features, LBP, and LDP. The HoPLIT feature does very well classifying texture as well as segmenting large texture mosaics. HoPLIT also shows a surprising robustness to noise. Object detection proves to be slightly more difficult than texture classification for HoPLIT. However, it continues to outperform LBP and LDP.

CHAPTER 1 INTRODUCTION

A large amount of research has been done on extracting texture features from images like: man-made objects, nature, remote sensing images, etc. Texture features are used in object classification, image segmentation, facial recognition, and many more. For this thesis, the classification of images is the main goal; however, image segmentation is also investigated. Now, there are a multitude of different texture feature extraction methods out in the literature like first order statistics, gray-level co-occurrence matrix (GLCM) statistical features, local binary pattern (LBP), and local directional pattern (LDP) to name a few.

In 1973, Haralick *et al.* proposed the use of the GLCM to generate 14 statistic features that describe the texture of an image [1]. They calculate these statistics over a normalized gray-level occurrence matrix, which records the frequency of pairs of gray-levels that are near each other. Numerous experiments and papers over the years use these features to describe texture. Most new texture extraction techniques use these features as a baseline when comparing performance.

The local binary pattern (LBP) is a well-known set of texture features in the literature due to its overall simplicity. This method was first proposed by He and Wang in the 1990s [2]. He and Wang referred to the idea as a texture spectrum and utilized a non-binary encoding. In 1994, Ojala *et al*. described the binary version of the texture spectrum and called it the LBP [3].

Along the same lines as LBP, Jabid *et al.* proposed a new texture descriptor that also uses binary encoding called the local directional pattern (LDP) [4]. The major

difference between LBP and LDP is that LDP uses filters or masks to assist in generating the binary encoding. By using filters, Jabid *et al.* believed LDP is more descriptive of local texture and more resilient to noise [5]. Although LDP processes adds some complexity, its implementation is rather simple making it very appealing. Because of its simplicity, LDP has the potential for multiple modifications.

Modifications to LDP come in two forms: filter modifications, and encoding modifications. Modification to the filters can be as simple as changing the number of filters or using various preexisting filter sets. Rivera *et al.* proposed a LDP using Gaussian filters [6]. Both Ishraque *et al.* [7] and Higashi *et al.* [8] utilize Log Gabor filters. Modifications to the pattern encoding appear more often in the literature. Some modifications retain the binary coding; however, the method generating the binary coding changes or augments the code. Common methods that fall under this category are the LDP variance [9], local directional number pattern [10], modified LDP [11], and optimized LDP [12]. Another route removes the binary coding altogether and utilizes the directional responses directly. Algorithms that fall into this modification area include adaptive quantization of local directional responses pattern [13], histogram of weighted local directions [14], and improved LDP [15]. Some of the modifications mentioned above also incorporated an alternative method to generate the histogram that best suits the algorithm.

For this thesis, we propose our own modification of LDP called the histogram of partitioned localized image textures, or HoPLIT for short. This algorithm utilizes any two-dimensional filter set to generate response vectors for each pixel. We then use a clustering algorithm to partition the response vectors to generate code words. Finally, an image

becomes a quantization of key textures with each pixel belonging to the partition whose center is closest to the pixel's response vector.

Following this chapter, we describe the Haralick texture features, LBP and LDP algorithms as well as the bag-of-words model. Next, we explain the proposed HoPLIT algorithm. Chapters 4 and 5 implement HoPLIT to solve texture classification and object detection problems. Finally, we conclude the thesis as well as propose possible areas of improvement to the HoPLIT features.

CHAPTER 2 BACKGROUND

In this chapter, we explain each feature extraction method and demonstrate them with a simple example. The first method is the GLCM statistics proposed by Haralick *et al*. Next, we describe the implementation for LBP. An in-depth description of LDP follows. Finally, we provide a basic description of the bag-of-words model. Within each subsection, we provide a citation pointing to a paper or book that describes the implementation in more detail.

2.1. Haralick Texture Features

Haralick *et al.* suggest that the texture is contained in the mean spatial relationship between gray tones of each pixel to each other within an image. These relationships are recorded in the form of a matrix. Given an image we can represent the intensities into N_g quantized gray levels. Let $L_x = \{1, 2,\ldots, N_x\}$ represent the horizontal domain, $L_y = \{1, 2,\ldots, N_y\}$ represent the vertical domain, and $G = \{1, 2,\ldots, N_g\}$ represent the quantized gray levels where N_x and N_y are the horizontal and vertical resolutions respectively. The image can be represented by the set $L_y \times L_x$ indicating the rows and columns of the image. Each pixel is assigned some gray level in G [1].

Around each pixel, there are eight adjacent neighbors representing four angles. Figure 2.1 provides a visual representation of the neighbors and their angle $\theta = \{0, 45, 90, 135\}$. It is assumed that the texture information of the image is sufficiently specified by a matrix of relative frequencies $P_{i,j}$ with each two neighboring pixels separated by a distance d with gray levels i and j and angel θ. The frequencies are defined by

$$P(i,j,d,\theta) = \#\{((k,l),(m,n)) \in (L_y \times L_x) | f(k,l,m,n,d,\theta), I(k,l) = i, I(m,n) = j\} \quad (2.1)$$

$$f(k,l,m,n,d,\theta) = \begin{cases} (k-m,|l-n|) = (0,d) & if\ \theta = 0° \\ (k-m,l-n) = (d,-d),(-d,d) & if\ \theta = 45° \\ (|k-m|,l-n) = (d,0) & if\ \theta = 90° \\ (k-m,l-n) = (d,d),(-d,-d) & if\ \theta = 135° \end{cases} \quad (2.2)$$

where # denotes the number of elements in the set. Also the $P_{i,j}$ matrices are symmetric and called gray-level co-occurrence matrixes (GLCM). Figure 2.2 shows an example gray level image that has been quantized into three levels 0 to 2 and the calculated frequency matrices. In Figure 2.2, the GLCM are un-normalized; however, they are easily normalized by dividing each element by the sum of all elements in the matrix.

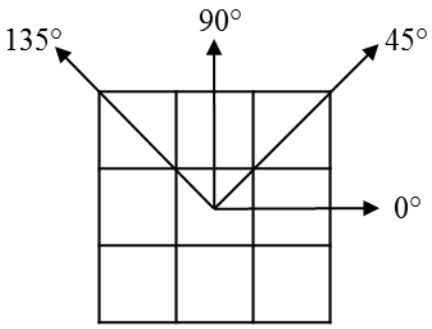

Figure 2.1: A pixel and its eight nearest neighbors and their corresponding angles θ. The GLCM generated using angles plus 180 degrees produce equivalent matrices (i.e. 0 degrees and 180 degrees, 45 degrees and 215 degrees, and etc.).

1	1	2	0	0
0	1	2	2	2
0	1	2	2	0
0	1	1	1	0
2	2	1	1	0

a)

#(0,0)	#(0,1)	#(0,2)
#(1,0)	#(1,1)	#(1,2)
#(2,0)	#(2,1)	#(2,2)

b)

$$P(i,j,1,0°) = \begin{pmatrix} 2 & 5 & 2 \\ 5 & 8 & 4 \\ 2 & 4 & 8 \end{pmatrix} \qquad P(i,j,1,45°) = \begin{pmatrix} 0 & 5 & 1 \\ 5 & 2 & 6 \\ 1 & 6 & 4 \end{pmatrix}$$

$$P(i,j,1,90°) = \begin{pmatrix} 8 & 1 & 4 \\ 1 & 10 & 3 \\ 4 & 3 & 6 \end{pmatrix} \qquad P(i,j,1,135°) = \begin{pmatrix} 0 & 3 & 3 \\ 3 & 8 & 3 \\ 3 & 3 & 4 \end{pmatrix}$$

c)

Figure 2.2: a) A 5x5 example image with 3 gray-levels ranging from 0 to 2. The GLCM takes the form of b) where each element in the matrix is the number of occurrences of the various level pairings. The un-normalized GLCMs for the image in a) are computed in c) with a distance of one and all four angles shown in Figure 2.1.

Using the GLCM, several statistics are computed to describe the image's texture. Haralick *et al.* lists fourteen different statistics. However, for this thesis we choose four out of the fourteen because the Matlab function utilized only computes four. The four statistics are contrast, correlation, angular second moment, and homogeneity. Homogeneity is similar to the inverse difference moment and measures how close the distribution of GLCM elements are to the diagonal. The equations for these statistics are

$$Contrast = \sum_{i,j} |i-j|^2 P(i,j) \tag{2.3}$$

$$Correlation = \sum_{i,j} \frac{(i - \mu_i)(j - \mu_j)P(i,j)}{\sigma_i \sigma_j} \tag{2.4}$$

$$Angular\ Second\ Moment = \sum_{i,j} P(i,j)^2 \tag{2.5}$$

$$Homogeneity = \sum_{i,j} \frac{P(i,j)}{1 + |i - j|} \tag{2.6}$$

In each statistic, the GLCM has been normalized so that comparisons between different angles are more easily observed. Each statistic generates a single value and can be concatenated together to form a feature vector. To create a descriptor containing maximum information the feature vector should contain the statistics for every angle. The feature vector can be of the whole image or of a sub region within the image.

2.2. Local Binary Pattern (LBP)

As mentioned in the introduction, He and Wang first proposed a new statistical approach to analyzing texture called the texture spectrum. LBP is in fact a special case of the texture spectrum. Just like the Haralick texture features, LBP looks at the relation a pixel has with its nearest neighbors. We can represent the smallest neighborhood as the set of nine elements $V = \{V_0, V_1, \ldots, V_8\}$, where the first element is the intensity of the center pixel and V_i with $i = \{1, 2, \ldots, 8\}$ being the intensity of a neighboring pixel. Figure 2.3 shows the neighboring pixels order around the center pixel.

V_1	V_2	V_3
V_8	V_0	V_4
V_7	V_6	V_5

Figure 2.3: The clockwise ordering of the elements in the set V = {V_0, V_1,…, V_8} where V_0 indicates the intensity of the center pixel.

Each intensity located at V_i is compared to the center pixel's intensity through subtraction. The sign of the difference dictates the value in the final encoding. The encoding formula is as follows,

$$E_i = \begin{cases} 1, & if\ V_i \geq V_0 \\ 0, & if\ V_i < V_0 \end{cases} \tag{2.7}$$

The encoding, E_i, is then represented as an eight-bit binary number with the order depicted in Figure 2.3. The eight-bit number is then converted to an integer value using

$$N_{TU} = \sum_{i=1}^{8} E_i \times 2^{i-1} \tag{2.8}$$

where N_{TU} is the unit number assigned to the center pixel in the encoded image. Figure 2.4 shows a simple example transforming a pixel's neighborhood to an eight-bit binary number and then to an integer.

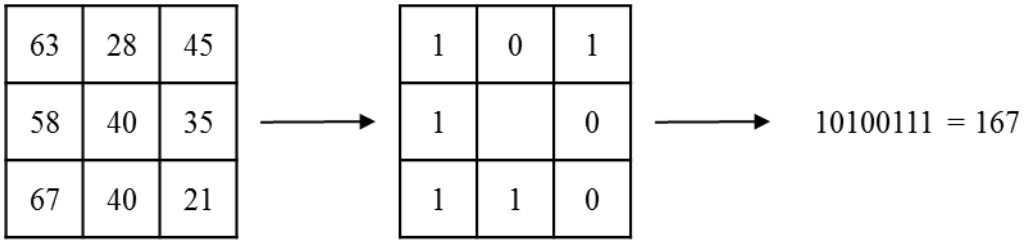

Figure 2.4: Simple example converting a neighborhood unit into a binary code and then into an integer that is used to represent the center pixel in the encoded image.

To generate the feature vector, the encoded image is converted into a histogram that shows the frequency of each encoded value. Typically, this histogram is normalized using the L2 norm. Like with GLCM, there is a distance *d* that can be set to look at a larger and more distant neighborhood. We choose to use the eight nearest neighbors, *d* equal to one, for this thesis.

2.3. Local Directional Pattern (LDP)

The local directional pattern is the most recently developed, compared to the other texture features described previously. This feature extraction method was developed in an attempt to improve performance in local texture based features over LBP. Although the two methods sound similar and accomplish the same task, there is minimal similarity in their implementation. Instead of comparing the intensity of neighboring pixels, LDP applies directional masks to the pixel of interest and its neighbors. These masks compute the directional responses and use them to encode an image's texture. The encoding of the directional responses takes the form of an eight-bit binary code where each pixel is assigned its own code. The code is generated by comparing the absolute directional responses values around the pixel. There are many different types of edge detectors in the literature like the Kirsch compass masks, used in [4] and [5], Figure 2.5.

$$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \quad \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

M_0 (East) M_1 (Northeast) M_2 (North) M_3 (Northwest)

$$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \quad \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

M_4 (West) M_5 (Southwest) M_6 (South) M_7 (Southeast)

Figure 2.5: Kirsch compass mask used to compute eight directional responses around a pixel.

A response for one mask it calculated by correlating the mask around the pixel. Mathematically the formula for this correlation is

$$m_i = \sum_{k=-1}^{1} \sum_{l=-1}^{1} M_i(k+1, l+1) \times I(x+k, y+l) \tag{2.9}$$

where the (x,y) is the location of a pixel in image I and m_i corresponds to the i-th directional response from mask M_i. the correlation shown in (2.9) is performed at every pixel and creates a directional response vector $m = (m_0, m_1, \ldots, m_7)$ for each pixel. Figure 2.6 shows an example of the Kirsch directional responses for the center pixel in the example image in Figure 2.4. As a note, for this thesis we chose to use image convolution over correlation, which is defined as

$$m_i = \sum_{k=-1}^{1} \sum_{l=-1}^{1} M_i(k+1, l+1) \times I(x-k, y-l) \tag{2.10}$$

The difference between correlation and convolution is a 180-degree rotation of the filter. For the Kirsch compass masks, this imposes a reordering of the location for each response in the binary code.

Once we compute all the directional responses, the absolute values of the responses are ranked from highest to lowest. The reason behind ranking the responses is to specify a significant response, m_k, as a reference. If one wants to make a comparison between LDP and LBP, then this significant response would represent V_0. Using the k-th reference, the LDP code is computed by

$$LDP_k = \sum_{i=0}^{7} b_i(m_i - m_k) \times 2^i \tag{2.11}$$

$$b_i(a) = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases} \tag{2.12}$$

where m_k is the absolute response value of the k-th ranked direction and m_i the i-th directional response. Also, note that each pixel has its own m_k depending on its local directional responses. Figure 2.6 also shows the binary code generated by selecting the third significant responses. Selecting the 8-th significant response as a reference serves no purpose because all codes then become 255. For this work, we will use the third significant response as our reference because it is the most commonly used in the literature.

Like LBP, LDP generates a feature vector of an image or sub region by building a histogram of the LDP codes. However, there is a unique method to determining the number of bins for LDP. The number of bins in the histogram is a function of k and is represented by C_k^8. In words this is eight choose k bins; which is calculated using the statistical combinations formula

$$C_k^n = \frac{n!}{k!\,(n-k)!} \tag{2.13}$$

where n is the number of directional masks and k is the significant rank. Since we will be using the third significant directional response as a reference, the number of bins in the histogram is 56. Let the input image I be of size $M \times N$, then the LDP histogram describing I can be calculated using (2.14) where C_i is the i-th LDP code value that falls into the i-th histogram bin. The histogram, $H = \left[H_1, H_2, \dots, H_{C_k^8}\right]$, is the descriptor for image I.

$$H_i = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} p\left(LDP_{(x,y)}, C_i\right) \tag{2.14}$$

$$p(t,a) = \begin{cases} 1 & if\ t = a \\ 0 & if\ t \neq a \end{cases} \tag{2.15}$$

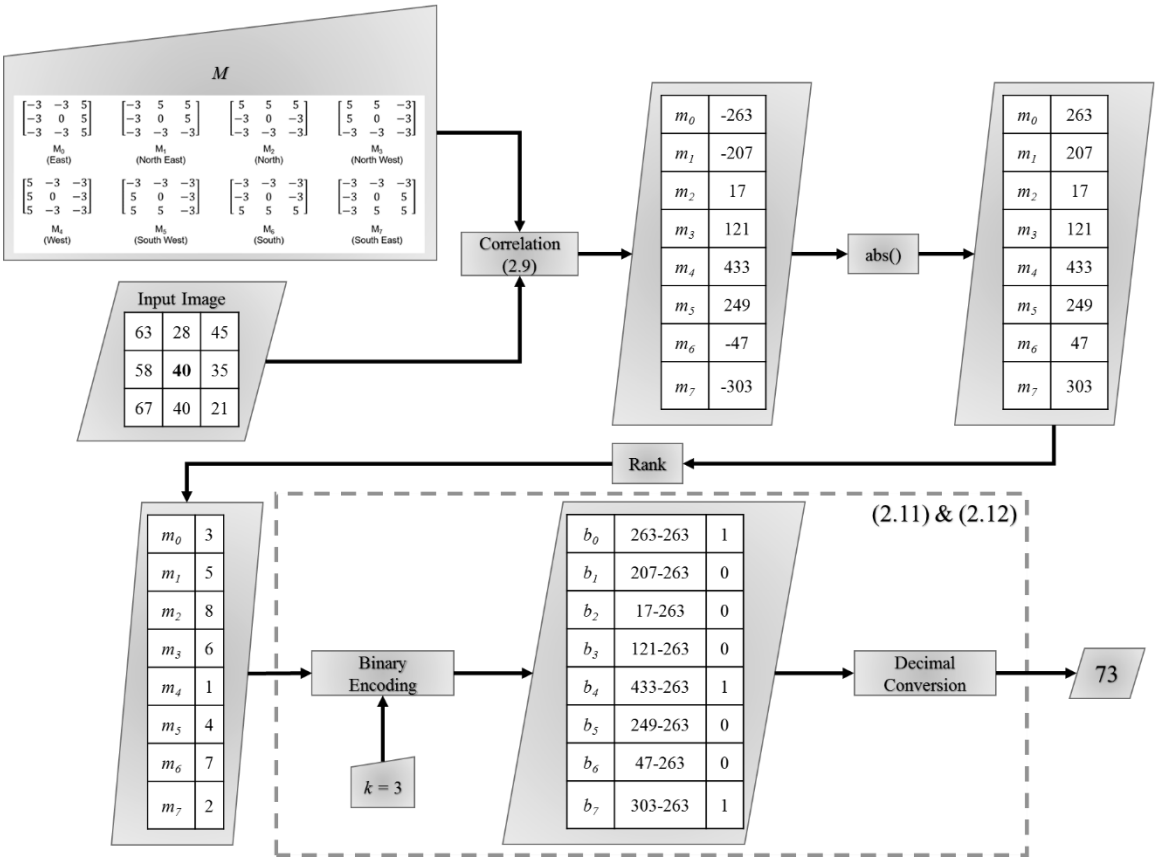

Figure 2.6: Diagram depicting the LDP process for the example image in Figure 2.4.

2.4 Bag-of-Words

In Chapter 3, we use a method called bag-of-words to produce histogram descriptors for images. We use an extremely simple implementation of this method; however, a more detailed account resides in [16]. The bag-of-words model, on its own, is very simple. Given a data set, for our purposes images, we extract features over all the images. These features can be small patches within each image around key points called textons [17], features extracted around key points like in SIFT [18], or they might be features extracted at every pixel. These low-level features may represent key components of the data set. To remain within the same realm of terminology let us call these components words.

An image is comprised of many words, but only a portion of these words help in identification. The smaller set of words are generated by clustering the words using a clustering or partitioning algorithm. K-means is probably the most common clustering algorithm found in the literature. We chose to use K-means over other clustering algorithms because we want to create crisp partitions in our data. The K-means algorithm assigns all data points to one of K clusters based on their distance to the cluster centers. The cluster centers generally represent the mean within each cluster. Algorithm 1 provides the steps required to implement K-means [19]. The cluster centers produced by K-means now represent our bag-of-words, or sometimes referred to as a codebook. Next, each data point in an image is assigned to their closest code word. Histograms are then created by counting the frequency each code word appears in the image. The histograms become the descriptors used for classification or segmentation.

Algorithm 1: K-means

Data set $\mathbb{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$
select K
initialize prototypes $\boldsymbol{\mu}_k$
while $\delta > convCriteria$ **or** $iter < maxIter$
  for $n = 1{:}N$
    for $j = 1{:}K$

$$r_{nk} = \begin{cases} 1 & if\ k = arg\min_j \left\|\boldsymbol{x}_n - \boldsymbol{\mu}_j\right\|^2 \\ 0 & otherwise \end{cases}$$

    end
  end
  for $j = 1{:}K$
    $\boldsymbol{\mu}_k = \sum_n r_{nk}\boldsymbol{x}_n / \sum_n r_{nk}$
  end
  $J_{new} = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$
  $\delta = \|J_{old} - J_{new}\|$
  $iter = iter + 1$
end

CHAPTER 3 THE HISTOGRAM OF PARTITIONED LOCALIZED IMAGE TEXTURES

After thoroughly examining the LDP implementation, the question of whether a binary patter is the best option arises. By converting to a binary pattern, the information regarding the degree of a filter's response is lost. The responses may hold vital information that helps better distinguish textures apart from each other. Instead of using a binary encoding, suppose we think of the image as a mixture or collection of various textures. Like LDP, let us encode the local texture information by applying localized filters around pixels and their neighbors as indicated by Figure 2.6. For our purposes, we will consider the collection of responses generated by the filters around a single pixel a texture. Some of the textures are similar and some are dissimilar. We can then partition the data into different texture zones.

Let $\mathbb{W} = \{\boldsymbol{I}_v\}$ with $v = 1,2, \dots V$ represent all the images in a data set. Each $\boldsymbol{I}_v \in \mathbb{R}^{X \times Y}$ and has $X \cdot Y = S$ pixels with each pixel having a location (x, y) within $\boldsymbol{I}_v$. We can represent the set of all pixels in $\boldsymbol{I}_v$ as $\mathbb{P}_v = \{p_1, p_2, \dots, p_S\}$ where p_i indicates the (x, y) pair corresponding to image index i.

Now let $\mathbb{M} = \{\boldsymbol{M}_1, \boldsymbol{M}_2, \dots, \boldsymbol{M}_n\}$ be a set of n two-dimensional filters of identical dimensions. Each $\boldsymbol{M}_j$ is then convolved with each p_i and its neighboring pixels. The resulting convolution creates a response vector $\boldsymbol{r}_i = [m_{i1}, m_{i2}, \dots, m_{in}]$ where

$$m_{ij} = \sum_{k=-1}^{1} \sum_{l=-1}^{1} \boldsymbol{M}_j(k+1, l+1) \times \boldsymbol{I}_v(x_i - k, y_i - l) \tag{3.1}$$

for $\boldsymbol{M}_j \in \mathbb{R}^{3\times3}$. Every $\boldsymbol{r}_i$ represents one set of possible response vectors that provide some insight into a local texture present in image $\boldsymbol{I}_v$. After all response vectors are calculated, we can construct a response matrix of $\boldsymbol{I}_v$,

$$\boldsymbol{R}^{(v)} = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \\ \vdots \\ \boldsymbol{r}_S \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ m_{S1} & m_{S2} & \dots & m_{Sn} \end{bmatrix} \tag{3.2}$$

By concatenating all $\boldsymbol{R}^{(v)}$ together we create a data matrix, $\boldsymbol{R}$, that contains the local textures contained in $\mathbb{W}$,

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}^{(1)} \\ \boldsymbol{R}^{(2)} \\ \vdots \\ \boldsymbol{R}^{(V)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{11} \\ \vdots \\ \boldsymbol{r}_{1S} \\ \boldsymbol{r}_{21} \\ \vdots \\ \boldsymbol{r}_{2S} \\ \vdots \\ \boldsymbol{r}_{V1} \\ \vdots \\ \boldsymbol{r}_{VS} \end{bmatrix} \tag{3.3}$$

From $\boldsymbol{R}$, we uniformly select a random percentage, α, of textures to use for codebook generation with $\boldsymbol{Q} \subset \boldsymbol{R}$ and $|\boldsymbol{Q}| = \alpha|\boldsymbol{R}|$. For all the experiments in this thesis, an α of 0.05 is used unless otherwise specified. Using $\boldsymbol{Q}$, we partition the texture space into K partitions using the K-means clustering algorithm described in Chapter 2. Each partition represents a texture zone. The K-means cluster center outputs,

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \\ \vdots \\ \boldsymbol{c}_K \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ m_{K1} & m_{K2} & \dots & m_{Kn} \end{bmatrix} \tag{3.4}$$

where $\boldsymbol{c}_k$ corresponds to the mean texture in zone k. The cluster centers assist in determining which zone a response vector belongs. Figure 3.1 provides a diagram demonstrating the codebook generation process.

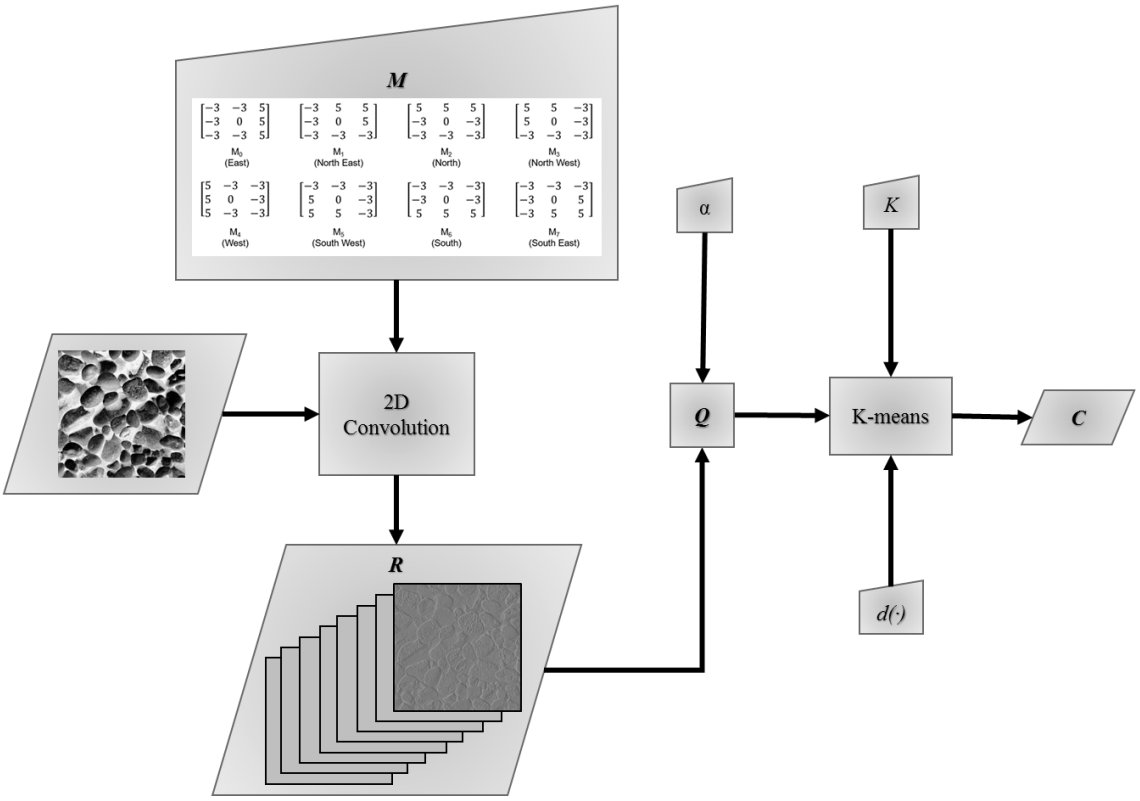

Figure 3.1: Diagram depicting the codebook generation process for a single 640x640 texture image using the Kirsch compass masks. The final output **C** is the partition centers with *K* partitions using distance formula $d(\cdot)$.

To extract features from image $\boldsymbol{I}$, we first compute $\boldsymbol{R}$. For each $\boldsymbol{r}_i \in \boldsymbol{R}$, we determine the closest $\boldsymbol{c}_k$ and record its index,

$$U_i = \underset{k}{\operatorname{argmin}} \{d(\boldsymbol{r}_i, \boldsymbol{c}_k)\}$$

$$\forall\, i = 1,2,\dots,S \; and \; k = 1,2,\dots,K. \tag{3.5}$$

where $d(\boldsymbol{r}_i, \boldsymbol{c}_k)$ is the distance function used to determine the closeness of $\boldsymbol{r}_i$ to $\boldsymbol{c}_k$ From the index vector $\boldsymbol{U}$, we generate a histogram descriptor H for $\boldsymbol{I}_v$, using

$$H_k = \sum_{i=1}^{S} b(U_i, k) \tag{3.6}$$

$$b(t,a) = \begin{cases} 1 & if\ t = a \\ 0 & if\ t \neq a \end{cases}. \tag{3.7}$$

Figure 3.2 provides a diagram demonstrating the processes used to build the descriptor. We call this new texture feature the histogram of partitioned localized image textures, or HoPLIT for short.

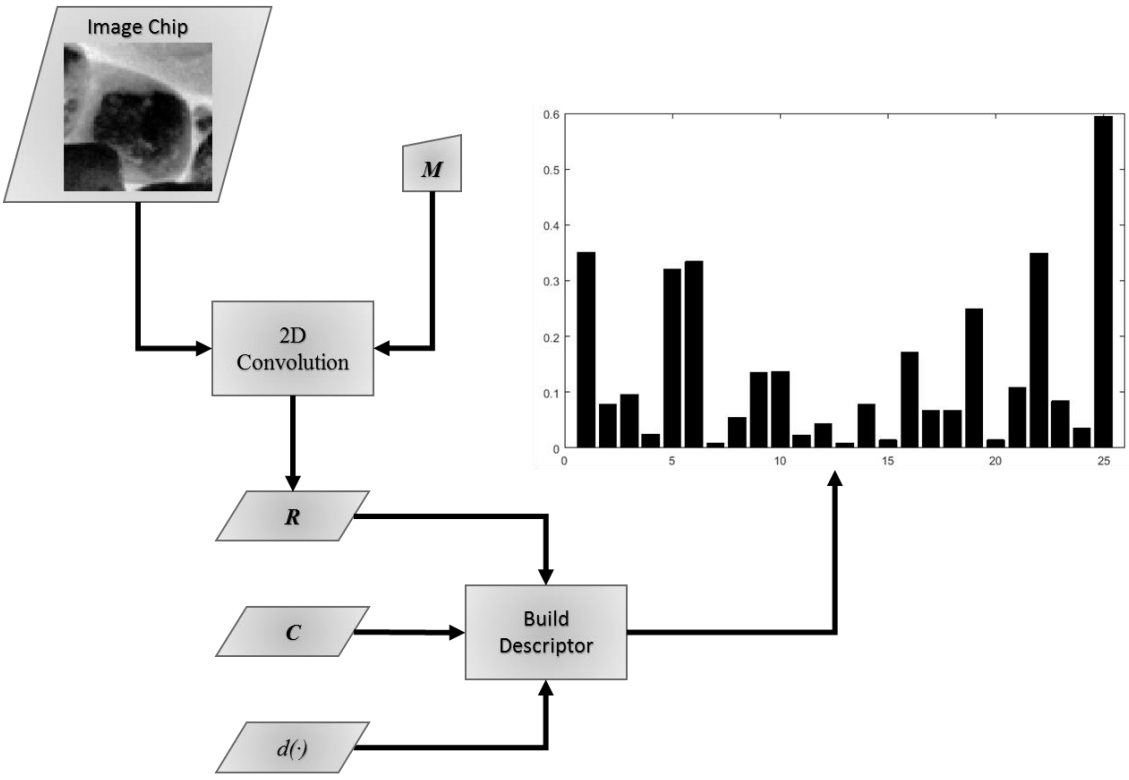

Figure 3.2: Diagram demonstrating the histogram generation process for a 64x64 image chip taken form the texture image in Figure 3.1. The partition centers were generated using 5% of the data in the 640x640 texture image, 20,480 response vectors, the squared Euclidean distance, and 25 partitions. The resulting histogram has 25 features that represent the number of pixels that fall into each partition. The histogram has been normalized by the L2 norm.

CHAPTER 4 TEXTURE CLASSIFICATION AND SEGMENTATION

In this chapter, we put the proposed HoPLIT feature to the test by segmenting large texture mosaics. The mosaics are segmented by classifying and labeling every pixel to one of the nine texture images present in the mosaic. First, we introduce the data set, which comes from the Brodatz texture album [20]. We use this data set because it is a benchmark data set used by many papers in the literature for texture classification and segmentation. Next, we explain the methodology for our experiments. Then we present and analyze the results for the various experiments. Finally, we investigate the robustness of our feature to noise.

4.1 Brodatz Textures

The data for this chapter uses two sets of real textures taken from the Brodatz texture album. The texture images are normalized to remove the grayscale background effect. The normalized images occupy the entire gray-level range with a more or less uniform distribution [21][22]. The experiments utilize two different sets of nine Brodatz textures. The first set, Figure 4.1, appears several times in the literature for texture classification and segmentation [23][24]. The original data set contains ten textures separated into ten classes, however, one texture does not belong to the Brodatz album; therefore, we remove it from our data set.

The second set of textures were selected to experiment with more textures included in the album. This set, Figure 4.2, should prove more difficult to classify and segment because some of the textures are not as uniform. We increased the difficulty of this mosaic by using

images depicting similar textures like bricks and textures that have their intensities inverted. For each set, the nine textures are merged into a 1920x1920 texture mosaic and segmented.

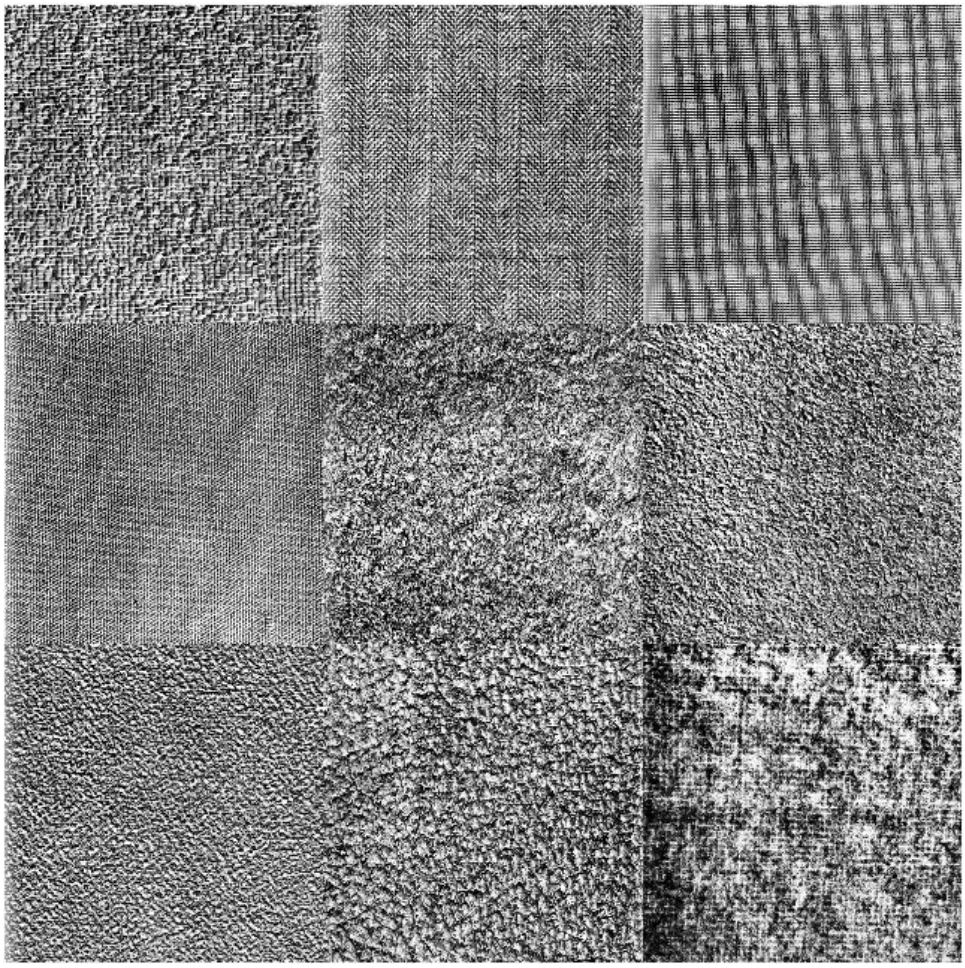

Figure 4.1: Nine Brodatz textures formed into a 3x3 mosaic that can be segmented. Top row: raffia (D84), herringbone (D17), French canvas (D21). Middle row: cotton canvas (D77), grass (D9), pressed cork (D4). Bottom row: handmade paper (D57), pigskin (D93), woolen cloth (D19). D84, D17, D21, and D77 are considered structured textures while D9, D4, D57, D93, and D19 are considered unstructured textures.

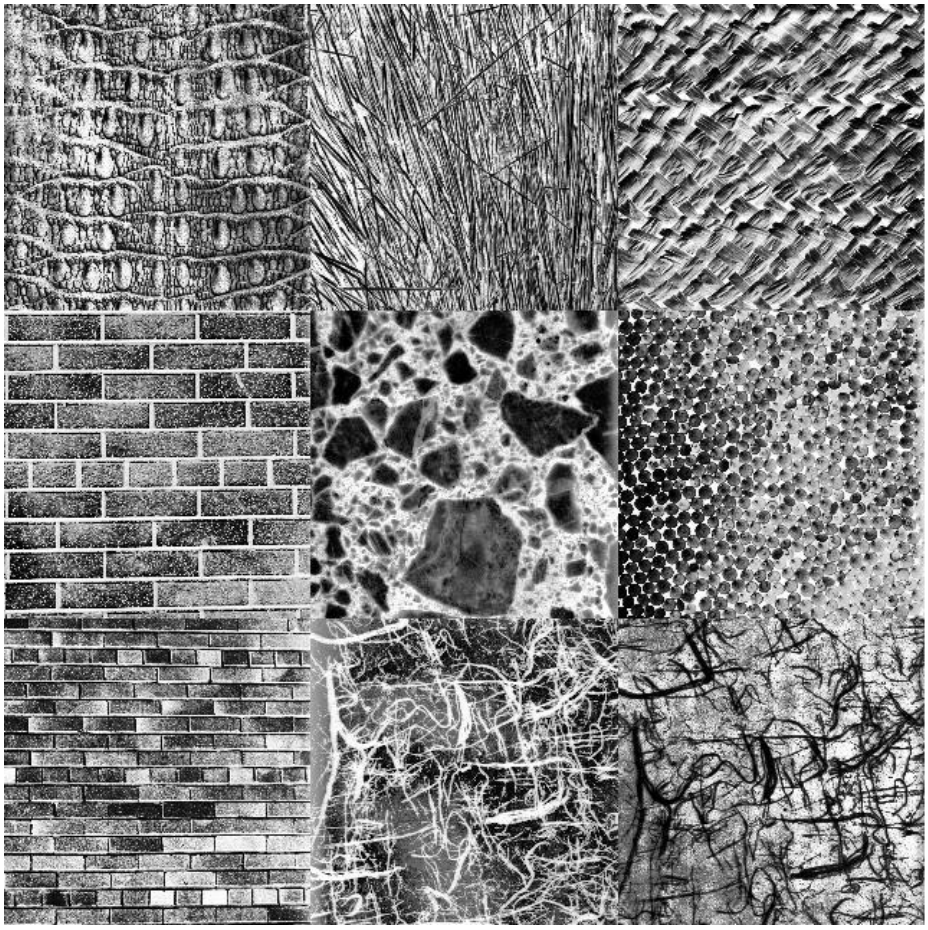

Figure 4.2: Nine Brodatz textures formed into a 3x3 mosaic that can be segmented. Top row: crocodile skin (D10), straw (D15), raffia weave (D18). Middle row: ceramic-coated brick wall (D26), European marble (D62), plastic pellets (D66). Bottom row: brick wall (D94), Japanese rice paper (D107), Japanese rice paper (D108).

4.2 Methodology

The first set of experiments with the Brodatz textures help to determine a good set of HoPLIT codebook parameters. These parameters include the number of partitions, the distance measure used in the clustering algorithm, and the set of filters used for convolution. The number of partitions are 5, 10, 25, and 50. The K-means clustering algorithm uses the squared Euclidean distance or city block distance. Three filter sets are

considered: Kirsch compass masks, Robinson compass masks, or Frei-Chen orthogonal filters [25]. Using these parameters, 24 combinations are formed and thus 24 different experiments. The method used to perform each experiment is rather simple. For each combination, the K-means algorithm was run ten times. The run with the lowest total summed distance was selected for use as the codebook.

Each texture image that makes up the data has a 640x640 resolution. Algorithm 2 shows the process used to generate the training data. For each independent texture image $\boldsymbol{I}_v \in \mathbb{W}$, we extract N subimages at random locations equating to training data with $9N$ subimage chips with the height, h, and the width, w, of the desired subimage size. For each subimage $\boldsymbol{X}$, centered around pixel (x,y), we extract its features. Using the training data, 12 different classifiers provided by Matlab's classification learner application are trained [26]. The classifiers are a linear discriminant, quadratic discriminant, linear support vector machine (SVM) quadratic SVM, cubic SVM, radial basis function SVM with σ =7.5, one nearest neighbor, ten nearest neighbors, inverse squared distance weighted ten nearest neighbors, bagged trees ensemble (i.e. random forest), subspace discriminant, and subspace k-nearest neighbors. The ensemble classifiers train weak classifiers on a subset of the feature space and implement a voting scheme to determine a class label. We then create a texture mosaic using the images in $\mathbb{W}$ by concatenating them together into a 3x3 image as shown in Figure 4.1 and 4.2.

In addition to the 24 parameter combinations we also tried experimenting with various subimage sizes: 64x64, 32x32, and 16x16. In order to test a feature's performance, the two mosaics shown in Figures 4.1 and 4.2 are segmented using the 12 trained classifiers. Algorithm 3 describes this process.

Algorithm 2: Train Classifier

Texture images $\mathbb{W} = \{\boldsymbol{I}_1, \boldsymbol{I}_2, \dots, \boldsymbol{I}_V\}$
Number of subimages N
Set parameters: $h, w, k = 1$
for $i = 1{:}V$
  for $j = 1{:}N$
    Select random pixel $(x, y) \in \boldsymbol{I}_i$
    $\boldsymbol{X} = \boldsymbol{I}_i(y - 0.5h{:}y + 0.5h, x - 0.5w{:}x + 0.5w)$
    $feat(k,:) = extractFeatures(\boldsymbol{X})$
    $lbls(k) = i$
    $k = k + 1$
  end
end
Train Classifier
Save Classifier

Algorithm 3: Segment Mosaic

Mosaic $\boldsymbol{M}$
$NumRow, NumCol = size(\boldsymbol{M})$
Set parameters: h, w
for $x = 0.5w + 1{:}NumCol - 0.5w - 1$
  for $y = 0.5h + 1{:}NumRow - 0.5h - 1$
    $\boldsymbol{X} = \boldsymbol{I}_i(y - 0.5h{:}y + 0.5h, x - 0.5w{:}x + 0.5w)$
    $feat = extractFeatures(\boldsymbol{X})$
    $lbls(y, x) = predict(feat)$
  end
end

To classify a pixel, the appropriate sized subimage around said pixel, has its features extracted. These feature vectors represent the local texture around that pixel. To remove any edge cases we create a border whose size depends on the subimages size on the outer edges of the mosaic. Once each pixel has its feature vector extracted, a label is generated using the appropriate trained classifier. The results of these classifications generates a segmented image where each pixel possesses a label between 1 and 9. We apply Algorithms 2 and 3 using the Haralick texture features, LBP, LDP, and HoPLIT. The

segmentations for each method are compared with the ground truth to compute a classification accuracy. Due to the randomness in selecting the training data, we average the classification error over ten interactions of Algorithms 2 and 3.

4.3 Results

Combining the parameter results and the various subimage sizes, we performed 72 experiments for each mosaic. Seventy-two experiments is too large to display in a table, therefore a visualization was devised to make it easy to determine the best parameterization for HoPLIT. Figure 4.3 and Figure 4.4 visualize the results for the mosaics in Figure 4.1 and Figure 4.2 respectively. The rows in the left image indicate what combination of parameters were used in the experiment as well as the subimage size. The rows in the right image show the performance of the 12 different classifies corresponding to the row in the left image. As an example, the top row in Figure 4.3 corresponds to an experiment using five partitions, the city block distance, the Frei-Chen filter set, and a subimage size of 16x16. To make it easy to identify the best configurations and classifiers, the higher classification accuracies are indicated by white pixels in the right image. For Figure 4.3, all the white pixels have classification accuracies greater than 97%. Two parameter configurations fall into this performance range.

In order to make a conclusion as to which parameter configuration is the best, we performed experiments on the more difficult mosaic, Figure 4.2. Figure 4.4 shows the resulting classification accuracies. Unlike Figure 4.3, a single pixel is labeled white and corresponds to a 90% classification accuracy. This one pixel lies in the same row as one of the better performing parameter configurations in Figure 4.3. This leads us to believe that

the best performing parameter configuration tried uses 50 partitions, the squared Euclidean distance, the Frei-Chen filter set, and a 64x64 subimage size.

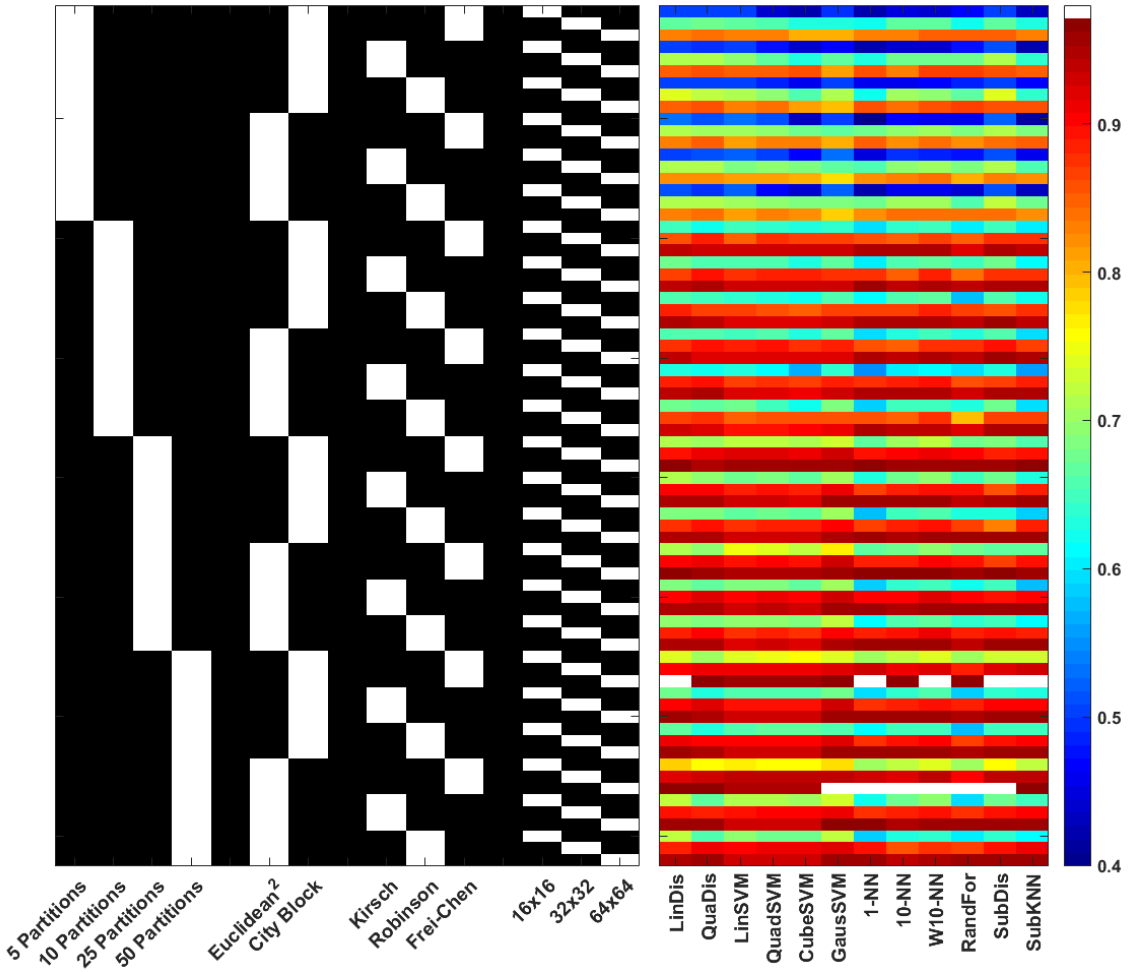

Figure 4.3: Classification results for 72 parameter configurations using the mosaic in Figure 4.1. Each row corresponds to a single parameter configuration experiment. (Left) A binary indicator image where white pixels indicate the active parameter setting in the four subgroups: number of partitions, distance measure, filter set, and subimage size. (Right) Classification accuracies at each parameter configuration for all 12 classifiers with white pixels indicating an accuracy greater than 97.1%.

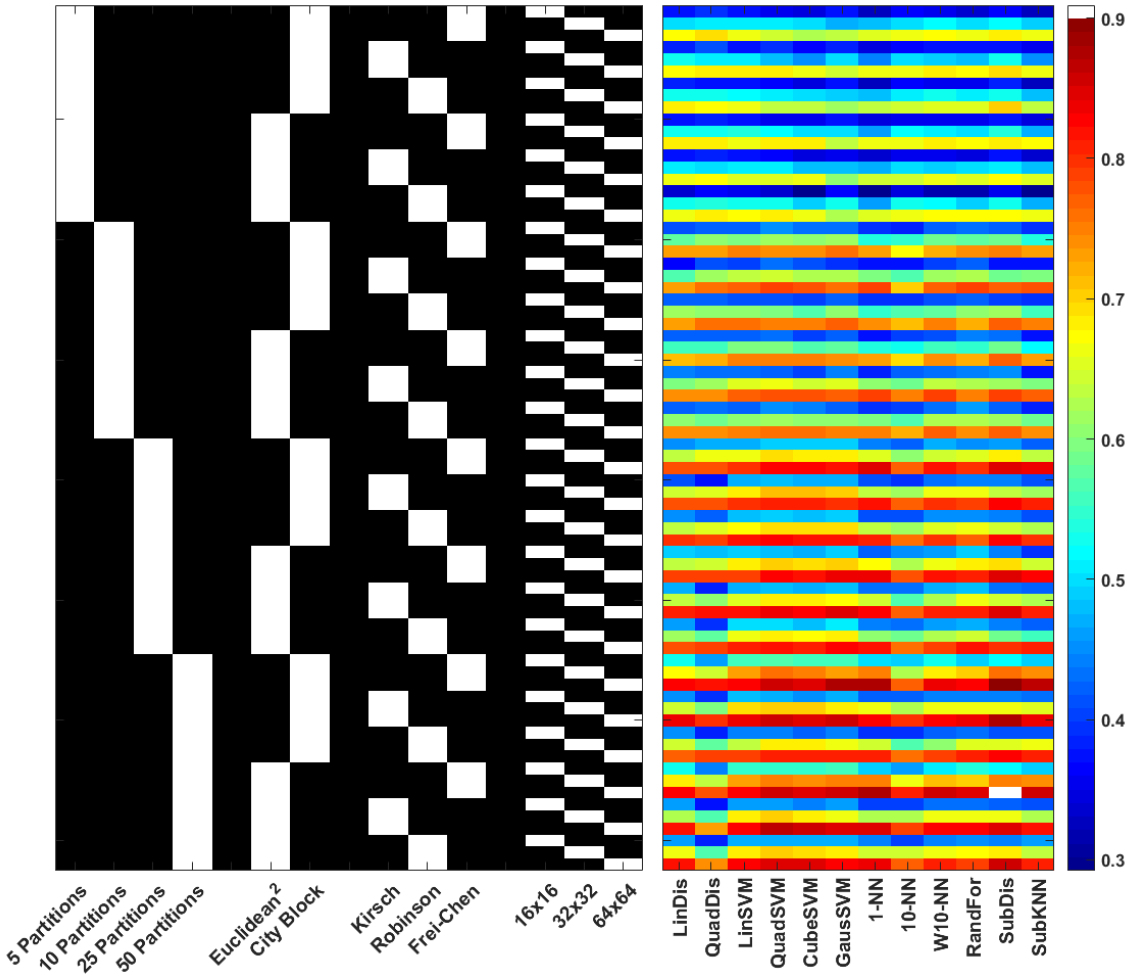

Figure 4.4: Classification results for 72 parameter configurations using the mosaic in Figure 4.2. Each row corresponds to a single parameter configuration experiment. (Left) A binary indicator image where white pixels indicate the active parameter setting in the four subgroups: number of partitions, distance measure, filter set, and subimage size. (Right) Classification accuracies at each parameter configuration for all 12 classifiers with white pixels indicating an accuracy greater than 90%.

Now that we have chosen our top performing HoPLIT parameterization, we can compare how well the feature performs against its predecessors, LDP, LBP, and the Haralick texture features. Table 4.1 records the mean classification results as well as one standard deviation of all 12 classifier for each set of features using the mosaic in Figure 4.1. From this table we see that the HoPLIT feature performs better than the other features for each classifier. The best performing classifier for the HoPLIT feature is the subspace discriminant classifier. In fact, three out of the four feature methods indicate a discriminant classifier as the best performing.

Table 4.1: Mean classification results and one standard deviation of ten training and testing iterations for the Haralick texture features, LBP, LDP, and the HoPLIT using the best parameterization when segmenting the mosaic in Figure 4.1. Each row corresponds to one of the 12 classifiers used in experimentation. A red number indicates the best performing feature for that specific classifier. Bold numbers indicate the best performing classifier for each feature extraction method.

	Haralick	**LBP**	**LDP**	**HoPLIT**
Linear Discriminant	0.9173±0.0093	**0.9670±0.0026**	**0.9643±0.0035**	0.9722±0.0019
Quadratic Discriminant	0.9131±0.0101	0.9547±0.0041	0.9524±0.0052	0.9597±0.0052
Linear SVM	0.9124±0.0068	0.9399±0.0094	0.9132±0.0252	0.9550±0.0043
Quadratic SVM	0.9219±0.0060	0.9439±0.0098	0.9147±0.0245	0.9561±0.0050
Cubic SVM	0.9182±0.0110	0.9422±0.0098	0.9092±0.0274	0.9539±0.0062
Gaussian SVM	0.9227±0.0078	0.9573±0.0068	0.9468±0.0053	0.9654±0.0061
1-NN	**0.9351±0.0084**	0.9323±0.0059	0.9347±0.0091	0.9774±0.0028
10-NN	0.8999±0.0172	0.9367±0.0070	0.9310±0.0146	0.9723±0.0031
Weighted 10-NN	0.9211±0.0112	0.9440±0.0070	0.9398±0.0145	0.9742±0.0024
Random Forest	0.9173±0.0062	0.9472±0.0064	0.9482±0.0044	0.9677±0.0020
Subspace Discriminant	0.9293±0.0173	0.9610±0.0047	0.9586±0.0030	**0.9789±0.0030**
Subspace KNN	0.8738±0.0220	0.8770±0.0329	0.8725±0.0272	0.9728±0.0023

Using the same HoPLIT parameterization, we train 12 new classifiers for each feature extraction method and compare the classification accuracies for the mosaic in Figure 4.2 with those of the other three features. Table 4.2 records the result. Unlike the mosaic in Figure 4.1, we see that classification is more difficult for Figure 4.2. All of the classification accuracies are significantly lower. Also from Table 4.2, we see that the HoPLIT feature is not always the best performer. For five classifiers, LBP holds the highest classification accuracy. All the other classifiers show HoPLIT as the top performer. The subspace discriminant classifier also produced the highest classification accuracy for the HoPLIT feature.

Table 4.2: Mean classification results and one standard deviation of ten training and testing iterations for the Haralick texture features, LBP, LDP, and the HoPLIT using the best parameterization when segmenting the mosaic in Figure 4.2. Each row corresponds to one of the 12 classifiers used in experimentation. A red number indicates the best performing feature for that specific classifier. Bold numbers indicate the best performing classifier for each feature extraction method.

	Haralick	**LBP**	**LDP**	**HoPLIT**
Linear Discriminant	0.7658±0.0095	**0.8618±0.0124**	0.7832±0.0113	0.8183±0.0126
Quadratic Discriminant	0.7691±0.0152	0.8539±0.0126	0.7964±0.0185	0.8068±0.0181
Linear SVM	0.7909±0.0125	0.8328±0.0187	0.7925±0.0210	0.8273±0.0162
Quadratic SVM	0.8072±0.0138	0.8432±0.0171	0.8044±0.0182	0.8496±0.0137
Cubic SVM	0.8053±0.0151	0.8312±0.0179	0.7897±0.0220	0.8479±0.0146
Gaussian SVM	0.7848±0.0104	0.8693±0.0092	0.8037±0.0255	0.8613±0.0140
1-NN	**0.8193±0.0104**	0.8078±0.0165	0.7926±0.0190	0.8666±0.0120
10-NN	0.7622±0.0108	0.7891±0.0232	0.7432±0.0246	0.7939±0.0203
Weighted 10-NN	0.8071±0.0112	0.8229±0.0227	0.7740±0.0180	0.8403±0.0162
Random Forest	0.8041±0.0127	0.8433±0.0120	**0.8064±0.0134**	0.8387±0.0084
Subspace Discriminant	0.7860±0.0377	0.8537±0.0202	0.7954±0.0134	**0.8914±0.0115**
Subspace KNN	0.7252±0.0226	0.7158±0.0431	0.7089±0.0225	0.8505±0.0118

4.4 Noise Robustness

The developers of LDP make the claim in [5] that their new local texture feature is more robust in the presence of noise than LBP. They demonstrate this claim by using a very simply and ill posed example. There example is shown in Figure 4.5a. In this example, the authors compute the LDP and LBP codes for a simple image. Then they apply Gaussian white noise to the image and re-compute the codes. They show that adding the noise caused a bit to change in the LBP code while the LDP code remained the same. However, without much thought, a situation can be devised that would cause the LDP code to change and the LBP code to remain the same as shown in Figure 4.5b.

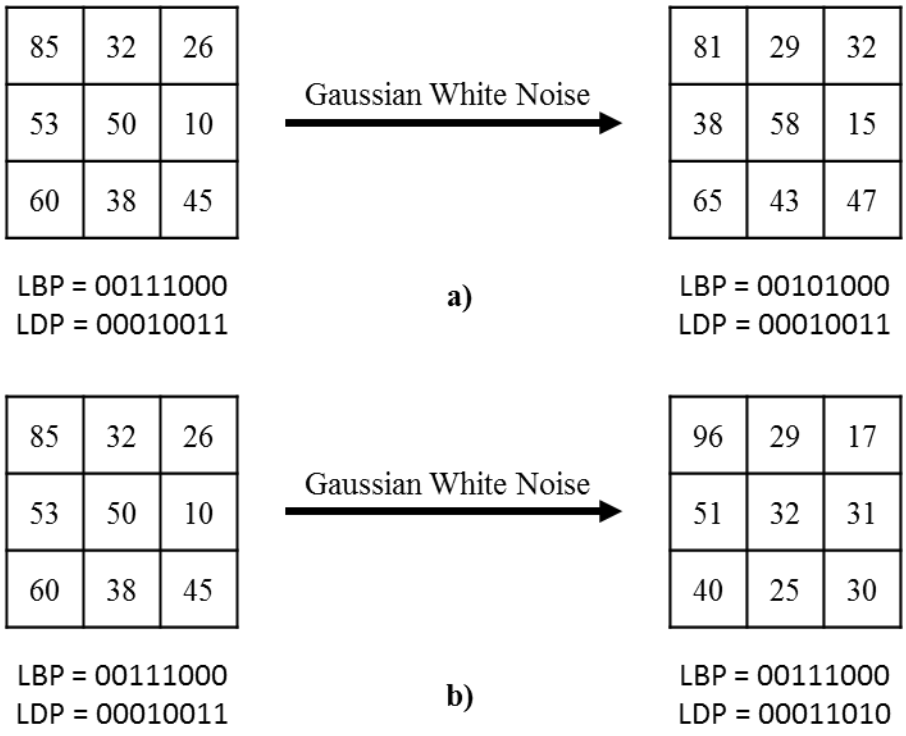

Figure 4.5: Example depicting the effect that adding noise to an image has on LBP compared to LDP. a) The LDP code remains constant while the 5th bit of the LBP code flips from 1 to 0. b) The LBP code remains constant while bits 1 and 4 flip in the LDP code. The third significant response was used as a reference in the LDP code computation.

Since HoPLIT was developed from LDP, we are curious as to whether our new feature is robust to noise. In the process, we also attempt to validate the claim that LDP is more robust to noise than LBP. The overall methodology for computing the features, training classifiers, and segmenting the mosaics remains unchanged. For these experiments, we simply add Gaussian noise to the mosaics before they are segmented using Algorithm 3. The training data is taken from the texture images without any applied noise. To adjust the amount of noise we increase the variance of a zero-mean Gaussian. The variances used for experimentation are 0, 0.001, 0.005, 0.01, 0.025, and 0.05. Figure 4.6 helps to visualize what adding noise does to the textures locally. Visually, adding noise does not alter the overall mosaic, however locally we can see where some features may show drops in performance. We repeat the noise experiment five times to produce an average segmentation where we assign the mode label to each pixel.

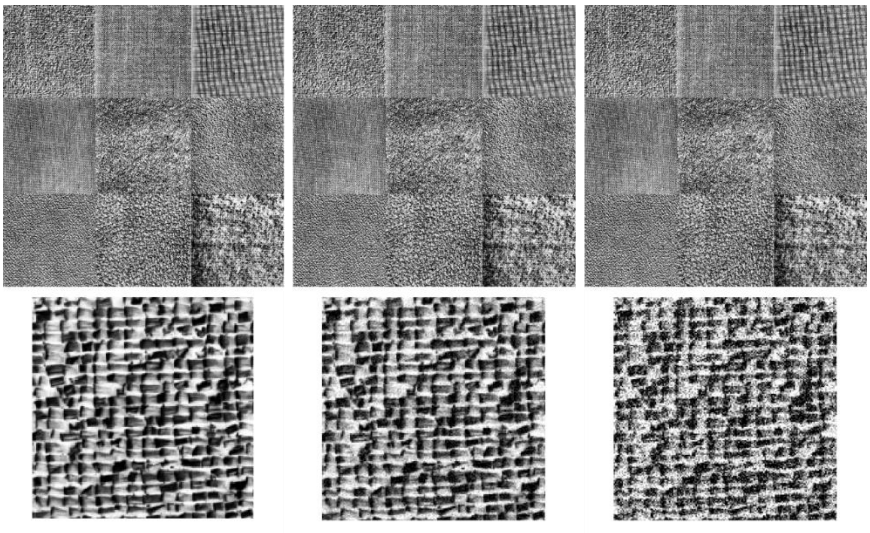

Figure 4.6: Images of the mosaic in Figure 4.1 with no noise (left), zero-mean Gaussian white noise with a variance of 0.005 (middle), and zero-mean Gaussian white noise with a variance of 0.05 (right). To prove that noise is in fact present in the images with noise, an image chip extracted from the NW texture is shown below each mosaic.

To evaluate each feature's performance we plot the classification as a function of variance in Figure 4.7 for the first mosaic image using the subspace discriminant classifier. By plotting the accuracy as a function of variance, we are able to visually asses how fast the accuracy drops as the noise increases. The more negative the slop in the plot the less robust that feature is to zero-mean Gaussian noise. Looking at Figure 4.7 it is clear that HoPLIT feature does not drop in accuracy as fast as the other three features. For the mosaic shown in Figure 4.1, the HoPLIT feature is robust to noise. It is also interesting that the HoPLIT feature has a higher classification accuracy at all noise levels.

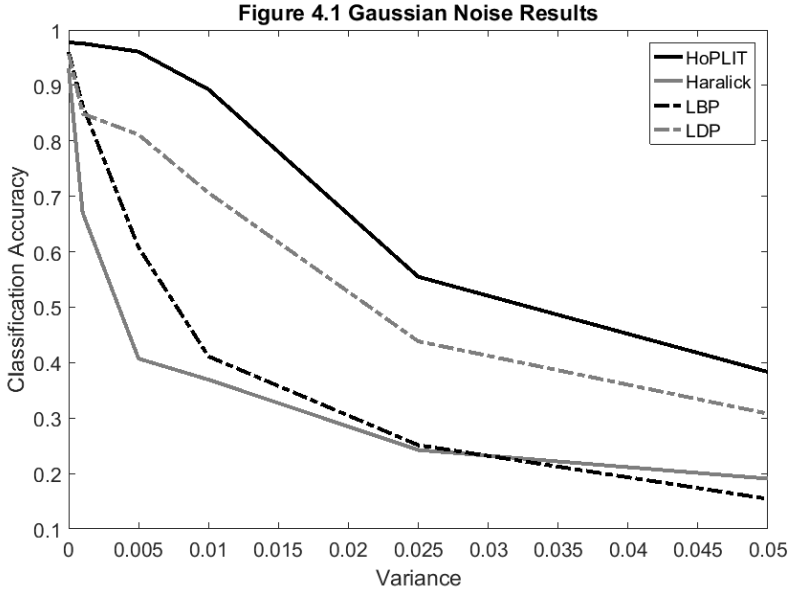

Figure 4.7: Plot of mean classification accuracies as a function of Gaussian noise amounts as indicated by the variance for the Figure 4.1 mosaic. The slop of the lines indicate whether a feature is robust to the Gaussian noise.

An alternative method for evaluating the performance of each extraction method is shown in Figure 4.8. In this figure, we plot the segmentation results along with its

corresponding ground truth. We assign the most frequent label between the five iterations as the final label. If each iteration produces a different label that pixel is given a label of zero. We can see that as the amount of noise increases, the segmentation becomes worse. For the Haralick texture features and LBP, this occurs somewhere between a variance of 0.001 and 0.005. Some textures appear to be more resilient to noise because they remain very well segmented at higher noise levels.

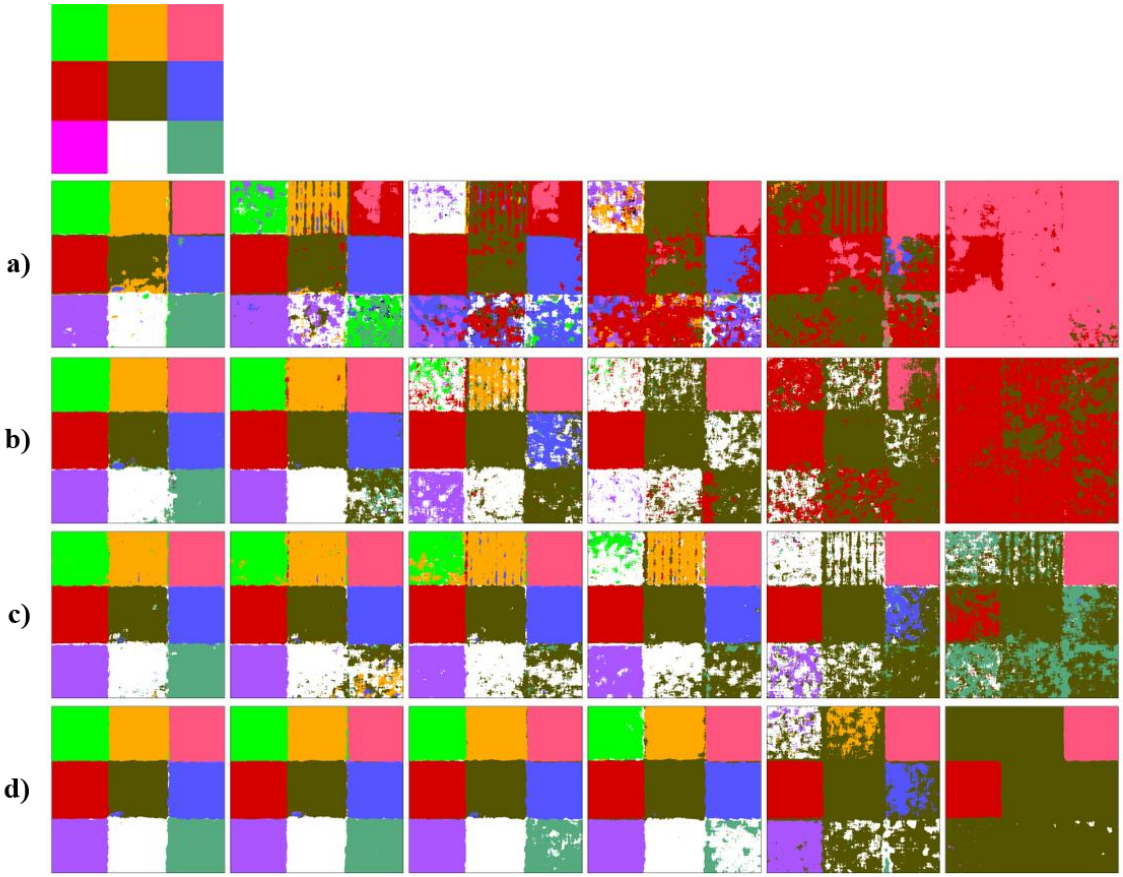

Figure 4.8: Segmentation results for the mosaic shown in Figure 4.1 using a subspace discriminant classifier for the a) Haralick texture features, b) LBP, c) LDP, and d) HoPLIT. The top most image shows the texture mosaic's ground truth. The bottom four rows depict the resulting segmentation after adding noise using a zero-mean Gaussian with variances of 0, 0.001, 0.005, 0.01, 0.025, and 0.05. Black pixels predicted different labels for each trial.

Also in Figure 4.8, we see that LDP does appear more robust to noise when compared to LBP. LDP is able to segment the majority of seven textures even after adding Gaussian noise with a variance of 0.01. The HoPLIT feature shows far greater robustness to noise as indicated by its segmented images. It appears that HoPLIT can segment eight textures for the first three noise levels rather well. The exception being the texture in the bottom right corner. Even at high levels of noise, the HoPLIT feature can accurately segment three textures while the other extraction methods can only segment one. This result is far better than we could have hoped.

To investigate further, we attempt to segment the more difficult mosaic in Figure 4.2. Figure 4.9 plots the classification accuracy as a function of variance for the four extraction methods. Right away, we notice the sharp drop in accuracy for the Haralick texture features, LBP, and LDP with a small amount of added Gaussian noise. On the other hand, the HoPLIT feature demonstrates a gentler slope. We believe that since the Haralick texture features and LBP are intensity based feature extraction methods, slight variations caused by the Gaussians noise alter some of the pixel relationships. For the Haralick texture features, this could result in a change in quantization level and thus changing the GLCM. In the case of LBP, a pixel that was less than the center pixel could become larger thus altering the binary pattern. LDP, on the other hand, should have been resilient noise. The sudden drop in classification accuracy for LDP could be due to the textures in Figure 4.2 having more macroscale texture compared to Figure 4.1, which could cause changes in the ranking of the response vectors. The HoPLIT feature retains a classification accuracy greater than 80% for the first noise level and does not drop below 50% until the variances greater than 0.01. These results, affirm the conclusion that the HoPLIT features are more

robust to zero-mean Gaussian noise. Unlike LDP, HoPLIT uses the response vectors to generate the feature vector not binary patterns. Adding noise to the image may alter the response vector values slightly however that change may not be enough to cause a pixel to change partitions. Thus retaining the same feature vector as if no noise was applied.

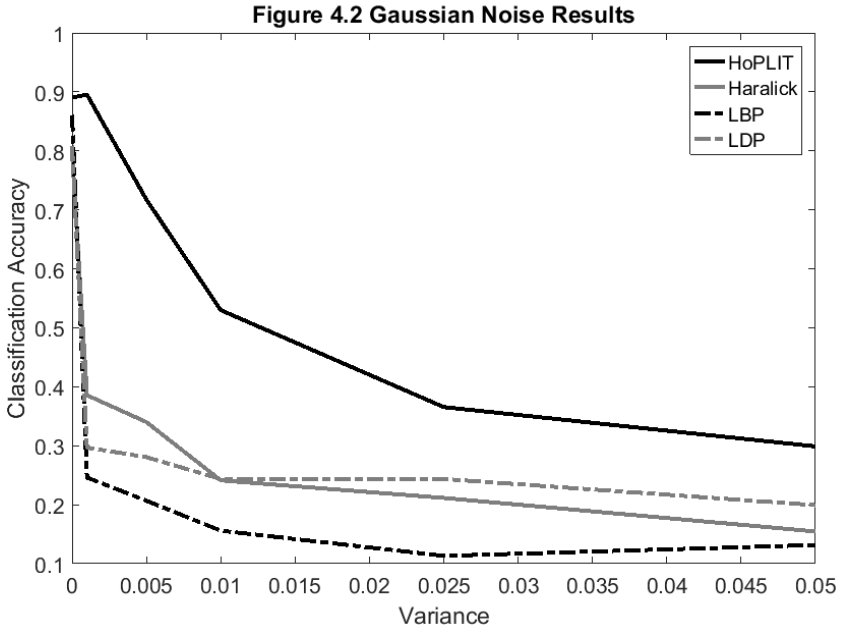

Figure 4.9: Plot of mean classification accuracies as a function of Gaussian noise amounts as indicated by the variance for the Figure 4.2 mosaic. The slop of the lines indicate whether a feature is robust to the Gaussian noise.

To drive the point home, we also show the visual segmentation results for the second mosaic in Figure 4.10. It is abundantly clear that HoPLIT is the better feature for segmenting this mosaic. Both LDP and LBP segment the majority of the mosaic to one texture after adding zero-mean Gaussian noise. The Haralick texture features do slightly better than LBP and LDP; however, at larger amounts of noise the mosaic tends to be segmented as one texture. Interestingly the Haralick texture features, LBP, and LDP segmentations converge to one of the brick textures as the noise level increases.

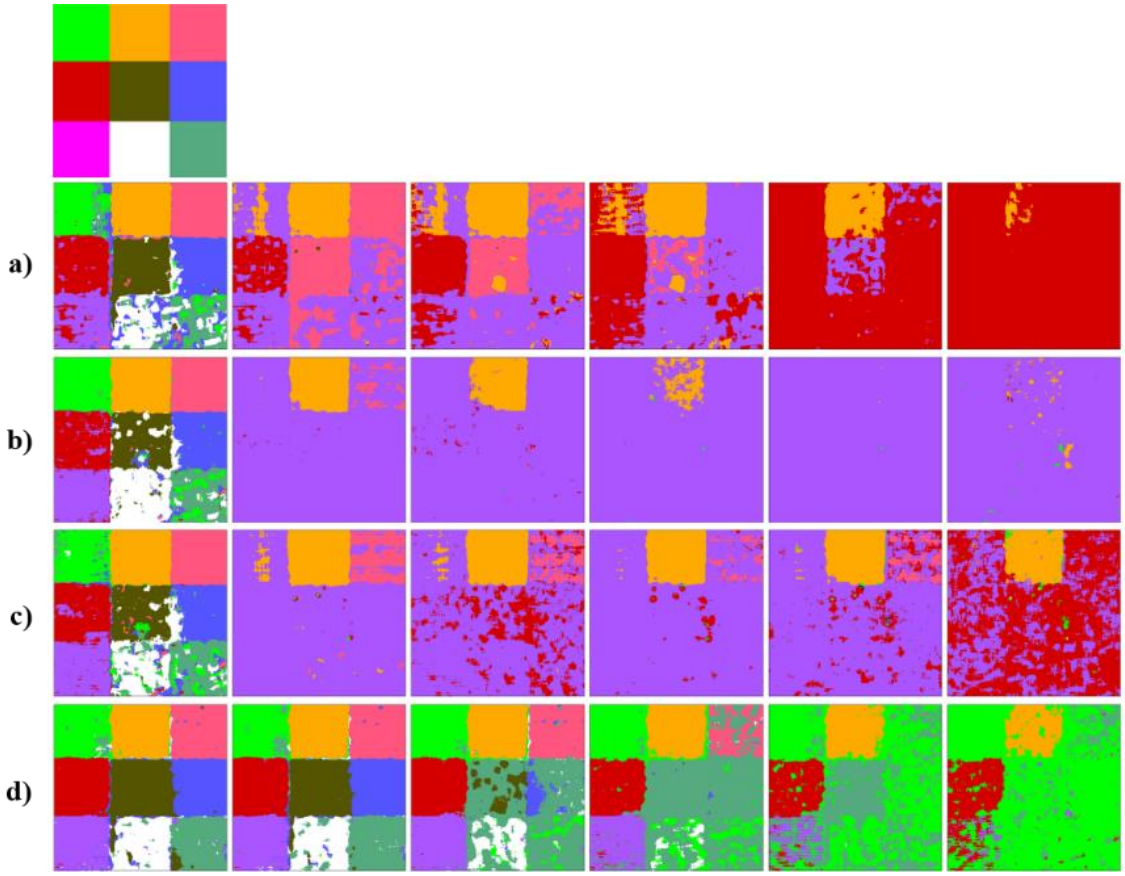

Figure 4.10: Segmentation results for the mosaic shown in Figure 4.2 using a subspace discriminant classifier for the a) Haralick texture features, b) LBP, c) LDP, and d) HoPLIT. The top most image shows the texture mosaic's ground truth. The bottom four rows depict the resulting segmentation after adding noise using a zero-mean Gaussian with variances of 0, 0.001, 0.005, 0.01, 0.025, and 0.05. Black pixels predicted different labels for each trial.

The HoPLIT feature vastly outperforms the other three methods. At the first noise level, the segmentation results show slight variations in segmentation. Even at higher levels of noise, the HoPLIT features are able to reasonably segment the majority of the three textures located in the upper left corner of the mosaic.

Due to the positive results shown by HoPLIT in the presence of Gaussian noise, we thought it would be interesting to perform an experiment using salt and pepper noise instead. For this experiment, we only performed one trial so determining the most frequent label is unnecessary. However, the overall methodology remains the same. Various amounts of salt and pepper were used ranging from 1 percent to 25 percent. In the Gaussian noise experiments the visual segmentations showed more compelling results. The segmentation results for the salt and pepper experiment, Figure 4.11 and 4.12, show less

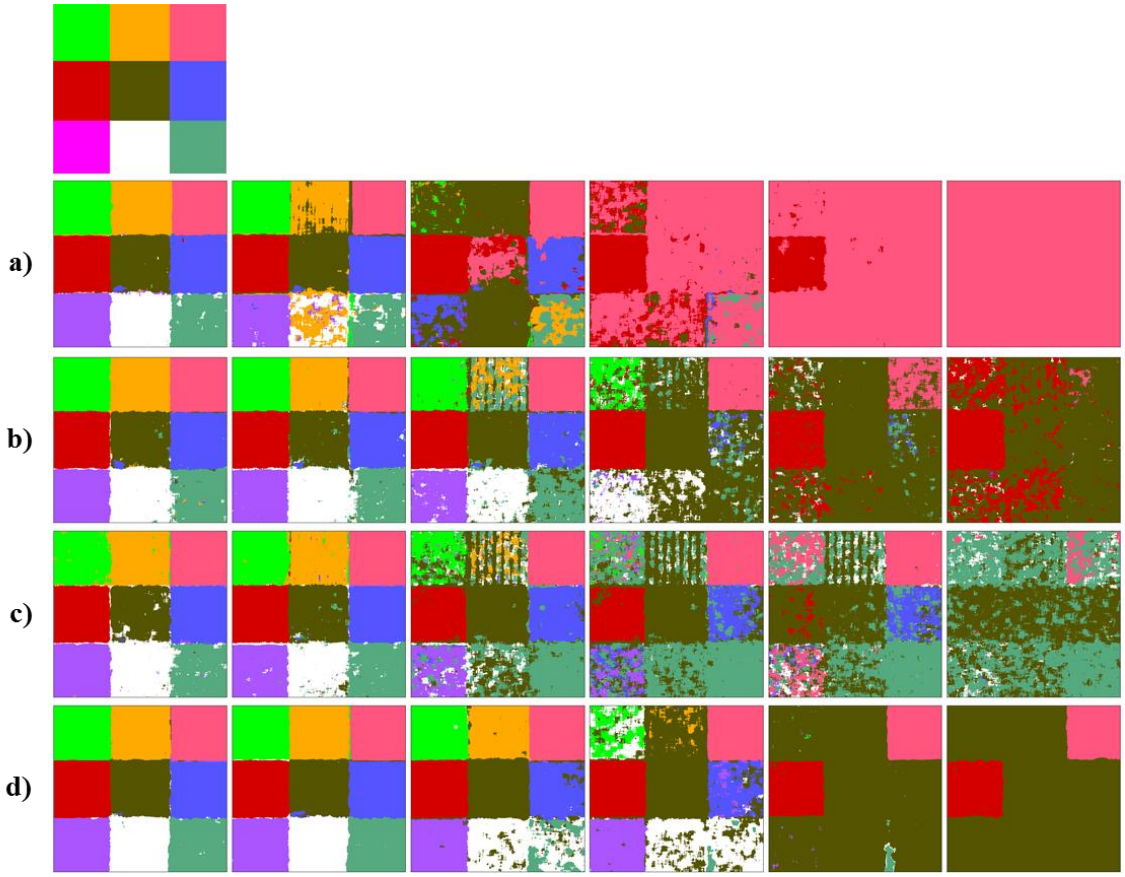

Figure 4.11: Segmentation results for the mosaic shown in Figure 4.1 using a subspace discriminant classifier for the a) Haralick texture features, b) LBP, c) LDP, and d) HoPLIT. The top most image shows the texture mosaic's ground truth. The bottom four rows depict the resulting segmentation after adding 0, 1, 5, 10, 15, and 25 percent salt and pepper noise.

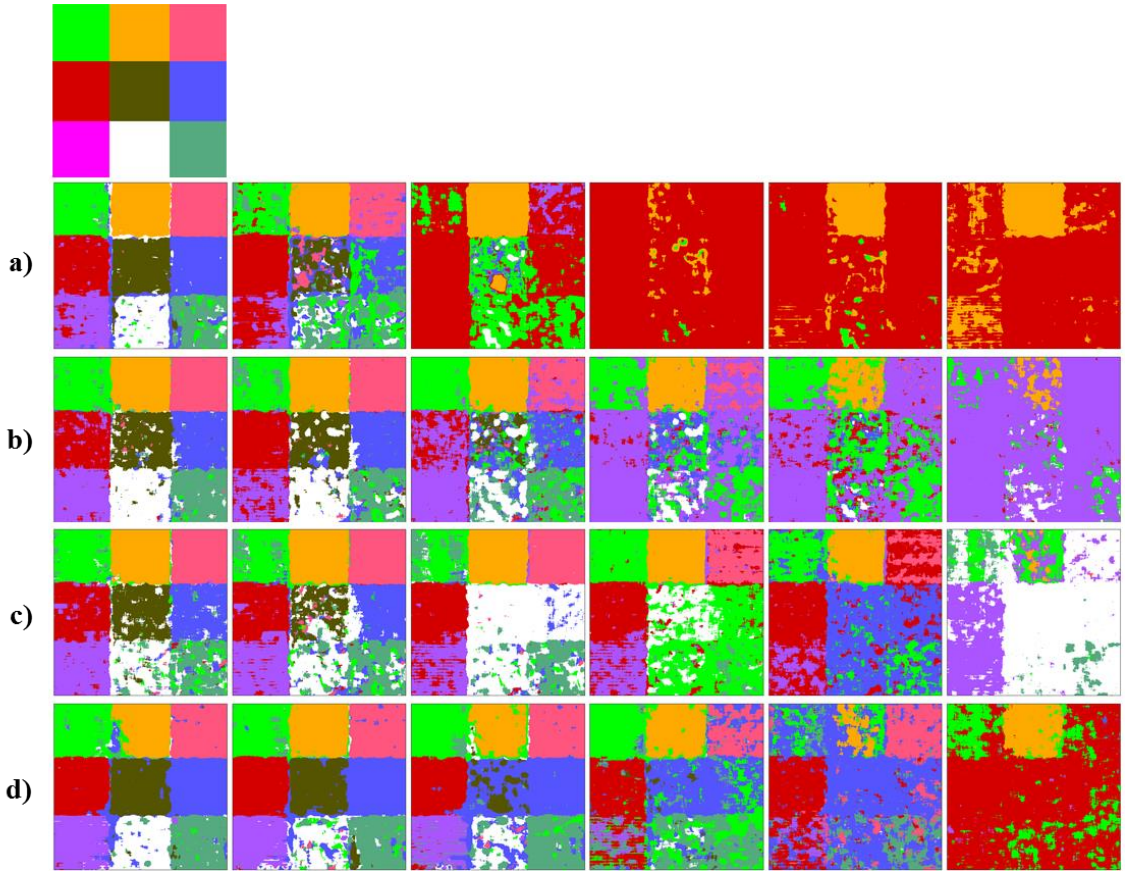

Figure 4.12: Segmentation results for the mosaic shown in Figure 4.2 using a subspace discriminant classifier for the a) Haralick texture features, b) LBP, c) LDP, and d) HoPLIT. The top most image shows the texture mosaic's ground truth. The bottom four rows depict the resulting segmentation after adding 0, 1, 5, 10, 15, and 25 percent salt and pepper noise.

compelling results. From these figures, determining which feature extraction method is more robust is more difficult. For this experiment using the plotted classification accuracies as a function of the percentage of salt and pepper noise proves more insightful, Figure 4.13 and 4.14. From these two plots, we see that the Haralick texture features display the least robustness to salt and pepper noise. On the other hand, HoPLIT, LBP, and LDP exhibit similar slopes, which indicates that they are all equally robust to the noise.

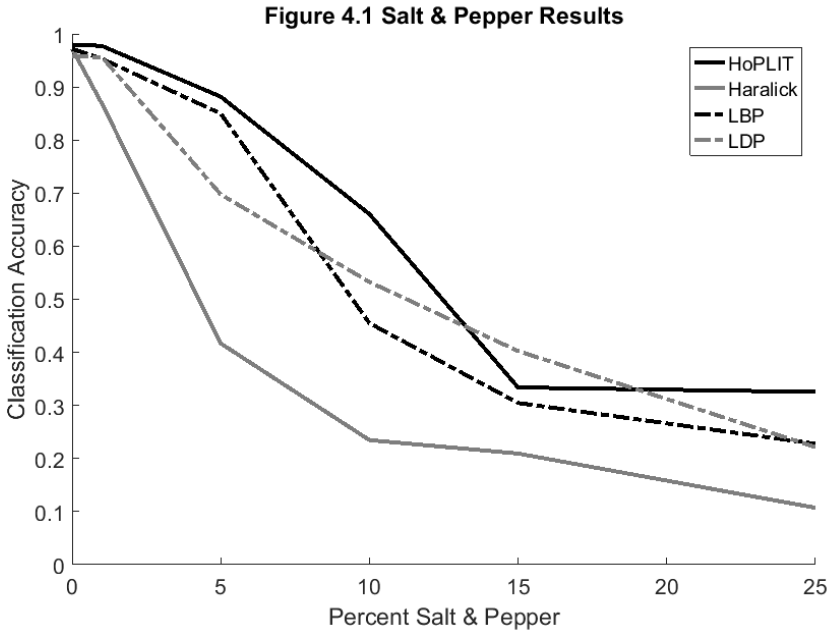

Figure 4.13: Plot of classification accuracies as a function of salt and pepper noise amounts as indicated by a percentage for the Figure 4.1 mosaic. The slop of the lines indicate whether a feature is robust to the noise.

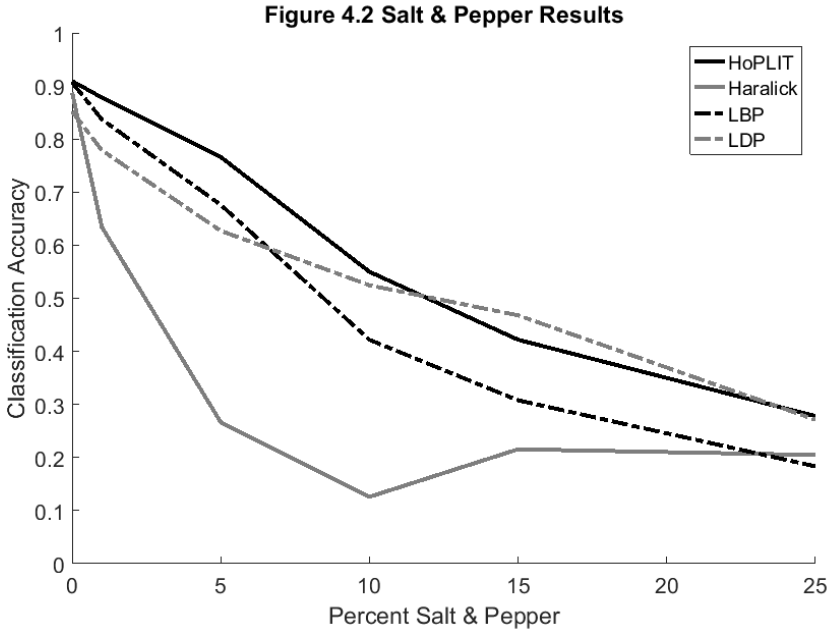

Figure 4.14: Plot of classification accuracies as a function of salt and pepper noise amounts as indicated by a percentage for the Figure 4.2 mosaic. The slop of the lines indicate whether a feature is robust to the noise.

CHAPTER 5 OBJECT DETECTION

In this chapter, we apply HoPLIT to an object detection problem. The data for this chapter was generated using a synthetic aperture acoustic sensor operating on US Army test lanes in an arid environment. Throughout the lanes, above ground explosive hazards are placed in the open as well as occluded by vegetation. The purpose of object detection is to identify the targets (i.e. explosive hazards) from the background with as few false alarms as possible. We start of the chapter with a description of the acoustic sensor and the data generated by it. Then we explain the various experiments we performed to evaluate the performance of HoPLIT. Finally, we analyze the results and draw some conclusions.

5.1 Sensor and Data

The synthetic aperture acoustic (SAA) sensor provides a low cost option compared to other popular modalities like radar and infrared imagery. Its low cost arises due to its simplicity and overall weight. The sensor requires a speaker and one or more microphones. The most comparable system to SAA is a side-scanning synthetic aperture sonar except it operates in air instead of water. Figure 5.1 shows the basic set up of the sensor on a vehicle. The sensor has a linear FM chirp ranging from 2-17 kHz, a pulse repetition frequency of 20 Hz, and swath width of 8.75m. For a more detailed description of the sensor and its parameters, refer to Luke *et al.* [27]. Sampling the returning sound waves allows for formation of images with spatial references. The images produced by the sound wave have unique textures. It is our belief that, explosive hazards possess a texture different from the background and other objects.

Figure 5.1: Perspective view of the SAA sensor system mounted on a manned all-terrain vehicle [27].

The SAA data used for experimentation uses a Pacific ACO microphone sampling at 80 kHz. During this sampling cycle, four lanes at an arid US Army test site were used to gather data; two containing unconcealed hazards (Lanes A and B) and two containing occluded hazards (Lane C and D). Occluded hazards appear in, beneath, or behind bushes, and in-between petrified logs, which obstruct the sensor's view. The raw binary data goes through a beamforming process. Knowing the three-dimensional location of all pixels and sensor, we can interpolate a value using the measured return signal and time traveled for each pulse. Integrating over multiple pulses for each pixel generates the approximate energy received at that location. The beamformed image has a 5 mm x 5mm resolution and covers the entire lane. Figure 5.2 shows a five-meter section of Lane D.

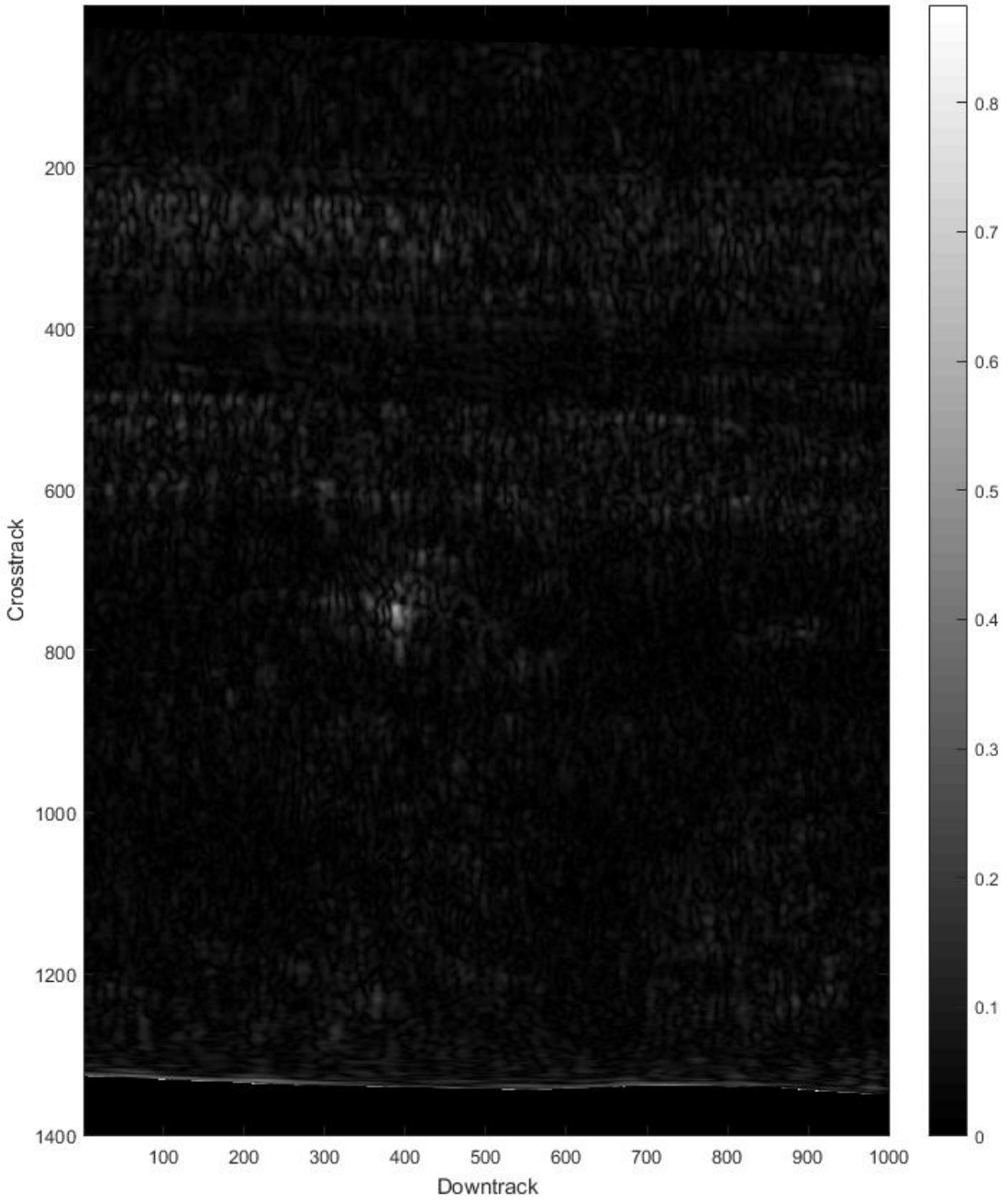

Figure 5.2: Beamformed 5 m section of Lane D with a hazard located at approximately (750,400). The image is scaled locally and the color map has been adjusted to highlight the unique texture of the acoustic sensor.

After beamforming the entire lane, a RX prescreener [28] with a 0.4m by 0.4m inner window, 0.6m by 0.6m guard window, and 1.3m by 1.3m outer window generates a hit list of possible alarm locations. The Mahalanobis distance between the inner and outer windows provides a confidence at the center pixel in the inner window. Using a maximum filter, we locate the peaks in the confidence surface creating the initial hit list. Applying non-maximal suppression with a radius of 0.5m helps to reduce the number of hits by suppressing low confidence alarms near high confidence alarms. The final set of hits create the data set used for experiments.

Before extracting features, each run of each lane goes through a normalization to the interval [0,1] because the range of intensity values between each individual run can vary. We normalize the image by subtracting the minimum intensity value and then dividing by the range of intensity values. We chose a linear normalization method because we want preserve the relations between pixels. Using a non-linear mapping like histogram equalization may cause the responses generated by the filters to be altered in a unexpected way. By normalizing each run, all the runs become less dependent on intensity variations and more dependent on the actual texture. Using the normalized data, we extract 0.6m by 0.6m (129 by 129 pixels) image chips at every corresponding hit location. We also created additional data sets by resizing the 129x129 pixels chips to 64x64, 32x32, and 16x16 pixels to observe what happens when the unique acoustic textures degrade. Figure 5.3 shows an example image chip and its resized versions. The smaller image chips also provide faster computation time when partitioning because the number of total pixels decreases. For each image chip, various texture features are extracted for classification by a support vector machine (SVM). Chapters 2 and 3 lay out the various extraction methods used in this

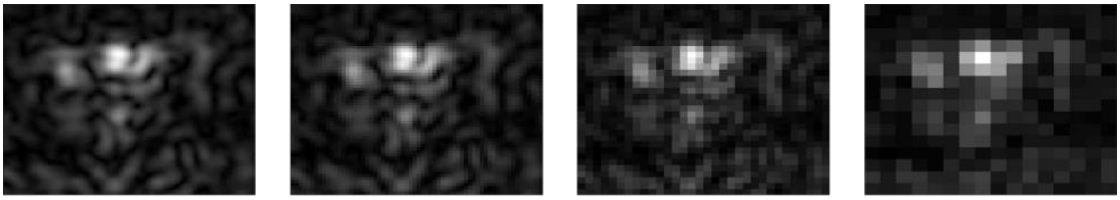

Figure 5.3: Example resized image chips where a) is the original 129x129 chip, b) 64x64, c) 32x32, and d) 16x16.

chapter. In the case of the Haralick texture features, we rescale the image chips to a 0 to 255 gray-level image before building the co-occurrence matrix.

5.2 Experiments

To determine the performance of the feature extraction methods, we preform several classification experiments. This particular set of experiments utilizes a SVM with a Gaussian kernel to produce a score whose sign dictates the predicted label of a testing sample. This score actually represents the distance to the decision boundary; where a positive distance indicates the explosive hazards class and a negative distance indicates a false alarm. Before the SVM can generate scores, it first needs to be trained. For all subsequent experiments, we generate a balanced data set comprised of hits located at most 0.5m from ground truth targets and randomly selected false alarms from Lanes A, B, and C. Each image chip in the training data set has its features extracted using the methods described in Chapters 2 and 3, producing different data sets.

In the cases of LBP and the Haralick texture features, the only parameters that can produce an alternative data set comes from using different image sizes generating four combinations each. On the other hand, along with changes in image size, LDP has the option of varying the set of filters used to generate the response vectors; thus creating 12

different combinations. The remaining 96 data sets arise from varying not only the image size and filter set, but also the number of partitions and distance measure used in the K-means algorithm. Foer each parameter combination, the HoPLIT parameter α was set to 1.0. The number of partitions selected for experimentation are 5, 10, 25, and 50. Due to the amount of time required to run K-means using many millions of 8-dimentsion points, larger numbers were not tested. Within the K-means algorithm, every point has its distance to each partition center calculated using some measure of distance. These experiments only uses two well-known measures of distance: squared Euclidean and city block. Due to the random initialization of the cluster centers, the K-means algorithm has a tendency to converge to a local minimum. In an attempt to keep this from happening the K-means was run five times. The cluster centers that produced the lowest total summed distances were kept for feature extraction. Using each combination of parameters, we generate 116 different data sets. Due to the randomness in selecting the false alarms included in the training data sets, we train 20 classifiers for each combination.

The testing data originates from alarms generated by a prescreener applied to Lane D, which is comprised entirely of concealed explosive hazards. Four separate runs sampled Lane D, and are each classified using the same trained classifier within a single trial. To represent the performance of each combination, we compute receiver operating characteristic (ROC) curves. Before building a ROC curve, we average the SVM scores between the 20 trials for each run individually. Then, using the average SVM scores as confidences, we generate a ROC curve for each Lane D run. Finally, we vertically average the ROC curves between the four separate runs based on confidence. The averaged ROC provides the general performance of the classifier and thus the features. The normalized

area under the ROC curve at a false alarm rate of 0.1 false alarms per m^2 provides a quantitative measure of a combinations performance.

5.3 Results

Table 5.1 catalogs the normalized area under the vertically averaged ROC curve for feature extraction methods, not including the proposed HoPLIT feature, at the four different image sizes. Clearly, the Haralick texture features outperform LBP and LDP. However, the LBP performance appears lacking. The poor performance may originate from normalizing the data to [0,1]. When using raw intensity values, the ROC curve shifts toward the upper left corner resulting in an approximate 0.1 increase in the normalized area. The exact reason behind the poor performance of LBP remains unknown. Without normalizing the data, LDP still outperforms LBP. The Haralick texture feature appear to benefit the most from normalizing the data. Each feature method shows a decrease in the normalized area when changing from 129x129 to 64x64. Excluding the LDP features using the Robinson compass masks, all other feature sets show a slight increase in the normalized area under the curve changing from 64x64 to 32x32. This phenomenon came very unexpectedly. In fact, for the Haralick features, the 32x32 image size outperforms all other feature methods shown in Table 5.1. Resizing the image to a 16x16 image shows the worse performance due to the loss of the more detailed localized textures.

Table 5.1: Resulting normalized area under the ROC curves for the data sets generated by resizing the original 129x129 pixel image into 64x64, 32x32, and 16x16 pixel images for the Haralick texture features, LBP , and three versions of LDP each using a different set of eight filters. The area under the curve was computed at a false alarm rate of 0.1 false alarms per m^2 using the ROC curves generated by vertically averaging the ROC curves of each run of Land D based on confidence.

	129x129	**64x64**	**32x32**	**16x16**
Haralick	0.7258	0.7235	**0.7280**	0.6777
LBP	**0.4670**	0.2813	0.2968	0.2479
Kirsch LDP	0.5818	0.6210	**0.6460**	0.5790
Robinson LDP	**0.6641**	0.6621	0.6518	0.5829
Frei-Chen LDP	**0.6503**	0.6117	0.6463	0.5636

Due to the large number of experiments, we needed a visualization method to determine the best set of parameters for the HoPLIT features. Figure 5.4 shows this visualization. Each binary section represents one parameter used to compute the features. Each row in the figure represents one parameter configuration. To determine the best performing parameter set, we sort the experiments by the normalized area under the vertically averaged ROC curve.

The first section represents the number of partitions produced. From this section, we can deduce that the top performing parameter sets prefer a larger number of partitions. The second section indicates the distance measure used to determine distance to a partition center. Clearly, the top performers favor the city block distance. Section three in Figure 5.4 corresponds to the set of filters convolved with the images. Within this section, we see that selecting a different set of filters has a minimal effect on the feature's performance. This may be due to the three filter sets possessing similar individual filters. For example, both the Kirsch and Robison compass masks as well as four Frei-Chen filters highlight edges parallel to cardinal directions.

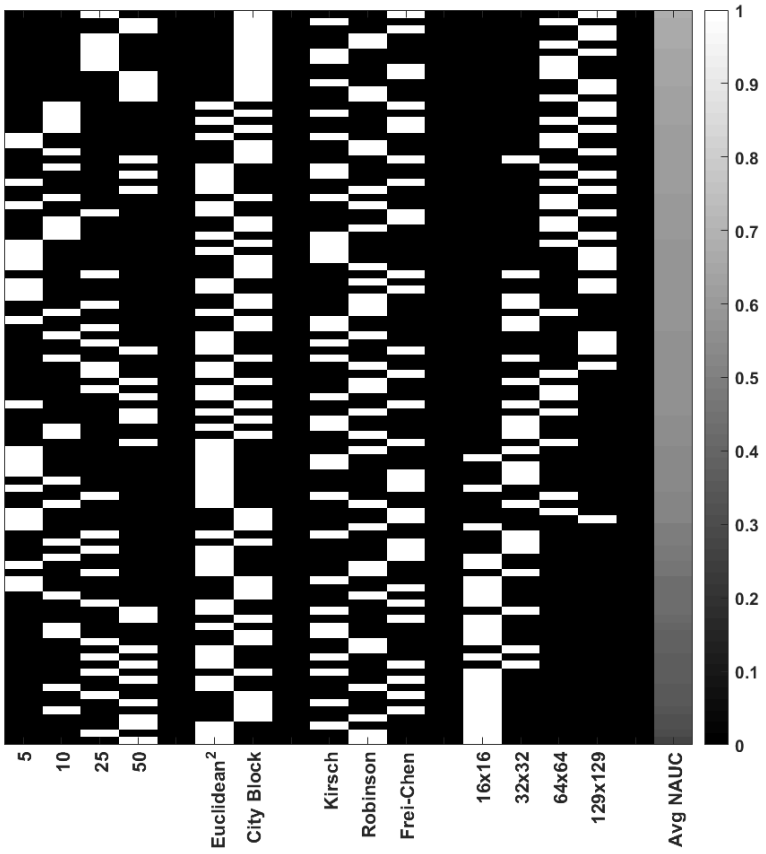

Figure 5.4: Various combinations of parameters for the HoPLIT feature sorted by the normalized area under the ROC curve generated by vertically averaging the Lane D runs together based on confidence. The area under the curve is at 0.1 false alarms per m^2. Each row represents one of the 96 combinations. Columns 1-4 corresponded to the number of partitions in increasing order. Columns 6 and 7 represent the squared Euclidean and city block distances, respectively. Columns 9-11 indicate the filter set. Columns 13-16 represents the image size, in increasing order. Finally, the last column indicates the normalized area under the curve.

The last binary section indicates the image size. Looking at this section in Figure 5.4, we see that for HoPLIT the use of larger images produces better features compared to using smaller images. The reason behind this may come from having more data and thus creating more optimal partitions within the data. Another possible reasoning stems from

the larger images possessing higher detailed textures. Resizing the images to a smaller number of pixels smooths the textures. Using Figure 5.4, the best preforming parameter configuration uses 25 partitions, city block as the distance measure, the Frei-Chen filter set, and an image size of 129x129.

To summarize the results of in Figure 5.4, Figure 5.5 plots the ROC curves for the top four HoPLIT parameter configurations. Looking at the ROC curves we see the minimal difference between the four configurations, however the top result does show a slightly greater area under the curve. Figure 5.6 summarizes the overall results of this experiment. From these ROC curves, we see that the best LBP result pales in comparison to the other three extraction methods. The Haralick texture features outperform the other features as indicated by a difference of at least 0.04 area under the curve. This difference equates to detecting one, maybe two targets, undetected by the other methods. Although the proposed HoPLIT feature did not achieve the top performance, it still slightly outperforms LDP.

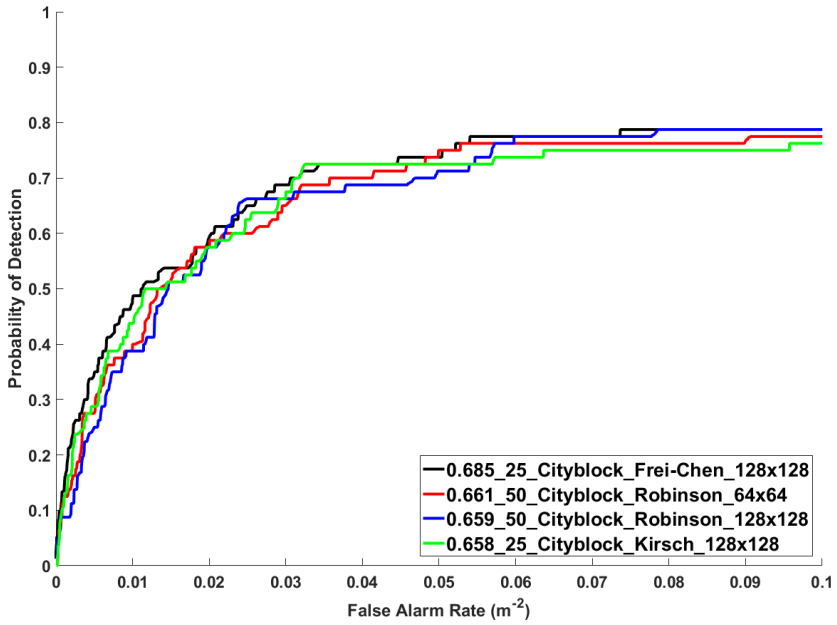

Figure 5.5: Land D average ROC curves for the top four combinations using the HoPLIT feature out to a false alarm rate of 0.1 false alarms per m^2. Inside the legend, the names indicate the normalized area under the curve, number of partitions, distance measure, filter set, and the image size.

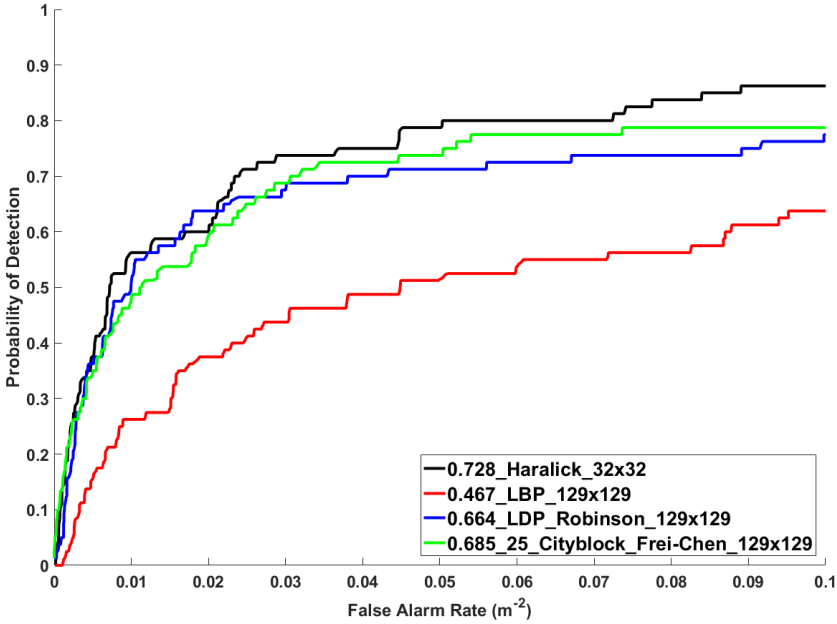

Figure 5.6: Land D average ROC curves for the top combination using each texture feature extraction method out to a false alarm rate of 0.1 false alarms per m^2.

CHAPTER 6 CONCLUSIONS AND FUTURE WORK

Within machine learning and pattern recognition research, one area focus on image classification and segmentation. One of the key types of features used in this area is texture. In this thesis, we investigated three common texture extraction methods found in the literature: the Haralick texture features, LBP, and LDP. Since LDP was developed, many researchers have modified the algorithm in attempts to improve upon its discriminative abilities. These modifications included changing the filters used to generate response vectors or augmenting the encoding process to retain more filter response information.

For this work, we propose a new texture feature called the histogram of partitioned localized image texture. This new feature modifies the LDP algorithm by first using various known filter sets. These filter sets may not necessarily be directional. For experiments, we treated the filter set as a parameter in experimentation. The second modification attempts to improve the aggregation of the filter responses around an image to retain as much information as possible. We do this by implementing the most basic bag-of-words model. We build the codebook by selecting a subset of response vectors belonging to the whole data set and then partitioning that subset using K-means. The K-means clusters became the partition centers and were used to determine which partition a response vector belongs. Within the K-means algorithm, we are able to select the number of partitions and the distance formula used to measure the distance between data points and the centers. The histogram descriptor for an image is simply the frequency of response vectors belonging to each partition based on the data points distance to the centers.

To evaluate the performance of our texture feature we attempt to solve two types of problems. The first problem is texture classification and segmentation. For the experiments we used textures taken from the Brodatz texture album and created texture mosaics, one easy and one hard. The performance of a feature was measured by the percentage of pixels classified correctly. Before comparing the new feature to existing features, we determined a parameterization that produced good results. We trained and tested 12 various classifiers falling into the discriminant, support vector machine, *k*-nearest neighbor, and ensemble subgroups. The best performing parameterization used 50 partitions, the squared Euclidean distance, the Frei-Chen filter set, and a subimage size of 64x64. This parameterization gave the top results for both the easy and hard mosaics. Next, we compared the features performance against the Haralick texture features, LBP, and LDP. For both the mosaics, the top performing classifier was the subspace discriminant classifier for the HoPLIT features.

To investigate the strength HoPLIT further, we performed an experiment that assessed how well the features were able to correctly classify textures under noisy conditions. We simply added increasing amounts of zero-mean Gaussian noise to each mosaic before extracting features during testing. The training data was taken from the original texture images. As the noise increased, by increase the variance, we saw a degradation in the local texture around a pixel and in turn a drop in classification performance. However, our feature demonstrated a more gradual drop in performance when compared to the other three, thus validating our conclusion that our feature is more robust to Gaussian noise.

The second problem on which we tested our feature was object detection. This problem used data gathered by a synthetic aperture acoustic sensor in an arid testing environment. Using this data, we attempt to identify above ground explosive hazards that are either visually exposed or occluded by vegetation. This problem was much more difficult for HoPLIT as well as for the others. Similar to the texture classification and segmentation experiments, we first try to determine the right set of HoPLIT parameters required to produce the best results. Performance for this problem used the area under the ROC curve. The best performing HoPLIT parameters were 25 partitions, the city block distance, and the Frei-Chen filter set. Unfortunately, in this problem domain, HoPLIT was not the top performer. Although it was not the best for this particular data set, it still performed slightly better than the LDP and vastly outperformed the LBP.

Since HoPLIT is still rather young, numerous avenues can be taken to improve the algorithm. A simple addition to the algorithm is to create histograms in a gridded image and then concatenate them together to form one descriptor. Many algorithms including LBP, LDP, and histogram of oriented gradients use this concept, or have the ability to. It would also be interesting to try a larger number of partitions. The one drawback with using K-means to partition the data is the time required to converge. If the data set is sufficiently large, millions of points, the algorithm may never converge. This problem may be solved by using a different method to partition the data. Another approach would be to not use the bag-of-words model and treat the response vectors as the features themselves.

The final way to further improve the algorithm lies in the filter set. One can simply dive further into the literature to find additional common filter sets to use. An alternative route is to develop a filter-learning algorithm. This learning algorithm would attempt to

create the most optimal set of filters for a given data set. More than likely, we would use a multi-objective genetic algorithm. The difficult part of this approach will be both the objective function and the chromosome representation. One alternative filter learning approach developed by Keller et al, used the Choquet integral to create 3x3 filters [29].There are very few filter-learning algorithms in the literature so this work would be very interesting and beneficial.

BIBLIOGRAPHY

[1] R. Haralick, K. Shanmugam and I. Dinstein, "Textural Features for Image Classification", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. -3, no. 6, pp. 610-621, 1973.

[2] Dong-chen He and Li Wang, "Texture Unit, Texture Spectrum, And Texture Analysis", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 509-512, 1990.

[3] T. Ojala, M. Pietikainen and D. Harwood, "Performance evaluation of texture measures with classification based Kullback discrimination of distributions", in *Proceeding of 12th International Conference on Pattern Recognition*, Jerusalem, 1994, pp. 582-585.

[4] T. Jabid, H. Kabir and O. Chae, "Local Directional Pattern (LDP) for Face Recognition", in *2010 Digest of Technical Papers International Conference of Consumer Electronics (ICCE)*, Las Vegas, NV, 2010, pp. 329-330.

[5] T. Jabid, H. Kabir and O. Chae, "Local Directional Pattern (LDP) – A Robust Image Descriptor for Object Recognition", in *2010 7th IEEE International Conference of Advanced Video and Signal Based Surveillance*, Boston, MA, 2010, pp. 482-487.

[6] A. Ramirez Rivera, J. Rojas and O. Chae, "Local Gaussian Directional Pattern for face recognition," *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Tsukuba, 2012, pp. 1000-1003.

[7] S. M. Z. Ishraque, A. K. M. H. Banna and O. Chae, "Local Gabor directional pattern for facial expression recognition," *2012 15th International Conference on Computer and Information Technology (ICCIT)*, Chittagong, 2012, pp. 164-167.

[8] A. Higashi, T. Yasui, Y. Fukumizu and H. Yamauchi, "Local Gabor directional pattern histogram sequence (LGDPHS) for age and gender classification," *2011 IEEE Statistical Signal Processing Workshop (SSP)*, Nice, 2011, pp. 505-508.

[9] M. H. Kabir, T. Jabid and O. Chae, "A Local Directional Pattern Variance (LDPv) Based Face Descriptor for Human Facial Expression Recognition," *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, Boston, MA, 2010, pp. 526-532.

[10] A. Ramirez Rivera, J. Rojas Castillo and O. Oksam Chae, "Local Directional Number Pattern for Face Analysis: Face and Expression Recognition," in *IEEE Transactions on Image Processing*, vol. 22, no. 5, pp. 1740-1752, May 2013.

[11] M. A. Mohamed, H. A. Rashwan, B. Mertsching, M. A. García and D. Puig, "Illumination-Robust Optical Flow Using a Local Directional Pattern," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 9, pp. 1499-1508, Sept. 2014.

[12] A. Rahman and L. Ali, "Optimized Local Directional Pattern for robust facial expression recognition," *2012 International Conference on Informatics, Electronics & Vision (ICIEV)*, Dhaka, 2012, pp. 560-564.

[13] Z. H. Xie and Z. Z. Wang, "Infrared face recognition based on adaptively local directional pattern," *2014 11th International Computer Conference on Wavelet*

Actiev Media Technology and Information Processing(ICCWAMTIP), Chengdu, 2014, pp. 240-243.

[14] S. Sivapalan, D. Chen, S. Denman, S. Sridharan and C. Fookes, "Histogram of Weighted Local Directions for Gait Recognition," *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Portland, OR, 2013, pp. 125-130.

[15] Fang Zhou, Jun Liu, Tanghui Wang and Xin Chen, "Hierarchical codebook for moving microbe detection in sewage," *2015 10th International Conference on Communications and Networking in China (ChinaCom)*, Shanghai, 2015, pp. 813-818.

[16] [G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, 1st ed. New York: McGraw-Hill, 1982.

[17] B. Julesz, "Textons the elements of texture perception and their interactions", Nature, no. 290, pp. 91-97, 1981.

[18] D. G. Lowe, "Object recognition from local scale-invariant features," *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Kerkyra, 1999, pp. 1150-1157 vol.2.

[19] C. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. New York: Springer, 2006, pp. 424-425.

[20] P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* New York: Dover Publications, Inc., 1966.

[21] S. Abdelmounaime and H. Dong-Chen, "New Brodatz-Based Image Databases for Grayscale Color and Multiband Texture Analysis", *ISRN Machine Vision*, vol. 2013, pp. 1-14, 2013.

[22] D. He and A. Safia, "Normalized Brodatz", *Multibandtexture.recherche.usherbrooke.ca*, 2016. [Online]. Available: http://multibandtexture.recherche.usherbrooke.ca/normalized_ brodatz.html. [Accessed: 10- Sep- 2016].

[23] Greenspan, H., Goodman, R.., Chellappa, R., & Anderson, C.H. "Learning texture discrimination rules in a multiresolution system." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 16, no. 9, pp. 894-901. 1994.

[24] S. Grossberg and J. Williamson, "A self-organizing neural system for learning to recognize textured scenes", *Vision Research*, vol. 39, no. 7, pp. 1385-1406, 1999.

[25] W. Frei and Chung-Ching Chen, "Fast Boundary Detection: A Generalization and a New Algorithm," in *IEEE Transactions on Computers*, vol. C-26, no. 10, pp. 988-998, Oct. 1977.

[26] Mathworks. (2016). *Classification Learner App*. [Online]. Available: https://www.mathworks.com/help/stats/classification-learner-app.html?requestedDomain=www.mathworks.com.

[27] R. Luke, S. Bishop, A. Chan, P. Gugino, T. Donzelli and M. Soumekh, "A synthetic aperture acoustic prototype system," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XX*, Baltimore, 2015.

[28] I. S. Reed and X. Yu, "Adaptive multiple-band CFAR detection of an optical pattern with unknown spectral distribution," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, no. 10, pp. 1760-1770, Oct 1990.

[29] Keller, J. M., P.D. Gader, and A. H. Hocaoglu. "Fuzzy integrals in image processing & recognition". in *Fuzzy Measures & Integrals: Theory & Applications*. M Grabisch, T Murofushi and M Sugeno. 1st ed. Heidelberg, NY: Physica-Verlag, 2000. pp. 435-466. Print.