Fractal Analysis of Seafloor Textures for Target Detection in Synthetic Aperture Sonar Imagery

---

A Thesis Presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment of the Requirements for the Degree

Master of Science

---

by

Thomas R. Nabelek

James M. Keller Ph.D., Advisor

May 2018

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

Fractal Analysis of Seafloor Textures for Target Detection

in Synthetic Aperture Sonar Imagery

presented by Thomas Nabelek, a candidate for the degree of Master of Science in Computer Engineering, and hereby certify that, in their opinion, it is worthy of acceptance.

Professor James Keller

Professor Alina Zare

Professor Mihail Popescu

DEDICATION

I dedicate this thesis to my grandparents: Dr. Igor Nabelek (Grandaddy), Dr. Anna Nabelek (Babcia), Klara Nabelek, Rev. Dr. Charles Russ (Grandpa), and Dorothy Russ (Grandma). The lives lived so well by these individuals constantly inspire me to aim high while always taking time to show love to those whom I meet along the journey, and to make a positive impact where I can.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Dr. Jim Keller, for pushing me to always produce high caliber work, for guiding a relatively inexperienced researcher through all that is involved in a large research project, for his education and the sharing of his technical expertise, and for putting up with me knocking on his door to ask questions throughout this process. I thank the larger Mizzou-University of Florida team for all the contributions that they have made to this work, particularly Dr. Alina Zare and Aquila Galusha. I also thank Dr. Keller, Dr. Zare, and Dr. Mihail Popescu for their participation on the thesis committee.

This work was funded by the Office of Naval Research (ONR) grant number N00014-16-1-2323 to support the U.S. Naval Surface Warfare Center (NSWC). Thank you to all those at the ONR and NSWC for the support that they have given to the larger project that this thesis work is derived from.

Finally, I thank my family for every kind of support that they have given me throughout my education, and for constantly pushing me to be the best person that I can be.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Fractal analysis of an image is a mathematical approach to generate surface related features from an image or image tile that can be applied to image segmentation and to object recognition. In undersea target countermeasures, the targets of interest can appear as anomalies in a variety of contexts, visually different textures on the seafloor. In this thesis, we evaluate the use of fractal dimension as a primary feature and related characteristics as secondary features to be extracted from synthetic aperture sonar (SAS) imagery for the purpose of target detection. We develop three separate methods for computing fractal dimension and produce both primary “slope” and secondary “intercept” and “lacunarity” features as candidates for classification application. Tiles with targets are compared to others from the same background textures without targets. The different features produced are tested with respect to how well they can be used to detect targets vs. false alarms within the same contexts. These features are evaluated for utility using sets of image tiles extracted from a SAS data set generated by the U.S. Navy in conjunction with the Office of Naval Research. We find that almost all features produced have potential to perform well in real-world classification tasks, with the slope and intercept features from a fractional Brownian motion model performing the best among those from the three individual methods. We also find that the secondary intercept features are just as useful, if not more so, in classifying false alarms vs. targets when compared to the primary slope features. The secondary lacunarity features, however, dominate as the most useful features produced. We also do experiments to address the high amount of compute time required to produce the features and to discover how the features change with distance from the image sensor.

1. INTRODUCTION AND BACKGROUND

In the applications of image segmentation and classification, among others, a key part of the processes used is feature extraction. Features extracted from any type of image can be used to determine many informative qualities. One of the most informative of these qualities is texture. Much research has previously been done to seek out the most informative texture features. What sort of textures does an image contain? Does an image contain multiple textures? How do the textures change across an image? These are all questions that can be answered through the use of extracted texture features. The local binary pattern (LBP) [1] [2], local direction pattern (LDP) [3], Haralick gray-level co-occurrence matrix (GCLM) [4], Sobel edge detector [5], and Histogram of Oriented Gradients (HOG) [6] features name just a few of the now well-known features used in the field.

One feature, which has been investigated and applied in domains related to our work to a much lesser extent, but having significant potential in those domains and quickly becoming more prevalent in the community, is termed “fractal dimension.” The fractal dimension of a given texture is essentially a description of the roughness, or ruggedness, of that texture. All of the three methods used to compute the fractal dimension of a texture that we investigate also produce a secondary “y-intercept” feature, and one of those three methods produces a secondary “lacunarity” feature as well. These are discussed to a greater extent later on. The work described here has been most significantly guided by the investigative work done by Keller [7] in the late 1980s, though the concept of using fractal geometry in image analysis extends further back than that.

Fractal geometry was initially explored by Benoit Mandelbrot [8] in the 1970s and has since been shown to be a valuable tool for describing the irregular and complex shapes seen in the natural world. As described by Crownover [9], one can think of elements such as a line segment, a square, and a cube as having dimensions D of 1, 2, and 3, respectively. If those elements are divided into N equal sub-segments, sub-squares, and sub-cubes, respectively, with each sub-piece being thought of as a scaled down version of the original element, with a scaling ratio r in all dimensions, then the relationship between N and r is given by the power law

$$Nr^D = 1. \tag{1}$$

For example, in the case of a cube, if one divides the cube into 8 equal sub-cubes, each side will have been scaled by $r = 1/2$ and $8 * (1/2)^3 = 1$. These elements – the line segment, square, and cube – all have integer dimensions, but this relationship can be extended to elements that we can think of as having non-integer dimensions.

If a bounded set A exists in Euclidean n-space where A is the union of N distinct copies of itself, each progressively scaled down by r in every dimension as described above, A is said to be self-similar [8] [9] [10]. The similarity dimension – or the "fractal dimension" – of A is given by taking the logarithm on both sides and rearranging Equation (1) to get

$$D = \frac{\log N}{\log{(1/\mathrm{r})}} \tag{2}$$

with this equation holding true for even non-integer values of D. Theoretically, we will always have $1 \leq D \leq 2$ for one-variable data and $2 \leq D \leq 3$ for two-variable data as in the case of our SAS imagery. D, then, should be a useful descriptor of the "roughness" or “ruggedness” of a surface. The higher the value of D, the more rugged the surface is, as illustrated in Figure 1-1 and Figure 1-2.

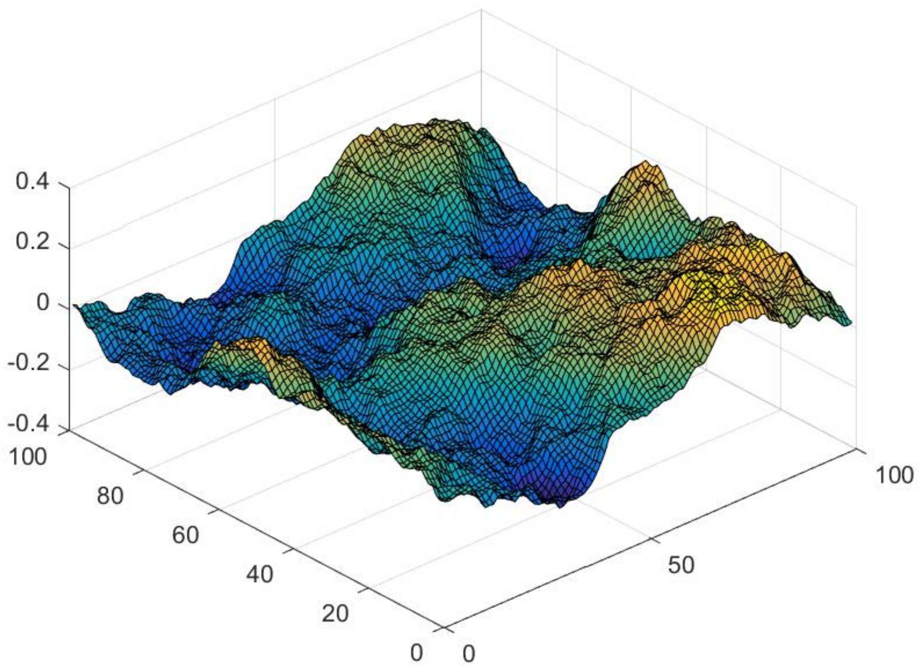

Figure 1-1 – An artificially generated surface having a fractal dimension value $D = 2.2$

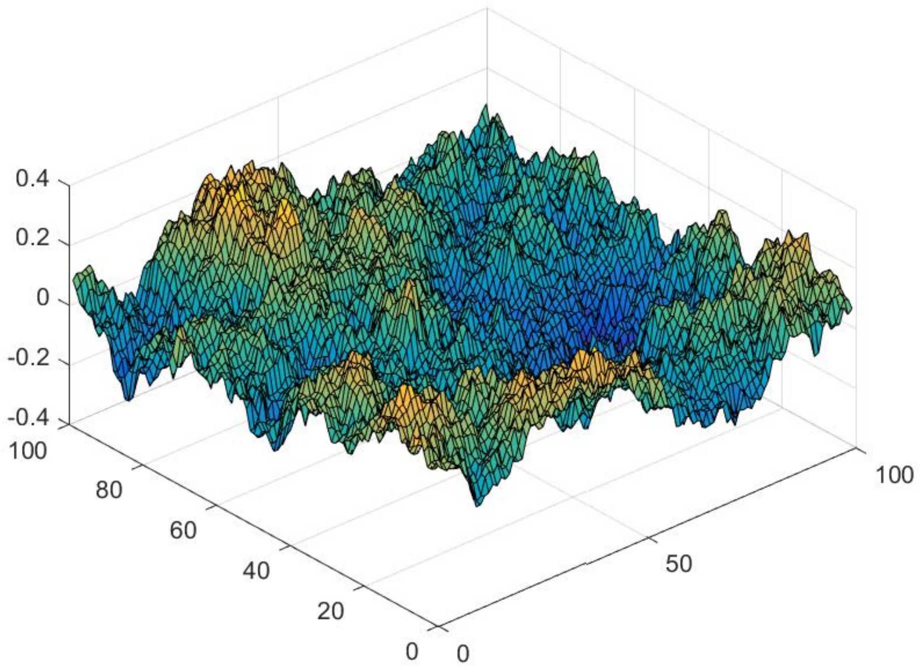

Figure 1-2 – An artificially generated surface having a fractal dimension value $D = 2.6$

More surfaces are shown in Figure 3-1, Figure 3-2, and Figure 3-3.

We hypothesize that the mathematical representation of the roughness, D, of image tiles containing targets will be distinguishable from D representing those tiles containing only background textures. In practice, there are multiple ways of estimating D. Here, we implement and examine two box counting methods and a method relying on a fractional Brownian motion (fBm) model. All of these methods also produce a secondary feature C or B, discussed in Section 2, that relates to some version of the power law given in (1) and should also be a distinguishing feature.

It is hypothesized that the further an image tile is from the path of the vehicle collecting the data, the higher the C and B constants of proportionality will be. Just as the scale of an object appears to decrease with increasing distance in visual imagery, the scale of the seafloor textures seen by the SAS sensor will also decrease with distance. The SAS beamforming method used to create the imagery corrects for this, so it is not obvious to the human eye. However, because we do not decrease the size of our box as it is moved further away from the sensor, the constant of proportionality should instead change. We discuss and show results of an experiment to test this hypothesis in Section 3.6.

As pointed out by Mandelbrot [11] and Voss [12], there are textures that may have very similar fractal dimension values but very different appearances or textures. For this reason, fractal dimension alone is not sufficient for describing natural textures. Mandelbrot [13] introduced "lacunarity" as a way to describe the characteristic of fractals geometries having the same dimension but different textures. Keller [7] introduces a method of finding lacunarity using the same data computed in our first box-counting method for fractal dimension computation (Method 1). We term this "Keller lacunarity" here. Voss [12] introduced another, termed “Voss lacunarity” here, again making use of the Method 1 computations. These two methods of computing lacunarity are discussed further in Section 2.1. Finally, Williams [14] discusses a third method of lacunarity computation, termed “Williams lacunarity” here. The Williams method does not make use of the work done in Method 1 and is discussed in Section 2.5.

2. METHODS OF FRACTAL DIMENSION AND SECONDARY FEATURE COMPUTATION

2.1. Method 1

The first method we use to compute the fractal dimension of a given surface is a modified version of the box counting method [7] [9] [12]. Box counting is based on the property that the number of boxes of size L, $N(L)$, that it takes to cover a fractal surface is proportional to L^{-D}. We define this box of size L as having dimensions L_{width} x L_{depth} x L_{height}. One can think of L_{width}, L_{depth}, and L_{height} as scaling factors applied to a cube with dimensions 1 x 1 x 1 along the first, second, and third dimensions, respectively. These values will be incremented to grow the size of the box as the algorithm proceeds.

To begin, we take a SAS image tile with y pixel rows and x pixel columns for which we wish to compute the fractal dimension. The intensity values for the entire data set being analyzed will range from z_{min} to z_{max}. These intensity values are determined across the data set as a whole rather than the individual image or image tile so that we have consistent dimension computation across the data set. Additionally, if y and x are the same or roughly the same for each tile, we can fix $L_{depth} = L_{width}$. We set

$$L_{height} = L_{width} * L_{hm} \tag{3}$$

where

$$L_{hm} = \frac{z_{max} - z_{min}}{mean(x,y)}. \tag{4}$$

The purpose of the L_{hm} multiplier term is to further scale the height of the box to better fit the data, accounting for the fact that the range $[z_{min}, z_{max}]$ may not be proportional to the depth and width of the tile footprint, y and x, respectively. If y and x are not the same for every tile, fixed values – perhaps the mean y and mean x across the data set – should be used in determining L_{hm}. Again, the purpose of this is to have consistent conputation when analyzing many tiles. This box scaling is best illustrated in Figure 2-4 where there are six boxes along the depth dimension, six along the width dimension, and seven along the height dimension. If, in that case, we had instead set $L_{hm} = 1$, we would have only one box along the height dimension and computing a useful fractal dimension value would not be possible.

We center a box of size L around the first pixel that allows the entire box to fit over the footprint of the SAS image tile. The placement of this first box is shown by the green box in Figure 2-1. With the box in place, the points that fall within the box are counted. In other words, if the pixel i around which the box is centered is located at (i_1, i_2) and has an intensity value of i_3, all of the pixels k for which the following conditions are met are counted:

(a) $$i_1 - floor\left(\frac{L_{depth}}{2}\right) \leq k_1 \leq i_1 + floor\left(\frac{L_{depth}}{2}\right) \tag{5}$$

(b) $$i_2 - floor\left(\frac{L_{width}}{2}\right) \leq k_2 \leq i_2 + floor\left(\frac{L_{width}}{2}\right) \tag{6}$$

(c) $$i_3 - \frac{L_{height}}{2} \leq k_3 \leq i_3 + \frac{L_{height}}{2}. \tag{7}$$

Figure 2-2 illustrates this box placement and point counting.

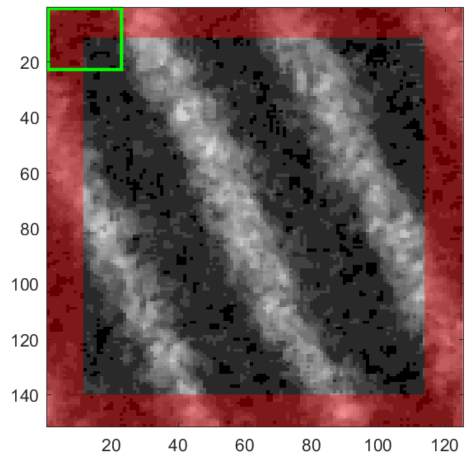

Figure 2-1 – Tile A taken from a SAS image of a sand ripple surface. The green box illustrates the placement of the first box placement in Method 1. The region highlighted in red are pixels for which the point count is not computed.

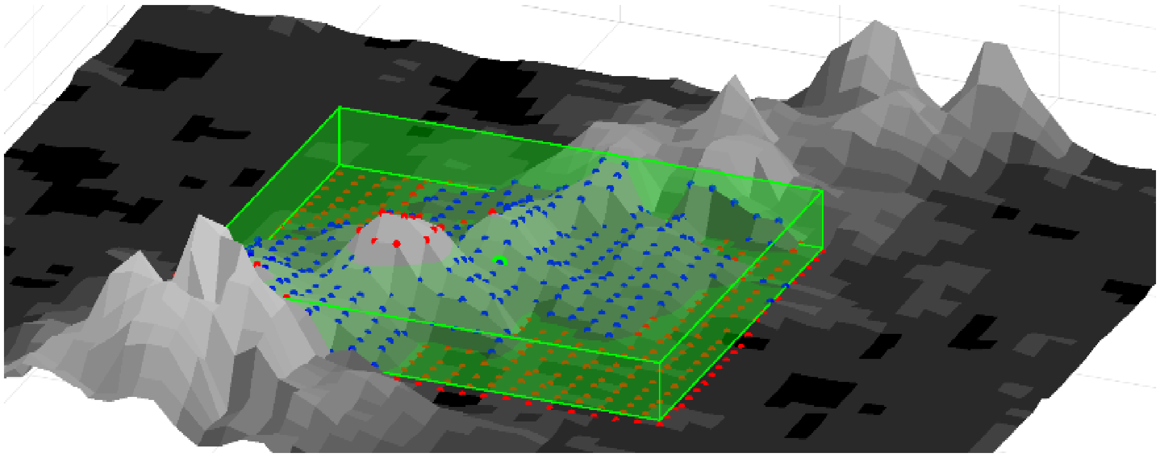

Figure 2-2 – Box placement and point counting on tile A

A box of size L is centered around the point highlighted in green. The red and blue points all fall within the footprint of the box, but only the blue points fall within the box itself. The blue points are counted and that count is stored in a matrix. The box is then moved by one pixel and the process repeats. This point counting happens at every pixel that allows the entire box to fit over the footprint of the image. The square annulus region highlighted in red in Figure 2-1 shows those pixels for which this point counting is not done for the given box size L. This border will always have a width of $floor\left(\frac{L_{depth}}{2}\right)$ at the top and bottom, and $floor\left(\frac{L_{width}}{2}\right)$ at the left and right sides. In Figure 2-1 and Figure 2-2, $L_{width} = 23, y = 151, x = 125$, and from Equations (3) and (4), we are able to calculate L_{height} knowing the z_{max} and z_{min} values for our data. The boxes in the figures appear to have different L_{depth} and L_{width} values because the image pixels are non-square; the depth and width of the box are in fact equal to L_{width} pixels.

Once the point count has been recorded at all possible pixels, the size of the box is increased. Because we have fixed $L_{depth} = L_{width}$, and L_{height} is dependent on L_{width} by Equation 3, we only need to increment L_{width} to grow the box in all three dimensions. We do this for L_{width} going from L_{min} to L_{max} by L_{step}. Only odd integer values are used for L_{width}, so L_{step} is generally set to 2, but may also be set to a multiple of 2. When setting L_{max}, it is important to remember that the larger the size of the box becomes, the more computationally intensive the image processing will be. Considerations for setting the parameters that define L are discussed further in Section 2.4.

Once the point counts are complete, we can let $P(m, L)$ be the probability that m points fall with a box of size L centered around any pixel in the image (not including the square annulus region). Then, for each L we have [7]:

$$\sum_{m=1}^{M} P(m, L) = 1 \qquad (8)$$

with M being the maximum possible number of points that could fall in a box of size L, i.e. $M = L_{depth} * L_{width}$. From this, we find the number of boxes needed to cover the image to be

$$N(L) = y * x * \sum_{m=1}^{M} \left(\frac{1}{m}\right) P(m, L). \tag{9}$$

Because $y * x$ is a constant factor, we take it out and instead set $N(L)$ to be

$$N(L) = \sum_{m=1}^{M} \left(\frac{1}{m}\right) P(m, L). \tag{10}$$

$N(L)$ is then related to the fractal dimension D of the image tile by $N(L) \propto {L_{width}}^{-D}$, or

$$-\ln N(L) = D \ln L + C. \tag{11}$$

If we find the least squares linear fit of points plotted as $\{\ln L_{width}, -\ln N(L)\}$, shown in Figure 2-3, the slope of the resulting line is the fractal dimension D of the image tile and our primary "slope" feature, and the y-intercept is C from Equation (11), which is the logarithm of the proportionality constant and our secondary "intercept" feature. A suitable number of different box sizes is used to obtain a slope that best represents the data.

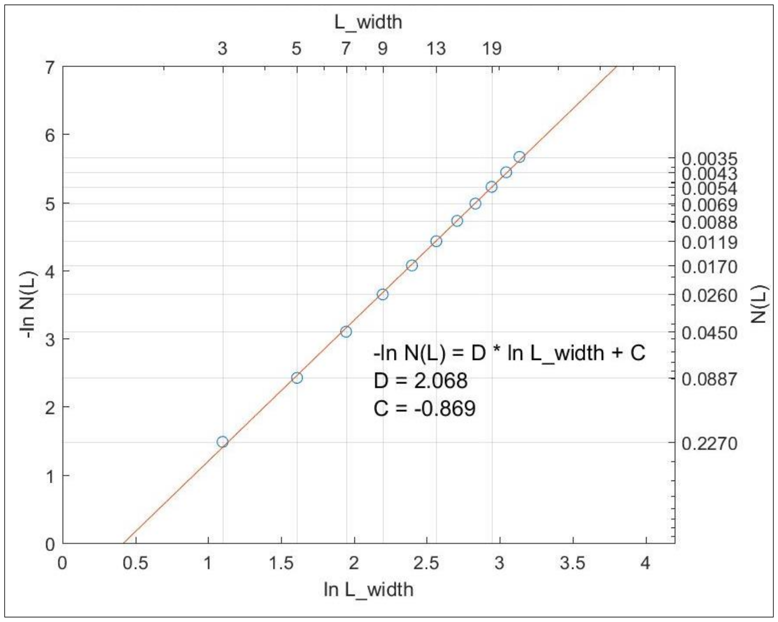

Figure 2-3 – Fractal dimension Method 1 computation on sand ripple tile A from Figure 2-1 with $L_{min} = 3$, $L_{step} = 2$, $L_{max} = 23$

Our experimentation shows that C may in fact be an even more discriminatory feature than D when it comes to classification, as discussed in Section 3.

To distinguish different textures that have the same D and/or C values, we make use of the secondary "lacunarity" feature. As stated in Section 1, the Keller lacunarity and Voss lacunarity features make use of the work already done in Method 1. The definition of lacunarity, K, that Keller [7] introduces and that we term "Keller lacunarity" here is defined as:

$$K(L) = \frac{\Gamma(L) - N(L)}{\Gamma(L) + N(L)} \tag{12}$$

with

$$\Gamma(L) = \sum_{m=1}^{M} mP(m, L). \tag{13}$$

The Voss lacunarity feature, Λ, is then found as:

$$\Lambda(L) = \frac{\Gamma^2(L) - (\Gamma(L))^2}{(\Gamma(L))^2} \tag{14}$$

with

$$\Gamma^2(L) = \sum_{m=1}^{M} m^2 P(m, L). \tag{15}$$

Therefore, there is a separate Keller lacunarity and Voss lacunarity feature computed for each box of size L. When the lacunarity features are computed along with the D and C features, repeated computation is avoided. However, the lacunarity, D, and C values will all be tied to the same values of L where it is possible that useful lacunarity features would result from values of L other than those used to find D and C.

2.2. Method 2

Method 2 is a more simple, intuitive, and classic method for computing fractal dimension. The method works by first specifying a box of size L as in Method 1. Then, the image space is filled with neighboring boxes, as illustrated in Figure 2-4. Because L_{height}

is determined as specified in Equations (3) and (4), we obtain roughly the same number of boxes in the height dimension as in the depth and width dimensions. Boxes must be allowed to go past the edges of the extracted image tile so that at any box size L, every image tile pixel will fall within some box. The number of nonempty boxes, i.e. the number of boxes that contain at least one point and the number of boxes needed to cover the tile, are counted. This count is stored as $N(L)$, and, as in Method 1, $N(L)$ relates to fractal dimension D by Equation (11).

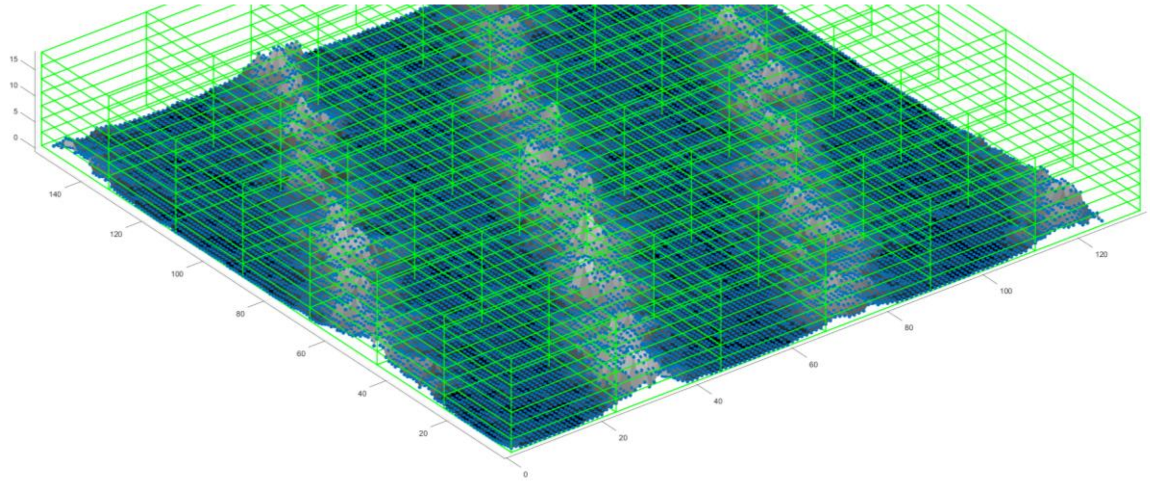

Figure 2-4 – The effect of scaling the box height by L_{hm} is to have roughly the same number of boxes in the height dimension as in the depth and width dimensions

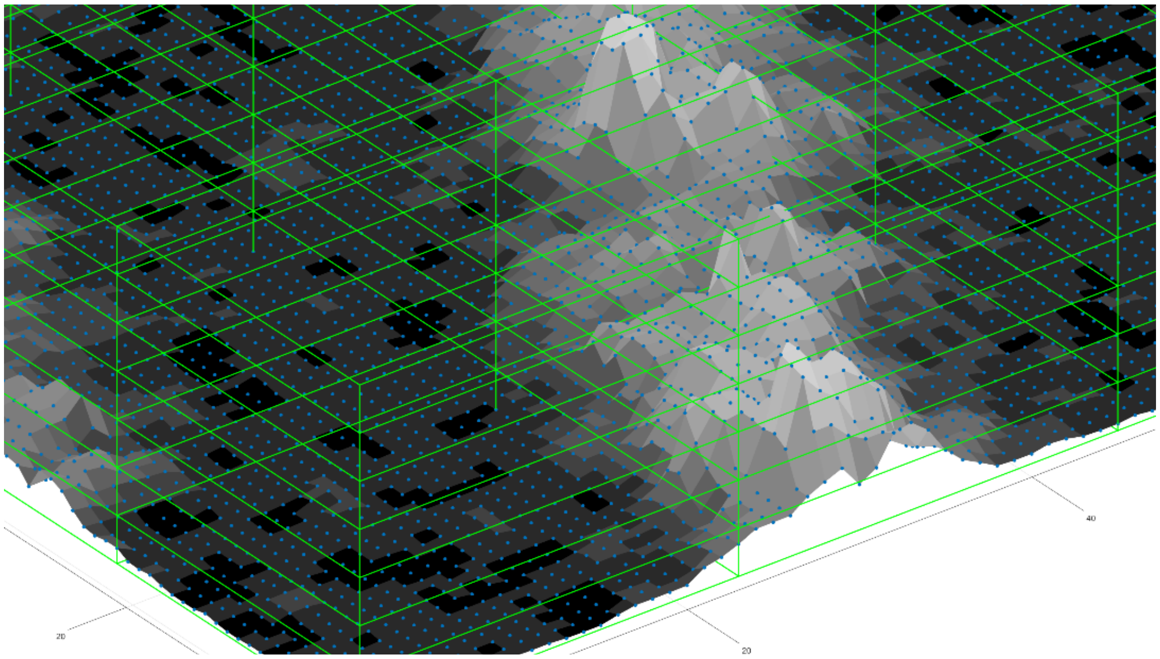

Figure 2-5 – A closer view of the box layout and points to be counted

To obtain D and C, we do the least squares linear fit as described in Method 1 and are able to plot the line and points as shown in Figure 2-6.

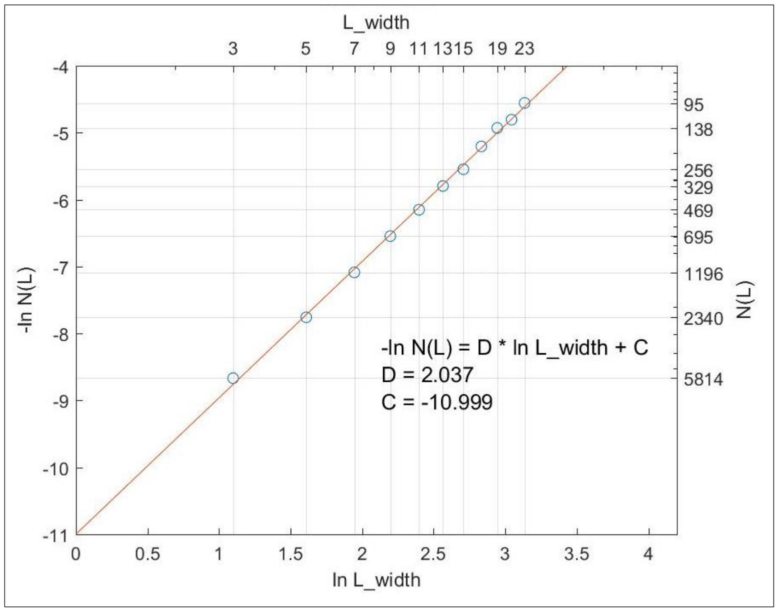

Figure 2-6 – Fractal dimension Method 2 computation on sand ripple tile A from Figure 2-1 with $L_{min} = 3$, $L_{step} = 2$, $L_{max} = 23$

2.3. Method 3

The final method of fractal dimension computation that we implement makes use of a fractional Brownian motion (fBm) model. FBm was initially explored by researchers such as Kolmogorov [15] in 1940 and Mandelbrot and Van Ness [11] in 1968, and described by Crownover [9] in 1995. An fBm is defined by a Hurst parameter H with $0 \leq H \leq 1$. The estimation of fractal dimension for an fBm makes use of the fact that the standard deviation of the increments of an fBm satisfy a power law:

$$std\left(T(i + p_i, j + p_j) - T(i,j)\right) \propto \|(p_i, p_j)\|^H. \tag{16}$$

A two-variable fBm, as in the case of our data, is then related to the fractal dimension by $D = 3 - H$. To determine H for a given image tile, we first find the standard deviation of the differences between pixels in the original image and pixels in a shifted image, with the distance of the shift being determined by a parameter p that is analogous to L in Method 1 and Method 2. Generally, we start with $p = 1$ before incrementing it up to p_{max} by p_{step}. The algorithm proceeds in the following manner, with T being our image tile and $std()$ being the standard deviation function:

```
for $p = 1$ to $p = p_{max}$ by $p_{step}$
    for $i = 1$ to $i = y$
        for $j = 1$ to $j = x - p_{max}$
            $dX(i,j) = T(i, j + p) - T(i,j)$
        end
    end
    $s(p) = std(dX(i,j)), i = 1$ to $i = x, j = 1$ to $j = x - p_{max}$
end
```

Here, the image is shifted to the right (or to the left if you instead set $p < 0$). However, the image can also be shifted vertically or diagonally by making simple modifications to the algorithm. It is important to note that for a diagonal shift where $p = a$, the shift distance is actually $\sqrt{2a^2}$ rather than just a. The slope of the line and Hurst parameter H from Equation (16), and the y-intercept, are then given by

$$\ln s(p) = H \ln p + B. \tag{17}$$

B is a secondary "intercept" and constant of proportionality feature like C in Method 1 and Method 2. While the B and C features are similar, we notate them differently because while C is the intercept of the line for which D is the slope, B is the intercept of the line for which H is the slope. The data points and line can be plotted as $\{\ln p, \ln s(p)\}$ as shown in Figure 2-7. In this case, $p_{max} = 10$ and the image was shifted to the right only.

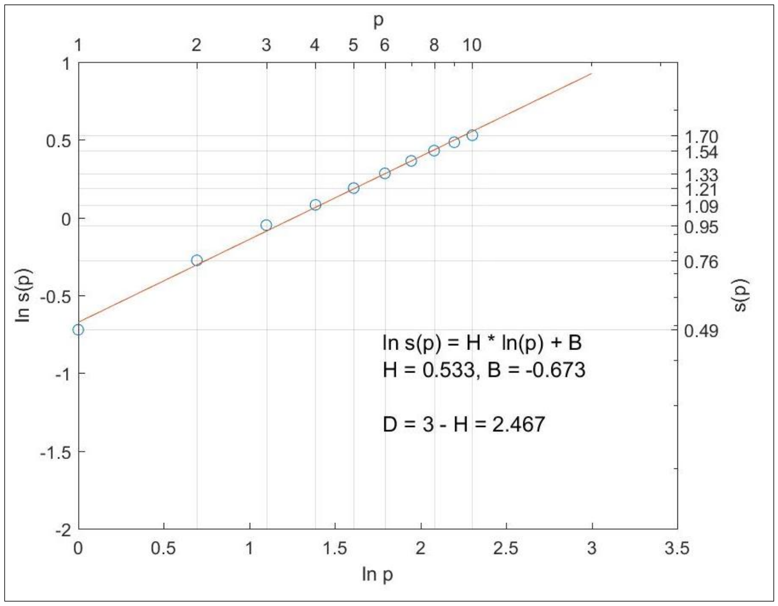

Figure 2-7 – Fractal dimension Method 3 computation sand ripple tile A from Figure 2-1 with $p_{min} = 1$, $p_{step} = 1$, $p_{max} = 10$, and the shift direction set to the right

While the resulting D values for sand ripple tile A from Figure 2-1 are very close when using Method 1 and Method 2 – shown in Figure 2-3 and Figure 2-6, respectively – this result depends largely on the selected parameters, as discussed in Section 2.4. Meanwhile, the Method 3 computation shown in Figure 2-7 for the same image tile shows a very different D when using the fBm model. The experimentation discussed in Section 3.1 show that even the Method 1 and Method 2 results quickly diverge from one another as the theoretical D value grows, although they are both box-counting methods.

2.4. Considerations for Setting L and p Parameters

It is important to note that changing the parameters used to define L (L_{min}, L_{step}, and L_{max}) in Method 1 and Method 2, or p (p_{min}, p_{step}, p_{max}, and direction of shift) in Method 3, may have non-negligible effects on the resulting D, C, and B values. Some of these effects are seen when comparing Figure 2-8 and Figure 2-9 to Figure 2-7.

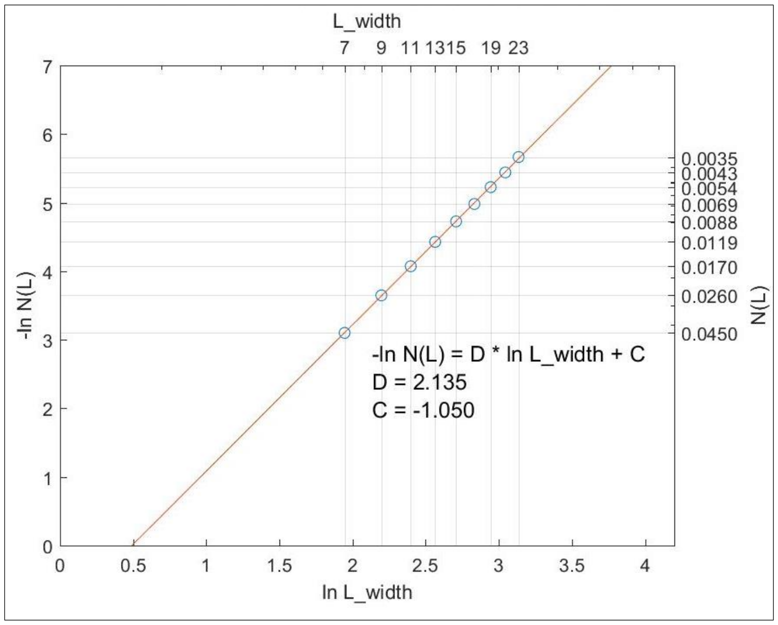

Figure 2-8 – Method 1 computation on sand ripple tile A from Figure 2-1 with $L_{min} = 7$, $L_{step} = 2$, $L_{max} = 23$

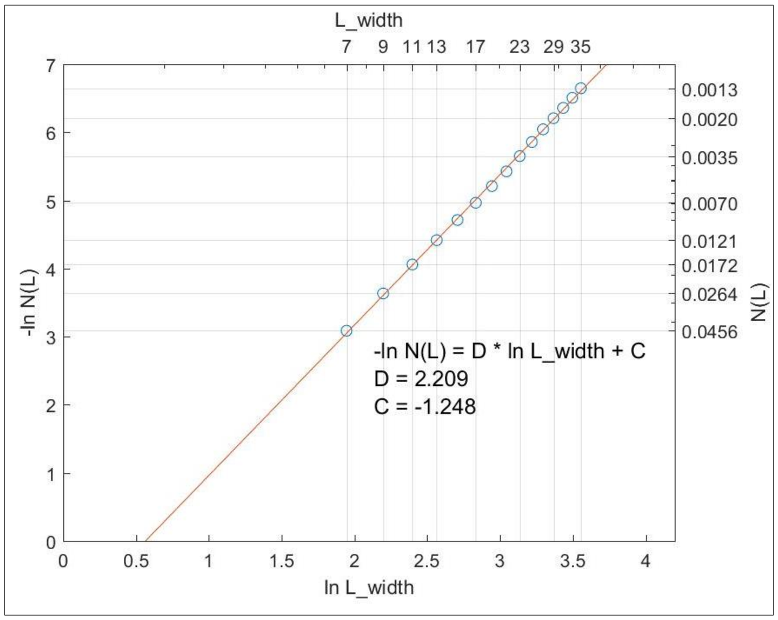

Figure 2-9 — Method 1 computation on sand ripple tile A from Figure 2-1 with $L_{min} = 7$, $L_{step} = 2$, $L_{max} = 35$

In both of these cases, after changing the L_{min} or L_{max} parameters, the resulting D value is higher and C value lower than those computed in Figure 2-7. It is also noted that $N(L)$ differs slightly in Figure 2-9 for those values of L_{width} used also in Figure 2-3 and Figure 2-8. Because L_{max} is larger for the computation in Figure 2-9 than it is for the computation shown in the other two figures, the square annulus region illustrated in Figure 2-1 is then also larger. This changes the $P(m, L)$ values used in Equation (10) and therefore the $N(L)$ values as well. We find that the most stable results are achieved when the parameters are set so that a linear trend is seen in the general case. For example, the point in Figure 2-3 corresponding with $L_{width} = 3$ is less in keeping with the linear trend than the remaining points. For this reason, in this case, we would set $L_{min} = 7$.

In Method 2, $N(1)$ will always be equal to the number of pixels in the tile $y * x$ and $N(y)$ will always be equal to 1 when $x = y$. In our experiments, setting $L_{min} = 1$ or $L_{max} = y$ (or $L_{max} = x$) may change D and C drastically. In Figure 2-10, the same experiment as that shown in Figure 2-6 is run with L_{min} instead set to 1. The additional point is not in keeping with the linear trend of the other points, pushing the slope D of the line out of the theoretical range $2 \leq D \leq 3$.

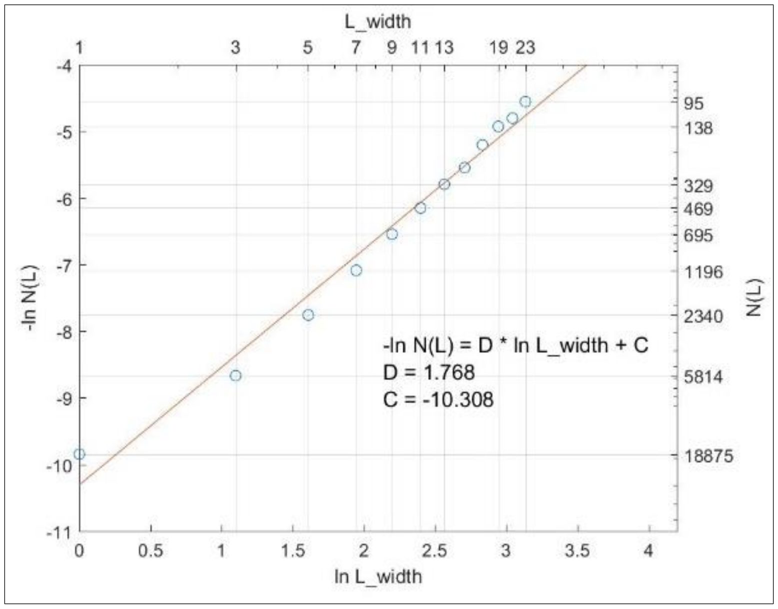

Figure 2-10 – Method 2 computation on sand ripple tile A from Figure 2-1 with $L_{min} = 1$, $L_{step} = 2$, $L_{max} = 23$

In Method 3, p_{step} and p_{max} should be set with some knowledge of the resolution of the tiles in the data set. If a high resolution and a low resolution image tile of the same texture is analyzed with the same p parameters, very different results may be produced, as a shift by t pixels for a high resolution image won't have as much of an effect as the same pixel shift in the low resolution image. Method 3 also has the property that for repetitive textures, such as the sand ripple captured in image tile A (Figure 2-1), the standard deviation of the pixel differences will begin to decrease again when the repetition is encountered. This is shown dramatically in Figure 2-11, where the repetitive texture produces points that violate the theoretical linear model.

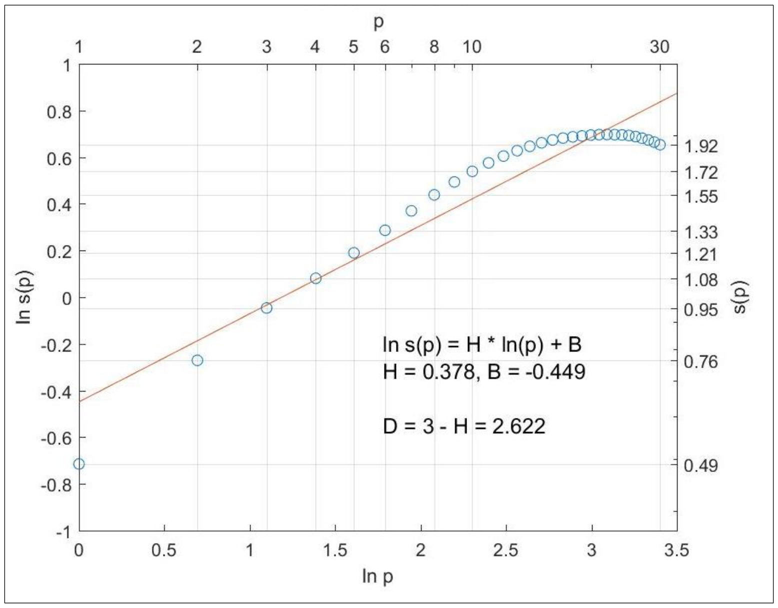

Figure 2-11 – Method 3 computation on the sand ripple tile A from Figure 2-1 with $p_{min} = 1$, $p_{step} = 1$, $p_{max} = 30$, and the shift direction set to the right

With a higher p_{max} value compared to that used in the computation shown in Figure 2-7, the computation shown in Figure 2-11 allows this repetition to pull the slope of the line down, decreasing H and increasing D. For a perfectly repeated surface and with p_{max} sufficiently high, H would eventually go to 0 and D to 3. Allowing the linear fit line to deviate so far from the plotted points is likely to produce unstable results. A change in D can also be seen when p_{max} is fixed but the shift direction is changed, as seen in Figure 2-12 and Figure 2-13 compared to Figure 2-7. This change is expected given that the sand ripple in the image tile runs up and to the right. This observation shows the effect that texture orientation can have on the produced feature values.

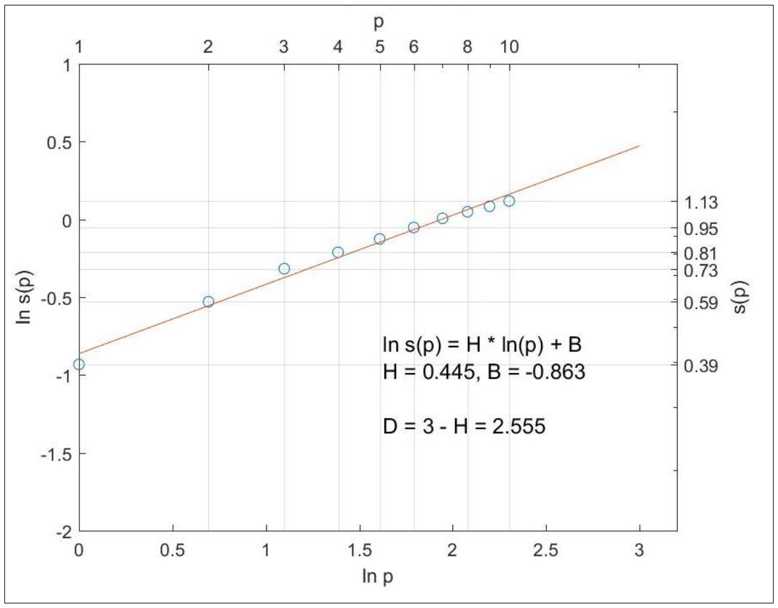

Figure 2-12 – Method 3 computation on sand ripple tile A from Figure 2-1 with $p_{min} = 1$, $p_{step} = 1$, $p_{max} = 10$, and the shift direction set to up

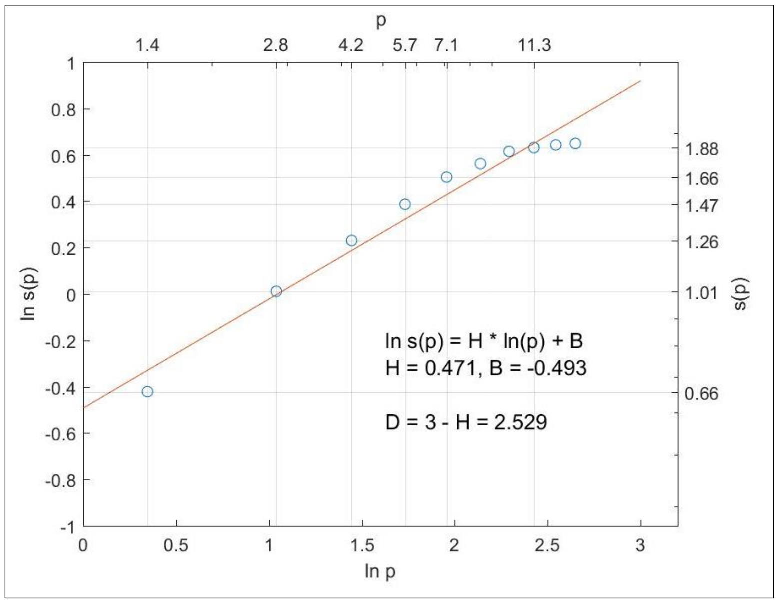

Figure 2-13 – Method 3 computation on sand ripple tile A from Figure 2-1 with $p_{min} = 1$, $p_{step} = 1$, $p_{max} = 10$, and the shift direction set to up and to the right

Noting the effects that tweaking the L and p parameters can have, it is obvious that parameters must be kept consistent when extracting features from a set of image tiles. We find that setting the parameters so that we obtain points plotted with as linear a trend as possible results in the most stable, and generally most desirable, results.

2.5. Williams Lacunarity

While the Keller lacunarity and Voss lacunarity features are computed along with D and C, the Williams lacunarity feature does not make use of the same computations. Williams's lacunarity, Υ, is simply defined as [14]:

$$\Upsilon = \frac{\sigma^2}{\mu^2} \tag{18}$$

where σ^2 is the variance of the pixel values being evaluated and μ is the mean of the same pixel values. We implement this using a sliding window across the tile and finding the mean Υ value across all windows. The size of the sliding window has dimensions L_{depth} x L_{width} like the boxes from Method 1 and Method 2. Similar to the other lacunarity features, this is done for different window sizes to produce multiple Williams lacunarity features. Williams's definition of lacunarity does not directly match Mandelbrot's [13], but it is a measure of how intensity distribution relates to the window size.

3. DATA SETUP, EXPERIMENTATION, AND RESULTS

3.1. Testing on Artificial Surfaces

To test our three methods of fractal dimension computation, we generated artificial surfaces, using an fBm model, for which the fractal dimension was predefined. The code [16] used for this purpose generates a different surface each time it is run. We generated ten surfaces for each value of $H \in \{0.0, 0.1, \dots, 1.0\}$ to obtain surfaces with $D \in \{3.0, 2.9, \dots, 2.0\}$, respectively. We set $y = x = 100$ for all surfaces. Examples are shown in Figure 3-1, Figure 3-2, and Figure 3-3.

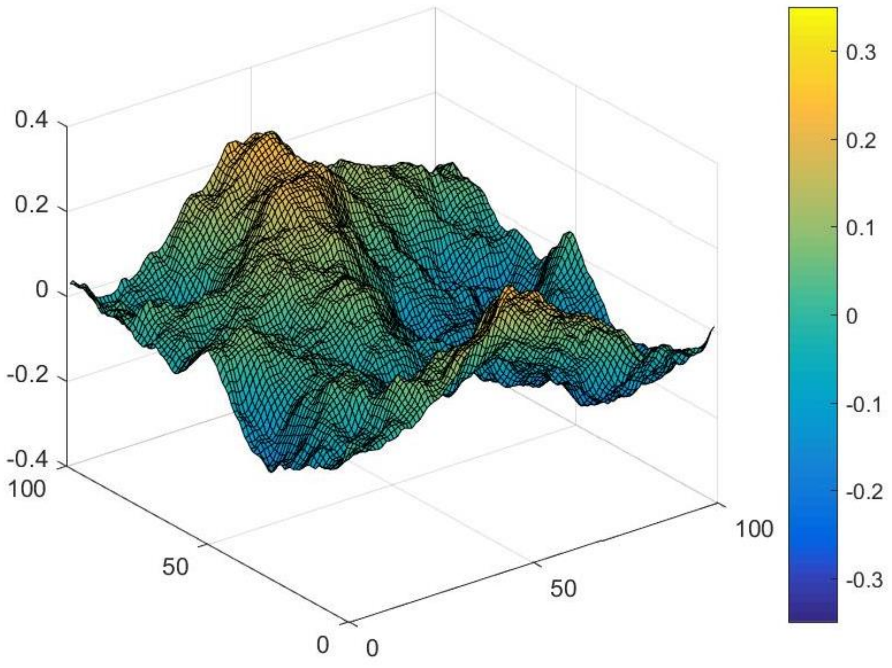

Figure 3-1 – An artificially generated surface with $H = 1.0$ ($D = 2.0$)

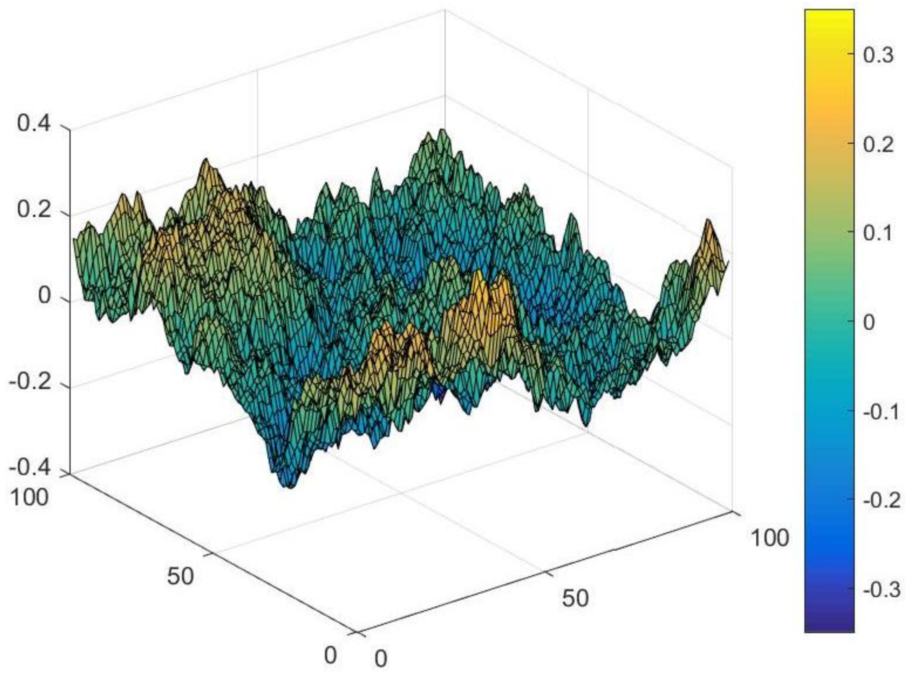

Figure 3-2– An artificially generated surface with $H = 0.5$ ($D = 2.5$)

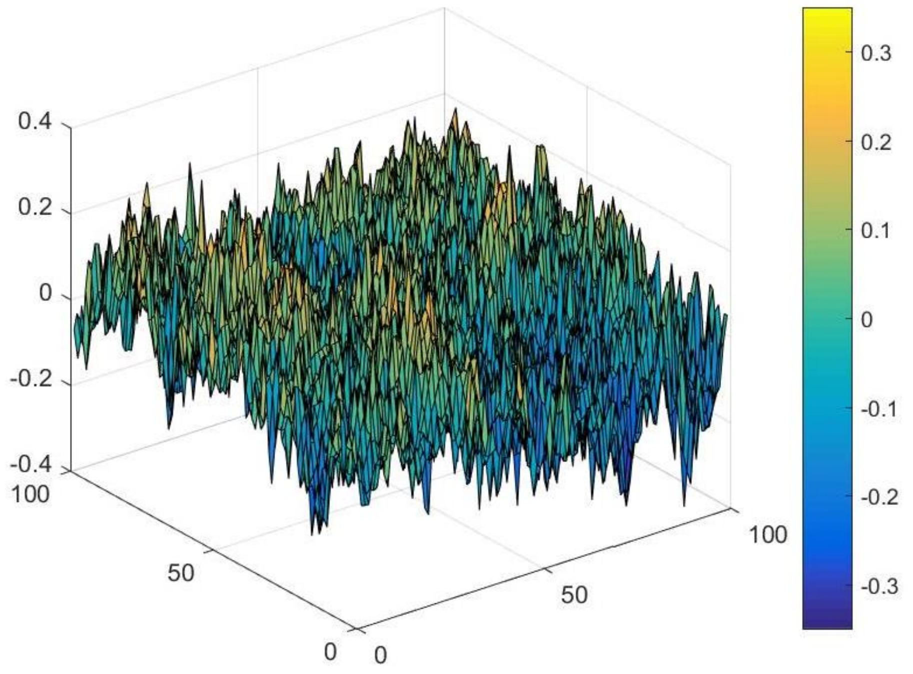

Figure 3-3– An artificially generated surface with $H = 0.0$ ($D = 3.0$)

We determined the z_{min} and z_{max} values for the entire artificial data set and then computed D for each of the surfaces using all three methods. After considering the issues discussed in section 2.4 and doing some experimentation, we set the parameters as shown in Table 3-1.

Table 3-1 – Parameters used in experimentation

	L_{min}	L_{max}	L_{step}	p_{min}	p_{max}	p_{step}	shift direction
Method 1	7	31	2				
Method 2	7	23	2				
Method 3				1	10	1	right only

We obtained the following results.

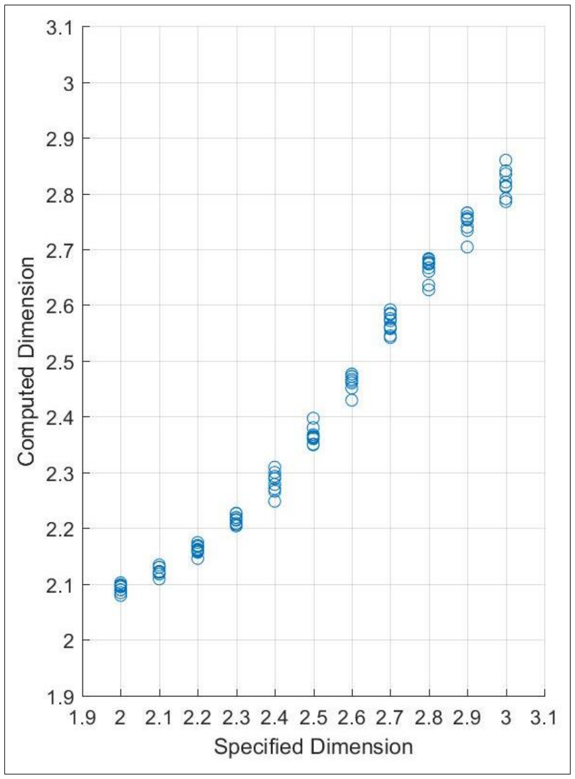

Figure 3-4 – Method 1 computed D on 100 artificial surfaces

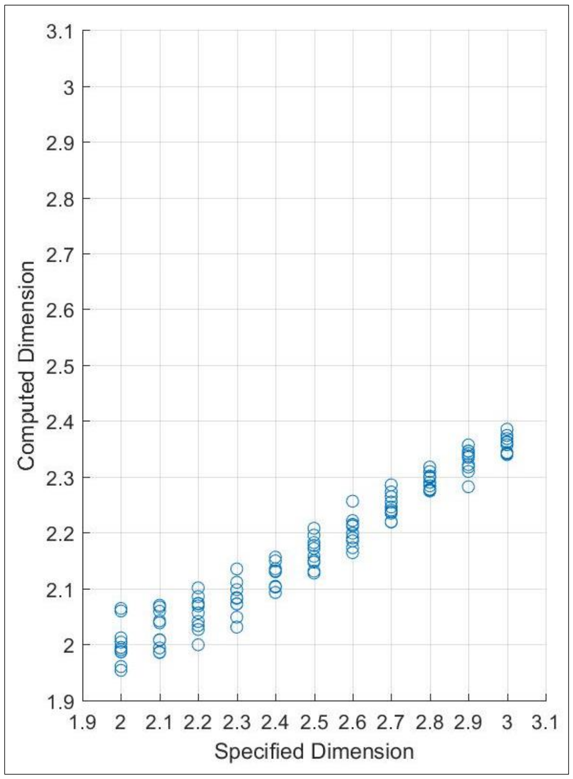

Figure 3-5 – Method 2 computed D on 100 artificial surface

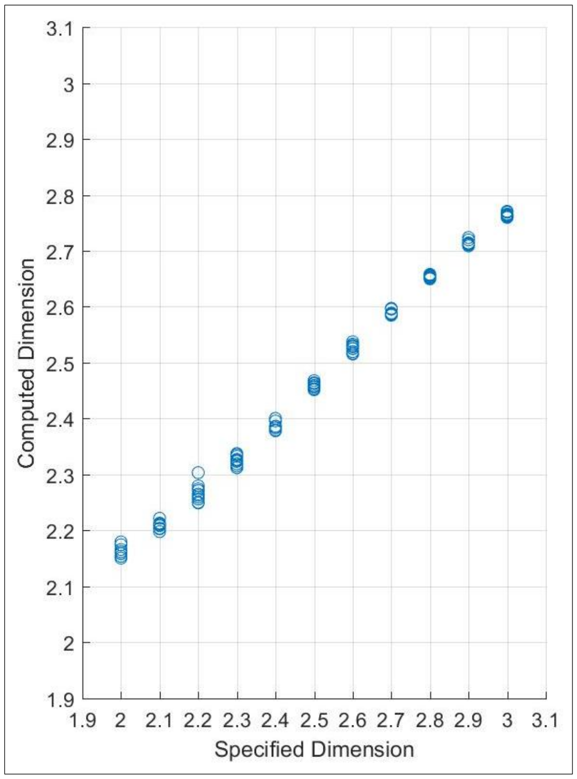

Figure 3-6 – Method 3 computed D on 100 artificial surfaces

Method 3 shows the tightest clustering of computed dimensions per specified dimension, perhaps indicating higher reliability. The generating model in this case was, after all, an fBm as discussed in Section 2.3. The resulting points for Method 2 have a much greater spread at each specified dimension and overall appear to be scaled down from the specified dimensions. From these tests on artificial surfaces, and assuming that the artificial surfaces do in fact have the specified theoretical fractal dimension, it is clear that none of the three methods will always find the theoretical value of D, or in some cases even be close. However, the estimated fractal dimension trend is obvious for all three methods. This encourages us to believe that when the same computations are run on real data, the results should be separable, at least in the cases of Method 1 and Method 3, and suitable for classification.

3.2. Data Setup

The real data set used in our experimentation was provided to us by the U.S. Naval Surface Warfare Center as part of an Office of Naval Research project. This data is made up of SAS imagery of the seafloor in different regions of the world. The dataset is comprised of high frequency (HF) and low frequency (LF) imagery with each HF image having a corresponding LF image, and vice versa. The dataset includes many different background seafloor textures including hard-packed sand, sand ripple, sea grass, rocky surface, coral, and more. A single image will typically include between one and three different textures. Examples of some of these textures seen in HF images are shown in Figure 3-7.

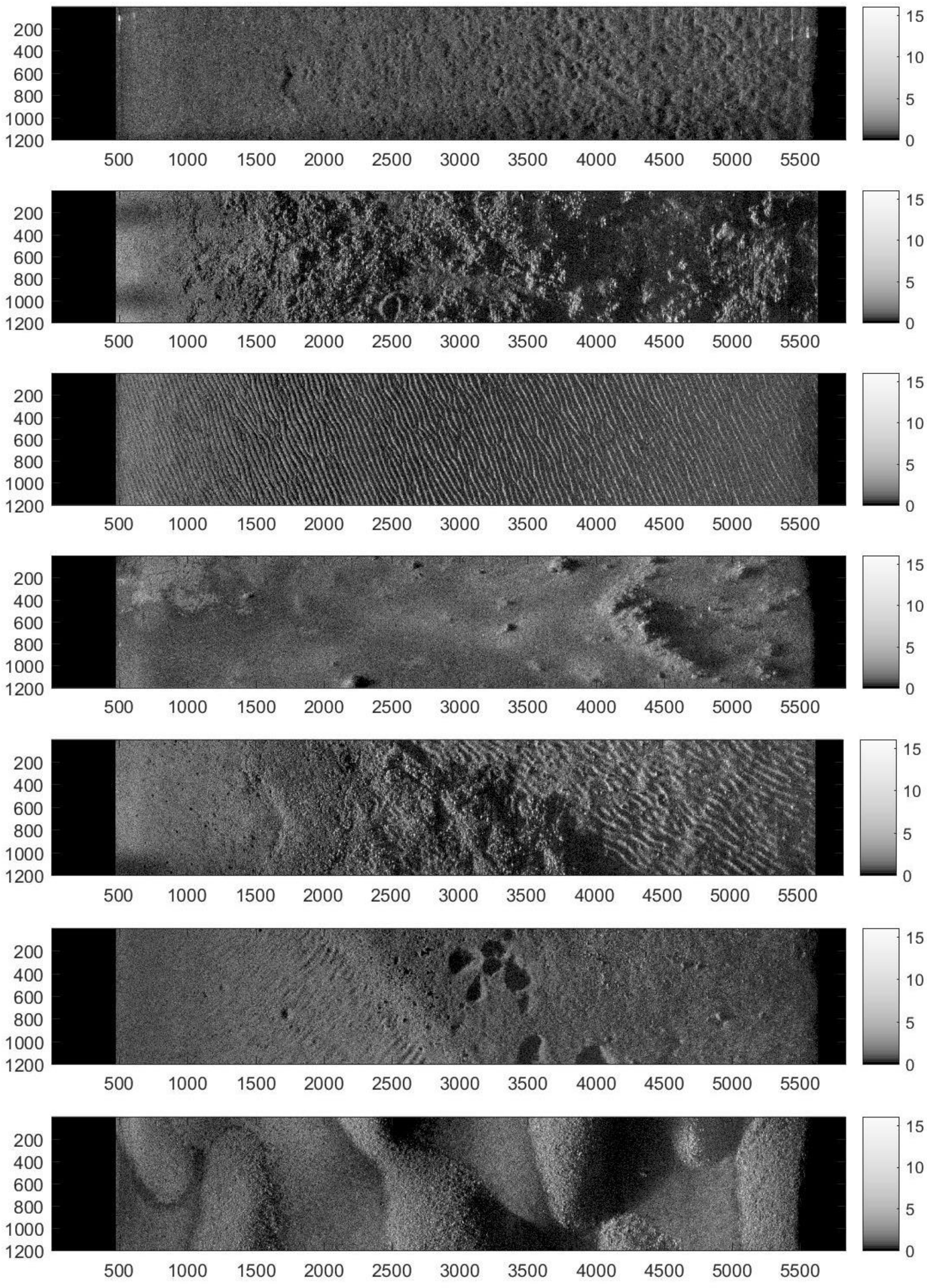
Figure 3-7 – Examples of various background textures seen in HF images

It is worth noting that, looking at the images shown in Figure 3-7 where the path of the vehicle is located on the left-hand side, one can generally see that the further from the sensor an image tile is, the more acoustic shadows will be seen. Closer to the vehicle, where the sonar signal travels between the seafloor and vehicle at a steeper angle, there is little to no occlusion of further surface features by closer surface features. If this has an impact on the extracted features, it should be seen when we analyze the effect that the distance from the image sensor has on the extracted features as discussed in Section 3.6.

The corresponding LF images tend to be much grainier with image details being more difficult for humans to decipher. For comparison, the same ground areas shown in the last two image strips in Figure 3-7 are also shown in Figure 3-8 from the corresponding LF images.

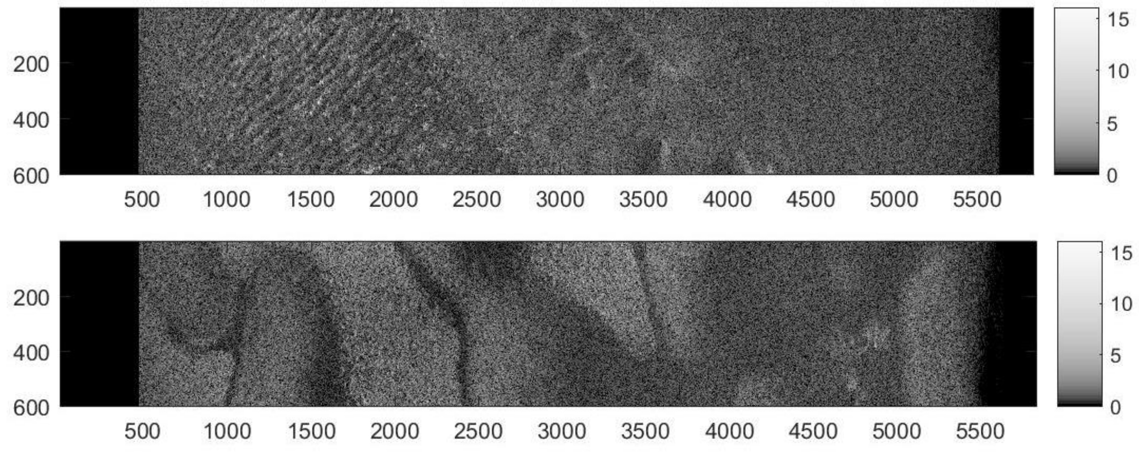

Figure 3-8 – Examples of background textures seen in LF images

Some images also show anomalies considered as targets. Ground truth coordinates were used to extract tiles around all anomalies marked as targets, except in the case of a few that were too close to the edge of the image to center a tile around. This tile extraction gave us 277 pairs of HF and LF target tiles from a variety of different background textures. We then created four separate sets of corresponding false alarm tiles. For each pair of target tiles, four different nearby false alarm hits were randomly selected and extracted as tiles. Each of these four tiles was added to a distinct set, with each set containing 277 pairs of HF and LF tiles when complete. This way, the same amount of different background texture types found in the set of target tiles would be included in each set of false alarm tiles. In addition, a balanced data set can be created by pairing the set of target tiles with any of the four sets of false alarm tiles. The false alarm hits were generated by a "combined" Reed-Xiaoli (RX) prescreener described by Galusha et al [17]. We designate the false alarm tile sets as "FA Set 0," "FA Set 1," "FA Set 2," and "FA Set 3."

It is important to note that the pixels in these images are non-square, and that the pixel heights vary slightly between images. Additionally, the LF imagery effectively has half the resolution, along the path of the vehicle collecting the data, as compared to the HF imagery. The non-square pixels and different image pixel sizes were considered when extracting the image tiles. The same seafloor size – 1.5 m x 1.5 m – was used for every extracted tile, resulting in slightly different y values across the image tiles having pixel dimensions y x x. To fix y so that we can find L_{hm} using Equation (4), we use the mean value of y from all of the HF tiles in our data set.

After creating our tile data set, we set the parameters for our experiments to the same as those given in Table 3-1. We then extracted the features listed in Table 3-2 from each pair of tiles.

Table 3-2 – Features extracted from tiles

1	Method 1, D (slope), HF	7	Method 1, D (slope), LF
2	Method 1, C (intercept), HF	8	Method 1, C (intercept), LF
3	Method 2, D (slope), HF	9	Method 2, D (slope), LF
4	Method 2, C (intercept), HF	10	Method 2, C (intercept), LF
5	Method 3, D (slope), HF	11	Method 3, D (slope), LF
6	Method 3, B (intercept), HF	12	Method 3, B (intercept), LF

13	Keller lacunarity, HF, L_width = 07	48	Keller lacunarity, LF, L_width = 07
14	Keller lacunarity, HF, L_width = 09	49	Keller lacunarity, LF, L_width = 09
15	Keller lacunarity, HF, L_width = 11	50	Keller lacunarity, LF, L_width = 11
16	Keller lacunarity, HF, L_width = 13	51	Keller lacunarity, LF, L_width = 13
17	Keller lacunarity, HF, L_width = 15	52	Keller lacunarity, LF, L_width = 15
18	Keller lacunarity, HF, L_width = 17	53	Keller lacunarity, LF, L_width = 17
19	Keller lacunarity, HF, L_width = 19	54	Keller lacunarity, LF, L_width = 19
20	Keller lacunarity, HF, L_width = 21	55	Keller lacunarity, LF, L_width = 21
21	Keller lacunarity, HF, L_width = 23	56	Keller lacunarity, LF, L_width = 23
22	Keller lacunarity, HF, L_width = 25	57	Keller lacunarity, LF, L_width = 25
23	Keller lacunarity, HF, L_width = 27	58	Keller lacunarity, LF, L_width = 27
24	Keller lacunarity, HF, L_width = 29	59	Keller lacunarity, LF, L_width = 29
25	Keller lacunarity, HF, L_width = 31	60	Keller lacunarity, LF, L_width = 31
26	Voss lacunarity, HF, L_width = 07	61	Voss lacunarity, LF, L_width = 07
27	Voss lacunarity, HF, L_width = 09	62	Voss lacunarity, LF, L_width = 09
28	Voss lacunarity, HF, L_width = 11	63	Voss lacunarity, LF, L_width = 11
29	Voss lacunarity, HF, L_width = 13	64	Voss lacunarity, LF, L_width = 13
30	Voss lacunarity, HF, L_width = 15	65	Voss lacunarity, LF, L_width = 15
31	Voss lacunarity, HF, L_width = 17	66	Voss lacunarity, LF, L_width = 17
32	Voss lacunarity, HF, L_width = 19	67	Voss lacunarity, LF, L_width = 19
33	Voss lacunarity, HF, L_width = 21	68	Voss lacunarity, LF, L_width = 21
34	Voss lacunarity, HF, L_width = 23	69	Voss lacunarity, LF, L_width = 23
35	Voss lacunarity, HF, L_width = 25	70	Voss lacunarity, LF, L_width = 25
36	Voss lacunarity, HF, L_width = 27	71	Voss lacunarity, LF, L_width = 27
37	Voss lacunarity, HF, L_width = 29	72	Voss lacunarity, LF, L_width = 29
38	Voss lacunarity, HF, L_width = 31	73	Voss lacunarity, LF, L_width = 31
39	Williams lacunarity, HF, L_width = 05	74	Williams lacunarity, LF, L_width = 05
40	Williams lacunarity, HF, L_width = 07	75	Williams lacunarity, LF, L_width = 07
41	Williams lacunarity, HF, L_width = 09	76	Williams lacunarity, LF, L_width = 09
42	Williams lacunarity, HF, L_width = 11	77	Williams lacunarity, LF, L_width = 11
43	Williams lacunarity, HF, L_width = 13	78	Williams lacunarity, LF, L_width = 13
44	Williams lacunarity, HF, L_width = 15	79	Williams lacunarity, LF, L_width = 15
45	Williams lacunarity, HF, L_width = 17	80	Williams lacunarity, LF, L_width = 17
46	Williams lacunarity, HF, L_width = 19	81	Williams lacunarity, LF, L_width = 19
47	Williams lacunarity, HF, L_width = 21	82	Williams lacunarity, LF, L_width = 21

After computing the features listed in Table 3-2, we are able to create histograms of the resulting feature values to compare how separable the features computed on false alarm tiles are from the features computed on targets tiles, and to get a sense of how useful a particular feature might be in classification. These histograms for features 1-12 extracted from the set of target tiles and FA Set 1 are shown in Figure 3-9.

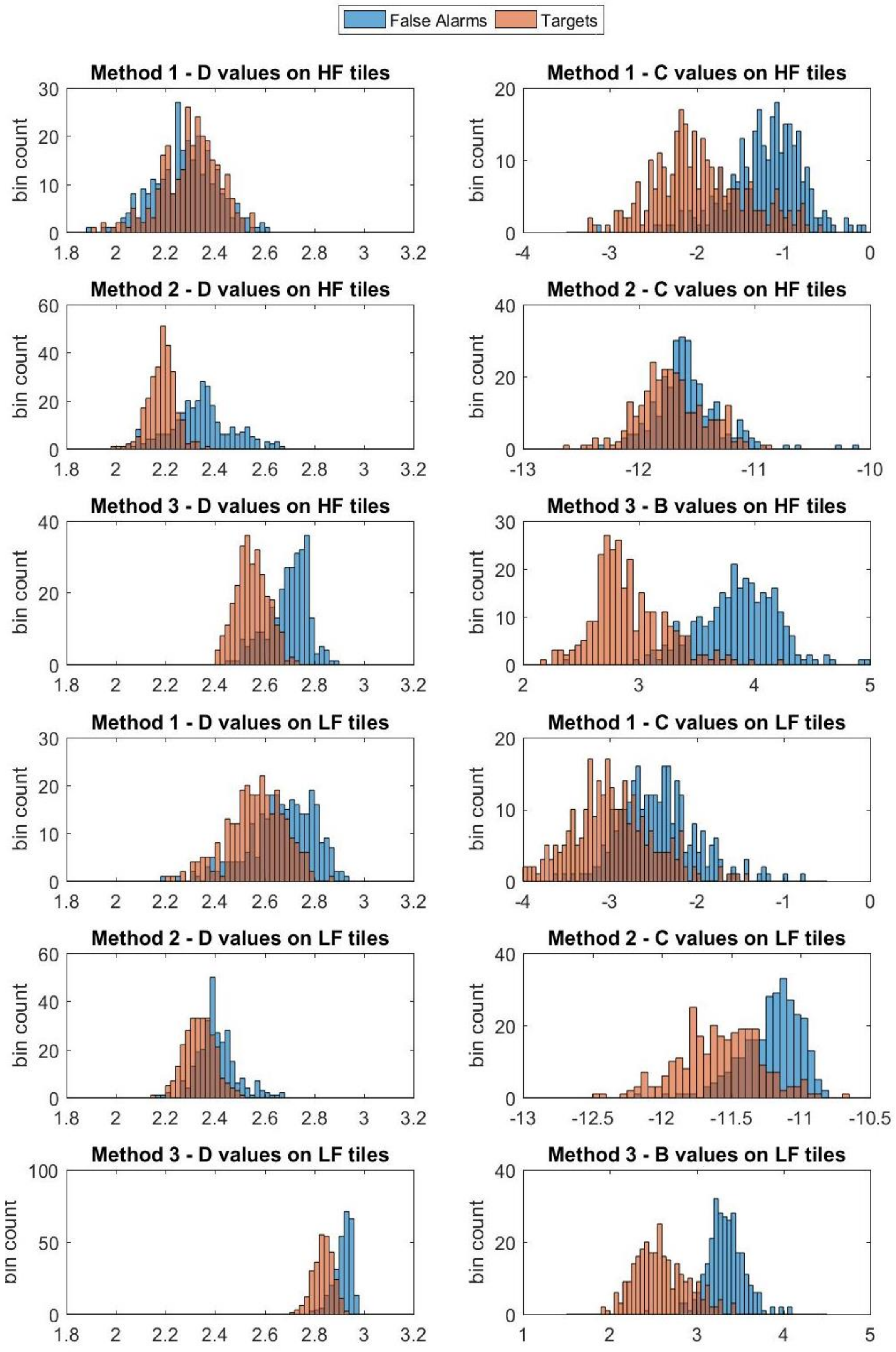

Figure 3-9 – Histograms of computed slope and intercept features FA Set 1, false alarms vs. targets

It is clear that in some cases, such as the D values from Method 1, the features computed on the false alarm tiles are not very separable from those computed on target tiles. However, in other cases, including some of the secondary C and B features, the values computed do appear to be highly separable. While this may seem counterintuitive, Keller et al [10] showed that at least for the fBm model, the constant of proportionality is in fact related to the fractal dimension, although it will vary with image scale. This is not a significant issue for SAS image construction since the vehicle collecting signals is roughly the same distance away from each tile of seafloor. It is then reasonable to say that the constant of proportionality C found in the box counting methods must also be related to the fractal dimension.

Histograms for the remaining 70 lacunarity features listed in Table 3-2 from FA Set 1 and all 82 features from FA Set 2 and FA Set 3 can be found in Appendix A.

3.3. Feature Selection

To test the utility of our extracted features in practice, we did a few simple classification tests using MATLAB’s built-in Classification Learner app. After briefly testing all of the classifiers found in the app using 10-fold cross validation on our library of false alarm and target tiles, we found that the support vector machine (SVM) classifiers generally resulted in the highest classification accuracies. We setup fourteen separate experiments for comparison, and in each experiment included different features in the feature vectors used for classification: All features in this experiment were extracted from the set of target tiles and FA Set 0.

Table 3-3 – Features included in classification experiment

Experiment Designation	Features Included in Feature Vector (from Table 3.2-1)
A	1-12 (all excluding lacunarity)
B	1, 2, 7, 8 (all from Method 1)
C	3, 4, 9, 10 (all from Method 2)
D	5, 6, 11, 12 (all from Method 3)
E	1, 3, 5 (D from all methods on HF imagery)
F	7, 9, 11 (D from all methods on LF imagery)
G	2, 4, 6 (C and B from all methods on HF imagery)
H	8, 10, 12 (C and B from all methods on LF imagery)
I	All lacunarity
J	Selected lacunarity
K	1-12 and selected lacunarity
L	5, 6, 11, 12 (all from Method 3) and selected lacunarity
M	All Williams lacunarity only
N	5, 6, 11, 12 (all from Method 3) and 39-47, 74-82 (all of Williams lacunarity)

Experiment N is significant because it includes all features that do not involve the more time-consuming computations of Method 1 and Method 2, and can be computed with relatively little time. We then ran 10-fold cross validation training and testing runs using each of the SVM classifiers, as well as the cubic k-nearest neighbors (KNN) classifier, and achieved the classification accuracies given in Table 3-4.

Table 3-4 – Classification accuracies (given as %) for one trial

		Experiment Designations (from Table 3.2-2)													
		A	B	C	D	E	F	G	H	I	J	K	L	M	N
Classifier	Linear SVM	94.6	94.6	93.0	94.6	88.8	92.8	91.9	92.6	96.8	96.6	97.8	97.1	92.2	96.4
	Quadratic SVM	95.8	94.4	93.5	94.9	86.8	93.3	92.6	92.6	97.1	96.8	97.3	97.3	94.2	96.4
	Cubic SVM	94.6	92.6	93.0	93.9	88.8	92.4	93.0	92.1	97.1	97.5	97.1	97.1	95.1	96.2
	Fine Gaussian SVM	93.1	92.2	91.9	93.5	89.4	92.8	93.1	92.2	94.9	95.7	91.3	93.9	90.6	93.5
	Medium Gaussian SVM	95.5	94.2	93.3	94.8	87.9	93.5	92.1	93.7	96.8	96.8	97.3	96.4	90.6	95.8
	Coarse Gaussian SVM	94.8	94.8	92.8	94.9	88.1	92.8	91.2	92.6	95.7	95.3	95.7	95.8	88.1	95.3
	Cubic KNN	95.7	93.7	92.8	94.8	88.6	93.9	90.8	92.4	95.8	96.6	96.2	96.2	90.4	96.0
	Average	*94.9*	*93.8*	*92.9*	*94.8*	*88.6*	*93.9*	*90.8*	*92.4*	*96.3*	*96.5*	*96.1*	*96.3*	*91.6*	*95.7*

It can be seen here that the highest average classification accuracy (highlighted in blue) comes from using selected lacunarity only. The selected lacunarity features are those fewer lacunarity features that visually separate targets and false alarms well in the histograms in Appendix A. Referencing Table 3-2, these are features 13-14, 26-29, 44-47, 48-49, 61-64, and 77-82. It is notable that this highest classification accuracy is achieved using no fractal dimension slope or intercept features, but rather the secondary lacunarity features only. This observation lends credence to the idea that lacunarity is just as, or perhaps more of, a discriminatory feature as the others. Experiments A through H rely only on the twelve slope and intercept features. The highest classification among these comes from using all of those twelve features from all three methods while the worst performing classification (highlighted in yellow) comes from using only the D values from the HF imagery. Confusion matrices for these two cases are given in Table 3-5 and Table 3-6.

Table 3-5 – Confusion matrix for Experiment A (using all features from all three methods) and the Quadratic SVM classifier

True Class	False Alarm	264	13
	Target	10	267
		False Alarm	Target
		Predicted Class	

Table 3-6 – Confusion matrix for Experiment E (using ***D*** values from the three methods on HF imagery) and the Quadratic SVM classifier

True Class	False Alarm	232	45
	Target	28	249
		False Alarm	Target
		Predicted Class	

Importantly, it can also be seen that the Method 3 features (Experiment D) perform as good as or better than the Method 1 (Experiment B) and Method 2 (Experiment C) features for every tested classifier, and only ~0.1% worse than all three methods combined. In addition, for all three methods combined, in the case of the HF imagery, the secondary C and B features (Experiment G) gave a higher average classification accuracy than the primary D feature (Experiment E). This was reversed for the LF imagery. Interestingly, the features from the LF imagery gave higher accuracies than those from the HF imagery. While these results are encouraging, it is necessary to evaluate these features in region-by-region cross-validation experiments that test on areas of seafloor with unique textures while training exclusively on the remaining textures. This experimentation is discussed in Section 3.4.

It is believed that the method used by the MATLAB Classification Learner app for cross-validation selects the tiles to include in each fold in a random or pseudo random manner each time a classification experiment is run. As a result, a slightly different classification accuracy is obtained each time the same experiment is run. For this reason, we treat the single-trial results shown in Table 3-4 as indicators only. To test the stability of the classification using these classifiers trained and tested on different folds of data, we repeat Experiment J and run 30 trials instead of one. We show the average classification accuracy and standard deviation for each classifier in Table 3-7.

Table 3-7 – Average and standard deviation of classification accuracy across 30 trials for different classifiers using selected lacunarity features

		Average Classification Accuracy	Standard Deviation of Classification Accuracy
Classifier	Linear SVM	94.8%	0.2%
	Quadratic SVM	95.8%	0.5%
	Cubic SVM	97.0%	0.3%
	Fine Gaussian SVM	95.4%	0.4%
	Medium Gaussian SVM	96.9%	0.3%
	Coarse Gaussian SVM	96.5%	0.2%
	Cubic KNN	97.3%	0.4%
	Average	*96.2%*	*0.3%*

The first two numerical columns in Table 3-10 show results with the same experimental setup instead using the Experiment D (Method 3) features.

To determine which features should be used in our region-by-region cross-validation experiment, we do a forward search feature selection to find the combination of features that is likely to give the highest classification accuracies when applied in a real-world setting.

We set up the forward search as follows using our set of target image tiles balanced with the set containing an equal number of false alarm tiles: Using MATLAB’s quadratic SVM classifier, we do 10-fold cross-validation using each feature individually. Because of MATLAB’s method for selecting the folds, we do five trials for each and average the classification accuracies of the trials together. We select the feature resulting in the highest average classification accuracy. We then repeat the process but pair the previously selected feature with each of the individual features to again find the highest average classification accuracy. The process is repeated for the desired number of selected feature – in our case, six. Numerical results from this experiment are given in Appendix B.

For FA Set 1, the resulting six selected features are those numbered 62, 38, 41, 16, 43, and 42 in Table 3-2, selected in that order. We designated these as our “Forward Search Features.” It is again noted here that all of these features are lacunarity features. A 98.4% average classification accuracy was achieved with these six features, higher than any of the classification accuracies shown in Table 3-4. When the same forward search experiment was instead run with set of target tiles and FA Set 0, the top six features were 63, 37, 18, 44, 13, and 47 – all very similar features. The same 98.4 % average classification accuracy was achieved.

3.4. Region-by-Region Cross-Validation Experiment

For a system being deployed to any part of the many oceans and seas covering the earth, it cannot be assumed that all textures that the system will encounter will be available beforehand for training purposes. Therefore, it is important to test our features by training a classifier on some textures and testing on others. Because the data available to us is from seven different regions around the world, with each region having characteristic textures, we are able to do this by separating our training and testing folds by region. Using our set of 277 target image tile pairs and three corresponding sets of false alarm tiles, we train on those tiles from six of the seven regions and then test on the seventh. This is done with each of the seven regions held out individually. The number of HF target tiles, LF target tiles, HF false alarm tiles, and LF false alarm tiles in each respective fold (the total number of tiles in each region is four times that given in the table) is given in Table 3-8.

Table 3-8 – Number of target and false alarm tile in each region

Region A	45
Region B	4
Region C	33
Region D	65
Region E	35
Region F	50
Region G	45
Total	*277*

We make use of MATLAB's 'fitcsvm', 'fitPosterior', and 'predict' functions [17] [18]. 'fitcsvm' takes a specified kernel function as a parameter. We test the 'Gaussian' and 'linear' kernel functions with our Forward Search Features. We run three trials for each experiment, with the first trial using FA Set 1, the second trial using FA Set 2, and the third trial using FA Set 3. We obtain the following ROC curves from the output confidence values and show them in the following figures.

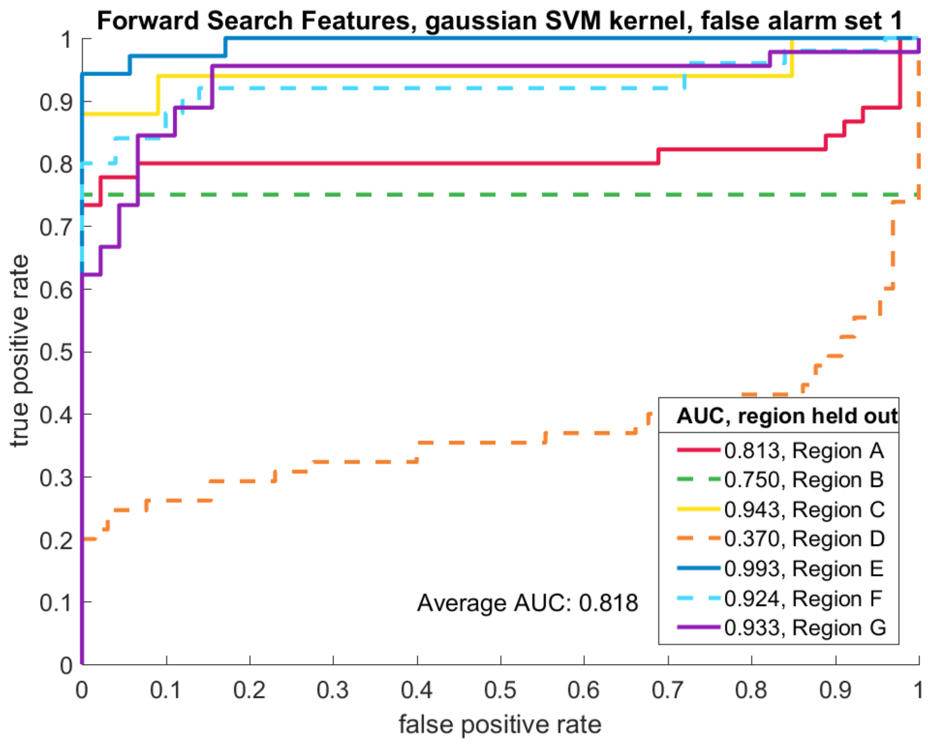

Figure 3-10 – ROC using Forward Search Features, Gaussian SVM kernel, and FA Set 1

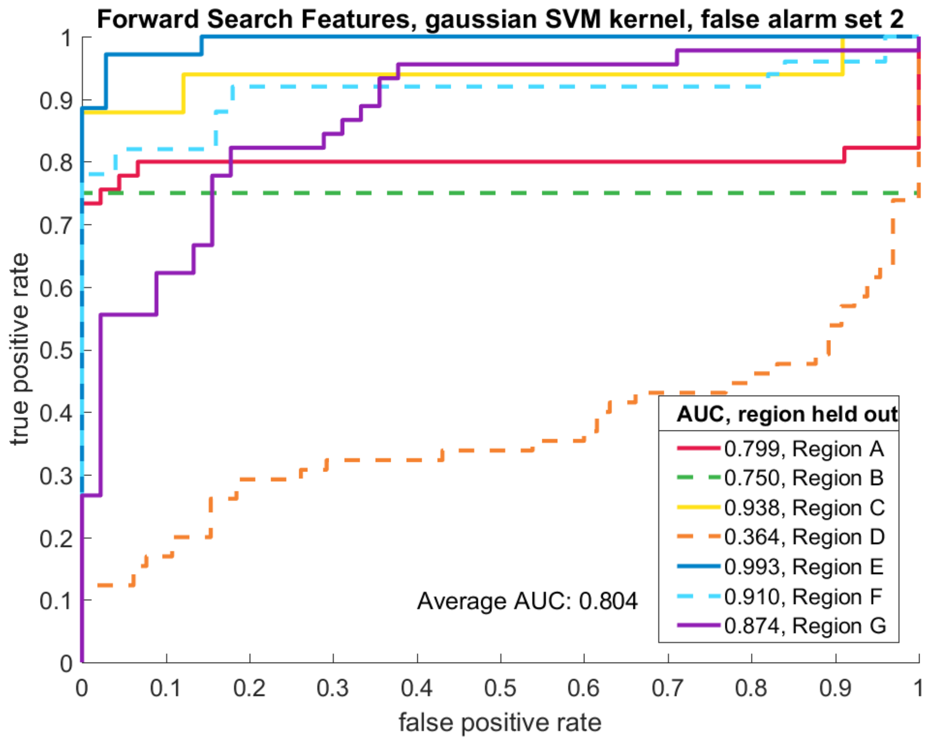

Figure 3-11 – ROC using Forward Search Features, Gaussian SVM kernel, and FA Set 2

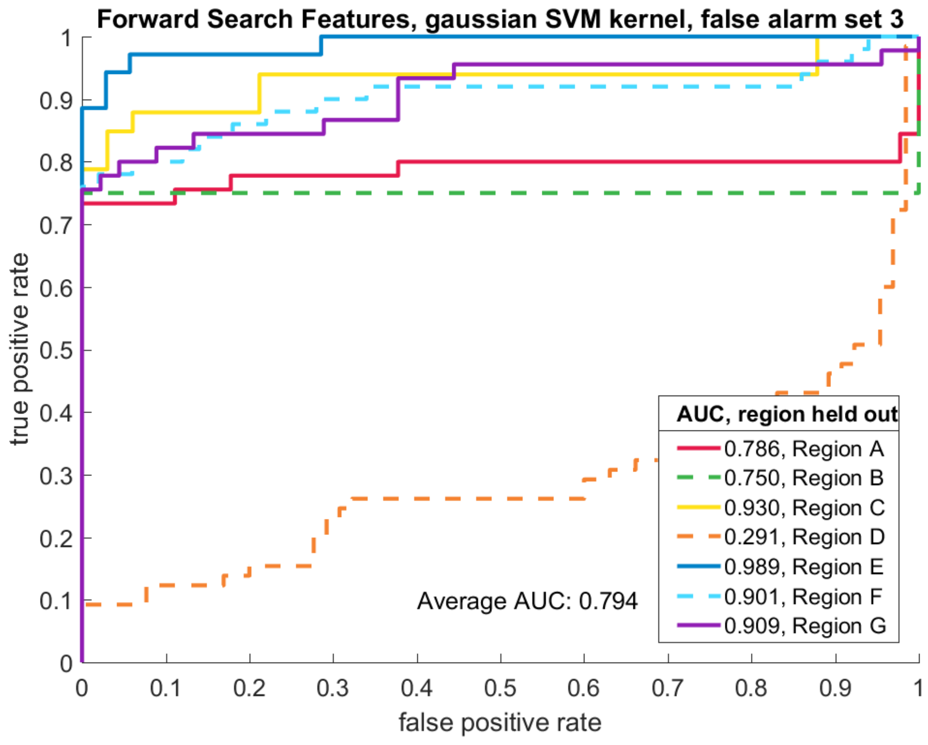

Figure 3-12 – ROC using Forward Search Features, Gaussian SVM kernel, and FA Set 3

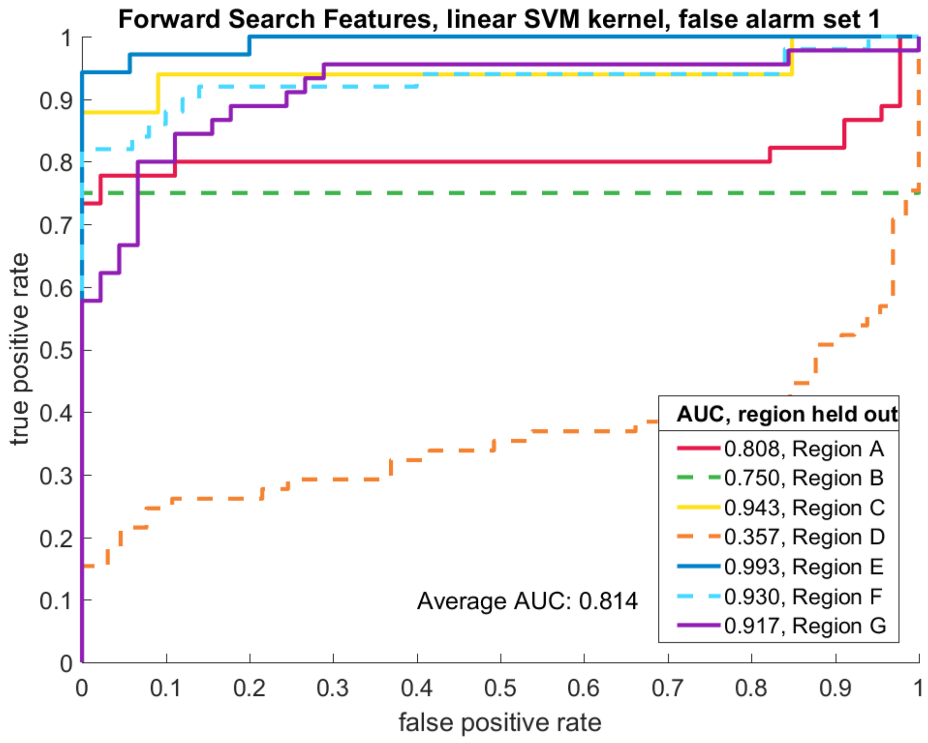

Figure 3-13 – ROC using Forward Search Features, linear SVM kernel, and FA Set 1

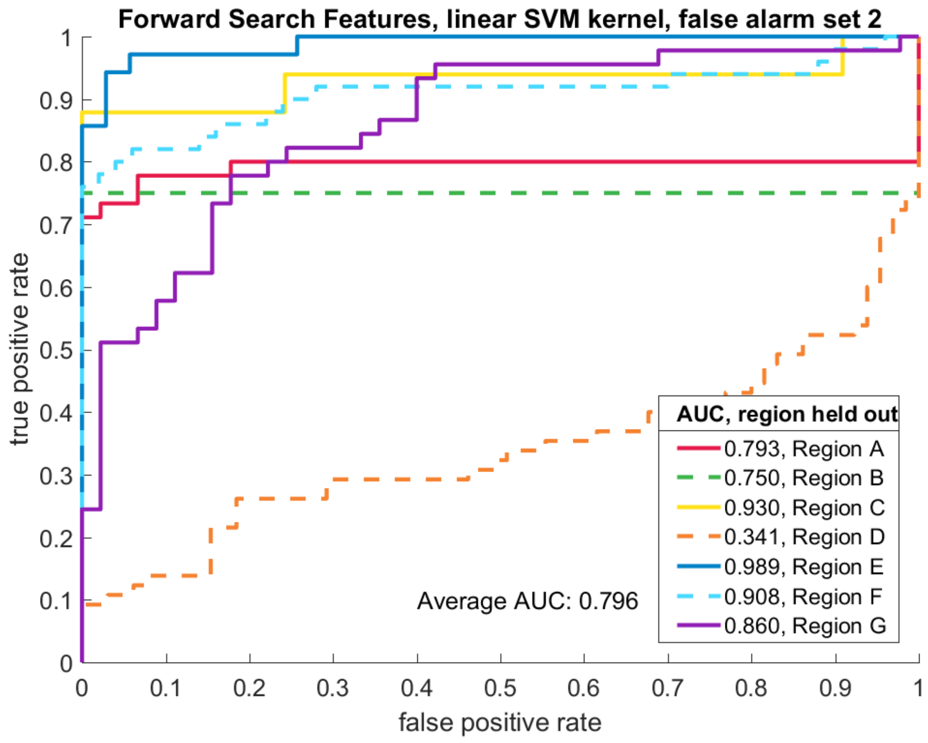

Figure 3-14 – ROC using Forward Search Features, linear SVM kernel, and FA Set 2

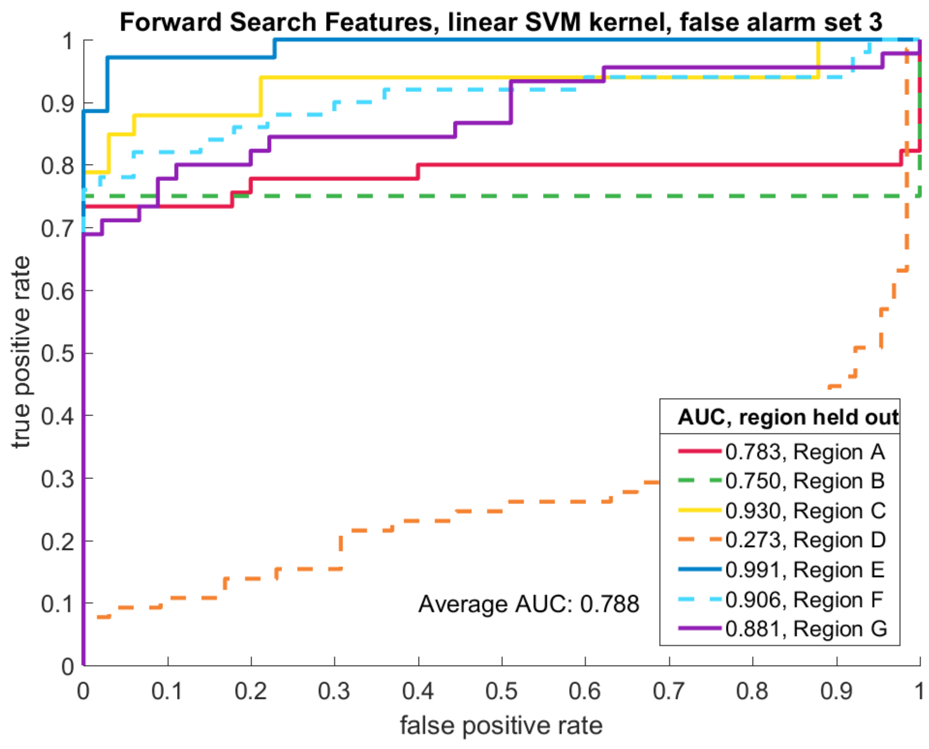

Figure 3-15 – ROC using Forward Search Features, linear SVM kernel, and FA Set 3

Noting that the Gaussian SVM kernel performs better than the linear kernel, we use the Gaussian kernel when evaluating the 12 slope and intercept (and no lacunarity) features only, shown in Figure 3-16, Figure 3-17, and Figure 3-18.

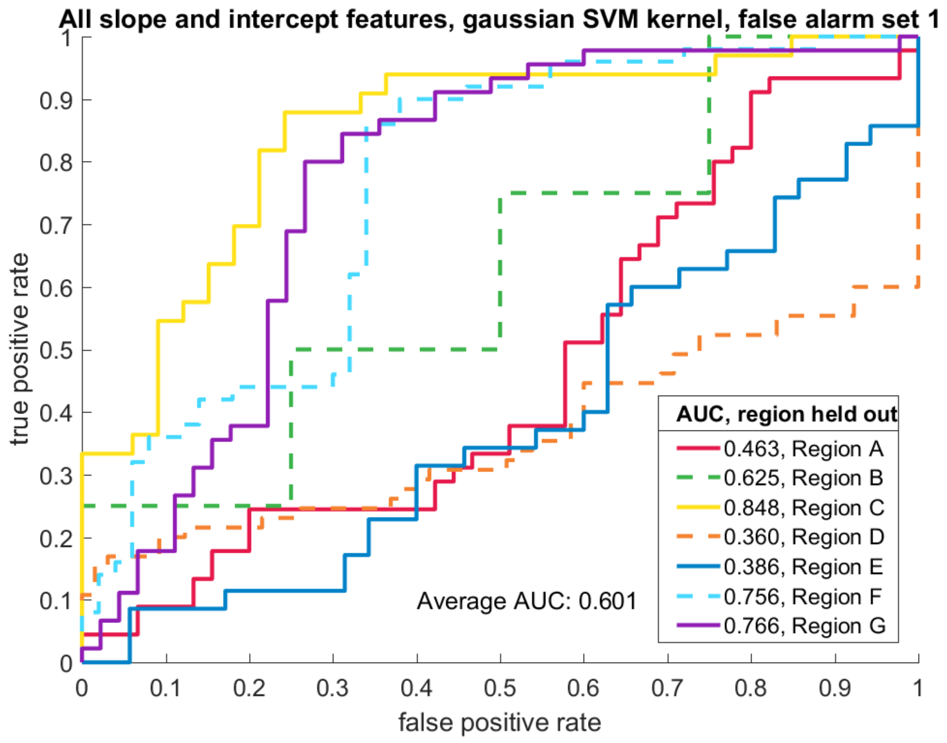

Figure 3-16 – ROC using slope and intercept features, Gaussian SVM kernel, and FA Set 1

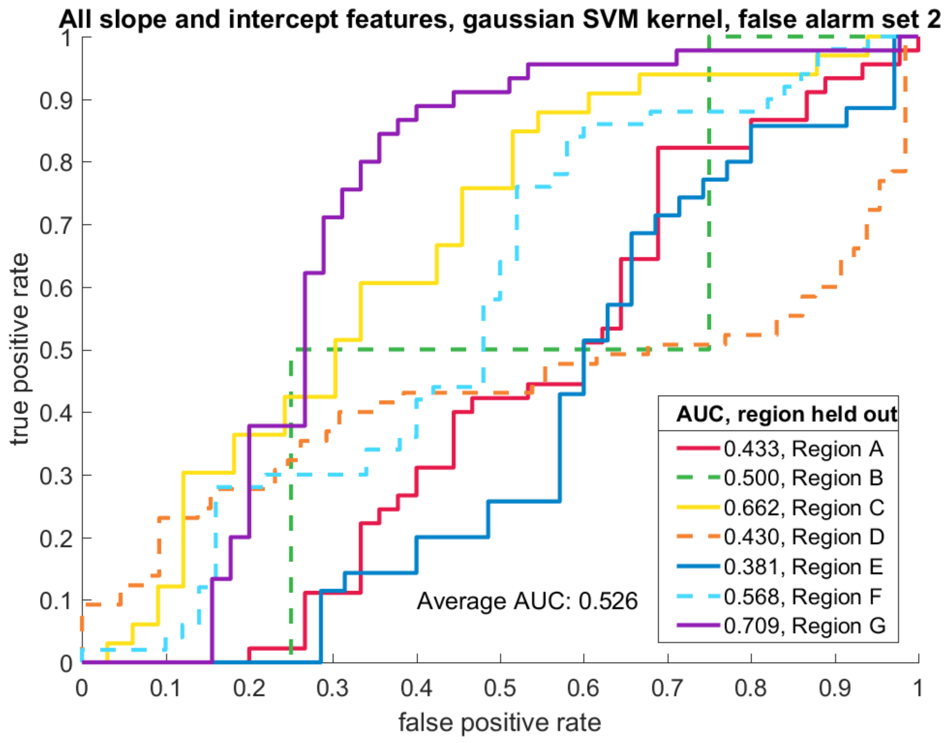

Figure 3-17 – ROC using slope and intercept features, Gaussian SVM kernel, and FA Set 2

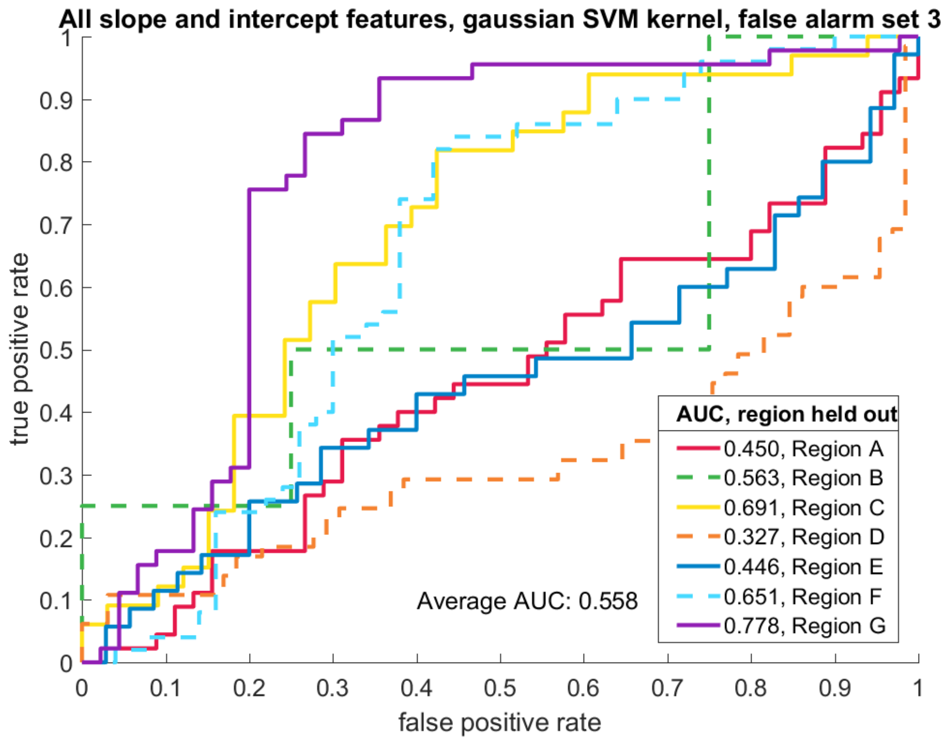

Figure 3-18 – ROC using slope and intercept features, Gaussian SVM kernel, and FA Set 3

Figure 3-16, Figure 3-17, and Figure 3-18 indicate that the primary slope and secondary intercept features are not as discriminatory as lacunarity in target vs. false alarm classification in a setting where not all textures are available for training.

Finally, we also do an experiment with the "fast" features – the four Method 3 features and the Williams lacunarity features – shown in Figure 3-19, Figure 3-20, and Figure 3-21.

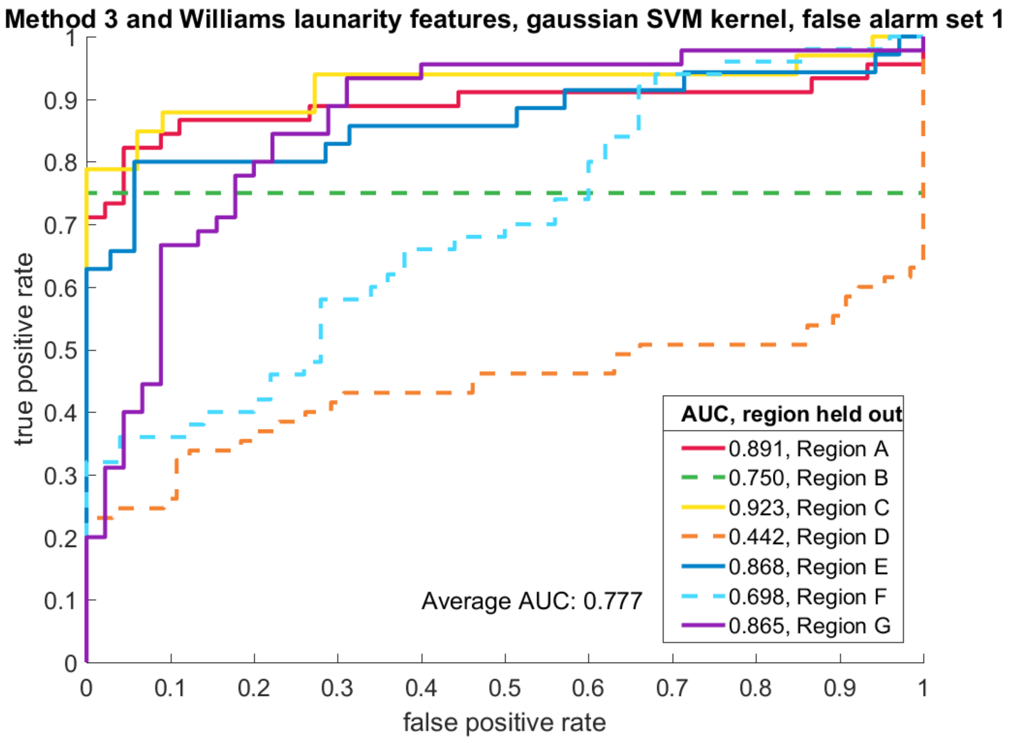

Figure 3-19 – ROC using Method 3 and Williams lacunarity features, Gaussian SVM kernel, and FA Set 1

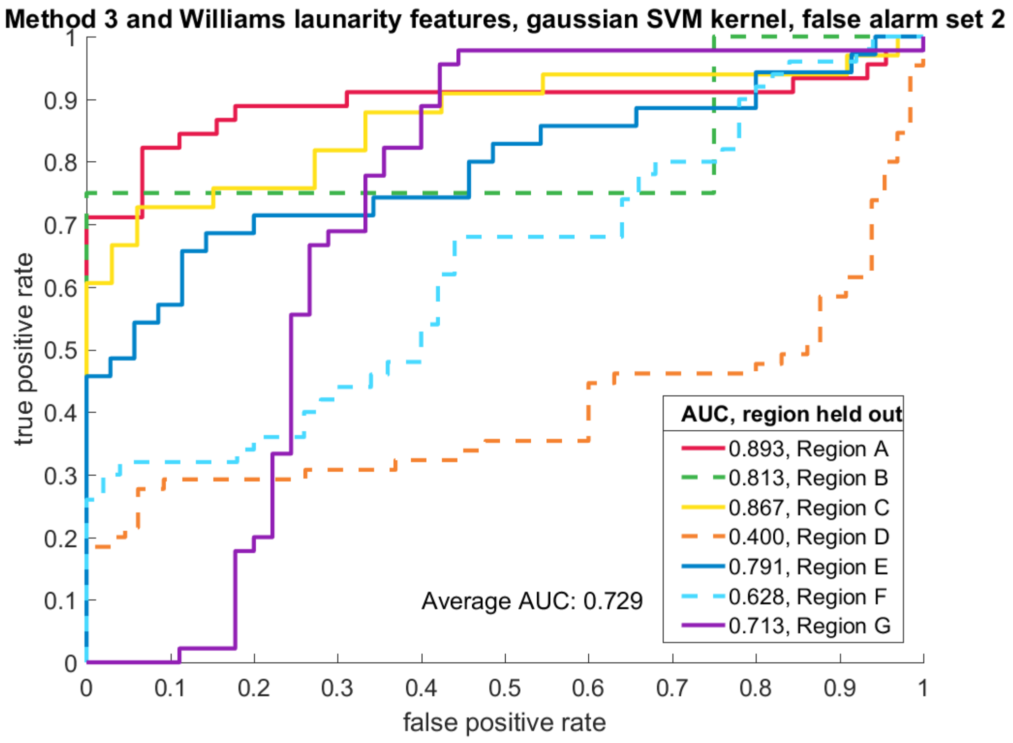

Figure 3-20 – ROC using Method 3 and Williams lacunarity features, Gaussian SVM kernel, and FA Set 2

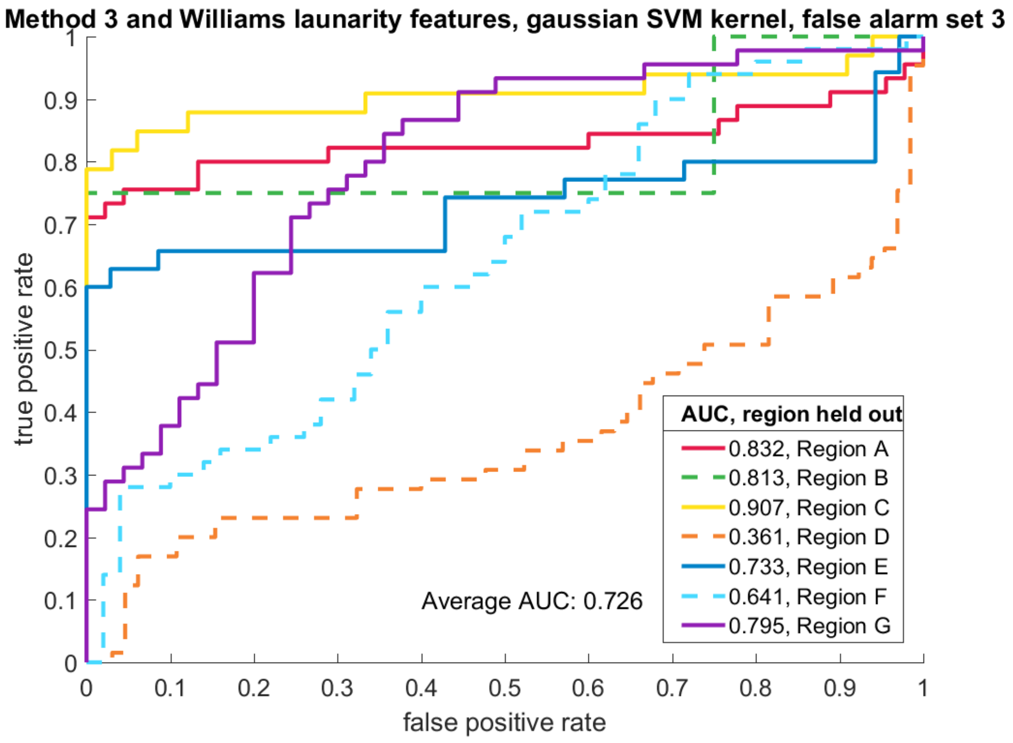

Figure 3-21 – ROC using Method 3 and Williams lacunarity features, Gaussian SVM kernel, and FA Set 3

We find here that, in some cases, the fast features are able to perform nearly as well as the features selected using our forward search experiment.

The results are generally encouraging, but one of the most striking observations drawn from these plots is the very poor results achieved when Region D is held out of the training and is then tested on. Visual inspection shows this region to include rocky textures including objects that have target-like appearances. It is also noted that Region D contains the highest number of extracted target and false alarm tiles, making up 65 of the 277 (23.5%) tile pairs in each set. This may indicate that 212 target tile pairs and 212 false alarm tile pairs are not sufficient for training.

We produce aggregate ROC curves with the confidence values of all seven cross-validation runs in each experiment aggregated and used to create one ROC curve. We show these curves in Figure 3-22, Figure 3-23, and Figure 3-24 with the four different experiments plotted together and each set of false alarm tiles used on a separate plot.

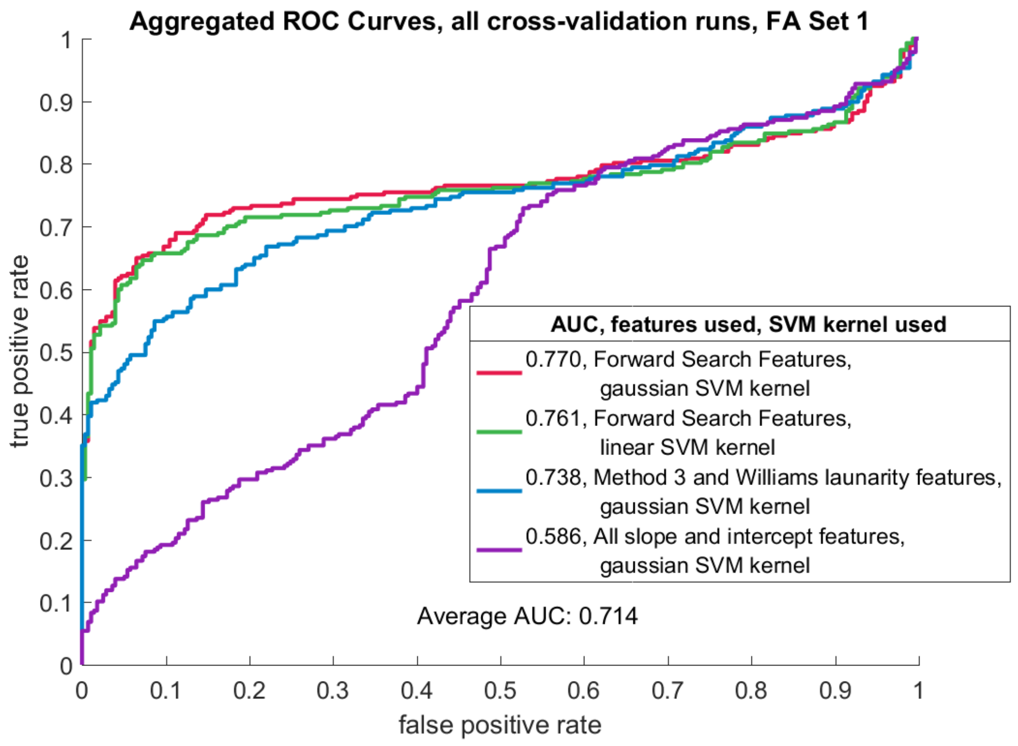

Figure 3-22 – Aggregate ROC curves using FA Set 1

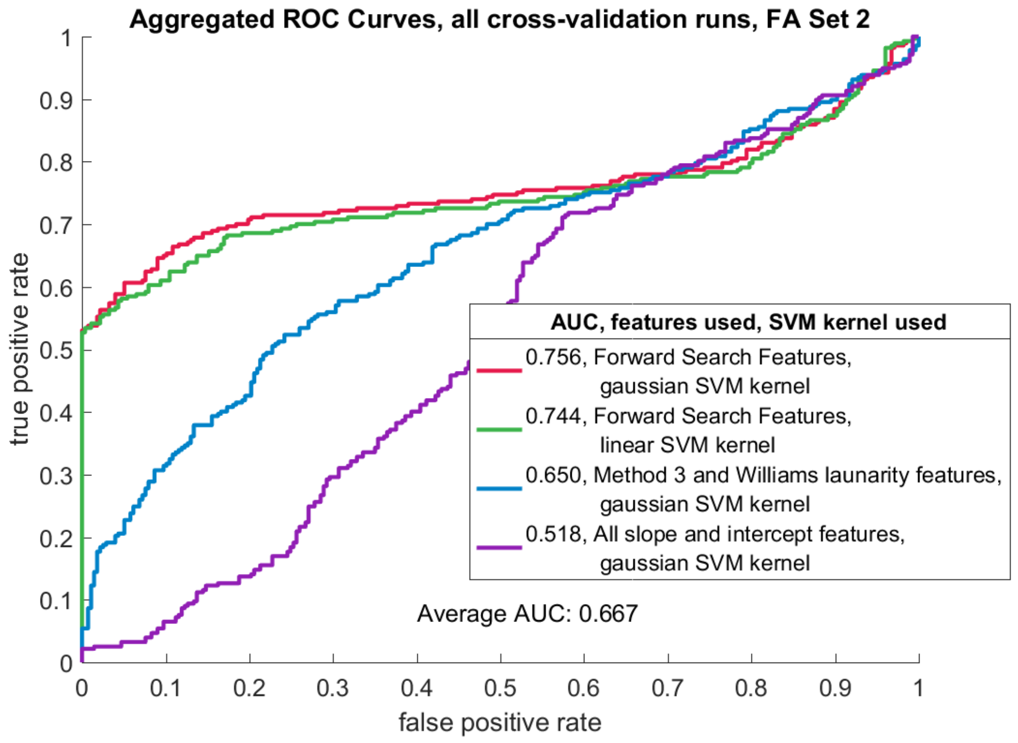

Figure 3-23 – Aggregate ROC curves using FA Set 2

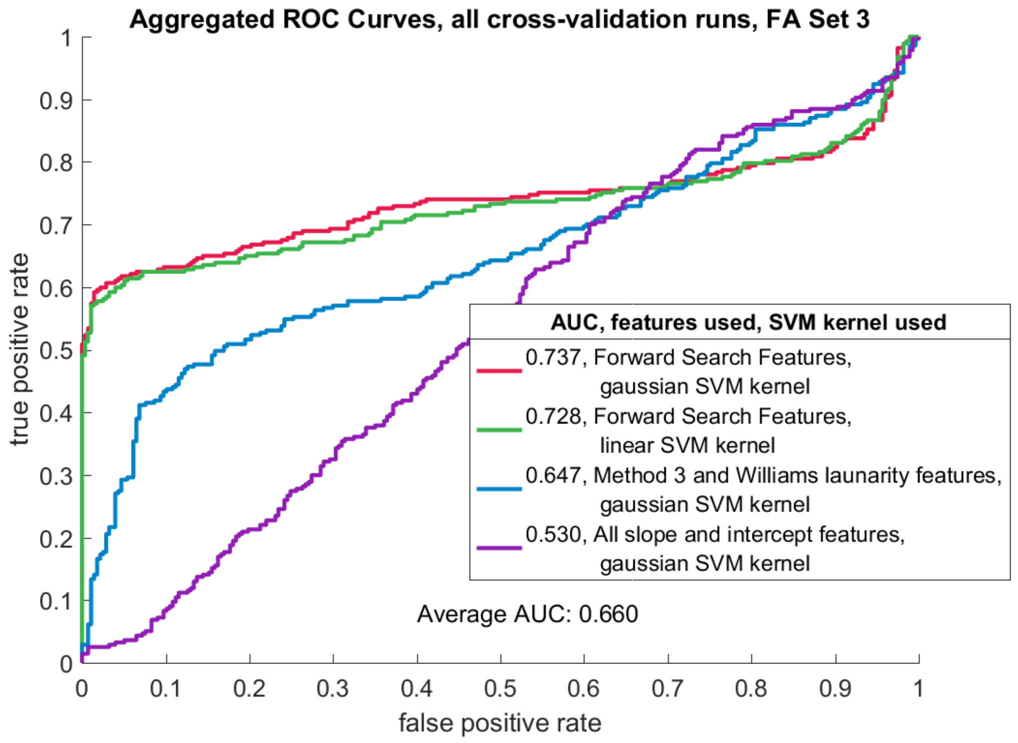

Figure 3-24 – Aggregate ROC curves using FA Set 3

When we remove from the aggregated data the confidence values produced by the cross-validation run in which the Region D tiles were held out of the training and used for testing, we are able to achieve an increase in the average AUC of ~0.1 for each FA tile set. These aggregated ROC curves are shown in Figure 3-25, Figure 3-26, and Figure 3-27.

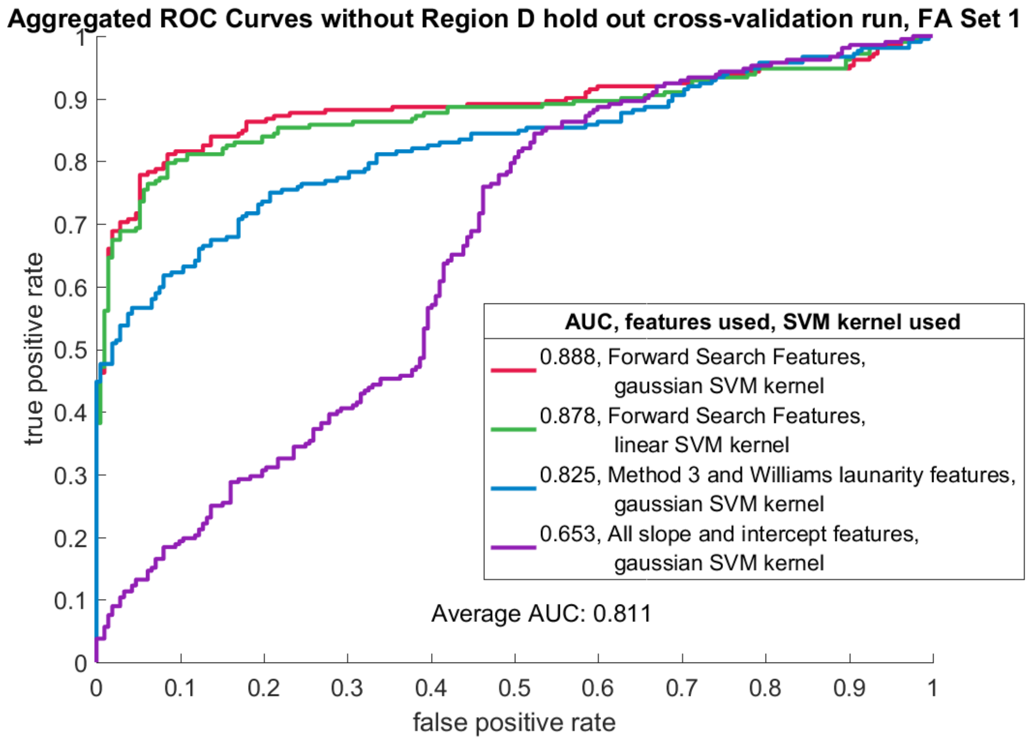

Figure 3-25 – Aggregate ROC curves with Region D test cross-validation run confidence values removed from aggregated data, using FA Set 1

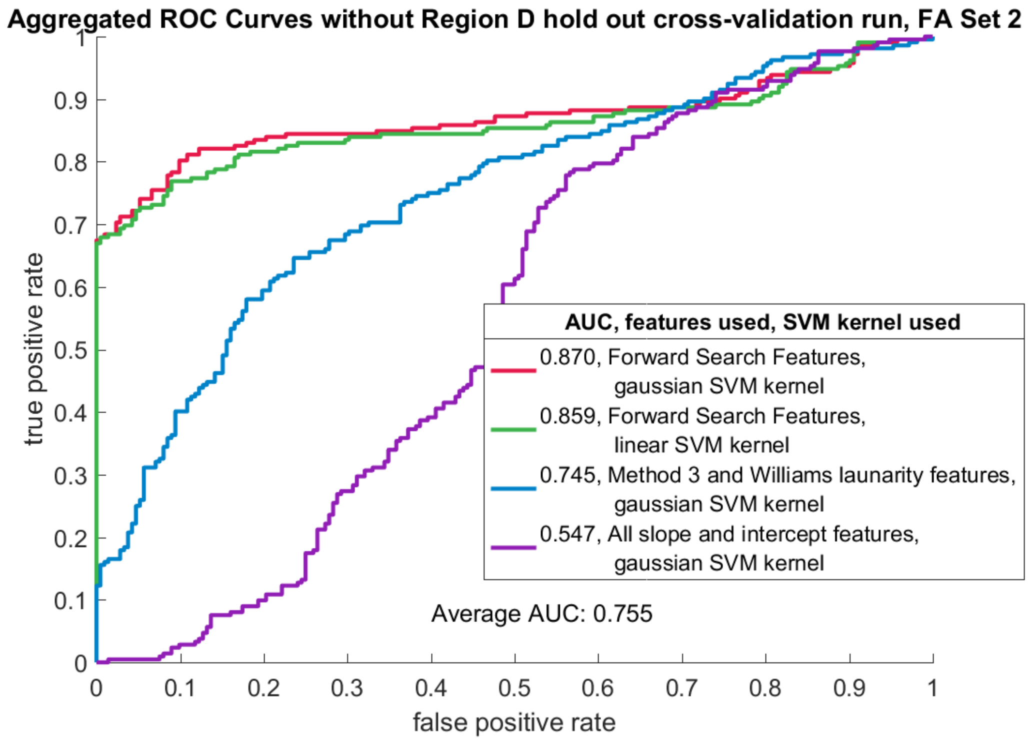

Figure 3-26 – Aggregate ROC curves with Region D test cross-validation run confidence values removed from aggregated data, using FA Set 2

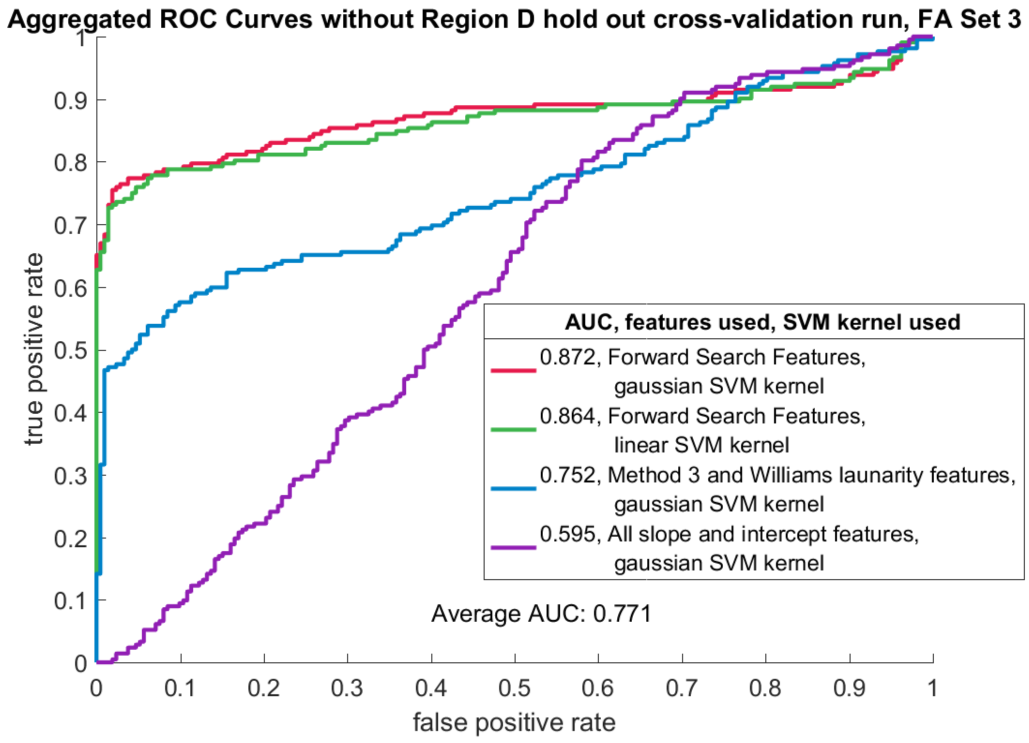

Figure 3-27 – Aggregate ROC curves with Region D test cross-validation run confidence values removed from aggregated data, using FA Set 3

It is made clear through this experiment that the Forward Search Features, all being lacunarity features, perform significantly better than the other feature ensembles tested.

3.5. Low-Resolution Feature Image Experiment

During our experimentation, we quickly realized that the time required to compute our features may be a significant hindrance to their usefulness, particularly in image segmentation applications or classification tasks with very large numbers of false alarms. In the case of a single image having dimensions 5600 px x ~5200 px, to compute all 82 features on 1.5 m x 1.5 m tiles surrounding each pixel in the image, we would need to process 27.4 million HF-LF tile pairs. Our single-threaded un-optimized code takes about 5.4 seconds to compute all 82 features per tile pair. This equates to 1720 days for a single image. Computing only the four Method 3 features requires about 0.09 seconds per tile pair, or about 30 days for the whole image. Even with highly optimized code running on GPUs, it is likely that completing the necessary computation for segmentation, or for classification where there are hundreds of thousands to millions of false alarms to be processed, would not be possible in a practical amount of time.

To solve this problem, we hypothesized that we could dramatically reduce the required compute time by dividing our image into a grid of tiles and processing each of those, rather than processing a tile around each individual pixel, and essentially creating a very low resolution feature image before than scaling it up to the original image size. We would then rely on the interpolation used during the scale-up process to determine the feature value at each pixel location.

Due to compute time required to compute the baseline image used for comparison, only limited testing was done. We tested with one image comprised entirely of the sand ripple texture. Only the fastest features – Method 3 slope and intercept features on HF and LF imagery – are used as a proof-of-concept. We divided the image into 1.5 m x 1.5 m tiles as shown in Figure 3-28 to obtain only 1476 pairs of tiles to be processed.

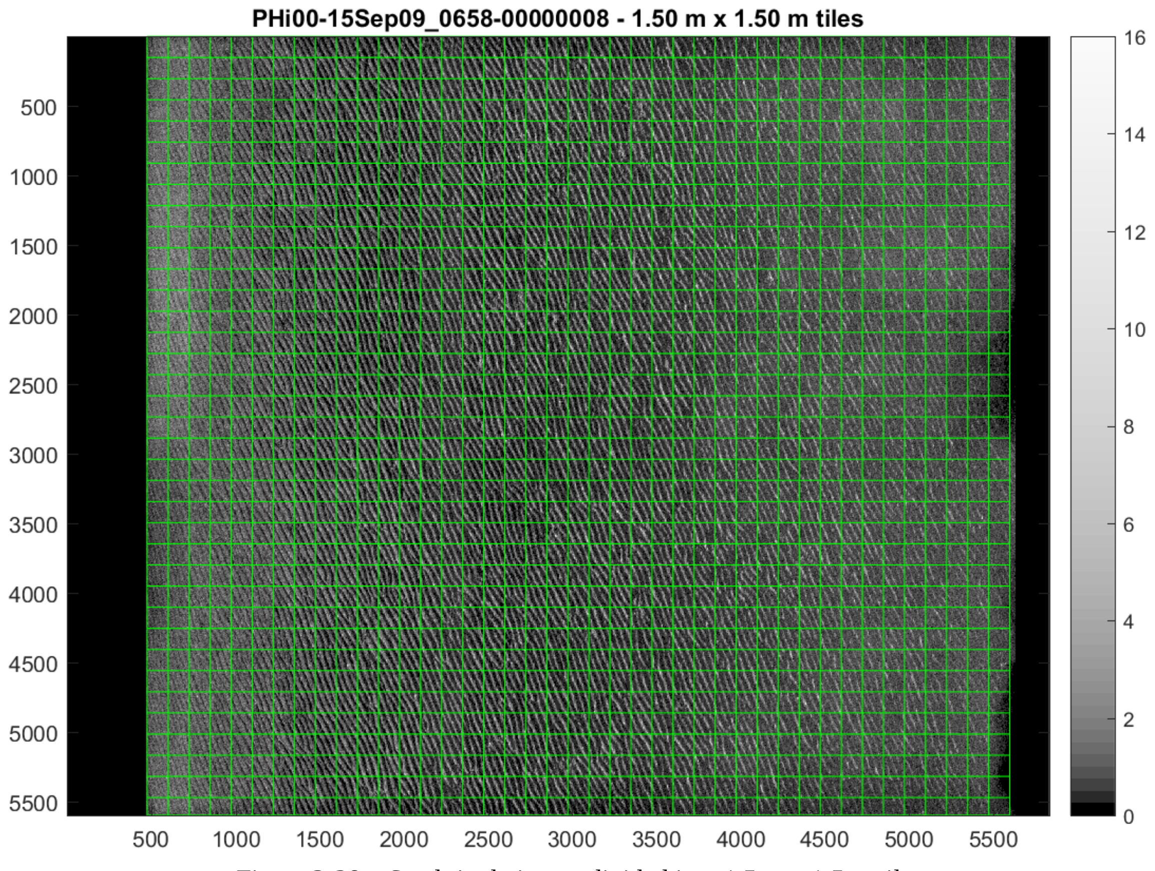

Figure 3-28 – Sand ripple image divided into 1.5 m x 1.5 m tiles

At 0.09 sec/tile pair for the four Method 3 features, we can compute our low-resolution feature image in 2.2 minutes. To produce a baseline image for comparison, we use a sliding 1.5 m x 1.5 m tile, stopping at every fourth pixel across and down, and computing the Method 3 features for each tile pair centered around the pixel at which we stopped. Both the grid-tile-feature image and sliding-tile-feature image are then scaled up to the original image size. We then find the error between these two images, as shown in the following figures. Note that the color axis has a different scale in each figure.

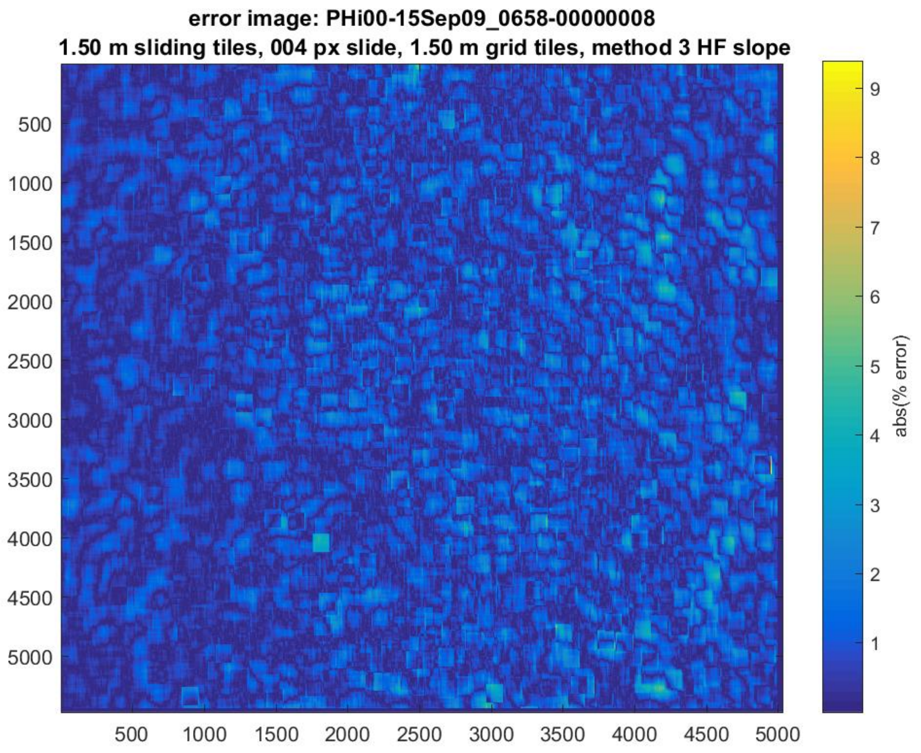

Figure 3-29 – Image of percent error between low-resolution feature image and sliding tile feature image for Method 3 HF slope feature, mean error is 0.77%, std of error is 0.65%

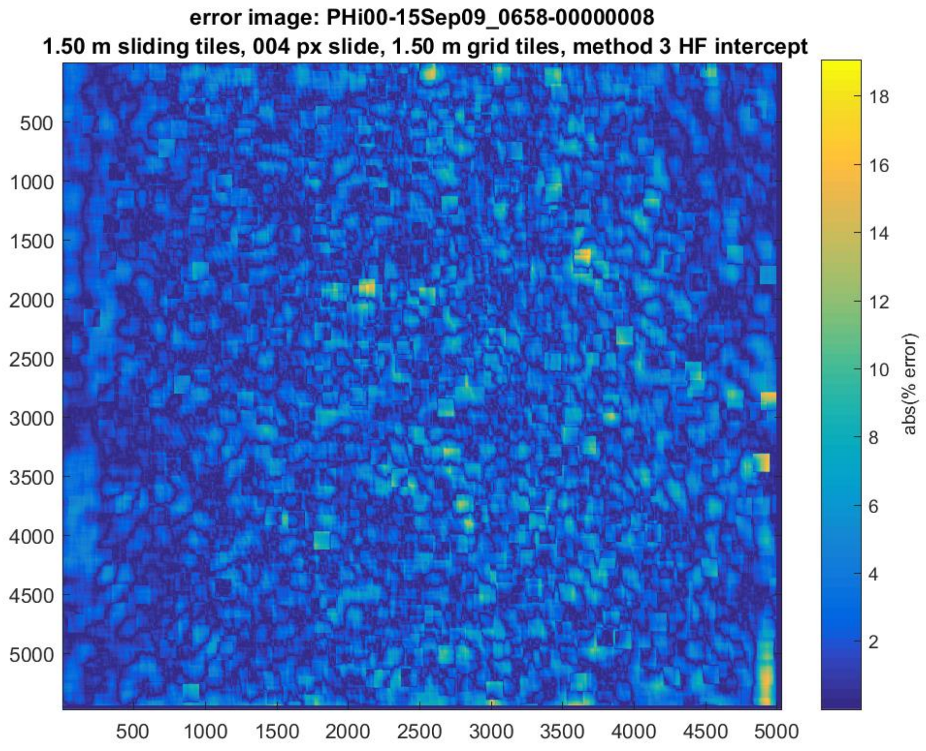

Figure 3-30 – Image of percent error between low-resolution feature image and sliding tile feature image for Method 3 HF intercept feature, mean error is 2.31%, std of error is 1.98%

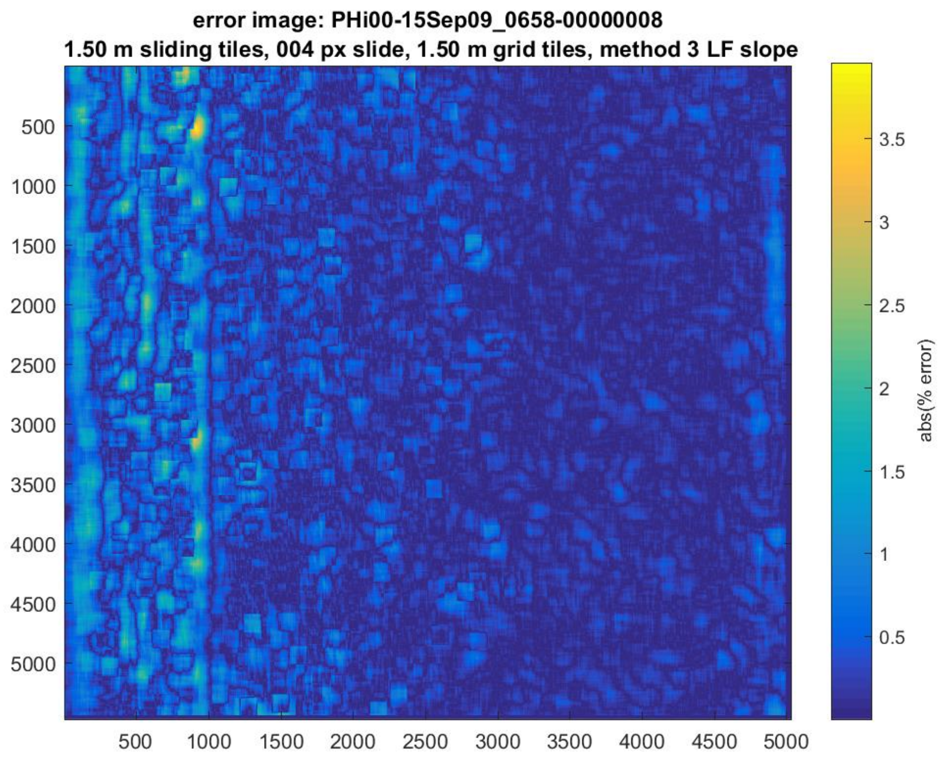

Figure 3-31 – Image of percent error between low-resolution feature image and sliding tile feature image for Method 3 LF slope feature, mean error is 0.30%, std of error is 0.34%

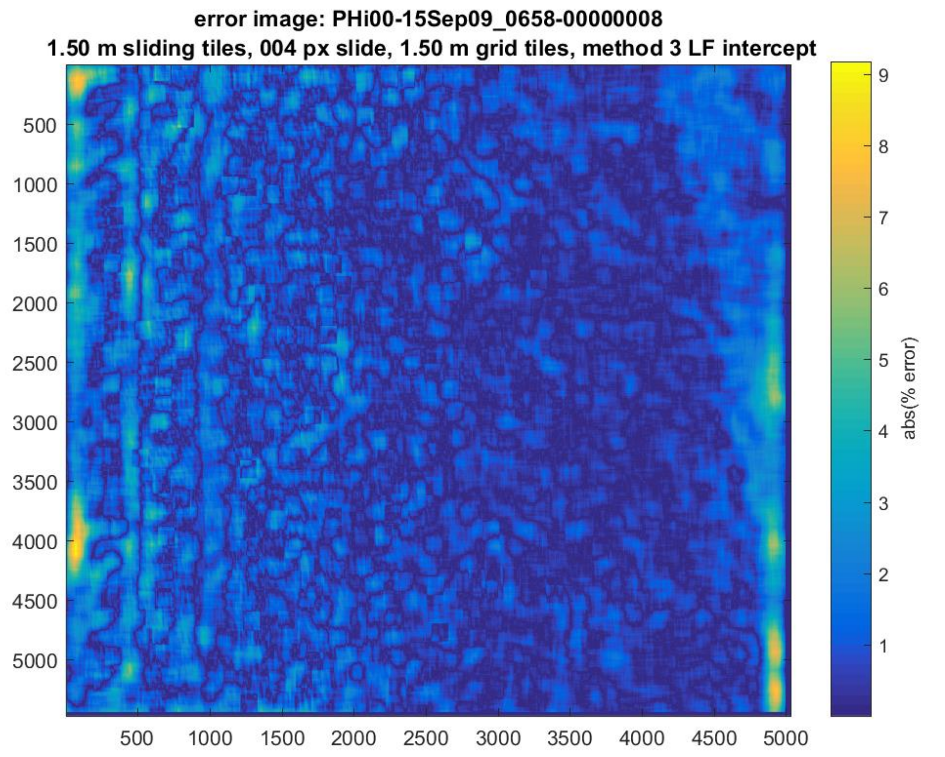

Figure 3-32 – Image of percent error between low-resolution feature image and sliding tile feature image for Method 3 LF intercept feature, mean error is 0.95%, std of error is 0.95%

It is observed that the regions with high error correspond with regions of shadow in the original image. If these regions were suppressed, it may be more obvious that much of the remaining error is near 0%. Similar results are achieved using 0.5 m x 0.5 m tiles, though these error images contain spikes much higher than 9% or 18%. Still, most of the error shown in those images is near 0%. The error images for the 0.5 m x 0.5 m tiles can be found in Appendix C. We summarize the data in Table 3-9.

Table 3-9 – Statistics from error images computed for interpolated Method 3 features

	Tile Size			
	0.5 m x 0.5 m		1.5 m x 1.5 m	
feature	mean	std	mean	std
Method 3, D (slope), HF	1.94%	1.93%	0.77%	0.65%
Method 3, B (intercept), HF	4.78%	4.72%	2.31%	1.98%
Method 3, D (slope), LF	0.76%	0.74%	0.30%	0.34%
Method 3, B (intercept), HF	1.96%	1.82%	0.95%	0.95%

While these results are encouraging, cross-validation experiments using features extracted by this method should be done to prove the worth of the method. To take a step in that direction, we do another 10-fold cross validation experiment using the MATLAB Classification Learner app and the same seven classifiers used previously. The folds in this experiment are not region-based. We create a set of the four Method 3 features extracted from our set of 277 target image tile pairs and FA Set 1. We then interpolate the same features using the method described in this section for the same hit coordinates around which the target tiles and tiles in FA Set 1 are centered. We compare the classification results across 30 trials between the image tile features and the interpolated features. Our classification accuracy averages and standard deviations are given in Table 3-10.

Table 3-10 – Comparison of average classification accuracies over 30 trials, using Method 3 features directly extracted from tiles vs. interpolated Method 3 featuers

		Tile Features		Interpolated Features		
		Average Classification Accuracy	Standard Deviation of Classification Accuracy	Average Classification Accuracy	Standard Deviation of Classification Accuracy	Average Classification Accuracy Difference
Classifier	Linear SVM	95.3%	0.2%	88.3%	0.3%	7.0%
	Quadratic SVM	95.3%	0.3%	89.0%	0.5%	6.3%
	Cubic SVM	94.0%	0.5%	87.6%	0.6%	6.4%
	Fine Gaussian SVM	94.8%	0.3%	86.8%	0.6%	8.0%
	Medium Gaussian SVM	95.3%	0.3%	87.8%	0.4%	7.5%
	Coarse Gaussian SVM	95.4%	0.2%	89.4%	0.3%	6.0%
	Cubic KNN	95.3%	0.4%	89.0%	0.4%	6.3%
	Average	*95.1%*	*0.3%*	*88.3%*	*0.4%*	*6.8%*

Again, these results show promise, but it is clear that in some cases there may be a significant loss in classification accuracy when using interpolated features. Further development is needed and more testing should be done with other tile sizes and features.

3.6. Distance from Sensor Experiment

As discussed in Section 1, we hypothesized that because the scale of the texture changes with distance from the SAS image sensor's point of view, the constant of proportionality, i.e. the C and B secondary intercept features from the three methods of fractal dimension computation, should increase with distance while the D primary slope feature should remain constant. To test this hypothesis, we used the same image tiles shown in Figure 3-28, compute the 12 slope and intercept features on the tiles, bin the features by the distance of their corresponding tile from the sensor, and find the mean and standard deviation of each bin. We then plot those values vs. the distance from the sensor, given in Figure 3-33 and Figure 3-34.

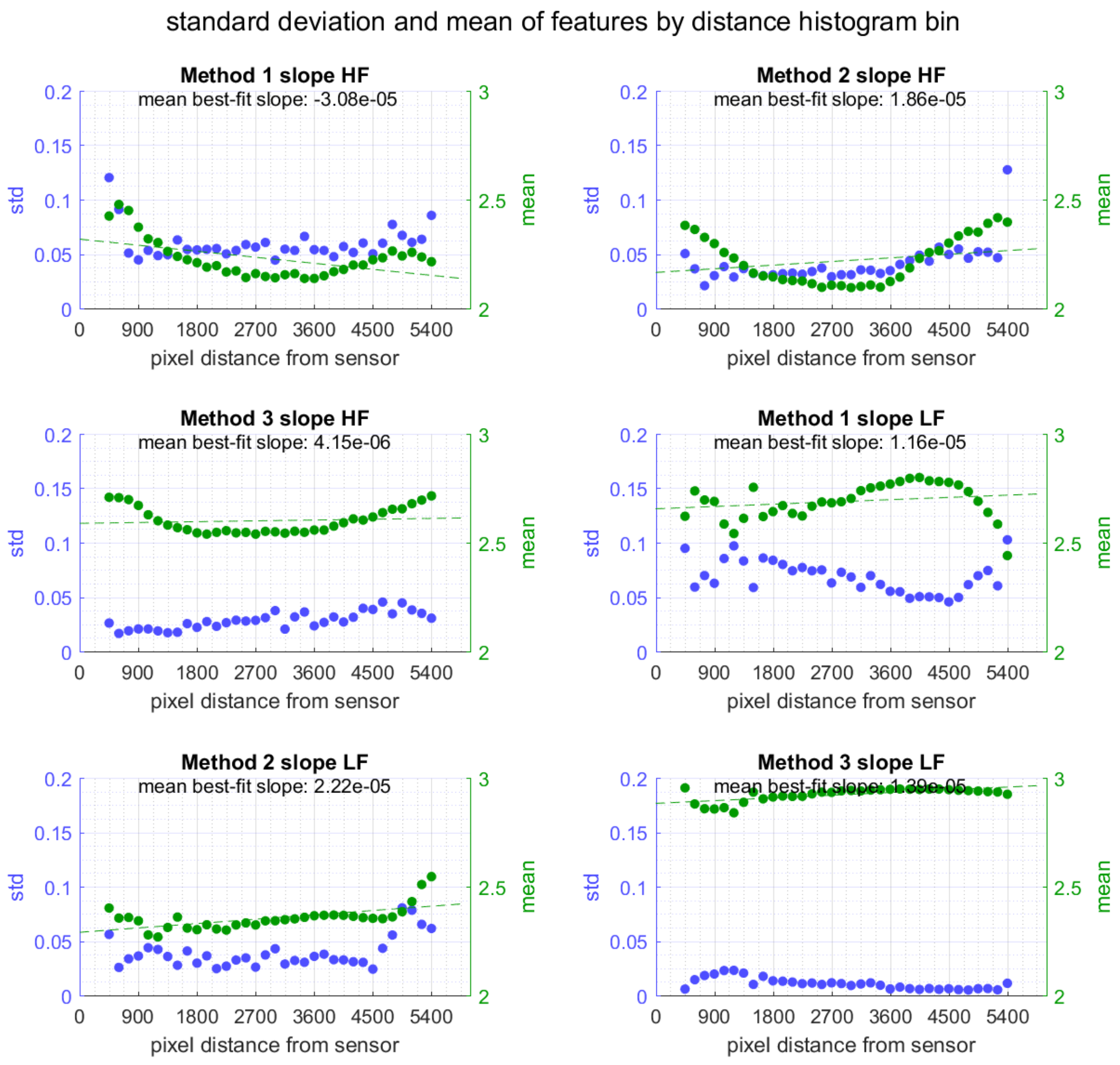

Figure 3-33 – Mean and standard deviation vs. distance from the sensor for the six slope features

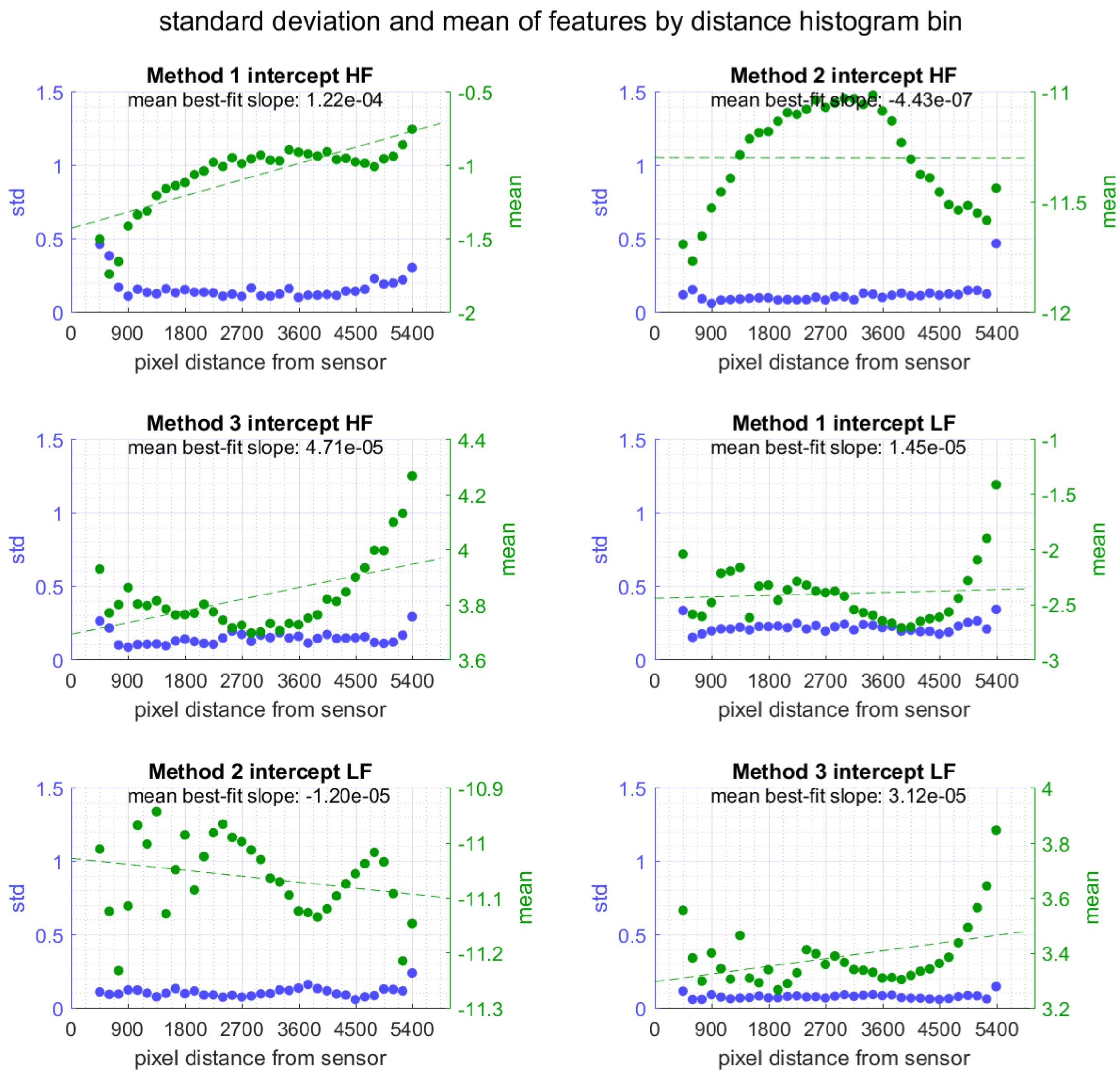

Figure 3-34 – Mean and standard deviation vs. distance from the sensor for the six intercept features

In Table 3-11, we summarize the data by tabulating the slopes of the best fit line for the mean values of the slope features, and the slopes of the best fit line for the mean value of the intercept features.

Table 3-11 – Slopes of best fit lines for mean values of slope and intercept features

	slope of best fit line for slope means	slope of best fit line for intercept means
Method 1 HF	-3.08e-05	1.22e-04
Method 2 HF	1.86e-05	-4.43e-07
Method 3 HF	4.15e-06	4.71e-05
Method 1 LF	1.16e-05	1.45e-05
Method 2 LF	2.22e-05	-1.20e-05
Method 3 LF	1.39e-05	3.12e-05

It appears that our hypothesis holds true for the Method 3 HF slope and intercept features, where the slope of the best fit line for the intercept feature mean values is an order of magnitude greater than that for the slope feature mean values. However, some of the other features do not lend support to the hypothesis, or even show declining intercept values from left to right across the image while the slope values increase. It is also recognized that the slope of the best fit line for the mean values many not be the best method of quantitative measurement, particularly for the intercept features, many of which have very nonlinear trends and in fact have mean values increasing towards the right hand side of the image – further away from the SAS image sensor.

4. CONCLUSION AND FUTURE WORK

In this thesis, we investigated the use of fractal dimension and related features in SAS target detection. It is shown that almost all of the features – the fractal dimension D, the y-intercepts C and B, and the lacunarity features – extracted from our SAS image tiles have some potential to perform well in classification tasks, and by extension, possibly in image segmentation applications also. However, the potential of the lacunarity features clearly dominates the potential of the primary fractal dimension slope features and the secondary intercept features. The fact that the Method 3 features appear to be essentially just as useful in initial classification testing as all of the slope and intercept features combined is very advantageous due to of the much lower compute time required for the Method 3 features alone.

Our results could be improved upon by continuing to analyze and experiment with the many parameters of the methods used and the characteristics of the experimental setup. By tweaking the values used to set the box size in Method 1 and Method 2, the shift distance and direction in Method 3, the window size used for the Williams lacunarity features, the size of the tiles centered around the targets and false alarms, and more, any number of feature sets could be produced for each image tile. Some sort of parameter selection study would help to address the concerns discussed in Section 2.4.

Further experimentation with different classifiers and their respective threshold levels would also be prudent. It is possible that the forward search algorithm could be improved by guiding the search with computed correlation values between features.

The region-by-region cross-validation should be run on a full set of data, rather than small balanced data sets. While we used balanced data sets of 277 target tile pairs and 277 false alarm target pairs, our data set includes ~1.35e6 false alarms. RUSBoost [19] should be investigated as a classifier able to deal with such unbalanced data.

To further show the usefulness of the features investigated here, optimized code should be developed that can be applied to GPU devices. Because of the “single instruction, multiple data” (SIMD) nature of our computation, these features are great candidates for GPU implementation.

Lastly, our features should be compared in forward search and cross-validation experiments to features that are already widely used in classification. Without such a comparison, it is difficult to say what contribution the fractal dimension and related features would make in a mixed feature ensemble.

5. REFRENCES

[1] 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.

[2] 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.

[3] 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.

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

[5] I. Sobel, G. Feldman, “A 3x3 Isotropic Gradient Operator for Image Processing”, presented at a talk at the Stanford Artificial Project, 1968); unpublished but often cited, orig. in: R. Duda, P. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, pp. 271–272, 1973.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886-89, San Diego, 2005.

[7] J. M. Keller and S. Chen, "Texture description and segmentation through fractal geometry," *Computer Vision, Graphics, and Image Processing*, vol. 45, pp.150-166, 1989.

[8] B. B. Mandelbrot, *Fractals: Form, Chance and Dimension*, W. H. Freeman & Company, San Francisco, 1977.

[9] R. M. Crownover, *Introduction to Fractals and Chaos*, Jones and Bartlett Publishers, Boston, pp. 244-254, 1995.

[10] J. M. Keller and Y. Seo, "Local fractal geometric features for image segmentation," *International Journal of Imaging Systems and Technology*, vol. 2, pp. 267-284, 1990.

[11] B. B. Mandelbrot and J. V. Ness, "Fractional Brownian motion, fractional noises and applications", *SIAM Rev.* 10, No. 4, 1968.

[12] R. Voss, "Random fractals: characterization and measurement," *Scaling Phenomena in Disordered Systems*, edited by R. Pynn, and A. Skjeltorp, Plenum, New York, 1986.

[13] B. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, San Francisco, 1983.

[14] D. P. Williams, "Fast unsupervised seafloor characterization in sonar imagery using lacunarity," *IEEE Transactions on Geoscience and Remote Sensing*, 53(11), 2015.

[15] A. N. Kolmogorov, *Wienersche Spiralen und Einige Interessante Kurven im Hilvertschen Raum*, *C.R. (Dokalady Acad. URSS (N.S.)*, Vol. 26, pp. 115-118, 1940.

[16] M. M. Kanafi, “Surface generator: artificial randomly rough surfaces,” *File Exchange*, MathWorks, [online] https://www.mathworks.com/matlabcentral/fileexchange/60817-surface-generator--artificial-randomly-rough-surfaces, 2016.

[17] A. Galusha, G. Galusha, J. Keller, and A. Zare, “A target detection algorithm for underwater Synthetic Aperture Sonar imagery”, *Proc. SPIE Conference on Detection And Remediation Technologies For Mines And Minelike Targets XXIII*, DS104-27, 2018.

[18] “fitcsvm,” *Documentation*, MathWorks, [online] https://www.mathworks.com/help/stats/fitcsvm.html, 2018.

[19] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse and A. Napolitano, "RUSBoost: A Hybrid Approach to Alleviating Class Imbalance," in *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 1, pp. 185-197, Jan. 2010.

APPENDIX A

As discussed in Section 3.2, this appendix contains additional histograms showing the features listed in Table 3-2 and computed on the set of target tiles and the three sets of false alarm tiles.

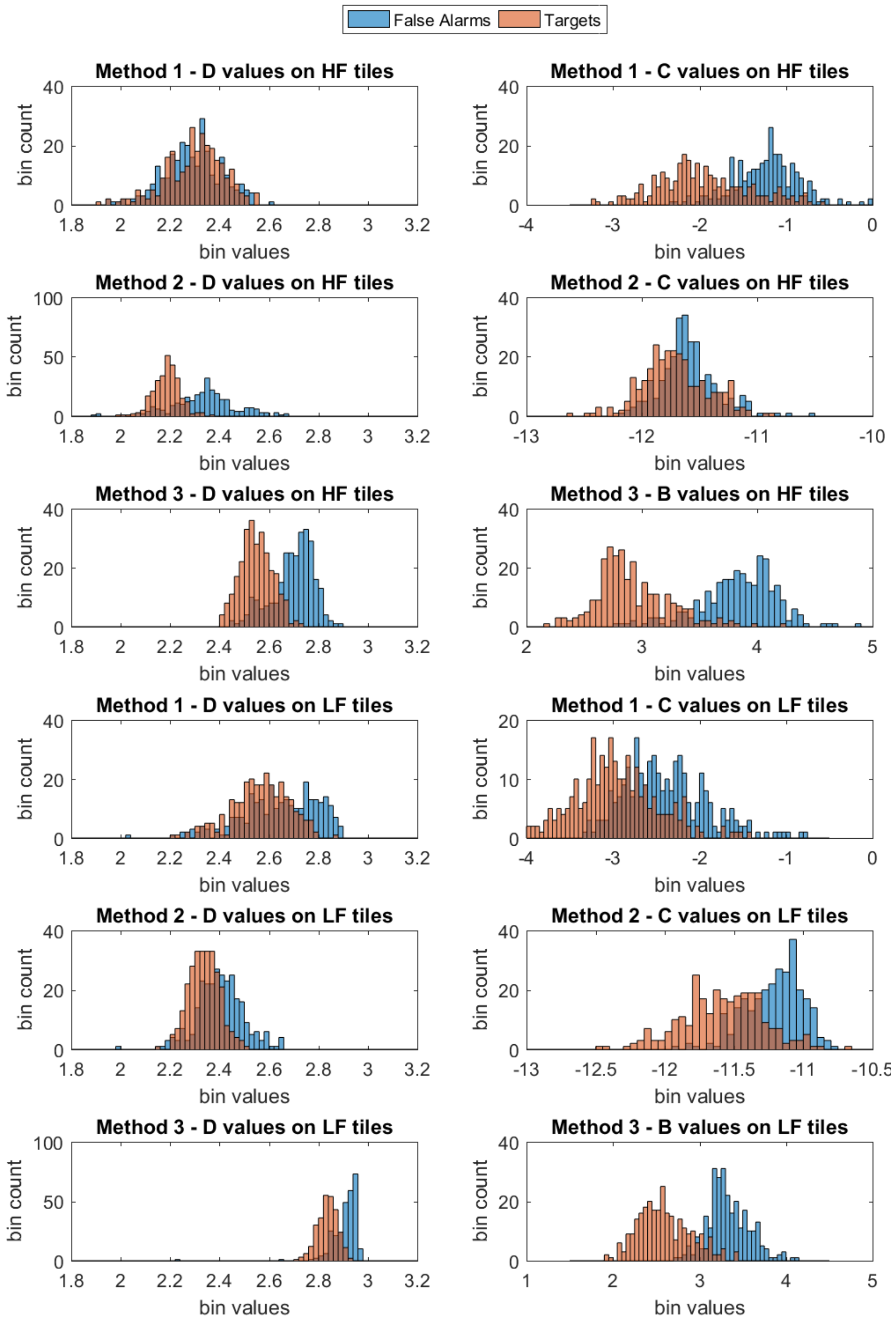

Figure A-1 – Histograms of twelve slope and intercept features from target tiles and tiles in FA Set 2

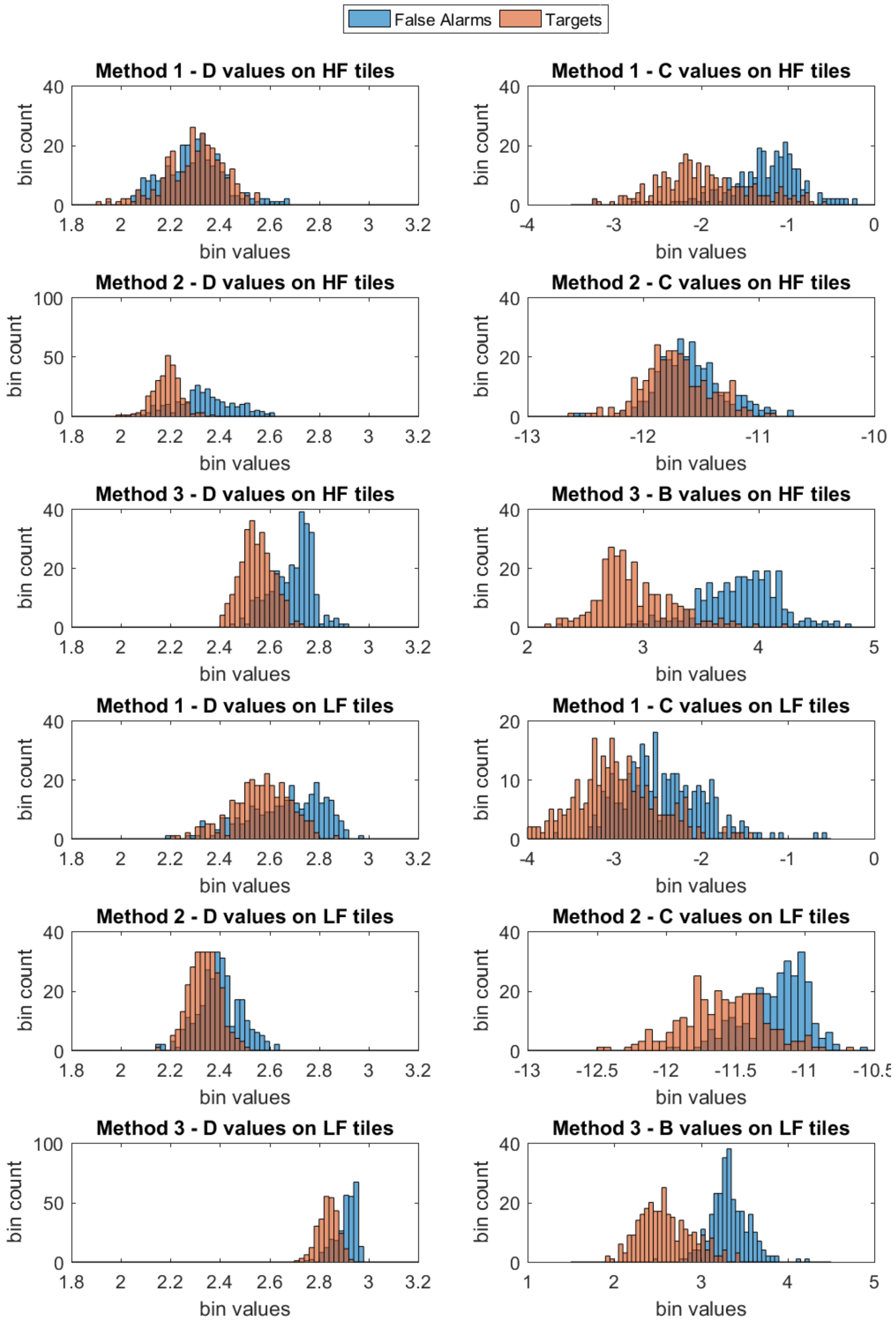

Figure A-2 – Histograms of twelve slope and intercept features from target tiles and tiles in FA Set 3

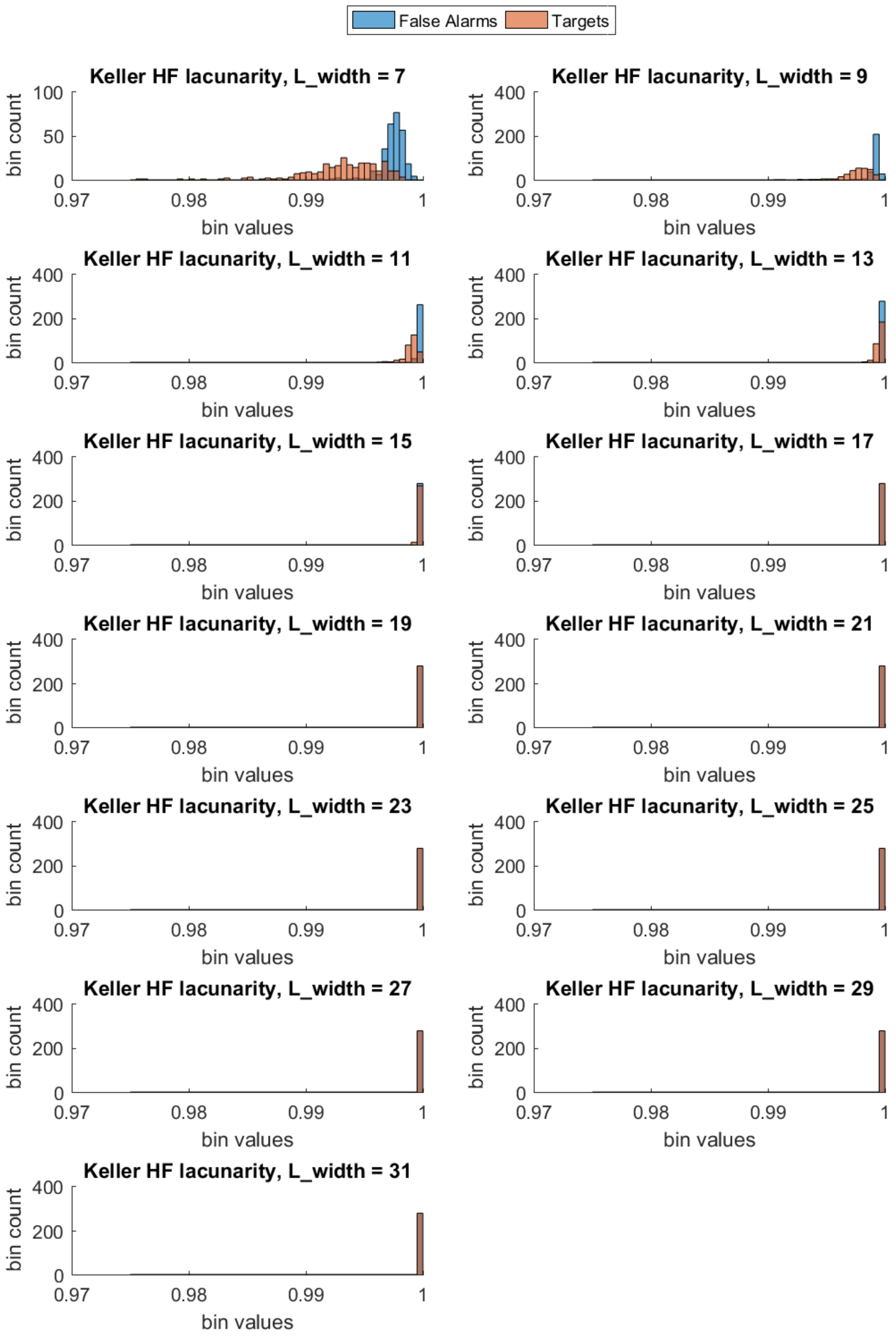

Figure A-3 – Histograms of computed Keller lacunarity features on HF imagery target tiles and tiles in FA Set 1

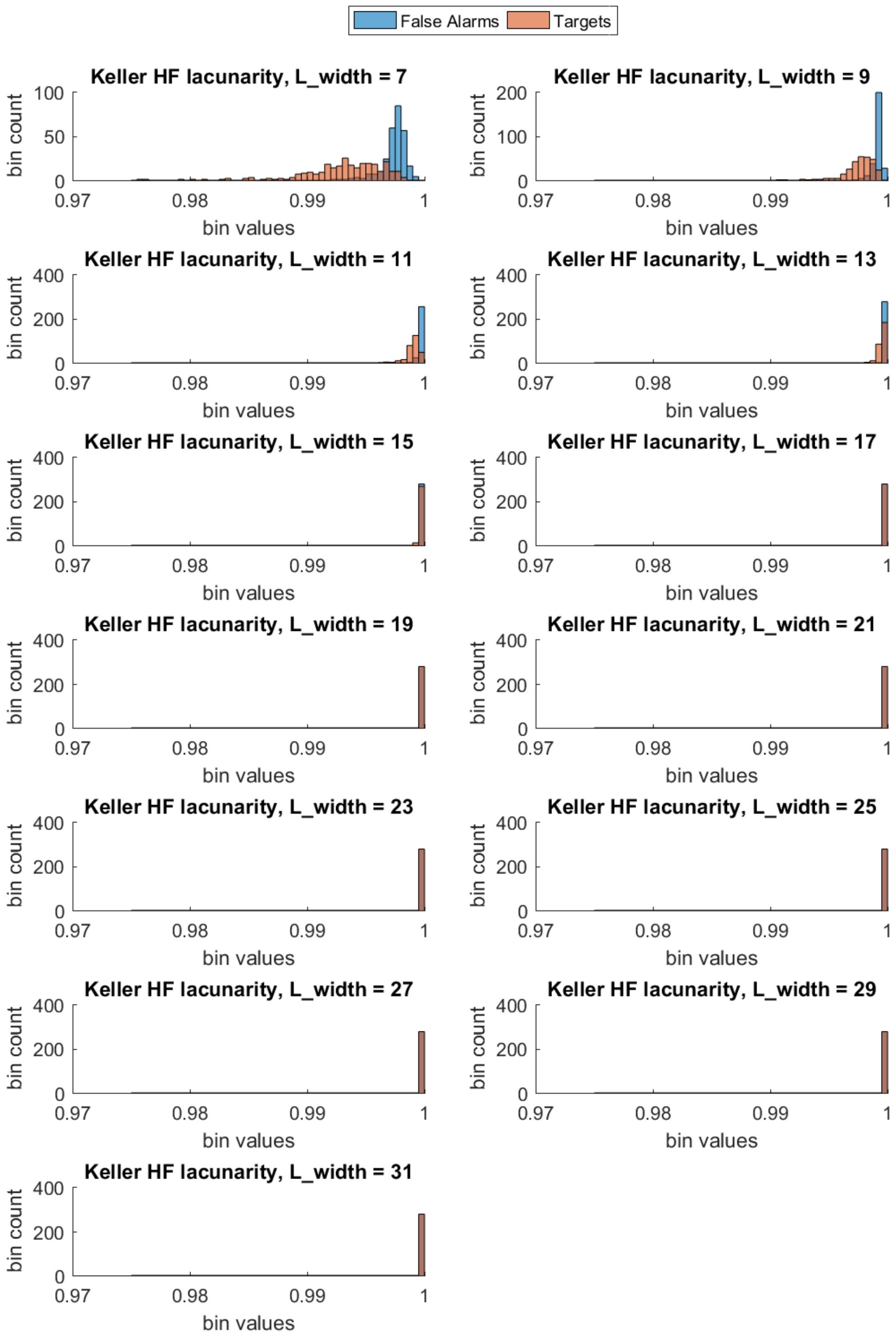

Figure A-4 – Histograms of computed Keller lacunarity features on HF imagery target tiles and tiles in FA Set 2

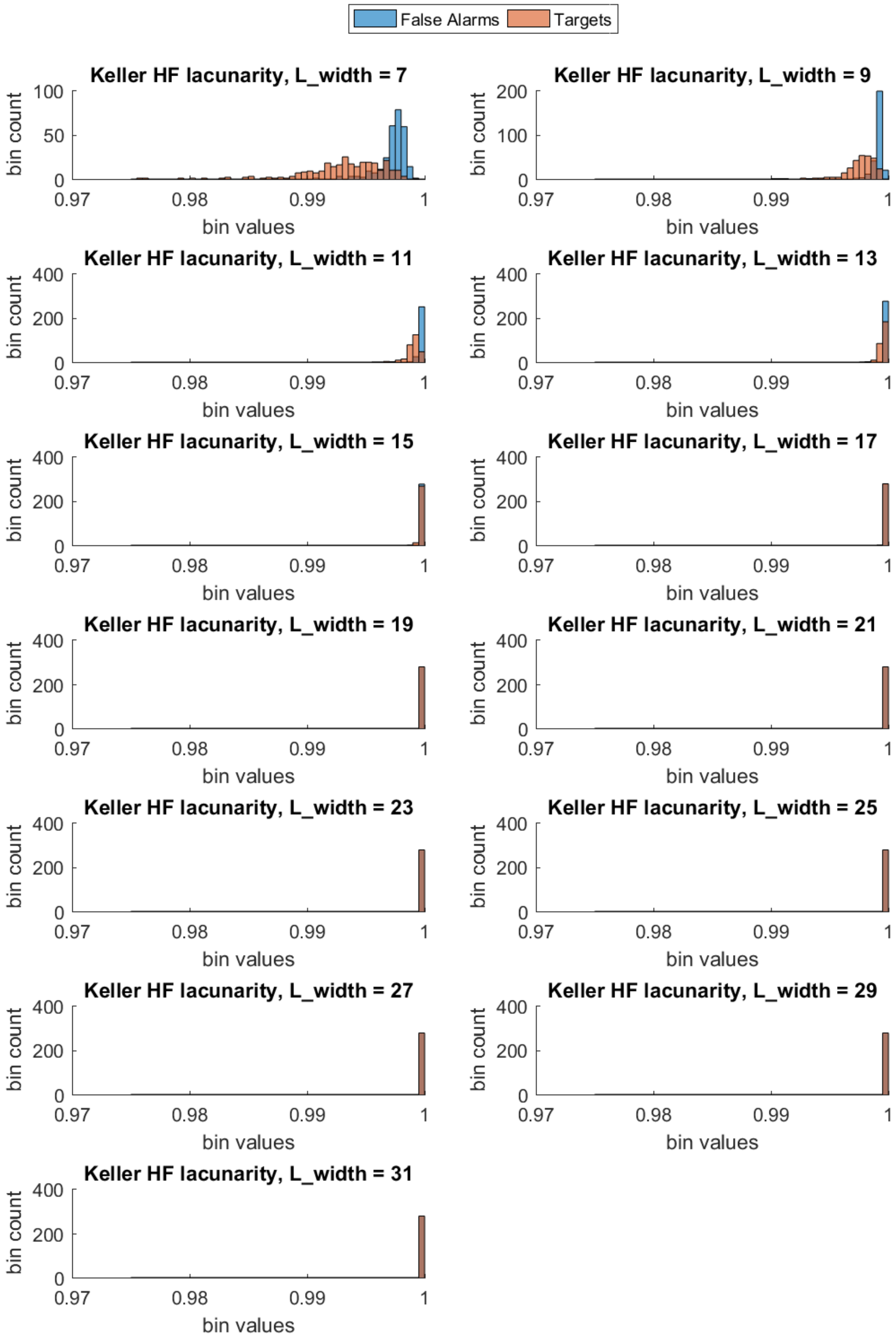

Figure A-5 – Histograms of computed Keller lacunarity features on HF imagery target tiles and tiles in FA Set 3

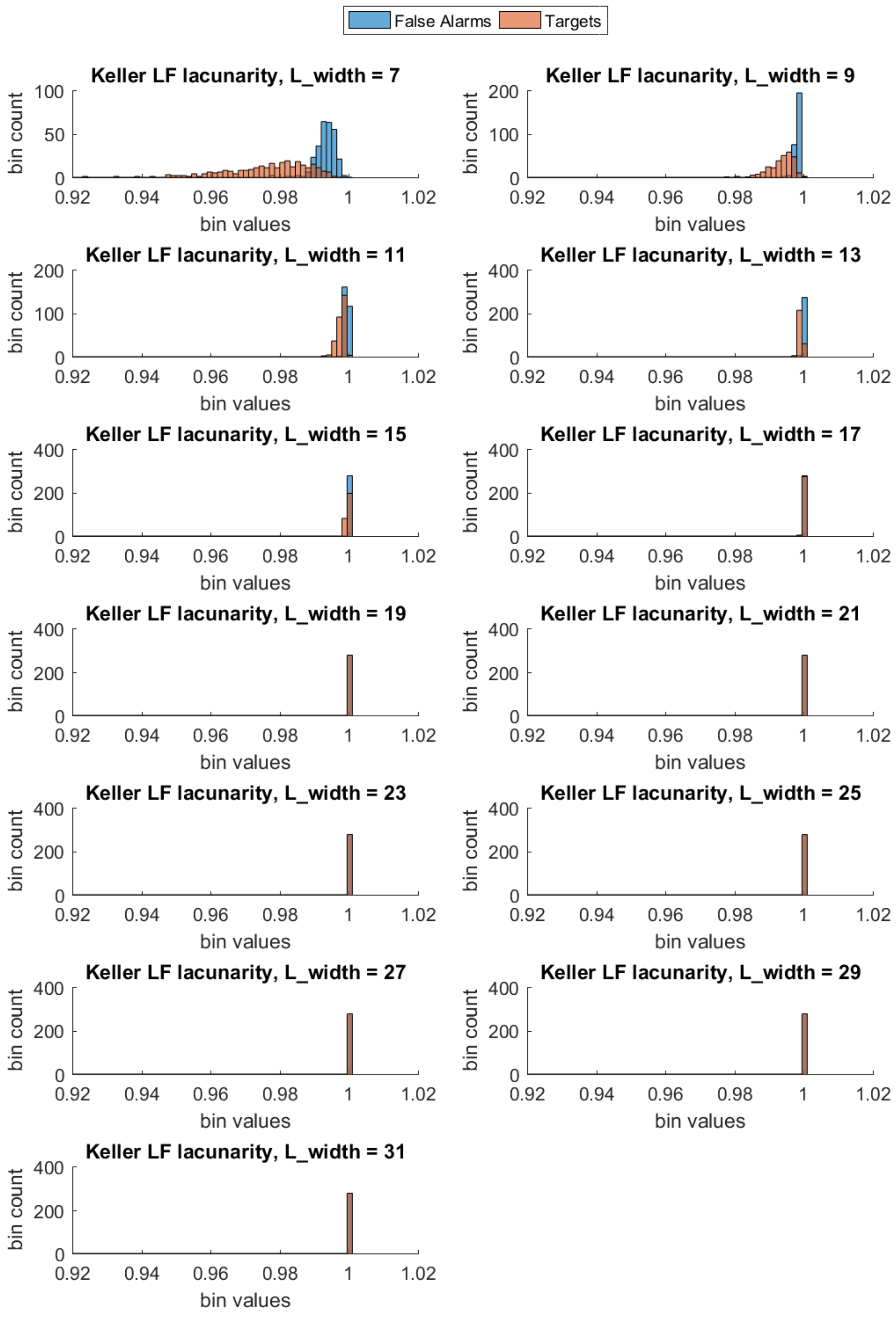

Figure A-6 – Histograms of computed Keller lacunarity features on LF imagery target tiles and tiles in FA Set 1

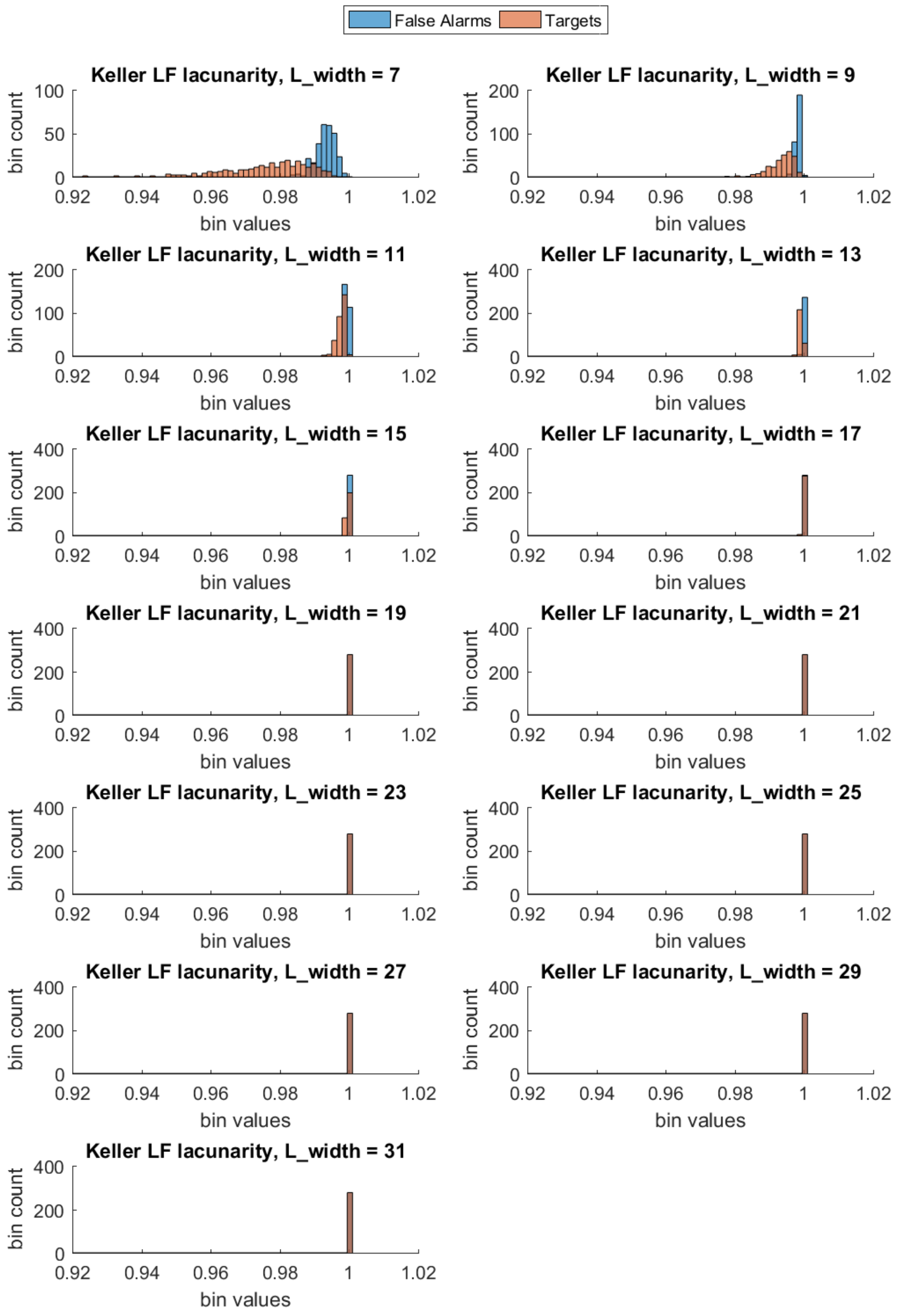

Figure A-7 – Histograms of computed Keller lacunarity features on LF imagery target tiles and tiles in FA Set 2

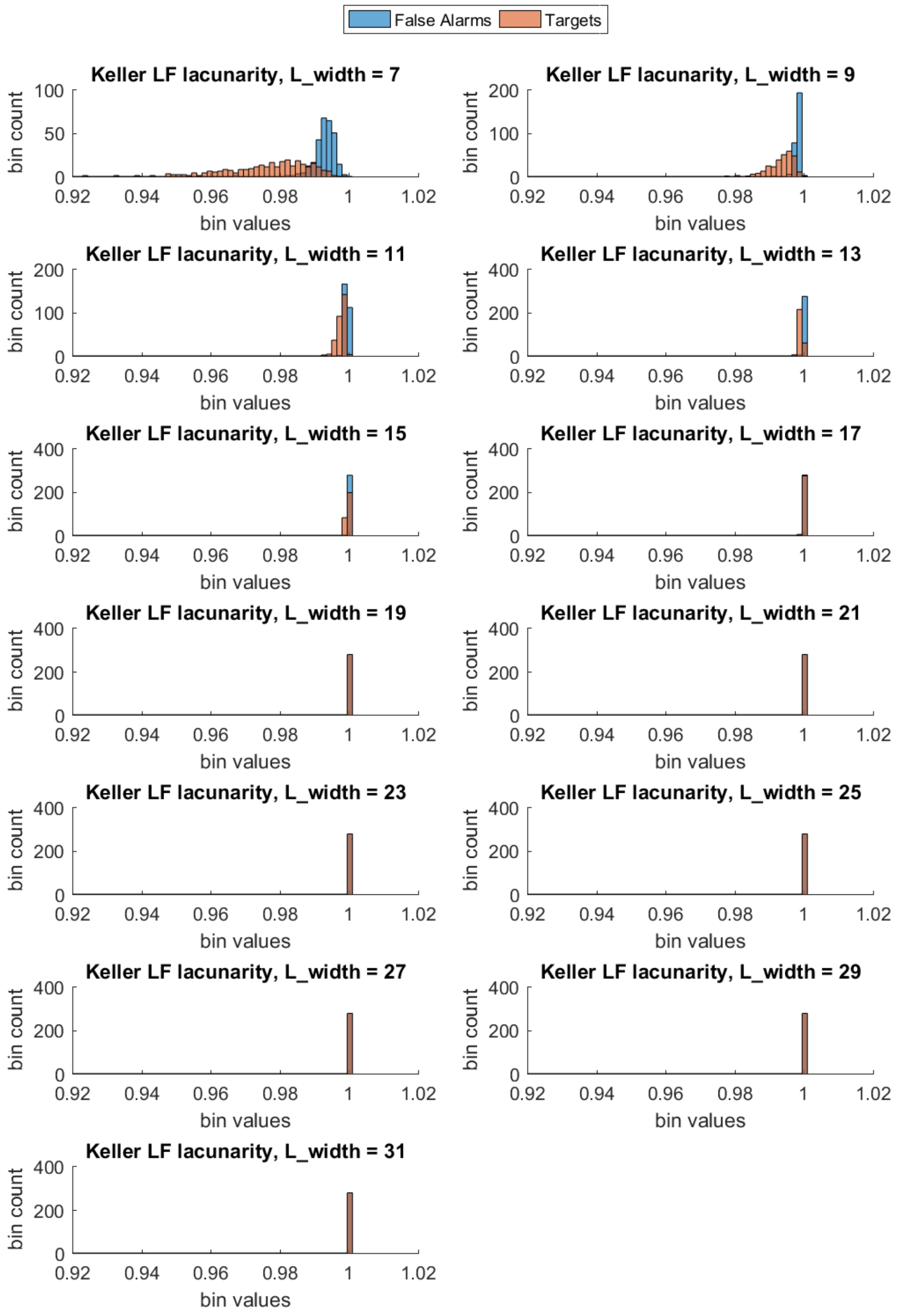

Figure A-8 – Histograms of computed Keller lacunarity features on LF imagery target tiles and tiles in FA Set 3

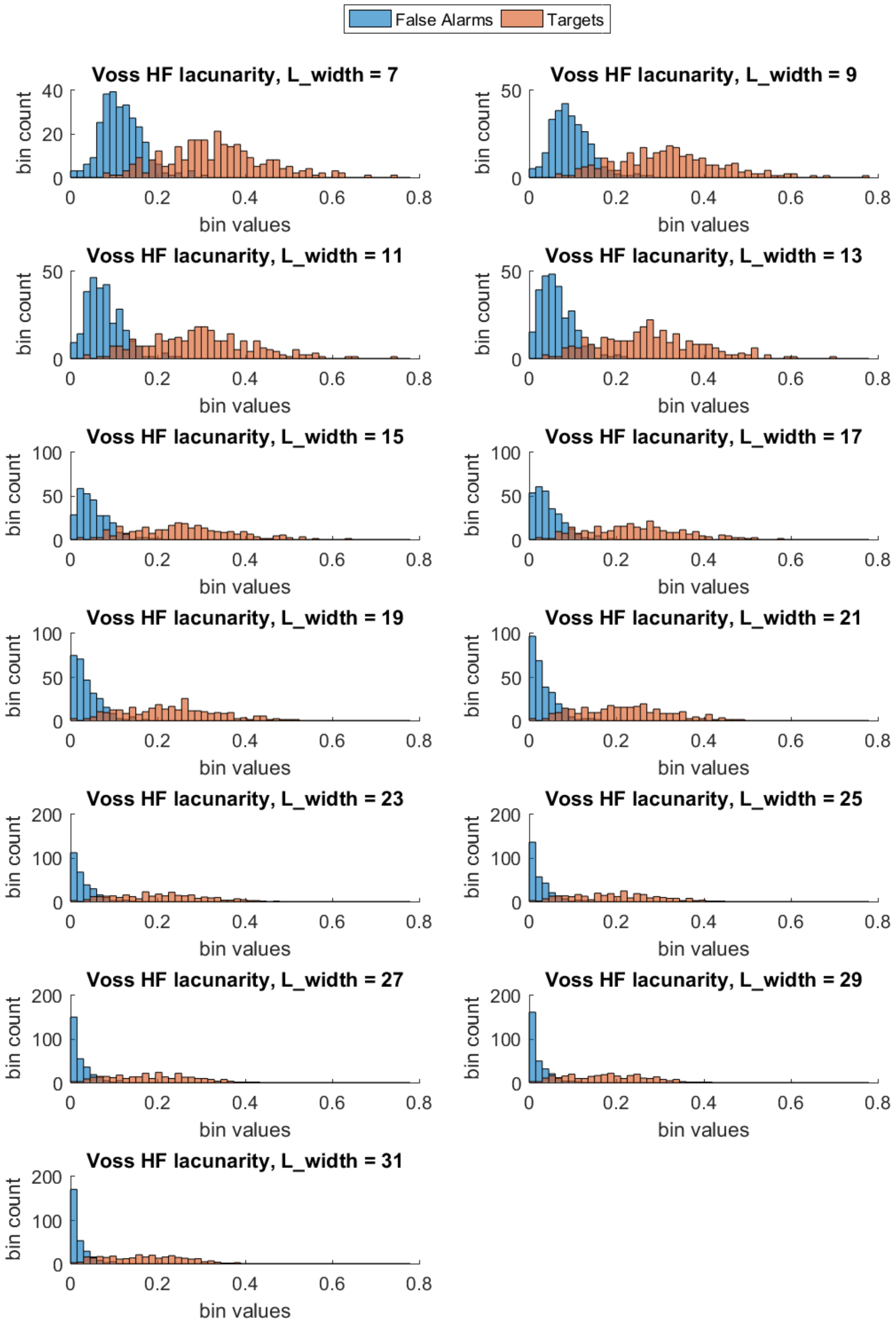

Figure A-9 – Histograms of computed Voss lacunarity features on HF imagery target tiles and tiles in FA Set 1

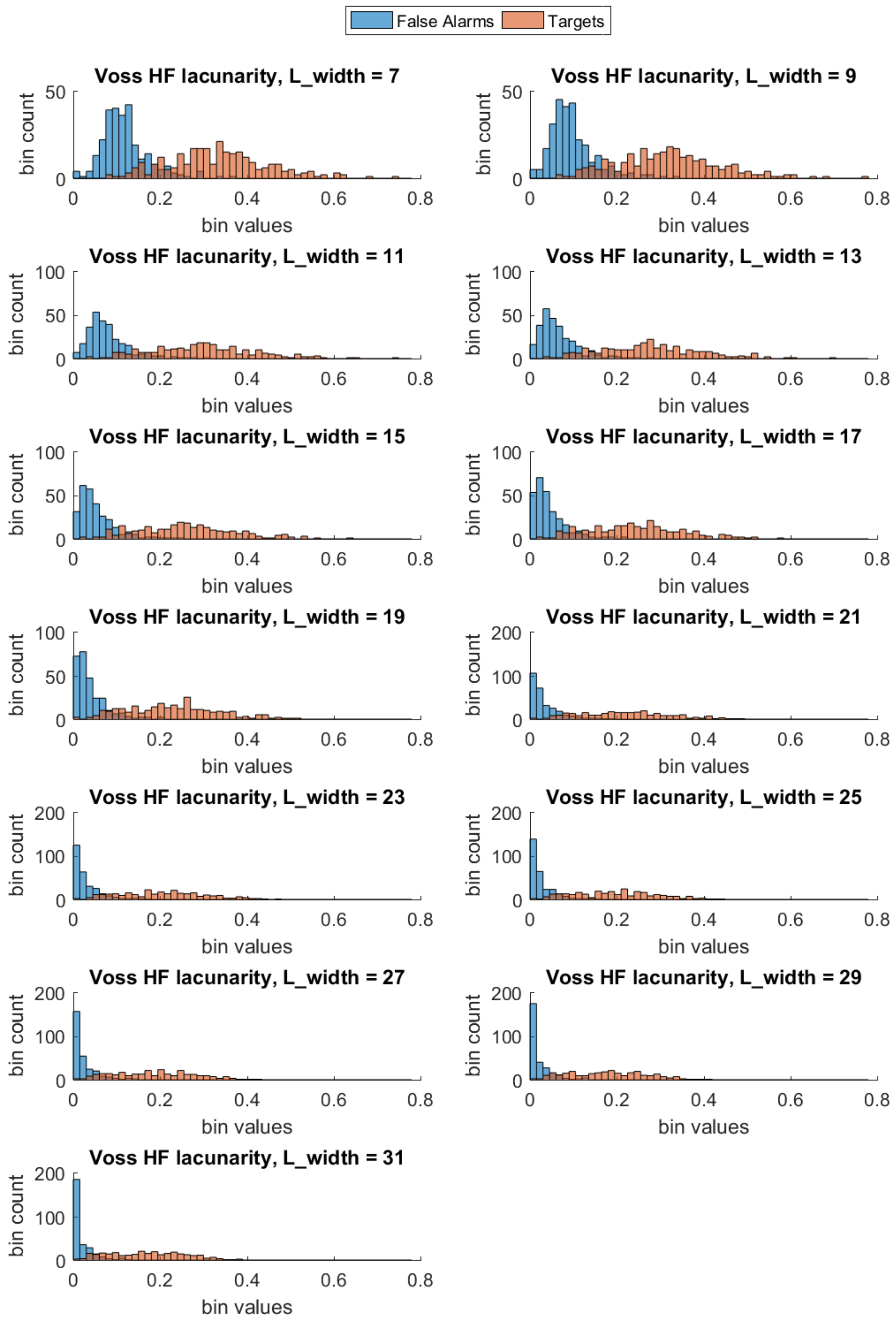

Figure A-10 – Histograms of computed Voss lacunarity features on HF imagery target tiles and tiles in FA Set 2

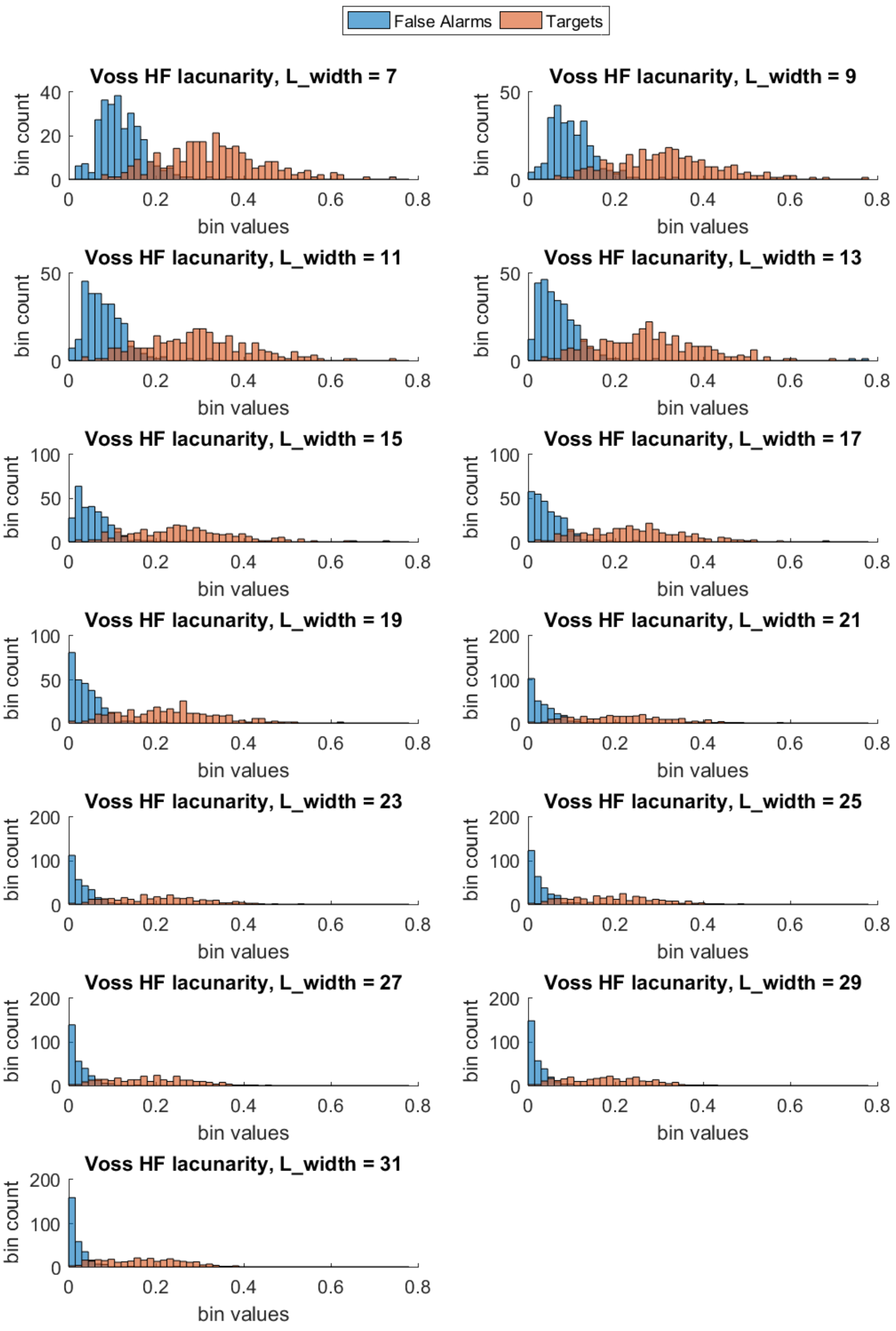

Figure A-11 – Histograms of computed Voss lacunarity features on HF imagery target tiles and tiles in FA Set 3

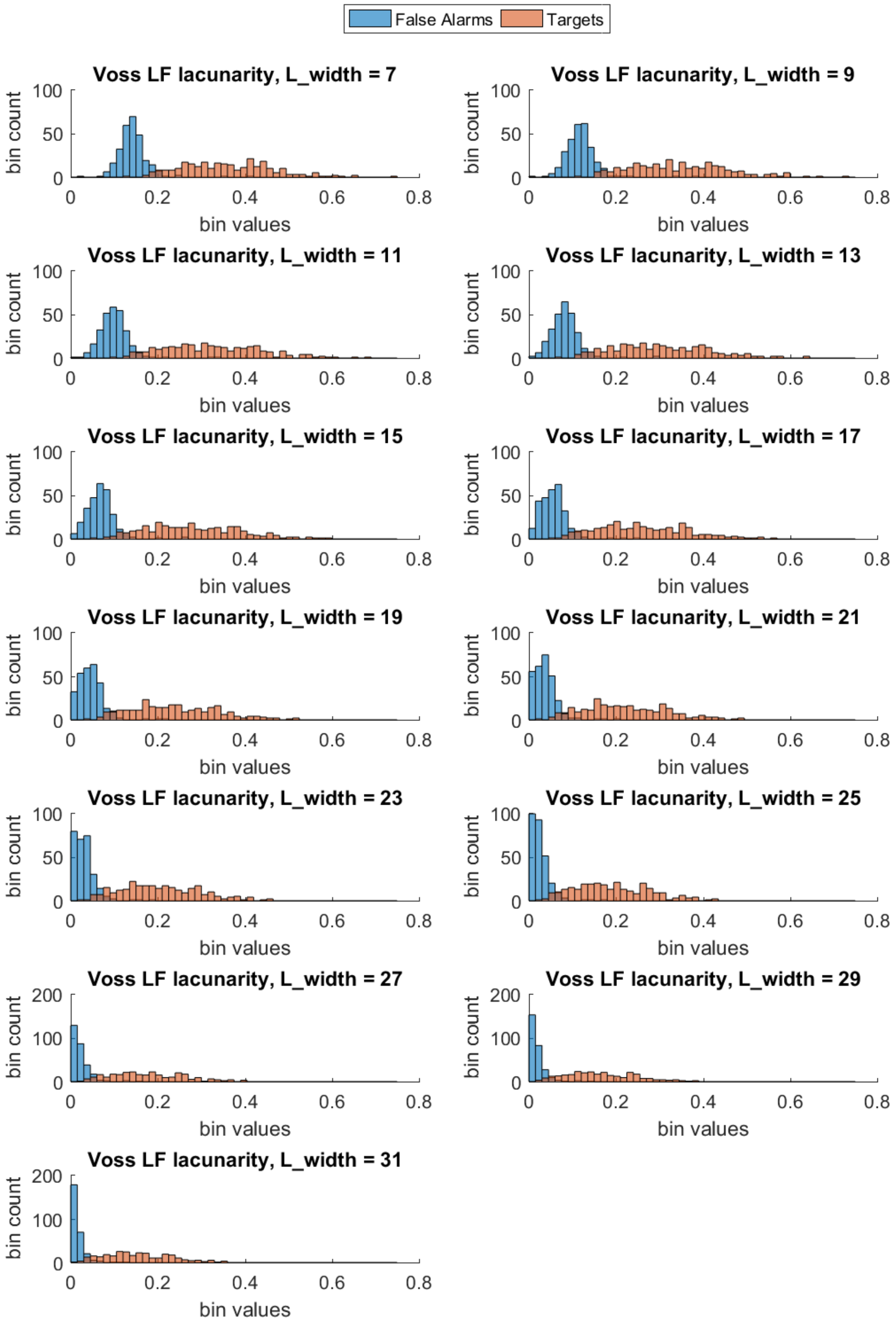

Figure A-12 – Histograms of computed Voss lacunarity features on LF imagery target tiles and tiles in FA Set 1

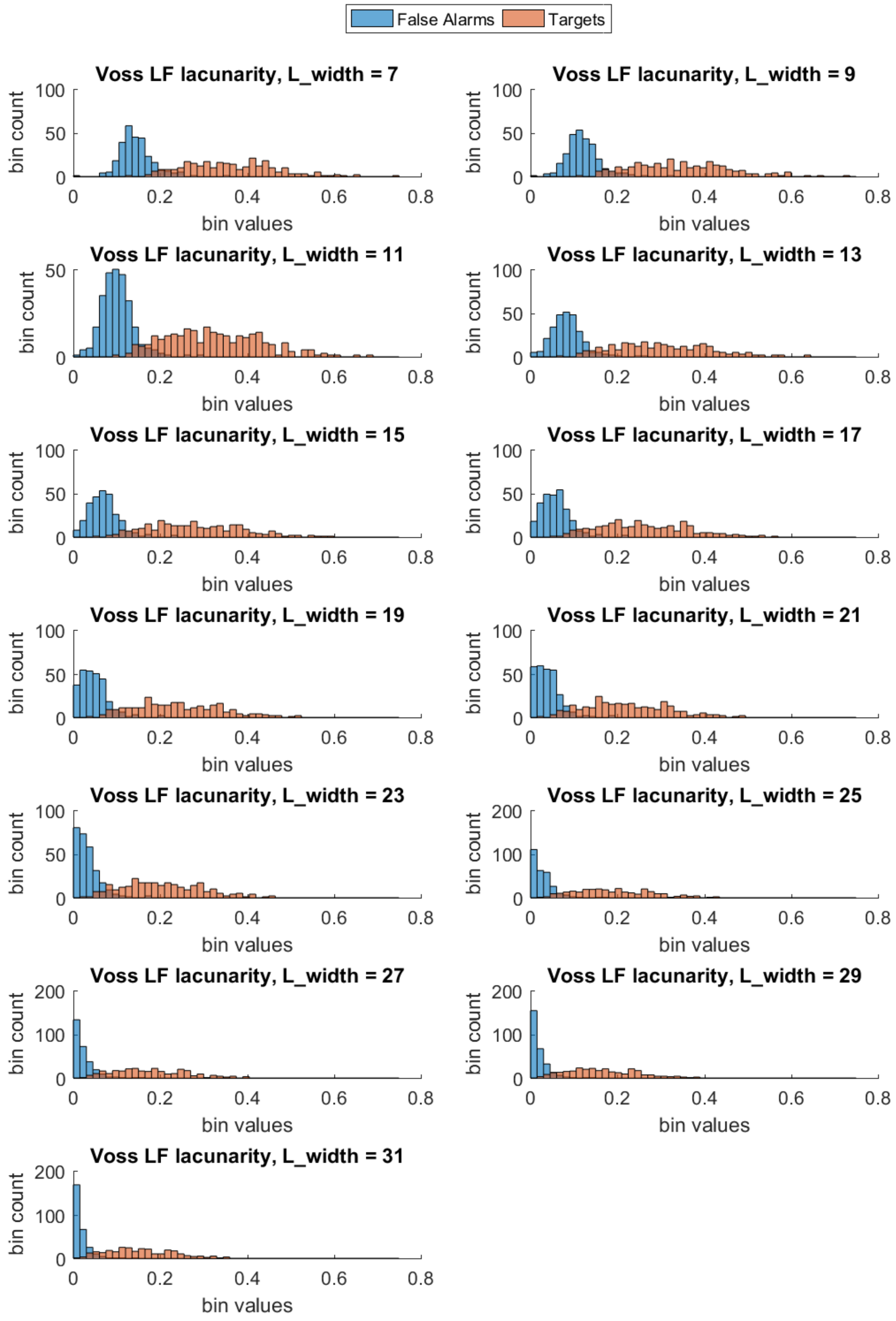

Figure A-13 – Histograms of computed Voss lacunarity features on LF imagery target tiles and tiles in FA Set 2

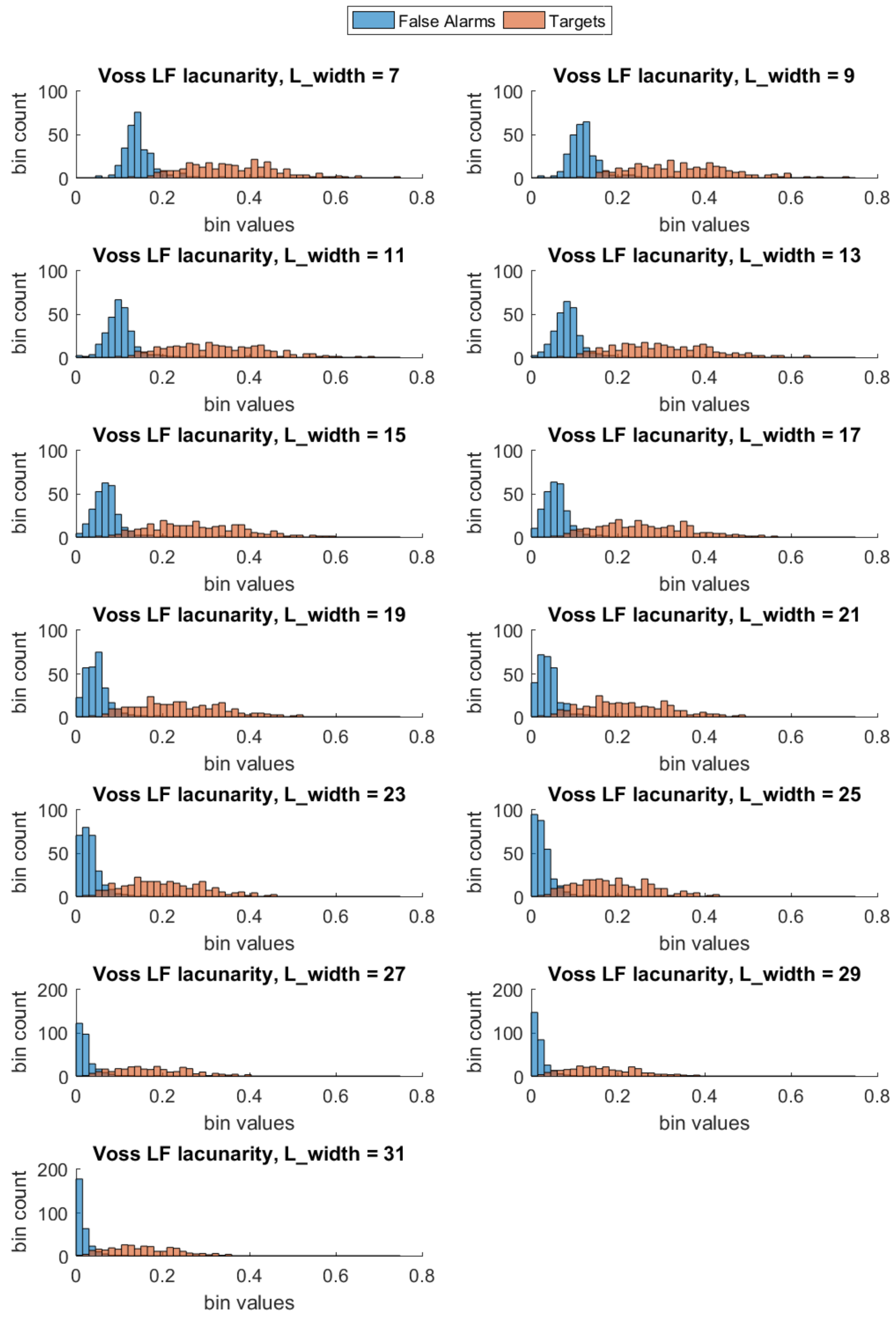

Figure A-14 – Histograms of computed Voss lacunarity features on LF imagery target tiles and tiles in FA Set 3

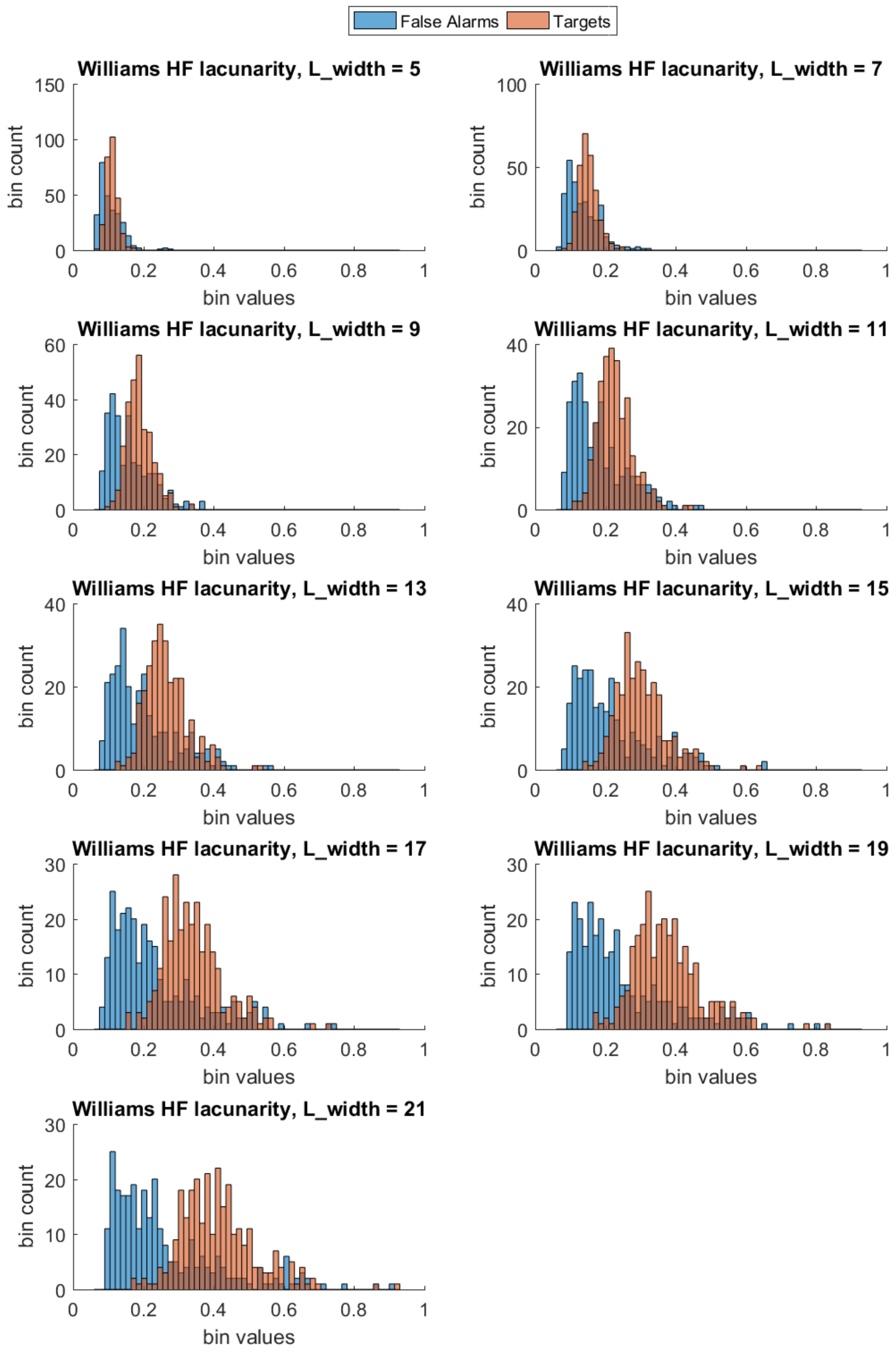

Figure A-15 – Histograms of computed Williams lacunarity features on HF target tiles and tiles imagery in FA Set 1

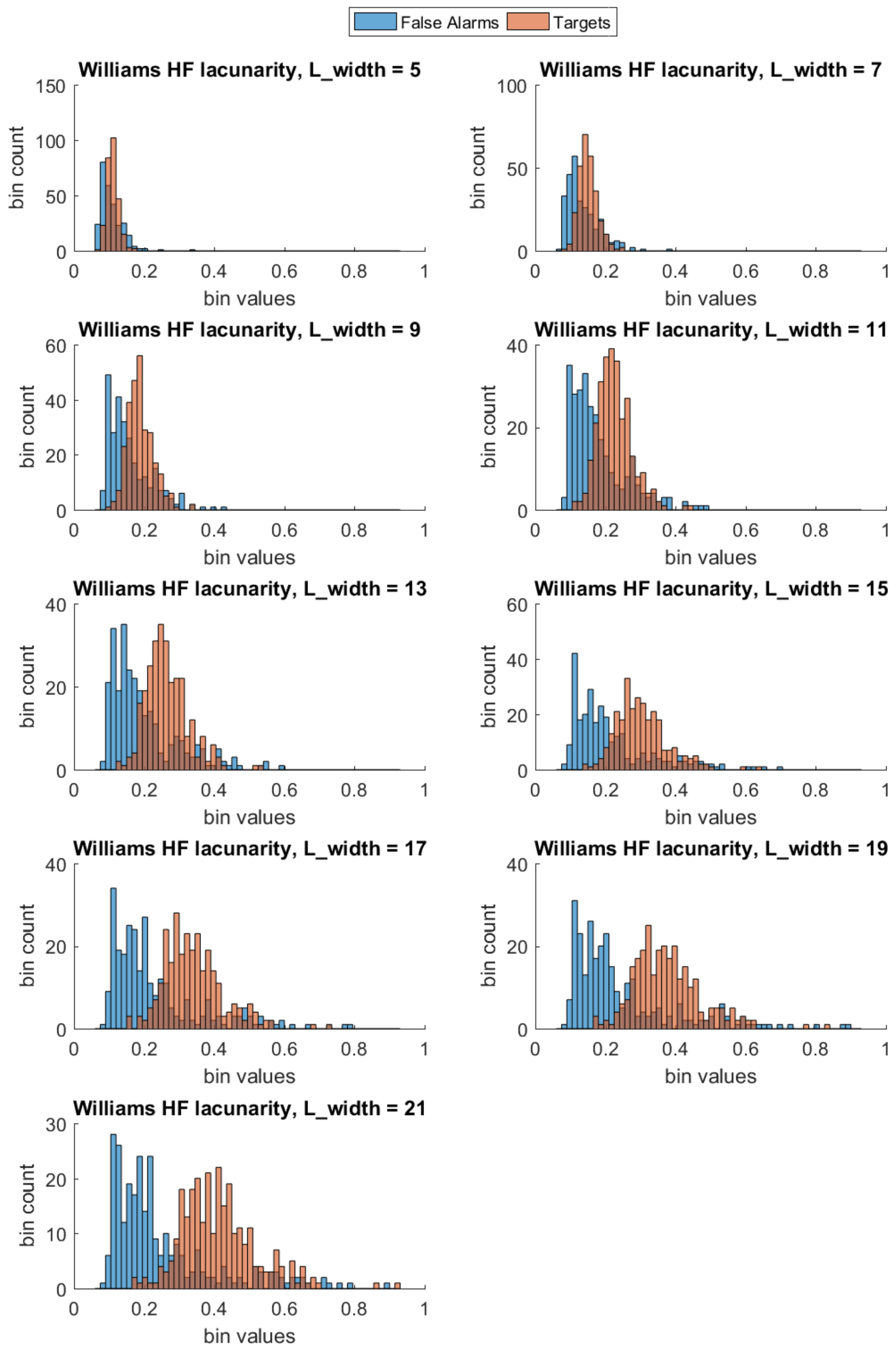

Figure A-16 – Histograms of computed Williams lacunarity features on HF target tiles and tiles imagery in FA Set 2

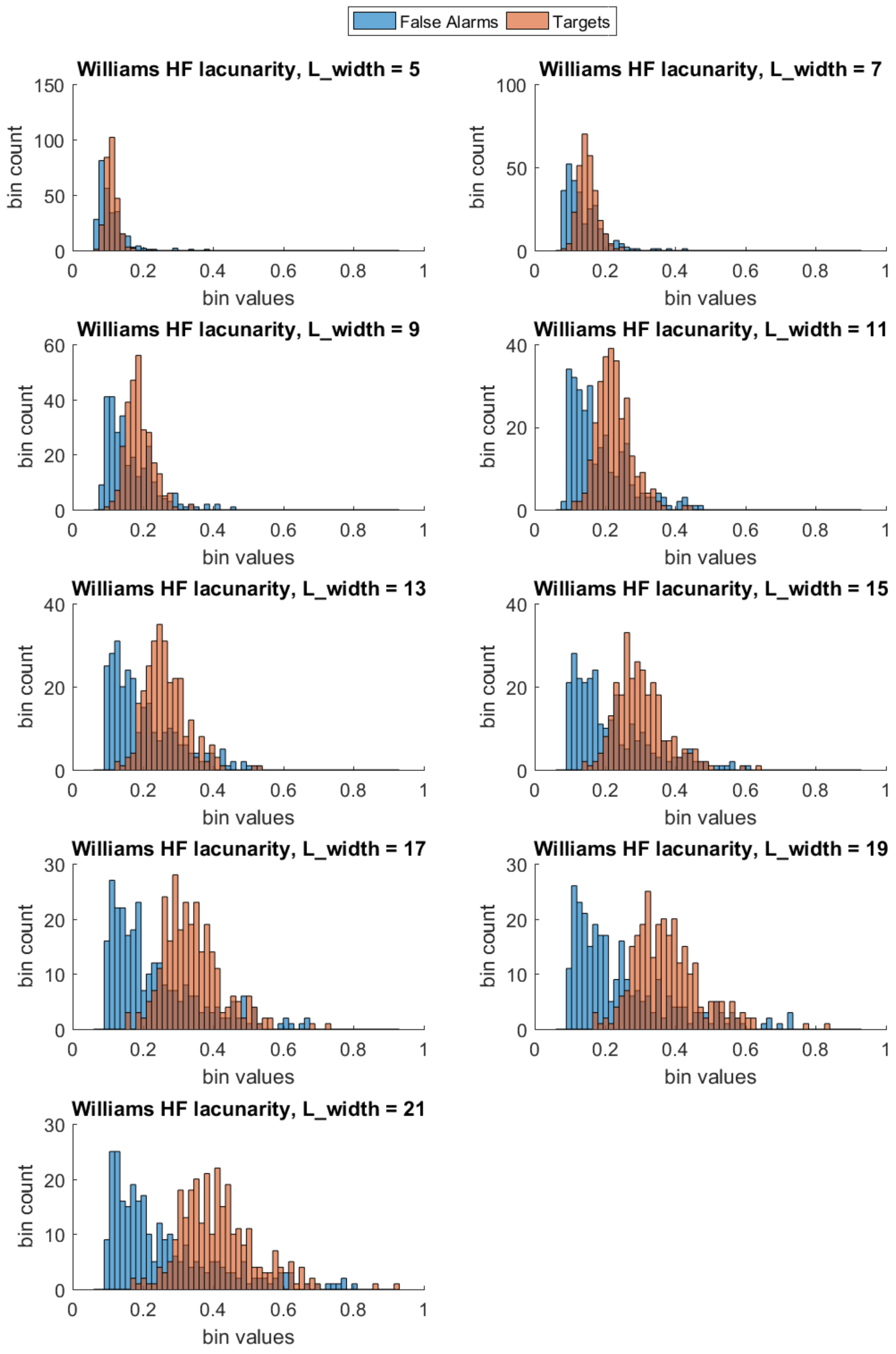

Figure A-17 – Histograms of computed Williams lacunarity features on HF target tiles and tiles imagery in FA Set 3

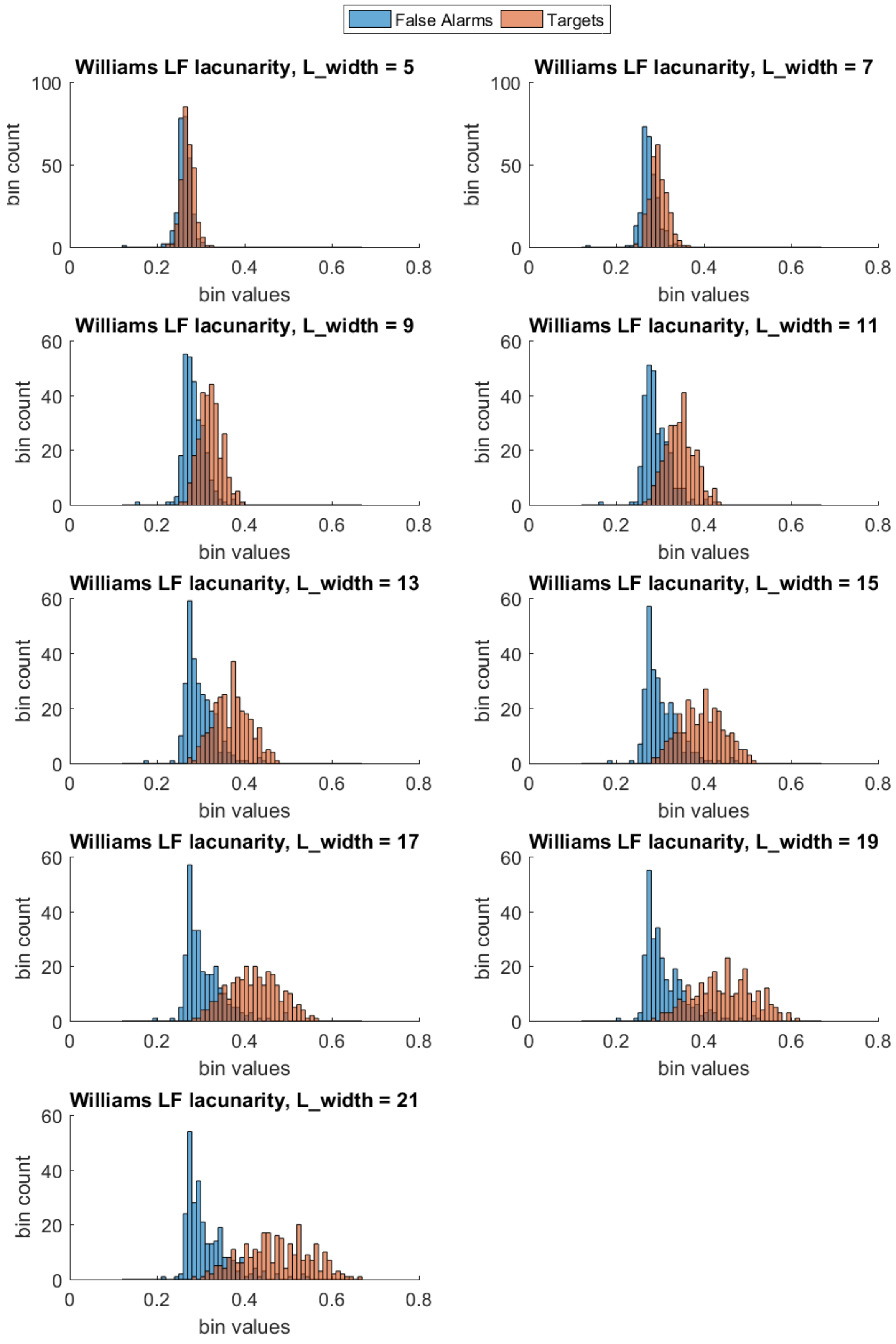

Figure A-18 – Histograms of computed Williams lacunarity features on LF target tiles and tiles imagery in FA Set 1

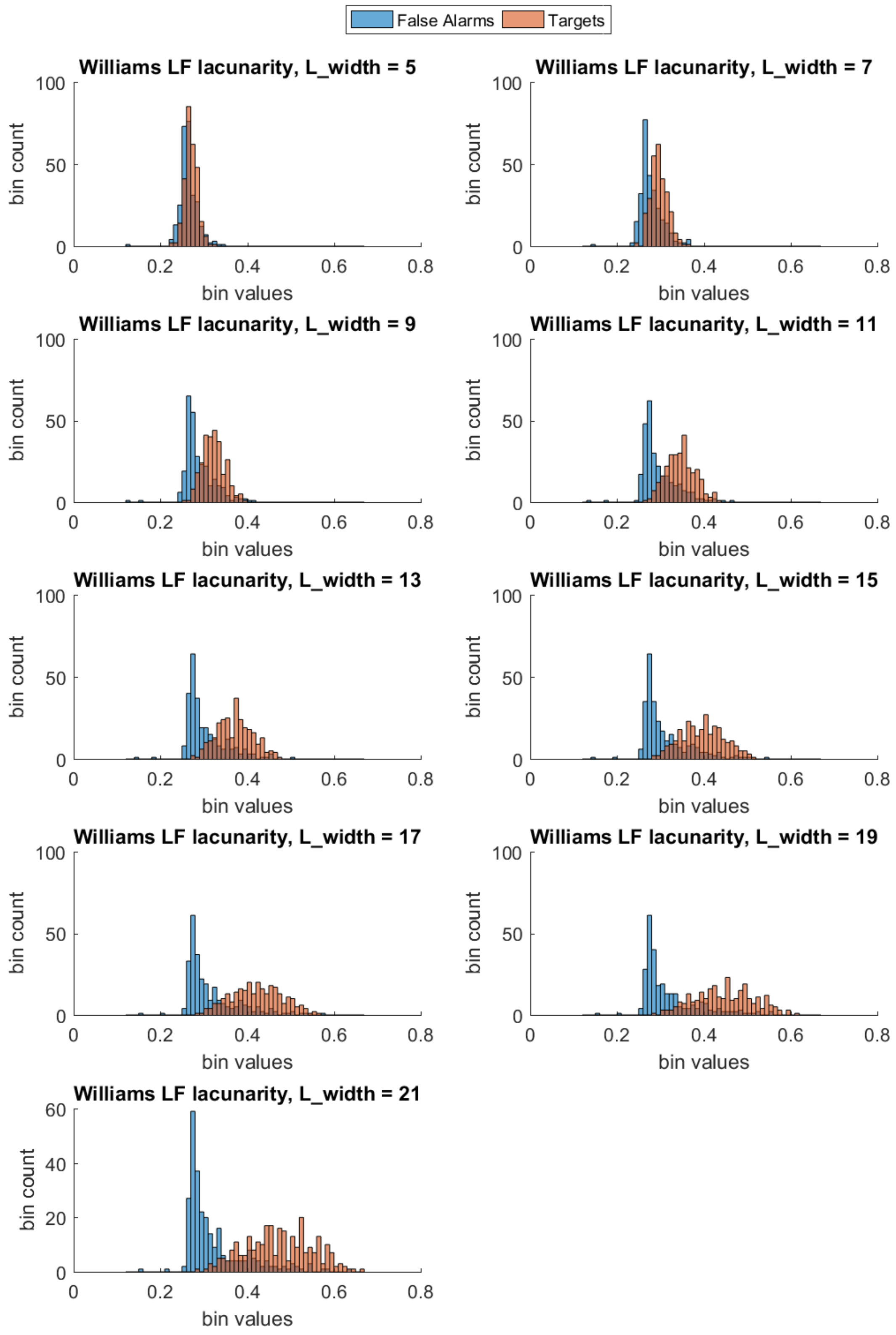

Figure A-19 – Histograms of computed Williams lacunarity features on LF target tiles and tiles imagery in FA Set 2

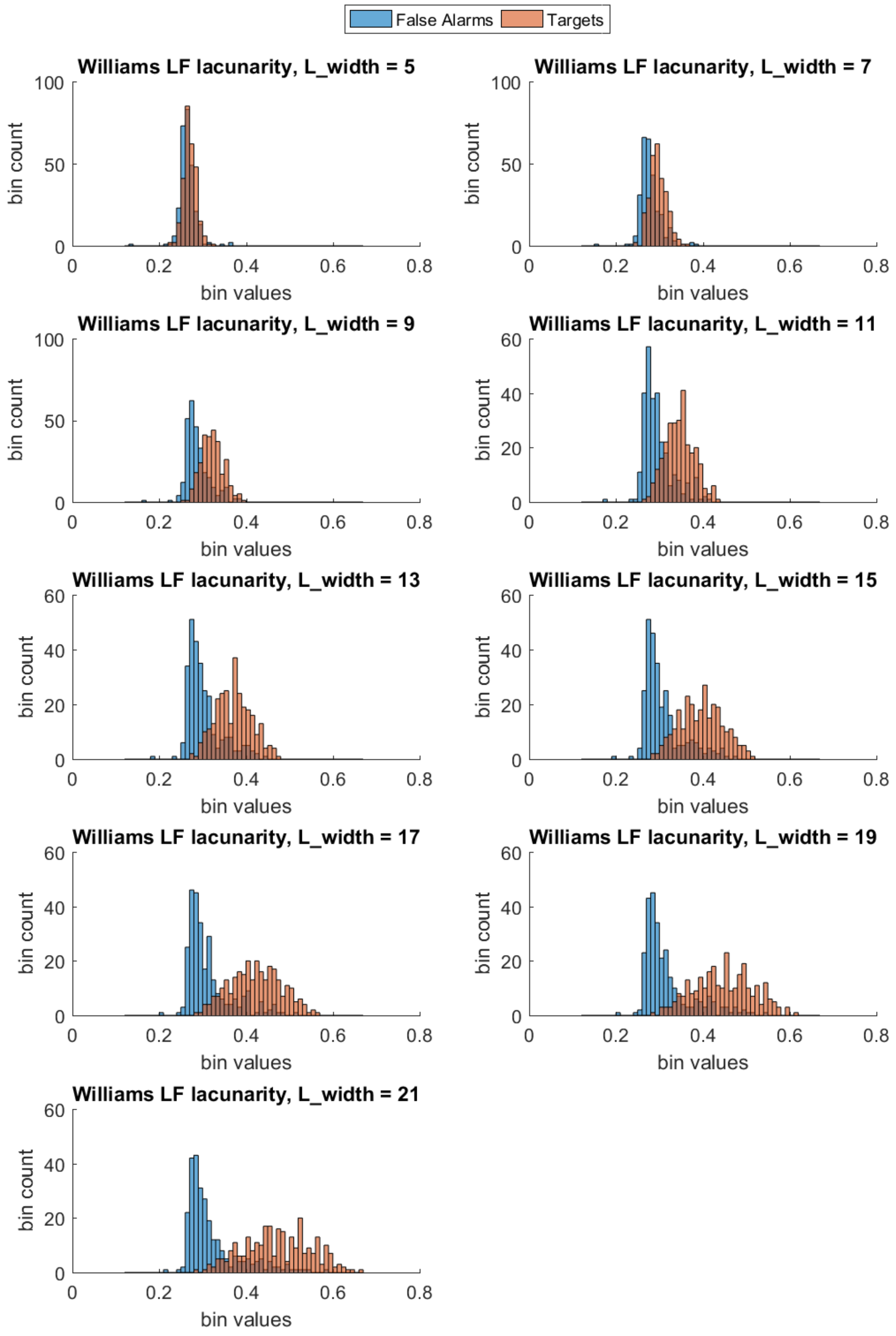

Figure A-20 – Histograms of computed Williams lacunarity features on LF target tiles and tiles imagery in FA Set 3

APPENDIX B

Our Forward search results are shown in Table B-1. The process to obtain these is explained in Section 3.3. Each column is colored independently with highest average classification accuracy in yellow, and second highest in bright green to the lowest in bright red. For the first column, each feature is used independently to determine its corresponding classification accuracy. The feature corresponding with the highest accuracy is selected. For each subsequent column, each feature is grouped separately with the previously selected feature(s) and those groups are used to compute the values for the current column. After each column of values is computed, the feature corresponding with the highest value is selected and added to the ensemble of selected features.

Table B-1 – Forward search results

1	Meth 1 slope HF	49.13%	95.74%	97.33%	97.94%	97.87%	97.51%
2	Meth 1 inter HF	71.95%	96.07%	97.40%	97.83%	97.98%	97.83%
3	Meth 2 slope HF	57.11%	96.46%	97.76%	97.80%	97.73%	97.58%
4	Meth 2 inter HF	51.77%	95.78%	97.87%	98.12%	98.01%	97.91%
5	Meth 3 slope HF	61.44%	96.14%	97.91%	98.05%	98.12%	98.01%
6	Meth 3 inter HF	90.40%	96.43%	96.79%	97.69%	97.87%	97.87%
7	Meth 1 slope LF	49.96%	96.10%	97.04%	97.91%	97.62%	97.91%
8	Meth 1 inter LF	51.55%	96.10%	97.08%	97.91%	97.83%	97.76%
9	Meth 2 slope LF	50.25%	95.78%	97.11%	97.55%	97.62%	97.83%
10	Meth 2 inter LF	62.67%	95.74%	97.15%	97.73%	97.87%	97.94%
11	Meth 3 slope LF	66.25%	95.78%	97.47%	98.34%	97.94%	98.09%
12	Meth 3 inter LF	92.24%	95.88%	97.33%	97.83%	97.91%	98.05%
13	Keller Lac HF, L_width = 07	81.23%	96.21%	97.98%	98.05%	98.12%	98.16%
14	Keller Lac HF, L_width = 09	79.96%	96.28%	97.51%	98.27%	98.16%	98.27%
15	Keller Lac HF, L_width = 11	87.04%	96.14%	97.65%	98.20%	98.20%	98.05%
16	Keller Lac HF, L_width = 13	88.81%	96.53%	97.08%	98.41%	98.01%	98.05%
17	Keller Lac HF, L_width = 15	88.52%	96.68%	97.40%	98.23%	98.05%	98.34%
18	Keller Lac HF, L_width = 17	90.40%	96.68%	97.58%	98.09%	98.30%	98.27%
19	Keller Lac HF, L_width = 19	87.26%	96.53%	97.37%	98.09%	97.83%	98.09%
20	Keller Lac HF, L_width = 21	90.79%	96.50%	97.51%	97.73%	98.09%	98.09%
21	Keller Lac HF, L_width = 23	90.00%	96.86%	97.33%	98.12%	97.80%	98.05%
22	Keller Lac HF, L_width = 25	91.01%	97.04%	97.44%	98.12%	97.83%	98.09%

23	Keller Lac HF, L_width = 27	89.57%	96.93%	97.37%	97.94%	98.12%	98.05%
24	Keller Lac HF, L_width = 29	91.30%	96.79%	97.18%	97.80%	97.98%	97.98%
25	Keller Lac HF, L_width = 31	91.77%	97.04%	97.33%	97.65%	98.20%	98.01%
26	Voss Lac HF, L_width = 07	90.72%	97.08%	97.22%	97.98%	97.98%	98.09%
27	Voss Lac HF, L_width = 09	90.98%	96.72%	97.37%	98.01%	97.91%	97.91%
28	Voss Lac HF, L_width = 11	91.05%	97.04%	97.51%	97.98%	97.94%	98.16%
29	Voss Lac HF, L_width = 13	91.48%	97.11%	97.04%	98.05%	97.91%	98.16%
30	Voss Lac HF, L_width = 15	91.30%	96.97%	97.26%	97.83%	98.30%	97.91%
31	Voss Lac HF, L_width = 17	91.19%	97.04%	97.44%	97.76%	98.01%	97.98%
32	Voss Lac HF, L_width = 19	90.98%	97.22%	97.58%	97.58%	97.91%	97.98%
33	Voss Lac HF, L_width = 21	91.05%	97.15%	97.40%	97.91%	97.98%	98.12%
34	Voss Lac HF, L_width = 23	91.30%	97.26%	97.26%	97.83%	98.16%	98.20%
35	Voss Lac HF, L_width = 25	91.44%	97.40%	97.37%	97.58%	98.01%	98.30%
36	Voss Lac HF, L_width = 27	91.66%	97.40%	97.37%	97.80%	97.94%	98.20%
37	Voss Lac HF, L_width = 29	91.59%	97.47%	97.33%	97.76%	98.09%	98.23%
38	Voss Lac HF, L_width = 31	91.59%	97.51%	97.29%	98.01%	98.09%	98.27%
39	Williams Lac HF, L_width = 05	43.18%	96.14%	97.62%	97.69%	97.80%	98.09%
40	Williams Lac HF, L_width = 07	47.22%	96.28%	97.76%	98.27%	98.09%	97.80%
41	Williams Lac HF, L_width = 09	53.72%	96.43%	98.27%	98.20%	98.30%	98.27%
42	Williams Lac HF, L_width = 11	52.31%	96.28%	98.01%	98.20%	98.30%	98.38%
43	Williams Lac HF, L_width = 13	51.16%	95.99%	98.16%	97.94%	98.38%	98.20%
44	Williams Lac HF, L_width = 15	56.82%	95.96%	98.12%	97.94%	98.34%	98.16%
45	Williams Lac HF, L_width = 17	60.00%	95.70%	97.98%	98.05%	97.91%	98.20%
46	Williams Lac HF, L_width = 19	52.56%	95.88%	97.83%	98.05%	97.98%	98.05%
47	Williams Lac HF, L_width = 21	57.91%	95.63%	97.87%	98.01%	98.09%	98.12%
48	Keller Lac LF, L_width = 07	90.04%	95.52%	97.26%	97.87%	97.55%	97.98%
49	Keller Lac LF, L_width = 09	89.60%	95.78%	97.51%	98.09%	98.01%	97.87%
50	Keller Lac LF, L_width = 11	91.34%	95.88%	97.29%	97.91%	97.83%	97.91%
51	Keller Lac LF, L_width = 13	92.02%	95.60%	97.33%	97.83%	97.69%	98.12%
52	Keller Lac LF, L_width = 15	91.84%	96.21%	97.44%	97.80%	97.80%	98.12%
53	Keller Lac LF, L_width = 17	92.02%	95.78%	97.22%	97.94%	97.98%	98.20%
54	Keller Lac LF, L_width = 19	92.27%	96.14%	97.33%	98.09%	97.98%	97.69%
55	Keller Lac LF, L_width = 21	92.89%	95.92%	97.55%	98.30%	97.76%	98.12%
56	Keller Lac LF, L_width = 23	92.96%	96.25%	97.33%	98.20%	97.87%	98.12%
57	Keller Lac LF, L_width = 25	93.86%	96.07%	97.15%	97.83%	97.87%	97.98%
58	Keller Lac LF, L_width = 27	93.97%	96.10%	97.51%	97.87%	97.80%	97.87%
59	Keller Lac LF, L_width = 29	93.94%	96.03%	97.51%	97.94%	97.83%	97.98%
60	Keller Lac LF, L_width = 31	93.54%	96.03%	97.40%	98.01%	98.16%	98.20%
61	Voss Lac LF, L_width = 07	95.92%	95.67%	97.62%	97.94%	97.98%	97.83%
62	Voss Lac LF, L_width = 09	96.10%	95.92%	97.76%	98.01%	97.98%	98.05%
63	Voss Lac LF, L_width = 11	96.07%	95.96%	97.55%	97.94%	97.94%	98.27%
64	Voss Lac LF, L_width = 13	95.74%	95.88%	97.44%	97.98%	97.80%	98.27%

65	Voss Lac LF, L_width = 15	95.56%	96.07%	97.55%	97.80%	98.12%	98.05%
66	Voss Lac LF, L_width = 17	95.31%	96.03%	97.47%	97.94%	98.01%	97.94%
67	Voss Lac LF, L_width = 19	95.13%	95.96%	97.58%	97.87%	97.87%	98.30%
68	Voss Lac LF, L_width = 21	94.95%	95.96%	97.51%	97.69%	97.91%	97.94%
69	Voss Lac LF, L_width = 23	94.87%	96.17%	97.26%	97.94%	97.94%	98.01%
70	Voss Lac LF, L_width = 25	95.13%	96.03%	97.33%	97.87%	98.01%	97.94%
71	Voss Lac LF, L_width = 27	94.98%	95.96%	97.29%	97.98%	97.73%	98.09%
72	Voss Lac LF, L_width = 29	95.20%	96.07%	97.29%	98.05%	98.01%	97.87%
73	Voss Lac LF, L_width = 31	95.31%	96.10%	97.47%	97.87%	98.16%	97.94%
74	Williams Lac LF, L_width = 05	51.70%	95.09%	97.04%	97.40%	97.62%	97.65%
75	Williams Lac LF, L_width = 07	48.70%	95.99%	97.47%	97.73%	97.65%	97.62%
76	Williams Lac LF, L_width = 09	52.89%	95.81%	97.15%	97.62%	97.69%	97.55%
77	Williams Lac LF, L_width = 11	62.17%	95.85%	97.11%	97.62%	97.76%	97.65%
78	Williams Lac LF, L_width = 13	68.92%	95.96%	97.15%	98.16%	97.65%	97.65%
79	Williams Lac LF, L_width = 15	77.15%	95.67%	97.37%	97.80%	97.51%	97.69%
80	Williams Lac LF, L_width = 17	73.32%	95.81%	97.18%	97.87%	97.76%	97.83%
81	Williams Lac LF, L_width = 19	76.75%	95.27%	97.18%	97.83%	97.80%	97.65%
82	Williams Lac LF, L_width = 21	78.45%	95.34%	97.26%	97.69%	97.76%	97.76%
Highest average classification accuracy achieved		96.1%	97.5%	98.3%	98.4%	98.4%	98.4%
Selected feature		62	38	41	16	43	42

APPENDIX C

As discussed in Section 3.5, this appendix contains additional images showing the error between the Method 3 features computed using a sliding window and those features computed using interpolation from a much lower resolution feature image.

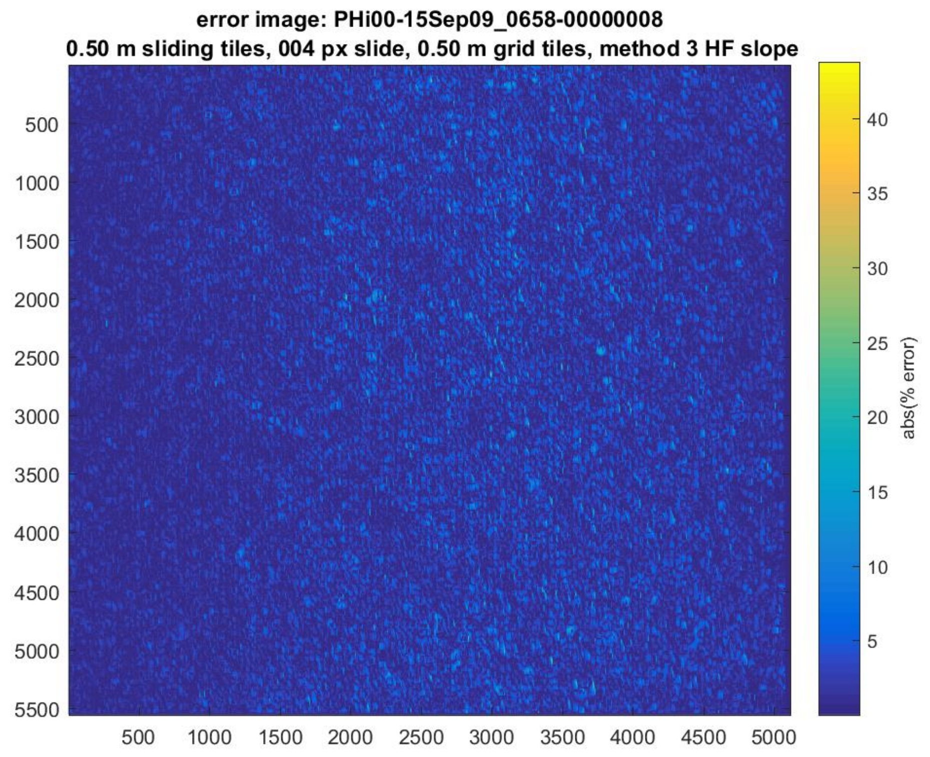

Figure C-1 – Image of percent error between low-resolution feature image and sliding tile feature image for Method 3 HF slope feature, mean error is 1.94%, std of error is 1.93%

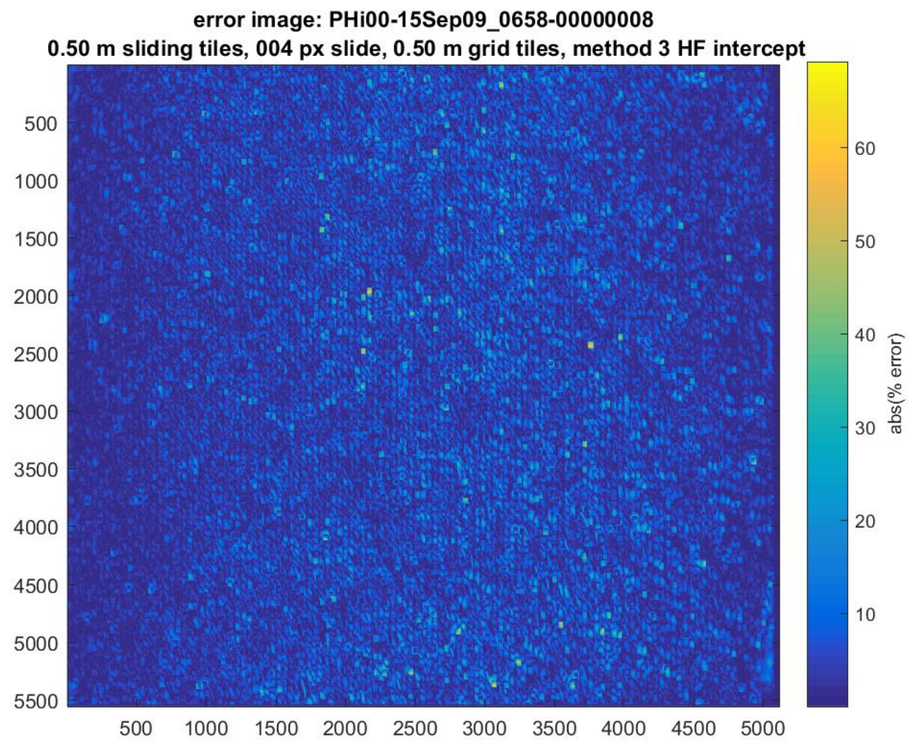

Figure C-2 – Image of percent error between low-resolution feature image and sliding tile feature image for Method 3 HF intercept feature, mean error is 4.78%, std of error is 4.72%

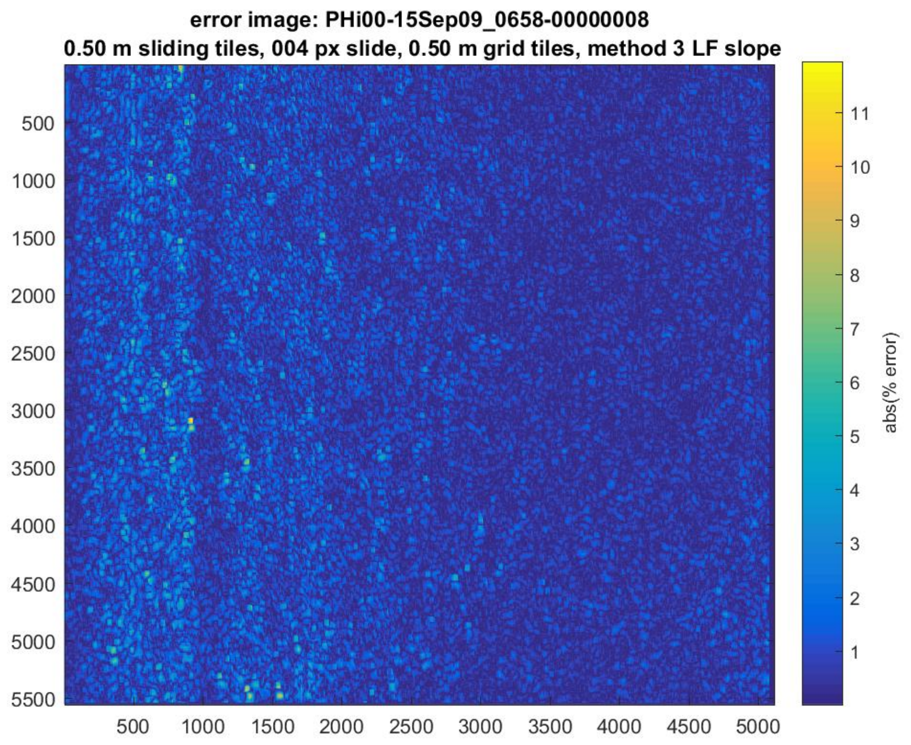

Figure C-3 – Image of percent error between low-resolution feature image and sliding tile feature image for Method 3 LF slope feature, mean error is 0.76%, std of error is 0.74%

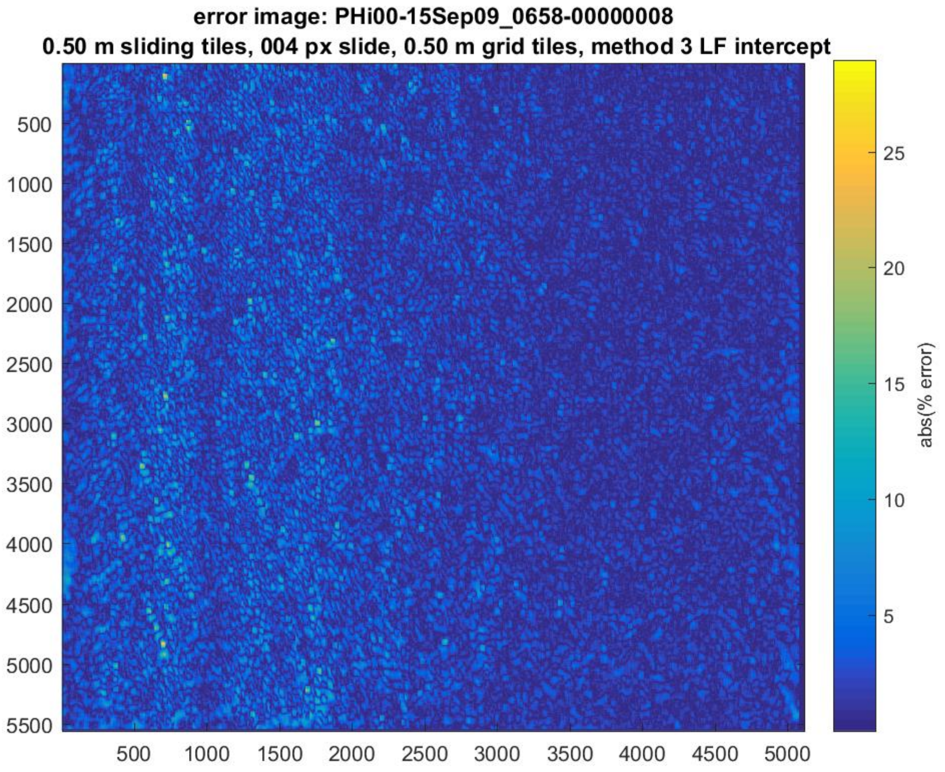

Figure C-4 – Image of percent error between low-resolution feature image and sliding tile feature image for Method 3 LF intercept feature, mean error is 1.96%, std of error is 1.82%