Frequency Domain Improvements to Texture Discrimination Algorithms

Ibrahim Cem Baykal
Adana Alparslan Turkes Science and Technology University
Computer Engineering Department
Sarıçam, Adana, Turkiye

***Abstract*—As the production speeds of factories increase, it becomes more and more challenging to inspect products in real time. The goal of this article is to come up with a computationally efficient texture discrimination algorithm by first testing their ability to localize defects and then increase their efficiency by removing less effective parts of them. Therefore, abilities of the most popular texture classification algorithms such as the GLCM, the LBP and the SDH to localize defects are tested on different datasets. These tests reveal that, on small windows GLCM and SDH perform better. Frequency properties of the textures are used to fine-tune the parameters of these algorithms. Further experiments on three different datasets prove that the accuracy of the algorithms are increased almost twice while decreasing the processing time considerably.**

***Keywords*—Machine vision; ANN; SVM; pattern recognition; co-occurrence; texture feature extraction**

I. Introduction

Despite being old, based on the yearly citations they receive, Haralick's grey level co-occurrence matrices (GLCM)[1], Unser's sum and difference histograms (SDH)[2] and Local Binary Patterns (LBP)[3] are the most popular texture classifiers in the literature. As shown in Fig. 1, great majority of machine vision systems inspect a product simultaneously while it is being produced. Usually, imaging sensors are stationary and the product moves under them. This causes the viewing angle of the cameras to always be static with respect to the product. The lighting conditions in a factory are usually under some degree of control and do not change radically. When used for such purpose, these algorithms are usually applied to the texture towards the same alignment because in a factory the camera is usually at a fixed position with respect to the production line or the conveyor belt. Therefore, throughout the experiments performed in this article, texture alignment is fixed.

Another difference between the texture recognition and the discrimination is that the recognition applications process the entire image while the discrimination applications process smaller windows to localize where the defect is. Unfortunately, there are not many experimental studies in the literature that measures the effectiveness of algorithms on different (non-overlapping) window sizes. Texture segmentation [4] applications use sliding windows which makes them very slow.

Because of the rapid progress of technology, materials in factories are produced faster and faster. Flat surfaced products such as wood [5], fabric [6] and metal [7] require machine vision systems to detect production defects. Unfortunately, high-speed production of these goods combined with the availability of high-resolution cameras put great strain on the network and the computers that process those images in real time. A typical line-scan camera, employed in web inspection tasks for products such as film, paper and fabric, generates between 15 and 150 MB of data per second and half a dozen cameras are required to cover the entire width of the web, meaning that gigabytes of data must be reliably transferred and processed. One approach to this problem is to pre-process images inside the cameras and send the image data to the main computer only when the camera suspects that there might be a defect [8]. In this study, Brodatz texture set is experimented on with popular texture classification algorithms to see which one performs better on small windows. Based on these results, improvements to these existing algorithms are proposed. These modified algorithms are tested on three different datasets.

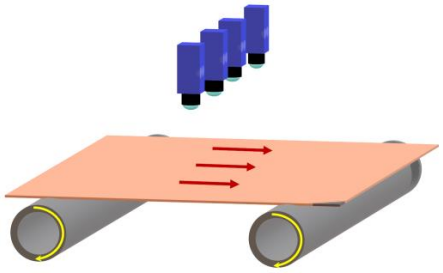


Fig. 1. Simultaneous manufacturing and inspection of a flat product inside a factory.

II. Related Work

GLCM are calculated by counting the relative pixel value occurrences that are away from each other (d) pixels. The result is a histogram matrix of size N×N. N is defined by the maximum number of gray levels in the image. Therefore, for greyscale images it is usually 256×256=64K elements. Haralick proposes 14 features that can be calculated from the GLCM, which requires processing of those 64K elements 14 times. This not only requires considerable amount of processing power but also consumes several 64 KB of memory. Considering the fact that the GLCM was invented in 1973, this was a problem for the computers back then. One solution to overcome this problem was to use 16 grey levels for the image. That solution reduces the matrix size to 16×16.

Unser proposed another optimization to this problem. He mathematically proved that almost the same results could be

achieved by using the sum and difference histograms (SDH) of those pixel pairs. Instead of a matrix, his algorithm generates two vectors, one for the sum of the pixels and one for the difference for every given distance 'd.' An image with 256 gray levels produces two vectors with 512 elements. Calculating those 14 features that Haralick suggested becomes computationally simpler. This method's main advantage is the fact that it reduces the amount of memory. However, based on our experiments (see Section VIII) it increases the computational complexity compared to the GLCM if reduced number of gray levels are used. In the literature, SDH has been tested on many texture classification problems and it has been proved that it is a quite successful texture classifier.

Texture analysis has been the focus of numerous studies over the years and as a result, over two dozen algorithms have been invented to classify texture. In volume II of "Handbook of Computer Vision and Applications," Wagner [9] experimented on seven different texture sets using 18 different texture classification algorithms. His experiments showed that Unser's SDH is not only the most successful among the 18 algorithms, but also the second fastest. Based on the measurements of that study, the only algorithm faster than the SDH is the Local Textural Features [10]. However, that algorithm had the second lowest success rate. Therefore, according to that study, SDH is computationally the most efficient texture classifier. However, it should be noted that the GLCM was probably calculated using the full 256 gray levels making it much slower compared to the SDH. The second most successful algorithm is Chen's geometric features [11] which is the slowest among the 18 algorithms (30 times slower than the SDH). The third successful algorithm is the Gabor Filters [12], which is the second most popular algorithm used for fabric defect detection in the literature. Gabor Filters compare energy of the specific frequency bands on fabric images to detect defects. Our method to select pixel distance and the orientation is mathematically related to Gabor Filters. Another popular algorithm in the literature is Law's Masks [13]. Laws has suggested five 1-D convolution masks for feature extraction. From these masks 25 2-D filter masks can be constructed and features are calculated from them. According to Wagner [9], Law's Masks performed slightly worse than the SDH, but is eight times slower. Galloway's Run-Length Matrices [14] should also be mentioned because of their speed. His technique calculates characteristic textural features from gray-level run lengths in different image directions and then calculates five features from them for each direction. However his algorithm did not score as high as the previously mentioned algorithms according to Wagner.

The literature still lacks a mathematical definition of the texture. Therefore, humans are the only referee when it comes to judging which texture is the same or different. That is why an interesting experiment conducted by Petrou [15], comparing human perception to the texture classification algorithms is so important. In this study, classification performance of numerous algorithms were compared to a group of humans. Variants of both the GLCM and the SDH, with different gray levels were experimented with. These experiments showed that direct use of GLCM or SDH as input to a classifier resembles to human perception much closer than using only the features derived from them. SDH was measured as one of the fastest. SDH using16 gray levels is closer to human perception than the GLCM as well. In his exhaustive experiments, Gonzales-Rufino [16] shows that GLCM and SDH perform better against the Local Binary Patterns, and the Neighboring Grey Level Dependence Statistics [17]. In the literature, there is abundant evidence that second order pixel statistics based texture classification is one of the most successful and the most efficient method. All this evidence suggests that, further improving SDH might give us computationally the most efficient texture discrimination method.

III. Performance Comparison on Small Windows

In the literature, mathematical definitions of the texture is very vague: "A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying, or approximately periodic" [18]. Because of the fact that a texture is defined in terms of "statistics" and "periodicity," a texture can exist only as a group of pixels, or in other words as a window. As mentioned before, texture discrimination requires localization of texture aberrance. Therefore, we need to measure the performance of texture classifiers on varying size of windows.

For this experiment, four different size windows were used on 13 different Brodatz textures. The size of the windows are 240×240, 120×120, 60×60 and 30×30. As shown in Fig. 2, the right half of the images are used for training and the left halves are used for testing. Overlapping blue squares represent training locations and green squares represent non-overlapping test locations. The ones at the top use 120x120 pixel windows and the ones at the bottom use 30x30 pixel windows. Green boxes indicate correctly classified windows and red ones indicate falsely classified windows. The numbers inside the red boxes indicate which of the 13 textures that window was misclassified as.

For this experiment, six different feature sets were used. The first two feature set is the four features obtained from the GLCM. In the literature GLCM is mostly used in either 256 color mode or 16 color mode. Therefore, both versions were tried. The third feature set is the 32 gray level histogram plus Unser's difference histograms with 16 colors (H-32+DH-16). The final three feature sets are LBP with (r=1;P=8), (r=1,2;P=8) and (r,P)={(1,8);(2,8)};(3,12)}. Either Artificial Neural Networks (ANN) or Support Vector Machine (SVM) were used as classifiers. Table I shows that ANN is slightly more successful than the SVM. While LBP is more successful on larger windows, 16 color GLCM and (H-32+DH-16) are more successful on smaller windows. In other words, these algorithms are more suitable for defect localization or texture segmentation applications. However, readers must keep in mind that "window size" is a relative value, which depends on the image resolution. If the image of the texture is high resolution, than these window sizes must be increased. More detailed results of this experiment were shared in a conference article [19].

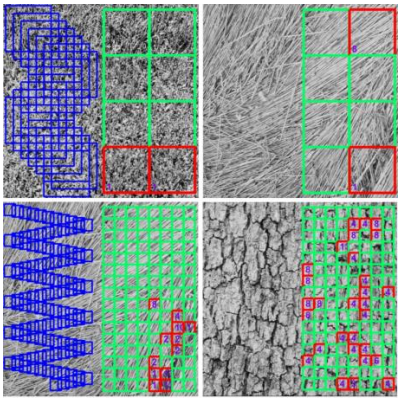


Fig. 2. Top: 120×120 pixel size window test. Bottom: 30×30 pixel size window test. Left half of the images are used for training. Overlapping blue boxes are training areas. Non-overlapping green windows on the right half of images are correctly classified windows.

TABLE I. NUMBER OF CORRECTLY CLASSIFIED WINDOWS

Window Size (pixels) (total test windows)	**GLCM-256**		**GLCM-16**		**H-32+DH-16**		**LBP(r=1;P=8)**		**LBP(r=1;P=8)**		**LBP(r=1;P=8)**	
	ANN	***SVM***	***ANN***	***SVM***	***ANN***	***SVM***	***ANN***	***SVM***	***ANN***	***SVM***	***ANN***	***SVM***
240×240 (**26**)	**26**	17	**26**	18	**26**	19	**26**	**26**	**26**	**26**	**26**	**26**
120×120 (**104**)	102	73	102	94	101	74	**104**	99	**104**	99	**104**	99
60×60 (**416**)	393	367	**408**	378	402	253	401	399	396	400	375	400
30×30 (**1664**)	1558	1480	**1575**	1536	1552	1556	1459	1483	1503	1519	1431	1526
TOTAL (2444)	2079	1937	**2111**	2026	2081	1902	1990	2006	2029	2044	1936	2051

IV. PROPOSED MODIFICATIONS

Despite the growing popularity of the LBP, the previous test shows that on smaller windows (patches), performance of GLCM and "histogram plus difference histograms" are superior. In his article, Unser proves that his sum and difference histograms (SDH) are mathematically equivalent of the GLCM. The reason why he proposes them is because SDH requires less memory and processing power. Since one of the main objectives of this study is to come up with a texture discrimination (defect detection) algorithm that can fit inside a camera, it is natural that we chose to continue with Unser's sum and difference algorithms. Unser also states that %99 of the texture information comes from the difference histogram rather than the sum histograms.

Another reason why difference histograms are used is that they are immune against illumination changes. Despite the fact that illumination conditions are under control in factory settings, we would still want our algorithm to be as robust as possible to illumination changes. This can be explained using the mathematical model of the light absorption. Let's assume that a light source is sending photons to a textured surface at an illumination strength of 200. Let the first area corresponding to the pixel-1 has a reflectivity of %90 and the second area corresponding to the pixel-2 has reflectivity of %50. Then the camera would detect the photons coming from these two areas at a strength of 180 and 100 respectively. The sum of these pixels would be 280, and their difference would be 80. If the illumination is increased by %20, then these pixels would be registered as gray levels of 240×0.9=216 and 240×0.5=120. Their sum would be 346 and their difference would be 96. The fluctuation for the difference histogram would be much lower because the increase in the intensities would cancel each other out. If we use 16 gray levels instead of 256, then the difference values for those two different illumination conditions would be 5 and 6. This is only one level shift for %20 change in the illumination, which means that for lower illumination changes or for pixels with less contrast difference, it would not be noticeable at all. Nevertheless, applying the histogram equalization would be enough for this algorithm to withstand any reasonable illumination fluctuations. On the other hand, even with the histogram equalization, sum vectors would be very sensitive to any illumination change.

A. Frequency Domain Fine Tuning

There are dozens of articles in the literature that present successful results of both Haralick's and Unser's method on texture classification and discrimination problems. As Haralick and Unser both did, almost every one of those studies set the pixel distance "d" to 1. There is no explanation in Haralick's paper why he used d=1. Using 1 for the pixel distance has become so common that some articles don't even mention what distance they used [20]. The rest of them either use 1 or use 1 along with some other pixel distance, such as 5s. For example, a fabric defect detection experiment done by Latif-Amet [21] also set pixel distance to 1. A defect detection study on aluminium surfaces conducted by Chondronasios [22] selects d=1 and Georgieva [23] uses two different pixel distances, d=1 and d=5 to classify the cork tiles.

Because the texture pixels are defined as "a region slowly varying, or approximately periodic" [6], we will build our mathematical model based on this. If we scan a texture through a straight line at any given angle, the one dimensional signal obtained should be:

$$f(x) = a_0 + \sum_{n=1}^{N} An\ cos\left(2\pi\frac{n}{k} + \theta_n\right) + \varphi(x) \qquad (1)$$

Where $\varphi(x)$ is the noise. Since we want to calculate histograms of pixels that are "d" pixels apart, and assuming

that the texture has frequencies that are integer multiple of themselves, the values of the two pixels that are "d" pixels apart would be:

$$f(p_{x1}) = a_0 + \sum_{n=1}^{N} An\ cos\left(2\pi \frac{nx1}{k} + \theta_n\right) + \varphi(x1) \quad (2)$$

$$f(p_{x2}) = a_0 + \sum_{n=1}^{N} An\ cos\left(2\pi \frac{n(x1+d)}{k} + \theta_n\right) + \varphi(x1 + d) \quad (3)$$

If our eyes can detect obvious periodicity in a texture, such as the ones created by periodic weavings of a fabric, then we can assume that most of the cosine terms have frequencies that are integer multiple of the main repetition. This means that, when the pixel distance "d" is equal to exactly one period of the texture, the difference of these two pixels will cancel out all of the cosine terms:

$$f(p_{x1}) - f(p_{x2}) = \varphi(x_1) - \varphi(x_1 + d) \quad (4)$$

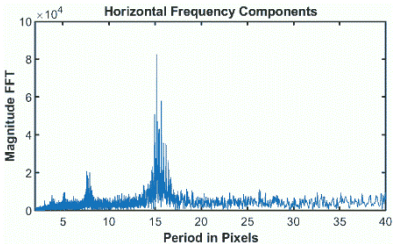
Naturally, we assume that the biases cancel each other out as well and all that remains are the two random variables. If the pixel distance does not equal to one period of the sinusoid, there will always be cosine terms in (4), meaning that the signal's variance would be much higher. High variance is not a desired property for a feature. This is the main reason why we use difference histograms (DH).

As shown in Fig. 3, if we scan through the texture image in four different orientations and calculate the magnitude of the Fourier Transform (FT) of that signal, the highest peaks in the magnitude of the FT will tell us which direction and distance must be used for the calculation of difference histogram.

We can use another method to verify our approach. If we have large number of defective samples (or areas), we can perform Fisher's Discriminant Analysis (FDA). The difference histogram vectors must be calculated on non-defective and defective windows for a range of "d" (let's say from 1 to 25) and for all of the orientations. Then the FDA of each element must be calculated and summed. The pixel distance and the orientation with the highest FDA sum should give us the optimum pixel distance "d". Fisher's Discriminant Ratio (FDR) is defined as:

$$FDR = \frac{(m_1 - m_2)^2}{\sigma_1^2 + \sigma_2^2} \quad (5)$$

In this equation, m1 and m2 are the respective mean values, and σ_1 and σ_2 are the respective deviations associated with the values of a feature in defective and non-defective classes. Collectively these modifications will be referred to as "Diftogram." The following sections will show how this is done through a few examples.



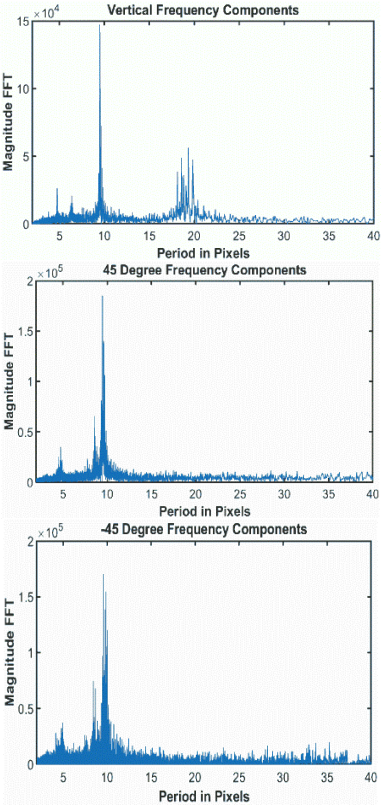


Fig. 3. Magnitude FFTs of the signals obtained by scanning the fabric image horizontally, vertically, at +45 and -45 degree angle.

V. DATASET-I

A typical fabric sample is used for the first experiment. Fabric defect detection is one of the areas of industrial machine vision where texture discrimination is widely used. In this experiment, 12 images with natural fabric defects such as broken thread, missing thread, stuck thread, weaving error, punctures, and lubricant oil stains are used. In order to compare the effects of modifications, the co-occurrence matrices are used as in the conventional way. In order to increase robustness against illumination changes, histogram equalization is applied to the area of interest. This can be noticed by the slight increase of contrast around the perimeter of the white boxes in all of the fabric images. GLCM are calculated for each of the 80×80 pixel windows on 1024×720 pixel images with pixel distance set as d=1. Co-occurrence matrices are calculated for orientations of 0, +45, +90 and -45 degrees using reduced gray levels of 16. Alongside the 14 features proposed by Haralick, the two features proposed by Clausi [24] and the three features proposed by Soh [25] are also calculated. This is necessary to prevent possible criticism that GLCM method is incomplete without the newly added features and that the results could have been better. In Section 3, the ANN performed better than the SVM. Therefore, a feed forward ANN with 25 neurons in hidden layer is used for all of the experiments.

Total of 19 features, 14 from Haralick, 2 from Clausi and 3 from Soh are calculated from every matrix. Since there is one matrix per each of the four directions, a total of 76 features are calculated to be used as input to the neural network. The training and the test sets are the same. The Neural Network

toolbox of the Matlab software was used to train the ANN. The ANN training process is not a deterministic one. Every time a neural net is trained, even with the same training set and the parameters, results will be slightly different. Because of that, the experiment was conducted more than 30 times and the most successful ANN among them was selected. Results of this experiment are shown in Table II and Fig. 4. The white windows represent 80×80 pixel squares that are defined as "normal" by the ground truth and are also correctly classified as non-defective by the algorithm. The green boxes are drawn if both the ground truth and the algorithm classifies them as defective. In other words, correctly classified areas are marked as either white or green boxes. The black color window represents that the window was marked as defective but the algorithm failed to detect it. If the window is red, this means that a non-defective area was incorrectly classified by the algorithm as defective. Table II reveals that the GLCM algorithm using d=1 and using four orientations was able to detect 14 defects out of 42, and misclassified 1 or 2 areas as defective.

In order to see the effect of using Unser's difference matrices as direct input only, without applying the method to find the most effective pixel distance and orientation, the experiment was repeated using Unser's difference histograms with d=1 and in the four orientations. It was observed that the ANN had a very low margin to distinguish defective and normal areas. Because of this, two different outcomes were observed. In some of the trials, ANN recognized 37 of the 42 defective regions with 27 misclassification of normal areas whereas in the other trials it recognized 28 defective areas with 19 misclassification of non-defective areas. Both of these results are given in Table II.

To measure the success rate of the proposed method against other algorithms, the most effective pixel distance 'd' needs to be calculated. In order to achieve that, a fabric image without any defects must be scanned through four directions to obtain one dimensional signals. Afterwards, the magnitude of the Fourier transform of these signals should be calculated as shown in Fig. 3. Each signal has one dominant peak. The pixel period corresponding to those peaks should be used as the pixel distance.

To verify, or perhaps to get a second opinion, Fisher's Discriminant Analysis should also be used. As the first step, we calculate all of the DH for each of the 80×80 pixel windows for all of the twelve fabric images, using all possible pixel distances between one and twenty five. Afterwards we calculate the FDR value for each element in the difference vector using (5), and then sum these FDR values. The pixel distance and orientation with the highest FDR sum value is the one that should be used for texture discrimination. Below are the FDR sums for the elements of the vectors obtained for 25 different distances and four orientations. For 0-degree orientation:

1 2 3 4 5 6 7 8
FDR(Θ = 0)=[0.39 0.26 0.76 1.12 2.20 3.02 3.94 4.23
9 10 11 12 13 14 **15** **16** 17 18
19
3.99 3.93 3.75 4.07 7.22 13.81 **21.88** **25.34** 17.65 10.89
6.66
20 21 22 23 24 25
4.48 3.93 4.48 5.23 5.63 5.33]

The largest FDR sum values are for d=15 and d=16 for Θ equals 0 degrees (horizontal direction). When we look at the Fourier transform of the horizontal signals in Fig. 3, we see that the peak magnitude is exactly at d=16. Proceeding with the vertical direction,

1 2 3 **4** **5** 6 7 8
FDR (Θ = 90) = [2.76 4.65 5.43 **6.02** **5.51** 4.58 3.44 3.00
9 10 11 12 13 14 **15** **16** 17 18 19 20
21
3.21 3.51 3.53 2.47 2.70 3.75 **5.02** **5.66** 4.54 2.62 2.87 3.68
1.74 1.15 2.49 3.03 2.82]

FDR sum values reach their maximum at d=(4,5) and d=(15,16). As seen in the Fourier analysis of the vertical signal, there is a small peak at d=5 but the most important peak at d=10 has no effect on the FDR values and instead of the flat peak around d=19 we the values at d=15 and 16 of the FDR are higher. It is possible that the very narrow peak at d=10 is not detected and the defects have their own characteristics that is altering the values of the FDR. Because there are more than one significant cosine components in the vertical signal, the FDR values in the vertical direction are far smaller than that of the horizontal. This increases the variance of the signal and as a result, reduces the FDR values. For +45 orientation we get d=9, and for -45 degree orientation we get d=10.

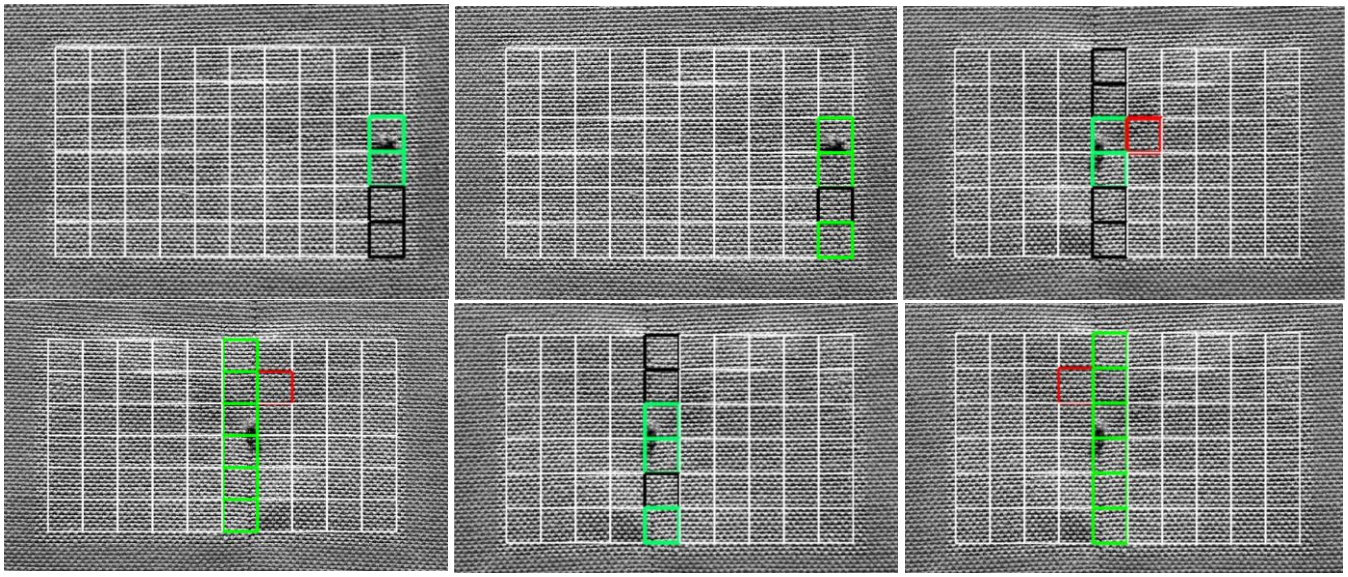
1 2 3 4 5 6 7 8
FDR (Θ = + 45)=[2.70 4.28 5.48 6.00 5.70 5.52 5.47 5.78
9 10 11 12 13…
10.49 6.78 1.26 1.92 3.31 4.25 5.03 5.43 3.92 3.61 6.10
2.38 0.76 1.36 2.81 3.55 3.56]
1 2 3 4 5 6 7 8
FDR (Θ = - 45)=[1.57 3.01 4.39 5.80 7.10 8.08 7.38 5.70
9 **10** 11 12 13…
5.62 **7.98** 3.24 0.61 1.59 3.88 5.32 5.74 4.69 2.67 2.39 4.05
3.29 1.47 1.68 3.45 4.12]

The FDR values for 0 degree orientation clearly indicate that difference histogram should be applied in the horizontal direction and the pixel distances should be set as d=15 and d=16. Using these parameters Diftogram's success rate is shown in Table II. The algorithm classified 37 defective areas out of 42 correctly and misclassified 6 areas as defective. That is almost 2.5 times (37/14=%264) higher defect recognition rate than the GLCM experiment.

As shown in Table II and Fig. 4, the results are considerably superior against the LBP. Some readers might be confused that the training and test sets are the same. The purpose of this study is not to measure the actual success rate of a single algorithm but to measure relative success rate of several algorithms with respect to each other. As long as the training conditions are the same, these results will give us which algorithm performs relatively better. This will be proven in Section VII (Dataset III) where we perform training on different training and test sets and the relative success rates remain similar.

TABLE II. DISCRİMİNATİON RESULTS FOR DATASET-I

Algorithm	Defects	False +	Accuracy	Precision
	Max 42	*Max 678*		
GLCM-256 d=1; 4 orientations	14	2	0.9583	0.8750
GLCM-256 d=1; 4 orientations	15	1	0.9611	0.9375
LBP (r=1,p=8) (r=2, p=8)	20	1	0.9681	**0.9524**
LBPri	25	2	0.9736	0.9259
DH d=1; 4 orientations	**37**	27	0.9556	0.5781
DH d=1; 4 orientations	28	19	0.9542	0.5957
DH d=**15,16** Horizontal	**37**	6	**0.9847**	0.8605



GLCM-256 with 4 orientations d=1. LBPri DH, horizontal, d=15 &16

Fig. 4. Two of the defective fabric samples: Broken thread (top) and stuck thread (bottom). White boxes are correctly classified normal areas and green ones are correctly discriminated defective ones. Black ones indicate the regions that the algorithm failed to discriminate and red ones are incorrectly classified defects.

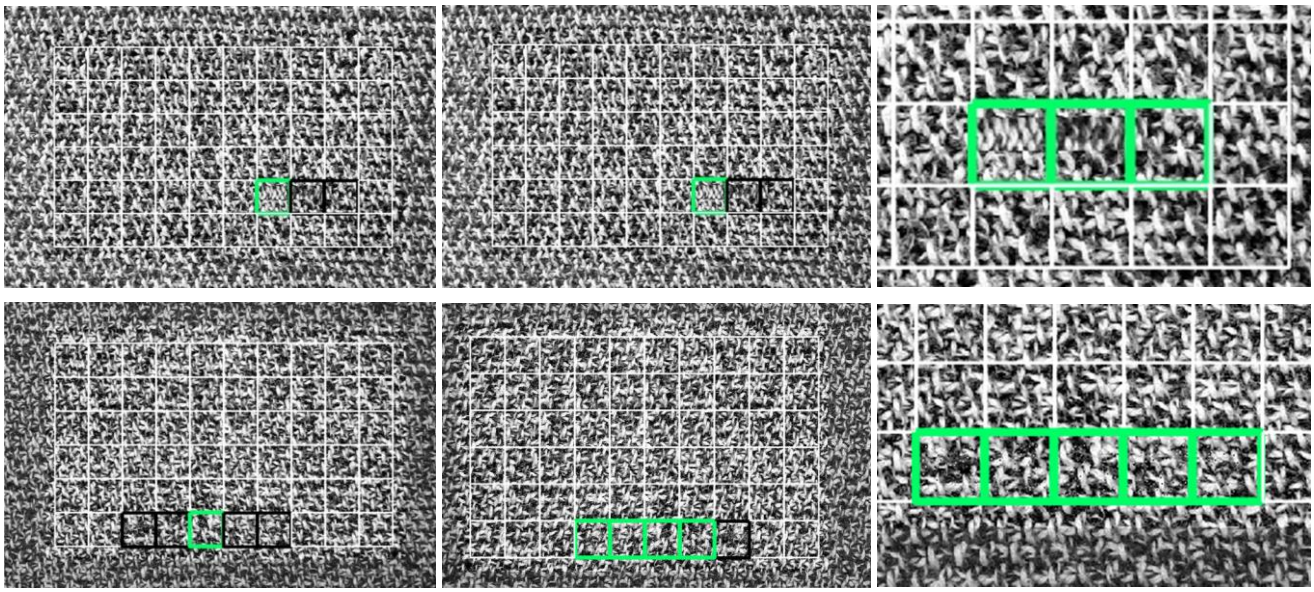
VI. DATASET-II

A slightly more challenging fabric sample is selected for this experiment. The thread used for weaving this fabric has an alternating colour making the noise level of the texture extremely high. The lighting conditions were also altered. For this experiment, multiple light sources were used. The first three fabric samples in the set were taken under %20 higher illumination then the rest to demonstrate illumination change's effect. Another challenging part of this experiment is that the defects were artificially created to make them extra difficult to detect. The defects are so mild that even the human eye can barely notice them. In order to do so, 15 different images of the fabric were taken and then using the GNU Image Manipulation Software, GIMP (https://www.gimp.org), different kind of defects were created using blurring, Gaussian and salt and pepper noise, contrast altering, spatial compression, pixel smearing and etc.

Table III and Fig. 5 shows the results of the GLCM algorithm using 256 and 16 grey levels, four orientations and 19 features calculated from each orientation. Pixel distance was set to d=1, similar to almost every GLCM experiment published in the literature. The classifier is the same ANN used for the previous experiments, a feed forward network with 25 neurons in hidden layers. The training set and the test set are the same, the training was repeated more than a dozen times and the most successful one is shown here. This combination of algorithms classified 20 defects correctly out of 64. Zero or two normal areas were misclassified as defective.

TABLE III. DİSCRİMİNATİON RESULTS FOR DATASET-II

Algorithm	Defects	False +	Accuracy	Precision
	Max 64	*Max 836*		
GLCM-256 d=1; 4 orientations	20	0	0.9511	**1.00**
GLCM-16 d=1; 4 orientations	22	2	0.9511	0.9167
LBP (r=1,p=8) (r=2, p=8)	28	0	0.96	**1.00**
LBPri	28	1	0.9589	0.9655
DH d=1; 0 & 90 degrees	**59**	1	**0.9933**	0.9833



GLCM-256 with 4 orientations d=1. LBPri DH d=1, 0 & 90 degrees

Fig. 5. Samples from dataset II. Above: A different kind of weaving was cut and paste on it as defect. Below: Added salt and pepper noise. Green boxes are correctly discriminated defective areas and black boxes are the ones that the algorithm failed. Images on the right shows zoomed defects.

Before proceeding to the second part of the experiment, the Fourier analysis of the fabric image must be performed. The absolute value of the FFT obtained by combining many lines from several images shows that there isn't any salient sinusoid in this signal. The FFT results are similar in other directions therefore they are not shown. Under such circumstances, the highest peak is usually obtained at zero frequency. This means that the FDR analysis should give us d=1. When we perform the FDR analysis that is exactly what we get:

FDR(Θ = 0)=[**5.97** 2.37 1.41 1.39 1.68 2.00 2.17 2.19 2.18 2.05 1.97 1.99 2.03 1.91 1.71 1.64 1.50 1.34 1.25 1.26 1.32 1.29 1.24 1.22 1.12]

FDR(Θ = 90)=[**5.50** 1.89 0.57 0.52 0.77 1.25 1.58 1.94 2.18 2.33 2.43 2.47 2.29 2.01 1.80 1.81 2.03 2.24 2.63 2.97 3.42 3.58 3.67 3.58 3.43]

Similar results are obtained for 45 and -45-degree orientations. It seems that selection of pixel distance as 1 is not a bad choice if the texture has a lot of noise or there are more than a few high amplitude periodic components in the texture.

For the next part of the experiment, Unser's difference histograms are directly used as the texture features. The grey levels are reduced to 16, pixel distance is set to one, and only two orientations, horizontal and vertical are selected. In the first part of the experiment where GLCM was used, same parameters were used except for the fact that, 45 and -45-degree orientations were also used with the GLCM. These two directions, 45 and -45 are skipped to demonstrate the power of the proposed method and also to make the algorithm twice as fast. The elements of the two difference vectors are directly fed to the same ANN used for the previous experiments. As shown in Table III, this algorithm recognized 59 defective regions out of 64 and misclassified one normal area as defective. Compared to the defect detection rate of the GLCM algorithm, this is almost %300 higher. Against the LBP variants, it is almost %200 more successful.

VII. DATASET-III

For the third experiment, a fabric sample with a patterned weaving is used. The fabric is weaved to display a repeating pattern, which creates a secondary pattern on top of the pattern created by the threads of the fabric. Lighting conditions were slightly altered for many of the 45 image samples of this fabric. The defects were created using the GIMP software, but this time with increased variety and larger surface area. These defects vary in strength; about half of them are easily noticeable by the human eye. The frequency analysis in Fig 6 shows that this fabric has a major component at a period of around 27 pixels in the horizontal direction. At 45 degrees, the major component is at 13 pixels (not shown) and for the vertical direction, there are three peaks at 9, 14 and 25 pixels. The FDR analysis show consistent results with these findings:

FDR(Θ = 0)=[3.63 2.11 1.60 0.56 2.12 0.46 1.55 0.76 0.54 1.03 0.79 1.35 2.07 1.79 1.66 1.13 1.09 0.43 0.6 0.53 0.30 1.24 1.29 0.86 1.12 2.18 **6.10** 4.92 1.89 0.95]

FDR(Θ = 45)=[3.75 2.57 2.60 0.75 0.51 1.51 2.66 0.75 0.47 0.55 1.32 1.97 **6.41** 4.25 1.21 1.55 1.17 0.74 0.69 0.90 0.56 0.3814 0.73 1.03 1.66 2.30 2.06 0.95 0.50 0.51]

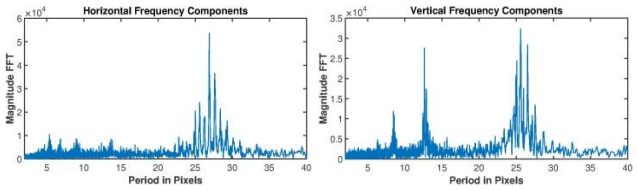
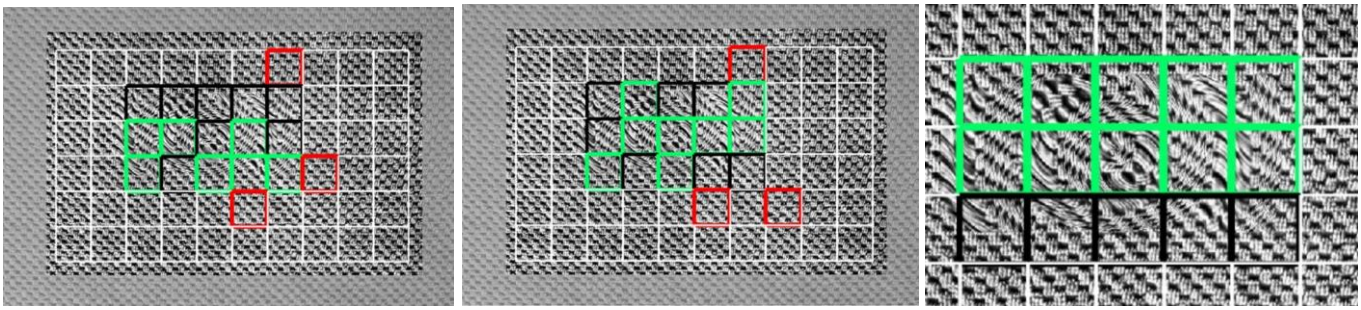


Fig. 6. Magnitude of the FFT of the signal obtained by scanning the fabric of dataset III, horizontally and vertically.

FDR(Θ = 90)=[**6.16** 2.65 1.21 0.86 0.82 1.08 1.18 1.57 1.85 1.56 1.40 1.19 1.07 1.54 2.01 1.91 1.68 1.18 0.87 0.72 0.64 1.15 2.11 3.14 3.87 2.83 1.52 0.78 0.76 1.17]

Because of the abundance of distinct frequency components in the vertical direction, (Θ = 90), the FDR analysis show that there is no advantageous pixel distance in the vertical direction other than d=1. For the 0-degree orientation, d=27 and for the 45-degree orientation d=13. Fig. 7 and Table IV shows the results of discrimination tests for the GLCM with d=1 in four directions, Diftogram with d=27 for the horizontal and d=one for the vertical orientations and the LBP. Diftogram is approximately %250 more successful than the GLCMs and %200 more successful than the LBPs. Notice also that the number of false positives are smaller than both the GLCMs and the LBPs. This result is slightly lower than the previous examples. The reason for that is because the fabric has two different patterns on it and the defects are stronger. Much higher results could be obtained by adding more orientations and pixel distances to the Diftogram if computational efficiency is not essential.

Because we are testing the reliability of different kind of features against each other, there is nothing wrong about using the same images for both training the ANN and testing it as long as the ANN is trained the same amount for each feature set and the ANN is not trained excessively to the point where it memorizes the samples. In other words, if the ANN has enough generalization capability, it will measure how well the features separate the defects from the background. The default parameters of the ANN trainer in Matlab has very good balance between memorizing and generalizing therefore, the experiments conducted so far has measured the reliability of the features quite well. Unfortunately, there might be people who do not trust such findings and demand that the training and the test sets should be different. For those people the third experiment is conducted again, this time using the odd numbered 23 images (1, 3, 5, ... 45) for training and the other 22 for testing. Table V shows that the Diftogram is still about two times more successful than both the GLCMs and the LBPs.



GLCM-256 with 4 orient. d=1. LBPri DH d=27, horizontal & d=1 vertical

Fig. 7. A sample from dataset III, with optical distortions on it. Black boxes indicate the regions that the algorithm failed to discriminate. White boxes are correctly classified normal areas and green ones are correctly discriminated defective ones. Red boxes are false positives.

TABLE IV. DİSCRİMİNATİON RESULTS FOR DATASET-III

Algorithm	Defects	False +	Accuracy	Precision
	Max 361	*Max 2339*		
GLCM-256 d=1; 4 orientations	124	16	0.9063	0.8857
GLCM-16 d=1; 4 orientations	118	17	0.9037	0.8741
LBP (r=1,p=8) (r=2, p=8)	159	33	0.9130	0.8281
LBPri	150	29	0.9111	0.8380
DH d=27; 0 deg. d=1; 90 degree	**292**	**6**	**0.9722**	**0.9799**

TABLE V. DİSCRİMİNATİON RESULTS FOR DATASET-III (SEPARATE TRAİNİNG SET)

Algorithm	Defects	False +	Accuracy	Precision
	Max 361	*Max 2339*		
GLCM-256 d=1; 4 orientations	67	21	0.8871	0.7614
GLCM-16 d=1; 4 orientations	68	20	0.8886	0.7727
LBP (r=1,p=8) (r=2, p=8)	66	22	0.8856	0.7500
LBPri	69	25	0.8856	0.7340
DH d=27; 0 deg. d=1; 90 degree	**141**	**11**	**0.9508**	**0.9276**

VIII. Complexity Measurements

In order to measure the computational complexity of each method, a C++ program was written. Measuring the processing time per one image is not possible on today's fast computers. To get an accurate measurement of how long each method takes to compute, 10 images were processed 500 times and then the total duration was divided to 5000. The computer was programmed to run a loop 5000 times, processing one of the 10 images in each iteration. This was done to prevent the CPU from copying an entire image in its cache. Processing the same image from the cache repeatedly would yield wrong results. Using The Microsoft Visual Studio compiler, with full optimization, measurements show that the calculation of the GLCM in one direction using 16 grey levels takes approximately the same amount of time with calculating the difference histogram in one direction with the same grey levels. Calculation of the difference histogram takes %98.5 of that of the GLCM. A few of the features that Haralick proposed were also implemented. The time required for their completion was negligible compared to the time required for the computation of a 16×16 GLCM. Over all, it seems that it takes %3-5 percent longer for the GLCM and its features to be calculated. However, for 256 grey levels, the GLCM becomes 256×256, making the calculation time for the features far longer. In Matlab, overall time for the calculation of 256 grey level GLCM takes 32 times longer compared to 16 grey level GLCM.

In order to compute the speed of LBP, a pure Matlab code is used. LBP (R=1, 2 p=8) took %43 longer to calculate than the GLCM-16 (Matlab code) in one direction. Therefore, it is computationally more efficient than the GLCM because it is more successful than the GLCM calculated in four directions. LBPri is % 419 slower than the GLCM-16 in one direction therefore it is only slightly more efficient than the GLCM. However, it should be noted that the LBP is cheating by using a gigantic; 16 MB look up table to speed up its calculations. It is essentially creating a processor inside the memory. Based on these numbers, Diftogram is computationally more efficient than both of the LBP algorithms are. According to a recent comparative study [26], among the LBP algorithms, LBPri is the fastest among all of them and it is only %15 less successful than the most successful variants such as the MRELBP [27]. Considering that the most successful ones are at least five times slower than the LBPri, (MRELBP is 9 times slower) it is seems that Diftogram is still computationally more efficient, even if it is not as reliable as the newest versions of the LBP.

In both of the previous experiments, GLCM was computed in four directions. On the other hand, based on our proposed method, the difference matrix (DM) was calculated in only two directions. Therefore, the proposed method is not only more successful at recognizing the defects, but also twice faster against GLCM-16.

IX. Results and Conclusion

In this article, two new approaches were proposed. The first one suggests that the use of Unser's difference histograms as input to a classifier yields better results at discriminating the texture than Haralick's method. Usually texture discrimination is a far more challenging problem then the texture classification; therefore, by using this method, similar superior results should be expected for classification as well. The second approach is that the orientation and the pixel distance to calculate the second order histograms must be selected based on the frequency properties of the texture. Two methods are proposed for the selection of these parameters. Another suggestion is the use of pixel average as an extra feature for more challenging patterns. Based on this design, three experiments are conducted.

Many of the defects used in the experiments are so faint that the human eye can barely notice them. In each of the experiments, the proposed set of methods performed approximately %250-%300 better at discriminating the texture, compared to the GLCM method with 19 features. The C++ implementation of the proposed algorithm, based on the parameters used for these experiments, should be approximately twice faster than the GLCM-16, which means, at least %500 improvement in efficiency. Against the LBP variants, the Diftogram is almost %200 more successful. The computational efficiency of the rotation variant plain LBP is comparable to that of the Diftogram while both the success rate and the efficiency of the LBPri is less.

Majority of the recent articles study rotation invariant texture recognition and discrimination algorithms. Therefore, the improvements discussed here might seem not very important for many academicians. However, engineering is performed to solve the problems of humanity, not the ones that we create in our own labs. Considering the fact that, every second, quality inspection is being performed on thousands of factories across the planet, hopefully, the improvements proposed in this article will find much greater use than many others published in the literature.

Acknowledgment

This study was financially supported by Adana Alparslan Türkeş Science and Technology University Scientific Research Coordination Unit. Project number: 16103010.

References

[1] R. M. Haralick, K. Shanmugam, I. Dinstein, "Textural Features for Image Classification," IEEE Trans. Syst., Man, Cybern., vol. SMC-3, No 6, pp. 610-621, 1973.

[2] M. Unser, "Sum and Difference Histograms for Texture Classification," in IEEE Trans. Pattern Anal. Machine Intell. Vol. 8, No.1 1986, pp. 118-125.

[3] T. Ojala, M. Piatikainen, T. Maenpaa, "Multiresolution grey-scale and rotation invariant texture classification with local binary patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence vol.24, no:7, 2002 pp 971–987.

[4] Abuhussein, Mohammed, and Aaron Robinson. "Obscurant Segmentation in Long Wave Infrared Images Using GLCM Textures." Journal of Imaging 8.10 (2022): 266.

[5] Teo, Hong Chun, et al. "A review of the automated timber defect identification approach." International Journal of Electrical and Computer Engineering 13.2 (2023): 2156.

[6] Meeradevi, T., et al. "An analytical survey of textile fabric defect and shade variation detection system using image processing." Multimedia Tools and Applications (2022): 1-30.

[7] Tang, Bo, et al. "Review of surface defect detection of steel products based on machine vision." IET Image Processing (2022).

[8] I. C. Baykal, G. A. Jullien “In-Camera Detection Of Fabric Defects” IEEE International Symposium on Circuits and Systems (ISCAS) 2004.

[9] Wagner T. (1999) Chapter 12 Texture Analysis. In: B. Jahne, H. Hauβecker, P. Geiβler(ed), Handbook of Computer Vision and Applications, vol 2, Academic Press, pp 276-307.

[10] Schramm, U., (1994). Automatische Oberflachenprufung mit neuronalen Netzen. Stuttgart: IRB-Verlag.

[11] Chen, Y., Nixon, M., and Thomas, D., “Statistical geometrical features for texture classification.” Pattern Recognition, 28(4): 1995, pp.537-552.

[12] Fogel, I. and Sagi, D., “Gabor Filters as texture discriminator,” Biological Cybernetics, no: 61, 1989, pp103-113.

[13] Laws, K. I., Textured Image Segmentation. PhD thesis, Faculty of the Graduate School, University of Southern California. 1980.

[14] Galloway, M. M., Texture analysis using gray level run lengths, Computer Graphics and Image Processing, no: 4, 1975, pp. 172-179.

[15] Maria Petrou, Alireza Talebpour, Alexander Kadyrov. “Reverse engineering the way humans rank textures” Pattern Anal. Applic. Vol. 10, No 2, 2007, pp. 101–114.

[16] E. Gonzalez-Rufino, P.Carrion, E.Cernadas, M.Fernandez-Delgado, R.Dominguez-Petit, “Exhaustive comparison of colour texture features and classification methods to discriminate cells categories in histological images of fishovary,” Pattern Recogniton, no 46, 2013, pp: 2391-2407.

[17] L.H. Siew, R.M. Hodgson, E.J. Wood, “Texture measures for carpet wear assessment,” IEEE Transactions on Pattern Analysis and Machine Intelligence 10 (1) 1988, pp: 92–104.

[18] Sklansky, J, “Image segmentation and feature extraction,” IEEE transactions on Systems, Man, and Cybernetics, SMC-8, 1978, pp 237-247.

[19] I. C. Baykal, “Performance Comparison of Texture Classifiers on Small Windows” IEEE Internatioanal Atificial Intelligence and Data Processing Symposium (IDAP) 2019.

[20] M. A. Selver, V. Avşar, H. Ozdemir, “Textural fabric defect detection using statistical texture transformations and gradient search,” in The Journal of The Textile Institute, 2014, 105:9, pp. 998-1007.

[21] A. Latif-Amet, A. Ertuzun, A. Ercil, “An efficient method for texture defect detection: sub-band domain co-occurrence matrices,” Image and Vision Computing, Elsevier, no:18 2000, pp: 543-553.

[22] A. Chondronasios, I. Popov, I. Jordanov, “Feature selection for surface defect classification of extruded aluminum profiles,” in Int. J. Adv. Manuf. Technol. No 83, 2016, pp 33-41.

[23] A. Georgieva, I. Jordanov. “Intelligent Visual Recognition and Classification of Cork Tiles with Neural Networks,” IEEE Transactions On Neural Networks, Vol. 20, No. 4, 2009, pp. 675-685.

[24] D A. Clausi, “An analysis of co-occurrence texture statistics as a function of grey level quantization,” Can. J. Remote Sensing, vol. 28, no. 1, pp. 45-62, 2002.

[25] L. Soh and C. Tsatsoulis, “Texture Analysis of SAR Sea Ice Imagery Using Gray Level Co-Occurrence Matrices,” IEEE Transactions on Geoscience and Remote Sensing, vol. 37, no. 2, 1999.

[26] L. Liu, P. Fieguth, Y. Guo, X. Wang, M. Pietikainen, “Local binary features for texture classification: Taxonomy and experimental study,” Pattern Recognition, no:62 pp 135-160, 2017.

[27] L. Liu,S.Lao,P.Fieguth,Y.Guo,X.Wang,M.Pietikainen, “Median robust extended local binary pattern for texture classification,” IEEETrans. Image Process. 25(3) pp:1368–1381, 2016.