# Square Binary Patterns: A Non-Circular Approach for Local Binary Patterns

Muhammad Ardi Putra, *Member*, *IAENG*, Agus Harjoko*, *Member*, *IAENG*, Wahyono, *Member*, *IAENG*, and Kanghyun Jo

***Abstract*— Local Binary Patterns (LBP) is one of many image features suitable for enhancing textures within an image. Despite its popularity, this feature as well as its variants actually still has some drawbacks: requiring the use of bilinear interpolation mechanism which leads to a computational bottleneck, being unable to capture macrostructures, and being prone to noise. The objective of this research is to propose a novel image feature named Square Binary Patterns (SBP) which is intended to solve the mentioned problems. Experiments were conducted using four texture datasets, namely KTH-TIPS, KTH-TIPS2-b, ALOT, and the texture dataset we created on our own. The existing LBP-based features including $LBP^{riu2}$, $LBP^{u2}$, $LBP^{ri}$, MS-LBP, LTP, FbLBP and ILBP are directly compared with SBP. The quality of these features is obtained by observing the classification accuracy of separate SVM models. The classifier which produces higher accuracy implies that its corresponding feature contains a more important information regarding the textures. Experimental results showed that our best-performing SBP variant was able to work 2.98 times faster than the conventional LBP while at the same time capable of producing accuracy comparable to that of the existing LBP-based features. This paper makes a significant contribution through the introduction of the SBP algorithm and the creation of a novel texture dataset.**

***Index Terms*—Feature Extraction, Local Binary Patterns, Machine Learning, Texture Classification**

## I. Introduction

TEXTURE feature extraction is one of the most interesting topics to be brought up into discussion as it is useful to be used in many classification tasks, especially when the classes of the images are distinguishable by looking at its textures, such as visual-based material analysis. There are actually several algorithms specialized for extracting texture features, one of which is LBP (Local Binary Patterns). The algorithm, which was first introduced by Ojala et al. [1], became popular in this field thanks to its ability in handling gray-level variations [2]-[4]. This notion is true because the binary encoding of each pixel within an image is generated only based on its local neighborhood, thus not affected by the illumination in the farther part of the image.

Despite its high popularity, this does not necessarily mean that LBP is flawless. Even though this feature extraction algorithm is considered to be computationally cheap [3], yet the authors of this research think that the process can still be further sped up. This is essentially because the sampling points of LBP follow a circular pattern, which causes plenty of points to not fall right at the center of a pixel. In such a case, bilinear interpolation operation is required to approximate the actual value of these sampling points, which authors perceive this process as a bottleneck. To the best of our knowledge, this phenomenon can be seen in all existing LBP variants, including those proposed by the same author in their subsequent paper, i.e., $LBP^{u2}$ (Uniform LBP), $LBP^{ri}$ (Rotation-Invariant LBP), and $LBP^{riu2}$ (Rotation-Invariant Uniform LBP) [5]. The second problem encountered by LBP is that according to [6] the feature extraction method is unable to capture macrostructures, i.e., large repeating patterns. This statement makes sense because LBP works by taking into account local neighborhood, limiting its ability to capture microstructures only. Third, LBP is also known to be prone to noise [2], [7], [8] since an extremely high or low value is captured as is by the algorithm which affects the resulting binary encoding.

In order to address the above issues, the authors of this research propose a new texture feature named SBP (Square Binary Patterns), in which it is intended to tackle the three problems simultaneously: the need for bilinear interpolation, the inability to capture macrostructures, and the difficulty in handling noise. The first one can be achieved since SBP, as the name suggests, uses square-shaped rather than circular pattern which guarantees the sampling points lie exactly at the center of a pixel. To address the macrostructure problem, SBP implements an adjustable parameter to control the sampling point radius to be taken into account. This method is actually inspired by MS-LBP (Multi-Scale LBP) which was previously proposed in [9]. Lastly, the noise problem is addressed by taking the average features from different radii, in which this process is adopted from the average smoothing algorithm.

The two main contributions of this paper are listed below:

- The proposal of SBP for addressing the problems of LBP as well as its existing variants mentioned earlier.
- The creation of a new texture dataset which will later be used to further validate the performance of SBP.

The remaining parts of this paper will be organized as follows: the second section explains previous works related to the existing LBP variants, the third section explains the details of SBP

Manuscript received November 15, 2024; revised March 25, 2025. This research is supported by Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi of Indonesia through PKPI-PMDSU Scholarship Program under Grant 2069/UN1/DITLIT/PT.01.03/2024. (Corresponding author: Agus Harjoko.)

Muhammad Ardi Putra is a doctoral student of Computer Science at the Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia (e-mail: muhammadardi2017@mail.ugm.ac.id).

Agus Harjoko is a professor at the Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia (e-mail: aharjoko@ugm.ac.id).

Wahyono is an associate professor at the Department of Computer Science and Electronics, Universitas Gadjah Mada, Yogyakarta, Indonesia (e-mail: wahyo@ugm.ac.id).

Kanghyun Jo is a professor at the Department of Electrical, Electronic, and Computer Engineering, University of Ulsan, Ulsan, South Korea (e-mail: acejo@ulsan.ac.kr).

algorithm and the experimental procedure, the fourth one discusses experimental results, and the fifth section concludes the research.

## II. Review on the Existing LBP Variants

### A. The Conventional LBP

Before getting into the details of the proposed SBP, it is necessary to discuss the conventional LBP feature extraction algorithm in advance. The original LBP has two main parameters, namely $P$ and $R$ in which they represent the number of sampling points and the sampling radius, respectively. Fig. 1 displays what the sampling points look like when $P$ and $R$ are assigned with different values. It is important to note that such a sampling mechanism is used in all LBP variants.

The circular pattern shown in Fig. 1 can be modeled using equation (1), where $x$ and $y$ denote spatial coordinates, $p$ represents the index of the sampling point, and $P$ is the total number of sampling points. As seen in the figure, most of the sampling points do not lie exactly at the center of a pixel. Thus, bilinear interpolation technique is used in order to obtain its exact value, which the mathematical expression is displayed in equation (2). The variables $Q_{11}$, $Q_{21}$, $Q_{12}$ and $Q_{22}$ represent the neighboring pixel values where the approximation is made from.

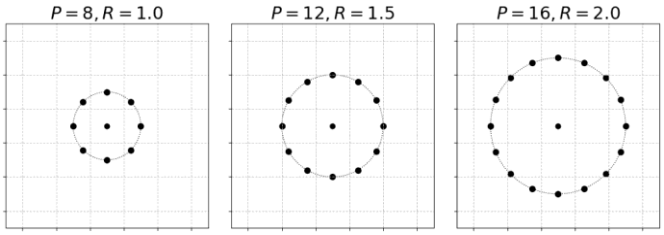


Fig. 1. The sampling points generated using different $P$ and $R$ (redrawing based on [5]).

$$x_p = x - R\,sin(\frac{2\pi p}{P})$$
$$y_p = y + R\,cos(\frac{2\pi p}{P}) \quad (1)$$

$$P = \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} Q_{11} + \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} Q_{21} + \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} Q_{12} + \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} Q_{22} \quad (2)$$

After the sampling points have been interpolated, the next step to do is to generate a binary sequence by comparing the value of a single pixel with the values of its corresponding sampling points. The binary number will then be converted back to decimal prior to being assigned as a new pixel value. Equation (3) shows the encoding procedure of a standard LBP, where $s$ is a sign function, $g_p$ is the value of $p$-th sampling point, while $g_c$ is the value of the center pixel.

$$LBP = \sum_{p=0}^{P-1} s(g_p - g_c)\,2^p \quad (3)$$

$$s(x) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases} \quad (4)$$

### B. LBP-Based Variants

Since its first introduction in 1994, LBP [1] has encountered plenty of modifications. The same author proposed its variants several years later, namely $LBP^{u2}$ (Uniform LBP), $LBP^{ri}$ (Rotation-Invariant LBP), and $LBP^{riu2}$ (Rotation-Invariant Uniform LBP) [5]. The creation of $LBP^{u2}$ was intended to suppress the influence of non-uniform patterns, as these are considered to contain less information than uniform patterns. Meanwhile, $LBP^{ri}$ is an LBP variant specialized to capture a textural structure regardless of its rotation. Combining both $LBP^{u2}$ and $LBP^{ri}$ results in $LBP^{riu2}$, in which it preserves the rotation invariance of a texture while at the same time highlighting only the uniform patterns of a texture.

Many other LBP variants proposed by other researchers started to emerge afterwards. The first one to discuss is the so-called MSJ-LBP (Multi-Scale Joint encoding of Local Binary Patterns) [9], in which it is an improvement of MS-LBP (Multi-Scale Local Binary Patterns). MS-LBP itself is essentially an ordinary LBP which is performed several times on multiple scales. The information captured by MSJ-LBP is claimed to be better than MS-LBP since it also takes into account the correlation of the encodings between all scales rather than the encodings alone. Not only that, there were also several other researchers who attempted to capture multi-scale LBP information. Dan et al. [10] proposed JLBPW (Joint LBP with Weber-like responses) while Prema et al. [11] created HDLBP (Hamming Distance based LBP) to do so. In addition to the latter, the authors implemented the algorithm for smoke detection which is similar to PCLBP (Pairwise Comparing LBP) proposed by Yuan et al. [12] and LBMP proposed by [13].

Next, there are at least two other LBP variants introduced within the same year as MSJ-LBP, namely LBPV (LBP Variance) [14] and CLBP (Completed LBP) [15]. The former works by capturing the contrast information within a small image region, whereas the latter combines information from the center pixel (CLBP_C), the magnitude between center and the sampling points (CLBP_M) as well as its sign (CLBP_S). It might be worth noting that CLBP_S is equivalent to the conventional LBP. FbLBP (Feature-based LBP) [3] actually also captures the similar information to CLBP, in which both FbLBP_F and CLBP_M are intended to capture the magnitude of the sampling points yet with different feature representations.

Not only CLBP and FbLBP, but ELBP (Extended LBP) [16] can also be broken down into smaller components: CI-LBP (Central Intensity LBP), NI-LBP (Neighboring Intensities LBP), RD-LBP (Radial Difference LBP), and AD-LBP (Angular Difference LBP). The idea of this approach is to capture both the information from each individual sampling points itself and the difference between one sampling point and another. In the next couple of years, the same author proposed MRELBP (Median Robust Extended LBP) [6], in which its objective was to improve ELBP in terms of its ability in capturing both micro- and macrostructures. Furthermore, [17] proposes $LBP^{OUS}$ which consists of three different LBPs, namely $LBP^{O}$, $LBP^{U}$, and $LBP^{S}$.

The authors claimed that these three sub-features contain different information which are complementary to each other, allowing $LBP^{OUS}$ to achieve high accuracy.

Different from the previous ones, LTP (Local Ternary Patterns) [7] employs 3 numbers to generate its encoding: -1, 0, and 1. This proposed method was proven to be less sensitive to the presence of noise. Next, LABP (Local Adaptive Binary Patterns) [18] is similar to the original LBP, except that the threshold used for creating the binary encoding is determined by the mean of the corresponding sampling points. This is basically the reason that this LBP variant is considered to be adaptive to the neighboring pixels. BELBP (Binarized Edge LBP) [19] is even more similar to the original LBP. However, what sets it apart is that BELBP works by applying LBP on edge features rather than the original image. The authors of this research utilized this texture features to detect abrupt-shot boundaries.

Talking more specifically about the other possible implementations of LBP variants, ILBP (Improved LBP) [20] was proposed and directly used by the authors to extract texture features from time-frequency image generated by diesel engine sound. Meanwhile, in the medical field, LMELBP (Local Maximum Edge Binary Patterns) [21] was proposed to perform segmentation on brain MR images. Not only that, [22] proposed MCG-LBP (Multi-Scale Continuous Gradient LBP) for leaky cable fixture detection in high-speed railway tunnels, whereas [23] utilized LBP-HF (LBP Histogram Fourier) for detecting image tampering through copy-move forgery.

Going back to the discussion regarding CLBP [15], there are actually several research papers that attempted to improve the algorithm. For instance, [24] proposed PC-LBP (Principal Curvatures LBP) while [25] combined CLBP with LTP, resulting in a new features called CLTP (Completed Local Ternary Pattern). However, [8] mentioned that CLTP is computationally expensive due to its high dimensionality. Thus, they propose to make further improvement on this algorithm by developing another one which is referred to as WCLTP (Wavelet Completed Local Ternary Pattern).

LBP-TOP (LBP from Three Orthogonal Planes) [26] can be considered as a special LBP variant thanks to its ability to capture texture as well as temporal information from a sequence of images. The idea was then adopted by [27], which proposed MCLBP (Multiple Channels LBP) to capture the texture features of a multi-channel image without needing to perform grayscale conversion. If these two LBPs work on 3D arrays, the ones proposed in [28] and [29] are specialized to perform LBP feature extraction on 1D array instead. The thresholding mechanism in such cases works in a similar way to a standard LBP, except that the sampling points are stretched along a single axis only, allowing it to extract meaningful information from waveforms.

The last three LBP variants to discuss are SCLBP (Sorted Consecutive LBP) [30], PNULBP (Prominent Non-Uniform LBP) [31], and $LBP^{mr}$ (Magnitude Ranking LBP) [32]. SCLBP works by exploiting the information contained in the consecutive bits in the binary encoding. Next, the author of PNULBP proposed this algorithm because they saw that $LBP^{u2}$ is missing some information as it almost completely discards the non-uniform pattern. The idea of PNULBP is to extract some more meaningful information from non-uniform patterns and combine it with $LBP^{u2}$ afterwards. Lastly, $LBP^{mr}$ was proposed because the weights used for creating the conventional LBP codes are fixed. The author of this algorithm regarded this as a problem and attempted to fix this issue by setting the sampling point with the largest magnitude as the MSB (Most Significant Bit) of the resulting binary code.

Among all LBP variations, there is no single one proposed specifically for handling the three drawbacks of LBP simultaneously, i.e., the requirement of bilinear interpolation, the inability to capture macrostructures and the sensitivity towards noise. Thus, this paper proposes a new LBP-based variant named Square Binary Patterns to handle these issues.

## III. Proposed Method and Experimental Setup

### A. Square Binary Patterns

This paper proposes an LBP-like feature which is named SBP (Square Binary Patterns). The procedure for extracting SBP features is shown in Fig. 5. The main steps written in the figure can actually be summarized as follows: center point determination, sampling points determination, thresholding, binary sequence generation, decimal conversion, and histogram generation. Each of the mentioned steps is going to be explained in the following sub-sections:

1) **Center point determination.** This stage is very similar to the conventional LBP. Every single pixel in the image is going to be indexed as a center pixel one by one starting from the top-left corner. The process finishes once it has reached the bottom-right corner of the input image. Later on, the intensity of each pixel will be compared with its surrounding neighbors.
2) **Sampling points determination.** Instead of following a circular pattern like what LBP does, SBP works by taking four groups of sampling points. These four groups are called “upper”, “right”, “lower” and “left” in which all of those groups are named after its position relative to the corresponding center pixel as shown in Fig. 2. The term “square” in SBP itself comes from these four sides of sampling points. Each group in each SBP level consists of 3 sampling points. Hence, the combination of those 4 groups will produce 12 sampling points. In the subsequent process, the 12 sampling points will be encoded in a clockwise direction starting from the leftmost pixel of the “upper” group.
3) **Thresholding.** SBP adopts the exact same thresholding method as LBP. The value of center pixel is going to act as the threshold value, where any sampling point that has lower value than the center will be encoded as 0, while the points with equal or higher value will be encoded as 1. See the illustration in Fig. 3.
4) **Decimal conversion.** With 12 binary digits, the maximum possible decimal value is 4095. The decimal values of all SBP levels are averaged and then normalized to ensure that the resulting value falls within the range of 0 to 255. The averaging process is inspired by the average smoothing method, which is useful for reducing noise effects. Meanwhile, the normalization process is necessary because the original range of 0 to 4095 is too large and redundant.
5) **Histogram generation.** Once the entire processes have been done to all pixels in the image, the resulting output is going to be in form of 2-dimensional array, which is referred to as the SBP feature. In order to use the feature for training a machine learning model, it is necessary to convert it into a single-dimensional array. This can

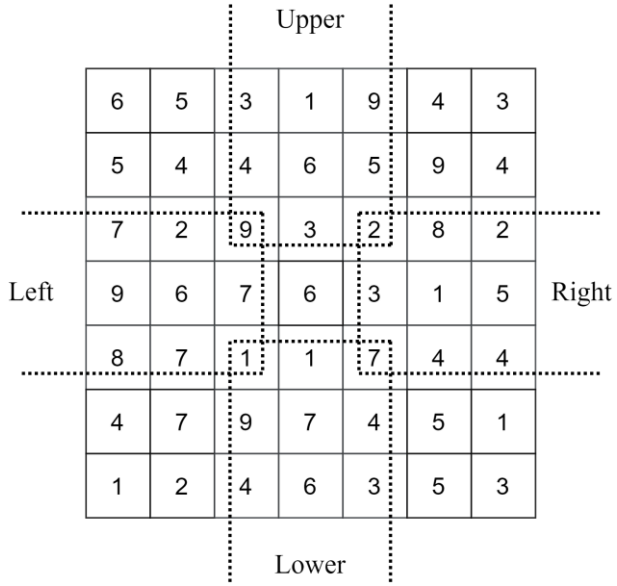


Fig. 2. The four groups of sampling points.

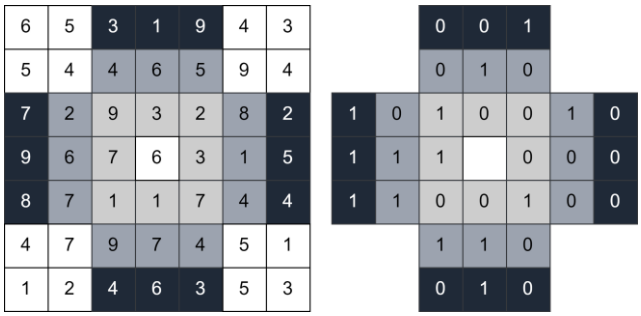


Fig. 3. Raw image to be processed (left) and the thresholding result (right).

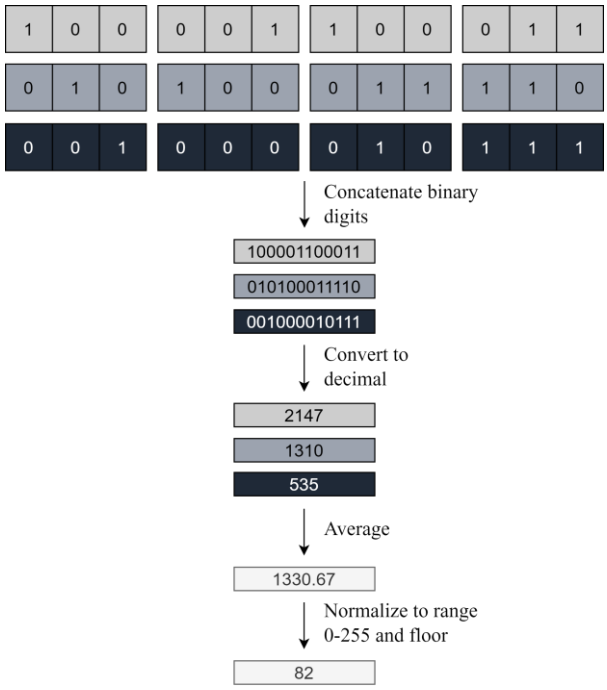


Fig. 4. An example of how SBP feature is computed. The initial binary sequence is based on the dummy image in Fig. 3.

simply be achieved by putting the pixel intensity values into a histogram.

The entire SBP feature extraction (excluding the histogram generation) can mathematically be expressed in equation (5). In this equation, $g_{np}$ denotes the pixel intensity at $n$-th level and $p$-th sampling point, while $g_c$ represents the intensity of center pixel. The binary encoding is then generated by the $s$ function which is basically the same as the one written in equation (4). The term $17/273$ in the equation is employed as an approach to keep the resulting intensity value to lie within the range of 0 to 255 only. Finally, the process of converting binary to decimal is done by the $2^p$ multiplier.

**Square Binary Patterns Feature Extraction Algorithm**

***Input:*** *SBP levels, intensity quantization levels, no of histogram bins*
***Output:*** *SBP features, SBP histogram*

```
for each pixel:
    for each SBP level:
        for each group (upper, right, lower, left):
            for each sampling point:
                take sampling point value
                threshold sampling points with center pixel value
                append thresholding result to the binary sequence
            concatenate binary encoding of all groups
        convert binary sequence to decimal
    calculate mean from all levels
    normalize the mean value
    update center pixel with the normalized decimal number
for each sub-block of SBP features:
    create SBP histogram
concatenate SBP histograms
return SBP features, concatenated SBP histograms
```

Fig. 5. The algorithm for extracting Square Binary Patterns.

$$SBP = \frac{17}{273 \times N}\left(\sum_{n=1}^{N}\sum_{p=1}^{12} s(g_{np} - g_c) \times 2^p\right) \qquad (5)$$

### *B. Datasets*

The author of this research utilized four datasets: KTH-TIPS (Kungliga Tekniska Högskolan – Textures Under Varying Illumination, Pose and Scale) [33], KTH-TIPS2-b [34], ALOT (Amsterdam Library of Textures) [35], and the one that we created ourselves named SuroTex (Surrounding Texture) [36]. The original KTH-TIPS dataset comprises 10 texture classes, each with 9 scales describing the pattern size. There are 81 images in each class, all with the dimensions of 200×200 pixels. To augment the number of images, the author applied a method that involved cropping with the size of 100×100 pixels from the top-left corner, top-right corner, bottom-left corner, and bottom-right corner of each image. By applying this method to all images, the dataset now has 324 images per class.

Despite having a similar name, KTH-TIPS2-b is actually different to KTH-TIPS. This dataset consists of 11 classes in which each of those has 432 texture images. However, further processing is required prior to using this dataset since the size of the images are somewhat not uniform. We decided to resize them all to 100×100 pixels in order to handle this issue. Next, ALOT is a dataset of 250 texture classes which contains 100 images each. Similar to KTH-TIPS2-b, since the image size is not uniform, hence it was resized to 100×100 as well before the

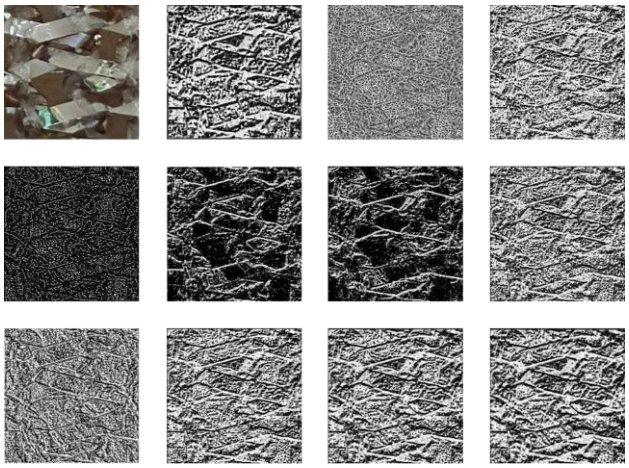

Fig. 6. First row: original image from SuroTex [36], LBP, $LBP^{riu2}$, $LBP^{u2}$. Second row: $LBP^{ri}$, LTP lower pattern, LTP upper pattern, FbLBP. Third row: ILBP, SBP1, SBP2, SBP3. (Images in all rows are arranged from left to right).

features being extracted. It is important to note that the author conducted the experiments on the first 30 and 50 texture classes as well as the entire 250 classes. Lastly, the SuroTex dataset was preprocessed in a similar way to ALOT. While the current version of this dataset contains 50 texture classes, this research utilized an earlier version of SuroTex which comprised 40 texture classes.

### *C. Experiment Steps*

The experiments conducted in this research followed the flowchart displayed in Fig. 7. The first step to be taken was to preprocess the images. The preprocessing stage consisted of grayscaling and smoothing with a kernel of size 5×5. In this case, average smoothing was chosen since it requires the least number of operations than the other smoothing methods. Once preprocessing was done, different texture feature extraction would be performed independently. The features to be compared in this research were LBP, $LBP^{riu2}$, $LBP^{u2}$, $LBP^{ri}$, MS-LBP, LTP, FbLBP, ILBP and the SBP itself. There were 2 and 3 variations of MS-LBP and SBP, respectively. Later on, these variations are going to be referred to as MSLBP2, MSLBP3, SBP1, SBP2, and SBP3, in which the numbers represent the neighbor distances to be taken into account. The resulting texture features were then used to train an SVM model. The training was done 20 times with different train-test split of ratio 80:20, in which the partition of each training iteration was selected randomly. Lastly, the model was evaluated by calculating the average testing accuracy score.

There were three types of experiments to be conducted. The first one was an experiment to test whether SBP is able to work faster than the other algorithms. Secondly, the classification performance of SBP under different textural scales were tested, in which this is useful to find out whether the proposed SBP feature is able to handle macrostructures. Lastly, there were also experiments done to find out the robustness of SBP under noisy conditions.

### *D. Experiment Parameter Settings*

Discussing the experimental parameter settings, the two parameters that we set to be fixed were $P$ and $R$ with the value of 8 and 1, respectively for all LBP-based variants. However, exceptions were made for MS-LBP2 which used $R$=1 and $R$=2, and MS-LBP3 which used $R$=1, $R$=2, and $R$=3. The resulting

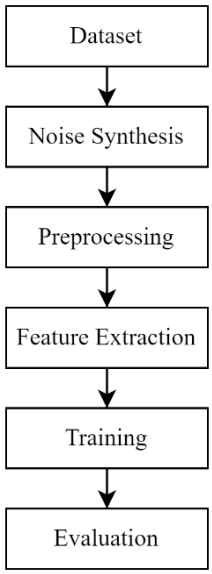


Fig. 7. The flowchart of this research.

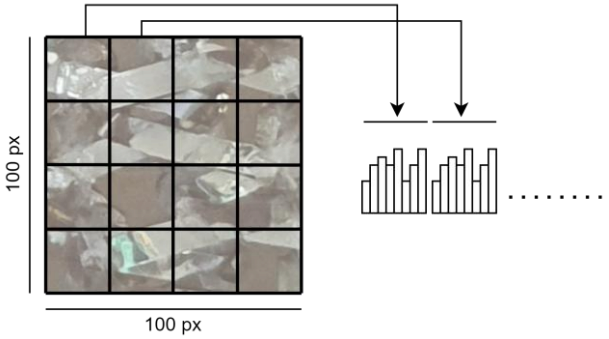


Fig. 8. How the histogram of each sub-image is concatenated.

TABLE I
FEATURE VECTOR LENGTHS

| Feature Name | Calculation | Feature Vector Length |
|---|---|---|
| LBP [1] | 30 × 16 | 480 |
| FBLBP [3] | 30 × 16 | 480 |
| LBPRIU [5] | 10 × 16 | 160 |
| LBPU [5] | 59 × 16 | 944 |
| LBPRI [5] | 36 × 16 | 576 |
| MSLBP2 [5] | 30 × 16 × 2 | 960 |
| MSLBP3 [5] | 30 × 16 × 3 | 1440 |
| LTP [7] | 30 × 16 × 2 | 960 |
| ILBP [20] | 30 × 16 | 480 |
| SBP1 (proposed) | 30 × 16 | 480 |
| SBP2 (proposed) | 30 × 16 | 480 |
| SBP3 (proposed) | 30 × 16 | 480 |

binary encoding of these MS-LBP radii were then concatenated to construct the final feature vector. Similarly, this mechanism is also applied to SBP. SBP1, SBP2, and SBP3 were all using $N$=1, $N$=2 and $N$=3, respectively. The major difference between SBP and LBP is the number of sampling points, where SBP used $P$=12 rather than 8. Lastly, LTP threshold was set to 5 in which this number was determined arbitrarily.

In addition to the parameters for feature extraction strategies, we also introduced two other parameters that were used to create histograms: *patchnum* and *binsperpatch*. The term *patchnum* refers to the number of divisions an image is partitioned into. The authors decided to set *patchnum* to 4 in all experiments. This basically means that a single image will be divided into 4 rows and 4 columns so that there will be 16 patches in total. The illustration shown in Fig. 8 displays an image of size 100×100

treated with *patchnum*=4. On the other hand, *binsperpatch* denotes the number of bins to be generated for each patch. In this case, the parameter was set to 30. It is important to note that the *binsperpatch* parameter does not apply to $LBP^{u2}$, $LBP^{ri}$, and $LBP^{riu2}$, as these methods have fixed histogram bin counts of 59, 36, and 10, respectively. Table I displays the final feature vector dimensions after taking into account all the 16 patches.

## IV. Results and Discussions

### A. Results on the Computation Time

The first experiment to be conducted was related to computation time, in which it was measured using Kaggle Notebook without GPU. In this experimental set all algorithms were tested independently to extract features from 100 images of size 100×100 pixels. The result in Fig. 9 shows that all SBP variants required the least time to complete the task, which essentially proves that our initial hypothesis is correct.

Before this experiment was done, we manually calculated the time complexity of all feature extraction algorithms. It was found that the presence of the bilinear interpolation operation causes the existing LBP-based algorithms to work much longer. The bilinear interpolation itself is basically a method for estimating the approximate value of a sampling point based on a weighted average of the values of its four neighboring pixels, which the calculation is done using equation (4). It is important to note that this equation is run for each sampling point requiring bilinear interpolation. Referring to Fig. 1, there are 4 sampling points need to be interpolated when $P$=8 and $R$=1, which are exactly the parameters we use for this experimental setup. The computation time of all LBP-based algorithms is going to be even longer especially when the number of sampling points $P$ is larger. Still referring to Fig. 1, it is seen that when $P$ and $R$ are set to 16 and 2, there are only 4 sampling points lie exactly at the center of a pixel, leaving the other 12 sampling points required to be interpolated.

Different from all other LBP-based approaches, SBP completely discards the bilinear interpolation process thanks to the square-shaped patterns. Taking a closer look at the algorithm, SBP appears to be computationally more expensive at first glance due to its fixed 12 number of sampling points, which theoretically result in a longer binary-to-decimal conversion process compared to other LBP-based algorithms that by default have 8 sampling points. However, our analytical calculation shows that the higher number of sampling points in SBP has a less significant impact on computation time compared to the bilinear interpolation process required by the existing LBP variants. This analysis is proven by the experimental result in Fig. 9, where SBP with $N$=1 completed the feature extraction process only in 33.2 seconds, which is approximately 4.6 times faster than the conventional LBP.

Apart from the bilinear interpolation, there are many other factors causing the existing LBP variants to work slower than SBP. The algorithm that we found slowest in this experiment is MSLBP3. It is seen in the same figure that it took 432.3 seconds to complete the task, which is 13 times slower than SBP1. This result makes sense because the algorithm essentially works by repeating the conventional LBP three times at different radii. MSLBP2 actually also has the same idea, yet it is faster than MSLBP3 since it only repeats the LBP process twice. The computation time of LTP appears to be close to MSLBP2, which

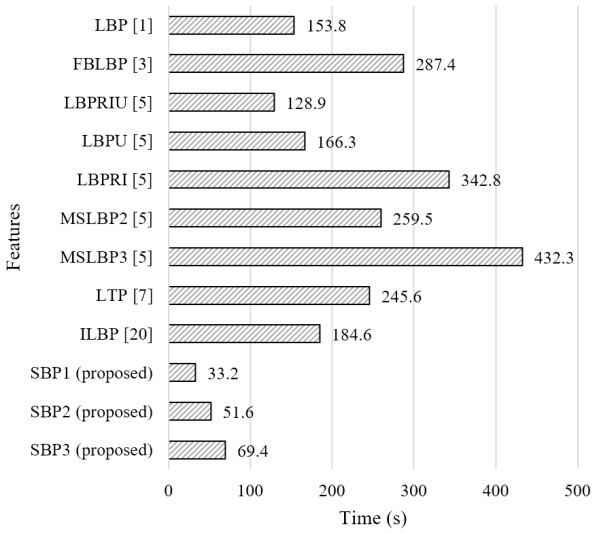


Fig. 9. The computation time of each texture feature extraction algorithm to process 100 images of size 100×100.

is because the process of extracting lower and upper patterns in the algorithm is also equivalent to running LBP twice. Next, LBPRI is slow due to the binary rotation mechanism which is done to achieve geometrical rotation invariance. Meanwhile, LBPU is algorithmically similar to the original LBP, yet it is slightly more expensive due to the pattern uniformity checking procedure. LBPRIU on the other hand, is computationally faster than the conventional LBP because it sums the binary values and treats the result as a decimal rather than performing the typical binary-to-decimal conversion. Lastly, the modern LBP variants of FBLBP and ILBP are also considered slow because of the pixel intensity averaging mechanism adopted in both algorithms.

### B. Results on Datasets of Different Textural Scales

The next experiment was related to the feature quality, in which it was examined by observing the accuracy produced by an SVM trained with each of the features. Higher accuracy indicates that the feature used for training contains a more meaningful information. All datasets mentioned earlier as well as the smaller version of ALOT, i.e., ALOT-30 and ALOT-50, would be employed to do the classification task.

Generally speaking, the performance of SBP appeared to be comparable with other features, as shown in the experimental results in Table II. It seemed like SBP with $N$=2 was the most optimal one, considering that SBP with other $N$ values always produced lower accuracy. The accuracy of SBP2 was observed to consistently surpass LBP, LBPRIU, LBPRI, LTP, and ILBP. FBLBP was able to obtain the exact same result as SBP2 on the 50-class ALOT dataset and was slightly better than SBP2 on our own texture dataset. Unfortunately, the accuracy of SBP2 was still lower than LBPU and MSLBP3. The higher accuracy of LBPU was likely because its smaller feature vector dimension contains more relevant information. In fact, this aligns with the principle of the uniform patterns in LBP, which was originally proposed to suppress the effect of non-uniform, irrelevant patterns that might appear in a texture. Meanwhile, the higher accuracy obtained by MSLBP3 might be due to the large amount of information contained in the resulting feature vector. This notion makes sense since it works by capturing three LBP features of different scales at once. Although the averaging

TABLE II
CLASSIFICATION RESULTS ON DIFFERENT DATASETS

| Feature Name | KTH-TIPS | KTH-TIPS2-b | ALOT-30 | ALOT-50 | ALOT-250 | SuroTex |
|---|---|---|---|---|---|---|
| LBP [1] | 0.783 | 0.806 | 0.804 | 0.807 | 0.747 | 0.858 |
| FBLBP [3] | 0.796 | 0.815 | 0.813 | 0.826 | 0.761 | 0.862 |
| LBPRIU [5] | 0.768 | 0.730 | 0.774 | 0.745 | 0.600 | 0.783 |
| LBPU [5] | 0.852 | 0.858 | 0.893 | 0.890 | 0.855 | 0.914 |
| LBPRI [5] | 0.806 | 0.733 | 0.670 | 0.613 | 0.380 | 0.829 |
| MSLBP2 [5] | 0.767 | 0.791 | 0.779 | 0.779 | 0.713 | 0.844 |
| MSLBP3 [5] | 0.828 | 0.843 | 0.850 | 0.850 | 0.766 | 0.897 |
| LTP [7] | 0.679 | 0.684 | 0.670 | 0.593 | 0.520 | 0.735 |
| ILBP [20] | 0.704 | 0.736 | 0.745 | 0.743 | 0.668 | 0.738 |
| SBP1 (proposed) | 0.740 | 0.785 | 0.766 | 0.766 | 0.689 | 0.813 |
| SBP2 (proposed) | 0.807 | 0.826 | 0.819 | 0.826 | 0.762 | 0.860 |
| SBP3 (proposed) | 0.776 | 0.823 | 0.789 | 0.794 | 0.758 | 0.811 |

Results matching the best SBP accuracy are highlighted in blue (■), results below the best SBP are highlighted in orange (■), and results above the best SBP are highlighted in green (■).

TABLE III
CLASSIFICATION RESULTS ON DIFFERENT SCALES IN KTH-TIPS DATASET

| Feature Name | Large Textures | Medium Textures | Small Textures |
|---|---|---|---|
| LBP [1] | 0.686 | 0.898 | 0.921 |
| FBLBP [3] | 0.704 | 0.903 | 0.921 |
| LBPRIU [5] | 0.692 | 0.907 | 0.926 |
| LBPU [5] | 0.777 | 0.947 | 0.963 |
| LBPRI [5] | 0.739 | 0.920 | 0.915 |
| MSLBP2 [5] | 0.651 | 0.875 | 0.891 |
| MSLBP3 [5] | 0.742 | 0.932 | 0.945 |
| LTP [7] | 0.547 | 0.795 | 0.731 |
| ILBP [20] | 0.613 | 0.792 | 0.850 |
| SBP1 (proposed) | 0.628 | 0.854 | 0.869 |
| SBP2 (proposed) | 0.707 | 0.927 | 0.938 |
| SBP3 (proposed) | 0.684 | 0.896 | 0.900 |

Results matching the best SBP accuracy are highlighted in blue (■), results below the best SBP are highlighted in orange (■), and results above the best SBP are highlighted in green (■).

mechanism adopted by SBP allows it to produce smaller feature vector dimension while still containing information from multiple scales, it might cause prominent information to be smoothed out. However, as demonstrated by the computational complexity experiments in Fig. 9, the slightly lower accuracy of SBP2 is a worthwhile trade-off, as it offers significantly higher computational speed, making it more suitable for real-time applications. In addition to Table II and the subsequent ones, the best-performing SBP as well as the accuracy that exactly matches the best SBP are going to be highlighted in blue. Meanwhile, all results worse than the best SBP will be highlighted in orange, yet if it is better, it will be colored in green.

In the upcoming experiments, authors attempted to perform the same classification task with the KTH-TIPS dataset that had been divided into three groups based on the scale of its textural patterns. The first group includes scales 1 to 3 with large textural patterns, while the next two groups comprise scales 4 to 6 and 7 to 9, presenting medium- and small-sized patterns, respectively. The results of this experiment are shown in Table III. The trend displayed in the table appeared to be similar to the previous one, except that the accuracies on larger textures were lower than that of the smaller textures. These experimental results essentially indicate that smaller textures are easier to classify than the larger ones. While LBPRI surpassed the best SBP in large textures, the latter was generally only outperformed by LBPU and MSLBP3. Especially in medium and small textures, SBP2 were only separated from the two features by a relatively small margin.

TABLE IV
ACCURACY DIFFERENCE OF EACH TEXTURE FEATURE TOWARDS THE CONVENTIONAL LBP

| Feature Name | Large Textures | Medium Textures | Small Textures |
|---|---|---|---|
| MSLBP3 [5] | 0.056 | 0.034 | 0.024 |
| SBP2 (bins=30) (proposed) | 0.021 | 0.029 | 0.018 |
| SBP2 (bins=100) (proposed) | 0.060 | 0.030 | 0.027 |

TABLE V
THE CLASSIFICATION ACCURACY OF SBP ON LARGE TEXTURES WITH DIFFERENT N VALUES

| SBP Variant | No of bins per patch | |
|---|---|---|
| | 30 | 100 |
| SBP1 (N=1) | 0.628 | 0.703 |
| SBP2 (N=2) | 0.707 | 0.722 |
| SBP3 (N=3) | 0.684 | 0.740 |

Next, authors are also interested in finding out how each feature performs compared to the conventional LBP in terms of the accuracy difference. It is seen in Table IV that MSLBP3 outperformed the original LBP by 2.4%, 3.4%, and 5.6% on small, medium, and large-sized textures, respectively. Such an increase in the accuracy difference when the textural scale gets larger indicates the superiority of multiscale-based LBP variant in capturing larger repeating patterns as compared to the original LBP. Thus, it makes sense to say that wider MSLBP radius somewhat correlates to its ability in recognizing large textures. This kind of behavior can also be observed in SBP2, where its accuracy difference towards LBP in small textures (1.8%) are lower than those of medium (2.9%) and large textures (2.1%). Note that these results are obtained on SBP2 with the number of histogram bins in each patch set to 30. During the experiments, authors attempted to increase the value of this parameter to 100 both for SBP2 and LBP in order to gain new insights. Using this configuration, it is found that the accuracy gap between the two reached 6% on large textures, strengthening the notion that LBP-based features with multiscale approach allows it to capture larger repeating patterns.

In order to support these findings, we performed further experiment on the $N$ parameter of SBP specifically on large textures, which the results are displayed in Table V. According to the table, it is seen that classification accuracy tends to improve as the $N$ is increased, with the results being more apparent when the number of histogram bins in each patch is set to 100. This confirms that the proposed SBP aligns with our previous findings where larger textures are able to be recognized better with multiscale-based feature extraction algorithms.

### C. *Results on Datasets of Noisy Images*

In order to find out the robustness of SBP in noisy conditions, we carried out more experiments on each textural scale group in the KTH-TIPS dataset that had been synthesized with noise. There were two types of noise used, namely Gaussian and s&p (salt and pepper) noise. In the case of Gaussian noise, variance was the parameter used to control noise severity. This parameter was set to 0.05, 0.1, 0.15, and 0.2, in which the latter is the one that causes the images to look the most unclear. This section is going to reveal which feature obtained the least accuracy

TABLE VI
THE EFFECT OF GAUSSIAN NOISE ON LARGE, MEDIUM, AND SMALL TEXTURES

| Feature Name | Large Textures | | | | | Medium Textures | | | | | Small Textures | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | No Noise | Variance | | | | No Noise | Variance | | | | No Noise | Variance | | | |
| | | 0.05 | 0.1 | 0.15 | 0.2 | | 0.05 | 0.1 | 0.15 | 0.2 | | 0.05 | 0.1 | 0.15 | 0.2 |
| LBP [1] | 0.686 | 0.401 | 0.333 | 0.314 | 0.276 | 0.898 | 0.517 | 0.459 | 0.406 | 0.393 | 0.921 | 0.578 | 0.518 | 0.435 | 0.387 |
| FBLBP [3] | 0.704 | 0.398 | 0.345 | 0.324 | 0.274 | 0.903 | 0.527 | 0.460 | 0.408 | 0.387 | 0.921 | 0.591 | 0.519 | 0.447 | 0.394 |
| LBPRIU [5] | 0.692 | 0.362 | 0.325 | 0.306 | 0.280 | 0.907 | 0.420 | 0.358 | 0.339 | 0.291 | 0.926 | 0.492 | 0.394 | 0.371 | 0.306 |
| LBPU [5] | 0.777 | 0.424 | 0.386 | 0.318 | 0.301 | 0.947 | 0.539 | 0.469 | 0.420 | 0.395 | 0.963 | 0.594 | 0.519 | 0.468 | 0.403 |
| LBPRI [5] | 0.739 | 0.377 | 0.326 | 0.299 | 0.282 | 0.920 | 0.491 | 0.422 | 0.373 | 0.349 | 0.915 | 0.513 | 0.451 | 0.410 | 0.369 |
| MSLBP2 [5] | 0.651 | 0.359 | 0.307 | 0.277 | 0.253 | 0.875 | 0.484 | 0.425 | 0.372 | 0.349 | 0.891 | 0.557 | 0.472 | 0.399 | 0.340 |
| MSLBP3 [5] | 0.742 | 0.465 | 0.387 | 0.341 | 0.296 | 0.932 | 0.569 | 0.491 | 0.440 | 0.415 | 0.945 | 0.575 | 0.490 | 0.428 | 0.372 |
| LTP [7] | 0.547 | 0.481 | 0.390 | 0.336 | 0.308 | 0.795 | 0.614 | 0.529 | 0.472 | 0.443 | 0.731 | 0.640 | 0.554 | 0.503 | 0.441 |
| ILBP [20] | 0.613 | 0.370 | 0.325 | 0.298 | 0.267 | 0.792 | 0.428 | 0.369 | 0.353 | 0.324 | 0.850 | 0.500 | 0.432 | 0.392 | 0.378 |
| SBP1 (proposed) | 0.628 | 0.352 | 0.308 | 0.287 | 0.259 | 0.854 | 0.491 | 0.434 | 0.376 | 0.365 | 0.869 | 0.533 | 0.455 | 0.422 | 0.375 |
| SBP2 (proposed) | 0.707 | 0.434 | 0.358 | 0.338 | 0.322 | 0.927 | 0.556 | 0.483 | 0.467 | 0.413 | 0.938 | 0.569 | 0.479 | 0.426 | 0.381 |
| SBP3 (proposed) | 0.684 | 0.434 | 0.379 | 0.329 | 0.288 | 0.896 | 0.529 | 0.439 | 0.400 | 0.368 | 0.900 | 0.519 | 0.461 | 0.402 | 0.388 |

Results matching the best SBP accuracy are highlighted in blue (■), results below the best SBP are highlighted in orange (■), and results above the best SBP are highlighted in green (■).

TABLE VII
THE EFFECT OF S&P NOISE ON LARGE, MEDIUM, AND SMALL TEXTURES

| Feature Name | Large Textures | | | | | Medium Textures | | | | | Small Textures | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | No Noise | Amount | | | | No Noise | Amount | | | | No Noise | Amount | | | |
| | | 0.05 | 0.1 | 0.15 | 0.2 | | 0.05 | 0.1 | 0.15 | 0.2 | | 0.05 | 0.1 | 0.15 | 0.2 |
| LBP [1] | 0.686 | 0.597 | 0.513 | 0.462 | 0.425 | 0.898 | 0.775 | 0.658 | 0.577 | 0.508 | 0.921 | 0.783 | 0.712 | 0.663 | 0.619 |
| FBLBP [3] | 0.704 | 0.598 | 0.522 | 0.480 | 0.427 | 0.903 | 0.775 | 0.664 | 0.580 | 0.512 | 0.921 | 0.780 | 0.719 | 0.672 | 0.618 |
| LBPRIU [5] | 0.692 | 0.568 | 0.492 | 0.416 | 0.380 | 0.907 | 0.822 | 0.649 | 0.559 | 0.463 | 0.926 | 0.800 | 0.682 | 0.596 | 0.538 |
| LBPU [5] | 0.777 | 0.658 | 0.569 | 0.494 | 0.438 | 0.947 | 0.843 | 0.719 | 0.618 | 0.544 | 0.963 | 0.830 | 0.738 | 0.687 | 0.634 |
| LBPRI [5] | 0.739 | 0.606 | 0.519 | 0.457 | 0.423 | 0.920 | 0.788 | 0.684 | 0.565 | 0.535 | 0.915 | 0.761 | 0.665 | 0.619 | 0.582 |
| MSLBP2 [5] | 0.651 | 0.569 | 0.483 | 0.418 | 0.388 | 0.875 | 0.746 | 0.631 | 0.543 | 0.469 | 0.891 | 0.752 | 0.682 | 0.630 | 0.592 |
| MSLBP3 [5] | 0.742 | 0.656 | 0.571 | 0.517 | 0.466 | 0.932 | 0.797 | 0.694 | 0.608 | 0.526 | 0.945 | 0.775 | 0.692 | 0.645 | 0.592 |
| LTP [7] | 0.547 | 0.532 | 0.517 | 0.487 | 0.441 | 0.795 | 0.697 | 0.666 | 0.642 | 0.592 | 0.731 | 0.741 | 0.719 | 0.700 | 0.668 |
| ILBP [20] | 0.613 | 0.501 | 0.428 | 0.392 | 0.353 | 0.792 | 0.564 | 0.506 | 0.444 | 0.403 | 0.850 | 0.695 | 0.611 | 0.544 | 0.501 |
| SBP1 (proposed) | 0.628 | 0.547 | 0.462 | 0.421 | 0.394 | 0.854 | 0.720 | 0.634 | 0.564 | 0.485 | 0.869 | 0.749 | 0.660 | 0.621 | 0.560 |
| SBP2 (proposed) | 0.707 | 0.644 | 0.553 | 0.495 | 0.443 | 0.927 | 0.792 | 0.693 | 0.608 | 0.550 | 0.938 | 0.805 | 0.699 | 0.648 | 0.582 |
| SBP3 (proposed) | 0.684 | 0.624 | 0.519 | 0.469 | 0.420 | 0.896 | 0.731 | 0.610 | 0.537 | 0.482 | 0.900 | 0.735 | 0.617 | 0.576 | 0.509 |

Results matching the best SBP accuracy are highlighted in blue (■), results below the best SBP are highlighted in orange (■), and results above the best SBP are highlighted in green (■).

degradation as the level of noise increases.

The first experiment related to noise was performed by applying Gaussian noise on images of large texture, in which the results are displayed in Table VI. Initially in a non-noisy condition, SBP2 ranked fourth behind LBPU, MSLBP3 and LBPRI with an accuracy of 70.7%. As the Gaussian noise added, the accuracy of LBPU and LBPRI drastically dropped to 42.4% and 37.7% respectively, causing them to be lower than SBP2 which obtained 43.4%. Despite this fact, SBP2 remained in the third rank since LTP somehow obtained 48.1%, surpassing both SBP2 and MSLBP3. Nevertheless, LTP dominated only at Gaussian noise variances of 0.05 and 0.1. Meanwhile, MSLBP3 became the best one at noise 0.15, at which point SBP2 ranked second with 33.8% accuracy, which was only 0.3% lower than MSLBP3. SBP2 finally took first place at the most severe noise level with the accuracy of 32.2%, exceeding MSLBP3 and LTP which had previously performed better at lower noise levels.

On the other hand, the conventional LBP, which is the main baseline of this research, was completely unable to surpass SBP2 at all. Similar behavior could also be observed on LBPRIU, MSLBP2, FBLBP, and ILBP. All these results prove that SBP is a texture feature suitable to be extracted in the case where the images contain large repeating patterns with a high level of Gaussian noise.

In the case when the same noise type was applied to medium-sized textures, LBPU appeared not to be able to surpass SBP2 at all despite its superiority over all other features in a normal condition as shown in Table II. It is interesting to see that LTP, which used to produce a relatively low accuracy in non-noisy condition, became the best one across all noise variance, exceeding both MSLBP3 and SBP2. Despite not being the best one, the classification rate of SBP2 was ranked third after LTP and MSLBP3, except that at 0.15 noise it ranked second behind LTP with a gap of only 0.5%, surpassing MSLBP3 by 2.7% margin. This essentially proved that SBP2 is a good feature to consider when it comes to classifying medium-sized textures in noisy condition since it can still achieve high accuracy while having much smaller computational complexity as shown in Fig. 9. In addition to this experiment set, the conventional LBP as well as FBLBP, LBPRIU, LBPRI, MSLBP2, and ILBP were completely unable to perform as good as SBP2, similar to the experimental results conducted on large textures.

Different from the previous two textural sizes, the experimental result on small textures highlights the weakness of SBP. According to Table VI, even the conventional LBP was able to obtain better results than the best SBP at noise variance 0.05, 0.1, and 0.15 with the margins of 0.9%, 3.9% and 0.9%, respectively. At noise variance 0.2, the accuracy of the best SBP was indeed better than LBP, yet the difference between the two was only 0.1%. This indicates that SBP is not suitable for

conditions with small, noisy textures. Meanwhile LBPU, which previously struggled to surpass SBP2 in medium-sized textures, is now able to do so across all noise levels. This phenomenon probably happened because both LBP and LBPU work by taking into account the pixels exactly next to it, enabling it to pay more attention to the smaller repeating patterns. On the other hand, SBP did not perform as effectively since it works by capturing the information from multiple levels and taking the average from them. Such an averaging mechanism causes it to lose some textural information, especially when noise starts to disrupt the images. Even though MSLBP3 also takes into consideration pixel values from multiple scales, yet it did not encounter this problem because these features of different scales are concatenated rather than averaged.

Similar experiment was also conducted using s&p noise, where the accuracy scores are shown in Table VII. In this type of noise, the value of 0.05, 0.1, 0.15 and 0.2 represent the proportion of noise compared to the total number of pixels within an image. Table VII shows the influence of s&p noise on textures of different sizes.

It is seen in the table that initially LBPU performed better than both SBP2 and MSLBP3 at noise amount 0.05 in large textures, but its accuracy got surpassed by both of them at noise amount 0.15 and 0.2. Even though SBP2 never achieved the best score in this texture size, yet its classification performance ranked second right after MSLBP3 at the two most severe noise levels with the difference of 2.2% at noise amount 0.15 and 2.3% at noise amount 0.2. The reason that it could not perform better than MSLBP3 is probably because the presence of noise, which should be able to be handled by SBP, was not as significant as the presence of large textures, which the information can be captured better by MSLBP3. Furthermore, the table also shows that there is no other feature performed better than SBP2 except LBPU and MSLBP3 across all noise amounts in large texture category, in which this behavior supports the notion that SBP is a good feature to be extracted when the dataset contains large repeating patterns especially when s&p noise is present.

Examining the impact of s&p noise on medium-sized textures, LBPU, MSLBP3 and LBPRIU initially obtained better accuracy than the best-performing SBP at noise amount 0.05. However, LBPRIU suddenly dropped significantly at the subsequent noise amounts such that its accuracy became lower than SBP2. The accuracy gap between LBPU to SBP2 and MSLBP3 to SBP2 were also getting narrower as the noise amount increased. This trend continued until eventually SBP2 surpassed both of those at the highest noise level. This experimental results proved that SBP2 is more robust against noise as compared to LBPU and MSLBP3. We do acknowledge that the result obtained by SBP2 was not as good as LTP at 0.2 noise amount, yet it can still be regarded as a good one considering that it performed better than all the remaining LBP variants.

The accuracy trend for s&p noise on small textures followed a similar pattern to that of Gaussian noise on the same textural size. Initially, the performance of SBP2 was actually good enough. It is seen in Table VII that it ranked second right after LBPU when the noise amount was set to 0.05 with the difference of 2.5%. However, the classification rate of SBP degraded abruptly, causing it to be surpassed by LBP, LTP, and FBLBP at noise amounts 0.1 and 0.15. The result became even worse even for the best SBP at noise amount 0.2 as it only performed better than LBPRIU and ILBP. These results suggest that SBP is not a good features to be used when the classification task contains small and noisy texture images.

## V. Conclusions

Based on the above discussions, it can be concluded that the proposed SBP feature has successfully addressed the three problems encountered by the conventional LBP, i.e., the need for bilinear interpolation, the inability to capture macrostructures, and the difficulty in handling noise. First, the proposed sampling points determination mechanism allows SBP to omit the bilinear interpolation process which causes it to be computationally much more efficient than other existing LBP-based algorithms. Secondly, even though SBP does not have the best accuracy in general texture classification task, yet it is usually ranked third just behind $LBP^{u2}$ and MS-LBP. This essentially indicates that SBP is able to work much faster than the two without sacrificing too much textural information. It is worth noting that both algorithms, especially MS-LBP, have an extremely slow computational speed which causes it not to be suitable to be used in real-time. Moreover, the same experiment set also proved that SBP has a better capability in capturing macrostructures as compared to the conventional LBP. Third, the experimental results also showed that the classification rate of SBP under heavy noise conditions is very good especially when the repeating patterns are large.

Despite all these strengths, this does not necessarily mean that SBP has no drawbacks. In fact, there is actually a case where SBP can not compete with the others including the conventional LBP itself, in which this particular phenomenon usually occurs when the dataset contains small-sized and noisy textures. Furthermore, it is also necessary to acknowledge that even though SBP is able to perform very well under heavy noise in large textures, yet its classification rate can still be considered low. This behavior is still useful in some cases since the classifier did not perform random guess. However, further research to improve the accuracy under such conditions is still necessary to be done in order to achieve more reliable results.

## References


[1] T. Ojala, M. Pietikäinen, and D. Harwood, "Performance Evaluation of Texture Measures with Classification Based on Kullback Discrimination of Distributions," in *Proceedings of 12th International Conference on Pattern Recognition*, IEEE, Oct. 1994. doi: 10.1109/ICPR.1994.576366.

[2] A. K. Bedi, R. K. Sunkaria, and S. K. Randhawa, "Local Binary Pattern Variants: A Review," in *ICSCCC 2018 - 1st International Conference on Secure Cyber Computing and Communications*, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 234–237. doi: 10.1109/ICSCCC.2018.8703326.

[3] Z. Pan, Z. Li, H. Fan, and X. Wu, "Feature based local binary pattern for rotation invariant texture classification," *Expert Syst Appl*, vol. 88, pp. 238–248, Dec. 2017, doi: 10.1016/j.eswa.2017.07.007.

[4] F. Bianconi, E. González, and A. Fernández, "Dominant local binary patterns for texture classification: Labelled or unlabelled?," *Pattern Recognit Lett*, vol. 65, pp. 8–14, Jul. 2015, doi: 10.1016/j.patrec.2015.06.025.

[5] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns," *IEEE Trans Pattern Anal Mach Intell*, vol. 24, no. 7, pp. 971–981, 2002, doi: 10.1109/TPAMI.2002.1017623.

[6] L. Liu, S. Lao, P. W. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Median Robust Extended Local Binary Pattern for Texture Classification," *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1368–1381, Mar. 2016, doi: 10.1109/TIP.2016.2522378.

[7] X. Tan and B. Triggs, "Enhanced Local Texture Feature Sets for Recognition Under Difficult Lighting Conditions," *IEEE transactions on image processing*, vol. 19, no. 6, pp. 1635–1650, 2010, doi: 10.1109/TIP.2010.2042645.

[8] A. M. D. Shamaileh, T. H. Rassem, L. S. Chuin, and O. N. Al Sayaydeh, "A New Feature-Based Wavelet Completed Local Ternary Pattern (Feat-WCLTP) for Texture Image Classification," *IEEE Access*, vol. 8, pp. 28276–28288, 2020, doi: 10.1109/ACCESS.2020.2972151.

[9] X. Qi, Y. Qiao, C. G. Li, and J. Guo, "Multi-scale joint encoding of local binary patterns for texture and material classification," in *BMVC 2013 - Electronic Proceedings of the British Machine Vision Conference 2013*, British Machine Vision Association, BMVA, 2013. doi: 10.5244/C.27.40.

[10] Z. Dan, Y. Chen, Z. Yang, and G. Wu, "An improved local binary pattern for texture classification," *Optik (Stuttg)*, vol. 125, no. 20, pp. 6320–6324, Oct. 2014, doi: 10.1016/j.ijleo.2014.08.003.

[11] C. E. Prema, S. Suresh, M. N. Krishnan, and N. Leema, "A Novel Efficient Video Smoke Detection Algorithm Using Co-occurrence of Local Binary Pattern Variants," *Fire Technol*, vol. 58, no. 5, pp. 3139–3165, Sep. 2022, doi: 10.1007/s10694-022-01306-2.

[12] F. Yuan, J. Shi, X. Xia, L. Zhang, and S. Li, "Encoding pairwise Hamming distances of Local Binary Patterns for visual smoke recognition," *Computer Vision and Image Understanding*, vol. 178, pp. 43–53, 2019, doi: 10.1016/j.cviu.2017.00.000.

[13] Yuanbin Wang, Qian Han, Yuanyuan Li, and Yujie Li, "Video Smoke Detection Based on Multi-feature Fusion and Modified Random Forest," *Engineering Letters*, vol. 29, no. 3, pp. 1115-1122, 2021.

[14] Z. Guo, L. Zhang, and D. Zhang, "Rotation invariant texture classification using LBP variance (LBPV) with global matching," *Pattern Recognit*, vol. 43, no. 3, pp. 706–719, 2010, doi: 10.1016/j.patcog.2009.08.017.

[15] Z. Guo, L. Zhang, and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Transactions on Image Processing*, vol. 19, no. 6, pp. 1657–1663, Jun. 2010, doi: 10.1109/TIP.2010.2044957.

[16] L. Liu, L. Zhao, Y. Long, G. Kuang, and P. Fieguth, "Extended local binary patterns for texture classification," *Image Vis Comput*, vol. 30, no. 2, pp. 86–99, 2012, doi: 10.1016/j.imavis.2012.01.001.

[17] Y. Song, J. Sa, Y. Luo, and Z. Zhang, "A comprehensively improved local binary pattern framework for texture classification," *Multimed Tools Appl*, 2024, doi: 10.1007/s11042-024-19877-3.

[18] D. Sharma and A. Selwal, "An intelligent approach for fingerprint presentation attack detection using ensemble learning with improved local image features," *Multimed Tools Appl*, vol. 81, no. 16, pp. 22129–22161, Jul. 2022, doi: 10.1007/s11042-021-11254-8.

[19] H. M. Nandini, H. K. Chethan, and B. S. Rashmi, "Shot based keyframe extraction using edge-LBP approach," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 4537–4545, Jul. 2022, doi: 10.1016/j.jksuci.2020.10.031.

[20] Y. Cai, G. Xu, A. Li, and X. Wang, "A Novel Improved Local Binary Pattern and Its Application to the Fault Diagnosis of Diesel Engine," *Shock and Vibration*, vol. 2020, 2020, doi: 10.1155/2020/9830162.

[21] N. Venu, "Segmentation Analysis for Local Maximum Edge Binary Patterns using Medical Images," *IJFANS International Journal of Food and Nutritional Sciences*, vol. 12, no. 1, 2023, [Online]. Available: https://www.researchgate.net/publication/361923631

[22] Y. Zhang, Z. Song, and W. Guo, "Multi-Scale Continuous Gradient Local Binary Pattern for Leaky Cable Fixture Detection in High-Speed Railway Tunnel," *IEEE Access*, vol. 9, pp. 147102–147113, 2021, doi: 10.1109/ACCESS.2021.3124676.

[23] Badal Soni, Pradip K. Das, and Dalton Meitei Thounaojam, "Dual System for Copy-move Forgery Detection using Block-based LBP-HF and FWHT Features," *Engineering Letters*, vol. 26, no. 1, pp. 171–180, 2018.

[24] Q. Kou, D. Cheng, L. Chen, and K. Zhao, "A Multiresolution Gray-Scale and Rotation Invariant Descriptor for Texture Classification," *IEEE Access*, vol. 6, pp. 30691–30701, 2018, doi: 10.1109/ACCESS.2018.2842078.

[25] T. H. Rassem and B. E. Khoo, "Completed Local Ternary Pattern for Rotation Invariant Texture Classification," *The Scientific World Journal*, vol. 2014, 2014, doi: 10.1155/2014/373254.

[26] Feng D and Ren F, "Dynamic Facial Expression Recognition based on Two-Stream-CNN with LBP-TOP," *5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, 2018.

[27] X. Shu, Z. Song, J. Shi, S. Huang, and X. J. Wu, "Multiple channels local binary pattern for color texture representation and classification," *Signal Process Image Commun*, vol. 98, Oct. 2021, doi: 10.1016/j.image.2021.116392.

[28] J. Prasanna, S. T. George, and M. S. P. Subathra, "Detection of neurodegenerative diseases using hybrid MODWT and adaptive local binary pattern," *Neural Comput Appl*, Nov. 2024, doi: 10.1007/s00521-024-10222-1.

[29] A. K. Jaiswal and H. Banka, "Local pattern transformation based feature extraction techniques for classification of epileptic EEG signals," *Biomed Signal Process Control*, vol. 34, pp. 81–92, Apr. 2017, doi: 10.1016/j.bspc.2017.01.005.

[30] J. Ryu, S. Hong, and H. S. Yang, "Sorted Consecutive Local Binary Pattern for Texture Classification," *IEEE Transactions on Image Processing*, vol. 24, no. 7, pp. 2254–2265, Jul. 2015, doi: 10.1109/TIP.2015.2419081.

[31] P. Kiran and K. Reddy, "Texture Classification Using Prominent Non Uniform Local Binary Patterns," *International Research Journal of Engineering and Technology*, vol. 3, no. 9, pp. 730–733, 2016.

[32] Y. Luo, J. Sa, Y. Song, H. Jiang, C. Zhang, and Z. Zhang, "Texture classification combining improved local binary pattern and threshold segmentation," *Multimed Tools Appl*, vol. 82, no. 17, pp. 25899–25916, Jul. 2023, doi: 10.1007/s11042-023-14749-8.

[33] M. Fritz, E. Hayman, B. Caputo, and J.-O. Eklundh, "THE KTH-TIPS database," 2004.

[34] P. Mallikarjuna, A. T. Targhi, M. Fritz, E. Hayman, B. Caputo, and J.-O. Eklundh, "THE KTH-TIPS2 database," 2006.

[35] G. J. Burghouts and J. M. Geusebroek, "Material-specific adaptation of color invariant features," *Pattern Recognit Lett*, vol. 30, no. 3, pp. 306–313, Feb. 2009, doi: 10.1016/j.patrec.2008.10.005.

[36] M. A. Putra, Wahyono, and A. Harjoko, "SuroTex: Surrounding Texture Dataset," *Data Brief*, vol. 59, p. 111292, Apr. 2025, doi: https://doi.org/10.1016/j.dib.2025.111292.

**Muhammad Ardi Putra** was born in Yogyakarta, Indonesia in 1998. He received a Bachelor of Computer Science degree in Computer Science from Universitas Gadjah Mada, Yogyakarta, Indonesia in 2021. Currently he is pursuing a Doctoral degree in the same university. His research interests include computer vision, machine learning, and deep learning.

**Agus Harjoko** received the Bachelor of Electronics and Instrumentation degree from Universitas Gadjah Mada, Indonesia in 1986, the Master of Computer Science from University of New Brunswick, Canada in 1990, and Ph.D. degree also from University of New Brunswick, Canada in 1996. He is currently a lecturer in the Department of Computer Science and Electronics, where he has been a professor since 2023. His research interests involve computer vision, pattern recognition and instrumentation.

**Wahyono** received his bachelor's degree in computer science from Universitas Gadjah Mada, Yogyakarta, Indonesia in 2010 and Ph.D. degree from University of Ulsan in 2017. He is currently a lecturer in the Department of Computer Science and Electronics, Universitas Gadjah Mada. His research area involves digital image processing, pattern recognition, machine learning, computer vision, surveillance systems and software engineering.

**Kanghyun Jo** received the Ph.D. degree in computer-controlled machinery from Osaka University, Osaka, Japan, in 1997. After a year of experience with ETRI as a Postdoctoral Research Fellow, he joined the School of Electrical Engineering, University of Ulsan, Ulsan, South Korea, where he is currently the Faculty Dean of the School of Electrical Engineering. His current research interests include computer vision, robotics, autonomous vehicles, and ambient intelligence. He was the Director or an AdCom Member of the Institute of Control, Robotics and Systems and the Society of Instrument and Control Engineers, the IEEE IES Technical Committee on Human Factors Chair, an AdCom Member, and the Secretary, until 2019. He has also been involved in organizing many international conferences, such as the International Workshop on Frontiers of Computer Vision, the International Conference on Intelligent Computation, the International Conference on Industrial Technology, the International Conference on Human System Interactions, and the Annual Conference of the IEEE Industrial Electronics Society. He is currently an Editorial Board Member for international journals, such as the International Journal of Control, Automation and Systems and Transactions on Computational Collective Intelligence.