Real-Time Texture Extraction Based on the Improved Median Robust Extended Local Binary Pattern

Mengnan Wang
School of Information and Electronics, Beijing Institute of Technology, Beijing, China 100081 CN
wangmn@bit.edu.cn

Yanjun Zhang*
School of Information and Electronics, Beijing Institute of Technology, Beijing, China 100081 CN
zhangyj@bit.edu.cn

ABSTRACT
Local binary patterns (LBP) are considered as the most computation efficient and high-performance texture features. Among all variants of the LBPs, Median Robust Extended Local Binary Pattern (MRELBP) [1] is considered as one of the highest-performance one. Based on the MRELBP, this paper proposes an improved real-time texture extraction architecture implemented on a field-programmable gate array (FPGA) device. The fixed-point fraction is used to replace the double float-point fraction, only results in less than 1% difference to the accuracy. The proposed system is implemented on Xilinx Zynq-7045, and the extraction time is linear with the size of the input image. The extraction time of the 128x128 image in the dataset Outex-TC is merely 85.64μs, which is more than 4800 times faster than Liu et al. implemented on MATLAB [2].

CCS Concepts
• **Hardware→Integrated circuits→Reconfigurable logic and FPGAs→Hardware accelerators.**

Keywords
MRELBP; real-time; feature extraction; Field-Programmable Gate Array (FPGA).

1. INTRODUCTION
Texture is a fundamental characteristic of many natural surfaces, which is considered as an important feature in the classification algorithms. Ojala et al. proposed the original local binary pattern (LBP) descriptor to extract the texture feature of the image [4]. The LBP is one of the most prominent texture descriptors, because of its high efficiency of computation and high quality of description. Thus a large number of LBP variants are subsequently proposed, such as the Completed Local Binary Pattern (CLBP) [5], the Binary Rotation Invariant and Noise Tolerant Texture Classification (BRINT) [6], the Scale Selective Local Binary Patterns (SSLBP) [7], and the Median Robust Extended Local Binary Pattern (MRELBP) [1].

Most of variants are realized by improving the topology structure, encoding method, and combining complementary features. Liu et al. performed performance evaluation approach of texture classification, which assessed thirty-two recent most promising LBP variants and eight non-LBP descriptors based on deep convolutional networks on thirteen widely-used texture datasets [2]. The evaluation result shows that the best overall performance is obtained by the MRELBP feature. It outperforms significantly, especially in random noise corrupted situations. Moreover, MRELBP has much lower feature dimensionality than some other method.

As an effective feature extraction method, LBP is widely used in image segmentation, object recognition, and classification. Generally, these fields have a high requirement for real-time performance. OMER et al. proposed a pattern recognition system based on an integrated approach of the local binary pattern (LBP) and content-addressable memory (CAM), in which the implementation result showed that the worst-case lookup time for one complete recognized pattern is merely 1.05μs [8]. Christos Kyrkou et al. proposed the acceleration of cascade SVMs through a hybrid processing hardware architecture optimized for the cascade SVM classification, using the LBP as the texture extraction step. The proposed architecture achieves a real-time processing rate of 40 frames/s for the benchmark face detection application [9]. Zhang et al. presented a hardware architecture of a face recognition algorithm based on LBP. The proposed system costs 5975 LUTs and the clock frequency can be up to 233MHz. The recognition speed is 74 times faster than in software [10]. Tomasz Kryjak et al. presented a head-shoulder detection algorithm implemented on FPGA, which used LBP for feature extraction and SVM for classification. The final system can process a video stream of resolution 640x480 @60 fps in real-time [11]. All of the above paper implemented the algorithm on the FPGA using original LBP for texture extraction. However, the accuracy of LBP is only around 90%, while the accuracy of MRELBP in software can be up around 99%. Although the MRELBP is considered as the most performance LBP variants, there is no precedent trying to implement MRELBP in hardware.

This paper proposed an improved MRELBP architecture which can be implemented in hardware and meet the requirement of the real-time as well. The key contributions of the proposed method are highlighted as follows:

- The extraction time of the image with resolution 128x128 in dataset Outex-TC is merely 85.64μs, and is more than 4800 times faster than in software [2].
- The improvement of the MRELBP_CI descriptor is proposed to avoid two-pass of the image, so that the MRELBP can be implemented in almost one-pass pipeline.
- To trade-off between computation efficiency and the performance, the simulation of the precision is conducted with 1-16 decimal digits on MATLAB. According to the simulation result, the precision with a double float-point is

ICCPR 2020, October 30-November 1, 2020, Xiamen, China

ACM ISBN 978-1-4503-8783-5/20/10…$15.00
https://doi.org/10.1145/3436369.3436371

not necessary in the MRELBP. The proposed architecture used 16 fractional bits in the fixed-point fraction, resulted in less than 1% difference to the accuracy.

The rest of this paper is organized as follows: Section II reviews the MRELBP algorithm and related concepts of the LBP. The proposed architecture implemented on FPGA is described in Section III. Experimental results are presented in Section IV, and Sections V concludes the paper.

2. MEDIAN ROBUST EXTENDED LOCAL BINARY PATTERN

2.1 Local Binary Pattern (LBP)

Given a center pixel x_c in the image, the LBP descriptor proposed by Ojala et al. [3] is calculated by comparing x_c with its neighbor pixels, which can be described as follows:

$$LBP_{r,p} = \sum_{i=0}^{p-1} s(x_i - x_c)2^i \quad s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{1}$$

Where $x_i (i = 0, \dots, p-1)$ is the neighbor pixel on the circle with radius r, and p is the total number of the neighbors. The neighbors that do not fall exactly in the center of pixels can be estimated by interpolation.

To make the LBP robust in terms of rotation variation, Ojala et al. proposed the rotation invariant LBP and the uniform LBP. In [3], the rotation invariant LBP is usually described as:

$$LBP_{r,p}^{ri} = \min\{ROR(LBP_{r,p}, i) \mid i = 0,1,\dots, p-1\} \tag{2}$$

where $ROR(x, i)$ performs i times circular bit-wise right shift on number x. $LBP_{r,p}^{ri}$ quantifies the occurrence statistics of the individual rotation invariant patterns corresponding to certain micro-features in the image. Further, Ojala et al. observed that certain patterns sometimes over 90 percent among all LBP patterns [3]. Thus, they introduced a uniformity measure U ("pattern"), which corresponds to the number of spatial transitions (bitwise 0/1 changes) in the pattern as follows:

$$U(LBP_{r,p}) = \sum_{i=1}^{p} | s(x_{\mathrm{mod}(i,p)} - x_c) - s(x_{i-1} - x_c) | \tag{3}$$

The patterns with $U > 2$ are grouped into one class named non-uniform pattern. Thus, the dimensionality of the LBP descriptor is reduced to $p(p-1)+3$ [3]. To further reduce the feature dimensionality and make the LBP rotation invariant, Ojala et al. proposed the rotation-invariant uniform LBP patterns defined as follows:

$$LBP_{r,p}^{riu2} = \begin{cases} \sum_{i=0}^{p-1} s(x_i - x_c), & \text{if } \mathrm{U}(LBP_{r,p}) \leq 2, \text{ uniform } LBP \\ p+1, & \text{otherwise} \end{cases} \tag{4}$$

Where U is defined in (3), $LBP_{r,p}^{riu2}$ groups the 2^p LBPs into $p(p+2)$ classes, leading to a significant dimensionality reduction.

2.2 MRELBP

To obtain both the micro texture and the macro texture information, and improve the performance and noise robust of the texture feature, Liu et al. proposed Median Robust Extended Local Binary Pattern (MRELBP) that combined MRELBP_CI, MRELBP_NI and MRELBP_RD descriptors [1]. The MRELBP is one kind of the RELBP which is improved from the ELBP [12]. The RELBP descriptor replaces a filter response $\phi()$ to an individual pixel in the ELBP. The joint histogram of RELBP_CI, RELBP_NI and RELBP_RD is used to represent a texture image, called RELBP. Given a pixel x_c and a patch filter $\phi()$, the RELBP_CI, RELBP_NI and RELBP_RD descriptors can be defined as follows [1]:

1) Center pixel representation

$$RELBP\_CI(x_c) = s(\phi(\mathbf{X}_{c,\omega}) - \mu_\omega) \quad s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{5}$$

Where $\phi(\mathrm{X}_{c,\omega})$ is the result of the filter $\phi()$ to $X_{c,\omega}$ which denotes the x in the local patch of the size $\omega \times \omega$ centered at the center pixel x_c, and μ_ω denotes the mean of $\phi(\mathrm{X}_{c,\omega})$ over the whole image.

2) Neighbor representation

$$\begin{aligned} RELBP\_NI_{r,p}(x_c) &= \sum_{n=0}^{p-1} s(\phi(\mathbf{X}_{r,p,\omega_r,n}) - \mu_{r,p,\omega_r})2^n \\ \mu_{r,p,\omega_r} &= \frac{1}{p}\sum_{n=0}^{p-1} \phi(\mathbf{X}_{r,p,\omega_r,n}) \end{aligned} \tag{6}$$

Where r represents the radius of the circle, p is the total number of the neighbors, and $\mathrm{X}_{r,p,\omega_r,n}$ denotes a local patch of size $\omega_r \times \omega_r$ centered at $x_{r,p,n}$, $\{x_{r,p,n}\}_{n=0}^{p}$ represents the circularly and evenly spaced neighbors of the center pixel x_c at radius r.

3) Radial difference representation

$$\begin{aligned} &RELBP\_RD_{r,r-1,p,\omega_r,\omega_{r-1}}(x_c) \\ &\quad = \sum_{n=0}^{p-1} s(\phi(\mathbf{X}_{r,p,\omega_r,n}) - \phi(\mathbf{X}_{r-1,p,\omega_{r-1},n}))2^n \end{aligned} \tag{7}$$

Where $\mathrm{X}_{r,p,\omega_r,n}$ is as same as the RELBP_NI, and the neighbors in the RELBP_NI and RELBP_RD that do not fall exactly in the center of pixels can be estimated by bilinear interpolation.

The overview of the proposed multiscale RELBP descriptor is illustrated in Fig. 1 [1].

After testing a large number of parameters, Liu et al. finally verified a set of parameters, which improved the performance of RELBP to a large extent [1]. Firstly, they tested different encoding schemes and proposed a novel approach called num to reduce the dimensionality. Although the encode scheme num shows the best performance, the authors continued using in subsequent experiments for consistency with other LBP variants. Also, in this paper, the encode scheme $riu2$ is used in later hardware implementation. Then, they compared three basic choices for filter $\phi()$, including Gaussian RELBP (GRELBP), averaging RELBP (ARELBP), and median RELBP (MRELBP). According to the result, both the GRELBP and ARELBP are linear filter, which exhibit sensitivity to noise, particularly salt-and-pepper and corrupted-pixel noise, thus the median filter is used as filter $\phi()$. They also compared a different number of the sample points in the single radius, different radius combination, and the different local patch size $\omega \times \omega$ for the median filter. In the experiment, they compared the number of the sample points (8,8,16,24) with a fixed $_{\mathrm{p=8}}$ in radius (1,3,5,7) respectively.

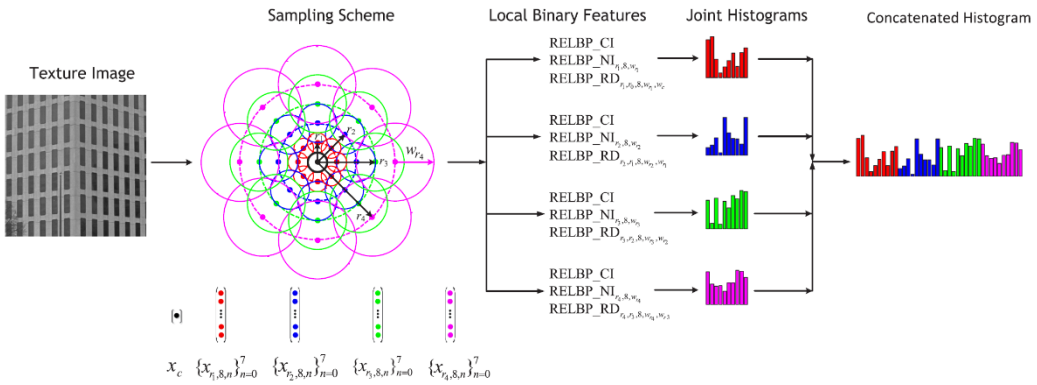

Figure 1. Overview of the proposed multiscale RELBP descriptor [1].

Although the former scheme showed improved performance for some individual descriptors, the joint descriptors of two schemes give a similar performance. The feature dimensionality of the proposed $RELBP_{r,p}^{riu2}$ in single radius is $2(p+2)(p+2)$, so the former scheme is 3552, while a fixed $p=8$ is only 800. Considering the similar performance, the authors propose to fix the number of sample $p=8$ at each radius. The joint histogram result also shows that the radius combination of (2,4,6,8) corresponds to the local patch size 3x3,5x5,7x7,9x9 respectively for median filter outperforms other schemes. In summary, the authors implement the MRELBP in software using the encode $riu2$ to be consistent with other algorithms, and fix sample neighbors $_{p=8}$ at multi-radius (2,4,6,8) which correspond to the local patch size 3x3,5x5,7x7,9x9 respectively.

3. HARDWARE IMPLEMENTATION

The process of the MRELBP as illustrated in Fig. 2, there are five main processes, including median, MRELBP_CI, interpolation, MERLBP_NI &MRELBP_RD, and histogram. To meet the real-time requirement and reduce resource utilization, it's better to implement the MRELBP in one-pass. However, the existing algorithm is not suitable for real-time situation, and needs to improve the degree of parallelism.

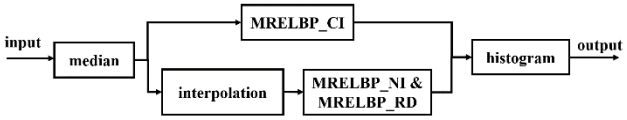

Figure 2. The process of the MRELBP.

3.1 Median Module

The median filter is the first module in the MRELBP, each pixel in the image needs to be calculated by the median module. The median filters of the proposed radius combination (2,4,6,8) correspond to the local patch size 3x3,5x5,7x7,9x9 respectively. The implementation of the 3x3 median filter is easy to meet real-time requirement, but the higher dimensionality 5x5, 7x7, and 9x9 median filter consume too much time to speed up the system. Yang et al. proposed a novel approach for implementing a 5×5 fast median filter using the merged insertion sorting algorithm, which just needs 90 times comparison [13]. The proposed algorithm as illustrated in Fig. 3, sort each column of pixels in the 5x5 filtered window in ascending order at first; Based on Fig. 3(b) sort each row of the pixels in ascending order; Then, based on Fig. 3(c) sort three elements in the 45-degree diagonal direction in ascending order according to the direction of the arrow; Finally, based on Fig. 3 (d) sort the three elements in ascending order, the median value is at the center of the 5x5 local patch.

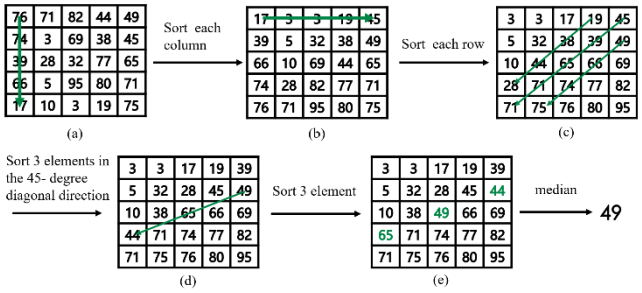

Figure 3. The 5x5 fast median filter algorithm.

The merged insertion sorting algorithm can be operated in pipeline. However, it just supports up to the 5x5 median filter. There is no suitable algorithm for the 7x7 and 9x9 median filter, thus the hardware efficient architecture to reduce the dimensionality of the median filter is proposed in this paper. The proposed implementation of the 7x7 median filter is illustrated in Fig. 4, and the steps can be described as follows:

Step1: Sort each column of pixels in the 7x7 filter window in ascending order;

Step2: Based on Fig. 4(b), sort each row in ascending order;

Step3: Based on Fig. 4(c), delete ten elements on the top left and bottom right respectively, which must not be the median value of the 7x7 median filter;

Step4: Based on Fig. 4(d), sort three elements in the upper left corner and three elements in the lower right corner separately, and then delete two smaller and two larger elements respectively;

Step5: Input to the 5x5 median filter;

The process of the 9x9 median filter is similar to the 7x7 median filter, reduce the dimensionality of the 9x9 median filter to 7x7 median filter and then to 5x5 median filter. The proposed approach can be implemented in pipeline and meet the real-time requirement. The 3x3, 5x5, 7x7, 9x9 median filter has been implemented in 4, 9, 15, 21 cycles respectively, and the clock frequency can be up to 200MHz on the Zynq-7045.

3.2 MRELBP_CI

The MRELBP_CI proposed by Liu et al. (5) needs to calculate the mean of the whole image, and then iterate through each data to get the MRELBP_CI descriptor, so the original approach is two-pass and it must buffer most of the whole image. Neither extraction time nor storage resources are efficient enough for real-time requirement in hardware.

However, both MRELBP_NI and MRELBP_RD descriptors are calculated across the local patch size $(2\times\text{radius}+1)^2$ centered at the center pixel x_c, while the MRELBP_CI calculating the mean of the whole image. The scope of three descriptors is different. Thus, modifying the scope of the MRELBP_CI to be consistent with the MRELBP_NI and MRELBP_RD is proposed, the MRELBP_CI can be described as follows:

$$MRELBP\_CI(x_c) = s(\phi(\mathbf{X}_{c,\omega}) - \mu_\omega) \quad s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (8)$$

where $\phi(\mathrm{X}_{c,\omega})$ is the median filter to the $\mathrm{X}_{c,\omega}$ which denotes the x in the local patch of the size $\omega \times \omega$ centered at the center pixel x_c, and μ_ω denotes the mean of $\phi(\mathrm{X}_{c,\omega})$ across the local patch size $(2\times\text{radius}+1)^2$ centered at the center pixel x_c.

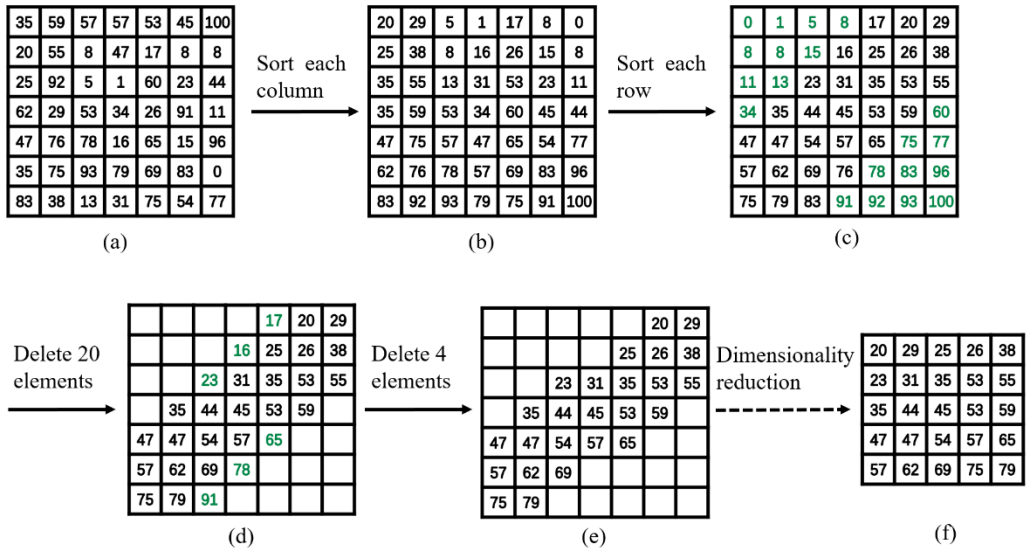

Figure 4. The 5x5 fast median filter algorithm.

Table 1. The information of texture dataset Outex-TC

Texture dataset	Texture classes	Sample size	Images per class	illumination	Rotation	Training images in total	Testing images in total	noise	bur	Other challenge
Outex_TC10	24	128 x 128	20	inca	00,05,10,15,30, 45,60,75,90	480	3840			
Outex_TC11bmp	24	128 x 128	20	inca	00	480	480			
Outex_TC11n	24	128 x 128	20	inca	00	480	480	Gaussian σ =5		11_bmp+noise
Outex_TC12bmp_000	24	128 x 128	20	Inca,horizon,TL84	00	480	960			
Outex_TC12bmp_002	24	128 x 128	20	Inca,horizon,TL84	00	480	480			
Outex_TC13lum	68	128 x 128	20	inca	00	680	680			
Outex_TC20	68	128 x 128	20	inca	00,05,10,15,30, 45,60,75,90	1360	10880			TC10 expand to 68 classes
Outex_TC21	68	128 x 128	20	inca	00	1360	1360			TC11 expand to 68 classes
Outex_TC22	68	128 x 128	20	inca	00	1360	1360	Gaussian σ =5		13+ noise
Outex_TC23	68	128 x 128	20	inca	00	1360	1360		Gaussian ρ =0.5	13+bur
Outex_TC24_000	68	128 x 128	20	Inca,horizon,TL84	00	1360	1360			TC12 expand to 68 classes
Outex_TC24_001	68	128 x 128	20	Inca,horizon,TL84	00	1360	680			TC12 expand to 68 classes
Outex_TC24_002	68	128 x 128	20	Inca,horizon,TL84	00	1360	680			TC12 expand to 68 classes

Table 2. The result (%) of the MRELBP of the individual descriptor

datasets	radius	Original MRELBP_CI		Proposed MRELBP_CI		Round the mean of the proposed MRELBP_CI	
		classification accuracy of the MRELBP	average	classification accuracy of the MRELBP	average	classification accuracy of the MRELBP	average
Outex_TC10	R2	100.000	99.648	100.000	99.694	100.000	99.694
	R4	98.906		99.141		99.089	
	R6	99.974		99.922		99.948	
	R8	99.714		99.714		99.740	
Outex_TC12_000	R2	99.896	99.818	100.000	99.870	100.000	99.922
	R4	99.688		99.792		99.896	
	R6	99.792		99.688		99.896	
	R8	99.896		100.000		99.896	
Outex_TC12_002	R2	95.625	95.469	95.833	95.573	95.833	95.677
	R4	95.208		95.417		95.625	
	R6	95.417		95.208		95.625	
	R8	95.625		95.833		95.625	

Table 3. The theoretical resource utilization of storage in interpolation module (W is the width of the image)

	Buffer median result (bit)	Without buffering median result (bit)
Buffer the 3x3 median result	4×W ×8	5×W ×8
Buffer the 5x5 median result	8×W ×8	0
Buffer the 7x7 median result	12×W ×8	0
Buffer the 9x9 median result	16×W ×8	0
Memory the interpolation result of radius=2	5×W ×4×(8+24)	5×W ×4×(8+24)
Memory the interpolation result of radius=4	5×W ×4×(8+24)	9×W ×4×(8+24)
Memory the interpolation result of radius=6	5×W ×4×(8+24)	13×W ×4×(8+24)
Memory the interpolation result of radius=8	5×W ×4×(8+24)	17×W ×4×(8+24)
total	2880W	5672W

The proposed approach can avoid iterating the whole image twice and reduce the storage resource in hardware. To compare the performance with the original MRELBP_CI, the experiment on the benchmark datasets Outex-TC10 and Outex-TC12 [16] has been conducted, the information of the dataset as summarized in Table 1. The source code provided by the original authors for texture extraction and the publicly available LIBSVM library [14] for SVM classification are used. To keep consistent, the linear kernel function and default parameter of the LIBSVM are set in all experiments in this paper. The individual descriptor's results as shown in Table 2, the mean of the local patch size $(2\times radius+1)^2$ is better than the whole image, thus the proposed approach is feasible.

3.3 The Storage Structure of Interpolation

After the median filter, MRELBP_NI and MRELBP_RD sample 8 points on each radius, and the neighbors in the MRELBP_NI and MRELBP_RD that do not fall exactly in the center of pixels need to be estimated by interpolation. However, the neighbors sample across the local patch size$(2\times radius+1)^2$, so buffering some lines of the median results or not becomes an issue. Two kinds of approaches are discussed the one with maximum resource efficient. 16-bit fixed fraction and 8-bit integer are used to implement interpolation module, the error and feasibility will be discussed later.

1) Buffering the median result

Buffering 4, 8, 12, 16 lines of the median results of radius 2, 4, 6, 8 respectively, until the whole local patch $(2\times radius+1)^2$ arrived to begin interpolation. The interpolation result of the inner radius needs to buffer five lines of result to align at the external one, due to the difference between the cycle of individual median filter.

2) Without buffering the median result

Without buffering the median result, the interpolation process is calculated in pipeline. However, the sample results before the last point in the local patch $(2\times radius+1)^2$ arrived need to be buffered, so it needs buffer at last 5, 9, 13, 17 lines of total eight sample points.

The BRAM resource on FPGA is used to buffer the results of the interpolation, the storage resource can be calculated as shown in Table 3. Although buffering the median results seems redundant, it just buffers the median results with 8-bit and avoids the storage of the large interpolation 24-bit results. Thus, implement the

interpolation with buffering the median results is more resource efficient than without buffering.

3.4 The Error of the Fixed Fraction

The fixed-point fraction is used to replace the double float-point fraction in software, which brings a randomly and unpredictable error when calculating the MRELBP descriptor. The error problem mainly focuses on the sign function with the average process. Take (8) as the example, the process of error problem can be described in Table 4. Because of the precision of the fixed fraction, some small errors are brought about when averaging, resulting in an opposite value in sign function. Further, the MRELBP descriptor is different.

The experiments on the benchmark datasets Outex-TC10 and Outex-TC12 is conducted to explore the effect of errors. First, the mean of the MRELBP_CI descriptor is round to the integer. The experiment result of the MATLAB code is shown in Table 2, comparing with the original proposed approach rounding the average value is better, so the average of the MRELBP_CI will be rounded in the later experiment. Moreover, the performance of different precision to the sign function in MRELBP_NI and MRELBP_RD is tested by fixing the precision from one to sixteen decimal fraction in the sign function and interpolation process. The performance of the joint histogram as shown in

Table 5, comparing with the Liu's original algorithm, the joint histogram of the proposed MRELBP_CI descriptor with the fixed-point fraction works better. Thus, although there is an error between double float-point fraction and fixed-point fraction in a single image, the classification performance on the whole dataset is very close. Weighing the computation efficiency and the performance, 16-bit fixed-point fraction is used to implement the MRELBP algorithm in hardware. (2^{16}=65536 , 1/65536=0.00001526)

Table 4. The process of the error problem

	double float-point fraction	4-bit decimal fraction
$\phi(X_{c,\omega})$	101	101
μ_ω	101.0000⋯0001	101.0000
Calculation process	$101 < 101.0000 \cdots 0001$	$101 \geq 101.0000$
$sign$ function	0	1

3.5 The Architecture of MRELBP in Hardware

After clarifying the problem above, the architecture of the MRELBP is implemented in parallel and pipeline as illustrated in Fig. 5. It is comprised of six regions: 1) median ready module; 1) median filter; 3) the MRELBP_CI module; 4) interpolation; 5) the MRELBP_NI and MRELBP_RD module; and 6) histogram module. Aiming to align the lines of four kinds of median filter, the first region is comprised of eight RAM-based shift registers. The second region is comprised of four median filters in parallel, which takes 4, 9, 15, 21 cycles to finish 3x3, 5x5, 7x7, 9x9 median filter in pipeline respectively. The third region is the MRELBP_CI module which contains four sets of the line-buffer, the size of the external radius impacts the number of line-buffer. The data of the fourth region is fetched in parallel from four median filter, and is also mostly implemented using thirty-three BRAMs and forty-four line-buffers. To meet real-time requirement, the whole interpolation module is in pipeline. The fifth region and the final region are dedicated to the calculation of the MRELBP_NI and MRELBP_RD descriptors and then output the histogram. The MRELBP architecture describes a fully unrolled implementation and allows for all radius to be processed in parallel, thus providing higher texture extraction speed.

Table 5. The classification result (%) of the MRELBP on MATLAB

dataset	Liu et al.' proposed MRELBP	Default precision in MATLAB	The Proposed MRELBP_CI						
			10e-1	10e-2	10e-3	10e-4	10e-5	10e-6	10e-7
Outex_TC10	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
Outex_TC12_000	99.896	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
Outex_TC12_002	95.625	95.833	95.833	95.833	95.833	95.833	95.833	95.833	95.833

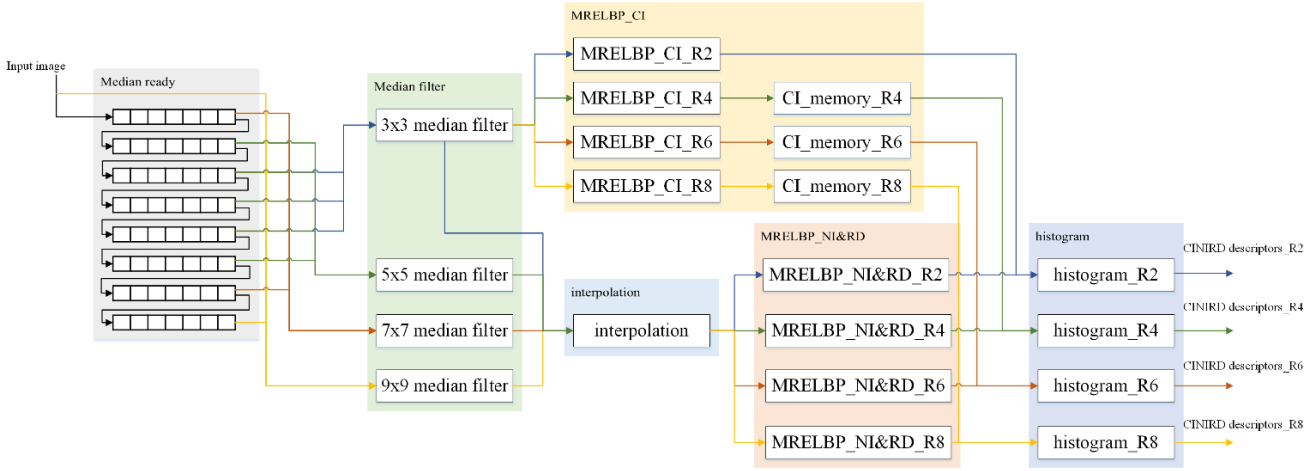

Figure 5. The structure diagram of the MRELBP extraction.

Table 8. Classification scores (%) of the proposed architecture in hardware

dataset	Liu's MRELBP algorithm on MATLAB					Proposed architecture in hardware				
	R2	R4R2	R6R4	R8R6	joint	R2	R4R2	R6R4	R8R6	joint
Outex_TC10	100.000	98.906	99.974	99.714	100.000	100.000	99.115	99.948	99.661	100.000
Outex_TC11bmp	98.542	99.792	99.167	98.542	100.000	99.167	99.583	97.708	97.292	100.000
Outex_TC11n	100.000	99.792	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
Outex_TC12bmp_000	99.896	99.688	99.792	99.896	99.896	100.000	99.896	99.896	100.000	100.000
Outex_TC12bmp_002	95.625	95.208	95.417	95.625	95.625	95.833	95.625	95.625	95.833	95.833
Outex_TC13lum	99.853	100.000	100.000	99.853	100.000	100.000	100.000	99.853	99.706	100.000
Outex_TC20	99.954	99.596	99.798	99.449	99.917	99.899	99.669	99.908	99.449	99.688
Outex_TC21	99.118	99.779	100.000	98.529	99.779	99.265	99.926	100.000	98.676	100.000
Outex_TC22	100.000	99.559	100.000	100.000	100.000	100.000	99.853	100.000	100.000	100.000
Outex_TC23	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
Outex_TC24_000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
Outex_TC24_001	100.000	100.000	100.000	99.926	100.000	100.000	100.000	100.000	99.926	100.000
Outex_TC24_002	100.000	100.000	100.000	99.963	100.000	100.000	100.000	100.000	99.963	100.000

4. FPGA IMPLEMENTATION RESULT AND PERFORMANCE EVALUATION

The proposed system is implemented on Xilinx Zynq-7045 FPGA with a speed grade −2. **Error! Not a valid bookmark self-reference.** shows the resources consumed by the proposed system. The LUT used by the proposed system is 49033, while the BRAM is 41.5 with 512 pixels as the max-width of the image. Because buffering the fixed lines data, the storage resource is linear with the width of the image. It is clear from **Error! Not a valid bookmark self-reference.** that MRELBP uses as low as 8% of the total available FPGA BRAMs resources.

Table 6. Resource consumption on Xilinx Zynq-7045 FPGA with a speed grade-2

Resource	Used	Available	Utilization
LUT	49033	218600	22%
FF	19969	437200	5%
BRAMs	41.5	545	8%

The proposed system has a linear relationship between extraction time and the size of the image, the cycles can be calculated as follows:

$$T_{cycle} = (h+4)w + 232 \quad (9)$$

The clock frequency can be up to 200MHz.

Table 7 shows the comparison result of extraction time with the MRELBP on MATLAB [2] and the LBP on FPGA [16]. Below are the descriptions of the table items:

- $C_{MRELBP,F}$: the extraction cycle of proposed architecture on FPGA
- $T_{MRELBP,F}$(ms) : the extraction time of proposed architecture on FPGA with 200MHz (ms)
- $T_{MRELBP,M}$(ms): the extraction time of the MRELBP on MATLAB [2] (ms)
- $ACC_{MRELBP,M}$: speed up $ACC_{MRELBP,M}=T_{MRELBP,M}/T_{MRELBP,F}$
- $T_{LBP,F}$(ms) : the extraction time of the LBP on FPGA [16] (ms)
- $ACC_{LBP,F}$: speed up $ACC_{LBP,F}=T_{LBP,F}/T_{MRELBP,F}$

Table 7. The comparison of the extraction time between the proposed system and related work

Image Size	128 x 128
$C_{MRELBP,F}$	17128
$T_{MRELBP,F}$(ms)	0.08564
$T_{MRELBP,M}$(ms)	416.6
$ACC_{MRELBP,M}$	4864.55
$T_{LBP,F}$(ms)	2.602
$ACC_{LBP,F}$	30.38

The extraction time from input the first pixel to output the last element of the histogram in the proposed system is 85.64μs. That means extracting the image of 1024x768 images can reach 250 frames per second. As can be seen in Table 7, the proposed architecture is more than 4800 times faster than in software. And comparing with the LBP on FPGA [16][16], although the MRELBP algorithm is more complex than the LBP, the proposed system speeds up more than 30 times.

The conducted experiment is on the benchmark datasets Outex-TC [14]. The simulation is conducted on Vivado 2018.3 for texture extraction and use the publicly available LIBSVM library for SVM classification on MATLAB [15]. Table 8 illustrates the proposed architecture in hardware based on the improved MRELBP has a similar performance with the original algorithm, even sometimes the performance is a little better. As the MRELBP in software, the MRELBP in hardware has a good performance on different rotation, luminance, and noise.

5. CONCLUSION

The hardware architecture based on the improved MRELBP has been proposed, which meets the real-time requirement for the texture extraction. The fixed-point fraction replaces the double float-point fraction in software without reducing accuracy, even it is a little better. The proposed architecture has attractive properties of strong discriminative, rotation invariance, noise robust, and computation efficiency. In the future work, we wish to applicate this proposed architecture in the entire classification system to implement the real-time extraction and classification, and make the architecture more resource efficient.

6. REFERENCES

[1] L. Liu, S. Lao, P. W. Fieguth, Y. Guo, X. Wang and M. Pietikäinen. 2016. Median Robust Extended Local Binary Pattern for Texture Classification. in *IEEE T Image Process*. vol. 25, no. 3, pp. 1368-1381.

[2] Liu L, Fieguth P, Guo Y, et al. 2016. Local Binary Features for Texture Classification: Taxonomy and Experimental Study[J]. *Pattern Recogn*. 62:135-160.

[3] T. Ojala, M. Pietikäinen, D. Harwood. 1996. A comparative study of texture measures with classification based on feature distributions. *Pattern Recogn*. 29(1) 51–59.

[4] T. Ojala, M. Pietikainen, and T. Maenpaa. 2002. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal*. vol. 24, no. 7, pp. 971–987.

[5] Z. Guo, L. Zhang and D. Zhang. 2010. A Completed Modeling of Local Binary Pattern Operator for Texture Classification. in *IEEE T Image Process*. vol. 19, no. 6, pp. 1657-1663.

[6] L. Liu, Y. Long, P. Fieguth, S. Lao, G. Zhao. 2014. Brint: binary rotation invariant and noise tolerant texture classification, in *IEEE T Image Process*. 23 (7),3071–3084.

[7] Z. Guo, X. Wang, J. Zhou and J. You. 2016.Robust Texture Image Representation by Scale Selective Local Binary Patterns. In *IEEE T Image Process*. vol. 25, no. 2, pp. 687-699.

[8] O. Mujahid, Z. Ullah, H. Mahmood and A. Hafeez. 2018. Fast Pattern Recognition Through an LBP Driven CAM on FPGA. in IEEE Access, vol. 6, pp. 39525-39531.

[9] C. Kyrkou, C. Bouganis, T. Theocharides and M. M. Polycarpou. 2016. Embedded Hardware-Efficient Real-Time Classification With Cascade Support Vector Machines. In *IEEE T Neural network*, vol. 27, no. 1, pp. 99-112.

[10] Y. Zhang, W. Cao and L. Wang. 2015. Implementation of high performance hardware architecture of face recognition algorithm based on local binary pattern on FPGA. *2015 IEEE 11th International Conference on ASIC (ASICON)* (Chengdu).pp. 1-4.

[11] T. Kryjak, M. Komorkiewicz and M. Gorgon. 2012. FPGA implementation of real-time head-shoulder detection using local binary patterns, SVM and foreground object detection. *Proceedings of the 2012 Conference on Design and Architectures for Signal and Image Processing* (Karlsruhe). pp. 1-8.

[12] L. Liu, L. Zhao, Y. Long, G. Kuang, and P. Fieguth. 2012. Extended local binary patterns for texture classification. In *Image Vis. Comput*. vol. 30, no. 2, pp. 86–99.

[13] K. Yang, M. Wei and L. Sun. 2018. Design of Median Filtering System Based on FPGA for Large Windows. *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*(Chongqing). pp. 78-82.

[14] T. Ojala, T. Mäenpää, M. Pietikäinen, J.K.J. Viertola, S. Huovinen. 2002. Outex—new framework for empirical evaluation of texture analysis algorithms. In *Proceedings of 16th International Conference on Pattern Recognition*. pp. 701–706.

[15] C.-C. Chang and C.-J. Lin. 2014. Library for Support Vector Machines. [Online]. Available: http://www.csie.ntu.edu.tw/-libsvm, accessed Dec 2014.

[16] Stekas N, Heuvel D V D. 2016. Face Recognition Using Local Binary Patterns Histograms (LBPH) on an FPGA-Based System on Chip (SoC)[C]. *IEEE International Parallel & Distributed Processing Symposium Workshops*.