Robust color image watermarking using multi-core Raspberry pi cluster

Khalid M. Hosny¹ • Amal Magdi¹ • Nabil A. Lashin¹ • Osama El-Komy¹ • Ahmad Salah²

Received: 22 December 2020 / Revised: 25 August 2021 / Accepted: 3 January 2022 /
Published online: 5 March 2022

© The Author(s) 2022

Abstract

Image authentication approaches have gotten a lot of interest recently as a way to safeguard transmitted images. Watermarking is one of the many ways used to protect transmitted images. Watermarking systems are pc-based that have limited portability that is difficult to use in harsh environments as military use. We employ embedded devices like Raspberry Pi to get around the PC's mobility limitations. Digital image watermarking technology is used to secure and ensure digital images' copyright by embedding hidden information that proves its copyright. In this article, the color images Parallel Robust watermarking algorithm using Quaternion Legendre-Fourier Moment (QLFM) in polar coordinates is implemented on Raspberry Pi (RPi) platform with parallel computing and C++ programming language. In the host image, a binary Arnold scrambled image is embedded. Watermarking algorithm is implemented and tested on Raspberry Pi model 4B. We can combine many Raspberry Pi's into a 'cluster' (many computers working together as one) for high-performance computation. Message Passing Interface (MPI) and OpenMP for parallel programming to accelerate the execution time for the color image watermarking algorithm implemented on the Raspberry Pi cluster.

Keywords Quaternion Legendre-Fourier moments · Image watermarking · Raspberry pi · Cluster · Parallel computing · Message passing interface · MPI · OpenMP

1 Introduction

Digital image watermarking technology secures sensitive digital images by embedding hidden information that proves these images' integrity [5]. Image watermarking has become a critical

✉ Khalid M. Hosny
k_hosny@yahoo.com

¹ Department of Information Technology, Zagazig University, Zagazig, Egypt

² Department of Computer Science, Zagazig University, Zagazig, Egypt

and essential image processing application due to the spread of the forged sensitive images over the internet. Proving the integrity and the copyright of the images is the watermarking role [28, 44].

The methods of invisible watermarking are mainly categorized into three groups: Fragile, Semi-Fragile, and Robust. Robust watermarking is used for copyright protection and ownership identification since it is intended to resist attacks attempting to erase or damage the mark. [33].

The robust image watermarking algorithms has three stages: Embedding, Detection, and Extraction of the watermark [9]. We embed a secret invisible signature in the host image in the embedding process. In the second step, we detected the hidden watermark information. Finally, if we could extract the invisible signature from the watermarked image, we are sure that the transmitted image is original. The correct extraction of the embedded watermark image from the watermarked host image reflects the utilized watermarking algorithm [27].

Natural images are color where color information plays a vital role in most computer-based systems; color image features extraction is the foundation of many applications of image processing as color image watermarking. Color images have more information than gray images. Traditional methods in color image watermarking ignore the correlation between color image channels. Quaternion orthogonal moments represent the color images holistically [14, 15].

Moment-based watermarking algorithms are successfully used in robust color image watermarking due to the quaternion moments' exact, fast, and computational stability. According to the extensive study [16], quaternion Legendre-Fourier moments outperformed the other quaternion moments in robustness against geometric, Scaling, Rotation, & Translation, and common attacks from signal processing [17].

Despite the high accuracy in the watermarking algorithms' CPU technology, the execution time for the watermarking algorithms in real-time applications is still an issue. The computations of color image moments and image reconstruction are intensive processes, mainly for higher moment orders and large image sizes. The key approach for solving the intensive computational problem is parallel computations in computing the image moments. Several attempts have been made to parallelize the image moment computations by using GPU, Multi-core CPU, and clusters of GPU [19, 20, 32, 45, 46]. For real-time applications, these approaches are not helpful.

Although many techniques were used to accelerate the watermarking algorithms, they still performed on ordinary P.C.s with limited portability because of their size and weight. The large size of the P.C.s prevents many smart-cities applications based on watermarking. To overcome the P.C. limits, we use embedded systems like Raspberry Pi, a small-portable platform. The significant advantage of Raspberry Pi is its small size and low cost compared to P.C. [34]. The Raspberry Pi is a single-board computer with many models with an open-source platform that can be used in many different projects [10]. Raspberry Pi model 4B has a Broadcom BCM2711, Quad-core CortexA72 (ARM v8) 1.5GHz processor with 1 G.B., 2 G.B., or 4GB LPDDR4-3200 SDRAM. Moreover, Raspberry Pi has an S.D. card slot for storage and I/O units such as USB, Ethernet, and HDMI. The Raspberry Pi low-cost single-board computer is used to reduce the complexity of systems in real-time applications [35].

Recently, much research work has been performed using the credit-card-sized open-source Raspberry Pi platform in various fields as image processing [22, 31, 39], the internet of things (IoT) [2, 4, 25], home automation [3, 21], and many other applications. Due to its portability, the user can control it via the internet, and the most important reason for its popularity among researchers is its low cost. When constructing massive supercomputing clusters, power

consumption has become an important metric. Instead of the more common high-end server CPUs, we use low-power embedded processors to minimize the power consumption in large clusters [6].

Due to its low cost and low power consumption, a single Raspberry Pi can be easily afforded, making it possible to create an in-expensive and energy-efficient cluster by combining several Raspberries into a cluster; we gain the ability to do more at once [1]. Many research works have been done using the Raspberry Pi platform in various fields as image processing, due to its portability, can be controlled via the internet and its low cost [30]. The Raspberry Pi is used in real-time applications to reduce system complications [36].

The need for portable, cheap, small, and powerful computers increased. A successful way to increase the computational power is by using parallel computing architecture such as OpenMP and Message Passing Interface (MPI) [26]. Raspberry Pi cluster has many advantages, such as its low cost and portability that is needed in smart cities applications, and running an open-source operating system on Raspberry pi makes it a preferred choice. The contribution of this paper are summarized as:

- A new parallel QLFM watermarking algorithm for color images is implemented on Raspberry Pi.
- The portable and small-sized embedded system (Raspberry Pi) is used to overcome the limited portability of the previous watermarking algorithms executed on P.C.
- The Message Passing Interface (MPI) and a cluster of four-nodes Raspberry pi are used to accelerate the watermarking processing to be suitable for real-time applications.

The rest of the article is arranged as follows: Section 2 provides a brief description of the color image watermarking algorithm using QLFM, Section 3 presents the implementation of the watermarked algorithm on Raspberry Pi, Section 4 presents the performance and experimental results of our watermarking scheme, and the conclusion is ultimately provided in Section 5.

2 QLFM color image watermarking

2.1 Quaternion color image

- The quaternion representation, as defined by Hamilton [11] as a generalization of complex numbers :

$$q = a + bi + cj + k \quad (1)$$

Where i, j, and k are imaginary units and a, b, c, and d are real numbers.

- Color image pure quaternion is defined as :

$$f(x, y) = f_R(x, y)i + f_G(x, y)j + f_B(x, y)k \quad (2)$$

The $f_R(x, y)$, $f_G(x, y)$ and $f_B(x, y)$ refer to the RGB components of the color pixels, respectively.

2.2 Host image

- The input host image of size $N \times N$ in Cartesian coordinate is first converted to the polar domain using cubic interpolation [41].
- The idea of polar conversion is splitting a unit disk into many circular rings (M); each circular ring is divided into several non-overlapping circular sectors (K_x) with the same area as the Cartesian image has uniform pixel size, as shown in Fig. 1 [18].
- The circular sectors number (K_x) in each i th ring and the Radial distance (R_x) of the x^{th} the circular ring is:

$$K_x = (2 * x + 1) * S, \quad R_x = \frac{(2 * x + 1)}{2 * M} \quad (3)$$

Where the value of M depends on the constraints $\frac{N}{2} \leq M \leq N$ where $N \times N$ is the color image size, the innermost circular ring is at $x=0$, and (S) is the number of the circular sectors in the innermost circle. The value of θ_{xy} can be calculated as:

$$\begin{aligned} &\text{For } x = 0 : M - 1 \\ &\quad K_x = (2 * x + 1) * S; \\ &\text{For } y = 0 : K_x - 1 \\ &\quad \theta_{xy} = 2 * \pi * (y + 0.5) / K_x; \\ &\quad \text{End} \\ &\text{End} \end{aligned} \quad (4)$$

Fig. 1 image mapping from Cartesian to polar coordinates

- Computation of QLFM in color images require calculations of LFM for each RGB channel $\hat{f}_R(r_x, \theta_{xy})$, $\hat{f}_G(r_x, \theta_{xy})$ and $\hat{f}_B(r_x, \theta_{xy})$, The LFM moments in polar coordinates :

$$M_{pq} = \frac{2p+1}{\pi} * \sum_y \hat{f}(r_x, \theta_{xy}) * L_{pq}(r_x, \theta_{xy}) \quad (5)$$

$$L_{pq}(r_x, \theta_{xy}) = I_p(r_x) * I_q(\theta_{xy}) \quad (6)$$

Where M_{pq} is LFM moment, $\hat{f}(r_x, \theta_{xy})$ is the interpolated image, $I_p(r_x)$ and $I_q(\theta_{xy})$ are the image kernels.

$$I_p(r_x) = \int_{U_x}^{U_{x+1}} \bar{P}_p(r) r \, dr \quad (7)$$

$$I_q(\theta_{xy}) = \frac{\hat{i}}{q} (e^{-iqV_{x,y+1}} - e^{-iqV_{x,y}}) \quad (8)$$

- Where the upper and lower limits of the radial integral are:

$$U_{x+1} = R_x + R_x/2U_x = R_x - R_x/2R_x = \frac{1}{M} \quad (9)$$

$$V_{x+1} = \theta_{xy} + \theta_{xy}/2V_{x+1} = \theta_{xy} - \theta_{xy}/2\theta_{xy} = \frac{2\pi}{K_x} \quad (10)$$

- And the Legendre Polynomial $-P_p(r)$ can be computed as follows:

$$-P_{p+1}(r) = \frac{2p+1}{p+1} (2r^2 - 1) - P_p(r) - \frac{p}{p+1} - P_{p-1}(r) \quad (11)$$

2.3 Binary watermark image

The binary 2D watermark image is first converted to a 1D sequence using Arnold scrambling algorithm ($b2(l)$) to embed this 1D sequence in the interpolated host image to improve the watermarking embedding process's robustness with the scrambling algorithm.

2.4 Watermark embedding

- After computing QLFM moments of host color image, we select suitable QLFM moments for embedding watermark that achieve these two factors: First, QLFM moments with $q=4$ m are ignored. Second, only the positive repetition q is selected to avoid information redundancy.

- After selecting the suitable moments, the binary sequence of watermark image $b2(l)$ is embedded in these selected moments by changing their magnitudes using the function of dither modulation [42] :

$$'M(l) = \left[\frac{M(l) - d_k(b2(l))}{\Delta} \right] * \Delta + d_k(b2(l)) \quad (12)$$

Where:

- $b2(l)$ is the scrambled binary watermark image
- $M(l)$ is the un-modified QLFM moment
- $'M(l)$ is the modified QLFM by dither modulation function
- Δ is the quantization step
- $d_k()$ is the dither function
- $d_k(0) \in (0, 1)$, $d_k(1) = \bar{2} + d_k(0)$

2.5 Color watermarked image reconstruction

The watermarked color image $f_w(r, \theta)$ can be represented with two components: the image from unchanged QLFM moments $f_{rem}(r, \theta)$ and the image formed from selected modified QLFM moments $f'_M(r, \theta)$ [14]:

$$f_{rem}(r, \theta) = f(r, \theta) - f_M(r, \theta) \quad (13)$$

$$f_w(r, \theta) = f_{rem}(r, \theta) + f'_M(r, \theta) \quad (14)$$

Where:

$$f_M(r, \theta) = \sum_{i=0}^{p*q-1} M_{p,q} * L_{p,q}(r, \theta) + M_{p,-q} * L_{p,-q}(r, \theta) \quad (15)$$

$$f'_M(r, \theta) = \sum_{i=0}^{p*q-1} M'_{p,q} * L_{p,q}(r, \theta) + M'_{p,-q} * L_{p,-q}(r, \theta) \quad (16)$$

$$f_M(r, \theta) = \sum_{i=0}^{p*q-1} M_{p,q} * L_{p,q}(r, \theta) + M_{p,-q} * L_{p,-q}(r, \theta) \quad (17)$$

2.6 Watermark extraction

To extract the binary watermark image from the watermarked image, we don't need the original image in the extraction process; we follow the same steps as the embedding process in selecting the QLFM moments, and with the same quantization step, we quantize $'M(l)$ moments, the binary image value is determined to be 0 or 1 by comparing the distance between $'M(l)$ and quantized $'M(l)$. the binary image is descrambled using the inverse Arnold scrambling algorithm to form the binary watermark image. All the steps of the QLFMs watermarking are summarized in Fig. 2.

3 Raspberry pi implementation

3.1 Single Raspberry pi

- Watermarking is an essential technology in the image processing field to secure the transmitted images over the internet. Because of the limited portability of P.C., embedded systems as Raspberry Pi are used.
- The Raspberry Pi has many models with an open-source platform. According to Raspberrypi.org, Raspberry Pi model 4B, as displayed in Fig. 3, has a Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 1.5GHz processor with a choice of ram between 1 G.B., 2 G.B., or 4 G.B. depending on how much RAM you need, and LPDDR4-3200 SDRAM. MicroSD card slot for the O.S. and data storage and I/O units such as USB, Gigabit Ethernet, Bluetooth, wireless networking, audio, RCA video, and 2 × micro-HDMI ports. The speed and performance of the new Raspberry Pi 4 is a step up from earlier models.
- The color image on Raspberry pi is first converted to the polar domain using cubic interpolation. Then the QLFM moment of the polar color image is calculated. The right QLFM moments are selected for the embedding process based on the two selection factors (positive repetition q and $q \neq 4$ m). Finally, embedding the scrambled binary watermark image in these calculated moments by modifying their magnitude using the dither modulation function.

Fig. 2 QLFM watermarking algorithm steps

Fig. 3 Raspberry pi model 4

3.2 Message Passing Interface and OpenMP

Message Passing Interface (MPI) is a library for writing parallel programs to accelerate the computing time using a cluster compared to a single computer; a cluster consists of at least two nodes. Message Passing Interface (MPI) is an interface that allows the head node (Master node) to distribute the computing task among all the slave nodes in the cluster in a parallel way as needed. The programmer must specify the number of processes in any Message Passing Interface (MPI) program. Each method has an identical program copy but executes only a specific part; the processes can send and receive data using MPI.

Message Passing Interface (MPI) allows parallel computation with the distributed memory where each node has its memory, and they are independent of each other. Therefore, MPI provides communications between the computing nodes. OpenMP is a programming extension for parallel computing. The memory is spread through all the computer systems. The program is divided into several parallelized processes that function on the memory that is shared. When the parallelized processes are finished, all the forked processes can quickly re-join the master process [40].

OpenMP is a C++ library that allows shared memory for multi-core and multi-processor programming. OpenMP also offers methods for synchronizing the running threads and schedule the tasks [8].

3.3 Raspberry pi cluster

- Because of its low cost, single Raspberry pi is easily afforded where it could be used to handle a light workload with low power consumption. To increase performance, we can use a cluster of Raspberry pi's [7].

- The Raspberry pi cluster has several advantages, including its portability due to its lightweight and small size, low cost, and low power consumption.
- For smart-cities watermarking application needs and the sake of portability, we can use Raspberry pi's cluster for watermarking purposes in a few seconds.
- Lightweight, portable clusters are useful in trains, planes, ships, and cars. A mobile cluster can be configured to continue running even though the failures of any cluster nodes. It would be difficult to fix the cluster in some instances, such as in military use.
- There is a continually increasing demand for small, portable, and low-cost, powerful computers that provide powerful computational power, increasing the computational power can be done by parallel computing, parallel computing on Raspberry pi cluster can achieve the computational power needs in applications where time is a critical metric in these applications, besides the low cost of the Raspberry pi cluster that makes it a better choice.
- By integrating many Raspberries as a cluster, we gain the ability to do more at once. A cluster consists of at least 2 nodes to facilitate parallel computing with distributed memory, and the Message Passing Interface (MPI) is required. MPI provides communication between nodes and distributes the computing task between them. The number of processes must be specified. Each will have the same copy of the program, but each node executes only a specific program part determined by the Master node in the cluster.
- The following items are required (as displayed in Fig. 4) to build our raspberry pi cluster:
 1. 4 Raspberry Pi's model 4
 2. S.D. cards for each Pi
 3. Power Cables for each Pi
 4. Networking Cables for each pi
 5. A Hub or a Router

Fig. 4 Raspberry pi cluster

Fig. 5 Color images used in testing the watermarking algorithm

- Our watermarking scheme is implemented on Raspberry pi model 4 using :
 1. C++ programming language
 2. OpenCV
 3. OpenMP
 4. MPI

4 Experiments results and discussion

Parallel robust QLFM color image watermarking algorithm using a C++ programming language is implemented on Raspberry Pi model 4 cluster, the cluster consists of 4 Raspberry pi's, the RPi's cluster degrades the time needed for the color watermarking algorithm, several experiments were done on a different number of RPi's cluster cores and different image sizes and moment order 40. The QLFM watermarking algorithm is tested on a different set of host color images, as displayed in Fig. 5.

4.1 QLFMS watermarking performance and numerical experiments

This section presents a set of numerical experiments to evaluate the performance of the QLFM watermarking algorithm. We evaluated the output in terms of visual imperceptibility and watermark robustness of the QLFM watermarking algorithm. The peak signal calculates the invisibility of the watermark to noise ratio (PSNR). In contrast, the robustness of the watermark is measured against multiple attacks using the bit error rate (BER) of the extracted watermark. The QLFM watermarking algorithm is tested on a different set of host color images and binary watermark images, as displayed in Figs. 5 and 6, respectively.

Fig. 6 Binary watermark images

Fig. 7 Reconstructed watermarked color image on Raspberry Pi

4.1.1 Watermark invisibility

The watermark invisibility of the watermarked image Fig. 7 is calculated in Table 1 to evaluate the algorithm performance using PSNR-peak signal to noise ratio on a color image of size 256×256 with quantization step ranges from [0.1 To 1], increasing the quantization step leads to decreasing the average PSNR [29]. SSIM-structural similarity image index is also calculated to measure the watermarked image quality and similarity [12, 37].

The PSNR can be calculated as follows:

$$PSNR = 10 \log_{10} \frac{255^2}{MSE} \quad (18)$$

Where MSE (mean square error) is defined as:

$$MSE = \frac{1}{N^2} \left(\sum_{i=1}^N \sum_{j=1}^N [f_w(i,j) - f(i,j)]^2 \right) \quad (19)$$

Table 1 The average SSIM and PSNR values

Δ	PSNR	SSIM
0.2	59.24	0.99999
0.4	53.51	0.9997
0.6	49.44	0.9993
0.8	46.99	0.9997
1	45.88	0.9994

Fig. 8 The extracted binary watermark under common attacks

Attacks		QLFM	BER
Rotation angle	10°		0.0361
	15°		0.0391
	20°		0.0352
	25°		0.0381
	30°		0.0313
	35°		0.0469
	40°		0.0391
	45°		0.038
Scaling factor	1.25		0.0059
	1.5		0.0068
	1.75		0.0068
	2		0.0127
Translation	(H 2, V 2)		0.0029
	(H 5, V 5)		0.002
JPEG compression	50		0.039
	70		0.024
	80		0.0068
	90		0.0029
Scaling 1.5 + Rotation 15°			0.0195
Scaling 1.5 + Rotation 25°			0.0244
Scaling 1.5 + JPEG compression 90			0.0088
Scaling 2 + JPEG compression 90			0.0137
Rotation 35° + JPEG compression 90			0.0518
Rotation 45° + JPEG compression 90			0.041

Order	QLFM		
20
40
60
80
100

Fig. 9 Reconstruction of QLFBMs watermarked color images

4.1.2 Watermark robustness

To evaluate the watermark robustness of the watermarking algorithm, the extracted watermark image ‘T’ with size 32×32 and its BER values are calculated in Fig. 8; each watermark is extracted under different kinds of attacks on the watermarked image (rotation, scaling, translation and JPEG compression). The experiments are done with image size 256×256 and order 40.

It is easy to notice that the extracted watermark images are very similar to the original binary watermark image as displayed in Fig. 8; these results prove the robustness of the QLFBM watermarking algorithm against the standard attacks.

The obtained BER values are approaching zeros which ensure the robustness of the watermarking algorithm.

Table 2 Average BER values of the watermarking methods for various attacks

Attacks			Proposed watermarking method	[43]	[24]	[38]
Rotation angle	5°	0		0.0052	0.006	0.0104
	15°	0		0.0112	0.0095	0.0104
	25°	0		0.0086	0.0138	0.0164
	35°	0		0.0069	0.0078	0.0095
	45°	0		0.0095	0.0156	0.0208
Scaling factor	0.25	0.2981		0.4383	0.302	0.4288
	0.5	0.0234		0.0546	0.0434	0.0885
	0.75	0.0039		0.0112	0.0049	0.0234
	1.25	0		0.0078	0.006	0.0069
	1.5	0		0.0112	0.006	0.0121
	1.75	0		0.0078	0.006	0.013
	2	0		0.0112	0.013	0.0138
	0	0.438		0.506	0.4557	0.4913
JPEG compression ratio	10	0.3326		0.3837	0.3406	0.3915
	20	0.0625		0.1753	0.1021	0.1267
	30	0.0234		0.0824	0.0369	0.0573
	40	0.0104		0.0312	0.0122	0.0173
	50	0.0049		0.0121	0.0086	0.0117
	60	0		0.0069	0.0086	0.0069
	70	0		0.0069	0.0086	0.0034
	80	0		0.0138	0.0086	0.0078
	90	0		0.0078	0.0112	0.0069

4.1.3 Bit Error Rate (BER)

The bit Error Rate (BER) metric is the ratio between wrongly extracted binary patterns and the length of the embedded binary bits. Therefore, the lower the BER, the greater efficiency of the embedding scheme [23]. BER of the watermark extracted bits measures watermark robustness

$$\text{BER} = \frac{\text{Berror}}{\text{nbits}} \quad (20)$$

Where *nbits* refer to the complete number of the watermarked image embedded bits, and *Berror* is the incorrectly extracted bits. Applying the BER function on our algorithm shows a similar watermark image extraction as the original binary watermark image embedded without extraction errors.

Table 3 The average PSNR and SSIM for quaternion watermarking methods with different values of quantization step (Δ)

Q.S.	Proposed watermarking method		[43]		[24]		[38]	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
0.2	59.24	0.9999	49.66	0.998	47.09	0.998	44.20	0.9922
0.4	53.51	0.9997	43.91	0.9918	41.10	0.9918	37.78	0.9707
0.6	49.44	0.9993	40.46	0.9811	37.10	0.9811	34.31	0.9474
0.8	46.99	0.9987	37.69	0.9692	34.96	0.9692	31.51	0.899
1	45.88	0.9984	35.82	0.9594	32.91	0.9594	29.81	0.8686

Table 4 Sequential profiling in seconds of QLFM watermarking with moment order 40 and image size 512×512 on Raspberry Pi model 4

Step	Execution Time in seconds	Weight%
Image interpolation	7.3	1.4%
Kernels computations (Ip & Iq)	9.65	1.85%
Moment computation	96.84	18.58%
Watermark embedding	0.001	0.0001%
Watermarked image reconstruction	407.5	78.17%
Total execution time	521.291	-----

4.1.4 Reconstruction accuracy

Numerical experiments were performed on color images to show the reconstruction capabilities for the QLFM watermarking scheme, as shown in Fig. 9. Image reconstruction is an excellent way to assess the utilized algorithm's accuracy, especially for higher-order moments.

4.1.5 Comparison of QLFMs and other quaternion moments

The efficiency of the QLFMs watermarking algorithm is compared to that of all other quaternion moments using numerical tests. After embedding and extracting a 128-bit watermark utilizing QLFMs and other quaternion moments, the average BER values are appropriately obtained, as shown in Table 2.

The average PSNR and SSIM are also computed, as depicted in Table 3. The comparisons explicitly reveal that QLFMs outperform all other quaternion moments. These observations and results convinced us to use this proposed method in our work.

4.2 Implementation of single Raspberry Pi

The first step of the watermarking algorithm parallelization is profiling the sequential time for the different stages of the algorithm to know the time-consuming stages that must be parallelized. The sequential execution time of each step in the QLFM watermarking algorithm on Raspberry pi is listed in Tables 4 and 5; these experiments were done using moment order 40 and color image size 512×512 and 256×256 and binary watermark image of size 32×32 .

Table 5 Sequential profiling in seconds of QLFM watermarking with moment order 40 and image size 256×256 on Raspberry Pi model 4

Step	Time in seconds	Weight%
Image interpolation	1.9	1.4%
Kernels computations (Ip & Iq)	3	2.28%
Moment computation	24.36	18.48%
Watermark embedding	0.001	0.0007%
Watermarked image reconstruction	102.58	77.83%
Total time	131.801	-----

Table 6 Parallelized QLFM watermarking profiling in seconds on 4 Nodes Raspberry Pi cluster for order 40 and image size 512×512

Step	Sequential time in seconds	4 Rpi cluster (16 cores) Time	4 Rpi cluster (8 cores) Time	4 Rpi cluster (4 cores) Time
Moment computation	96.84	6.4	12.9	25.65
Nodes' communication time Moment computation step	-----	0.18	0.25	0.46
Reconstructing the watermarked image	407.5	29.57	53.97	102.5
Nodes communication time in the image Reconstruction step	-----	4.7	5.6	6.1

4.3 Implementation on four nodes Raspberry pi cluster

As shown in Tables 4 and 5, The QLFM watermarking algorithm's two most intensive computational steps are the computation of the moment step and the watermarking image reconstruction step; these two steps can be parallelized using Message-passing Interface (MPI) on Raspberry pi cluster to accelerate execution time of the watermarking algorithm, these intensive computing steps are distributed into portions over the cluster nodes, and each portion on each node is parallelized over the number of cores available in the raspberry pi, each node execute only a specific part determined by the master node, each node has its memory, and they are independent of each other. MPI provides communications between the computing nodes; the nodes can send and receive data using MPI.

The parallelized execution time of intensive computing steps in the QLFM watermarking algorithm on the Raspberry pi cluster is listed in Tables 6 and 7. The authors performed experiments on different cluster cores moment order 40 and color image size 512×512 and 256×256 and binary watermark image of size 32×32 .

4.4 Speedup and improvement ratio

To measure the efficiency of the watermarking algorithm implementation on the Raspberry pi cluster, we calculate the speedup and the execution time improvement ratio.

1. The speedup is the metric of efficiency to determine how much faster parallel execution is versus serial performance. The algorithm's sequential runtime ratio to the time taken by the parallel algorithm to solve the same problem on (n) processors.

Table 7 Parallelized QLFM watermarking profiling in seconds on 4 Nodes Raspberry Pi cluster for order 40 and image size 256×256

Step	Sequential time in seconds	4 Rpi cluster (16 cores) Time	4 Rpi cluster (8 cores) Time	4 Rpi cluster (4 cores) Time
Moment computation	24.36	1.61	3.2	6.48
Nodes communication time Moment computation step	----	0.05	0.2	0.24
Reconstructing the watermarked image	102.58	7.44	13.55	25.83
Nodes communication time in the image Reconstruction step	-----	1.22	1.54	1.9

Table 8 Speedups and ETIR for the parallelized QLFM watermarking on 4 Nodes Raspberry Pi cluster for order 40 and image size 512×512

Step	4 Rpi cluster (16 cores) Time		4 Rpi cluster (8 cores) time		4 Rpi cluster (4 cores) Time	
	Speedup	Improv.	speedup	Improv.	speedup	Improv.
Moment computation	14.72	93.21%	7.36	86.42%	3.71	73.04%
Reconstructing the watermarked image	11.89	91.59%	6.84	85.38%	3.75	73.35%

$$speedup = \frac{T_{sequential}}{T_{parallel}} \quad (21)$$

The optimal speed up is 16x, 8x, and 4x for the 16, 8, and 4 core cluster clusters. Practically this value cannot be reached.

2. The execution time improvement ratio is the ratio of comparing the two execution times. It can be calculated as [13]:

$$ETIR = \frac{(T_{sequential} - T_{parallel})}{T_{sequential}} * 100 \quad (22)$$

Tables 8 and 9 show the speedup and ETIR of the parallelized QLFM watermarking algorithm on 4 raspberry pi cluster nodes. The speedup and ETIR are calculated for different cores with moment order 40 and color image size 512×512 and 256×256 .

Based on the speedups and ETIR listed in Tables 8 and 9, it's evident that parallelizing the QLFM watermarking algorithm is now much faster on Raspberry pi's cluster. For smart-cities watermarking application needs and the sake of portability, this technique is applicable. Since even after one of the cluster nodes fails, clusters can be configured to continue to run, we need this in some situations where repairing the cluster will be difficult.

Table 9 Speedups and ETIR for the parallelized QLFM watermarking on 4 Nodes Raspberry Pi cluster for order 40 and image size 256×256

Step	4 Rpi cluster (16 cores) Time		4 Rpi cluster (8 cores) time		4 Rpi cluster (4 cores) Time	
	Speedup	Improv.	speedup	Improv.	speedup	Improv.
Moment computation	14.68	93.19%	7.17	86.04%	3.63	72.4%
Reconstructing the watermarked image	11.85	91.56%	6.8	85.29%	3.7	72.97%

5 Conclusions

The transmission of sensitive images over the internet is still an issue, and a challenge against attackers—the limited portability of P.C. limits the smart-cities for new applications to be applied. Watermarking becomes an essential step in securing sensitive transmitted images using many algorithms. One of these robust watermarking schemes is quaternion Legendre Fourier moment (QLFM) for color images; detailed analysis and watermarking scheme steps have been presented. Searching for outstanding performance, lower power consumption, portability, and lower cost simultaneously, we can use a cluster of Raspberry pi's to perform the watermarking task in a few seconds. For smart-cities watermarking application needs and portability, this technique uses C++ programming language and parallel computing on Raspberry pi's cluster that shows excellent performance compared to ordinary expensive P.C. with low specifications to Raspberry pi specifications with its low cost and small size.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Abrahamsson P, Helmer S, Phaphoom N, Nicolodi L, Preda N, Miori L, Bugoloni S (2013) Affordable and energy-efficient cloud computing clusters: The Bolzano raspberry pi cloud cluster experiment. In 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, vol 2. IEEE, pp 170–175
2. Ahmad I, Pothuganti K (2020) Design & implementation of real-time autonomous car by using image processing & IoT. In 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE, pp 107–113
3. Akour M, Radaideh A, Shadaideh K, Okour O (2020) Mobile voice recognition based for smart home automation control. *Int J Adv Trends Comput Sci Eng* 9(3):3788–3792. <https://doi.org/10.30534/ijatcse/2020/196932020>
4. Atmaja AP, Hakim E, Wibowo A, Pratama LA (2020) Communication systems of smart agriculture based on wireless sensor networks in IoT. *J Robot Control (JRC)* 2(4):297–301
5. Begum M, Uddin MS (2020) Digital image watermarking techniques: a review. *Information* 11(2):110
6. Cloutier MF, Paradis C, Weaver VM (2016) A raspberry pi cluster instrumented for fine-grained power measurement. *Electronics* 5(4):61
7. Cox SJ, Cox JT, Boardman RP, Johnston SJ, Scott M, O'brien NS (2014) Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Comput* 17(2):349–358
8. Dagum L, Menon R (1998) OpenMP: an industry standard API for shared-memory programming. *IEEE Comput Sci Eng* 5(1):46–55
9. Darwish MM, Kamal ST, Hosny KM (2020) Improved color image watermarking using logistic maps and quaternion Legendre-Fourier moments. *Studies in Computational Intelligence*, vol 884. Springer, Berlin, pp 137–158. https://doi.org/10.1007/978-3-030-38700-6_6
10. Gupta S, Raikar U, Patil BMP, Molavade R (2018) Image processing based intelligent traffic control system by using Raspberry Pi. *Int J Res Appl Sci Eng Technol* 6(IV):66–70
11. Hamilton WR (1866) *Elements of quaternions*. Longmans, Green, & Company, London

12. Hore A, Ziou D (2010) Image quality metrics: PSNR vs. SSIM. In 2010 20th International Conference on Pattern Recognition. IEEE, pp 2366–2369
13. Hosny KM (2008) Fast computation of accurate Zernike Moments. *J Real-Time Image Proc* 3(1–2):97–107
14. Hosny KM, Darwish MM (2018) Robust color image watermarking using invariant quaternion Legendre–Fourier moments. *Multimed Tools Appl* 77(19):24727–24750
15. Hosny KM, Darwish MM (2019) Resilient color image watermarking using accurate quaternion radial substituted Chebyshev moments. *ACM Trans Multimed Comput Commun Appl (TOMM)* 15(2):1–25
16. Hosny KM, Darwish MM (2019) Performance analysis of invariant quaternion moments in color image watermarking. *Handbook of Multimedia Information Security: Techniques and Applications*. Springer, Berlin, pp 167–208. https://doi.org/10.1007/978-3-030-15887-3_8
17. Hosny KM, Darwish MM (2019) Invariant color images representation using accurate quaternion Legendre–Fourier moments. *Pattern Anal Appl* 22(3):1105–1122
18. Hosny KM, Shouman MA, Salam HMA (2011) Fast computation of orthogonal Fourier–Mellin moments in polar coordinates. *J Real-Time Image Proc* 6(2):73–80
19. Hosny KM, Darwish MM, Li K, Salah A (2018) Parallel multi-core CPU and GPU for fast and robust medical image watermarking. *IEEE Access* 6:77212–77225
20. Hosny KM, Salah A, Saleh HI, Sayed M (2019) Fast computation of 2D and 3D legendre moments using multi-core CPUs and GPU parallel architectures. *J Real-Time Image Process* 16(6):2027–2041
21. Iromini NA, Alimi TA (2020) Development of a Raspberry Pi secured management system for home automation. *i-Manager's. J Embed Syst* 8(2):1
22. Manikandan LC, Selvakumar RK, Nair SAH, Kumar KS (2020) Hardware implementation of fast bilateral filter and canny edge detector using Raspberry Pi for telemedicine applications. *Journal of Ambient Intelligence and Humanized Computing* 12(5):4689–4695
23. Mittal M, Kaushik R, Verma A, Kaur I, Goyal LM, Roy S, Kim TH (2020) Image watermarking in curvelet domain using edge surface blocks. *Symmetry* 12(5):822
24. Niu PP, Wang P, Liu YN, Yang HY, Wang XY (2016) Invariant color image watermarking approach using quaternion radial harmonic Fourier moments. *Multimed Tools Appl* 75(13):7655–7679
25. Petrović N, Kocić Đ (2020) IoT-based system for COVID-19 indoor safety monitoring. *ICETRAN Belgrade*
26. Puri S (2019) SpatialMPI: Message passing interface for GIS applications. *Geographic Information Science & Technology Body of Knowledge*. 2019(Q2)
27. Qasim AF, Meziane F, Aspin R (2018) Digital watermarking: Applicability for developing trust in medical imaging workflows state of the art review. *Comput Sci Rev* 27:45–60
28. Roy S, Rawat U, Sareen HA, Nayak SK (2020) IECA: an efficient IoT-friendly image encryption technique using programmable cellular automata. *Journal of Ambient Intelligence and Humanized Computing* 11(11): 5083–5102
29. Roy S, Shrivastava M, Pandey CV, Nayak SK, Rawat U (2020) IEVCA: An efficient image encryption technique for IoT applications using 2-D Von-Neumann cellular automata. *Multimedia Tools and Applications* 80(21):31529–31567
30. Sagar S, Choudhary U, Dwivedi R (2020) Smart home automation using IoT and Raspberry Pi. Available at SSRN 3568411
31. Sajjad M, Nasir M, Muhammad K, Khan S, Jan Z, Sangaiah AK, Baik SW (2020) Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities. *Futur Gener Comput Syst* 108:995–1007
32. Salah A, Li K, Hosny KM, Darwish MM, Tian Q (2020) Accelerated CPU-GPUs implementations for quaternion polar harmonic transform of color images. *Futur Gener Comput Syst* 107:368–382
33. Sathik MM, Sujatha SS (2010) An improved invisible watermarking technique for image authentication. *Int J Adv Sci Technol* 24:61–73
34. Senthilkumar G, Gopalakrishnan K, Kumar VS (2014) Embedded image capturing system using raspberry pi system. *Int J Emerg Trends Technol Comput Sci* 3(2):213–215
35. Shilpashree KS, Lokesh H, Shivkumar H (2015) Implementation of image processing on Raspberry Pi. *Int J Adv Res Comput Commun Eng* 4(5):199–202
36. Walter J, Fakh M, Grütner K (2014) Hardware-based real-time simulation on the raspberry pi. 2nd. Workshop on High performance and Real-time Embedded Systems
37. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. *IEEE Trans Image Process* 13(4):600–612
38. Wang X, Niu Y, Panpan W (2016) Quaternion exponent moments based robust color image watermarking. *J Comput Res Dev* 53(3):651
39. Widodo CE, Adi K, Gunadi I (2020) The use of raspberry pi as a portable medical image processing. In *Journal of Physics: Conference Series*, vol 1524, no 1. IOP Publishing, Bristol, p 012004

40. Wu Q, Spiryagin M, Cole C, McSweeney T (2020) Parallel computing in railway research. *Int J Rail Transp* 8(2):111–134
41. Xin Y, Pawlak M, Liao S (2007) Accurate computation of Zernike moments in polar coordinates. *IEEE Trans Image Process* 16(2):581–587
42. Xin Y, Liao S, Pawlak M (2007) Circularly orthogonal moments for geometrically robust image watermarking. *Pattern Recogn* 40(12):3740–3752
43. Yang HY, Wang XY, Niu PP, Wang AL (2015) Robust color image watermarking using geometric invariant quaternion polar harmonic transform. *ACM Trans Multimed Comput Commun Appl (TOMM)* 11(3):1–26
44. Yu X, Wang C, Zhou X (2017) Review on semi-fragile watermarking algorithms for content authentication of digital images. *Future Internet* 9(4):56
45. Yubo Xuan D, Li W, Han (2018) Efficient optimization approach for fast GPU computation of Zernike moments,. *J Parallel Distrib Comput* 111:104–114
46. Yang Z, Tang M, Li Z, Ren Z, Zhang Q (2020) GPU Accelerated polar harmonic transforms for feature extraction in ITS applications. *IEEE Access* 8:95099–95108

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.