An Efficient In-Memory Computing Architecture for Image Enhancement in AI Applications

MERIEM BETTAYEB¹, FAKHREDDINE ZAYER¹², HEBA ABUNAHLA¹, GABRIELE GIANINI³, 
(MEMBER, IEEE), AND BAKER MOHAMMAD.¹, (Senior Member, IEEE)

¹System on Chip Lab, Electrical Engineering and Computer Science Department, Khalifa University of Science and Technology, Abu Dhabi, UAE
²Center for Autonomous Robotic Systems (KUCARS), Khalifa University of Science and Technology, Abu Dhabi, UAE
³Università degli Studi di Milano and with the Emirates Innovation Center (EBTIC) at the Khalifa University of Science and Technology (e-mail: gabriele.gianini@unimi.it)

Corresponding author: Baker Mohammad (e-mail: baker.mohammad@ku.ac.ae).

This work was supported in part by Khalifa University CIRA project number 2018-026 and System on Chip Lab.

ABSTRACT Random Spray Retinex (RSR) is an effective image enhancement algorithm due to its effectiveness in improving image quality. However, the algorithm computing complexity and the required hardware resources and memory accesses hampered its deployment in many applications scenarios, for instance in IoT systems with limited hardware resources. With the rise of Artificial Intelligence (AI), the use of image enhancement has become essential to improve the performance for many emerging applications. In this paper, we propose the use of the RSR as a pre-processing filter before the task of semantic segmentation of low-quality urban road scenes. Using the publicly available Cityscapes dataset, we compare the performance of a pre-trained deep semantic segmentation network on dark noisy images and on RSR pre-processed images. Our findings confirm the effectiveness of RSR in improving segmentation accuracy. In addition, to address the computation complexity and suitability to edge devices, we propose a novel efficient implementation of the RSR using resistive random access memory (RRAM) technology. The architecture provides highly parallel analog in-memory computing (IMC) capabilities. A detailed, efficient and low latency implementation of the RSR using RRAM-CMOS technology is described. The design is verified using SPICE simulations with measured data from fabricated RRAM and 65nm CMOS technologies. The approach provided here represents an important step towards low-complexity and real-time hardware-friendly architecture and design for Retinex algorithms for edge devices.

INDEX TERMS Memristor Crossbar, Multiply and Add (MAC) Operations, Random Spray Retinex, Scale-to-Max Filtering, In-Memory Computing

I. INTRODUCTION

DIGITAL images captured in various application areas such as medical imaging, space exploration and underwater environment are often characterized by low-quality [1]. This can be due to insufficient lighting conditions or by the dynamic character of the environment. This is also the case for the images acquired for autonomous vehicle driving applications. An issue common to those images is that the difference in lightness between the bright and dark areas is large. This poses a challenge to subsequent processing tasks, such as image segmentation. Thus, a quality enhancement pre-processing step is normally added to the pipeline with the aim of increasing the overall effectiveness of the processing.

In this respect, one has to consider the tension between global and local aspects of the processing. Increasing the brightness of the overall image will improve the visibility of the dark areas, but reduce the visibility of the details in the bright areas. Several global techniques have been developed to overcome the low-light enhancement problem such as histogram equalization [2], gamma correction and tone mapping [3]. Nevertheless, these techniques result in a sort of over-exposure when the grey level in an image is concentrated at a certain intensity [4]. In addition, intensity saturation would result from inconsistent enhancement performed by these algorithms as they rely on global information from the image. This calls for the use of a differential treatment of the different image regions, based on both local and global
information.

To address these challenges, more complex algorithms have been developed, such as adaptive contrast enhancements [4], [5], adaptive histogram equalization [6], [7], and Spatial Color Algorithms (SCA) [8], [9]. SCAs are driven by Retinex principles, inspired to the behavior of the Human Visual System (HVS). These algorithms – RSR [10], STRESS [11], STAR [12], and ReMark [13], to mention a few that belong to the Milano Retinex family [14] – are widely employed to enhance real-world images. They implement two important characteristics of the human color vision system: (i) the independent analysis of the color components of the visual signal; (ii) the color adjustment based on local spatial and visual information. The algorithms of the Retinex family – originally created as a model of the Human Vision System – turned out to be endowed with several desired image enhancement properties [15].

These algorithms use both local and global information, and this results in efficient enhancement for the dark areas.

However, these algorithms are typically computationally intensive: this makes it hard to deploy them in real-time applications (especially on resource-limited edge devices) [16].

To address these challenges, this paper presents a novel low-complexity and real-time HW-friendly architecture and design of the Retinex algorithm. Furthermore, to the best of our knowledge, this work is the first to utilize the In-Memory Computing (IMC) feature of emerging memristor devices to reduce the power and improve the speed of the conventional Retinex algorithms.

Memristors are a type of Resistive RAM (RRAM) technology that provides low-power solutions at a low cost. A memristor contains a thin oxide film sandwiched between two metal electrodes [17] and has the ability to save information with zero leakage current, high endurance, relatively fast write time, and small cell size. Furthermore, the memristor has both storage and computing capabilities, which make it a suitable building block for IMC [18]–[20].

The new paradigm presented in this paper supports parallel computing and provides efficiency gains in area and power. It utilizes the analog computations in the memristor crossbar and uses the same physical elements for both processing and storage [18], [19], [21]. As a result, it substantially reduces the computing complexity resulted from the data-intensive Retinex algorithm.

As a case study, among the many Retinex variants, Random Spray Retinex (RSR) has been selected as a test for the innovative proposed architecture due to its proven high effectiveness in image enhancement [22], (we quantify the image quality using several quality assessment metrics). We use RSR as a pre-processing filter before the task of semantic segmentation of low-quality urban road scenes. Using the publicly available Cityscapes dataset [23], we compare the performance of a pre-trained deep semantic segmentation network on dark noisy images and on RSR pre-processed images. In addition, the image pixel accuracy is studied for different numbers of bits to understand the impact of this choice approach on accuracy and energy consumption. The latter study is essential for the implementation phase as it helps to identify the desired quantization level.

The main contributions of this paper are summarized as follows.

1) We implement an efficient RSR algorithm using emerging memristor technology.  
2) We use the RSR algorithm as pre-processing filter for image enhancement before applying image segmentation to demonstrate the effectiveness of such implementation in a practical setting.

The remainder of the paper is organized as follows. Section II presents a thorough background on Retinex algorithms. After that, the proposed architecture for memristor-based Retinex is described in Section III. Then, the simulation results for the proposed architecture are provided in Section IV. Finally, Section V presents the conclusions and the planned future work.

II. RELATED WORKS

A. THE RETINEX ALGORITHM FAMILY

Hereafter we frame the RSR algorithm within the family of Retinex algorithms and summarize its main variants.

Several Retinex algorithms have been developed throughout the years [24], based on the original Retinex, which was formulated in the 60’s and relied on random walks. This algorithm was defined by several mechanisms that specified the processing of the information collected by the random path and its progressive integration in the corrected output image. Among those mechanisms the most characteristic was the so-called reset, which allowed to refer the correction to maximum values found in the vicinity of the corrected pixel. The reset mechanism was preserved by several algorithms later developed in the Retinex family, among them the algorithms of the subfamily called Milano-Retinex [14], to which RSR belongs. For the sake of completeness we mention that this core mechanism was dropped by other simplified algorithms, among them the so-called NASA-Retinex [25]: this simplification made the processing more efficient, at the price of giving up to one of the distinctive characteristics of the Human Visual System.

The reset-based Retinex algorithms produce an enhanced image where the chromatic dominant of the light and any smooth gradients are partially suppressed [26], while scene details and edges are enhanced [27]. The Retinex algorithm performs a spatial color processing, i.e. it process colors based on their positions and combines these information by a specific equation aggregator. As for the spatial exploration, some use random walk processes [28], or their probabilistic representation [13], and some others use point sampling processes [29], [30] or their probabilistic representation [31]–[34]), while the aggregation can involve various kinds of averaging of the local intensity maxima.

RSR relies on a point sampling process, and was created by Provenzi and others [10] based on the observation that a point...
sampling process can explore the surrounding of a pixel in a less redundant way than a random walk. The pixel samples were dubbed sprays, hence the name Random Spray Retinex (RSR). In RSR, for each pixel to be corrected – the target – the algorithm generates several collections of informative pixels in the neighborhood, extracts from each collection the maximum intensity and eventually uses a suitable average of those values as a white reference for rescaling the input brightness of the target to compute the output value. The authors demonstrated that the spray technique outperforms the path-based strategy for neighbor information collection.

Later, Banić et al. [22] proposed Light RSR, which provided an efficient algorithmic implementation reducing the computational time while keeping the same spatial sampling method of RSR. Further study for the same authors in [35] studied the feasibility of using the RSR for global illumination estimation using the local RSR results. Moreover, a new light Random Spray Retinex-based image enhancement method was proposed by Banić and Lončaric [36]. It can be used as a color correction method, brightness adjustment method, or both. Although it operates locally, it performs a fixed number of operations per pixel, which means that its computational speed is almost independent of the parameter values used.

Later on, RSR was combined with the Automatic Color Equalization (ACE) algorithm. The two algorithms are complementary in their spatially variant approach. As a result the output images of the two algorithms exhibit complementary advantages and defects [37]. RSR shows good saturation properties, but has insufficient detail recovering capabilities. ACE, instead, has a propensity to put in evidence details, but it tends to wash out images.

Furthermore, Lecca et al. [38] modified the RSR algorithm to control the locality of the color filtering by considering the spatial image information. The spatial information is integrated into the RSR channel lightness computation at each pixel through a weighting function inversely proportional to the distance from the spray center. Finally, Tanaka et al. [39] produced two variants of RSR by concentrating on the region of interest (ROI): the first variant proposed a cone distribution based on anatomical data as a ROI, while the second variant focuses on the visual field information’s visual resolution and considers it an ROI. Among the most efficient and fastest implementations of RSR is FuzzyRSR [29], which exploits the same spray for the correction of several pixels.

B. REAL TIME HARDWARE IMPLEMENTATIONS

Some hardware implementations of Retinex algorithms have been proposed in the literature. For example, a digital signal processor (DSP)-based real-time realization of the NASA-Retinex algorithms (as we mentioned in the previous subsection, algorithms very different from RSR and much less demanding in terms of performance) – a Single-Scale Retinex algorithm applied to monochrome images, and a simplified version of the Multi-Scale Retinex with color restoration [25] – were proposed in [40], [41]. In this case, however, the system performances are significantly lower than the ones of the original, not simplified, algorithm [42], [43], furthermore the DSP solution itself is not suitable for edge devices due to the high power and cost of the hardware.

Conventional architectures show huge computational costs due to the required processing layers, arithmetic operations, and the number of iterations. In order to address this issue and increase the speed of these algorithms, the implementation of a hardware accelerator based on a field-programmable gate array (FPGA) was proposed in [42]–[45].

Li et al. [42] presents a completely parallel architecture based on FPGAs for the implementation of multi-scale Retinex in an outdoor application. Address encoding and distributed arithmetic are used to optimize the Gaussian kernel, and concurrent multi-scale convolutions are accomplished. Furthermore, ustukov et al. [44] modifies the multi-scale Retinex algorithm. The algorithm performance is improved by using different methods of picture blurring such as tabular value replacement instead of computing logarithm values. The method’s ability to combine algorithms allows it to be implemented on FPGA as a threading conveyor.

Park et al. [1] provided a concept for the retinex video enhancing method that is low-cost and high-throughput. The hardware (HW) architecture is built using an FPGA and has a throughput of 60 frames per second for a 1920 x 1080 picture with little delay. By employing a tiny line buffer instead of a frame buffer, using the notion of approximation computing for the complex Gaussian filter, and creating a novel and nontrivial exponentiation operation, the suggested FPGA architecture lowers HW resources while retaining quality and speed. Moreover, Masri et al. [45] suggested a flexible and effective architecture for real-time video frame augmentation that may be implemented in a single FPGA. The video enhancing algorithm is based on Retinex. To regulate the dynamic range of poorly lighted photos while keeping visual details, a novel illuminance estimate methodology was used. The video enhancement settings are regulated in real time by an inbuilt microprocessor, allowing the system to adapt to peculiarities of the incoming pictures and ambient lighting.

Nonetheless, FPGA has limited memory capacity and requires the image to be stored in external dynamic random-access memory (DRAM), which increases energy consumption and latency [42], [43], [45]. Furthermore, one has to consider the complex trade-off between performance, hardware (HW) resources, and efficiency degradation in terms of HW design. Li and Tsai [46] proposed the implementation of low-cost and high-speed HW for contrast-preserving and dynamic range compression. However, the Gaussian filter used in their algorithm can take only a small size.

Furthermore, Moore et al. [47] proposed a hardware implementation, which consists of resistive grids that average or smooth the pixel intensities. However, it lacks the RESET operation, which is the essential feature of the retinex theory. To address the challenges associated with the above-mentioned implementations, this paper presents a novel low-complexity and real-time HW-friendly architecture and de-
In this section, we report the study of the effects of illumination changes and contrast enhancement on the effectiveness of the semantic segmentation of urban road scenes. This case can be motivated by several settings, including the one of autonomous vehicles control. Autonomous vehicle applications need to operate correctly across different scenarios; however, environmental factors, such as weather and poor illumination, can deteriorate the quality of the acquired images, thus compromising safety.

Using the publicly available Cityscapes dataset, we simulate the underexposed images and compare the performance of a standard pre-trained deep semantic segmentation network on original and dark images. The Cityscapes Dataset focuses on semantic understanding of urban street scenes and it has 30 classes. These semantic objects contain: road, sidewalk, parking, rail track, person, rider, car, truck, bus, on-rails, motorcycle, bicycle, caravan, trailer, building, wall, fence, guard rail, bridge, tunnel, pole, pole group, traffic sign, traffic light, vegetation, terrain, sky, ground, dynamic, static.

The training and evaluation of our approach are carried out using a public large-scale Audi Autonomous Driving Dataset (A2D2) [48], which contains over 40,000 labeled images, from which we take a subset of 12,497 images with dimensions of 1920 × 1208 pixels. Training images are cropped and resized to the 384 × 384 pixels size, while test images are resized so that the largest dimension is 768 pixels with the original aspect ratio preserved.

The effect on the dark image is done using the approach proposed by Christopher et al. [49]. They used equation (1) to model the consequences of underexposure on each pixel in an image:

$$
V_2 = \begin{cases} 
\frac{V_1}{\theta_1/\theta_2} & \text{if } V_1 \leq \theta_1 \\
(1 - \theta_2) V_1 - \theta_2 & \text{if } V_1 > \theta_1 
\end{cases}
$$

The brightness values (i.e. the V channel when the picture is converted to HSV color space) of the relevant pixels in the original and changed images are thus $V_1(0, 1)$ and $V_2(0, 1)$, respectively.

$\theta_1$ is a randomly generated threshold for each picture such that $(\mu - \sigma) \leq \theta_1 \leq \mu$ is obtained, where $\mu$ is the mean and $\sigma$ the standard deviation of all pixel values V throughout the whole image;

$\theta_2$ is a second, lower threshold that controls the amount of compression applied to dark picture sections. For the selected dataset, $\theta_2$ is set to $\theta_2 = \theta_1 \times 0.1$ such that pixels with $V_1 < \theta_1$ yield a $V_2 = V_1 \times 0.1$. Essentially, the dynamic range of pixel values below the threshold is compressed, while the dynamic range of pixel values above the threshold is increased. Figure 1 shows a sample picture before and after the application of equation (1).

The pre-trained DeepLab v3+ architecture [50] is used in this paper to compare the segmentation performance between the original, dark, quantized, and enhanced images. As illustrated in Figure 5, the dark noisy images are passed...
through a random spray retinex algorithm to enhance the lightening of these images. Then, these enhanced images are passed to the pre-trained semantic segmentation model (DeepLab). Finally, an error is calculated as the mean pixels cross-entropy loss between the output of the segmentation model and the ground truth segmentation.

The Accuracy (Acc), Recall (Rec), Precision (Prec), and Jaccard metric (intersection over union, IoU) of the results are calculated compared to the ground truth data for the different classes presented in the dataset. To perform the comparison, the segmentation metrics shown in Table 1 are calculated for the original image, dark image, the enhanced image after applying RSR, and the quantized image with the RSR, which is the one used for the memristor implementation shown in the following sections. The accuracy value improved after applying RSR with n=3 and a=10, which is better than the dark simulated noisy pictures but not as perfect as the real image. For a broader image processing and computer vision community, we expanded the results in Table 1 with baseline techniques such as histogram equalization and some of its more advanced forms. Adjusting image intensity values, histogram equalization, and contrast-limited adaptive histogram equalization are three functions that are especially well suited for contrast enhancement. The three functions’ differences are reflected in Figure 2. The first method is adjusting the image intensity values or color map, which will boost the image’s contrast. By default, 1% of the data is saturated at low and high intensities of the input data. The second method is histogram equalization. It improves visual contrast by altering the values in an intensity image such that the output image’s histogram closely matches the desired histogram (uniform distribution by default). The contrast of each tile is increased such that the histogram of each output area comes close to matching the required histogram (uniform distribution by default). To prevent increasing any noise that may be present in the picture, the contrast enhancement might be minimized. Furthermore, Figure 3 shows how both n and a have a significant impact on the brightness adjustment.

Moreover, a test was performed on images from the ColorChecker image dataset [36], which contains 568 8-bit sRGB images, most of which have the size 874 × 583 as shown in Figure 4.

We add that the purpose of this subsection was not carrying on a comprehensive comparison of the performance of the RSR variants in terms of enhancement of the image quality: similar studies can be found in [27], [51] and in several papers devoted to variants of RSR such as [52]–[55].

### C. Typical Conventional Hardware Implementation for RSR

Most traditional versions of Retinex work offline, due to their computational complexity, although some versions of RSR, such as LightRSR [56] and FuzzyRSR [29] have reduced considerably the computational load, some, as SuPeR [57], at the price of some extra complexity in the code implementation. Some research has been carried on about the possibility of emulating Spatial Color Algorithms by learning the corresponding function by Artificial Neural Networks [58]: in those cases, the algorithm is fast and can be used online, however the time for training on a class of examples is non-negligible. We chose to use RSR instead of LightRSR, because the latter uses weights so that its implementation is slightly more involved than the former. The former is also more commonly adopted, this makes it easier to compare our results with other papers.

Conventional architectures show huge computational costs due to the number of iterations and the required processing layers and arithmetic operations. To the best of the authors’ knowledge, sparsity in Retinex is not yet employed in hardware. This comes with the fact that high-dimensional sparse data are usually beyond what is allowed on commodity hardware. The complexity of RSR processing is reported in terms of the number of scale to max operations, their accumulations over a given augmented spray, and the required memory resources.

Figure 6 presents the conventional architecture for RSR. The n masks represent the augmented random sprays generated from the input image by shifting (e.g., block A is a unique random spray for a given targeted pixel). This is equivalent to the use of image extensions with distributed arithmetic and convolution filters [42]. The multiple random sprays are generated following a pipeline data flow. FIFOs (First-In-First-Out) are used as line buffers for most FPGA-
FIGURE 2: Original image and enhanced images using adjusting image intensity values, histogram equalization, and contrast-limited adaptive histogram equalization example from cityscapes dataset.

(a) $n=1, a=0$  
(b) $n=1, a=0.2$  
(c) $n=1, a=0.5$  
(d) $n=1, a=1$  
(e) $n=1, a=5$  
(f) $n=1, a=2$  
(g) $n=5, a=2$  
(h) $n=10, a=2$  
(i) $n=15, a=2$  
(j) $n=20, a=2$

FIGURE 3: Examples of the RSR application for different values of $n$ and $a$ for an image from dark cityscapes dataset. Only a brightness adjustment has been performed.

(a)  
(b)  
(c)  
(d)  
(e)  
(f)  
(g)  
(h)  
(i)  
(j)

FIGURE 4: (a) original image from [36]. (b) application of guided image filtering for $n=4$ and $a=0.55$.

based image processing, which means data reading is a serial operation. Pipelining can be realized with address encoding based on the random generation method.

The address-encoding concept controls the address when the sprays from the image are serially read out from the storage. As presented in Figure 6, the $x \times y \times n$ pixel values are fed to the line buffers for pipeline data flow for comparison, max scaling, accumulation, averaging, and resampling with respect to the input image. The $x, y,$ and $n$ denote the rows, columns, and the number of pixel elements per spray, respectively. Given the methods mentioned above for HW-based retinex implementation, memristor-based in-memory computing paradigms are used to perform the scale to maximum operations and their accumulation as highlighted in Figure 6 (i.e. (b) and (c) blocks). The latter reduces memory access and computational complexity that serves the track for efficient hardware processing toward high intensive tasks with a high order of sparsity.

FIGURE 5: The training process for the RSR algorithm. First, an input image (a) is modified to simulate camera underexposure (c), then it is used as input to the RSR algorithm (e), which enhances the lightening of the image, then input that for a trained Deeplab segmentation model (f,d), which segments this enhanced image and the output is compared to the ground truth segmentation (b).

D. QUANTIZATION VS IMAGE QUALITY METRICS

Since memristor requires a limited number of states, the intensity needs to be quantized. To study the impact of quantization, we use seven different full-reference metrics to assess the image quality and compare it between the original image and the quantized image. The Matlab code for evaluating these metrics is provided in [59], which covers
the following metrics: Structural Content (SC), Mean Square Error (MSE), Peak Signal to Noise Ratio (PSNR in dB), Normalized Cross-Correlation (NKK), Average Difference (AD), Maximum Difference (MD) and Normalized Absolute Error (NAE).

The MSE is the most common metric used in the literature for assessing image quality as it is simple and does not involve costly computations [60], [61]. MSE works satisfactorily when distortion is mainly caused by contamination of additive noise [62]. Furthermore, another famous image quality metric used by the prior studies is the PSNR, which is the ratio between the maximum power of a signal to the maximum power of a noise signal [63], [64]. PSNR is measured according to peak signal power. PSNR involves simple calculations, has a clear physical meaning, and is convenient in the context of optimization. However, PSNR is not according to the characteristics of the human visual system (HVS) [62].

As shown in Table 2 and Table 3, as the quantization level increases, the PSNR increases, while the MSE and the NAE decrease. Moreover, the structural differences between reference and test images will generally increase as the quantization step size becomes larger. Hence, the AD is a monotonically decreasing function of the quantization step size, but the SC and MD are monotonically increasing functions of the quantization step size [65].

IV. MEMRISTOR-BASED IN-MEMORY COMPUTING ARCHITECTURE FOR RSR

In this section, the novel proposed hybrid CMOS-Memristor architecture for the random spray retinex algorithm is detailed. In addition, the experimental and simulation results for the proposed model are also revealed.

A. 4-BIT MEMRISTOR MODEL

The results provided in section II show that a 4-bit quantization is generally sufficient to achieve an acceptable image resolution, (see Table 2 and Table 3 for Figure 7 and Figure 8 respectively). In this work, an approximate 16-level fixed-point conductance state is adopted. This number is used to be consistent with the real memristor device fabricated by our group [21]. The writing process requires the memristive states to be separated within the switching window over the same interval.

As shown in Figure 9(a), a behavioral model is performed to fit the experimental gradual switching of the memristor. The resistance changes with a number of pulses \((v(t))\) are described with the following equations:

\[
F = R_{\text{max}} - F(1 - e^{\alpha(v(t)-v_{\text{max}})})
\]

(2)  

\[
F = R_{\text{max}} - \frac{R_{\text{min}}}{1 - e^{-\alpha v_{\text{max}}}}
\]

(3)

where \(R_{\text{max}} = 2800\Omega\), \(R_{\text{min}} = 157\Omega\) and \(v_{\text{max}} = 21\), represents, respectively, the maximum resistance, minimum resistance, and the maximum pulse number required to switch the device between the minimum and maximum resistance states. These parameters are directly extracted from the experimental data. \(\alpha = 1.5V^{-1}\) is the parameter that controls the nonlinear behavior of resistance update,
and $F$ is a function of $\alpha$ that fits the state transition within the range of $R_{max}$ and $R_{min}$. As shown in Figure 9(b), nonlinear change in device resistance can be obtained by tuning $\alpha$. This is equivalent to 4-bit characterization, which is used to code physical data acquired from the environment to be compatible with RRAM voltage pulse programming and efficient in-memory processing.

**B. LOGICAL MEMRISTOR-BASED IN-MEMORY COMPUTING ARCHITECTURE**

Realizing a practical memristor crossbar-based in-memory computing system usually requires the integration of multiple memristor crossbar arrays [66]. In general, splitting resistive states into different arrays is beneficial for parallel computing, increasingly needed with increasing system scales. A single memory computing macro-core of the proposed architecture is presented in Figure 10. The latter has the fundamental one–transistor–one-memristor (1T1R) topology [19], which enables reliable and uniform analog switching behaviors. With the proposed hybrid-processing scheme, the parallel processing is implemented as in [19], which reduces the latency by a factor of $n^2$. The input pixel values are encoded by the pulse number according to its quantized bit number. This allows direct writing of resistive values to the target cross-point memristors. Using the in-memory comput-
ing paradigm, the augmented sprays are used to realize the scale-to-max operation in terms of resistance evolution over time through the selected memristor cross-point cells with the application of the equivalent modulated pulse signal. Furthermore, memristor arrays are highly efficient in achieving parallel Multiply and Add (MAC) operations under shared inputs for different arithmetic values.

Different fetched batches of random sprays are passed into memristor cross-point cells separately by applying the input signals as described in the previous section. The training scheme for the memristor crossbar sets the constraint for which a batch of the intermediate signal will not be supplied as input until these constraints are met. Thus, the previous batch needs to be already used to calculate the desired weight updates. Furthermore, the corresponding memristor conductance needs to be already well-tuned to the greater modulated pixel amplitude. The desired max updates of the states with respect to the pulse sequence are stored in the accessed memristor element from the crossbar. Then, the second-row memristor conductance is updated after inputting the second input batch for the same-targeted pixel. During this processing stage, another cycle batches are inputted from another target pixel on the memristor crossbar and are fed into the unoccupied memristor-based-scale to a max operator in parallel [19]. These operations are repeatedly done through in-memory computation until the random paths are written in the memristor-based scale to max operators.

After inputting the encoded pulses that correspond to pixel intensity into the bit lines, the output currents through the source lines are sensed and accumulated. The current is the weighted sum corresponding to the input patch and the associated and predefined target pixel inputs. The total number of memory access to compute the required arithmetic operations for a given target pixel from a random path information in a traditional FPGA solution is calculated as follows:

\[ Mem_{acc} = Mem_{acc}(p_{row}) \times Mem_{acc}(p_{cols}) \times n \times N, \quad (4) \]

where \( p_{row} \) and \( p_{cols} \) denotes the access to x,y image coordinates from line buffers of Random pixel generation and pipelining block, as shown in Figure 6. \( n \) is the set-pixel-points per a single random path and \( N \) presents the processing cycles. Additional \( n \times N \) accesses to the scaled-to-max-values that are stored in the FIFOs in order to be accumulated, as well as \( n \times N \) for the averaging and resampling processes. While for the memristor-based IMC architecture, the \( n \) set-point are required to hold in-memory the given random spray and process the four-step algorithm of the targeted pixel in analog manner, as shown in Figure 10, leading to \( Mem_{acc}(p_{row}) \times Mem_{acc}(p_{cols}) \times N \) memory access process. In the following, RSR’s default values, \( n=250, N=25 \) are used for the objective assessments.

B. ARITHMETIC OPERATIONS

Based on Figure 6, using traditional FPGA implementations, the number of arithmetic operations needed for the processing of a given pixel, are:

\[ Ar.op = N.(n + 1)Comp(.) + n.N(Acc(.) + Av(.) + Resamp(.)) \quad (5) \]

Where \( Comp(.) \), \( Acc(.) \), \( Av(.) \), \( Resamp(.) \) are the comparators to obtain the augmented spray, accumulators, averaging and resampling operations.

While using the IMC paradigm, \( N.(n + 1) Comp(.) \) and \( Acc(.) \), are done in-memory intrinsic computations (i.e. only the access pulse train is required across its nodes).
FIGURE 10: In-memory computing macro core architecture of the simulated memristor-based Retinex processing unit and the relevant block modules.

The memristor-based IMC, instead of traditional FPGA implementation, shows a vast decrease in both: the number of memory accesses and arithmetic operations as shown in Table 4.

TABLE 4: Required resources and performance assessments for target pixel computations.

| Parameter          | Estimation with conventional design-based FPGA solutions | IMC solution                  |
|--------------------|---------------------------------------------------------|--------------------------------|
| Mem_{acc}           | Pixel generator: $164 \times 10^7$                     | Pixel voltage generator: $16.4 \times 10^7$ |
|                    | Scale to max operation: 6250                          | Scale to max operation: 25 |
|                    | Accumulation: 6250                                    | Accumulation: 25             |
|                    | Averaging and resampling: 6250                         | Averaging and resampling: 25 |
| Ar_{op}            | 20025                                                   | 12550                         |
| A(p^2)             | $750 \times 10^7$ (SRAM blocks)$+8 \times 10^3$ (other blocks) | $25+260$ (WL, BL switch matrices +WL decoders) |

VI. CONCLUSION

The paper presented a novel efficient hybrid CMOS-Memristor approach for random spray retinex. The proposed solution provided high speed and energy/area-efficient architecture compared to the conventional retinex scheme. Image quality was assessed using several image quality metrics and different quantization levels. 4-bit memristor was used as the computing element. The latter operated as a scale to maximum resistive values. MAC operations were performed in parallel for the accumulation of output currents. Generally, this design can be extended to other memristor-based in-memory computing systems that use the scale to max operators and employ sparse input data to boost their overall performance efficiently. The proposed approach is considered a great asset towards developing efficient memristor-based computer vision and deep learning applications. In addition, the usage of the RSR algorithm as pre-processing step for AI applications was investigated. The results showed improved accuracy when RSR was deployed on noisy images.

REFERENCES

[1] J. W. Park, H. Lee, B. Kim, D.-G. Kang, S. O. Jin, H. Kim, and H.-J. Lee, “A low-cost and high-throughput fpga implementation of the retinex algorithm for real-time video enhancement,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 28, no. 1, pp. 101–114, 2019.
[2] T. Arici, S. Dikbas, and Y. Altunbasak, “A histogram modification framework and its application for image contrast enhancement,” IEEE Transactions on image processing, vol. 18, no. 9, pp. 1921–1935, 2009.
[3] X. Guan, S. Jian, P. Hongda, Z. Zhiguo, and G. Haibin, “An image enhancement method based on gamma correction,” in 2009 Second International Symposium on Computational Intelligence and Design, vol. 1. IEEE, 2009, pp. 60–63.
[4] S.-C. Huang, F.-C. Cheng, and Y.-S. Chiu, “Efficient contrast enhancement using adaptive gamma correction with weighting distribution,” IEEE transactions on image processing, vol. 22, no. 3, pp. 1032–1041, 2012.
[5] D. Zhang, W.-J. Park, S.-J. Lee, K.-A. Choi, and S.-J. Ko, “Histogram partition based gamma correction for image contrast enhancement,” in 2012 IEEE 16th International Symposium on Consumer Electronics. IEEE, 2012, pp. 1–4.
[6] J.-Y. Kim, L.-S. Kim, and S.-H. Hwang, “An advanced contrast enhancement using partially overlapped sub-block histogram equalization,” IEEE transactions on circuits and systems for video technology, vol. 11, no. 4, pp. 475–484, 2001.
[7] T. K. Kim, J. K. Paik, and B. S. Kang, “Contrast enhancement system using spatially adaptive histogram equalization with temporal filtering,”
IEEE Transactions on Consumer Electronics, vol. 44, no. 1, pp. 82–87, 1998.

[8] A. Rizzi and J. J. McCann, “On the behavior of spatial models of color,” in Color Imaging XII: processing, hardcopy, and applications, vol. 6493. International Society for Optics and Photonics, 2007, p. 649302.

[9] M. Lecca, “A gradient-based spatial color algorithm for image contrast enhancement,” in International Conference on Image Analysis and Processing. Springer, 2019, pp. 93–103.

[10] E. Provenzi, M. Fierro, A. Rizzi et al., “Random spray retinex: a new retinex implementation to investigate the local properties of the model,” IEEE Transactions on Image Processing, vol. 16, no. 1, pp. 162–171, 2007.

[11] O. Kolás, I. Farup, and A. Rizzi, “Spatio-temporal retinex-inspired envelope with stochastic sampling: a framework for spatial color algorithms,” Journal of Imaging Science and Technology, vol. 55, no. 4, pp. 40503–1, 2011.

[12] M. Lecca, “Star: a segmentation-based approximation of point-based sampling milano retinex for color image enhancement,” IEEE Transactions on Image Processing, vol. 27, no. 12, pp. 5802–5812, 2018.

[13] G. Gianini, A. Rizzi, and E. Damiani, “A Retinex model based on absorbing Markov chains,” Information Sciences, vol. 327, pp. 149–174, 2016.

[14] A. Rizzi and C. Bonanomi, “Milano retinex family,” Journal of Electronic Imaging, vol. 26, no. 3, pp. 031207, 2017.

[15] B. Li, S. Wang, and Y. Geng, “Image enhancement based on retinex and lightness decomposition,” in 2011 18th IEEE International Conference on Image Processing. IEEE, 2011, pp. 3417–3420.

[16] S. Setty, N. Srinath, and M. Hanumantharaju, “Development of multiscale retinex algorithm for medical image enhancement based on multi-rate sampling,” in 2013 International Conference on Signal Processing, Image Processing & Pattern Recognition. IEEE, 2013, pp. 145–150.

[17] Y. Lu, A. Alvarez, C.-H. Kao, J.-S. Bow, S.-Y. Chen, and L.-W. Chen, “An electronic silicon-based memristor with a high switching uniformity,” Nature Electronics, vol. 2, no. 2, pp. 66–74, 2019.

[18] Y. Halawani, B. Mohammad, M. Al-Qutayri, and S. F. Al-Sarawi, “Memristor-based hardware accelerator for image compression,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 12, pp. 2749–2758, 2018.

[19] F. Zayer, B. Mohammad, H. Saleh, and G. Gianini, “Ram crossbar-based in-memory computation of anisotropic filters for image preprocessing,” IEEE Access, vol. 8, pp. 127569–127580, 2020.

[20] F. Zayer, W. Dghais, and H. Belgacem, “Modeling framework and comparison of memristive devices and associated stdp learning windows for neuromorphic applications,” Journal of Physics D: Applied Physics, vol. 52, no. 39, p. 393002, 2019.

[21] H. Abunahla, Y. Halawani, A. Alazzam, and B. Mohammad, “Neuromem: Analog graphene-based resistive memory for artificial neural networks,” Scientific Reports, vol. 10, no. 1, pp. 1–11, 2020.

[22] N. Banić and S. Lončarić, “Light random sprays retinex: exploiting the noisy illumination estimation,” IEEE Signal Processing Letters, vol. 20, no. 12, pp. 1240–1243, 2013.

[23] M. Cords, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset,” in CVPR Workshop on the Future of Datasets in Vision, vol. 2, 2015.

[24] J. J. McCann, “Retinex at 50: color theory and spatial algorithms, a review,” Journal of Electronic Imaging, vol. 26, no. 3, pp. 031204, 2017.

[25] D. Jobson, Z.-u. Rahman, and G. A. Woodell, “A multiscale retinex for bridging the gap between color images and the human observation of scenes,” IEEE Transactions on Image Processing, vol. 6, no. 7, pp. 965–976, 1997.

[26] G. Gianini, C. Mio, L. G. Fossi, and A. Rizzi, “Gradient attenuation as an emergent property of reset-based retinex models,” in Proceedings of the 11th International Conference on Management of Digital EcoSystems, 2019, pp. 324–329.

[27] M. Lecca, A. Rizzi, and G. Gianini, “Review and comparison of random spray retinex and of its variants stress and qbrix,” Cultura e Scienza del Colore-Color Culture and Science, vol. 9, pp. 55–64, 2018.

[28] ———, “Energy-driven path search for Termite Retinex,” Journal of Optical Society of America, A, vol. 33, p. 1, 2016.

[29] G. Gianini and A. Rizzi, “A fuzzy set approach to retinex,” in Proceedings of the CVPR Workshop on the Future of Datasets in Vision, vol. 2, 2015.

[30] G. Gianini and A. Rizzi, “A fuzzy set approach to retinex,” in Proceedings of the 11th International Conference on Management of Digital EcoSystems, 2019, pp. 324–329.

[31] E. Provenzi, M. Fierro, A. Rizzi, L. De Carli, D. Gadia, and D. Marini, “Random spray retinex and of its variants stress and qbrix,” Cultura e Scienza del Colore-Color Culture and Science, vol. 9, pp. 55–64, 2018.

[32] ———, “Energy-driven path search for Termite Retinex,” Journal of Optical Society of America, A, vol. 33, p. 1, 2016.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3171799, IEEE Access
N. Banić and S. Lončarić, “Using the random sprays retinex algorithm for global illumination estimation,” arXiv preprint arXiv:1310.0307, 2013.

N. Banić and S. Lončarić, “Smart light random memory sprays retinex: a fast retinex implementation for high-quality brightness adjustment and color correction,” Journal of the Optical Society of America, vol. 32, no. 11, pp. 2136–2147, 2015.

E. Provenzi, C. Gatta, M. Fierro, and A. Rizzi, “A spatially variant white-patch and gray-world method for color image enhancement driven by local contrast,” IEEE transactions on pattern analysis and machine intelligence, vol. 30, no. 10, pp. 1757–1770, 2008.

M. Lecca and A. Rizzi, “Tuning the locality of filtering with a spatially weighted implementation of random spray retinex,” Journal of the Optical Society of America, vol. 32, no. 10, pp. 1876–1887, 2015.

M. Tanaka, M. P. Lanao, T. Horiiuchi, and A. Rizzi, “Random spray retinex extensions considering region of interest and eyemovements,” Electronic Imaging, vol. 2020, no. 15, pp. 60–403–1, 2020.

G. D. Hines, Z.-u. Rahman, D. J. Jobson, and G. A. Woodell, “Dsp implementation of the retinex image enhancement algorithm,” in Visual Information Processing XIII, vol. 5438. International Society for Optics and Photonics, 2004, pp. 13–24.

G. D. Hines, Z.-u. Rahman, D. J. Jobson, G. A. Woodell, and S. D. Harrah, “Real-time enhanced vision system,” in Enhanced and Synthetic Vision, 2005, vol. 5802. International Society for Optics and Photonics, 2005, pp. 127–134.

Y. Li, H. Zhang, Y. You, and M. Sun, “A multi-scale retinex implementation on fpga for an outdoor application,” in 2011 4th International Congress on Image and Signal Processing, vol. 4. IEEE, 2011, pp. 1788–1792.

H. Tsutsui, H. Nakamura, R. Hashimoto, H. Okuhata, and T. Onoye, “An fpga implementation of real-time retinex video image enhancement,” in 2010 World Automation Congress. IEEE, 2010, pp. 1–6.

D. I. Ustukov, Y. R. Muratov, and V. N. Lantsov, “Modification of retinex algorithm and its stream implementation on fpga,” in 2017 6th Mediterranean Conference on Embedded Computing (MECO). IEEE, 2017, pp. 1–4.

S. Marsi and G. Ramponi, “A flexible fpga implementation for illumination–reflectance video enhancement,” Journal of real-time image processing, vol. 8, no. 1, pp. 81–93, 2013.

S. A. Li and C.-Y. Tsai, “Low-cost and high-speed hardware implementation of contrast-preserving image dynamic range compression for full-hd video enhancement,” IET Image Processing, vol. 9, no. 8, pp. 605–614, 2015.

A. Moore, J. Allman, and R. M. Goodman, “A real-time neural system for color constancy,” IEEE Transactions on Neural networks, vol. 2, no. 2, pp. 237–247, 1991.

J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühllegg, S. Dorn et al., “A2d2: Audi autonomous driving dataset,” arXiv preprint arXiv:2004.06320, 2020.

C. J. Holder, M. Khonji, and J. Dias, “Input optimisation network for semantic segmentation of underexposed images,” in 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, 2020, pp. 298–303.

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 4, pp. 834–848, 2017.

M. Lecca, G. Simone, C. Bonanomi, and A. Rizzi, “Point-based spatial colour sampling in milano-retinex: a survey,” IET Image Processing, vol. 12, no. 6, pp. 833–849, 2018.

K. Kočševec, V. Stipetić, E. Provenzi, N. Banić, M. Subašić, and S. Lončarić, “Hd-race: Spray-based local tone mapping operator,” in 2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA). IEEE, 2021, pp. 264–269.

N. Banić and S. Lončarić, “Color rabbit: Guiding the distance of local maximums in illumination estimation,” in 2014 19th International Conference on Digital Signal Processing. IEEE, 2014, pp. 345–350.

—, “Towards hardware-friendly retinex algorithms,” in Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis. IEEE, 2017, pp. 104–108.