IECON2015-Yokohama
November 9-12, 2015

A Maximally Stable Extremal Regions System-on-Chip For Real-Time Visual Surveillance

Ehab Salahat, Hani Saleh, Andrzej Sluzek, Mahmoud Al-Qutayri, Baker Mohammad and Mohammad Ismail
Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi, U.A.E.
ehab.salahat@ieee.org

***Abstract*— This paper presents a novel implementation of the Maximally Stable Extremal Regions (MSER) detector on system-on-chip (SoC) using 65nm CMOS technology. The novel SoC was developed following the Application Specific Integrated Circuit (ASIC) design flow which significantly enhanced its realization and fabrication and overall performance. The SoC has very low area requirement (around 0.05 mm^2) and is capable of detecting both bright and dark MSERs in a single run, while computing simultaneously their associated regions' moments, simplifying its interfacing with other image algorithms (e.g. SIFT and SURF). The novel MSER SoC is power-efficient (requires 2.25 mW) and memory-efficient as it saves more than 31% of the memory space reported in the state-of-the-art MSER implementation on FPGA, making it suitable for mobile devices. With 256×256 resolution and its operating frequency of 133 MHz, the SoC has a 2034 frames/second processing rate, making it suitable for time-critical real-time applications such as visual surveillance when integrated with other suitable algorithms in one system..**

Keywords—Maximally Stable Extremal Regions, Low power Union-Find, System-on-Chip, ASIC, Real-time.

I. INTRODUCTION

VISUAL SURVEILLANCE of dynamic scenes is an active area of research in robotics and computer vision. The research efforts are primarily directed towards object detection, recognition and tracking from a video stream. Intelligent visual surveillance has a wide spectrum of promising governmental and commercial oriented applications. Some important applications are in the field of security, given the ever-increasing demand in this domain, which includes, but not limited to, access control, crowd control, human detection and recognition, traffic analysis, vehicular tracking, Unmanned Aerial Vehicle (UAV) and detection of military targets [1]. The bottleneck of these applications is primarily hardware-related, including system's capability, scalability, memory requirements, power consumption, and the ability to interface various video formats. In fact, the issue of memory overhead prevents many systems from being built on hardware and mobile devices to achieve real-time performance, especially when general purpose processors are used. In these situations, the typical solutions are either to scale-down the resolution of the video frames or to inadequately process smaller regions of interests within the frame [2]. Digital Signal Processors provide improvement over general purpose processors due to the availability of optimized DSP libraries, however, DSPs still suffers from limited execution speeds that deems it insufficient for real-time applications. FPGA platforms, on the other hand, with their inherently parallel digital signal processing blocks, large numbers of embedded memory and registers and high-speed memory and storage interfaces offer an attractive solution to facilitate hardware realization of many image detection and recognition algorithms. Computationally-expensive algorithms are usually implemented on FPGA to utilize its capabilities. State-of-the-art developments in computer vision confirm that processing algorithms will make a substantial contribution to video analysis in the near future as the utilized algorithms might overcome most of the issues associated with the power- and memory-demanding needs, and might yield breakthroughs [2] [3]. Such intelligent computer vision systems demand novel system architectures capable of integrating and combining computer vision algorithms into configurable, scalable and transparent systems. These systems would require high performance devices. However, many research areas are yet to be properly addressed, for example, up to our knowledge, the Maximally Stable Extremal Regions (MSER) detector algorithm (which, once integrated with description/classification algorithms, can be used robustly used for real-time visual surveillance) has only a single reported hardware implementation attempt with limited success [4] [5], even though the MSER was introduced over than a decade ago.

In this paper, a novel implementation of the MSER detection algorithm is developed on SoC following the ASIC design flow using the 65nm CMOS technology. The SoC area is 0.05 mm^2 (memory-off chip) and it is capable of detecting both bright and dark MSERs in a single run, while computing, in parallel, their associated regions' moments, simplifying its interfacing with other image algorithms (e.g. SIFT [6] and SURF [7]). The novel MSER SoC is power-efficient (requires 2.25 mW) and memory-efficient as it saves more than 31% of the memory space reported in the state-of-the-art MSER implementation on FPGA. With 256×256 resolution and its operating frequency of 133 MHz, the SoC has a processing rate of 2034 frames/second, making it suitable for demanding real-time applications.

The remaining part of this paper is structured as follows. A brief overview of the MSER algorithm is presented in section II. In section III, the new MSER SoC architecture is presented in details and is accompanied with sample MSER detection results. The system's memory requirement and the number of read/write memory accesses and arithmetic operations are illustrated in section VI. Finally, the paper's findings and contributions are summarized in section V.

II. MAXIMALLY STABLE EXTREMAL REGIONS

The MSER algorithm can be informally described as follows. Assume that we have an M×N grid that corresponds to an M×N intensity image. Starting with a zero threshold, the threshold is increased by Δ increments and all pixels above or

(a) (b) (c)

Fig. 1: (a) The test image (b), its intensity and (c) inverted intensity images.

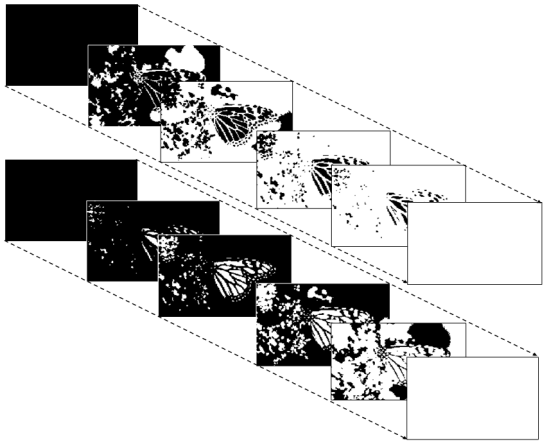


Fig. 2: Sample thresholding stages for the (a) intensity and (b) inverted intensity test images.

equal to the current threshold are painted white so that some black spots (regions) start showing, surrounded by white areas. If we keep increasing the threshold, new black regions appear, and the previous black regions grow, and possibly merge. Ultimately, when we reach the maximum intensity value, the whole image is black. During this process, we keep monitoring the size (cardinality) of each region, $|Q(t)|$, as a function of the threshold t [8]. An MSER is detected if $q(t)$ has a local minimum, where

$$q(t) = \big||Q(t+\Delta)| - |Q(t-\Delta)|\big|/|Q(t)|\,, \qquad (1)$$

where $|\cdot|$ denotes the cardinality. Those MSERs correspond to the minimal regions. For maximal regions, we first invert the intensity image and follow the same procedure. Fig. 1 illustrates a test image, its intensity and the inverted intensity versions. Fig. 2 illustrates sample thresholding stages of Fig. 1 (a) and (b), respectively.

III. System-on-Chip Architecture

The MSER SoC architecture is shown in Fig. 3, which follows few main stages. Note that the efficient union-find algorithm is used for the labeling task the image regions at each threshold. In the design, there are five main parameters that controls the detected MSERs, namely the maximum and minimum allowable number of pixels of the MSER, the maximum allowable growth rate specified by the stability function (1), the threshold increment and the nesting tolerance. Different choices of these parameters yield different detected MSERs. The first two control parameters, MinArea and MaxArea, are needed to exclude too small or too large MSERs, i.e., all detected MSERs must satisfy the condition:

$$\text{MinArea} \le Q \le \text{MaxArea}. \qquad (2)$$

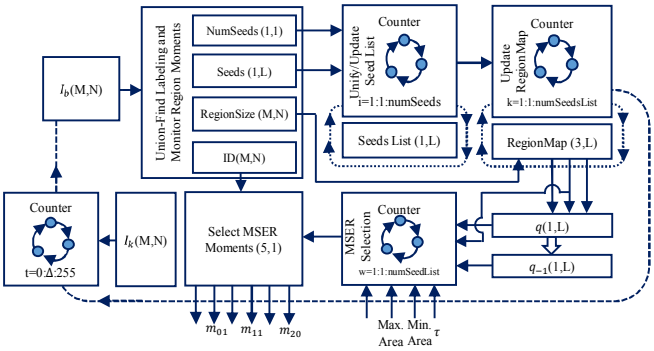


Fig. 3: High Level SoC Architecture of the MSER.

The third parameter, the Maximum Acceptable Growth Rate, specifies how stable the detected MSERs are, i.e., all detected MSERs must satisfy the condition:

$$q(t) = |Q(t+\Delta)\backslash Q(t-\Delta)|/|Q(t)| \le \text{MaxGrowth}. \qquad (3)$$

The fourth parameter, the nesting tolerance, is used to select the most stable MSER from a family of nested MSERs. Finally, the threshold increment, Δ, is selected in general between 4 to 7 to speed up the detection process (for example, an MSER detector with Δ set to 5 is nearly five times faster than that with Δ equals to 1. The details of the MSER SoC architecture design consists, therefore, of the following main stages (refer to Fig. 3).

A. Thresholding

The input intensity image (frame) is thresholded, starting with threshold of value zero with Δ increments up till 255, i.e., each frame requires $255/\Delta + 1$ thresholdings (e.g. for Δ equals to 5, 52 thresholding processes are required for each frame).

B. Labeling

The union-find algorithm (or [9]) can be used to label the binary image. The algorithm will output the labeled image, the seed and the size (the number of pixels with the same label) of each region, plus the number of labels used, respectively referred to as ID, Seeds, SeedsRS and NumSeeds.

C. Unifying/Updating Region Seeds

This is step is necessary for the system to work properly (even though not explicitly stated in [4] or [10]) due to the following rationale. The union-find algorithm returns labeled regions and their corresponding sizes and seeds. The seed of each region at this threshold is the first pixel location that the algorithm encounters of every region. Due to the threshold increments, previous regions might grow or even merge and new regions might appear. This means that the union-find will label those regions with labels, still unique but not necessarily similar to previous labels or with the same seeds. More importantly, since the regions grow/merge, the first pixel location that the union-find encounters for the growing region, i.e. its current seed, will be definitely different from the previous seed, even though both refer to the same region. This effect is shown in Fig. 4, where different labels are marked with different colors. To overcome this issue, all seeds that get

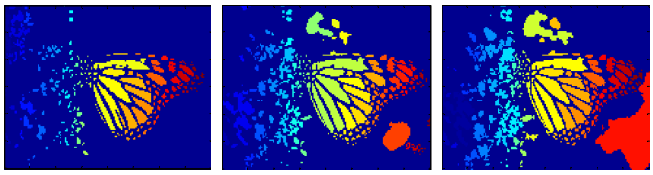


Fig. 4: Regions grow, labels and seeds change.

stored at this threshold, in the Seeds memory, are compared with the seeds previously detected and stored in the SeedList. This is simply done by comparing the labels, stored in ID, at the locations specified by the Seeds at the current threshold, and the stored SeedList. If a match is found, the old seed is maintained, otherwise a new seed is appended to the SeedsList.

D. RegionMap

This is a dedicated memory that is needed to store the seeds' region sizes, consisting of 3×#seeds stored in the SeedList registers, to store the value of $|Q(t+\Delta)|, |Q(t)|$ and $|Q(t-\Delta)|$ for each seed; the values are needed to calculate the stability function for each seed in the SeedList. If more seeds are appended to the SeedList at threshold $t+\Delta$, then new locations for this new seed are also appended to the RegionMap, where the region size for this threshold is added in the $|Q(t+\Delta)|$ while $|Q(t)|$ and $|Q(t-\Delta)|$ are filled with ones (to avoid division by zero). Note that since $|Q(t+\Delta)|$ is not available at the first two threshold values, then the calculation of (1) starts at the third threshold, i.e., $q(t)$ is calculated for threshold t at threshold $t+\Delta$. A sample RegionMap filling scenario is shown in Fig. 5.

SeedsList	Seed#1	Seed#2	Seed#3	Seed#4	blank
$\|Q(t-\Delta)\|$	25	49	102	4	blank
$\|Q(t)\|$	120	120	135	11	blank
$\|Q(t+\Delta)\|$	155	155	173	44	blank

(a) At the third threshold.

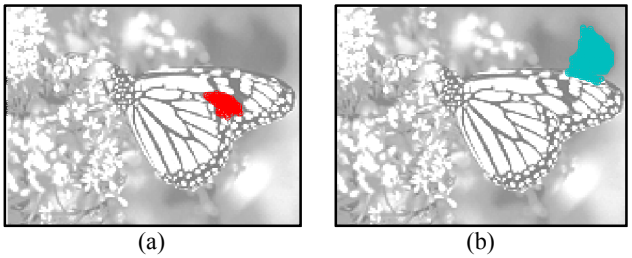
SeedsList	Seed#1	Seed#2	Seed#3	Seed#4	Seed#5
$\|Q(t-\Delta)\|$	120	120	135	11	1
$\|Q(t)\|$	155	155	173	44	1
$\|Q(t+\Delta)\|$	203	203	244	244	13

(b) At the fourth threshold

Fig. 5: Sample SeedList and RegionMap Scenario.

E. MSERs Selection

At this stage, using $q(t)$ previously calculated, in conjunction with $|Q(t)|$ stored in RegionMap, MSERs are selected to satisfy the conditions (2)-(3). Sample MSER detections are shown in Fig. 6.



(a) (b)

Fig. 6: Sample MSER detection: (a) bright, and (b) dark regions.

F. MSER Pixels, Moments and Ellipse Parameters

For every MSER that passes the condition in (2)-(3), we find the Pixels List, i.e., the x and y coordinates for the labeled region, stored in ID, and defined by its seed stored in the SeedList, and use these coordinate values to calculate the region moments as:

$$m_{pq} = \sum_{\{x,y\} \in R(\tau)} x^p y^q, \quad x, y \in \mathcal{R}(\tau) \qquad (4)$$

where x and y denote the pixel coordinates of the region $\mathcal{R}(\tau)$ at the current threshold. Subsequently, the region can be approximated by the best-fit ellipse. The ellipse equation is given by [11]:

$$\frac{(x-x_0+\tan(\alpha)(y-y_0))^2}{a^2(1+\tan^2(\alpha))} + \frac{(y-y_0+\tan(\alpha)(x-x_0))^2}{b^2(1+\tan^2(\alpha))} = 1, \qquad (5)$$

where (x_0, y_0), a, b, and α are the center of gravity (center of the ellipse), the major and minor axis lengths and the angle with respect to the horizontal axis. Those parameters can be calculated from the region moments $m_{00}, m_{01}, m_{10}, m_{11}, m_{02}$ and m_{20} as shown in [11]. Sample MSER ellipses are shown in Fig. 7 for test image. It's noteworthy that since the MSER algorithm is well-established, our main objective is the evaluation of the SoC and not the algorithm itself.

Fig. 7: (left) test image 1 and (right) its MSER ellipses.

IV. SYSTEM-ON-CHIP SPECIFICATIONS

Following the ASIC design flow, the SoC was synthesized, placed and routed using 65nm CMOS technology. The final chip-finished details are shown in Fig. 8. A summary of the reported SoC specifications are given in Table I, e.g. total chip area, operating frequency, processing rate, and required power, and are compared to the only reported MSER implementation on FPGA, given in [4]. Both our SoC and the FPGA are configurable, i.e. the MSER detection parameters can be modified. However, the new SoC is capable of detecting both bright and dark MSERs in one single run, which is unlike [4] where two runs are required (for bright and dark MSERs). Additionally, since our SoC computes the region moments of every detected MSER, the SoC can either display the regions as they are, or in terms of their moments (which can be used to calculate their associated MSER ellipses), unlike [4] that displays only the regions. Further, the maximum resolution that SoC supports in scalable (configurable), however, the maximum resolution in [4] for the FPGA 350×350 (or equivalently 122,500 pixels). More importantly, our SoC operating frequency is 133.33 MHz, which is more than three time faster than that in [4]. The most significant difference can be noticed in terms of the processing rate, given in frames/second (fps), which is 2034 fps vs. 54 fps reported in [4], making it an excellent candidate for real-time

applications. The required SoC area is 57,101 μm^2 (memory-off chip), with very high power efficiency low-power as it only requires 2.25 mW, which makes it suitable for mobile devices with limited power sources. In terms of the required memory space (storage), the SoC approximately requires:

$$\text{Memory Required} \approx [1.125+\lceil \log_2(\mathcal{R}) \rceil/4]\mathcal{R}, \quad \text{bytes} \quad (6)$$

where $\lceil \cdot \rceil$ denotes the *ceil* function and $\mathcal{R}$ denotes the image resolution (details are omitted due to space limitation, but we refer the reader to the memory blocks shown in Fig. 3). This memory requirement is significantly lower than reported in [4](Eq. 2), given in Table I for comparison (even though direct comparison between two different technologies is inadequate, however, we refer to it as our only reference for comparison). Fig. 9 compares the memory space of the SoC vs. the FPGA in [4], emphasizing that our SoC detects both bright and dark MSERs in one run, whereas [4] detects either but not both at the same run. Our memory savings reaches 31% at 256×256 resolution. This considerable memory saving is partially due to the superiority of the adopted ASIC design over FPGA, and partially due to the efficient SoC architecture. This SoC is planned to be fabricated in the middle of this year.

V. CONCLUSION

In this paper, a novel implementation of the MSER detector on SoC using 65nm CMOS technology is presented. The SoC was developed following the ASIC design flow which significantly enhanced its implementation. The SoC is capable of detecting both bright and dark MSERs in a single run, and is very power- and memory-efficient, outperforming the state-of-the-art FPGA implementation. With its configurable design, high operating frequency and processing rate, the SoC tends to be an excellent candidate for real-time visual surveillance applications and mobile devices.

ACKNOWLEDGEMENT

This work was supported by the Mubadala-SRC Center of Excellence for Energy Efficient Electronic Systems research contract 2013-HJ2440.

DISCLOSOURE

Two provisional U.S. patent applications have been filed on the contents of this paper.

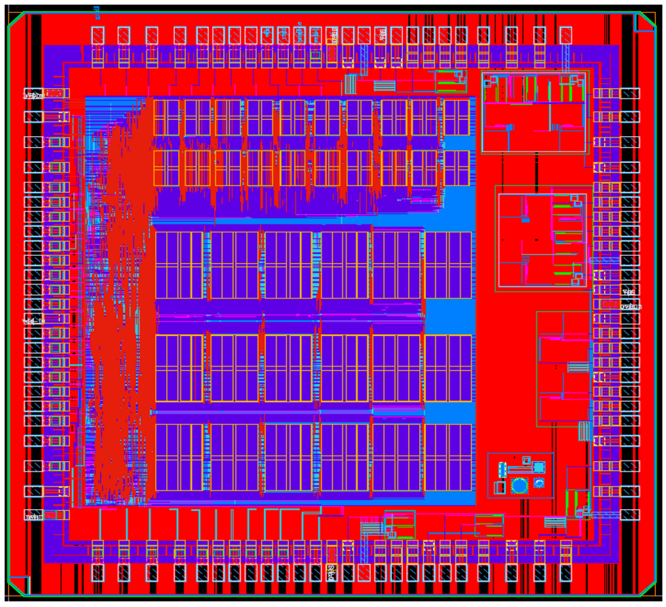


Fig. 8: The final synthesized, placed and routed MSER SoC.

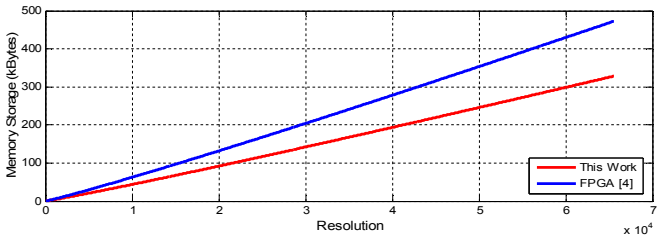


Fig. 9: The final synthesized, placed and routed MSER SoC.

TABLE I: OUR MSER SOC VS. [4] PERFORMANCE COMPARISON.

Performance Metric		This Work	FPGA [4]
MSER	Parameters	Configurable	Configurable
	Representations	Regions and Ellipse Moments	Regions
	Detection (per run)	Bright and Dark (in parallel)	Either Bright or Dark
Maximum Resolution		Scalable (Configurable)	122,500 pixels
Testing Resolution ($\mathcal{R}$)		65,536 pixels	76,800 pixels
Memory Storage (bytes)		$\approx[1.125+\lceil \log_2(\mathcal{R}) \rceil/4]\mathcal{R}$	$\approx[11+3\log_2(\mathcal{R})]\mathcal{R}/8$
Frame Rate (fps)		200 (expected)	54
Operating Frequency		133.33 MHz	42 MHz
Area		57,101 μm^2	N/A
Power		2.2578 mW	N/A

REFERENCES

[1] M. Valera and S. Velastin, "Intelligent Distributed Surveillance Systems: A Review," *IEEE Proceedings on Vision, Image, Signal Processing,* vol. 152, no. 2, pp. 192-204, April, 2005.

[2] H. T. Ngo, R. W. Ives, R. N. Rakvic and R. P. Broussard, "Real-time Video Surveillance on an Embedded, Programmable Platform," *Journal of Microprocessors and Microsystems,* vol. 37, no. 6-7, p. 562–571, August–October, 2013.

[3] H. Liu, S. Chen and Naoyuki Kubota, "Intelligent Video Systems and Analytics: A Survey," *IEEE Trans. on Industrial Informatics,* vol. 9, no. 3, pp. 1222-1233, August, 2013.

[4] F. Kristensen and W. J. MacLean, "Real-Time Extraction of Maximally Stable Extremal Regions on an FPGA," in *IEEE International Symposium on Circuits and System,*, New Orleans, LA, 27-30 May , 2007.

[5] E. Salahat, H. Saleh, B. Mohammad, M. Al-Qutayri, A. Sluzek and M. Ismail, "Automated Real-Time Video Surveillance Algorithms for SoC Implementation: A Survey," in *IEEE International Conference on Electronics, Circuits, and Systems*, Abu Dhabi, UAE, December, 2013.

[6] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[7] H. Bay, T. Tuytelaars and L. J. V. Gool, "SURF: Speeded Up Robust Features," in *9th European Conference on Computer Vision*, Graz, Austria, 7-13 May, 2006.

[8] E. Salahat, H. Saleh, S. Salahat, A. Sluzek, M. Al-Qutayri, B. Mohammed and M. Ismail, "Extended MSER Detection," in *The IEEE International Symposium on Industrial Electronics*, Rio de Janeiro, Brazil, 3-5 June 2015.

[9] E. Salahat, H. Saleh, A. Sluzek, M. Al-Qutayri, B. Mohammad and M. Ismail, "Novel Fast and Scalable Parallel Union-Find Implementation For Real-Time Digital Image Segmentation," in *Annual Conference of the IEEE Industrial Electronics Society*, Yokohama, Japan, 9 -12 November, 2015.

[10] J. Matas, O. Chum, M.Urban and T. Pajdla, "Robust Wide Baseline Stereo From Maximally Stable Extremal Regions," in *13th British Machine Vision Conference*, Cardiff, 2002.

[11] F. Chaumette, "Image Moments: A General and Useful Set of Features for Visual Servoing," *IEEE Trans. on Robotics,* vol. 20, no. 4, pp. 713-723, August, 2004.