International Journal of Applied Mathematics

Volume 28 No. 3 2015, 275-290
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
doi: http://dx.doi.org/10.12732/ijam.v28i3.7

A BIT-STREAM MORPHOLOGICAL IMAGE PROCESSING METHOD FOR PLANETARY IMAGES

Jean Adriel Silva Faria[1], Maurício Araújo Dias[2] §, Almir Olivette Artero[3], Erivaldo Antônio da Silva[4], Guilherme Pina Cardim[5]

[1,2,3,4,5]Faculty of Science and Technology
Univ. Estadual Paulista - UNESP (So Paulo State University)
Roberto Simonsen street, 305
Presidente Prudente, SP, 19060-900, BRAZIL

Abstract: Cartographers, in many ground control centers, need better computational solutions to process planetary images for cartographic mapping of some planets and natural satellites. In this paper, we propose a method that allows to apply a morphological image processing while receiving a planetary image by a bit-serial communication channel, starting the processing before the transmission of the image has finished. Thus, this paper presents a bit-stream morphological image processing method for planetary images. To demonstrate the method, we implemented a library of morphological operations in hardware, since hardware implementations speed up the bit-stream processing. The achieved results are presented and they show the relevance of this method. The success rates for the method are 100%, when borders are omitted. This method speeds up morphological processing of images transmitted through a bit-serial communication, for example, from a probe or landing module to a ground control center where cartographers are waiting to start cartographic mappings, but it is not limited to this use.

AMS Subject Classification: 03E99, 68U10, 86A30, 85-08
Key Words: mathematical morphology, image processing, method, planetary image, bit-stream, serial communication

Received: April 8, 2015

§Correspondence author

1. Introduction

Cartographic mapping of planetary surfaces is essential, for example, to decide where landing modules can perform safety landings. Probes that orbit some planets and natural satellites of our solar system acquire many new planetary images that will be processed according to the desired cartographic mapping. In ground control centers, cartographers need to wait until transmissions of planetary images (usually performed through a bit-serial communication channel from probes or landing modules, according to Nagy et al. [9]) have finished to start the image analysis or processing necessary to the cartographic mapping.

One of the most common tasks involving the processing of planetary images for cartographic mapping is the impact crater detection. Impact craters can be visually detected and manually highlighted by specialists, but it demands long time and effort. If the cartographic mapping needs to be fast, another way to do this processing is by automation based on digital image processing [8].

There are many kinds of implementations describing the processing of planetary images in the scientific literature. One of them is a software implementation that uses Mathematical Morphology (MM) [13], a very important concept of digital image processing used in many areas of human knowledge, to process planetary images [10]. The authors used specifically images acquired from Mars. They have created a routine that detects impact craters on Mars surface using MM.

Another software implementation uses an algorithm that extracts ellipsoidal features of planetary images [15]. It is based on a combination of techniques of image processing like Canny, Hough and Watershed. The authors extracted important curvy structures of the images, like rocks and craters.

In other software implementation, the authors created an algorithm to identify craters in high-resolution planetary images [16]. The authors used MM for crater detection. Scale and rotation-invariant filters were used to recognize crescent-like regions. A supervised machine learning algorithm was used to separate which parts of the image are craters and which ones are not craters.

Other paper describes a software implementation developed to recognize and detect impact craters in planetary images [3]. Each image is pre-processed (smoothing, edge detection and threshold). After that, the software extracts the features of the image with Hough transform. Finally, an unsupervised classification was applied to extract more features of the image.

In another paper, the authors created a new approach to detect impact craters in images of the Mars surface [2]. They use an algorithm that starts enhancing contours in the image. After that, the algorithm uses template match-

ing to detect impact craters. The borders of the craters are defined by using watershed transform.

The aforementioned papers show the intensive use of MM to process planetary images and the importance of software implementations for this kind of scientific research. However, hardware implementations achieve better performance than software ones. There are many implementations for image processing in hardware, such as [1], [14], [17], [18].

One possibility is a customized Application-Specific Integrated Circuits (ASIC) implementation, so we could get high-processing rates, but the disadvantage is that full-custom ASICs have the impossibility of reconfiguration. An alternative and efficient solution to this problem is the implementation of a dedicated hardware on a Field Programmable Gate Array (FPGA) [4], [5], [6]. The most important advantage of working with FPGAs is the capacity of reconfiguration. This way, a project can be easily reconfigured whenever necessary.

An example of implementation using FPGA, which performs image processing operations (erosion and dilation) is found in [1]. It was developed in three different hardware description languages (HDL): Handel-C, VHDL and Verilog. The authors have compared these three HDL and noticed that the most efficient was Handel-C for basic processing operations. Unfortunately, they did not implement operations like opening and closing.

Another implementation is presented by Tickle et al. [14]. This implementation uses FPGA to perform some image processing operations, like erosion, dilation, opening and closing. In this case, complete Digital Signal Processing (DSP) blocks were used and then they were transformed in VHDL code. The simulation was relatively slow, thus this implementation is unable to accelerate the processing of planetary images.

Despite of the previous related papers describing different software and hardware implementations to process images, cartographers remain waiting for new computational solutions to process planetary images for cartographic mappings faster than the already proposed implementations. In this paper, we propose a method that allows to apply a morphological image processing while receiving a planetary image by a bit-serial communication channel, starting the processing before the transmission of the image has finished. Thus, this paper presents a bit-stream morphological image processing method for planetary images.

To demonstrate the method, we implemented a library of morphological operations in hardware, since hardware implementations speed up the bit-stream processing. This library is composed by erosion, dilation, opening and closing

operations and it is implemented on FPGA using electronic schematics. Our method speeds up morphological processing of images transmitted through a bit-serial communication, for example, from a probe or landing module to a ground control center, where cartographers are waiting to start cartographic mappings, such as the mapping of impact craters on planetary surfaces.

To clarify our proposal, this paper is organized as follows. Section 2 presents the theoretical fundamentals. Section 3 describes the materials and methods used to develop our library. Section 4 comments the performed experiments. The results are shown in Section 5. Section 6 shows some discussions and conclusions about this library.

2. Theoretical Fundamentals

Mathematical Morphology (MM) is used to extract components of an image to represent and describe the shape of determined part of the image, like an skeleton, a segmentation or an edge. Set theory is the basis of MM concepts. An image is a set of pixels that are organized in a two-dimensional structure. It's possible to perform a lot of operations over any set of pixels of an image by using a structuring element (SE) to get or manipulate information related to the shape of this set. In this work, MM is applied to binary images. The pixels of a binary image is composed by two values: one (1) for the foreground and zero (0) for the background.

MM is based on convolution, which is an operation performed by probing a SE over a main image or, in mathematical terms, convolution is an operation that involves two matrix. One of them is the main (original) image (matrix A) and the other is the SE (matrix B). The SE can represent any function and will be applied to every pixel of the main image. When the SE passes across the main image, a new image (matrix C), which is the resulting image from the convolution, is created. In summary, a main image is probed by a smaller image (SE), which has to fit within the foreground of the main image to generate the result that is determined by the form and the size of the SE.

2.1. Erosion

Considering A as a main image, x as a point in the same image and B as a smaller image (SE): The translation of B by x is

$$B_x = \{b + x : b \in B\}; \tag{1}$$

The erosion of A by B is defined by

$$A \ominus B = \{x : B_x \subset A\}. \tag{2}$$

Erosion is the most basic operation of MM [7]. Erosion shrinks foreground structures of the image every time the SE fits in the foreground structures while the convolution is being performed. The resulting image after the erosion is always a subset of the main image.

Binary erosion most common effects are: decreasing of the foreground in an image, elimination of all foreground structures of the image smaller than the SE, separation of near objects, increasing of the voids.

2.2. Dilation

According to Dougherty and Lotufo [7], considering A as a main image, x as a point in the same image, B as a smaller image (SE), and A^c as the complementation of the set A, the dilation of A by B is

$$A \oplus B = (A^c \ominus \breve{B})^c, \tag{3}$$

where:

$$\breve{B} = \{-b : b \in B\}. \tag{4}$$

Binary dilation most common effects are: increasing of the foreground in an image, and elimination of all holes smaller than the SE in the foreground.

2.3. Opening

The opening of a binary image A by B (the SE) is defined by

$$A \circ B = (A \ominus B) \oplus B. \tag{5}$$

According to Dougherty and Lotufo [7], opening is an erosion immediately followed by a dilation using the same SE. Opening is frequently used to remove salt noise of an image.

2.4. Closing

The closing of a binary image A by B (the SE) is defined by

$$A \bullet B = (A \oplus B) \ominus B. \tag{6}$$

According to Dougherty and Lotufo [7], closing is a dilation immediately followed by an erosion using the same SE. Closing is frequently used to remove pepper noise of an image.

3. Materials and Methods

In order to demonstrate our method, we implemented a library of morphological operations in hardware, since hardware implementations speed up the bit-stream processing. We developed our MM library using *Quartus II Web Edition FPGA design software 9.0 service pack 2*. The used FPGA was Cyclone-II EP2C70F896C6N, as part of the development kit DE2-70 of Altera Corporation. The clock of the DE2-70 can reach up to 260 MHZ. We did the implementation in electronic schematics, as presented by Figure 1, and the simulations using the Quartus II Waveform Editor.

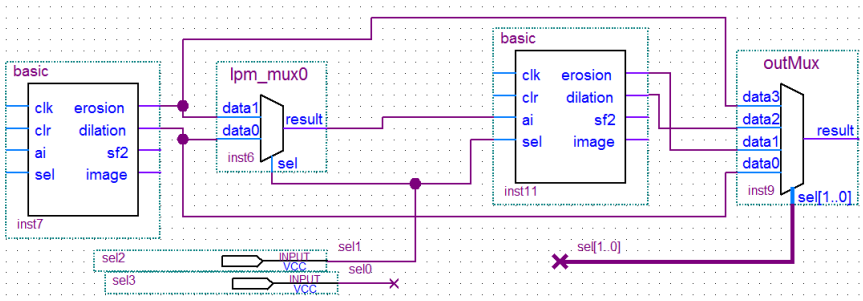


Figure 1: Part of the electronic schematic of our library

During the simulations, we used only one dataset composed by twenty digital images of the Martian surface. The size of each image was 256 X 256 pixels. However images in different datasets can have different sizes. Therefore, our library must be changed to match the size of images of each dataset. This is advantageous because it optimizes the performance of the library. The change is easily performed by hardware reconfiguration, what justifies the use of FPGAs to develop the library of Figure 1.

Our library has a main unit named *basic*, which is divided in three modules: Buffer, Control Unit (CU) and Structuring Element (SE). Buffer is composed by smaller blocks that are serially connected in the following sequence: *Buffer1*, *Buffer2*, *Shift Register* and *Buffer3*. For this library, any buffer is a FIFO memory. The image is inserted pixel-by-pixel (serially) in our library, according to the clock frequency supported by the FPGA.

The library commands multiplexers to select the morphological operation to be performed. Each row of Table 1 presents the morphological operation

that can be processed by our library and its respective command.

Table 1: Morphological operations of the library

Operations	Commands
Erosion	**00**
Opening	**01**
Closing	**10**
Dilation	**11**

For all operations, the main image needs to be converted to a binary image before to pass through the circuit presented by Figure 1. For dilation, according to Equation (3), pixels of the main image also need to be inverted before and after the image processing.

In our library, the unit responsible to perform erosion or dilation (*basic*) is physically duplicated and the second *basic* is serially connected to the first one, as presented by Figure 1. This serial connection allows to perform opening and closing operations at high speed. The serial connection was implemented, since an opening is an erosion followed by a dilation, while a closing is a dilation followed by an erosion, according to Equations (5) and (6), respectively.

Our library starts its task filling the *Buffer1* pixel-by-pixel. When *Buffer1* is full, each pixel of the image continues its serial path filling *Buffer2*. Next, when *Buffer2* is full, each pixel of the image continues its serial path filling the *Shift Register*. In other words, before starting the image processing, our library must be filled with the first two lines of the image. Finally, when *Shift Register* is full, each pixel of the image continues its serial path filling *Buffer3*. The Control Unit (CU) controls this serial flow of pixels through those buffers and the image processing.

During the image processing, the CU continues filling our library with pixels of the image (pixel-by-pixel), following the order of pixels present in the image (from the top-left pixel to the final bottom-right pixel). This way, when an image is being processed by our library, the SE remains static in a fixed position, while set-by-set of pixels of the image flows below it. The image processing ends when the last set of pixels is processed.

3.1. Buffer

Buffer is the part of the library that handles the input image. The image is serially inserted through three FIFO memories (*Buffer1*, *Buffer2* and *Buffer3*). *Buffer1*, whose size is the same of the width of the input image, receives the

data (pixels or bits) of the image and sends this data to *Buffer2*. The size of *Buffer2* is calculated by

$$Size\_Buffer2 = (Width\_Image) - (Size\_Shift\_Register) \quad (7)$$

The output of *Buffer2* is sent to a five bits shift register, part of which represents the left, center and right pixels of the SE. A shift register behaves similar to a FIFO memory, but it is a sequential circuit. *Buffer3* has the same size of *Buffer1* and its input comes from the second more significant bit of the shift register. The second more significant bit of the shift register is positioned exactly below the center of the SE (considering a $3X3$ cross shaped SE) and represents the pixel currently processed (our library processes the image pixel-by-pixel in this position).

Buffer is controlled by CU, according to clock cycles. CU controls the write and read enable commands of each Buffer as well as their outputs. After *Buffer1* and *Buffer2* are both completely full, the image starts to pass below the SE until the image ends.

3.1. Structuring Element

Structuring Element (SE) is another module of the library. The SE is a 3X3 cross shaped binary entity that remains static in a fixed position while the image is processed. The five pixels of the SE are all equal to one. Set-by-set of pixels of the image flows below the SE. Each set is composed by: the three most significant bits of the *Shift Register*, that correspond to the west, the center and the east of the SE respectively; the output of *Buffer1*, that corresponds to the north of the SE; the output of *Buffer3*, that corresponds to the south of the SE. The image processing finishes when the last set of pixels is processed under the SE.

According to the MM theory, since all pixels of the SE are equals to one, the SE will fit all 3X3 cross shaped sets whole composed by pixels equals to one in the image. Therefore, in this case, the core of an erosion operation can be performed by an *and* logic operation. Figure 2 shows the core of the electrical schematic used by our library to perform erosion, using a $3X3$ cross shaped SE. The SE is composed by the three most significant bits of the buffer's shift register (west (*shft[4]*), center (*shft[3]*) and east (*shft[4]*) of the SE, respectively) and, the outputs of *Buffer1* and *Buffer2* (north (*si*) and south (*sf*) of the SE, respectively).

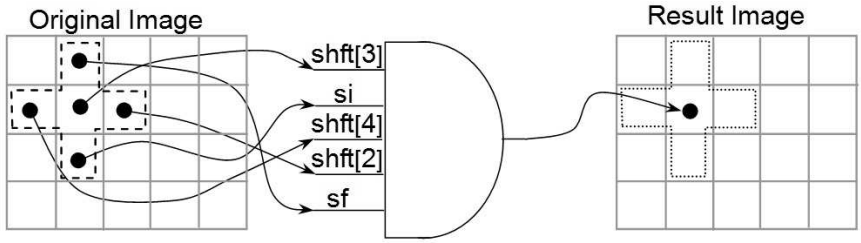


Figure 2: Core of the electrical schematic used by our library to perform erosion, using a 3X3 cross shaped SE.

3.2. Control Unit

Control Unit (CU) has two main functions: it controls the Buffer and it controls when the output image begins and when it ends.

To control the Buffer, the CU needs to synchronize the write and read enables of each FIFO. CU performs this task based on the clock of the FPGA and some counter circuits.

As a result of the MM image processing, a new image is generated. The CU will start the image processing, generating the first pixel of this new image, when *Buffer1* and *Buffer2* are both completely full with the first two rows of the main image. The last pixel of the new image is generated when the last pixel of the main image is being processed under the central pixel of the SE.

4. Experiments

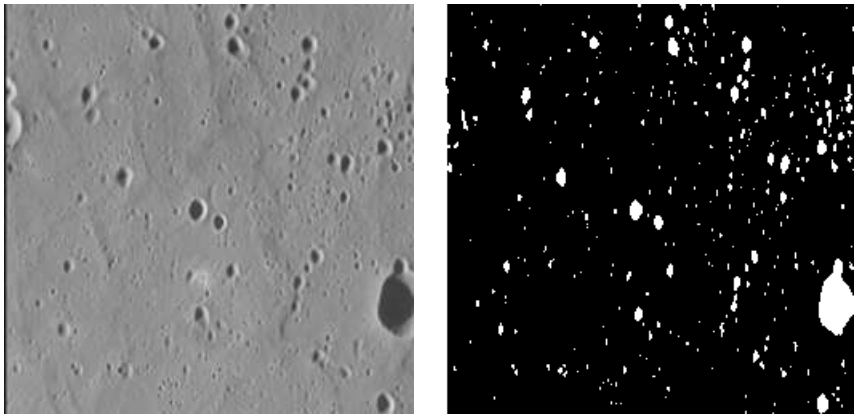
The experiments were performed over a dataset composed by twenty digital images of the Mars surface, obtained by the Mars Odyssey space probe, such as the example showed by Figure 3(a). Each image was previously resized to 256 X 256 pixels and converted to a binary image before the morphological processing, since our library was firstly developed to process binary images with this size. However, our library can be reconfigured to process images with other sizes, as already mentioned before. Figures 3(b), 4(a), 5(a), 6(a) and 7(a) present the image of the Figure 3(a) resized to 256 X 256 pixels and converted to a binary image to facilitate comparisons.

In order to perform the experiments, our library was positioned between two RAM memories, since the transmission of the images through a bit-serial communication channel was simulated by using a RAM as source and another

RAM as destination of the bit-serial data. For all tested images, the morphological processing started before the transmission of each image was finished.

The simulation using RAM blocks in the Quartus II required to load the source memory with each binary image. So, each image was firstly converted from a *.tif* file to a *.hex* file before each processing. After each processing, the simulator generated another *.hex* file in the destination memory. Each *.hex* file was converted to a *.tif* file.

During the experiments, all images of the dataset were eroded, dilated, opened and closed by using our library, as respectively exemplified by Figures 4(b), 5(b), 6(b) and 7(b). These morphological operations help cartographers to perform cartographic mappings, such as the mapping of impact craters on planetary surfaces.



(a) (b)

Figure 3: Digital image of the Mars surface, obtained by the Mars Odyssey space probe: (a) Gray scale view; (b) Binary view.

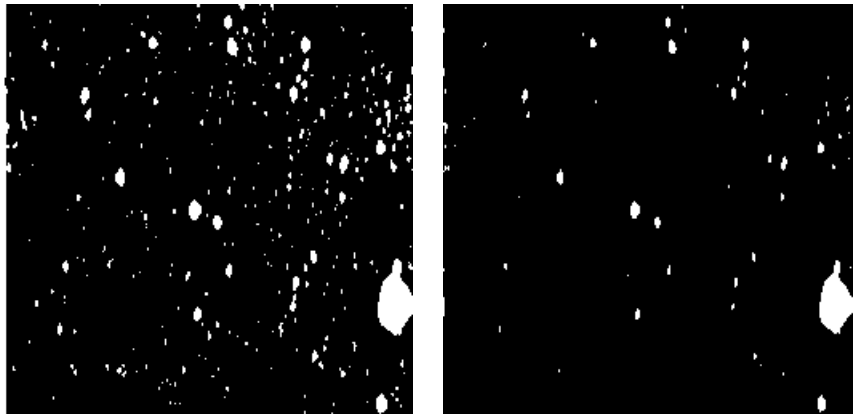


Figure 4: Digital image of the Mars surface: (a) Binary view; (b) The same image eroded by our library

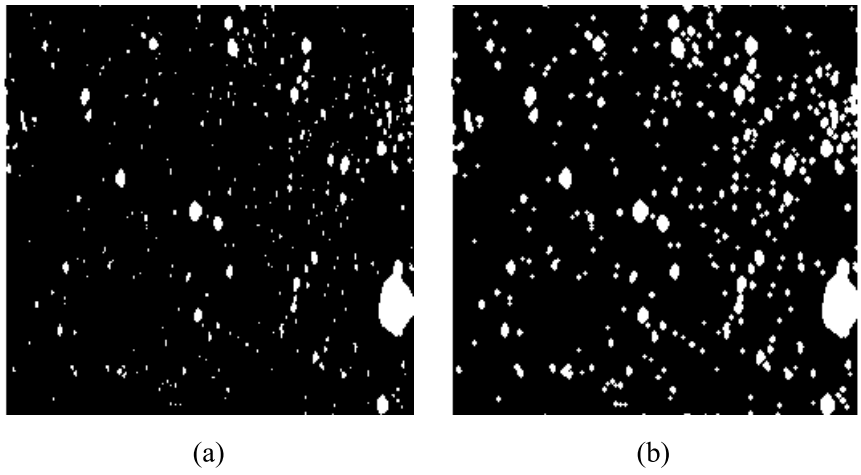
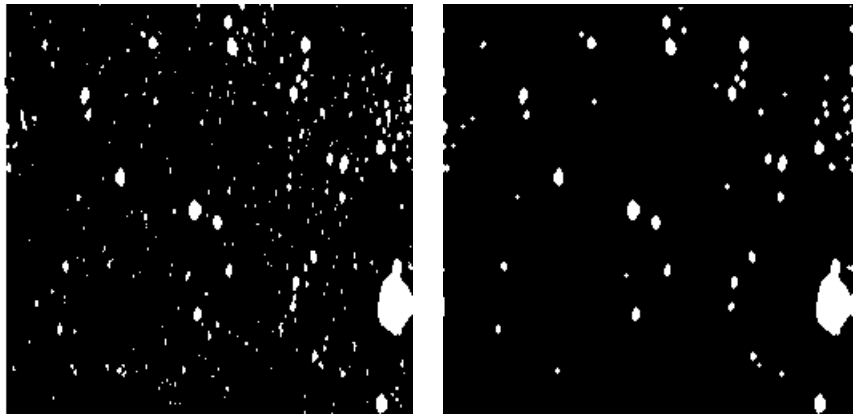
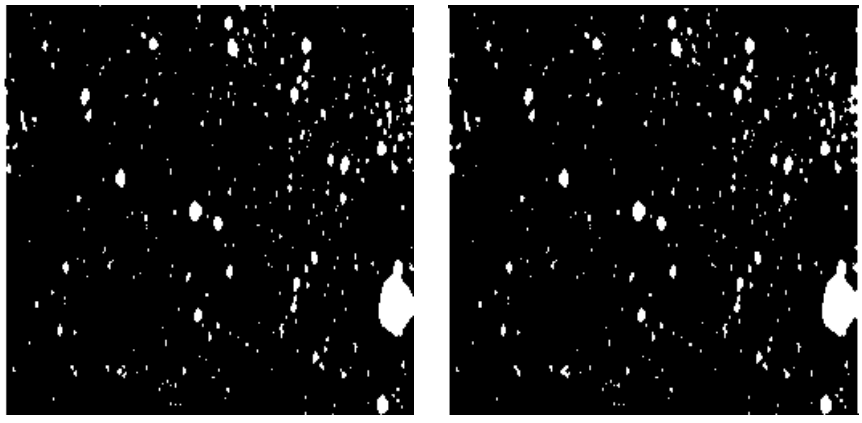


Figure 5: Digital image of the Mars surface: (a) Binary view; (b) The same image dilated by our library



(a) (b)

Figure 6: Digital image of the Mars surface: (a) Binary view; (b) The same image opened by our library



(a) (b)

Figure 7: Digital image of the Mars surface: (a) Binary view; (b) The same image closed by our library

5. Results

The results were achieved based on the previously commented experiments. For binary images of planetary surfaces, geological features such as impact craters, for example, are usually represented as foreground (white pixels) in contrast to the flat surface represented as background (black pixels). In order to quantitatively validate our method applied to process binary images of planetary surfaces, we used an adapted version of the metric presented by Prati et al. [11] and Radke et al. [12], as follows:

$$PSP = \frac{(TP + TN)}{TP + FP + TN + FN}. \tag{8}$$

In Equation (8), we have: PSP as the percentage of pixels successfully processed; TP as true positives, i.e. pixels of the foreground correctly processed by our library; TN as true negatives, i.e. pixels of the background correctly processed by our library; FN as false negatives, i.e. pixels of the foreground incorrectly represented as background; FP as false positives, i.e. pixels of the background incorrectly represented as foreground.

According to Equation (8), and when borders are omitted, our library presented 100% of success, when applying erosion, dilation, opening and closing operations for all planetary images of the dataset processed in the tests. Moreover, the time used by our library (in number of clock cycles) to erode or dilate each image is given by Equation (9), and to open or close each image is given by Equation (10),

$$CCED = 2W + W\text{x}H, \tag{9}$$

$$CCOC = 4W + W\text{x}H. \tag{10}$$

In Equations (9) and (10) we have: $CCED$ as the number of clock cycles used by our library to erode or dilate each image, and $CCOC$ as the number of clock cycles used by our library to open or close each image; W as the width of the image (in number of pixels); H as the height of the image (in number of pixels).

If all images of the dataset are continuously passed to our library during the tests, without intervals among the images, the constants of Equations (9) and (10) are considered only to the first image.

For all performed tests, our method was successful applying a morphological image processing while receiving a planetary image by a bit-serial communication channel.

6. Discussion

In this paper, we proposed a method that allows to apply a morphological image processing while receiving a planetary image by a bit-serial communication channel, starting the processing before the transmission of the image has finished. Thus, this paper presented a bit-stream morphological image processing method for planetary images. To demonstrate the method, we implemented a library of morphological operations (composed by erosion, dilation, opening and closing) on FPGA, since hardware implementations speed up the bit-stream processing.

Transmissions of planetary images from probes or landing modules are usually performed through bit-serial communication channels, according to Nagy et al. [9]. Contrary to software implementations, such as [2], [3], [10], [15] and [16], that need to wait the transmission of the images has finished before to start the image processing, our method allows to perform the morphological processing during the transmission of the images. Hardware implementations speed up morphological image processing, however, contrary to our method, implementations such as [1], [14], [17], [18] do not explore the advantage of the bit-stream processing.

Therefore, the current work contributes: presenting a method that speeds up morphological processing of planetary images transmitted through a bit-serial communication; making available to cartographers a computational solution to process planetary images for cartographic mapping of some planets and natural satellites, such as the mapping of impact craters on planetary surfaces; providing a reconfigurable library of morphological operations to process images with other sizes.

6.1. Conclusions

We conclude that our method allows to apply morphological image processing while receiving planetary images by a bit-serial communication channel, starting the processing before the transmission of the images has finished. Therefore our method speeds up morphological processing of images transmitted this way, and our method represents a computational solution to process planetary images for cartographic mappings faster than the already proposed implementations. Moreover, when borders are omitted, our library achieves 100% of success rates when performing erosion, dilation, opening and closing.

In the future, we intend to: incorporate other morphological operations to the library, e.g., top-hat opening and closing, and morphological gradient;

implement the library in VHDL, so that it becomes portable; adapt the library so that it works with other sizes and shapes of SE; perform border treatment.

References

[1] R. Arunmozhi and G. Mohan, Implementation of digital image morphological algorithm on FPGA using hardware description languages, *International Journal of Computer Applications*, **57**, No 5 (2012), 1-4; DOI: 10.5120/9107-3253.

[2] T. Barata, E.I. Alves, J. Saraiva and P. Pina, Automatic recognition of impact craters on the surface of mars, In: *Inter. Conf. on Image Analysis and Recognition (ICIAR 2004)*, Springer, Porto (2004), 489-496; DOI: 10.1007/978-3-540-30126-4_60.

[3] L. Brunozze, L. Lizzi, P.G. Marchetti, J. Earl and M. Milnes, Recognition and detection of impact craters from EO products, In: *Proc. of ESA-EUSC (2004) - Theory and Applications of Knowledge-Driven Image Information Mining with Focus on Earth Observation*, CD, Madrid (2004), 17-18.

[4] K. Compton and S. Hauck, *An Introduction to Reconfigurable Computing*, Technical Report, Northwestern University, Dept. of ECE (1999).

[5] P. Dillien, Electrically reconfigurable arrays-ERAs, In: *IEE Colloquium on New Directions in VLSI Design*, IET, London (1989), 6/1-6/6.

[6] D.H. Dwight and E. Detjens, FPGA design principles (a tutorial), In: *29th ACM/IEEE Design Automation Conf.*, ACM/IEEE, (1992), 45-46.

[7] E.R. Dougherty and R.A. Lotufo, *Hands-on Morphological Image Processing*, SPIE Publications, Washington (2003).

[8] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Prentice Hall, Upper Saddle River (2007).

[9] J. Nagy, B. Sdor and S. Szalai, *Improvement of EGSE Architecture and Software in Last Decades*, Technical Report, Budapest, Hungary (2012).

[10] M.M. Pedroza, E.A. Silva, M.A. Dias and J.R. Nogueira, The use of mathematical morphology in the detection of impact craters on digital images of the martian surface, *Journal of Communication and Computer*, **9** (2012), 1351-1357.

[11] A. Prati, I. Mikic, M.M. Trivedi and R. Cucchiara, Detecting moving shadows: algorithms and evaluation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**, No 7 (2003), 918-923; DOI: 10.1109/TPAMI.2003.1206520.

[12] R.J. Radke, S. Andra, O. Al-Kofahi and B. Roysam, Image change detection algorithms: a systematic survey, *IEEE Transactions on Image Processing*, **14**, No 3 (2005), 294-307; DOI: 10.1109/TIP.2004.838698.

[13] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London (1982).

[14] A.J. Tickle, J.S. Smith and Q.H. Wu, Development of morphological operators for field programmable gate arrays, *Journal of Physics: Conf. Series 76 012028*, (2007), 1-17; DOI: 10.1088/1742-6596/76/1/012028.

[15] G. Troglio, J. Le Moigne, J.A. Benediktsson, G. Moser and S.B. Serpico, Automatic extraction of ellipsoidal features for planetary image registration, *IEEE Geoscience and Remote Sensing Letters*, **9**, No 1 (2012), 95-99; DOI: 10.1109/LGRS.2011.2161263.

[16] E.R. Urbach and T.F. Stepinski, Automatic detection of sub-km craters in high resolution planetary images, *Planetary and Space Science* **57**, No 7 (2009), 880-887.

[17] J. Velten and A. Kummert, FPGA-based implementation of variable sized structuring elements for 2D binary morphological operations, In: *The First IEEE Intern. Workshop on Electronic Design, Test and Applications*, IEEE, Christchurch (2002), 309-312; DOI: 10.1109/DELTA.2002.994636.

[18] J. Velten and A. Kummert, Implementation of a high-performance hardware architecture for binary morphological image processing operations, In: *The 47th IEEE Intern. Midwest Symposium on Circuits and Systems (MWSCAS '04)*, IEEE, Hiroshima (2004), II-241 - II-244; DOI: 10.1109/MWSCAS.2004.1354137.