Computers and Electrical Engineering 43 (2015) 9–16

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

An analysis of computational models for accelerating the subtractive pixel adjacency model computation ☆

Marisol Rodriguez-Perez, Alicia Morales-Reyes, René Cumplido *, Claudia Feregrino-Uribe

Instituto Nacional de Astrofisica, Optica y Electronica, Sta. Ma. Tonantzintla, Puebla 72840, Mexico

ARTICLE INFO

Article history:
Received 20 August 2013
Received in revised form 5 January 2015
Accepted 9 January 2015
Available online 11 March 2015

Keywords:
Steganalysis
SPAM
FPGA
CUDA
GPU

ABSTRACT

Detecting covert information in images by means of steganalysis techniques has become increasingly necessary due to the amount of data being transmitted mainly through the Internet. However, these techniques are computationally expensive and not much attention has been paid to reduce their cost by means of available parallel computational platforms. This article presents two computational models for the Subtractive Pixel Adjacency Model (SPAM) which has shown the best detection rates among several assessed steganalysis techniques. A hardware architecture tailored for reconfigurable fabrics is presented achieving high performance and fulfilling hard real-time constraints. On the other hand, a parallel computational model for the CUDA architecture is also proposed. This model presents high performance during the first stage but it faces a bottleneck during the second stage of the SPAM process. Both computational models are analyzed in detail in terms of their algorithmic structure and performance results. To the best of Authors' knowledge these are the first design proposals to accelerate the SPAM model calculation.

1. Introduction

The increasing transmission of personal data, mainly through the Internet, has motivated the creation of new methods to protect information. Several algorithmic techniques have been developed to tackle main data protection problems. For example, cryptographic techniques have been developed to encode information against unauthorized access and steganographic methods aim at protecting sensible data without perceptible modifications, trying to be unnoticed to the third eye [1]. However, protecting data via covered communications are also used in terrorism and pornography [2,3], therefore it has been also necessary to develop inverse algorithmic techniques known as steganalytic methods which help to discover covered data in order to detect hidden information.

Steganalysis is classified in specific and universal. Specific steganalysis requires knowledge of the targeted steganographic method, thus limiting its application arena. On the other hand, universal steganalysis recognizes if an image has been modified by any steganographic technique [4]. To achieve its goal, universal steganalysis uses feature extraction techniques to train a classifier in order to distinguish distortions caused by steganographic methods. Depending on feature extraction domain, universal steganalysis is classified in spatial and transform.

☆ Reviews processed and approved for publication by the Editor-in-Chief.

* Corresponding author.
E-mail addresses: marisolr@inaoep.mx (M. Rodriguez-Perez), a.morales@inaoep.mx (A. Morales-Reyes), rcumplido@inaoep.mx (R. Cumplido), cferegrino@inaoep.mx (C. Feregrino-Uribe).

http://dx.doi.org/10.1016/j.compeleceng.2015.01.004
0045-7906/

Designing steganalysis methods has focused on improving detection rates. However, computational cost of steganalytic techniques is high and optimizing processing times is nowadays necessary due to the massive amounts of data transferred through large networks. In order to improve the processing ratio, some authors have proposed steganalytic hardware implementations. For image data, Sun *et al.* [5] developed an FPGA-based architecture for the RS algorithm, a specific steganalysis method proposed by Fridrich *et al.* [6] which recognizes LSB (Least Significant Bit). The proposed architecture uses a three stage pipeline and was synthesized in a Xilinx Virtex II FPGA (now obsolete). In 2013, Gutierrez-Fernandez *et al.* introduced an FPGA-based architecture for transform domain universal steganalysis in JPEG images [7]. This architecture is based on JPEG's compatibility algorithm proposed by Fridrich *et al.* [8]. Authors proposed a pipeline scheme implemented in VHDL and synthesized in a Xilinx Virtex 6 FPGA. However, to the best of the authors' knowledge, hardware architectures for universal steganalysis on the spatial domain have not been reported.

Spatial features usually focus on modeling pixels neighborhoods. Such as the rich model proposed in 2012 by Fridrich *et al.* [9], where different pixel dependency sub-models are used as features. Using different types of sub-models facilitates detection of different embedding artifacts; however, dimensionality increases substantially. An ensemble is used for classification.

In 2010, Guan *et al.* proposed a method called Neighborhood Information of Pixels (NIP), in which, differences between pixels within the neighborhood and the central pixel are calculated and subsequently codified using rotation invariant [10]. The result is processed as a histogram removing empty values. In 2011, Arivazhagan *et al.* used 4×4 segments where pixels differences are calculated according to nine paths within the neighborhood [11]. Results between -4 and 4 are placed within a co-occurrence matrix and are used as characteristics vectors.

In 2010, Pevny *et al.* proposed a universal method where differences between neighboring pixels are modeled by first and second order Markov Chains [12]. This method was designed primarily for spatial based steganography, however, it demonstrated a good performance when detecting transform domain steganography. Because of its ability in both domains, this method is considered in this investigation from a hardware perspective to speed-up its process.

Among several universal steganalytic methods, the Subtractive Pixel Adjacency Model (SPAM) has shown a good performance for detecting stego-images watermarked with the most popular steganographic methods, both in the spatial domain and in the transform domain. The model is also scalable to color images, and its algorithmic design makes it a good candidate from a hardware architectural perspective. Moreover, many steganalytic methods share common processing tasks carried out in SPAM (pixel differences and transition probabilities). Therefore, the proposed computational models are flexible and can be adapted, as steganalytic processing cores, to other spatial domain techniques.

In this paper, two parallel computational models are investigated, an FPGA-based hardware architecture and a CUDA based algorithmic model is proposed for the first order SPAM model. A complete analysis of performance results in both implementation platforms is presented. In the next section, the SPAM model is detailed and analyzed from an architectural perspective. Section 3 presents both parallel computational models: an FPGA based architecture's design and a GPU programming model. Results are analyzed for each proposed approach in the same section. Final remarks are presented in Section 4.

2. Subtractive pixel adjacency model

The Subtractive Pixel Adjacency Model (SPAM) is a spatial domain method for features extraction in images with possible stego-data. This model has shown the best detection rate among other state of the art techniques. Moreover, operations required to obtain feature vectors are arithmetically simple which makes this model suitable for optimization from a hardware perspective. In [12], it is stated that correlation between neighboring pixels in a natural image can be used to detect some irregularities caused by stego-data. However, the resulting model would not be practical due to its dimension. To reduce model's dimension, it has been proposed using pixels differences versus pixel co-occurrences or neighborhood histograms. Thus, obtaining the difference model of several stego-images demonstrated that most significant information lay in small differences. Therefore, differences range is reduced together with its dimensionality.

The algorithm is mainly divided in three stages, see Fig. 1. First, the model calculates differences between pixels in eight directions $(\uparrow, \nearrow, \longrightarrow, \searrow, \downarrow, \swarrow, \longleftarrow, \nwarrow)$ in the spatial domain. For example, the horizontal differences are calculated by $A_{i,j} = I_{i,j} - I_{i,j+1}$ and $B_{i,j} = I_{i,j} - I_{i,j-1}$, where I is an (m, n) image, and $i \in [1 \ldots m]$, $j \in [1 \ldots n]$. Second, to model pixel dependencies along the eight directions, a Markov chain is used between pairs of differences (first order chain) or triplets (second order chain). For dimensionality reduction of the transition probability matrix, only differences within a limited range are considered. Thus, the transition probability matrix is calculated just for pairs within $[-T, T]$. In [12], authors propose $T = 4$ for first order and $T = 3$ for second order, because of their relevance in steganalysis.

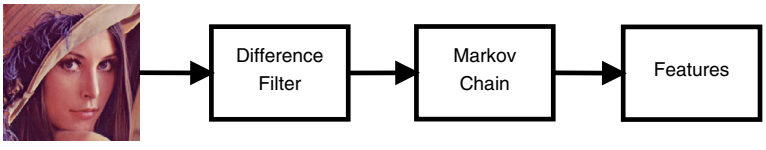


Fig. 1. SPAM main stages.

Thus, for horizontal first order Markov chain calculation:

$$PA_{(x,y)} = P(A_{(i,j+1)=x} | A_{(i,j)=y}),$$

$$PB_{(x,y)} = P(B_{(i,j)=x} | B_{(i,j+1)=y})$$

where $x, y \in [-T, T]$. Once the eight transition probability matrices have been calculated, the average of the four horizontal and vertical and the average of the four diagonal matrices are used as features on a binary support vector machine (SVM) classifier. In the next section, the proposed parallel computational models are described in detail.

3. Parallel computational models

Calculating the Subtractive Pixel Adjacency Model (SPAM) involves computationally exhaustive tasks. In order to calculate both feature vectors, difference matrices in several directions are obtained for every image pixel. A total of eight difference matrices are calculated from which eight frequency vectors and eight frequency matrices are derived and then averaged to obtain the final feature vectors. The operations involved are arithmetically simple only addition and subtraction are required during the most exhaustive algorithmic part.

In this article, two parallel computational models are introduced in order to efficiently obtain the SPAM model. On the one hand, an application oriented hardware architecture has been designed taking advantage of the available physical resources in reconfigurable fabrics. This allows a tailored SPAM architecture delivering optimized performance. On the other hand, a CUDA computational parallel model implemented on a GPU platform is also presented. Using GPUs to solve general purpose computational tasks has been recently opened to the wider research community, allowing expensive computational problems to be parallelized and thus being solved with improved performance.

3.1. FPGA hardware architecture

The proposed architecture consists of 3 stages: differential filtering, transition probabilities calculation and features calculation. In Fig. 2 the overall block diagram is presented. In the first stage, differential filtering, the input image is stored in a 32-bit memory in such a way that at every iteration four pixels are obtained j, $j+1$, $j+2$ and $j+3$ from rows i and $i+1$, see Fig. 3, an address generator determines processing memory allocations. These four bits are processed by differential filters calculating pixels differences according to every direction shown in Fig. 4. In Fig. 5, difference calculations for A and D directions (East and South) are shown. Pixels in the same row are subtracted for A direction while for D direction pixels from different rows are subtracted.

Results from differential filtering are converted into addresses and are accumulated in a register file P with length $2T+1$, containing the number of occurrences of difference values between $-T$ and T. A file register F with length $(2T+1)^2$ contains also the number of occurrences per pair of values within the threshold. Once all image differences are calculated the second processing stage begins.

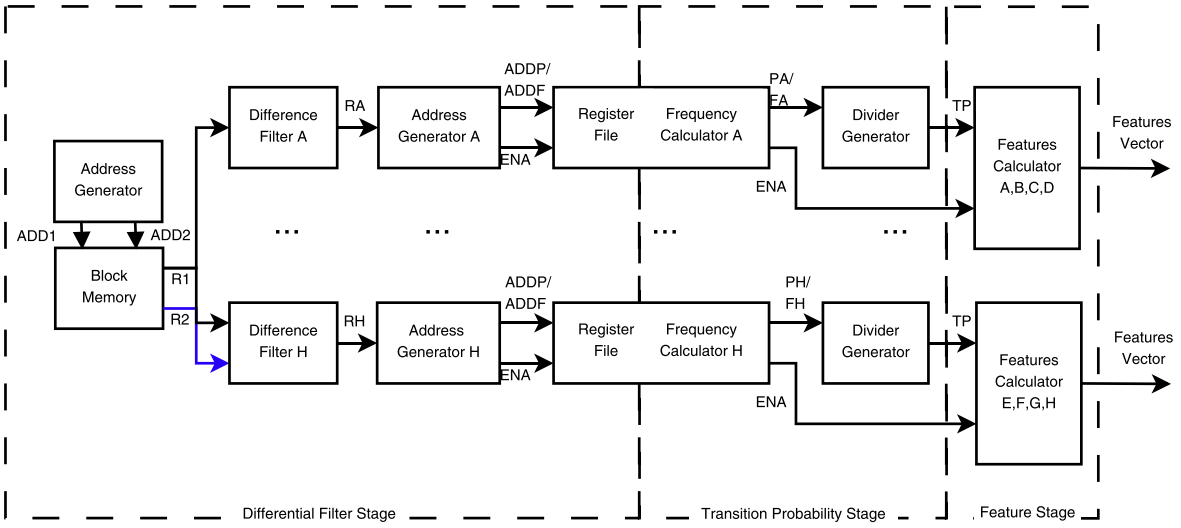


Fig. 2. SPAM block diagram.

159	210	254	253	209	...
90	98	142	143	108	...
106	100	92	93	118	...
113	115	110	113	116	...
106	118	119	113	112	...
118	120	117	116	116	...
104	109	106	113	153	...

Fig. 3. Pixels acquisition.

During the transition probability stage, stored values are read sequentially from register files P and F, which are then divided to finally obtain pixels transition probabilities within the threshold. In Fig. 6, two examples are presented for calculating P and F occurrences on A direction (East) for values with $T = -4$ and $T = -3$.

To start the final stage (features extraction), divider's latency is considered (18 cycles). Two features are processed every clock cycle corresponding to directions probabilities A, B, C and D and E, F, G and H respectively. Fig. 7 shows one of the modules implemented for features calculation.

3.1.1. Performance results

In Table 1, results obtained with characterization data on a Xilinx FPGA device Virtex-6 XC6VSX315T are presented. The proposed architecture is implemented using System Generator Version 3.0 running on Matlab Version R2011a and synthesis results are obtained with Xilinx ISE Design Suite 13.2. To the best of our knowledge no other hardware architecture design for the SPAM model has been reported in the literature. In [5], an FPGA parallel implementation for the RS algorithm achieves high throughputs and thus is able to deal with large amounts of data. However, the RS algorithm is not as robust as the SPAM model for detecting stego-images.

The SPAM architecture herein presented deals with input image sizes of 512 by 512 pixels, it processes 4 pixels blocks at a time, achieving an operational frequency of 87.035 MHz. Thus the proposed FPGA hardware architecture is able to process 1328.069 frames per second (fps) due to a latency of 2^{16} clock cycles. In terms of hardware resources, the complete architecture requires less than half fully used LUT-FF and less than 10% of registers available on the Virtex-6 device.

3.2. CUDA parallel approach

The CUDA algorithmic approach for the SPAM model takes advantage of several characteristics such as the SIMT (Single Instruction Multiple Thread) execution model [13]. Fig. 8 shows an overall view of the processing data flow. An input image is read and loaded into global memory, a square grid configuration using an optimum number of threads for parallelization is defined. Because there is no reuse of input values in the calculation of difference matrices, one image pixel per thread is evaluated. In contrast of having multiplication and accumulation operations which are common to many computationally expensive tasks; the SPAM model requires subtraction only at an initial stage, thus using a higher memory level, such as shared memory, in the CUDA architecture makes no difference in the final performance. The difference matrices calculation is performed at a maximum parallelization level, having a proper block-grid configuration, threads are executed following the optimum warp size of 32 threads ad hoc to the FERMI Nvidia architecture [14].

A characteristic of the SPAM model is the repetition of values when differences are calculated for opposite directions, for example, the difference calculated in the right direction of any given image pixel is equal to the difference calculated in the left direction at the same image position. Similar behavior is repeated for other directions (up/down, left-up/right-down,right-up/left-down). This has a significant impact on the overall parallel model because only half the calculations are needed to obtain the full set of frequency vectors and matrices.

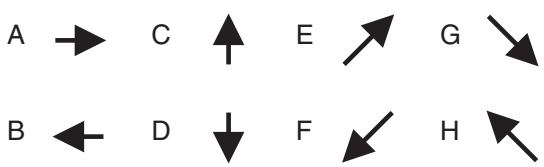
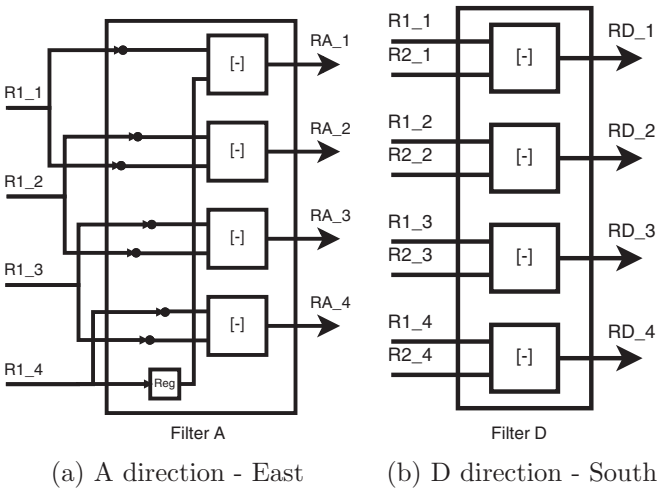


Fig. 4. Differences directions.



(a) A direction - East (b) D direction - South

Fig. 5. Differential filtering for A and D directions.

In order to calculate frequency vectors, which means identifying the number of differences within the pre-defined range $(-T, T)$, a binary matrix for differences within the range is obtained. For optimal reduction to calculate every vector and matrix coefficient, an implemented function of the Thrust CUDA's library is used [15]. Thrust is a library based on the Standard Template Library (STL) which allows implementing high performance parallel applications through a high level interface for full interaction with the CUDA architecture.

The parallel process in the GPU is as follows: an input image is loaded from the host (CPU) and is allocated in memory at the GPU. Once in global memory, the difference matrix in every direction is calculated in parallel taking full advantage of the optimal configuration per warp in the CUDA FERMI architecture.

Differences for A direction (East) are the same than for B direction (West), see Fig. 4. This also occurs for differences calculated in opposite directions: C (North) and D (South), E (North-East) and F (South-West), G (South-East) and H (North-West). Therefore, frequency vector and matrices are calculated once for only four directions out of eight. Occurrences values corresponding to opposite directions have different positions in the *P* vector and *F* matrices. For *P*, values appear rotated while for *F*, values are not only rotated but reversed in such a way that rows for A direction (East) correspond to columns for B direction (West) of every pixel.

For every coefficient in each difference matrix, a binary vector is obtained indicating whether a value is within the threshold with 1 or not with 0. This binary vector is of the same size as the input image. Thrust CUDA library function *count* optimally reduces vector arrays to determine occurrences values per coefficient within the threshold $(-T, T)$.

Because every thread is dealing with one pixel at a time, and due to the nature of operations, subtractions mainly, there is not performance improvement if the process is taken one memory level up, to shared memory [16]. This is because there are not iterative operations per pixel in the SPAM model and racing conditions would occur when trying to write occurrences values from different threads to the same memory location, possibly resulting in almost sequential processing. Thus, it was decided to calculate occurrence binary vectors and reduce them using an optimized tool like the Thrust CUDA library. For every *P* and *F* vector and matrix coefficients, the Thrust function *count* is used. It affects the GPU performance because of the number of coefficients that need to be calculated but it provides an acceptable performance on different GPU platforms.

3.2.1. Performance results

SPAM's CUDA parallel computational model is assessed for efficiency on two different GPU platforms: GTS-450 (Intel core i7 2.93 GHz) and GTX-480 (Intel(R) Celeron(R) M processor 1.50 GHz). In Table 2 technical details for the used GPU modules are provided. Throughput among used graphics modules is expected to vary significantly due to their very different processing power. The number of CUDA cores available on each targeted device is as follows: GTS 450 and GTX 480 have 192 and 480 cores respectively. The CUDA architecture is organized in streaming multiprocessors (SM) of 32 or 48 (GTS 450) CUDA cores; each SM executes groups of 32 threads called warps. Thus, GTS 450 and GTX 480 Nvidia modules execute 4 and 15 warps respectively.

Table 3 shows processing time results per graphics module. Transfer time for the input image from the host (CPU) to the device (GPU) is reported showing a fastest transfer for the GTX-480 in comparison to the GTS-450. However, a fair comparison is not possible because each module operates on different CPU platforms. No image pre-processing is performed in the CPU, thus CPU overhead is disregarded from the main processing load. CUDA Toolkit 4.2 is used with both GPU modules.

Two main stages are considered in the CUDA computational model. Calculating difference matrices is performed fully in the GPU. The processing time in the GTS-450 and the GTX-480 is 0.299602(ms) and 0.113648(ms) respectively. After, the transition probabilities stage is performed using the Thrust library in the GPU with a host to device pointer; results are

pA =

8	8	9	10	69	10	13	14	7

fA =

0	0	0	0	1	0	2	1	0
0	1	1	1	2	0	1	0	1
2	1	1	1	1	1	0	0	0
0	0	1	0	0	3	1	2	0
1	0	0	1	55	0	0	0	0
0	0	0	1	2	1	0	2	0
0	0	0	1	0	1	2	2	2
0	0	0	1	1	0	1	0	2
0	0	3	0	0	0	1	1	0

(a) P and F occurrences on A direction - East with $T = -4$

pA =

8	8	9	10	69	10	13	14	7

fA =

0	0	0	1	0	2	1	0	
0	1	1	1	2	0	1	0	1
2	1	1	1	1	1	0	0	0
0	0	1	0	0	3	1	2	0
1	0	0	1	55	0	0	0	0
0	0	0	1	2	1	0	2	0
0	0	0	1	0	1	2	2	2
0	0	0	1	1	0	1	0	2
0	0	3	0	0	0	1	1	0

(b) P and F occurrences on A direction - East with $T = -3$

Fig. 6. Transition probabilities calculation using register files.

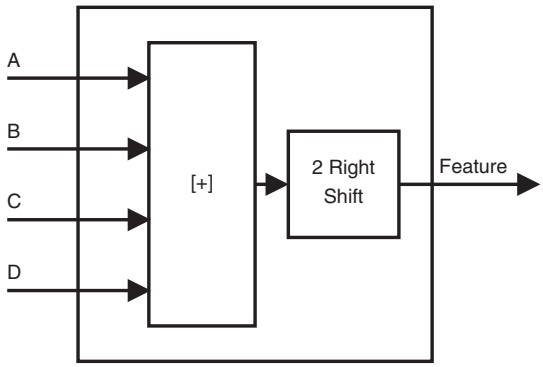


Fig. 7. Features calculation.

passed to the final stage, features stage, in the CPU. The processing time for this combined stage increases significantly to 95.39872(ms) in the GTS-450 and 65.08325(ms) in the GTX-480. This occurs because of the number of coefficients when

Table 1
FPGA architecture results.

Device	Slice registers	Slice LUTs	LUT-FF	Bonded IOBs	Operational frequency (MHz)	Frames per sec.
Virtex 6	38,023 (9%)	88,355 (44%)	37,885 (42%)	117 (19%)	87.035	1328.069

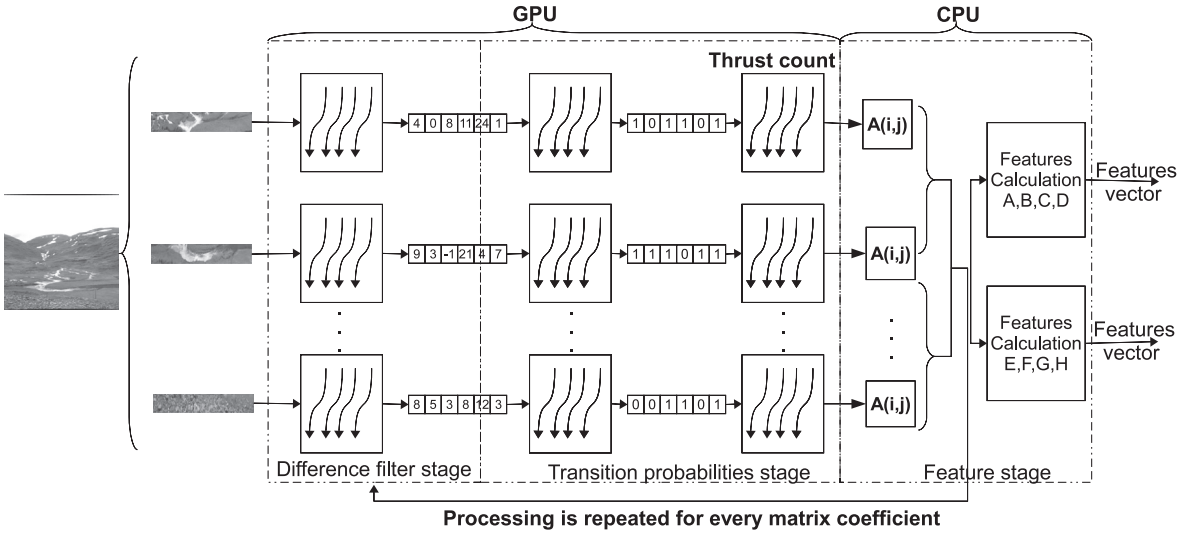


Fig. 8. CUDA SPAM model.

Table 2
GPU modules.

Platform	SM	Cuda cores	Global memory MB	GPU clk rate MHz	Processing power GFLOPS
GeForce GTS-450 [17]	4	192	1024	1057	601.34
GeForce GTX-480 [18]	15	480	1536	1400	1344.96

Table 3
GPU performance results.

	GTS-450 (ms)	GTX-480 (ms)	GTS-450 (fps)	GTX-480 (fps)
CPU–GPU	0.4684	0.3661	2134.7086	2730.9272
Difference stage	0.2996	0.1136	3337.7614	8799.0989
Transition probability and features stages (GPU–CPU)	95.398	65.0832	10.4823	15.3649

calculating the reduction for P and f vector and matrices. Although, each independent coefficient is efficiently calculated, there are approximately 90 coefficients to calculate per direction in the SPAM model. This significantly increases the overall processing time.

4. Conclusions

The SPAM model is a robust technique for stego-images detection. It is a spatial domain method that has shown the best detection rates among other state of the art techniques. Its algorithmic structure is suitable from an implementation perspective aiming to achieve high performance and thus fulfilling hard real-time constraints. Two approaches are presented in this article: a hardware architecture targeting reconfigurable fabrics and a parallel computational model targeting GPU platforms. Experimental results show that the FPGA architecture achieves high performance while maintains an acceptable usage of hardware resources of less than 50% of the total available. On the other hand, the proposed CUDA parallel computational model has shown limited performance due to the calculation of more than 90 coefficients per direction which is carried out by the Thrust's library. There is a significant bottleneck when simplifying occurrences vectors which cannot be accelerated by using higher memory levels in the CUDA architecture. However, during the first stage to calculate differences

matrices, the GPU processes three and eight times more frames than the FPGA. This means that applying specialized parallelization techniques to the second stage would imply a significant improvement of the SPAM performance in the GPU.

Acknowledgements

Marisol Rodriguez-Perez was supported by CONACYT studentship number CVU-412162. This work was done under partial support of CONACYT project Grant CB-SEP-2010-01-158135.

References

[1] G. Simmons, The prisoners' problem and the subliminal channel, "The prisoners' problem and the subliminal channel", in: Proceedings of CRYPTO '83, Advances in Cryptology, 1983, pp. 51–67 <http://link.springer.com/chapter/10.1007/978-1-4684-4730-9_5>.
[2] K. Maney, Bin Laden's Messages Could be Hiding in Plain Sight, USA Today <http://usatoday30.usatoday.com/tech/columnist/2001/12/19/maney.htm>.
[3] N. Robertson, P. Cruickshank, T. Lister, Documents Reveal Al Qaeda's Plans for Seizing Cruise Ships, Carnage in Europe, CNN <http://www.cnn.com/2012/04/30/world/al-qaeda-documents-future>.
[4] Nissar A, Mir A. Classification of steganalysis techniques: a study. Digital Signal Process. 2010;20(6):1758–70. http://dx.doi.org/10.1016/j.dsp.2010.02.003. <http://linkinghub.elsevier.com/retrieve/pii/S1051200410000412>.
[5] Sun K, Pan X, Wang J. Hardware based steganalysis. Signal Process. Images 2008:269–78.
[6] J. Fridrich, M. Goljan, R. Du, Reliable detection of LSB steganography in color and gray-scale images, in: Proceedings of the ACM Workshop Multimedia Security, 2001, pp. 27–30.
[7] E. Gutierrez-Fernandez, M. Portela-García, C. Lopez-Ongil, M. Garcia-Valderas, FPGA-based Implementation for Steganalysis: A JPEG-Compatibility Algorithm, in: Proceedings of SPIE 8764, VLSI Circuits and Systems VI, vol. 8764 (876407), http://dx.doi.org/10.1117/12.2017476, http://dx.doi.org/10.1117/12.2017476.
[8] J. Fridrich, M. Goljan, R. Du, Steganalysis based on JPEG compatibility, in: A.G. Tescher, B. Vasudev, V.M. Bove, Jr. (Eds.), Proceedings of SPIE 4518, Multimedia Systems and Applications IV, vol. 4518, 2001, pp. 275–280, http://dx.doi.org/10.1117/12.448213, http://dx.doi.org/10.1117/12.448213.
[9] Fridrich J, Kodovsky J. Rich models for steganalysis of digital images. IEEE Trans. Inf. Foren. Sec. 2012;7(3):868–82.
[10] Q. Guan, J. Dong, T. Tan, An effective image steganalysis method based on neighborhood information of pixels, in: IEEE International Conference on Image Processing, 2011, pp. 2721–2724.
[11] S. Arivazhagan, W.S.L. Jebarani, M. Shanmugaraj, A novel approach to low volume generic steganalysis, in: 21st International Conference on Systems Engineering, 2011, pp. 153–158.
[12] Pevny T, Bas P, Fridrich J. Steganalysis by subtractive pixel adjacency matrix. IEEE Trans. Inf. Foren. Sec. 2010;2(5):215–24.
[13] Corporation N. NVIDIA Cuda Programming Guide. NVIDIA Corp.; 2010.
[14] Corporation N. NVIDIA Cuda Best Practices Guide: A Hands-on Approach. NVIDIA Corp.; 2010.
[15] N. Corporation, Cuda Toolkit Documentation, CUDA NVIDIA <http://docs.nvidia.com/cuda/thrust/index.html> (current April 2013) (2007–2012).
[16] Kirk DB, mei W, Hwu W. Programming Massively Parallel Processors. NVIDIA Corporation; 2010.
[17] C. Nvidia, Geforce GTS 450 (May 2014) <http://www.geforce.com/hardware/desktop-gpus/geforce-gts-450/>.
[18] C. Nvidia, Geforce GTX 480 (May 2014) <http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-480/>.

Marisol Rodríguez Pérez received the M.Sc. degree in computer science from the National Institute of Astrophysics, Optics and Electronics, Mexico in 2014. Her current research interests include hardware architectures and image processing.

Alicia Morales-Reyes obtained the Ph.D. degree at the University of Edinburgh UK, in 2011. She received her M.Sc. degree in Computer Science (INAOE) in 2006 and her BEng degree in Electrical and Electronics Engineering (UNAM) in 2002, Mexico. She is an associate researcher at Computer Science Department at INAOE collaborating within the Reconfigurable and High Performance Computing research group.

René Cumplido is a Professor at the Computer Science Department at INAOE. He holds a B.Sc. degree in Computer Systems, a M.Sc. in Electrical Engineering and a Ph.D. Electrical Engineering from Loughborough University, UK. His research interests are reconfigurable computing applications, FPGA technology, and custom hardware architectures.

Claudia Feregrino-Uribe is a researcher at the Computer Science Department at INAOE, Puebla, Mexico. Her research areas are Data Compression, Cryptography and Steganography (Watermarking), Digital Systems Design, FPGA applications. She received her M.Sc. in Electrical Engineering with Telecommunications option from the CINVESTAV, Guadalajara and Ph.D. from Electronic Engineering in Digital Systems from Loughborough University in the United Kingdom.