# HYPERSPECTRAL IMAGES CLUSTERING ON RECONFIGURABLE HARDWARE USING THE K-MEANS ALGORITHM

*Abel Guilhermino da S. Filho, Alejandro C. Frery , Cristiano Coêlho de Araújo , Haglay Alice, Jorge Cerqueira, Juliana A. Loureiro, Manoel Eusebio de Lima , Maria das Graças S. Oliveira , Michelle Matos Horta*

UFPE - Universidade Federal de Pernambuco
Caixa Postal 7851 - 50740-540 - Recife - PE, Brasil
(agsf,frery,cca2,hans,mel,mgso,mmh)@cin.ufpe.br

***Abstract.*** *Unsupervised clustering is a powerful technique for understanding multispectral and hyperspectral images, being k-means one of the most used iterative approaches. It is a simple though computationally expensive algorithm, particularly for clustering large hyperspectral images into many categories. Software implementation presents advantages such as flexibility and low cost for implementation of complex functions. However, it presents limitations, such as difficulties to exploit parallelism for high performance applications. In order to accelerate the k-means clustering a hardware implementation could be used. The disadvantage in this approach is that any change in the project requires previous knowledge of the hardware design process and can take several weeks to be implemented. In order to improve the design methodology, an automatic and parameterized implementation for hyperspectral images has been developed in a hardware/software codesign approach. An unsupervised clustering technique k-means that uses the Euclidian Distance to calculate the pixel to centers distance was used as a case study to validate the methodology. Two implementations, a software and a hardware/software codesign ones, have been implemented. Although the hardware component operates in 40MHz, being 12.5 times lesser than the software operating frequency (PC), the codesign implementation was approximately 2 times faster than software one.*

## 1 Introduction

Some modern optical sensors can produce data cubes with hundreds of spectral channels and million of pixels. An alternative to cope with this amount of information is to organize the data so that pixels with similar spectral indices are clustered in the same category (class). This supplies an understanding of the data and a segmentation of the image, that can be useful in the diverse stages of processing and analysis of the data.

Hyperspectral image data is increasingly available from a variety of sources, including commercial and government satellites, as well as airbone and ground-based sensors. This is accompanied by an increasing in spatial resolution and in the number of spectral channels. Hyperspectral images contain hundreds of spectral channels per pixel.

The data of remote sensoring have shown extremely useful for the study, survey and the management of natural resources [4][5], mainly because of the following three resolution characteristics:

- *temporal*: allows information collection at different periodic time of the year in distinct years, allowing possible dynamic studies of a region;
- *spectral*: allows the information attainment on a target in the nature in distinct regions of the specter;
- *space*: makes possible the information attainment in different scales, since regional until the local ones.

The use of collected data by sensory systems also requires, frequently, methods to data clustering that may be classified as supervised and unsupervised.

In the supervised method, information on the data is known, while in the unsupervised methods, this knowledge is not available. A classic approach for the clustering of hyperspectral data is the k-means algorithm. This is an unsupervised iterative method that produces successive clustering [2] in a expensive computational processing.

Leeser [6] discusses algorithm level transforms that enable k-means to be implemented in hardware. In general, computation using floating point arithmetic and the multiplication-heavy Euclidian distance calculation are fine on a general purpose processor, but they have large area and speed penalties when implemented on an FPGA. In this work the algorithm based on Manhattan metric has been implemented on the Annapolis Wildstar board. After, in another approach [2], a parameterized implementation of the k-means algorithm, based on the number of pixels in an image, the number of channels per pixel, and the number of bits per channel, as well as the number of clusters has been developed. It has the added advantage that the parameterized design compiles approximately three times faster than the original.

The standard software implementation of k-means uses floating-point arithmetic and Euclidean distances. In the Euclidean distance there are some advantages in relation to other metrics (Manhattan, Max or linear combination) since it is rotationally invariant [2]. On the other hand, Euclidean distance algorithm has an expensive implementation, being this one of the main reasons for implementing some k-means algorithm

Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI'03)
0-7695-2009-X/03 $17.00 © 2003 **IEEE**

modules in a hardware component, which the parallelism can be explored.

In this paper we apply the k-means unsupervised clustering algorithms to Airbone Visible Infrared Imaging Spectrometer (AVIRIS) data sets [1]. A single AVIRIS image contains 614x512 pixels with 224 16-bits channels, or approximately 138Mbyte of data. An simple example of a cube of data of this image with 4 pixels and 4 bands is showed in Figure 1.

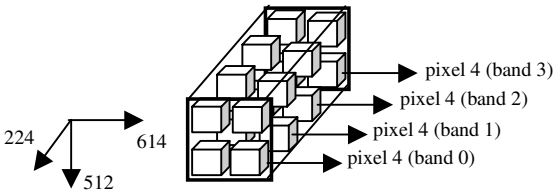


Figure1- Cube of data example with 4 pixels and 4 bands

A methodology for hyperspectral image segmentation using k-means algorithm as alternative to speed up the clustering and to improve the performance in the process of image segmentation has been developed.

As case study, was implemented software and codesign approaches from k-means algorithm. In order to manage the two processing approaches, also was implemented a segmentation tool to validate this methodology based on k-means algorithm on the PCI [9] Xilinx Spartan-II board has been prototyped. The PCI board operates at 40MHz and is connected to a PC K6 II 500MHz. The k-means algorithm was partitioning manually in modules, based on performance requirements. The software approach from k-means algorithm was all implemented in C++ language following the above specification. Otherwise, in the codesign approach, some modules are implemented in C++, and one module (*BlockClassifier*) was implemented in VHDL language.

In the next section we present the AVIRIS sensor. Section 3 we presents the hyperspectrals images segmentation and segmentation techniques focusing on unsupervised category. In section 4 the hardware/software and k-means implementation are discussed. Section 5 presents the current implementation of the k-means algorithm on the PCI board Xilinx FPGA and the software tool developed. Section 6 presents conclusions and future works.

## 2 The AVIRIS Sensor System

AVIRIS, [1] is an optic sensor that supplies calibrated images of the spectral radiance at about 224 bands, with wave length of 400 to 2500nm.

The current platform of system AVIRIS is an airplane U-2 model, that was modified to adapt it to the sensor. Flies 20km above sea level and 730km/h speed approximately.

AVIRIS system uses a device that sweeps the scene in the longitudinal direction ("Whisk Broom" mode) [1] as shown in Figure 2, producing 614 pixels for the 224 detectors of each sweeping. Each pixel produced by the instrument covers an area of 20x20 meters in the land, thus reaching to each covering of the land 11 km width approximately.

Due its high spectral resolution, the AVIRIS system is capable to reproduce the spectral response curves of some types of targets, allowing a good characterization of the absorption peaks as reflectance. These peaks are essential to characterize certain types of targets (mineral in geology, for example). Traditional sensors, when compared with the spectral information presented by the hyperspectrals sensors are inadequate for many applications [1].

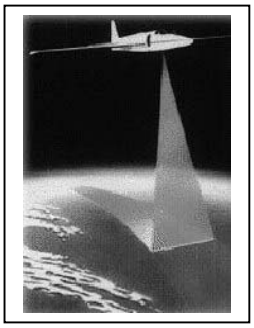

Figure 2 - (AVIRIS - "Whisk Broom" mode)

A single AVIRIS image contains 614x512 pixels with 224 16-bits cannels, or approximately 138Mbyte of data.

## 3 K-Means Clustering

The main goal of the image analysis is to identify important aspects of the image under processing. This kind algorithms segment the image by clustering pixels into classes based on the spectral similarity of each pixel to other members of the class. Each pixel in the image is represented by a pointer to the spectral class associated with that pixel. At the same time, these clustering algorithms also provide a very real data compression.

The unsupervised methods attribute to the technique or to the chosen algorithm, the task to identify, by itself, the existing class in a data set [7].

A possible approach to solve a problem of pixel clustering, frequently used in remote sensoring, consists of looking at pixels clusterings in the space. These clusterings are composed by a group of similar pixels, in accordance with a determined criterion (Euclidean distance, Manhattan or Max for instance). In this paper we use the unsupervised method k-means and the Euclidian distance metric method.

In this method was chosen the notation used by Leeser [6], which given a set of N pixels, each composed by C spectral channels, and represented as a point in C-dimensional Euclidian space (that is, $x_n \in \Re^C$, with

n=1,…,N); the pixels are split into K clusters with the property that pixels in the same cluster are similar. Each cluster is associated with a "center" value which is representative of (and close to) the pixels in that class.

One measure of the quality of a partition is the within-class variance; this is the sum of squared distances Euclidian from each pixel to that pixel's cluster center.

For a fixed partition, the optimal (in the sense of minimum within-class variance) location for each center is the mean of all pixels in each class. For a fixed choice of centers, the optimal partition assigns points to the cluster whose centers are closest. The k-means clustering algorithm provides an iterative scheme that operates over a fixed number (K) of clusters, while attempting to simultaneously optimize center locations and pixels assignments.

Firstly, the k-means algorithm, chooses the "centers" $c_i$ of each cluster. In some works [3], a refinement process is done through a function probability associated with the set of pixels, in order to find a local minimum for one faster convergence of the method.

Secondly, a cluster is associated for each pixel of the image based on minimum distance between a given pixel $x_i$ and the "center", $c_i$ of that cluster. Once the distance between the pixel and the center of each cluster is calculated, the pixel is associated to the cluster that obtained the minimum distance. In this stage the Euclidian distance is used as metric. Consider a point **x** and a cluster center **c**, where **i** indexes the spectral components of each cluster. The Euclidian distance is defined as following:

$$\|x-c\|^2 = \sum_i |x_i - c_i|^2$$

Once associated a cluster for each pixel of the image, the clusters centers are recomputed. To compute the cluster center, the mean of all pixels of that cluster are calculated. If there are N points, K centers and C spectral channels, then there will be O(NKC) operations. For the Euclidian distance, each operation requires computing the square of a number. The stop criteria (***sc***), $sc: \max[|c_i - c_j|] < \varepsilon$, is reached when the module of difference between the news value of the center and the old one, is less than an accurancy factor ( $\varepsilon$ ). Otherwise, the process continue with the news clusters. The work uses $\varepsilon = 0$.

This procedure is very expensive when implemented in hardware since large multiplications and square of numbers have to be performed resulting in larges bit vectors and so area and high speed processing.

## 4 Hardware/Software approach

In a general way, digital systems implementations in hardware, have as advantage to explore concurrency increasing and so the data processing speed. Unfortunately, applications implemented in such way derive high cost prototyping and low flexibility design. Implementations in software in turn, present advantages such as flexibility and low cost of complex functions implementation. However, it presents limitations such as difficulties to explore parallelism and high speed applications.

On the other hand, a hardware/software codesign approach offers a more flexible alternative for this kind of problem, based on certain constraints (cost function), such as: silicon area, speed, application and communication cost.

In this methodology, the software component can be represented by microcontrollers, or general propose microprocessors and the hardware one by a ASIC or a Field Programmable Gate Array (FPGA) [10]. The FPGA is an array of logic cells and interconnection cells that can be configured in the field to implement a desired designed function.

The k-means segmentation algorithm consists basically of two main steps: (1) Association of each pixel to one of the k clusters, and (2) Recomputation of the clusters centers. To recompute the clusters centers, the pixels must be associated with the cluster to be computed. Also, the values of all pixels associate to each cluster with its bands must be accumulated.

The figure 3 depicts the k-means hardware/software architecture implemented in this work. The design consists of a classifier for hyperspectrals images that performs unsupervised segmentation and implements the k-means algorithm. The classifier receives the hyperspectral image, with a typical number of bands, approximately 200, and as result of the algorithm processing, a classified image is generated, representing the thematic map of the original image.

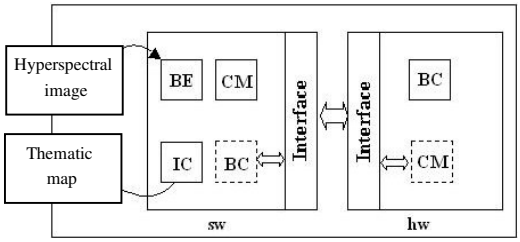

Figure 3. Classifier hardware/software partitioning

The classifier is composed by software and hardware modules. The partitioning is done manually based on performance requirements. The modules *BlockExtractor* (BE), *ClassifierManager* (CM) and *ImageComposer* (IC) were implemented in software, and *BlockClassifier* module (BC) was implemented in the hardware. The modules indicated for dashed line in hardware side representing proxies, and the dashed line in software side are used for communication, through a parallel interface.

In the direction to evaluating the advantages of the hardware/software approach, this classifier was implemented in two versions. In the first version, all k-means algorithm was implemented in software. In this in

case, all its specification was made in C++, and executed in a PC using the Windows operating system.

In the second version, based on a codesign approach, the module BC was implemented in hardware while the others modules (BE), (CM) and (IC), were implemented in software. In this architecture, a PC, using Windows operation system, connect to a FPGA prototyping board, through a PCI interface has been used. The hardware component implemented in FPGA, was totally specified in a hardware description language, VHDL (VHSIC Hardware Description Language) and synthesized in a CAD Xilinx Foundation 3.1i tool.

## 5 Modules Implementation

The classifier performs three basic functions: management and user interface, reading of the original image, segmentation and generation of the thematic map. These functions are implemented by processes which had been partitioned in four main modules: *ClassifierManager* (CM), *BlockExtractor* (BE), *BlockClassifier* (BC) and *ImageComposer* (IC), as mentioned in the previous section. Figure 4 shows a general vision of these modules.

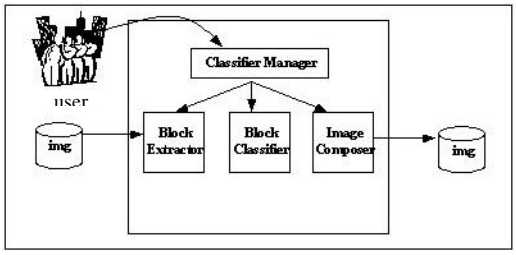


Figure 4. General vision of the modules

The *ClassifierManager* block has two main functions, the first one to provide a friendly user graphical interface; the second function is to manage the execution of the algorithm.

The *ClassifierManager* defines the block size (set of pixels) to be read from the original and classified images. This module sends commands to the *BlockExtractor* that extracts a block of pixels from the original image, repasses this block to the *BlockClassifier* and finally it sends commands to the *ImageComposer* that stores this block of pixels. *ClassifierManager* repeats these operations until a stop criterion defined by the user is reached. After these steps, the *ImageComposer* generates a classified image (thematic map).

The *BlockExtractor* block reads a file containing the hyperspectral image and returns a set (block) of pixels, being each pixel composed by its bands.

The *BlockClassifier* block implements the main part of k-means algorithm being responsible by segmentation of each pixel of the original image. An internal vision of this module is shown in Figure 5. This module receives a set of pixels of the original image and a set of centers from all clusters and provide as result a classified set of pixels with the same size.

The *BlockClassifier* block is composed of two submodules: *CalcDistance* and *ClassSelector*. The first one is responsible to calculate the distance between one pixel and the center of a cluster, and the second, selects the cluster which the pixel belongs. The classification or segmentation is made by calculating the distance of each pixel to the center of each cluster.

In the codesign approach, this module was chosen to be implemented in hardware due its implicit concurrency features. In this way the parallelism can be adequately explored. The distance processing is then calculated simultaneously for each one of the clusters; as well as, several pixels can be processed in parallel. For each pixel is calculated the Euclidian distance between the center of that cluster and the pixel. The pixel is associated to the cluster with the minimum distance.

The Figure 5 shows as the parallelism aspect is explored in *CalcDistance* module, calculating the Euclidian distance between each pixel and all K clusters centers. This procedure for one pixel, may be perform simultaneously for the other N pixels. The maximum value for N is 614x512 pixels per channel, and there are 220 channels. The limitation of this approach is the hardware area.

Finally, the *ImageComposer* block generates the thematic map from original image.

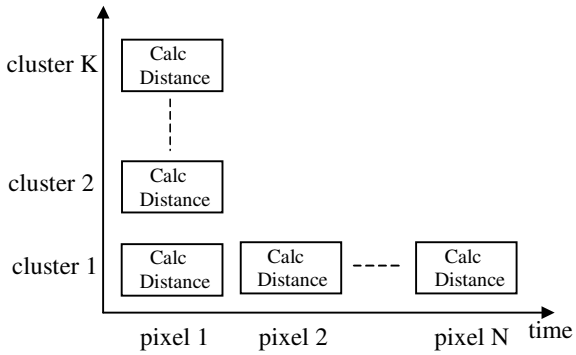


Figure 5 – *BlockClassifier*

The k-means algorithm is an iterative process, and it spends a certain time to converge. Two different approaches of the k-means algorithm stop criterion were implemented, the "*minimum error*" and the "*number of iterations*".

When the algorithm stop criterion chosen by user is "*minimum error"*, the algorithm must converge when the clusters centers are identical during two consecutive iterations. After convergence, the application save an image file with each cluster in a different color.

The user can also choose the number of iterations ("number of iterations" cr iterion). In this case, the processing usually does not guarantee the same accuracy reached by the former criterion, because normally the algorithm could not have been converged.

## 6 Case Study

Two case study implementations, a software and a hardware/software ones, have been developed.

In the hardware/software approach, the hardware component is represented by the *BlockClassifier* which implements the k-means algorithm on a XESS PCI board [9] with one FPGA Xilinx Spartan-II of 200.000 gates. The board is connected to a 500MHz K6 II workstation by PCI interface of 33MHz. The hardware component was specified in VHDL and synthesized in the Xilinx Foundation 3.1i CAD tool.

The *ClassifierManager*, *BlockExtractor*, *ImageComposer* compose the software part of the codesign approach that were implemented in C++. In the software version, all blocks are implemented in C++. A tool has been developed in C++, to classify images of any data size of 16 bits per pixel per channel, type BIP (*Band Interleave Pixel*) of an AVIRIS image and, returning as result, a thematic map with the same size of the original image, 614x512 pixels, one 8-bit size channel.

The original image was obtained from AVIRIS system, from Air base of Moffet Field, in California. It is located in initial latitude of +037.449470º, final latitude of +037.449820º, initial longitude: -121.806630, final longitude: -122.216380, initial altitude: 21412.400 and final altitude: 21384.600. A image composition may be seen in Figure 6(a), (b) and (c). The Figure 6(a) depicts a general vision of the California localization [8] in the map. The Figure 6(b) depicts [8], in more details, a zoom performed in the rectangle delimited in the figure 6(a). The Figure 6(c) shows the original image, obtained through AVIRIS [1], that was used as case study.

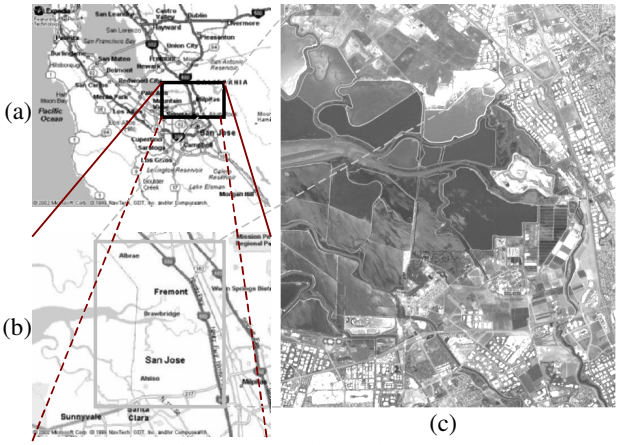


Figure 6 - Composition of original image

A classification tool for hyperspectrals images, with graphical interface was developed to manage both implementations. In this tool the user has the option to select the image, bands, number of clusters, stop criterion ("*maximum error*" or "*number of iterations*") besides choosing software or hardware/software codesign approach.

When the user chooses to classify in codesign approach, for each iteration, the software application writes the clusters centers in the FPGA by a PCI communication interface. The software partition waits for the k-means algorithm execution in hardware.

At the end, an interruption is generated to the software component signaling the end of the classification of each pixel. The software receives the pixel classified, storage it in a file and draws the pixel in the user interface. After, the software sends a new pixel to the FPGA and the process is repeated until the image is completely processed.

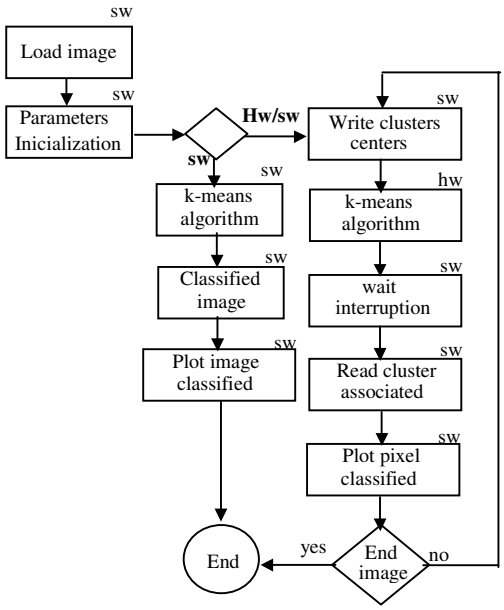


Figure 7 - Design flow

In this case study, the image classification involves a computation of 614x512 = 314368 pixels and 3 clusters, each one with 2 channels (bands). Each pixel on the hyperspectral image is an integer type (with size 2 bytes), and the thematic map pixel generated by algorithm processing, is a char type (1 byte) both in the software and codesign approaches.

The main part to be observed, is that, the "*BlockClassifier*", the more complex block of the algorithm. When implement in hardware, in the codesign approach, is twice faster than when it is implemented in software approach. That means that even the FPGA platform working at 40MHz clock, 12.5 times lesser than the workstation, it still presents a better performance (2 times) than the same block implemented in software as show the Table 1.

The algorithm for 3 clusters and 2 channels was implemented in codesign version and takes 91% area of the FPGA Spartan II, from Xilinx running at 40MHz and 3500 C++ code lines. As the platform does not support the substitution of the FPGA it is not possible to implement a case study with more bands and /or clusters. This limitation is due to the FPGA capacity only and not of the algorithm that is parameterizable. The hardware

part in codesign approach, was successful simulated, synthesized and mapped and the software part compiled on C++ Builder tool.

In both approaches the "*minimum error"* was used as stop criterion. The k-means algorithm converged in 17 iterations.

The codesign implementation spends at about twice the software implementation in terms of code lines as showed in Table 1, because of the high complexity to implement some functions as (square root, square, division) in hardware.

| | **Software** | **Codesign** |
|---|---|---|
| Code lines | 3816 | 3427(Hw)<br>3500(Sw) |
| Number of slices | - | 2141 (91%) |
| Time processing | 500MHz<br>(18seg) | 40MHz<br>(8,91seg) |

Table 1

Figure 8 illustrates the result of the classification using 3 clusters and 2 bands. It was detect three different clusters showed in figure 8. In this work, was focused to determine the clusters location, given a original hyperspectral image. As future work, and another problem to solve, is how to determine what is a chosen cluster.

Figura 8. Thematic map of the AVIRIS image f970620t01p02_r03_sc02.c.rfl

## 7 Conclusions

In this work, a hardware/software codesign approach implementation for k-means algorithm has been implemented. A comparison with a software version was also presented in order to show the advantages of the former approach for this kind of problem, where high computation and parallelism processing are important issues. Although the limitation of the FPGA and its low operating frequency, 40MHz, 12,5 times lesser than software solution, the codesign approach was 2 times fast. In the future, could be used state-of-the-art solutions, as embedded processors and SoC platforms, as Excalibur (Altera) or Microblaze (Xilinx).

## 8 References


[1] AVIRIS; (Image Download); Ordering Free AVIRIS Standard Data Products ; Download in 02/09/2002; "ftp://popo.jpl.nasa.gov/pub/outgoing /f970620t01p02_r03.rfl.tar.gz".

[2] Belanov, P.; Leeser M.; Estlick M.; Gokhale, M.; Szymanski, J.J. and Theiler, J. "Applying reconfigurable hardware to the analysis of multispectral and hyperspectral imagery." *Proc. SPIE* 4480 (2001) 100-107.

[3] Bradley, P.S.; Fayyad, U.M. "*Refining initial points for k-means clustering"*. In Proceedings of the 15 International Conference on Machine Learning, pages 91-99. Morgan Kaufmann, 1998.

[4] Bryan F.J. Manly, Multivariate Statitical Methods, Chapman & may, Second edition, 1994.

[5] Duda R. Hart P.; Stork, D.G.; Pattern Classificatiuon, 2nd edition, New York: John Wiley and Sons, 2001.

[6] Leeser, M.; Estlick, M.; Theiler, J. and Szymanski, J. J.; "Algorithmic Transformations in the Implementation of k-means Clustering on Reconfigurable Hardware", FPGA 2001, Copyright 2001 ACM p.103-110.

[7] Selim, S. Z.; Ismail, M.A.; "K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality." IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol. PAMI-6 n.1, pp.81-86. (1984)

[8] Expedia. Localização da imagem Hiperespectral. ; http://www.expedia.com/Default.asp ; Download in 24/10/2002

[9] Xilinx. Localization in http://www.xilinx.com; Download in 24/10/2002

[10] Hauk, S. – "The Roles of FPGAs in Reprogrammable Systems" *Proceedings of the IEEE*, Vol. 86, No. 4, pp. 615-639, April, 1998.