A New Web-Based System for Unsupervised Classification of Satellite Images from the Google Maps Engine

Ángel Ferrán[a], Sergio Bernabé[b], Pablo García-Rodríguez[a] and Antonio Plaza[b]

[a]Grupo de Ingeniería de Medios
University of Extremadura, Cáceres, SPAIN
E-mail: angelmff@gmail.com, pablogr@unex.es
[b]Hyperspectral Computing Laboratory
University of Extremadura, Cáceres, SPAIN
E-mail: {sergiobernabe, aplaza}@unex.es

ABSTRACT

In this paper, we develop a new web-based system for unsupervised classification of satellite images available from the Google Maps engine. The system has been developed using the Google Maps API and incorporates functionalities such as unsupervised classification of image portions selected by the user (at the desired zoom level). For this purpose, we use a processing chain made up of the well-known ISODATA and k-means algorithms, followed by spatial post-processing based on majority voting. The system is currently hosted on a high performance server which performs the execution of classification algorithms and returns the obtained classification results in a very efficient way. The previous functionalities are necessary to use efficient techniques for the classification of images and the incorporation of content-based image retrieval (CBIR). Several experimental validation types of the classification results with the proposed system are performed by comparing the classification accuracy of the proposed chain by means of techniques available in the well-known Environment for Visualizing Images (ENVI) software package. The server has access to a cluster of commodity graphics processing units (GPUs), hence in future work we plan to perform the processing in parallel by taking advantage of the cluster.

Keywords: Web-based system, satellite image classification, Google Maps API

1. INTRODUCTION

Remote sensing image analysis and interpretation have become key approaches that rely on the availability of web mapping services and programs. This resourceful increase has led to the exponential growth of the user community for satellite images, not long ago only accessible by government intelligence agencies.[1,2] In particular, the wealth of satellite imagery available from Google Maps, which now provides high-resolution satellite images from many locations around the Earth*, has opened the appealing perspective of performing classification and retrieval tasks via the Google Maps application programming interface (API).

The combination of an easily searchable mapping and satellite imagery tool such as Google Maps, with advanced image classification and retrieval features,[3] can expand the functionalities of the tool and also allow end-users to extract relevant information from a massive and widely available database of satellite images (this service is also free for non-commercial purposes). By using the Google Maps Javascript API V3, the full Google Maps site can be embedded into an external website application. Similar services currently available include Yahoo Maps† and OpenStreetMap‡. The characteristics of Yahoo Maps are similar to Google Maps (though the spatial resolution of the satellite imagery in Yahoo Maps is generally lower than Google Maps). OpenStreetMap is a collaborative project aimed at creating a free editable map of the world, a design inspired by sites such as Wikipedia.

Google Maps offers important competitive advantages, such as the availability of high resolution satellite imagery, the smoothness in the navigation and interaction with the system, the availability of a hybrid satellite

*http://code.google.com/apis/maps/index.html
†http://maps.yahoo.com
‡http://www.openstreetmap.org

Satellite Data Compression, Communications, and Processing VIII,
edited by Bormin Huang, Antonio J. Plaza, Carole Thiebaut, Proc. of SPIE Vol. 8514,
85140R · © 2012 SPIE · CCC code: 0277-786X/12/$18 · doi: 10.1117/12.929788

view which can be integrated with other views (e.g. maps view), and adaptability for general-purpose web applications. The possibility lacking in Google Maps is the unsupervised or supervised classification of satellite images at different zoom levels,[4,5] even though image classification is widely recognized as one of the most powerful approaches in order to extract information from satellite imagery.[6,7] The incorporation of such a function into Google Maps would allow relevant information withdrawal from a massive, widely available database of satellite images and the possibility to perform content-based image retrieval (CBIR) tasks,[8] which are of great interest for the exploitation of this database and others for satellite images.

In this paper, we describe a new web-based system (which represents a follow-up of our previous work in[9]) that allows an inexperienced user to perform an unsupervised classification of satellite images obtained via Google Maps. This is run over a web-based system that incorporates a fully unsupervised processing chain based on two well-known clustering techniques: ISODATA[10] and k-means,[11] followed by spatial post-processing based on majority voting.[12] The processing chain has been implemented in C language and integrated into our proposed tool, developed by using HTML5, JavaScript, PHP, AJAX, and other web programming languages.

In previous work,[9] the tool was characterized by being a desktop system and developed in JAVA. The main drawbacks resolved were:

- The image acquisition with the API (only compatible with web applications) is much faster and efficient because we obtain the image directly, a full mosaic compared with the library swingX-WS, in which we had to manually create the mosaic.
- The ability to combine and change the color of the class labels and process images at different zoom levels allows the tool an improved interaction with the user who is now able to supervise the final result.

The previous functionalities are necessary to use efficient techniques for image classification and the incorporation of content-based image retrieval (CBIR), which are main goals in both systems.

The remainder of the paper is organized as follows. Section 2 describes the system architecture, including relevant aspects such as the map, server and client layers. Section 3 describes the processing chain implemented by the proposed methodology, including aspects such as the image acquisition process, the graphical user interface (GUI) that allows end-users to interact with the proposed system, the image processing algorithms implemented, and the procedure adopted for data saving and end product distribution to the users. Section 4 performs an experimental validation of the classification results obtained by the proposed system by comparing the classification accuracy of the proposed chain in terms of the techniques available in the well-known Environment for Visualizing Images (ENVI) software package[§]. Finally, Section 5 concludes the paper with some remarks and hints at plausible future research.

2. SYSTEM ARCHITECTURE

This section describes the architecture of the system, displayed in Fig. 1. It is a web application comprised of several layers or modules. Each module serves a different purpose, and the technology adopted for the development of the system is based on open standards and free software. A combination of different tools has been used for the development of the system, including Apache web server, PHP (a widely-used general-purpose scripting language especially suited for Web development, feasibly embedded into the hypertext markup language such as HTML5, used in our development), JavaScript libraries, Asynchronous JavaScript And XML (AJAX), Jquery and its user interface (UI), cascading style sheets (CSS) and, last but not least, C language for carrying out the image processing tasks.

As shown by the architecture model described in Fig. 1, the proposed system can be described from a high level viewpoint using three different layers, which are completely independent from each other. Due to the adopted modular design, any of the layers can be replaced. Also, the system is fully scalable, allowing for the incorporation of additional layers. The communication between two layers is carried out over the Internet via the hypertext transfer protocol (HTTP). As a result, the system performance will depend largely (as expected) on the available bandwidth. Both the map layer (currently provided by Google Maps) and server layer (by ourselves) are available from any location in the world.

[§]http://www.exelisvis.com/language/en-us/productsservices/envi.aspx

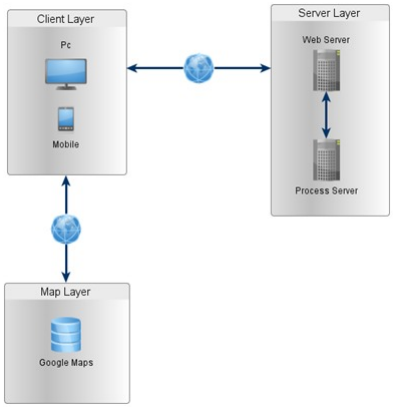

Figure 1. Architecture of the proposed system expressed in the form of different modular layers.

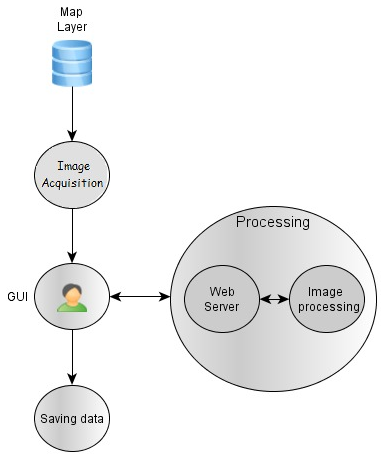

Figure 2. Flowchart describing the methodology adopted for the development of the system.

3. METHODOLOGY

This section describes the methodology adopted for the development of the proposed system. Several main tasks have been identified: image acquisition, graphical user interface (GUI), web server, image processing and image saving. These tasks, summarized in the flowchart given in Fig. 2, will be described in detail in this section.

3.1 Image acquisition

Image acquisition is the starting point of the system operation. The images to be processed are considered from two different viewpoints. On the one hand, the images are parts of a map and, on the other, the images can be considered as specific captures or snapshots of a larger map. The maps are dynamic entities that can be dragged, zoomed (i.e. displayed in more or less detail), but the captures can be seen as static parts of a map which are selected by the end-user via the interface. These captures or snapshots can then be sent to the server and processed in spite of the components of the map layer, in our case supported by the Google Maps engine.

The methodology implemented in our system for the image capture retrieval from the map layer has been developed using JavaScript libraries. These processes have access to the collection or "puzzle" of images that compose a certain map, thus taking advantage of the browser's cache memory to optimize such an operation. The query is directed to the map layer in case that the image is not already in the cache memory (a situation that seldom occurs). This option leads to some advantages, mostly to high speed achieved by the system in the task of image captures regardless of the latency of communications with the server. This feature reduces the communication traffic and increases the performance in the local management of image captures.

In order to achieve such a functionality, several layers of images from the server are considered once the image captures have been processed, thereby obtaining a stack of images in which each layer represents a class (as determined by the considered processing algorithms, i.e. k-means and ISODATA). The layers are completely independent, thus allowing visualization as individual entities or as a combination between layers, providing great flexibility in the analysis of the obtained results and a specific management of layers. Finally, the system also allows for the rapid acquisition of multiple captures from the same map, along with the simultaneous operation of multiple maps.

3.2 Graphical user interface (GUI)

GUI is important because it is the visible part of our system and allows the user to perform all operations available in the application. An HTML page and JavaScript libraries have been used for development. These libraries are the Jquery framework (version 1.6.2) and Jquery-UI (version 1.8.16). Other developments using JavaScript libraries have been accomplished in order to add new functionalities to the created widgets. As the whole GUI runs on the client layer, usability and speed of response are guaranteed, and the adopted design is very flexible. The GUI has been developed in the form of a single HTML page, avoiding repeated requests and responses back to the server.

Several features within the HTML5 standard have been used in order to design the GUI for our system application. The most important object used is the canvas, which allows for an efficient management of the images to be processed. Pixel-level access to the content of a canvas object is possible, thus largely simplifying the implementation of image processing operations. The container can carry out map captures (snapshots), to access the information of each pixel of the capture, to transfer all such information to the server layer, and to save the obtained information (processed images). As noted above, a stack of images is obtained as an outcome with as many layers as the classes identified by the processing algorithms k-means and ISODATA. The obtained layers can be merged in order to simplify the interpretation of the obtained results.

Our application is fully accessible from mobile devices: although the application is developed to be accessed primarily from a PC browser, it is also operational on mobile applications, such as cell phones or tablets, as far as these devices use browsers that support HTML5. Fig. 3 shows an example of the designed GUI, designed for simplicity purposes. It consists of a single web page with a working panel, a container of maps, and a capture container. The work panel features the creation of maps, the update of a map's captures, the zoom level shift, and the selection of processing parameters for the k-means, ISODATA and spatial post-processing algorithms implemented for image analysis tasks in the current version. The map container can hold multiple maps of individual sizes, while the capture container allows for several captures of the same map. Different captures of the same map always have the same size as the original map size.

Finally, Fig. 4 provides a display of our application. Different layers are managed in this case. In this screenshot, we aim to exemplify how an image has been processed and several classes have been identified by one of the considered processing algorithms. The system can let users show, hide and merge different classes identified by such processing algorithms. The colors associated to these classes can also be edited and customized. The layers can be superimposed on the original image (capture) to be processed, thus generating a final product which comprises an unsupervised classification of a certain area whose location, size, dimensionality, zoom level, etc. are defined by the end-user.

3.3 Image processing

Two different modules deal with image processing in our system. First, the web server receives the image, processes it with the methodologies implemented in the system, and then forwards the obtained result to the client layer. Secondly, the compute server processes the image effectively (e.g. applying the considered clustering and spatial post-processing algorithms).

The main task of the web server is to receive requests from clients. Such requests are handled as follows. First, the client selects the image capture to be processed and an access to a canvas object in HTML5 is performed. From this access the image content is extracted using a method called toDataURL of the canvas object. In this way the image content can be encapsulated into an AJAX request together with all processing options, and sent to the compute server asynchronously using the HTTP protocol, and specifically a function called post.

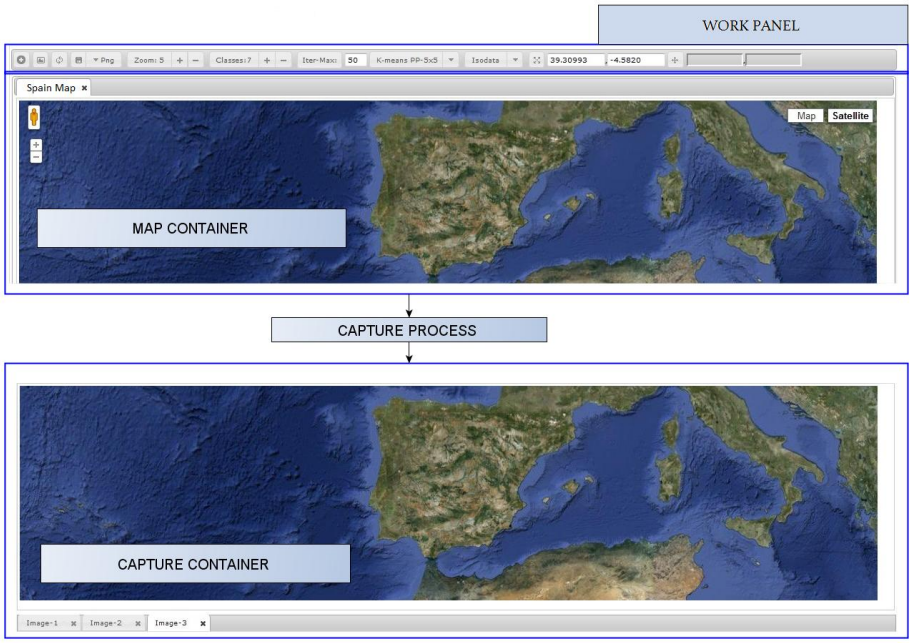

Figure 3. Graphical user interface in the application.

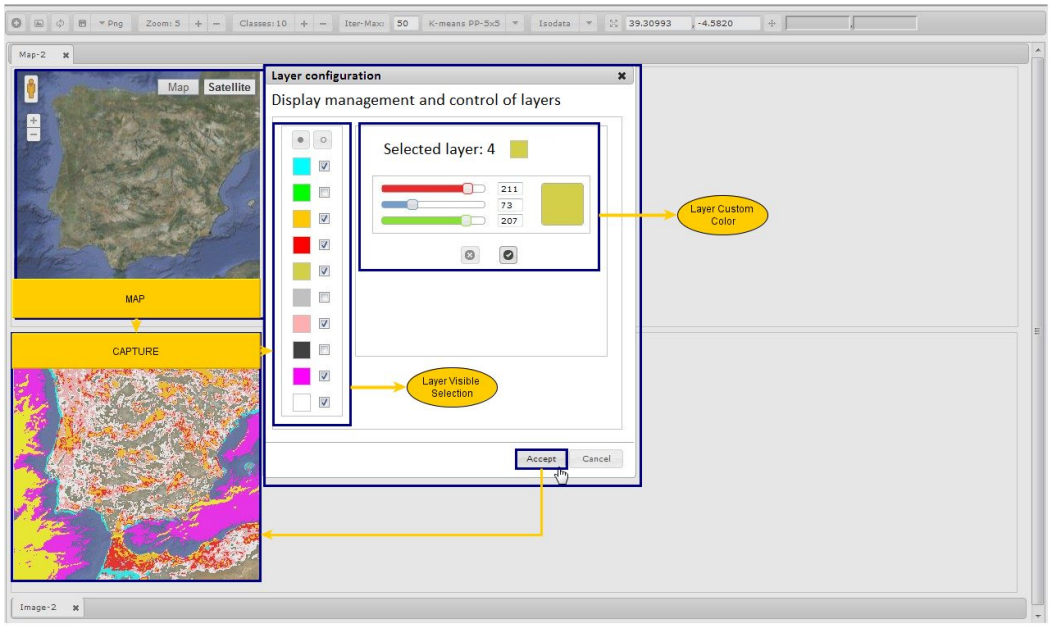

Figure 4. An example illustrating the management of different layers in the application.

Once the web server has received the full request, a PHP function is used for receiving the image to be processed and the parameters needed for such a task (number of classes, number of iterations, etc.) Then the compute server assigns a timestamp to the image to identify it as a unique entity. The image is stored onto a temporary folder, and the web server calls the compute server indicating the location and unique identification of the image, so that the data can be efficiently processed by the compute server (both the web server and compute server can access the image structure and its particular location). Finally, the compute server generates a final product (in our case, the processed image) from the information received, and stores the output on another

temporary location. After the processing task has been completed, the web server takes control again. It collects the final product generated by the compute server and sends it to the client in response to the original AJAX request originated at the client layer, thus closing the communication cycle with the client that originated the request.

3.3.1 Saving the final results

This part specifies how final results are saved, as this requires a special treatment in the implementation. Specifically, the generated product is not stored on any server when the processing is completed, and it is only located in the local memory of the browser at the client. The results can be expressed in different forms, e.g. as a processed image, as a collection of layers that can be superimposed with the original data set, or as a combination of both. Two specific actions are taken:

1. A JavaScript library (called canvas2image) saves the contents of the canvas object on the local device using different image formats, such as JPEG, PNG or Bitmap.
2. The combined result is saved after putting together different layers of the results inside the canvas object by applying another canvas container which integrates all the data layers to be displayed. Once the image is saved, the initial container is not retained. This process is transparent to the user and is also optimized from the viewpoint of computational performance.

4. EXPERIMENTAL RESULTS

In this section, we perform an experimental validation of the unsupervised classification features with and without spatial post-processing over the considered processing chain of our tool tested using satellite images obtained from Google Maps acrross different locations. The experimental validation of unsupervised ISODATA classification algorithm with and without spatial post-processing has been conducted by comparing the results provided by our implementations with those available in a well-known commercial software package: the Environment for Visualizing Images (ENVI) package distributed by ITT Visual Information Solutions.

In the following, we present the obtained results in a specific case study focused on the satellite-based image taken of Mérida, Spain [see Fig. 5(a)], offers a high spatial resolution of approximately 1.2 meters per pixel. The image was collected over the Roman Theater of Mérida (dating back to 16 - 15 BC, but renovated later on). The theatre is located in one of the most extensive archaeological sites in Spain. It was declared a World Heritage Site by UNESCO in 1993. The theatre was located at the edge of the Roman city near the walls. The grandstand consists of a semicircular seating area (cavea), with a capacity for 6,000 spectators eventually divided into three areas: the lowest tier called the ima cavea (22 rows), the medium tier called the media (5 rows), and a top tier called the summa, this one in less good condition. The Roman theater is the most visited monument in the city, and its festival classic theater is performed for the first time in 1933 and still continues today.

This monument has been chosen as an example of remotely sensed archeology, and we have decided to enhance a view offered by Google Maps to improve the visualization of the structure and the scale of this relevant monument for the region. Fig. 5(b) and (c) respectively show the unsupervised classification results obtanied from our processing chain using the ISODATA algorithm and the classification obtained from applying spatial post-processing (using a processing window of 3×3 pixels) over the classification result obtained from (b). Fig. 5(d) and (e) respectively show the unsupervised classification of the ISODATA algorithm and the classification obtained from applying spatial post-processing using the same parameters and the ones obtained from our implementation.

In this experiment, a fixed number of $c = 6$ and a window size of 3×3 pixels have been considered. Also, although the color class labels for the implementations are different, the classification map provided by our implementation (without spatial post-processing) and the ones obtained using ENVI are very similar. In both cases, the algorithm parameters have been set to exactly the same values. Table 1 reports the classification percentages of agreement measured after comparing our ISODATA classification maps with and without to apply a spatial post-processing. As shown by Table 1, the agreement between the maps is always very high (about a 90%). The confusion matrices for ISODATA and ISODATA with a spatial post-processing are respectively

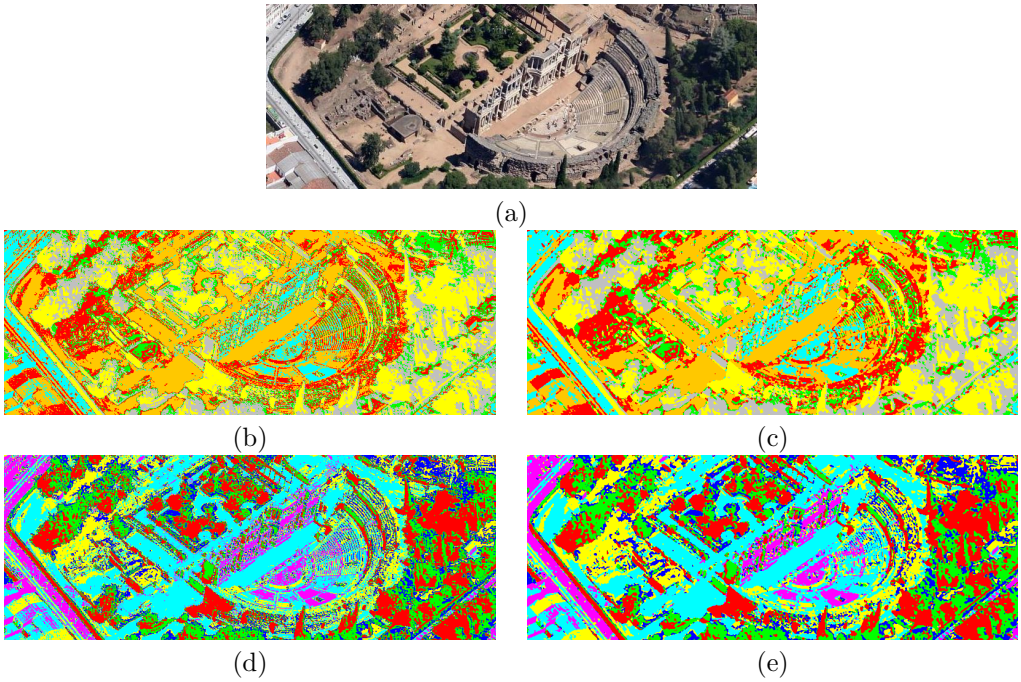

Figure 5. (a) Satellite image collected over the Roman city of Mérida, Spain. (b) Classification using our processing chain with ISODATA. (c) Classification using ENVI's ISODATA. (d) Classification using our processing chain with ISODATA with spatial post-processing. (e) Classification using ENVI's ISODATA with spatial post-processing.

Table 1. Percentage of agreement after comparing the classification map in Fig. 5(b), produced by our tool (with ISODATA), with the classification map in Fig. 5(d), produced by ENVI, and after comparing the map in Fig. 5(c) produced by our tool (with spatial post-processing) with the classification map in Fig. 5(e) produced by ENVI (also with spatial post-processing).

Clustering algorithm	Sand (Red)	Structure (Cyan)	Trees (Grey)	Shadows (Yellow)	Rocks (Green)	Pavement (Orange)	Overall agreement
ISODATA	87.90	89.23	93.64	100.00	84.56	91.33	91.11
ISODATA with spatial post-processing	79.48	91.09	87.02	98.42	79.32	89.36	87.45

provided in Tables 2 and 3. With this example, the potential of our proposed tool for perform classification with and without applying a spatial post-processing over a satellite image is shown.

5. CONCLUSIONS AND FUTURE RESEARCH LINES

This paper has described a new web-based system for computationally efficient processing of satellite images. The system, developed with the Google Maps applications programming interface (API), incorporates functionalities such as unsupervised classification for image portions selected by the user (at the desired zoom level) using the k-means and ISODATA clustering algorithms, followed by spatial post-processing. Most importantly, the processing of satellite images is conducted by means of a centralized server which receives the image to be processed, performs the analysis efficiently, and returns the classification result to the end-user. This represents an improvement over a previous development desktop application presented in.[9]

Our experimental results, conducted by comparing the obtained classification results with those provided by commercial products such as the popular ENVI software package, reveal that the proposed web-based tool provides classification maps with high similarity in relation to those provided by ENVI for the same satellite imagery, but with the possibility to perform the classification of any image portion available in the Google Maps engine.

In the future, we plan to incorporate other advanced classifiers to the proposed web-based system, such as random forests and Support Vector Machines (SVMs). Also, we would like to extend the developed tool with

Table 2. Confusion matrix obtained after comparing the classification map in Fig. 5(b), produced by our system (with ISODATA) with the map in Fig. 5(d), produced by ENVI.

Class	Sand (Yellow)	Structure (Magenta)	Trees (Grey)	Shadows (Red)	Rocks (Blue)	Pavement (Orange)
Sand (Red)	35,986	0	0	0	0	5,248
Structure (Cyan)	0	17,928	0	0	0	0
Trees (Grey)	0	0	40,335	0	5,187	0
Shadows (Yellow)	0	0	2,739	45,373	0	0
Rocks (Green)	4,951	0	0	0	25,947	0
Pavement (Orange)	0	2,088	0	0	0	55,298

Table 3. Confusion matrix obtained after comparing the classification map in Fig. 5(c), produced by our system (with ISODATA plus spatial post-processing), with the classification map in Fig. 5(e), produced by ENVI.

Class	Sand (Yellow)	Structure (Magenta)	Trees (Grey)	Shadows (Red)	Rocks (Blue)	Pavement (Orange)
Sand (Red)	32,503	77	336	137	494	4,935
Structure (Cyan)	515	17,547	122	30	246	924
Trees (Grey)	186	20	37,425	198	4,524	51
Shadows (Yellow)	60	12	3,703	46,062	199	26
Rocks (Green)	6,189	57	1,261	287	21,590	751
Pavement (Orange)	1,440	1,550	158	88	165	56,132

the incorporation of Content-Based Image Retrieval (CBIR) functionalities. For that purpose, the strategy will be based on a query system linked to feature extraction from an image repository (such as Google Maps). The retrieved features (which will comprise shape descriptors, texture features, etc.) will be stored on a database of features and used to compare the feature vector of the input query with those recorded by means of a similarity function. This facility will provide a result to the end-user in the form of image portions (across different locations) that have enough similarity in relation to the features of the input query.

Finally, we want to experiment with different forms of high performance computing architectures. The most promising strategy seems to be the parallelization of the different software modules for efficient processing in multiple cores as well as in graphics processing units (GPUs). Both mechanisms allow incorporation of high performance computing capabilities at relatively low cost in order to speed-up the computations performed by the developed tool.

Acknowledgment

This work has been supported by the European Community Marie Curie Research Training Networks Programme under contract MRTN-CT-2006- 035927, Hyperspectral Imaging Network (HYPER-I-NET). Funding from the Spanish Ministry of Science and Innovation (CEOS-SPAIN project, reference AYA2011-29334-C02-02) and by the Spanish Government Board (National Research Projects) and the European Union (FEDER founds) by means of the grant reference TIN2008-03063 are also gratefully acknowledged.

REFERENCES

1. R. A. Schowengerdt, *Remote Sensing: Models and Methods for Image Processing, 2nd ed.*, Academic Press: New York, 1997.
2. D. A. Landgrebe, *Signal Theory Methods in Multispectral Remote Sensing*, John Wiley & Sons: New York, 2003.
3. J. A. Richards and X. Jia, *Remote Sensing Digital Image Analysis: An Introduction*, Springer, 2006.

4. M. Fauvel, J. A. Benediktsson, J. Chanussot, and J. Sveinsson, "Recent advances in techniques for hyperspectral image processing," *Remote Sensing Environment* **113**, pp. 110–122, 2009.
5. A. Plaza, J. A. Benediktsson, J. Boardman, J. Brazile, L. Bruzzone, G. Camps-Valls, J. Chanussot, M. Fauvel, P. Gamba, J. Gualtieri, M. Marconcini, J. Tilton, and G. Trianni, "Spectral and spatial classification of hyperspectral data using svms and morphological profiles," *IEEE Transactions on Geoscience and Remote Sensing* **46**(11), pp. 3804–3814, 2008.
6. E. Quirós, A. M. Felicísimo, and A. Cuartero, "Testing multivariate adaptive regression splines (mars) as a method of land cover classification of terra-aster satellite images," *Sensors* **9**(11), pp. 9011–9028, 2009.
7. D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, "Active learning methods for remote sensing image classification," *IEEE Trans. Geosci. Remote Sens.* **47**(7), pp. 2218–2232, 2009.
8. A. Plaza, J. Plaza, and A. Paz, "Parallel Heterogeneous CBIR System for Efficient Hyperspectral Image Retrieval using Spectral Mixture Analysis," *Concurrency and Computation: Practice and Experience* **22**(9), pp. 1138–1159, 2010.
9. S. Bernabé, A. Plaza, P. R. Marpu, and J. A. Benediktsson, "A new parallel tool for classification of remotely sensed imagery," *Computers & Geosciences* **46**, pp. 208–218, 2012.
10. G. Ball and D. Hall, *ISODATA: A novel method of data analysis and classification*, Technical Report AD-699616, Stanford University, 1965.
11. J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society, Series C (Applied Statistics)* **28**, pp. 100–108, 1979.
12. P. Gamba, F. Dell'Acqua, F. Ferrari, J. A. Palmason, and J. A. Benediktsson, "Exploiting spectral and spatial information in hyperspectral urban data with high resolution," *IEEE Geoscience and Remote Sensing Letters* **1**, pp. 322–326, 2005.