REMOTE SENSING ANALYTICAL GEOSPATIAL OPERATIONS DIRECTLY IN THE WEB BROWSER

J. Masó [1], A. Zabala [2], I. Serral [1], X. Pons [2]

[1] CREAF, Fac Ciencies UAB, 08193 Bellaterra Barcelona Spain - joan.maso@uab.cat, ivette@creaf.uab.cat
[2] Dep. Geografia, Universitat Autonoma de Barcelona, Edifici B 08193 Bellaterra Barcelona Spain - Alaitz.Zabala@uab.cat, Xavier.Pons@uab.cat

Commission IV, ICWG IV/III

KEY WORDS: Remote Sensing, Analysis, Geospatial, Web, JavaScript.

ABSTRACT:

Current map viewers that run on modern web browsers are mainly requesting images generated on the fly in the server side and transferred in pictorial format that they can display (PNG or JPEG). In OGC WMS standard this is done for the whole map view while in WMTS is done per tiles. The user cannot fine tune personalized visualization or data analysis in the client side. Remote sensing data is structured in bands that are visualize individually (manually adjusting contrast), create RGB combinations or present spectral indices. When these operations are not available in map browsers professional are forced to download hundreds of gigabytes of remote sensing imagery to take a good look at the data before deciding if it fits for a purpose. A possible solution is to create a web service that is able to perform these operations on the server side (https://www.sentinel-hub.com). This paper proposes that the server should communicate the data values to the client in a format that the client can directly process using two new additions in HTML5: canvas edition and array buffers. In the client side, the user can interact with a JavaScript interface changing symbolizations and doing some analytical operations without having to request any data again to the server. As a bonus, the user is able to perform queries to the data in a more dynamic way, applying spatial filters, creating histograms, generating animations of a time series or performing complex calculations among bands of the different loaded datasets.

1. INTRODUCTION

The technology for sending maps on the web for visualization is now mature. On one hand, mass market tools like Google maps have popularized maps on the Internet and their APIs allowed for an explosion of myriad of implementations (Miller, 2006). On the other hand, the Web Map Server (WMS) standard (de la Beaujardiere, 2004) has made possible that interoperable maps from different origins can be combined and shown together in a single view. Recent web services crawlers has revealed and increasing number of map services. In 2010 a study detected 1126 WMS services (Li et al., 2010). A second study done only one year later using a different crawler technology revealed 3712 WMS services (Lopez-Pellicer et al., 2011) with almost no problems reported on accessing them. The later study also found that most of geospatial web services as map services (57%) and there is a lower interest in sharing the data, with more that 60% of services declared (WFS+WCS) but not active at the time of the study (only 909 downloading services active). There where only 17 processing services active at the time of the study. Web map services are used to serve remote sensing data and derived higher level products. Just to mention a couple of examples, NASA developed in 2002 the HDF-EOS Web GIS Software Suite (NWGISS) that included WMS support (Di et al., 2002) and, the web service for the Fire Information for Resource Management System (FIRMS); a product derived from MODIS data was released later (Davies et al., 2009).

Current map viewers that directly run on modern web browsers are mainly using 2 strategies. The first one is based on requesting the server an image that can fill the user view port. The image is generated on the fly in the server side and transferred in pictorial format that the web browser can display, such as PNG or JPEG (OGC WMS may be used to make this approach standard and interoperable). Secondly, small tiles that presented together form a map that is shown in the screen can be requested. Normally, tiles are prerendered or rendered on the fly in predefined styles and transferred in pictorial format that the web browser can display, such as PNG or JPEG. OGC Web Map tile Service (WMTS) may be used to make this approach standard. In both cases the server executes an internal portrayal algorithm that is applied to the internal data, to generate a data visualization result (that is transmitted to the client) in a lossy process that cannot be reversed by the client to get the values of the original data. A viewer that limits itself to what is possible to do with an standard WMS can only offer data visualization and simple point based queries (GetFeatureInfo).

Current limitations on the common usage of WMS services constrain users that are not able to download the data to visual analysis. Following the statistics provided by Lopez-Pellicer et al. (2011), there is only a 25% active downloading services in proportion of the active WMS number of services. There is almost no chance to process the data remotely given the low proportion of processing services. Assuming that the current situation persists, a possible solution could be to add extra functionalities to WMS clients and services to, at least, support some analytical capacity in a standard way.

A significant previous attempt to improve the current situation in the case of gridded data is ncWMS that introduced several extensions in GetFeatureInfo, symbolization and vertical and temporal dimensions (Blower et al., 2013). This paper presents an alternative: to enrich current web map clients by moving some of the portrayal capabilities from the server (such as Blower et al. (2013) approach) to the client. The proposed solution empowers the client that can request to the server arrays of remotely sensed gridded values for each band and store them in memory. This is possible with a very simple extension of WMS consisting in adding a binary raw data

format. In the client side, the user can interact with a JavaScript interface that encodes the portrayal functionalities and provide the same visual representation. With the binary raw data, the client can do much more than simple data visualization, such as simple statistical calculations on what is in the screen or pixel based operations among layers from different sources.

The proposed solution works better for information representing gridded data representing continuous variables, like the ones coming from remote sensing imagery and derived higher level continuous products as well as categorical maps that cover all territory (such as land cover maps), so this is the focus area of the presented paper.

2. METHODOLOGY

We propose and implement a new strategy were the server communicates the data values to the client in a format that the client can directly process using two new additions in HTML5: canvas edition and binary arrays.

If we follow the GIS and remote sensing optical tradition, 2d images are transported in a raster binary encodings listing the values of a georeferenceble grid. With time, several formats have been suggested that include sophisticated headers describing the content, some optimizations in terms of tiling, multi scales, multi dimensions, compression etc. The open source multi raster format GDAL library (Warmerdam, 2008), that is an engine for reading raster formats many GIS open source software usr, currently offers 155 drivers to deal with an equal number of formats, most of them binary encoded. In the proposed implementation of our map browser we have selected the simplest possible binary encoding, sometimes referred as "raw" format. In the raw format, there is no header that describes the file content. It directly starts by the sequence of the values of the grid, represented as an array of binary encoded numbers, all of the same binary type. Since the size of the selected binary type is fixed, there is no need to separate them with a marker. The assumption is that each cell in the grid contains a single value and the grid is described starting by the top left value providing all values of the first row of the grid immediately followed by the values of the second row, with no separation, and so on until the last value of the last row is provided. The data type, the size of the grid in terms of rows and columns and the georeference information (mainly a bounding box and a coordinate reference system) have to be communicated by other means.

The proposed JavaScript client is a classical map browser that has a map area where grid data at screen resolution can be presented. This area has a stack of several overlaying HTML transparent divisions of the exact same size, each one ready to represent the data coming from a layer of a WMS server. The map area is active and responds to the common actions of pan zoom and point query requested by the user mouse movements. A JavaScript detects the actions of the user and translates the in terms of changes in the extent (bounding box in geoferenced coordinates) of images needed. Then, the JavaScript code uses the bounding box coordinates and the fixed number of columns and rows to elaborate a request to a WMS service that will result in a response that contains the values corresponding one-to-one to the pixels of a map area in the screen, just by adding "format=application/x-img". WMS request is sent in an asynchronous mode using the XMLHttpRequest() function configured in a way that binary responses are allowed by means of this code:

```
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function(){
if (xhr.readyState ===
   XMLHttpRequest.DONE &&
   xhr.status === 200)
   {
          // xhr.response contains the
          // binary array
   }
}
xhr.open("GET", path, true);
xhr.responseType = "arraybuffer";
xhr.send();
```

A modified WMS server interprets the format MIME type and, instead of returning a image portrayal of the data (e.g. a PNG file), it generates a little-endian binary array as a response of the GetMap request. Using the new a JavaScript array buffer object, the data can be stored in a JavaScript variable an accessed by creating a new DataView() object. Then, depending on the data type of the binary values, an extraction of the value of a cell is done like this:

```
var dv=DataView(arrayBuffer);
var i_byte=(ncol*ifil+icol)
if (datatype=="uint16")
   getUint16(i_byte*2, true);
else if (datatype=="float32")
   getFloat32(i_byte*4, true);
else if ...
```

Next step should be to represent the data in the map area of the screen. A canvas area is created in a division inside the map area to be able to manipulate the presentation of this layer. By requesting to the canvas an image with createImageData(), an array of numbers is returned. The first 4 numbers (from 0 to 255) corresponds to the RGBA values of the first pixel (A is opacity), the following 4 numbers corresponds to the RGBA values of the second pixel and so on, covering the predefined set of pixels in the screen. The binary arrays returned by the WMS service are not directly appropriate for populating the RGBA values. In the map browser, at least one style was defined and associated to the layer. Styles are a set of rules that can be performed by the JavaScript client to transform binary arrays into the proper RGB values that inform about the colour intensity of an image that represents the data. In our map browser a style can have one of the following modes: colour map, RGB and calculation.

The first mode is the simplest one. A single binary array of values is used to generate the visualization by mapping the values to a colour map consisting on a predefined list of RGB values. In case that the value represents a category, the value of the cell is interpreted as an index into the list of RGB values (the colour map needs to have the same number of colours than the numbers of categories). In case that the value represents a continuous value, a linear transformation is done to map the minimum possible value in the binary array to the first colour of the colour map list and the maximum possible value in the binary array to the last colour of the colour map list (we call this operation a "rescale") (see

Figure 1). This mode will be appropriate for an NDVI image or for a land cover (LC) map.

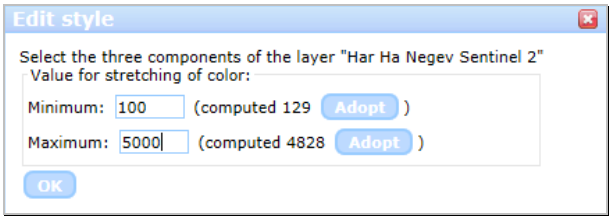

Figure 1. HTML representation of a JavaScript generated window to define the maximum value and the minimum value to stretch the colour map in an image with continuous values.

The second mode requires that 3 bands are requested to the server. The first returned binary array will be rescaled to represent the R colour channel intensity between 0 and 255 of all pixels of the screen; the second will be rescaled to represent the G colour channel and the third to represent the B colour channel. The individual bands can be obtained from 3 different layers of a WMS server or from the same layer but with an extra dimension parameter to extract individual bands (e.g. DIM-BAND=4). This mode is useful to represent band combinations (e.g. obtaining a natural colour image from a Sentinel 2 image by requesting the bands 4, 3, 2 from the RGB values) (see Figure 2 and Figure 4).

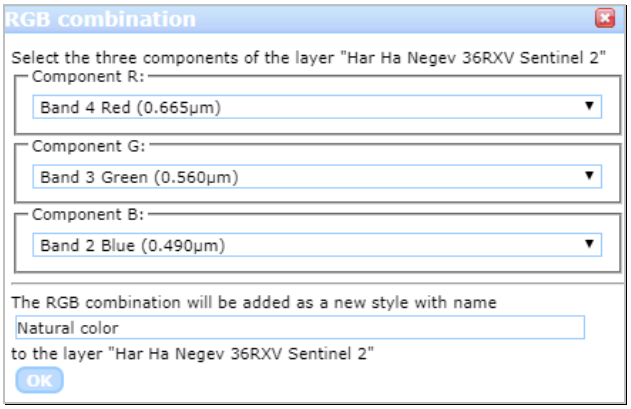

Figure 2. HTML representation of a JavaScript generated window to define new RGB combinations from individual components.

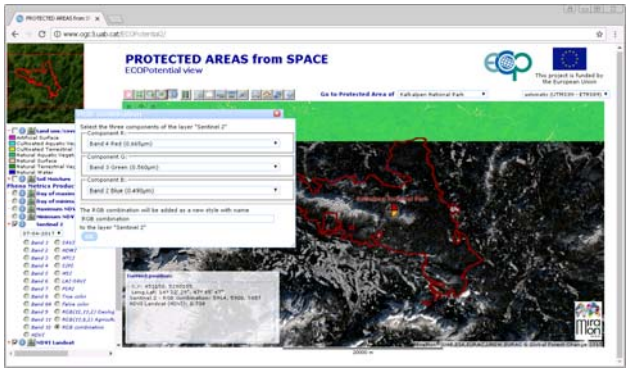

Figure 3. Screenshot showing an RGB on the fly combination from Sentinel-2 dataset for the Kalkalpen National Park for a particular date.

The third mode takes full advantage from the fact that the layers in the overlaying stack are correlated and since they requested at the same number of columns and rows and all cells have the same pixel size, a cell position of a layer is situated in the same place in the Earth. This means that the JavaScript code can perform pixel by pixel layer calculations using the actual values of the binary arrays and immediately present the result at screen resolution to the user. Calculations can be relatively simple spectral indices such as EVI or SAVI (Huete 1988) or a complex model involving several layers. Internally, the JavaScript code benefits from the eval() function: any calculation that can be expressed as a pixel by pixel mathematical expression can be executed by the JavaScript code (see Figure 4). The mathematical expressions will be encoded using a generic name of the layer bands. A loop for all pixels will be done and for each pixel, the generic name of the layer band will be replaced by the values of that pixel and will be given to the eval() function to get the result for this pixel.

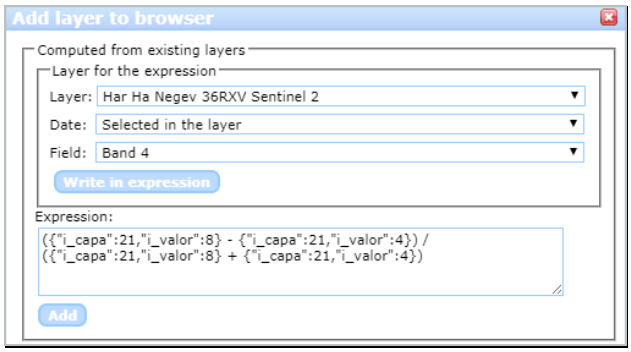

Figure 4. HTML representation of a JavaScript generated window to define calculations from existing layers. The example shows the definition of a Normalized Difference Vegetation Index using bands 8 and 4 of a Sentinel 2 image.

The capacity of having the actual arrays of values directly in the client side opens new set of possible applications that were not available with a classical WMS approach. If the binary arrays used for visualization are stored for later use, with a little bit of programming a histogram or a pie chart representing the area that is occupied by each rang of values or each class can be presented to the user. The statistical summaries can be easily calculated by the JavaScript code while one of the available diagram libraries can be used to plot the graphical representations. In our case, we opted for chart.js because it easily supports that each bar of a bar chart has a different colour; a characteristic that permits us to assign a bar the same colour as the central value of the bar receives in the map when a applying the styling rules. Another immediate operation that can be improved is the point query that will not need to rely on the GetFeatureInfo but can extract the queried value from the binary array, fast enough to show the value of all visible layers by just hovering over them with the mouse. This technology also allows for the capacity to filter out some values of the image that do not meet a condition set by another layer (e.g. represent NDVI values only if the elevation is lower a certain value, or only for a certain land use category) (see Figure 5). The style of the generated selection layer can also be edited to better adjust the visualization among the values.

Figure 5. A filter of the NDVI that has a DEM value higher than 2000 m for the Gran Paradiso Natural Park

With classical WMS services, the generation of animations of a time series is already possible in the server side by specifying a time interval in the extra TIME parameter. It is also possible to build a WMS client that requests individual time frames to a classical WMS and generates an animation directly on the client side by overlaying all frames and making only one visible in sequence at the right time. With the capacity of having binary arrays, and in addition to the animation, it is possible to present plots of the temporal evolution of one or more points in the animated area and eventually detect anomalies by comparing them with the mean and variance of the visible values in the bounding box.

In many occasions a pure raw format could result in a very repetitive encoding as contiguous pixels in an image tend to repeat the same value. For these cases, we needed to extend the application/x-img format with Run-Length Encoding (RLE), a simple encoding that stores repetitions in a more compact way generating a simple encoding simple enough to be implemented in real time JavaScript routines. In many cases, the reduction in bandwidth compensates the increase in the time spend by the JavaScript code to interpret the binary array. In this extension of the format, the following pattern is repeated: The first byte is the number of repetitions and the next bytes are the binary encoding of the value that is repeated (that might be an int16, float32, etc). If the number of repetitions is marked as zero this indicates a special case where the following byte specifies the number of non repeated numbers that will be followed by n byte groups representing the n values. In this implementation, the number of repetitions is a single byte, so it cannot exceed 255 repetitions. The start of a new image row shall be a number of repetitions (in other words, rows are compressed independently).

3. RESULTS

In ECOPotential, remote sensing data, in-situ data and modelling results are created and organized for the benefit of selected protected areas mainly in Europe. The technology described in this paper has been implemented in the MiraMon (Pons, 2002) map browser open source code (https://github.com/joanma747/MiraMonMapBrowser) and has been adopted as the project map browser in the Europena Union Horizon 2020 ECOPotential project (http://maps.ecopotential-project.eu). The *Protected Areas from Space* browser aims at providing a way to discover and explore the potential of remote sensing data for the management of 20 protected areas without the need to bulky downloads and setup complex remote sensing tools. After loading it in the map browser, a dropdown list, permits to zoom in and focus on one of the protected areas. The map browser adopts automatically the projection and coordinate reference system appropriated for that protected area. In this context, 129 layers have been processed so far, some of them consisting in time series and thus giving a total of 4102 time frames (see Table 1). The map browser allows discovering, visualizing, querying, animating and downloading; and also integrates metadata and quality descriptions associated to each layer.

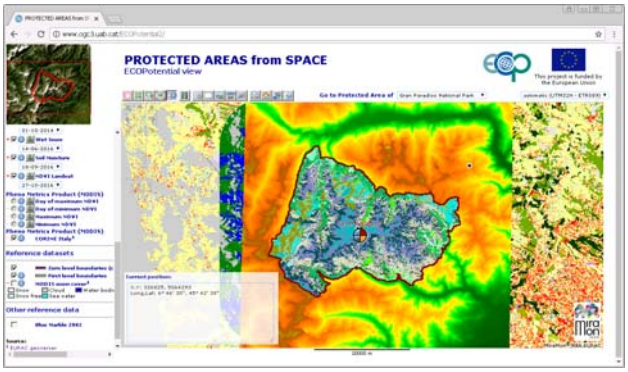

Figure 6. Screenshot for the Gran Paradiso National Park, showing, on the left, the legend with some of the layers, and in particular, the phenometrics layer.

Protected Area	Layer number	Time series frames
Abisko	5	75
Camargue	7	30
Curonian Lagoon	4	18
Danube Delta	8	57
Doñana	7	41
Gran Paradiso	10	415
Har Ha Negev	6	110
Hardangervidda	6	172
High Tatra	6	50
Kalkalpen	9	109
Kruger	4	115
La Palma	9	138
Montado	7	88
Murgia Alta	4	18
Ohrid Prespa Lake	6	26
Peneda Gêres	5	63
Samaria	8	159
Sierra Nevada	9	2358
Swiss National Park	6	43
Wadden Sea	3	17
Total	**129**	**4102**

Table 1. Layers and frames in time series processed by Protected Area

Regarding visualization, appropriate styles have been carefully crafted considering the characteristics and the semantics of the information shown. In some cases, more than one style is provided for the same data. In some others, styles are showing a different component of the same dataset; such as different spectral bands, different results of different methods to extract the same variables, or different variables related to the same concept. For example, in the *phenometrics* layer for *Grand Paradiso National Park*, the maximum and minimum NDVI value of the year, as well as the day of the year with when the maximum and the minimum NDVI occurred are offered as 4

layers (see Figure 6) but in each layer, the results of two methods (DLogistic and LinIP) are presented for the Terra and Aqua satellites as 4 styles. In the case of the *vegetation metrics* layer for the *Swiss National Park*, two styles are provided *above the ground biomass* and *canopy height model*.

When protected areas are analysed, they cannot be seen as isolated places but in relation to its surroundings (Hansen, 2007). That is the reason why most of the datasets included in the map browser are not limited to the protected area perimeter. In addition 2 additional layers have been created: a line showing the border of the area and a mask of the interior of the protected area. The later is not visible but it is available for analysis as will be explained below.

With histogram analysis users can easily identify the distribution of the values for a certain quantitative layer; for instance the NDVI values as shown in Figure 7. In this case, for the *Gran Paradiso Natural Park*, and for a November date, the maximum number of the values are around 0 (and with a second peak approaching to 1), which indicates a relatively good state of the vegetation. Pie charts are also a valuable visualization tool that allows users to quickly understand the representation of categorical variables, such as Land Use / Land Cover Maps (Figure 3). As a matter of example, for the *Doñana National Park*, is easy to realize that *cultivated and natural terrestrial vegetation* are the most common categories in the park, followed by *natural water* (what is expected since Doñana is a wetland ecosystem system). Attention is made to the point that the histograms and pie charts are dynamically computed on the client, and represent only the visible area in the browser, so zooming in to a certain interest area to obtain its histograms and pie charts is always possible.

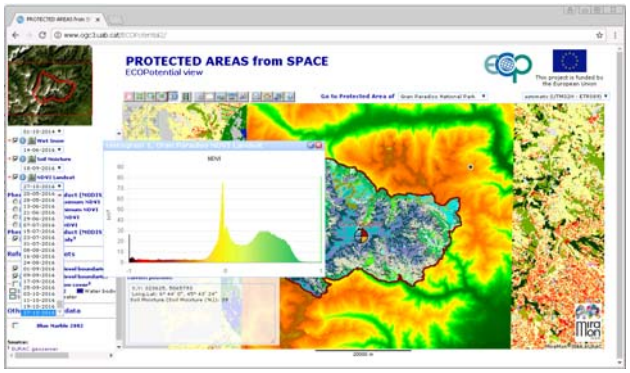

Figure 7. Screenshot showing the histogram representation over the NDVI dataset for the Gran Paradiso National Park for a particular date.

The produced histogram or pie chart covers the whole area visible in the map browser. This means that it might include the protected area and the surroundings. To prevent the surroundings to contaminate the analysis a *selection* operation can be done to exclude all area outside the park. This is done by filtering out the pixels that are not on top of the mask of the park. The selection is done directly in client side. In Figure 9, we can see that the pie chart of the park and its surroundings has a much bigger presence of cultivated areas than the interior of the park alone.

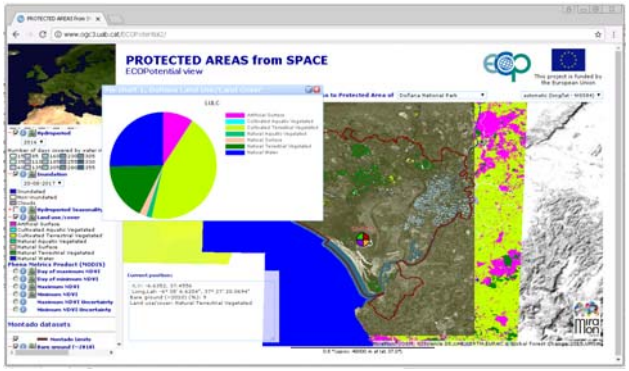

Figure 8. Screenshot showing a pie chart representation for a categorical dataset (a Land Use Land Cover Map) for the Doñana National Park.

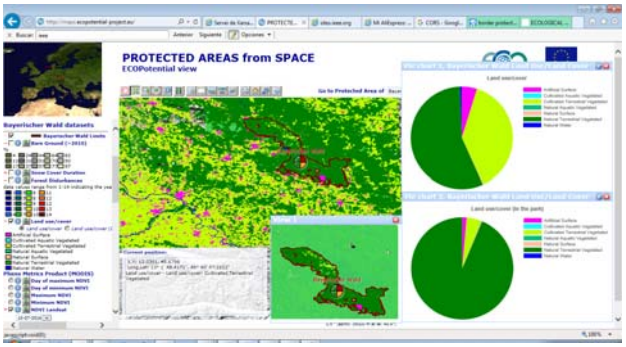

Figure 9. Screenshot showing a pie chart of the land use categories of the Bayerischer Wald Park considering the surroundings of the park (upper part) and only the park area (lower part).

The combination of this feature with the *selection* one opens a wide range of applications allowing the user to perform complex queries on the browser such as: how the percentage of some habitats categories varies with altitude? Which is the NDVI histogram for crop areas in a certain protected area and its comparison with the NDVI for the whole protected area? The answer to these questions can be obtained with applying a few operations in the map browser. Thus, analytical queries can be done to extract the NDVI in areas of *Gran Paradiso National Park* over 3000 meters (Figure 10, selection on the left), and under 1500 meters (Figure 10, selection on the right).

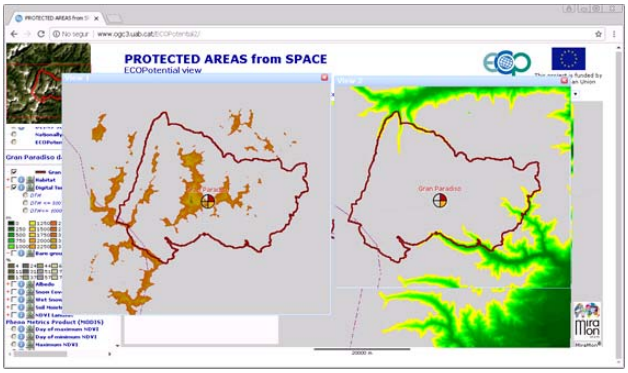

Figure 10. Selections from the Gran Paradiso Natural Park DEM. On the left, areas over 3000 m, and on the right, areas under 1500 m.

Figure 11. Pie chart for the Gran Paradiso Natural Park Habitats over 3000 m.

Among these selections, habitats distributions can be extracted, showing pie charts on the proportions of each present categorical variable. Different habitat distributions can be observed from the Figure 11 pie chart (habitats over 3000 m) and the Figure 12 pie chart (habitats under 1500 m) and we can realize that altitudes under 1500 m have a bigger variety of habitat types that the altitudes above 3000 m.

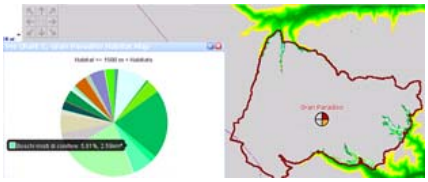

Figure 12. Pie chart for the Gran Paradiso Natural Park Habitats under 1500 m.

Analytical queries over the histogram for the NDVI can show the differences among areas. For *Har Ha Negev* (mostly a desertic area) Figure 13 shows the histogram for the Sentinel-2 NDVI for the whole protected area, while Figure 14 only presents the values of NDVI that are *terrestrial cultivated* category (by creating a selection of the NDVI values only over the LULC layer).

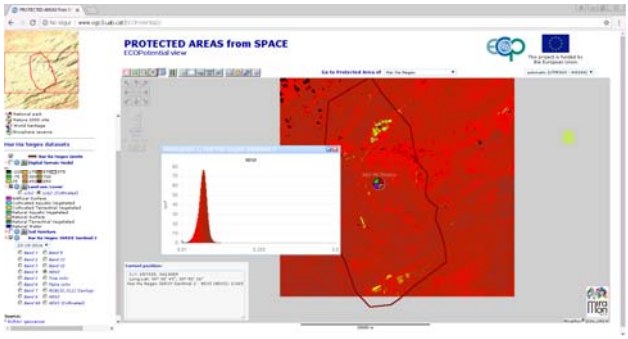

Figure 13. NDVI histogram over the Har Ha Negev Protected Area.

A different visualization of the same concept can be done by asking the client to load the entire Sentinel-2 NDVI time series and perform a video animation. This can be done over the entire area, or just over the selection previously done for the *terrestrial cultivated* areas.

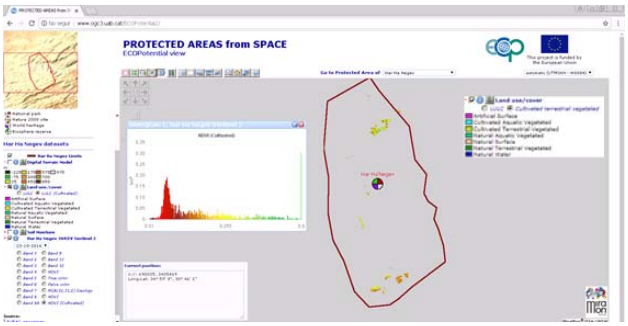

Figure 14. NDVI histogram over the Har Ha Negev Protected Area, only for the terrestrial cultivated category extracted from the LULC layer.

While showing the animation, the user can click on a particular pixel and get a graphic representation of the complete series and compare the shape of the curve at different points (see Figure 15 and Figure 16). In particular, we can see the opposite behaviour in the NDVI for the points A and B probably related to different agricultural practices.

Figure 15. Time series video for the Sentinel-2 NDVI on the small area of the terrestrial cultivated LULC category.

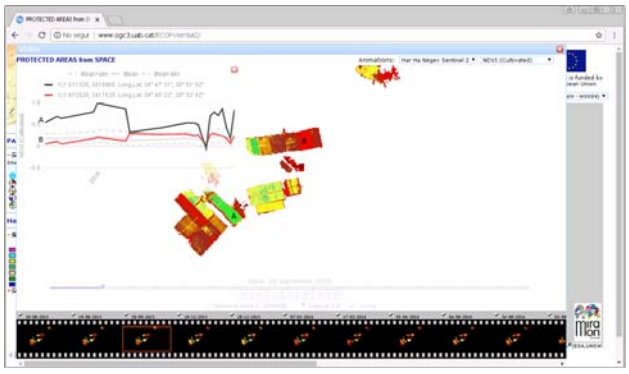

Figure 16. Clicking on particular points, a graphic on the evolution of the NDVI for the whole series can be derived. In this case, the comparison is among points A and B.

4. LIMITATIONS

The main limitation of this approach is that the operations described before are done at the screen resolution. In practice, this means that you can only cover a big area at a coarse resolution. This has no impact in the visualization, since the data cannot be presented at a better resolution than the one in

the screen, but can have impact in other operations. Indeed, the exact values of the statistical calculation of the area of each class represented in histograms and pies depend on the cell size and the generalization algorithm done by the server when responding to the client.

The second limitation is due to the JavaScript execution engine speed. JavaScript was initially intended to deal with simple tasks in the client such as validation of HTML FORM user inputs to avoid unnecessary communication to the server of FORM inputs that are clearly wrong and can be filtered out. With time, both JavaScript language definition and JavaScript execution engines have been evolving. The language rapidly evolved into a much more complete and engines became much faster. Soon it was clear that it was possible to encode sophisticated applications in JavaScript (Bradenbaugh J, 1999). Still there are characteristics of the language that makes it difficult to optimize such as the absence of numerical data types that are related with the CPU architectures and that are common on other languages (such as long integers and double precision floats) (Wagner, 2017). In our case, the first implementation of the routine that takes a binary array and converts it in the numerical RGBA values array needed for the canvas rendering was unacceptably slow taking 5-6 seconds to process a window of 1000x800. We did considerable progress by optimizing the code and we were able to reduce time below 1 second in applying a colour-map and 1-2 seconds in generating an RGB combination from 3 binary arrays. For the moment, the performance is acceptable for real time visualization but there are two competing factors that might affect the future adoption of the presented approach: Desktop screens are becoming bigger and pixel size in mobile phone screens is decreasing fast. In the near future, map windows will require bigger number of pixels to be processed, making the described conversion slower. At the same time, JavaScript engines are becoming better and could compensate the effect.

There is also another technical limitation that is originated by the web browser. In order for the JavaScript to be able to read and operate with the array buffer, both the client and the server need to be in the same server. If they are from different servers, the web browser prevents the data from being read for security reasons. Cross Origin Resource Sharing (CORS) offers an escape to this situation but it is up to the server to implement it. This is a limitation that is not present in classical WMS clients showing PNGs and could prevent to overlay array buffers from different origins conditioning the interoperability of the whole approach. That is why is especially important that CORS solutions are applied in the server that implements the WMS array buffer extension. The solution consists on the server returning the right headers Access-Control-Allow-Origin and Access-Control-Allow-Methods. Our implementation of the server takes care of CORS to allow other clients to be able to interoperate with our server data using the approach presented in this paper.

5. CONCLUSIONS

We propose a new way of implementing map browsers on the web, based on some new characteristics of HTML5 such as canvas and array buffers. This solution permits to move some functionality to the client side reducing the number of interactions with the server and giving more analytical tools to the user. The solution focuses on the requirements of data that is continuous describing a given area, such as the remote sensing raw data and higher level products, including both categorical and continuous values.

The proposed solution is still based on the interoperable and mature OGC WMS service and only requires a reinterpretation of the outputs expected from a GetMap request. In the future, this could be described in a small extension of the current version of WMS describing how to provide arrays of binary data. In practice, any WMS service could adopt the proposed solution with only small modifications.

Results show that JavaScript is able to perform fast enough to process the necessary information to populate the canvas RGBA array that is rendered in the screen and to permit a fluid interaction with the user. The solution gives the user much more flexibility in the way colour-maps and RGB combination can be defined and, in addition, the client can offer a new set of operations that were not possible with a classical implementation of a WMS client. This solution provides a much more convenient client side implementation of the point query operation that replaces and removes the need to send a GetFeatureInfo request for each pixel queried.

Doing analysis at screen resolution has advantages and disadvantages. The main advantage is the minimization of the necessary bandwidth while maintaining visual quality and providing some fast analytical tools for exploring the data. The main drawback is that only a subset of analytical operations is meaningful at screen resolution. Even if it is meaningful, the exact result may vary with the "zoom level" of the data presented and should be interpreted with caution. The role of such analysis should be exploratory and final results should be confirmed by downloading the data at full resolution and repeating them with desktop remote sending or GIS tools.

ACKNOWLEDGEMENTS

This work was supported by the European Union's Horizon 2020 Programme [ECOPotential (641762-2)]; Spanish Ministry of Economy and Competitiveness [ACAPI (CGL2015-69888-P MINECO/FEDER)]. Xavier Pons is recipient of an ICREA Academia Excellence in Research Grant (2016–2020).

REFERENCES

Blower J. D., Gemmell A. L., Griffiths G.D., Haines K., Santokhee A., and Yang X., 2013, A Web Map Service implementation for the visualization of multidimensional gridded environmental data. *Environmental Modelling and Software* 47, pp. 218-224

Bradenbaugh J., 1999, JavaScript Application Cookbook by Paperback Book, O'Reilly Media, Incorporated ISBN: 1565925777

Davies D.K., Ilavajhala S., Wong M.M., and Justice C.O., 2009, Fire Information for Resource Management System; Archiving and Distributing MODIS Active Fire Data, *IEEE Transactions on Geoscience and Remote Sensing*, 47 (1) pp. 72-79

de la Beaujardiere, J. (2004) OGC Web Map Service (WMS) Interface, Ver.1.3.0, OGC 03-109r1. Available from: http://portal.opengis.org/files/?artifact_id=5316.

Di L., Yang W., Deng M. and Deng D., 2002, Interoperable Access of Remote Sensing Data Through NWGISS. *IEEE International Geoscience and Remote Sensing Symposium.* DOI: 10.1109/IGARSS.2002.1025004

Andrew J., Hanse A. J. and DeFries, 2007, Ecological Mechanisms Linking Protected Areas to Surrounding Lands. Ecological Applications, 17, 4, pp. 974–988.

Huete A. R., 1988, A soil-adjusted vegetation index (SAVI), *Remote Sensing of Environment*, 25, 3, pp. 295-309.

Li W., Yang C. and Yang C, 2010, An active crawler for discovering geospatial Web services and their distribution pattern – A case study of OGC Web Map Service, *International Journal of Geographical Information Science*, 24:8, 1127-1147, DOI: 10.1080/13658810903514172

Lopez-Pellicer F. J., Béjar R., Florczyk A. J., Muro-Medrano P. R., and Zarazaga-Soria F. J., 2011, A Review of the Implementation of OGC Web Services across Europe, *International Journal of Spatial Data Infrastructures Research*, 6, pp. 168-186.

Miller C. C., 2006, A Beast in the Field; The Google Maps Mashup as GIS_2, *Cartographica*, 41 (3) pp. pp. 187-199.

Pons, X., 2002, MiraMon. Sistema d'Informació Geogràfica i software de Teledetecció Centre de Recerca Ecològica i Aplicacions Forestals, CREAF. Bellaterra. ISBN: 84-931323-4-9,

Wagner L., 2017, WebAssembly Will Finally Let You Run High-Performance Applications in Your Browser. *IEEE Spectrum*

Warmerdam F., 2008, The Geospatial Data Abstraction Library, pp 87-104. In: Hall G.B., Leahy M.G. (eds) *Open Source Approaches in Spatial Data Handling. Advances in Geographic Information Science*, Vol 2. Springer, Berlin, Heidelberg