Vector Pyramids: A Multi-scale Vector Rendering and Processing Algorithm

Mohammed Rashad[1], KS Rajan[2]
Lab for Spatial Informatics
International Institute of Information Technology
Hyderabad, India
[1]mohammed.rashad@research.iiit.ac.in , [2]rajan@iiit.ac.in

Abstract

Visualization of large spatial data is a complex process and many GIS packages use their own assumptions or special storage formats to render quality data. While Image pyramids and tile based rendering are used for raster data visualization, vector datasets use feature limiting mechanism for progressive rendering. For vector datasets these methods improve rendering but have very little visual appeal and make visual interpretation difficult. This paper proposes a method for rendering Vector data, called Vector pyramids, that is designed to improve rendering and visual interpretation of large vector datasets. The method is a combination of geometric aggregation and attribute level amalgamation to generate multi-scale data that can be appropriately called for rendering based on both the zoom level and map display extent. Vector pyramids is an automatic method, that mines the attributes of the thematic layer data itself to suitably amalgamate and hierarchically rank them. For a dataset with more than 200,000 features, the results of the algorithm show that Vector pyramids perform at orders of 2 to 14 times faster than current implementations of vector data rendering, depending on the zoom level and map display extent, while improving the data visualization quality. The algorithm is suitable for both desktop or web-based rendering of geospatial data.

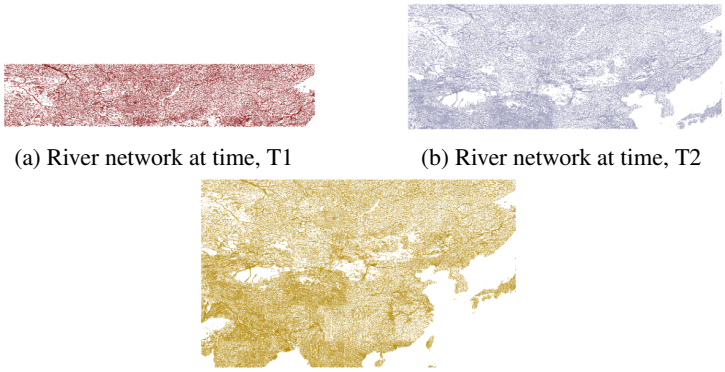
Keywords

Attribute ranking, Generalization, Rendering, Spatial data handling, Vector

1 Introduction

Data visualization is one of the very first steps in handling geospatial dataset. Size of data plays an important role in this visualization process irrespective of platform of choice as spatial data is stored in a variety of formats and nature of storage highly depends on the selected format. For instance GML2.0 format doesn't include topology but the same dataset in GML3.0(OGC 2002) format takes more disk space due to the storage of topology within the same file. Even with the increasing computational power in these systems large datasets are computationally expensive and every GIS packages has its own model of data rendering, while trying to keep a balance with the data quality. Though visualization of data implies the on-screen display of the data contents, the model of rendering the data can vary from simple geometry display to symbology and labeling in a way appropriate to the end user needs.
Most GIS packages use a feature count based filtering approach for vector data, hereinafter referred to as feature-count approach, to progressively load the geospatial data by reading or accessing a limited feature set that is pre-defined or randomly selected from the entire data and loaded at a time, sequentially till the whole dataset is displayed. Once such data loading is done, based on the user zoom level and the map extent, the requisite data is fetched from the primary or secondary memory and displayed on-screen. This technique is used in both desktop and web based GIS applications. While, this does help render the data, the problem with the feature-count approach is that the first and successive N features that are selected may have no relation to the users' area of interest within the map that the user wants to visualize. For instance, given a single thematic layer of *river network* with say four categories of *river_types : main rivers, perennial, non-perennial and drainage channels*; the feature-count approach will scan first N features which may be a mix of all categories of rivers which will result in showing up of parts of the entire data at a time, as seen from Figure 1, and not based on either the data category or area of interest.

In case of large raster data display, the feature-count approach does not work and so, the commonly adopted approach is to use the image resolution as the filtering key leading to a sub-sampled and smoothened image, usually known as Image Pyramids(Anderson et al. 1984). While in some other cases, based on the model of data storage like BIL(ESRI 1999) an interleaved image can be progressively displayed as the data is read from the memory. This may or may not be tied to the zoom level related to the display extent.



(a) River network at time, T1

(b) River network at time, T2

(c) River network at time, T3

Figure 1: Feature-count rendering of river network dataset (DCW 2013) at different instances

With the change towards more and more use of web-based geo-applications and services, visualization of the geospatial data within a browser has gained importance. But, most of the initial and current developments in within-browser display, irrespective of the data model- vector or raster, are based on generating raster images in the form of png, jpg, etc.(Shekhar et al. 2006), at the server-side and posting them in the browser using Web Map Serives (WMS) (OGC 2001) and WMTS protocols (OGC 2010). These map-servers internally use a pre-defined logic of tying the data to be displayed with the requested zoom-level and are similar in approach to the feature-count method. For example, in WebGIS applications such as OpenStreetMap (Haklay & Weber 2008), the *Slippy Map* technique is used which stores generated image tiles based on a hierarchical model tied to the zoom level and resolution (Haklay & Weber 2008). Even though the OGC defined WFS offers direct fine-grained access to geographic information at the feature and feature property level (OGC 2005), the implementation has no control over the level of detail being rendered. Though WFS provides extent based clipping but if the user request is a global extent the data may be too crowded and visual inspection and interpretation can be very difficult. A work-around that can be implemented is by pre-processing the data to create singe category attribute based thematic layers and call the appropriate layer while displaying at a given zoom level, a similar approach was advocated by (Bertolotto & Egenhofer 1999). For example in the above stated *river* thematic layer, based on each *river_types* attribute, it can be separated into four different layers such as *major_rivers, perennial_streams, non_perennial_streams and drainage_channels* and say at the coarser zoom levels, only the *major_rivers* geometry with its attributes is called for display. This work-around may not be suitable

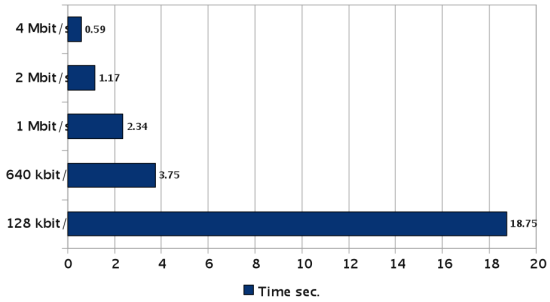


Figure 2: Values of bandwidth vs Time taken to deliver a 300,00 response payload (Source: http://opengeo.org, May, 2013)

for varied reasons ranging from increased data redundancy to making the data unsuitable for geospatial analysis.

OGC Web Feature Service allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Servers. The minimal requirements of a Web Feature Service is that the interface must be defined in XML and features must be expressed in GML(OGC 2002). When the size of the dataset increases most of WFS implementations will not be able to handle it perfectly. This is because HTTP request returns *timed out* if the response data size is large, inspite of having a reasonable network bandwidth, and this is an issue while handling large volumes of data rendering. Most commonly used workarounds are: Limit response size, Optimize request response, Allocate network bandwidth(GeoServer 2013). Limiting response size is done by putting a maximum number on the features being retrieved in a single HTTP request. Features pulled from a WFS server in this scenario maybe fully irrelevant to what user is looking for in the data. This is because limiting factor doesn't consider hierarchy/rank of attributes. Optimizing the request response can help in such situations where large data is handled. When the request response becomes heavier, WFS server write the response to a GIS format (eg: GML, GeoJSON etc.) and returns the path to dataset. The dataset will also be compressed before being written to disk if the selected format supports compression(eg: zipped shapefiles). Higher bandwidth networks can transmit more features than lower bandwidth in a single request. Figure 2 shows various values of bandwidth and the time taken to deliver a 300,000 response payload. (This is a typical screen-sized WMS image, or 8,000 WFS features in GZIP-compressed format.) (GeoServer 2013). It should be noted that no one solution can be applied to every case and it depends on the availability of resources. Allocating higher bandwidth may not be possible for

many projects if the dataset grows daily on an exponential basis such as in crowd-sourcing projects. The rendering of data is not of importance nor considered in any of these workarounds used by current WFS Implementation. So there us a need to improve the WFS mechanism that not only has a control on the data relevance per request but also accounts for the visual quality of the data rendered.

On the other hand, cartographic generalization techniques (McMaster & Shea 1992, Worboys & Duckham 2004) have been used for improving the quality of the map display by processing the map geometry data. The choice of the generalization techniques like simplification, aggregation, exaggeration, etc.(Longley et al. 2005), is largely based on the cartographic or thematic needs of the user and/or based on the map scale/resolution. These methods are still largely based on human intervention and are not suited to the evolving WebGIS needs for an on-the-fly rendering of the user-appropriate geo-content based on not just the geometric information but also the categorical attributes present. Though the Encoded Polyline Algorithm (Google 2013) of Google Maps is a simplification method where the location or geometry is tied to the zoom level, the related attributes that are displayed do not seem to be logical, hierarchical or preferential. With increased move towards GIS-on-the-web, there is a need to generate automatic or semi-automatic generalization techniques that can provide better geospatial data visualization and rendering experience to the user.

With the advent of new web technologies such as HTML5 (W3C 2010), rendering within a browser can now be a client-side processing activity rather than a server-side image generation and posting to the browser. The client-side rendering is possible using the technologies like Canvas, SVG and VML, which can draw the vector features within a browser. While HTML5Canvas (W3C 2012) follows a pixel based approach and is suited for raster rendering, SVG allows to draw geometries (point/line/polygon) without converting them to pixels and maintain the Document Object Model (DOM) elements representing the geometry and associated styling information (Rashad & Rajan 2012). This allows the browser client to interact with the vector features without making a request to server. These capabilities make it possible to incorporate the data generalization techniques to develop on-the-fly multi-scale vector rendering algorithm without disturbing the primary or original data.

1.1 Objective

This paper presents *Vector pyramids* algorithm that aims to improve vector data rendering performance by selecting appropriate features to display based on its cat-

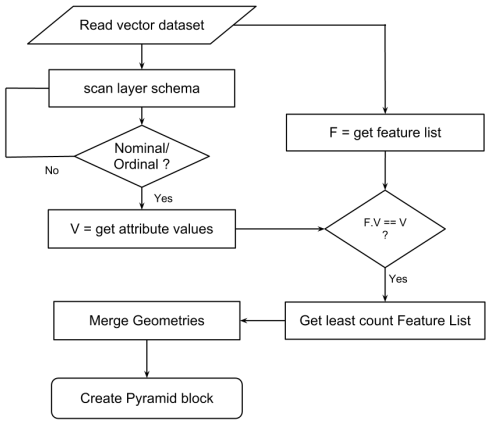


Figure 3: Flowchart - Attribute Ranking

egorical attributes rather than depending on a random feature count. The algorithm mines the attributes of the spatial data itself to automatically rank the features and links them to the rendering pipeline. The subset or part of the spatial data, though will load less features, shall show data based on a hierarchical flow of data detail, relevant to the user. It is expected that this will also lower the computational costs of rendering as the size of data to be displayed is tied to the display extent with less features than the total feature count. This automatic generalization of vector data will also improve the visual representation of the features at every scale by controlling the level of detail being rendered.

2 Vector Pyramids

Vector pyramids algorithm consists of two stages: Attribute Ranking as a pre-processing stage and Spatial filtering during rendering run-time. The overall flowchart of the pre-processing pipeline is shown in Figure 3. The respective stages are described in the following sections.

2.1 Attribute Ranking

2.1.1 Attribute Field Identification

First step in attribute ranking is to collect all the attribute field information. Given the Layer's schema definition as input, it is scanned using the GDAL/OGR (GDAL.org 2013) library to check for relevant attribute field which can be used to group the

features. The relevant attribute fields are those with either nominal or ordinal values.

2.1.2 Attribute Value Set

Attribute value set is a set of features having the same value for a selected attribute field and is a subset of the original master dataset. This is based on attribute value matching from the stored list created in the previous step. Each set contains information about the feature geometries that belongs to it. This process is repeated for all the unique attribute values. Post this, based on the number of features contained in each set, the sets are ranked to form a hierarchical chain. For instance, Table 1 shows a rank hierarchy for a sample river network, where major rivers having the least feature count has the highest rank. This ranking is later used to associate the display zoom levels with pyramid blocks. These sets can be either stored in primary or secondary memory depending on the choice of the pre-processor configuration.

Table 1: Rank Hierarchy

River Type	Rank
Major Rivers	4
Perennial Streams	3
Non-Perennial Streams	2
Drainage Channels	1

2.1.3 Aggregation

Each Attribute value set generated above is passed into the aggregator which merges the geometries of all those features that belongs to it. Union operation available through OGR API (GDAL.org 2013) is used for this step. An updated attribute table is generated for the respective sets with its value, an index attribute and feature ID's of the merged geometries. This provides access to the features in the original master dataset for querying purposes later.

2.1.4 Mapping Pyramids

Aggregator produces a set of geospatial vector datasets, along with its rank hierarchy. The datasets corresponding to the respective ranks are mapped automatically to one or more zoom levels depending on the total number of zoom levels configured. The set with the highest rank is mapped to the zeroth zoom level. These

ranked datasets can be further stored in the secondary memory to speed up rendering in GIS packages or toolkits that can read OGR supported formats.

2.2 Spatial Filter

Spatial Filter is primarily a display extent clip operator that removes any excess features which does not fit in the current display extent. This further reduces the size of the feature set to be rendered and on each zoom/pan the spatial bounds of the display extent are recalculated appropriately. *SetSpatialFilterRect*(GDAL.org 2013) method provided by GDAL/OGR is used in this step. At very high zoom levels even though the feature count of the subset is almost equal to the total feature count due to the spatial filter, only a small portion of the dataset is taken into account for rendering.

Vector pyramids algorithm has been implemented in C++ and uses GDAL/OGR library for some of the processing tasks.

3 Data description

Two different sized datasets where chosen to evaluate the proposed Vector pyramids algorithm and its performance.

3.1 Dataset #1

This dataset consist of Administrative Boundary level 2 information of India. The data was available in ESRI Shapefile (ESRI 1998) format and contains 599 polygon features. It contains country, state and districts attribute fields. The spatial extent of dataset is (68.1628E, 6.74714N, 97.4033E, 37.097N) and spatial reference system used is EPSG:4326. Table 2 gives the layer schema and the dataset has a total size of 2.4 Megabytes. Figure 4 shows a full rendered view of this dataset.

3.2 Dataset #2

Global Administrative Areas(GADM) Version 2 (GADM.org 2013) dataset is used as another test data in this work. It consist of World Administrative Boundary with country, state and district boundaries. Dataset also contains sub-administrative details for some areas wherever available. The data is available in ESRI Shapefile format and contains 2,18,238 polygon features. Dataset has spatial extent of

Figure 4: Dataset #1 (count = 599)

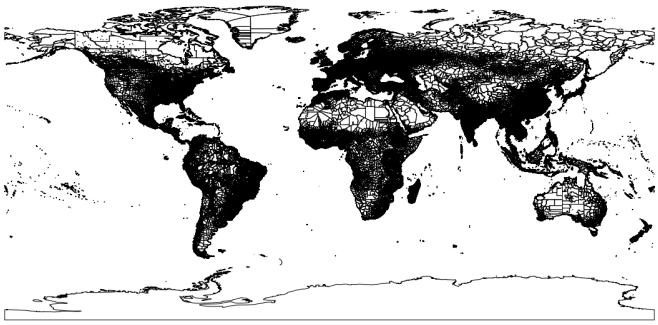


Figure 5: Dataset #2 (count = 2,18,238)

(-180E, -90N, 180E, 90N) and spatial reference system used is EPSG:4326. A relevant snapshot of layer definition is given in Table 3. Total size of the dataset is 1.50 Gigabytes and Figure 5 shows a full rendered view of this dataset.

4 Results

The proposed Vector pyramids has been tested on the two datasets, mentioned in the earlier section. Attribute Ranking on dataset #1(India) generates three subsets of the dataset. At first level only the country's National administrative boundary, the highest level, is available. This level consists of a single large polygon as shown in Figure 6b and if rendered without Vector pyramids all features are rendered as

Table 2: Layer Schema of dataset #1

Id	Name	Type	Length
1	NAME2	String	25
2	COUNT	Real	11
3	STATE	String	70
4	DISTRICTS	String	21
5	COUNTRY	String	16
6	AREA	Real	16
7	PERIMETER	Real	16
8	HECTARES	Real	16

Table 3: Layer Schema of dataset #2

Id	Name	Type	Length
1	ID_0	Integer	9
2	ISO	String	3
3	NAME_0	String	75
4	ID_1	Integer	9
5	NAME_1	String	75
6	VARNAME_1	String	150
7	NL_NAME_1	String	50
8	HASC_1	String	15
9	CC_1	String	15

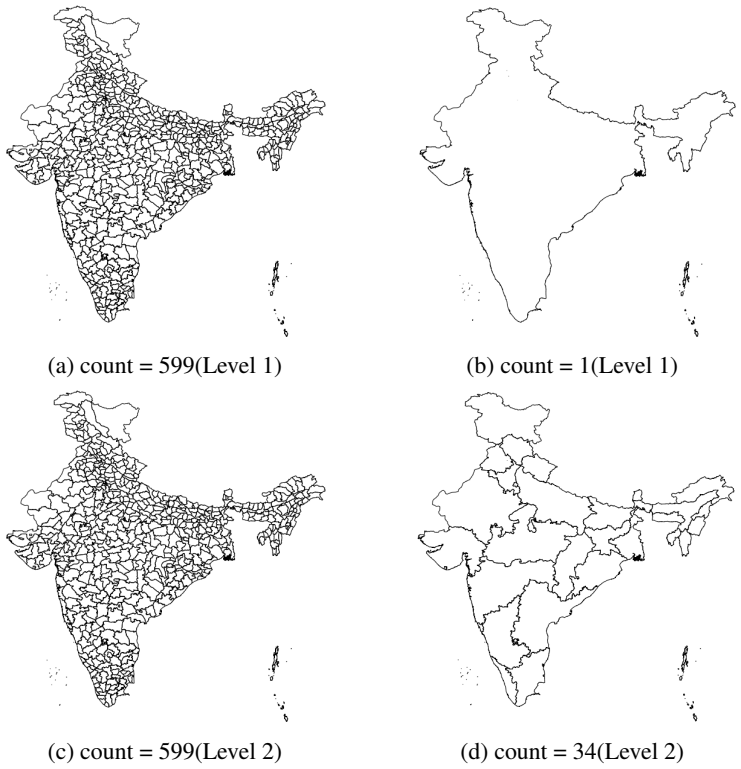
shown in Figure 6a. As can be seen from the figures, the feature details cannot be differentiated between the state and district boundaries even though boundaries are drawn. But user is able to identify country's administrative boundary in the first zoom level in both the cases. Using ranking information created by Vector pyramids algorithm, the output layer is able to skip those features that can cloud the feature set information and create a uncluttered view. When we move onto next level state boundaries are visible because the ranking of the features are updated and rendered as seen in Figure 6d, which otherwise would have shown all the features like the earlier level without any difference (See Figure 6c). In addition, it can be seen that the number of features selected at second level is 34 and is lesser than the total feature count. Similar output has been obtained for dataset#2 and is shown in Figure 7. Figure 7a 7c and 7e shows Normal rendering without spatial filter and Figure 7b, 7d and 7f shows rendering of the data by the proposed Vector

pyramids algorithm.

When we zoom in further, number of features in the subset increases and at some point it will be equal to features in the original dataset. But since spatial filter is running during zoom/pan operation, it eliminates excess features which does not fall within the display extent. This is because ranking of features are rearranged within the dataset when zoom value changes. For every change in zoom value display extent becomes smaller. This is done by re-calculating display extent based on zoom value and view port size. There are cases where total number of levels generated in pre-processing is very less compared to the zoom level. For instance, during testing with dataset #1 we had 15 zoom levels and 3 pyramid blocks(sub-dataset) in the pyramid. One method that can be used is to map every sub-dataset with multiple zoom levels. But that can lead to reading only from the last pyramid block. For instance, if a user is focused on a specific district in the whole country and reaches bottom-most pyramid block(N), spatial filter will read only those districts and its immediate neighbors by setting filter extent from the display extent. It has also been observed that when dataset changes as we move on to a higher level there is a shift in number of features which then keep on decreasing within the sub-dataset. Figure 8a, 8c and 8e shows normal rendering output with spatial filter and Figure 8b, 8d and 8f shows Vector pyramids output at the respective zoom levels. These figures have been further magnified and a part of them is shown in the figures to highlight the difference in feature detail that is rendered at these respective zoom levels.

Further analysis of the Vector pyramids was carried out to evaluate its performance runtime and the reduction in feature count at different zoom levels, on both these datasets. Attribute Ranking generates five blocks in the pyramid for the dataset#2 and three blocks of pyramid for dataset#1. For the purpose of this evaluation, Normal vector data rendering has been carried out using the QGIS desktop GIS application (Quantum GIS Development Team 2009). For dataset#2 normal rendering takes 13590ms at startup and remains constant during every rendering operation. When using Vector pyramids it takes 16530ms at first zoom level and second level takes up only 6250ms. As we move to higher zoom level, the time required to render features starts decreasing exponentially. For dataset#1 normal rendering takes 318.435ms at startup and remains constant, while Vector pyramids takes 77.917ms at first zoom level, 65.548ms at second and as we move to higher zoom level say 7, the time required to render layer becomes 52.148ms and goes on decreasing as we move further. The increase in time for dataset #2 at the first zoom level can be due to very large sized polygons being rendered, while for dataset #1

that is not observed. Figure 9a & 9b shows runtime comparison with Normal rendering, Normal with Spatial filter and Vector pyramids for dataset#1 and dataset#2 respectively. The comparison of number of features accessed in Normal rendering, Normal with Spatial Filter and Vector pyramids for both datasets and is shown in Figure 10a & 10b.



(a) count = 599(Level 1)

(b) count = 1(Level 1)

(c) count = 599(Level 2)

(d) count = 34(Level 2)

Figure 6: Normal Rendering vs Attribute Ranking Output(Dataset #1). Red outline box shows the successive levels of zoom applied on the data.

5 Discussion & Conclusion

Current implementation of Vector pyramids has been as a plugin for QGIS desktop application and LSIViewer, an online vector data rendering application (LSI 2013). Vector pyramids have shown that the rendering speeds can improve by an order of 2 to 14 times in comparison with other existing approaches. Though the performance reported is against the rendering in native QGIS desktop application, further work is needed to evaluate the order of performance gain in web services implementations like WMS/WFS. The later was not reported as this requires modification of the code to measure time performances at each user interaction. Vector pyramids designed and developed here can be added to any desktop or web GIS packages including WMS and WFS servers to improve their rendering performance. In case of very large datasets the preprocessing step of Vector pyramids might be computationally demanding, but given the quality of the rendered data and its relevance to the user, it can be a very useful tool in many applications like multi-level data visualization and spatial data analysis platforms. In addition, along with improved data visualization, the hierarchical model of managing the attribute values may provide important clues to user-semantic based geodata management approaches and help explore improved experience to the users.

Attribute Ranking is a pre-processing technique which can be configured to generate pyramids on-the-fly or as separate datasets depending on whether the data handling will occur on the client-side or the server-side. Even though in this implementation, ESRI Shapefiles are used for evaluation and testing of the algorithm, the coupling with GDAL/OGR enables Vector pyramids to read-write any OGR supported format. Increasingly, in the WebGIS environments with data-centric approaches the geo-formats such as GML, KML and GeoJSON are commonly used to provide a better control on the data exchange. As Vector pyramids is also a data-centric design, it is anticipated that its integration into these formats can be achieved by suitably by adding additional elements to the specification. For example, while generating GML datasets *gml:pyramid* can be introduced to represent a Vector pyramid level as an inherent element of *gml:FeatureCollection*. This element can include a reference to a file on disk or the complete feature information. Though it is out of scope for this paper, future work will include developing and testing such additional elements. Another area of extension for Vector pyramids could be to provide it within the data-editing framework over the web, by incorporating it within WFS-T, to enable a richer and user controlled editing environment.

References

Anderson, C. H., Bergen, J. R., Burt, P. J. & Ogden, J. M. (1984), *Pyramid Methods in Image Processing*, Vol. 29, RCA Research and Engineering.

Bertolotto, M. & Egenhofer, M. J. (1999), Progressive vector transmission, *in* '7th ACM Symposium on Advances in Geograaphic Information System', Kansas City, USA.

DCW (2013), 'China DCW GIS Data'.
URL: *http://www.fas.harvard.edu/ chgis/data/dcw/*

ESRI (1998), *ESRI Shapefile Technical Description*.

ESRI (1999), *Extendable Image Formats for ArcView GIS 3.1 and 3.2*.

GADM.org (2013), 'GADM database of Global Administrative Areas version 2.0'.
URL: *http://www.gadm.org/*

GDAL.org (2013), 'GDAL - Geospatial Data Abstraction Library'.
URL: *http:/gdal.org*

GeoServer (2013), 'OpenGeo: GeoServer in Production'.
URL: *http://opengeo.org/publications/geoserver-production/*

Google (2013), 'Encoded Polyline Algorithm'.
URL: *https://developers.google.com/maps/documentation/utilities/polylinealgorithm*

Haklay, M. M. & Weber, P. (2008), 'OpenStreetMap: User-Generated Street Maps', *IEEE Pervasive Computing* **7**(4), 12–18.
URL: *http://dx.doi.org/10.1109/mprv.2008.80*

Longley, P. A., Goodchild, M. F., Maguire, D. J. & Rhind, D. W. (2005), *Geographic Information Systems and Science*, John Wiley & Sons.

LSI (2013), 'LSI Viewer v2'.
URL: *http://lsi.iiit.ac.in/lsi/lsiviewer/*

McMaster, R. B. & Shea, K. S. (1992), *53 - Generalization in Digital Cartography*, Assoc. of American Geographers, Washington, D.C.
URL: *http://geoinformatics.ntua.gr/courses/admcarto/lecture_notes/generalisation/bibliography/mcmaster_s*

OGC (2001), *Web Map Service Implementation Specification*.

OGC (2002), *OpenGIS Geography Markup Language (GML) Encoding Standard*.

OGC (2005), *Web Feature Service Implementation Specification*.

OGC (2010), *OpenGIS Web Map Tile Service Implementation Standard*.

Quantum GIS Development Team (2009), *Quantum GIS Geographic Information System*, Open Source Geospatial Foundation.
URL: *http://qgis.osgeo.org*

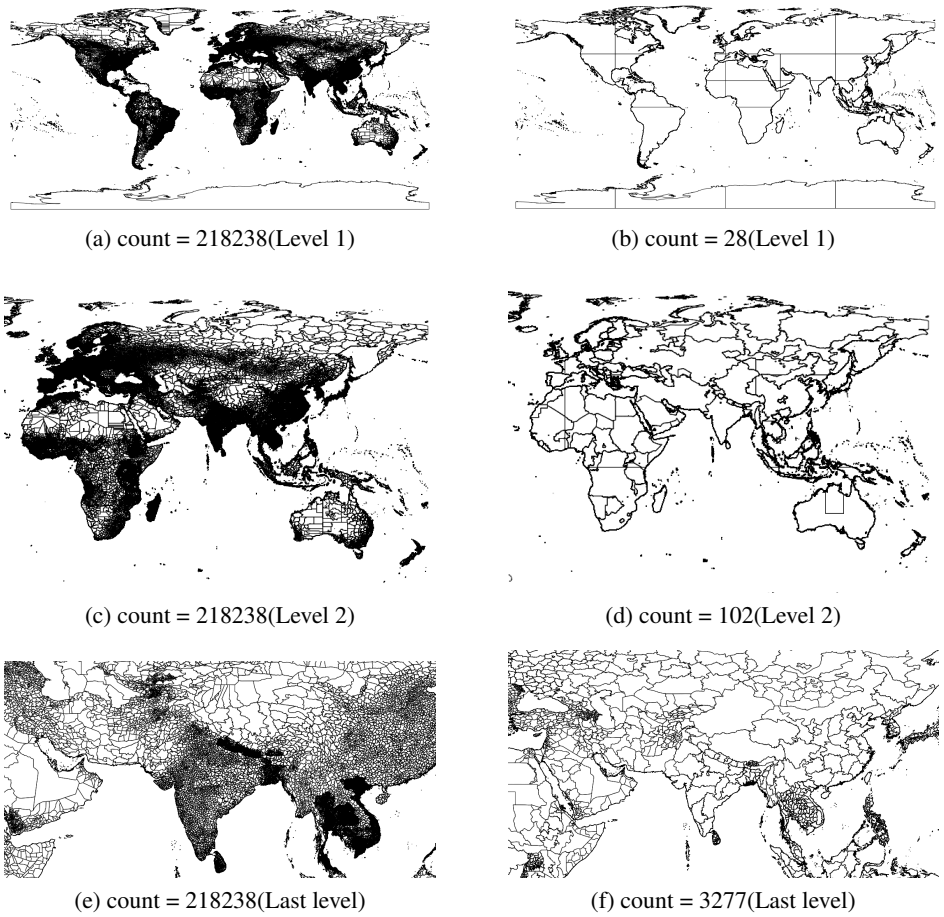
Rashad, M. & Rajan, K. (2012), Lsiviewer: An online viewer for spatial vector data, *in* 'Proceedings of FOSS4G 2012 - First National Conference on Open Source Geospatial Resources to Spearhead Development and Growth', Hyderabad, India.

Shekhar, S., Vatsavai, R., Burk, T. & Lime, S. (2006), *Geographic Information Science*, Vol. 4197 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg.

W3C (2010), 'HTML5 Reference - The Syntax, Vocabulary and APIs of HTML5'.
URL: *http://dev.w3.org/html5/html-author/*

W3C (2012), 'HTML Canvas 2D Context, Level 2'.
URL: *http://www.w3.org/TR/2dcontext2/*

Worboys, M. & Duckham, M. (2004), *GIS: A Computing Perspective, 2nd Edition*, CRC.



(a) count = 218238(Level 1)

(b) count = 28(Level 1)

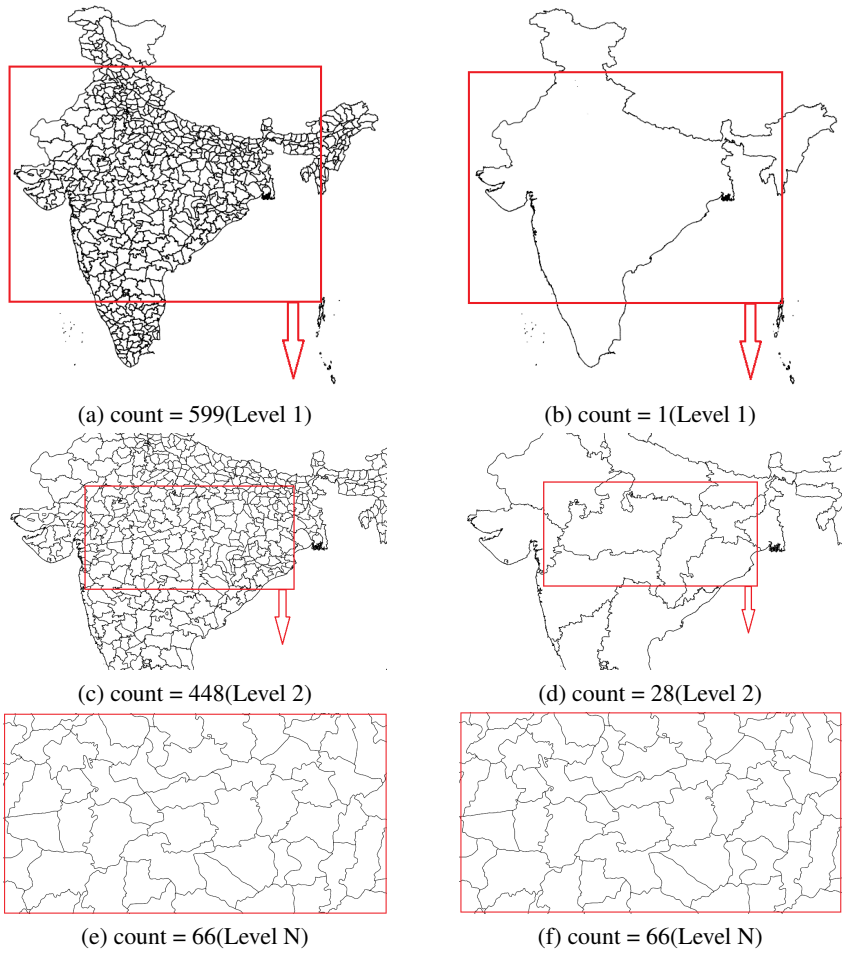
(c) count = 218238(Level 2)

(d) count = 102(Level 2)

(e) count = 218238(Last level)

(f) count = 3277(Last level)

Figure 7: A magnified view of the dataset#2 rendered using Normal Rendering(without spatial filter) and Attribute Ranking is shown for better appreciation of features.



(a) count = 599(Level 1)

(b) count = 1(Level 1)

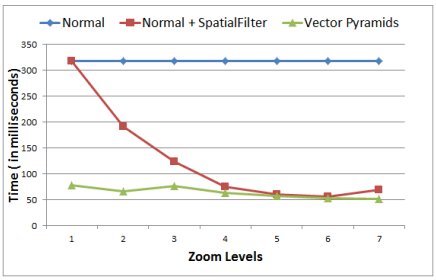
(c) count = 448(Level 2)

(d) count = 28(Level 2)

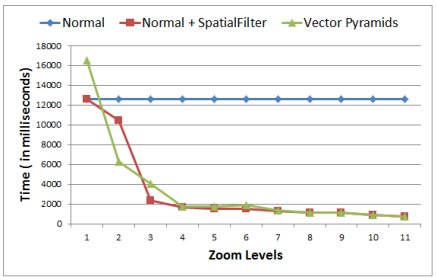
(e) count = 66(Level N)

(f) count = 66(Level N)

Figure 8: Normal Rendering vs Vector Pyramids including spatial filter (Dataset #1)

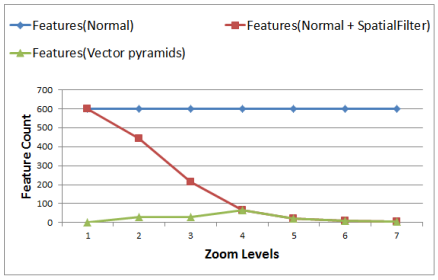


(a) Dataset#1(milliseconds)

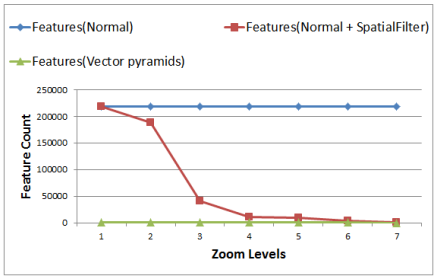


(b) Dataset#2(milliseconds)

Figure 9: Runtime for (a) Dataset#1 and (b) Dataset#2 in milliseconds



(a) Dataset#1



(b) Dataset#2

Figure 10: Feature count comparison for (a) Dataset#1 and (b) Dataset#2