Image-Based Highly Interactive Web Mapping for Geo-Referenced Data Publishing (2002)

by Haixia Zhao, Ben Shneiderman

TR 2005-35

Image-based highly interactive Web mapping for geo-referenced data publishing[1]

Haixia Zhao, Ben Shneiderman
Human Computer Interaction Lab &
Computer Science Department
University of Maryland, College Park, MD 20742
{haixia, ben}@cs.umd.edu
December 8, 2002

Abstract

This paper describes an image-based technique that enables highly interactive Web choropleth maps for geo-referenced data publishing and visual exploration. Geographic knowledge is encoded into raster images and delivered to the client, instead of in vector formats. Differing from traditional raster-image-based approaches that are static and allow very little user interaction, it allows varieties of sub-second fine-grained interface controls such as dynamic query, dynamic classification, geographic object data identification, user setting adjusting, as well as turning on/off layers, panning and zooming, with no or minimum server support. Compared to Web GIS approaches that are based on vector geographic data, this technique has the features of short initial download time, near-constant performance scalability for larger numbers of geographic objects, and download-map-segment-only-when-necessary which potentially reduces the overall data transfer over the network. As a result, it accommodates general public users with slow modem network connections and low-end machines, as well as users with fast T-1 connections and fast machines. The client-side (browser) is implemented as light-weight Java applets. YMap, an easy-to-use, user-task-oriented highly interactive mapping tool prototype for visual geo-referenced data exploration is implemented using this technique.

Keywords: Web GIS, choropleth map, information visualization, dynamic query, universal usability
Web supplement: http://www.cs.umd.edu/hcil/census/YMap122002/smap.html

1. Introduction

The Internet has become an important media for geo-referenced data publishing for public access, because of the well-documented benefits in terms of distributed access, centralized control for updates, and modest development cost. Existing mainstream online mapping techniques can be classified into two categories: raster-imaged based and vector based. In a typical raster-imaged based approach, servers generate maps as pictures in one of the standard raster graphic formats supported by graphical Web browsers. Interaction is accomplished by submitting a request to the server for a new map image. Even simple user actions such as turning on or off display attributes often require such a "round-trip" and complete screen refreshes. Examples include multiple Web mapping sites powered by ESRI's ArcView IMS, MapObjects IMS, or ArcIMS [ESRI 2002], MapInfo's MapeXtreme and MapXsite [MapInfo 2002].

This typical architecture places severe restrictions on the map interactivity and interface design flexibility, and poses additional limitations such as slow map update, increased network load, and often the time poor scalability in terms of number of simultaneous users (Andrienko and Andrienko 1999, Zaslavsky 2000). On the other hand, varieties of exploratory visual interaction methods such as dynamic query and linked brushing have been proposed long time ago, employed in some desktop mapping tools (Monmonier 1989, Symanzik 1996, MacEachren and Kraak 1997, Dang et al 2001), and proved to be a powerful facilitation for users to explore large data sets, discover trends and extract knowledge from the data.

[1] This article includes a word which is or is asserted to be a proprietary term or trade mark. Its inclusion does not imply it has acquired for legal purposes a non-proprietary or general significance, nor is any other judgement implied concerning its legal status.

However, these sub-second-response, fine-grained interface controls characterizing desktop environments are typically missing in the Web environment. Researchers have been exploring the possibility of highly interactive Web mapping by switching to client-side solutions, and some excellent examples have already emerged. Client-side solutions typically ship vector geographic data (in the format of such as ESRI shapefile, Vector Markup Language (VML), Scalable Vector Graphics (SVG) or Geography Markup Language (GML)) to the client computer, where the data is interpreted and rendered typically 1) by software implemented in Java, such as Descartes (Andrienko and Andrienko 1999) (which later became part of CommonGIS (CommonGIS Consortium, 2002)), Interactive Map Applet (Sorokine et al 1998), and CIESIN's Demographic Viewer (CIESIN 2002), 2) or with the help of various specialized browser plug-ins (or special browsers), such as AXIOMAP (Zaslavsky 2000), CDV (Dykes 1997), GeoMedia Web Map (Intergraph Corp. 2002), Autodesk MapGuide (Autodesk 2000), and Flash Geovisualization prototype (Steiner et al 2002). Mapping software is either pre-installed on the client computer, or is downloaded each time along with map and data information. However, these approaches typically face some of the following problems:

1. Large size of geographic data files to be transmitted over the network. This causes the initial download time to be very long, especially over modem network connections. The problem becomes more severe when the number of geographic features (regions, lines, etc.) increases, e.g., a map of 3140 USA counties is about 1.52MB (as ESRI Shapefile) and takes more than 3 minutes to download over a 56K modem connection to a Pentium-III 1.0G CPU, 256Mb RAM notebook (including the time to initially render the map). The long initial download and rendering time has a strong negative effect on users, often causing occasional users to give up the attempt. Although more compact vector standards could be defined especially for Web GIS to replace the widely used high-quality standards for desktop mapping, by sacrificing the map quality by using less precision of coordinates, the compression is limited, considering the distortion problem. Also, it is costly and hard to convert the huge amount of geographic data that is already in use, owned by government agencies, communities, organizations and the mapping industry.
2. Large download size for software. While rich GIS functionality can be delivered to the client, the mapping software to be downloaded is usually large (1MB on average, according to (Kendall 1999)).
3. Unsatisfactory interaction performance scalability to the number of geographic features, or unable to efficiently render complex maps. E.g., in dynamic query, users alter query criteria by adjusting sliders and immediately observe corresponding search result changes graphically. Such dynamic interaction requires the response time to be no more than 0.1 second in order to ensure a smooth change feeling (Shneiderman 1998). However, CommonGIS shows significant map update delay for a USA county view map during dynamic query (with dynamic map update on). AXIOMAP does not scale up beyond about 200 or so regions using VML. It can render more using SVG, but still far from enough for complicated maps such as a USA county view map. Performance problems are also reported in (Steiner et al 2002).
4. Possible incompatibility with a client computer. E.g., AXIOMAP requires Microsoft Internet Explorer 5.0 or SVG plug-in. CDV requires Tcl/Tk plug-in. In terms of client compatibility and platform independence, Java applet, which can be executed by almost every Web browsers, are most suitable (Kahkonen et al 1999, Brinkhoff 2000).

The above limitations have important social implications. First, it inhibits data sharing with the general public, contradicting the goal of universal usability, as stated in (Hochheiser and Shneiderman 2001). Imagine that recently retired users dial up from home, trying to figure out the cost of living and retired population distribution from the statistical data published on the Census Bureau Web site, to help them make a decision of where to move to. The scenario addresses some of the issues that will not be solved any time soon for a large portion of public users or users from developing countries. 1) Slow network connection. Even in the USA, recent data shows that most users still access the Internet via modem connections (56K and less) (based on the data by Nielsen//NetRatings), and analysts do not expect the majority of homes to have broadband access (fast access) for at least five years (Romero 2002). 2) Home users may not have powerful PCs. 3) Occasional users usually do not have the required plug-ins or special software (if any) installed, and may not want to invest the time or do not have the knowledge to install them. Secondly, before common consensus is reached on an acceptable-quality, highly compact map data standard for Web GIS and conversion tools are developed and distributed, it still remains infeasible for most map-referenced data publishers to provide highly interactive mapping service on the Web and make it accessible to general public users. Even when final consensus is reached, the cost for the organizations and companies to transform existing map data into the new standard is high, enforcing a de-facto "digital mapping divide" (only big organizations who can afford powerful servers and expensive mapping packages are able to publish their data) as noted in (Zaslavsky 2000).

The paper presents our image-based technique that enables highly interactive Web choropleth maps with sub-second fine-grained interface controls such as dynamic query, dynamic classification, object data identifying, adjusting user setting, as well as turning on/off layers, panning and zooming, with no or minimum server support. We target to make the service widely accessible and usable to general public users with modem-speed connections and moderately fast home machine, as well as users with T-1 speed connection and fast machines. Additionally, we target to make it affordable for organizations to provide the services. The client-side (browser) is implemented as light-weight Java applets, requiring no special plug-ins thus minimizing users' effort. Short initial download and rendering time and minimum need for server support during interactions make real-time (sub-second) interaction possible even for public users with slow network connections. The technique also has the benefit of near constant performance scalability to the number of geographic objects. And it can use geographic data in many formats (with the assistance of any mapping tools, freeware or commercial, that can interpret the data and export maps as images. After the basic map images are generated, the system can run without any external mapping tool support.) Using the image-based technique, we also developed YMap, a prototype of an easy-to-use, task-oriented, highly interactive Web choropleth map exploration tool.

The paper is organized as follows: Section 2 introduces the key idea of the image-based technique by describing the raster image base maps and how geographic knowledge is encoded/decoded into/from the raster images. Section 3 gives the overall system architecture. Section 4 describes how varieties of interactions can be fulfilled under the technique, with our YMap prototype as an illustration. Section 5 compares and discusses the performance of YMap and vector-based Web GIS approaches. And Section 6 concludes.

2. *Raster image base maps and geographic object encoding/decoding*

The core idea of the image-based technique is to use colour-coded raster images to store geographic object knowledge. The colour-coded images are called *base maps*. Geographic object knowledge is delivered to the client in the form of base maps, which are much smaller than vector files (refer to section 5 for details). The mapping rules to encode/decode geographic object knowledge into/from base maps are as following:

Encoding at the server-side:

$$\begin{aligned}
&\mathrm{Red(ID)} = \mathrm{Min}\ \{\ \{\mathrm{floor}[\mathrm{ID} / (\mathrm{ng} * \mathrm{nb})] \bmod \mathrm{nr}\} * (2^8 / (\mathrm{nr}-1))\ ,\quad 2^8 - 1\} \\
&\mathrm{Green(ID)} = \mathrm{Min}\ \{\ \{\mathrm{floor}[\mathrm{ID} / \mathrm{nb}] \bmod \mathrm{ng}\} * (2^8 / (\mathrm{ng}-1)),\quad 2^8 - 1\} \\
&\mathrm{Blue(ID)} = \mathrm{Min}\ \{\ (\mathrm{ID} \bmod \mathrm{nb}) * (2^8 / (\mathrm{nb}-1))\ ,\quad 2^8 - 1\}
\end{aligned}$$

Decoding at the client-side:

$$\begin{aligned}
\mathrm{ID} = {} & \mathrm{floor}[(\mathrm{Red} + (2^8 / (\mathrm{nr}-1) / 2)) / (2^8 / (\mathrm{nr}-1))] * \mathrm{ng} * \mathrm{nb} \\
& + \mathrm{floor}[(\mathrm{Green} + (2^8 / (\mathrm{ng}-1) / 2)) / (2^8 / (\mathrm{ng}-1))] * \mathrm{nb} \\
& + \mathrm{floor}[(\mathrm{Blue} + (2^8 / (\mathrm{nb}-1) / 2)) / (2^8 / (\mathrm{nb}-1))]
\end{aligned}$$

- ID: the unique Id assigned by the system to a geographic object (region, river, etc.) in a map layer.
- nr: the number of different red colours used.
- ng: the number of different green colours used
- nb: the number of different blue colours used
- Red: the red value of the colour used to colour-code the geographic object, based on the 24-bit colour scale.
- Green: the green value of the colour used to colour-code the geographic object, based on the 24-bit colour scale
- Blue: the blue value of the colour used to colour-code the geographic object, based on the 24-bit colour scale

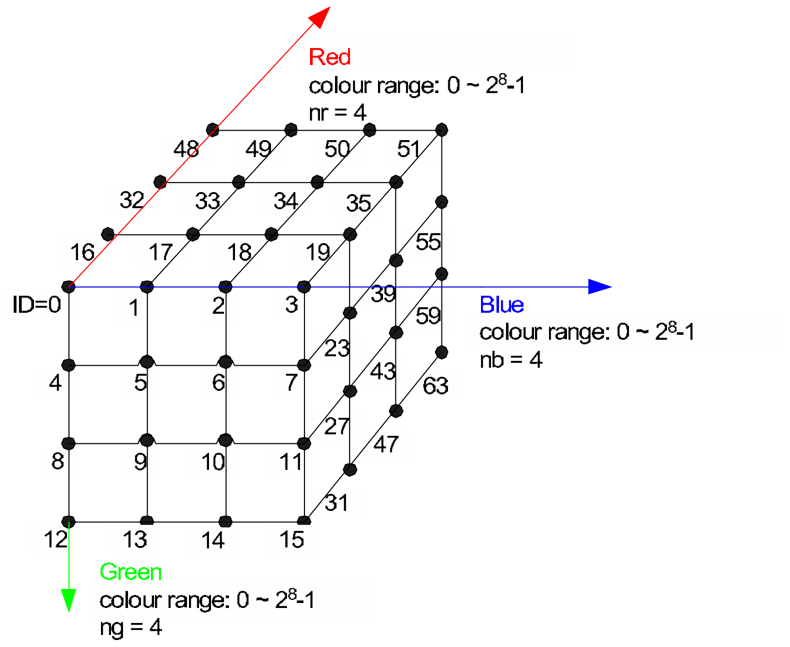
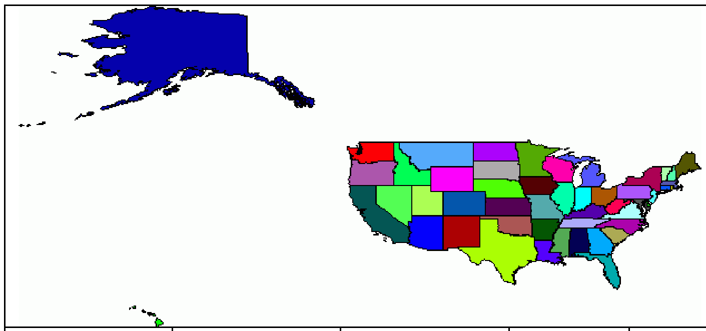


Figure 1 : The colour-coding space

Figure 1 shows the colour-coding space and illustrates the relationship between the geographic object IDs and the colours used to encode them. (nr = ng = nb = 4). The grid vertices represent the colours that are used to encode geographic object IDs.

GIF format is chosen as the base map format in our prototype because of its high compression rate and excellent image quality for images with many pixels but not many colours. A GIF image can have maximum 256 colours. Theoretically a GIF image can store the knowledge of 256 geographic objects. In our prototype implementation, only 64 (4*4*4) colours are used, and a colour is rounded to the nearest grid vertex colour during decoding, considering the quantization errors introduced by the GIF image compression tool we used. Figure 2 shows a sample base map consisting of 51 USA states and a corresponding geo-referencing data file. Colour black (0x000000) and white (0xFFFFFF) are used for the region border and background correspondingly. The geographic objects here are not restricted to regions, but can also be of other shapes, such as roads and rivers.



Attribute1	**Attribute2**	**State name**	**Assigned ID**	Colour for encoding (not in the data file)
			Border	0x000000
4319154	404038	Alabama	1	0x000055
609311	55004	Alaska	2	0x0000AA
4554966	366533	Arizona	3	0x0000FF
2522819	235062	Arkansas	4	0x005500
32268301	2978585	California	5	0x0055AA
3892644	329447	Colorado	6	0x0055FF
	……			……
			Background	0xFFFFFF

Figure 2: A sample base map (6.3KB, 612 x 288 pixels) and a corresponding geo-referencing data file

When the number of geographic objects exceeds 256 (in our prototype 64), a hierarchical colour-coding method is used. Figure 3 illustrates a two-level colour-coding where 2 colours distinguish 4 states. So theoretically two GIF images can store the knowledge of 256 * 256 = 65536 geographic objects. (Our prototype has 64 * 64 = 4096 colours, which is far more than enough even for a USA county view that has 3140 counties.) Figure 4 shows the USA county view base maps used in our prototype.

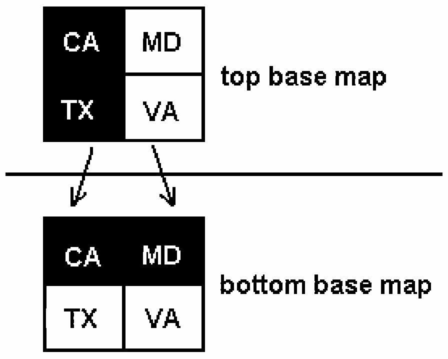


Figure 3: Hierarchical base map colour-coding

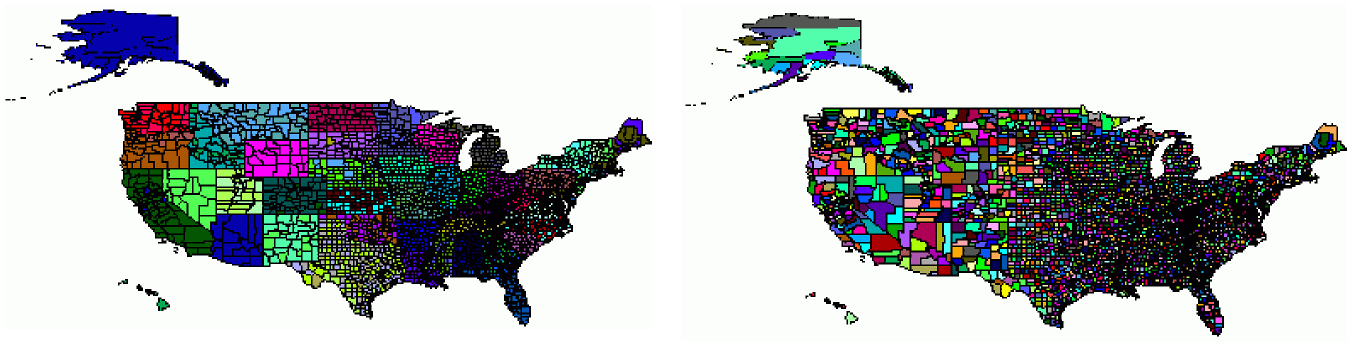


Figure 4: Two-level base maps for USA 3140 county view (top 10KB, bottom 11KB, both 612 x 288 pixels)

3. *Architecture*

Figure 5 shows the architecture of the technique. Upon the client's request of new geographic objects (such as regions, rivers, etc.) and geo-referencing data, base maps that carry the knowledge of geographic shapes are delivered to the client applet along with the geo-referencing data to the geographic objects. Interactions such as dynamic query and dynamic classification are handled solely by the Java applet by decoding geographic knowledge out of the base maps and updating client-side map display in sub-second, without communicating with the server.

The round-trip of requesting and delivering happens only when a new set of attribute data or geographic shapes are needed (explained in more detail in the next section). Additionally, base maps are very small, so are the geo-referencing data files (depending on the number of attributes requested and number of

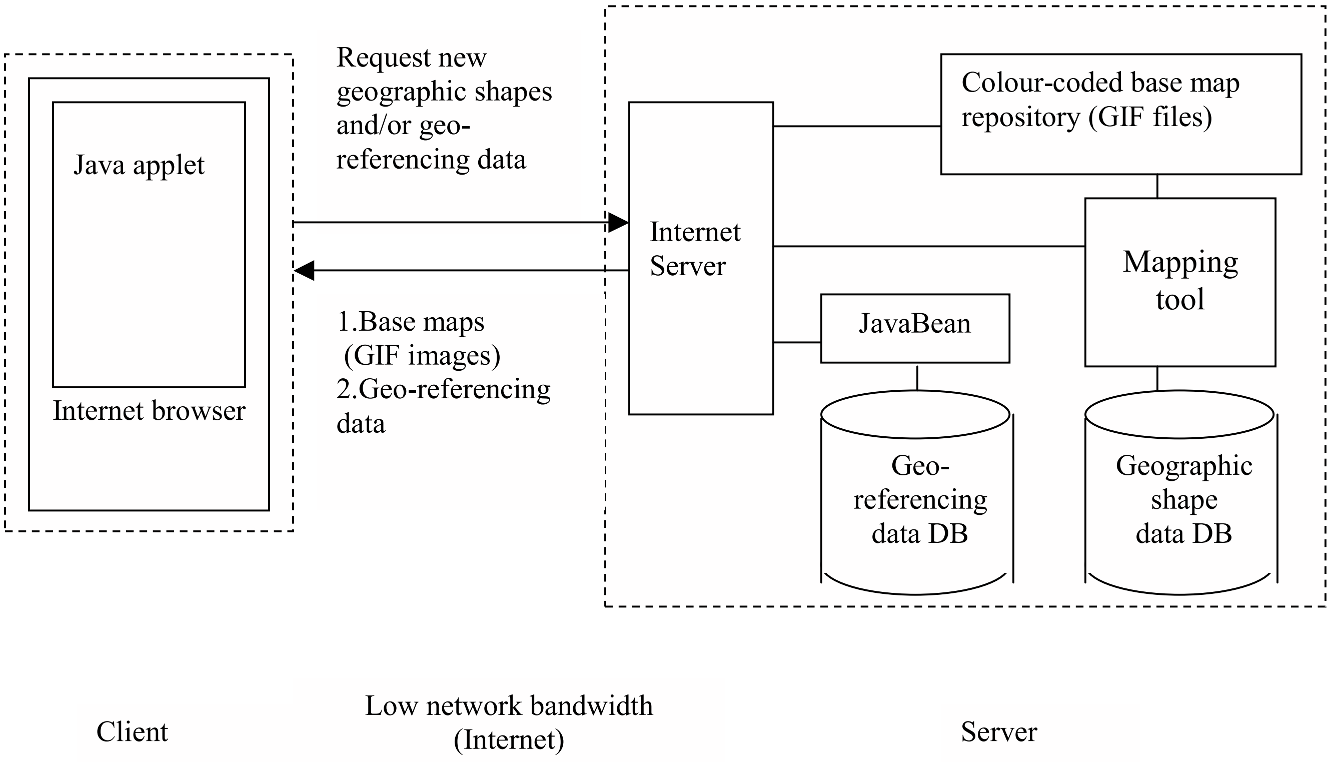
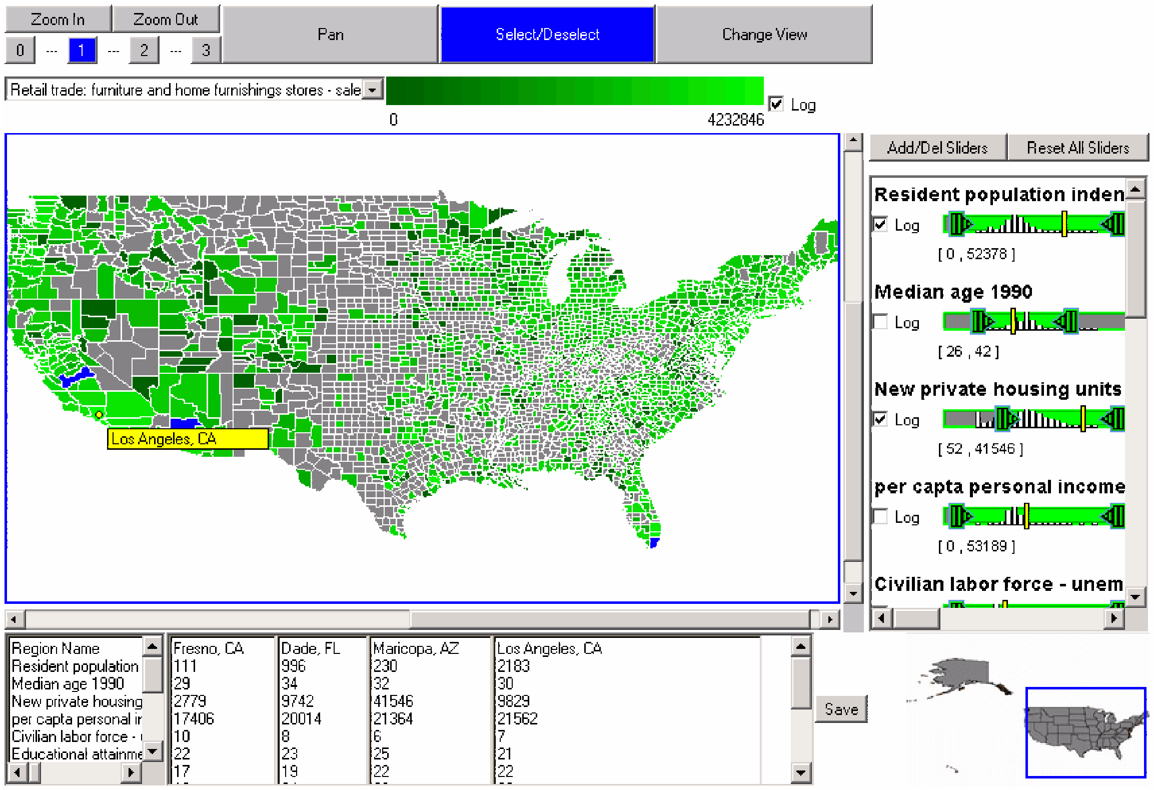


Figure 5: The architecture

regions, e.g., ~100KB for 15 attributes for 3140 USA counties.). As a result, the network transfer load is relatively low. GIF base maps can be provided in two different ways, depending on how flexible the zooming needs to be. If zooming levels are pre-defined for the system, as is often the case with online mapping services, a hierarchical set of GIF base maps can be pre-generated for those zooming levels, residing on the server as base map file repository for later use. Any mapping tools, commercial or free shareware, can be used to generate those base maps, as long as they can read and render the geographic shape data of some common standards, and export the map images into files. The mapping tool is then not needed when the service is running. If arbitrary zooming scales are desired, the mapping tool can generate base maps on-the-fly. (However, dynamically generating response will increase the response delay and the server burden, as is always true for any Web service systems.)

4. *Highly interactive online map manipulation*

This section explains in detail how varieties of interactions can be fulfilled with the image-based technique, with our YMap prototype as the illustration but not limited to functionality available in the prototype. Figure 6 shows a screen snapshot of YMap applet. It mainly contains a choropleth map canvas (center), a set of dynamic query double-box sliders (right), a detail data panel (bottom), a drop-down-list-box containing the attributes to choose from for shading the map (above the map, with continuous shading legend), an overview map (right-bottom) and a toolbox for panning, zooming and toggling between a state view and a county view (top). The prototype shows an important general interface design methodology for exploring large data sets [Shneiderman 1998]: overview of the data distribution via visualization (on the shaded choropleth map and histogram bars on the sliders), dynamic query (by adjusting double-box sliders) to filter out unwanted entries to narrow down to the result set of interests, and details-on-demand to further examine individual entries of interests (mouse over a region to view the data associated with that region, or select regions to compare their data in the detail information panel). The snapshot shows a county view of continental USA shaded by the year 1992 furniture and home furnishing store sale volume. Counties with low number of new private housing units or with median population age out of range [26, 42] are filtered out (in gray)



Region Name	Fresno, CA	Dade, FL	Maricopa, AZ	Los Angeles, CA
Resident population	111	996	230	2183
Median age 1990	29	34	32	30
New private housing	2779	9742	41546	9829
per capta personal ir	17406	20014	21364	21562
Civilian labor force - (	10	8	6	7
Educational attainme	22	23	25	21
	17	19	22	22

Figure 6 A screen snapshot of our YMap prototype

The following features are considered to be important for an interactive choropleth map tool (not limited to the features that are already implemented in our prototype.) They can be implemented with the image-based technique as following:

- **Dynamic query by scrolling sliders**. Dynamic query is a very powerful exploratory interaction method for both open-ended questions (such as "where is a nice place to live?") and specific tasks (such as "which state has the lowest population density?") Dynamic query requires real-time acquisition and update of the result on map (usually less than 0.1 seconds, otherwise users will notice the delay.) Dynamic query will be impossible if any server support is involved, because of the unpredictable network delay, regardless of the network bandwidth. In our image-based technique, geographic knowledge is encoded into the GIF base maps. The Java applet simply scans the pixels in the base maps, uses the encoding-decoding rule to obtain geographic object IDs from the pixel colours, uses the IDs to query the geo-referencing attribute data set, decides if the geographic objects should be filtered out, and changes the display accordingly. The geographic objects noted here is not limited to regions, but also include other common shapes, such as lines (rivers, roads) and dots (cities)
- **Dynamic classification**. The map in the screen snapshot is continuously shaded. Sometime classified (categorized) shading is preferred. Dynamic classification is useful in exploring optimal statistic class breaks. Our image-based technique allows real-time dynamic classification on map, using a method similar to the one described above for dynamic query.
- **Scatterplot and linked brushing with the choropleth map**. With the attribute data file on the client, scatterplots can be easily implemented using the Java graphic package. The mapping between map colour and data set record index enables real-time linked brushing since the client can get geographic object IDs from base map colours and vice versa, without server support.
- **Multiple thematic layers, background layers and labels:** Sometimes users want to view a thematic layer on a background layer (e.g., a highly interactive river layer with the state borders as the background) or view multiple thematic layers overlapped together. All the thematic or background layers that do not need to be dynamically manipulable can be generated on the server and sent to the client as raster images with transparent background, and rendered on top of the dynamically updated choropleth map layers. Labels can either be part of the base maps or overlap as a separate layer.
- **Zoom:** Zooming is important for observing data patterns in smaller or denser regions. Zooming could be done either by simply enlarging/shrinking the current available base map or having the server deliver a new base map of the desired zoom factor. The first choice does not need server support but may have aliasing problem during interpolation/aggregation, and does not generate as good quality map images as the second choice. It can be used alone when the zooming scale does not jump by more than a factor of 60-80%. The second choice ensures good map quality. When map tools are used on the server to generate base maps on-the-fly (for arbitrary zooming), the client simply notifies the server about the geographic range of the desired base map. The base map file size does not vary too much since the client map view port has fixed size. If a repository of pre-generated base maps is used, the pre-generated base maps must be designed carefully, because a complete base map of a close zoom-in may be very large.

In order to minimize network transfer volume and response time, a large complete base map is broken down to several much smaller base map segments (0.5 ~ 33KB each in our prototype) that could restore the complete map when put together. Only those segments needed for the current client view will be delivered to the client. Figure 7 illustrates a case where a base map is broken down to 4 x 4 = 16 equal-sized segments, and only two segments (row2-col3 and row3-col3) need to be delivered to the client for the current view. If the size of segments is the same as the client map view port size (for both width and height), then the number of segments that needs to be delivered to the client varies from 1 to 4 when zooming-in, depending on which part the user is viewing. Since the client view port size is fixed (and relatively small, no bigger than the screen resolution), the size of GIF base maps transferred on the network is upper bounded by a small value. This deliver-only-when-needed mechanism assures fast system response upon user zoom-in, and potentially significantly reduces the total size of base maps to be transferred on the network, since most of the map segments may not need to be delivered at all, because in a lot of cases users are only interested in viewing a small range of the map after the overview.

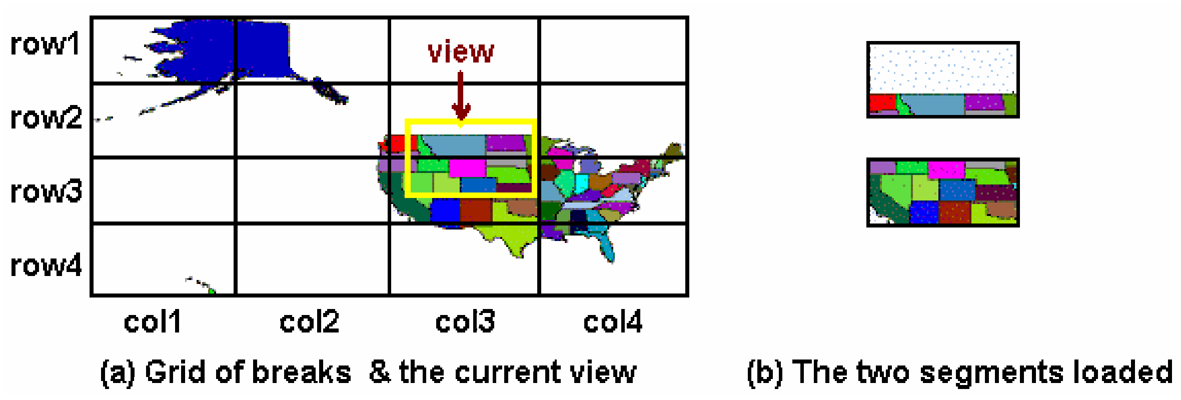

row1
row2
row3
row4
col1
col2
col3
col4
view
(a) Grid of breaks & the current view
(b) The two segments loaded

Figure 7 : A big base map is broken down to several smaller segments. Only segments overlap with current view will be fetched to the client from the server.

- **Pan:** On panning, the view is refreshed when users mouse-up. The system action is similar to that for zooming. During panning, only the part of map that changes visibility is updated in shading and filtering status.
- **Obtain information of a geographic object or multiple geographic objects**. When the users mouse-over a geographic object, its attribute data can be displayed in the detail information panel in sub-second. At the same time, yellow bars appear on the sliders, indicating the positions of the object's attribute values in the distributions. Multiple objects can be selected to view and compare their attribute data.
- **Change state to county level and vice versa:** In the prototype, the user can quickly switch between state view and county view.

5. *Performance comparison and benefit analysis*

Table 1 and table 2 show some performance data about our Ymap, including package and data size, the system response time under different network connection speeds and for maps with different complexities. We chose CommonGIS [CommonGIS2002] to compare to YMap, because CommonGIS is one of the fastest Web GIS packages that uses vector geographic data, and has comparable functions.

	CommonGIS		**YMap**	
Package size	~ 1 MB		~ 60 KB	
Package download time	**56K modem**	**T-1**	**56K modem**	**T-1**
	~ 180 s	~ 13 s	~ 12 s	~ 2 s

Table 1: Compare the package size and package download time of vector-based CommonGIS and image-based YMap under different network connection speeds. The data was obtained using Microsoft Internet Explorer 5.0 in Windows 2000 Professional, on a Pentium-III 1.0 GHz CPU, 252 MB RAM Fujitsu notebook. The modem connection is running at 31Kbps.

	51 USA States		**3140 USA Counties**	
	CommonGIS	**YMap**	**CommonGIS**	**YMap**

Initial downloaded geographic data size	~ 223 KB		6.4 KB		~ 1.52 MB		~ 20.8 KB	
	56K modem	**T-1**	**56K modem**	**T-1**	**56K modem**	**T-1**	**56K modem**	**T-1**
Initial data download time (geographic data + a 5-attriute data file)	~ 35 s	~ 5 s	< 5 s	< 2 s	~ 190 s	~ 14 s	~ 20 s	< 2 s
Total initial download time (package + data)	~ 115 s	~ 18 s	< 17 s	< 4 s	~ 370 s	~ 27 s	~ 32 s	< 4 s
Dynamic Query (w/ continuous update)	Smooth				Significant delay, not smooth		Smooth	
Zoom & Pan (First-time view)	Sub-second	Sub-second	Usually < 3 s	Sub-second	Sub-second	Sub-second	2 ~ 40 s, mostly < 10 s	~ 1 s
Zoom & Pan (once visited and cached)	Sub-second	Sub-second	Sub-second	Sub-second	Sub-second	Sub-second	Sub-second	Sub-second

Table 2: Compare the initial download size, download time, and interaction performance of vector-based CommonGIS and image-based YMap under different network connection speeds. The data was obtained using Microsoft Internet Explorer 5.0 in Windows 2000 Professional, on a Pentium-III 1.0 GHz CPU, 252 MB RAM Fujitsu notebook. The modem connection is running at 31Kbps.

From the tables, we can see that both CommonGIS and YMap show smooth dynamic query on a map of a small amount of geographic regions (e.g., 51 states). But when the number of regions increases (e.g., 3140 counties), CommonGIS shows significant delay in dynamic query, while YMap still shows smooth response. Actually, YMap has near- constant performance scalability for dynamic query (also other dynamic features, such as dynamic classification) because the number of base map image pixels needed to be checked is bounded by the size of current client map view port. Under fast network connection (T-1), both CommonGIS and YMap have reasonable initial download times and fast response to zoom and pan, though CommonGIS takes at least 4 times longer than YMap to initially download the software package and data. However, in 56K modem connection, CommonGIS needs more than 6 minutes to download the package and the map data about USA counties, before anything can be rendered onto the screen for the user. On the contrast, YMap only takes about 30 seconds. Zoom and Pan are fast in both CommonGIS and YMap under T-1 connection or for relatively simple maps (e.g., 51 USA states). For a complicated map of 3140 USA counties, and under 56K modem connection, zoom and pan in YMap usually takes less than 10 seconds for the first-time zoom/pan. Once a part of the map has been viewed at a certain zoom scale, it is cached by the applet and takes less than one second for later visit. Sometime, initial zoom and pan will take longer time (e.g., 40 seconds), when 6 ~ 8 base map segments (two level encoding) need to be downloaded at once, and those segments contain dense tiny regions information. The sizes of base map segments are mostly under 5K, but it can vary from 0.5K to 33K, depending on where the user zooms-in/pan-to. E.g., a segment of size 612 x 288 about the north-east part can be 33K, a segment of the same size about the middle west part can be 9Kb, a segment about Hawaii can be only 1K, and a background segment of the same size can be only 0.5K. A variety of techniques can be used to reduce the maximum value and also the deviation of response delay during first-time zoom/pan. These techniques are not implemented in our current prototype, but will be described shortly.

Considering initial download time, zoom and pan, CommonGIS and YMap show different Internet delay patterns under slow network connections. CommonGIS takes a long time to download everything before anything becomes viewable and available for the user to explore. This could be a serious problem for occasional first-time public users who are uncertain about the result quality and not willing to spend a lot of time downloading something that may turn out to be useless. Also, precious network bandwidth could be wasted on transferring big trunk of information that the users find not relevant and discard. By contrast, YMap responds rapidly, providing users with an overview of the result, which is also something that users can use immediately to explore the geo-referenced data. Additionally, YMap uses "levels of detail" for map information, and defers the map segment transfer until necessary, which may reduce the overall data transfer volume significantly, because users may simply discard the result or only be interested in examinating a small part of the map after overview. Although sometimes zoom/pan takes a little longer, a variety of techniques can be applied to further speed it up:

- Combine the usage of a newly delivered high-quality base map with the usage of an enlarged version of the currently available base maps. Upon zooming, a base map already cached on the client can be interpolated/aggregated to produce a larger/smaller base map. This base map can immediately be rendered to respond to the user's zoom-in request. At the same time, another Java thread fetches a new high-quality base map from the server. When the new base map arrives, it is rendered to replace the interpolated/aggregated one. This way, users can continue the task without waiting and feels the response is real-time.
- Pre-fetching: When users are using other functions such as dynamic query on the current zoom-level map, the system can predict the user's next zoom/pan need and use a separate Java thread to pre-fetch the base maps from the server.

6. Conclusion

We described an image-based technique for highly interactive Web mapping for geo-referenced data publishing and visual exploration. Compared to traditional vector-based approach, this technique has significantly shorter initial download time, near-constant performance scalability, and can potentially significantly reduce the overall data transfer over precious Internet bandwidth. It can accommodate general public users with slow network connections and low-end machines. Extended use of this technique could be any highly interactive Web applications in which shape data for visualization and dynamic update is large, but cannot be easily constructed from just the attribute data set.

7. Acknowledgement

This work is partially supported by a grant from the US Census Bureau. Also, we thank Dr. Gennady L. Andriyenko, German National Research Center for Information Technology, and Dr. Ilya Zaslavsky, University of California, San Diego Supercomputer Center, for providing information and help with their systems (CommonGIS and AXIOMAP respectively). We thank Dr. Catherine Plaisant, Human-Computer Interaction Lab, University of Maryland, and Dr. Evan Golub, Computer Science Department, University of Maryland, for their suggestions and comments during the work.

8. References

Andrienko, G.L. and Andrienko, N.V., 1999, Interactive maps for visual data exploration, In *International Journal of Geographical Information Science*, vol. 13 (4), June 1999, pp.355-374.

Autodesk, Inc. 2002, Autodesk MapGuide. http://www.autodesk.com

Brinkhoff, T., 2000, The impacts of map-oriented internet applications on Internet clients, map servers and spatial database systems. In *Proceedings 9th International Symposium on Spatial Data Handling* 10-12 August 2000, Beijing, China.

CIESIN (Center for International Earth Science Information Network), 2002, Demographic data viewer, http://www.ciesin.org, http://plue.sedac.ciesin.org/plue/ddviewer/ddv20/index.html (demo), last accessed on Nov 8th, 2002.

CommonGIS Consortium, 2002, The CommonGIS project, http://www.commongis.com, 2002

Dang G., North C., Shneiderman B., 2001, Dynamic queries and brushing on choropleth maps, In *Proc. International Conference on Information Visualization 2001*, Available from IEEE Press (2001).

Dykes, J.A. 1997, Exploring spatial data representations with dynamic graphics. In *Computers & Geosciences*, **23**(4), 1997, pp.345-370

ESRI, 2002, http://www.esri.com

Hochheiser, H., and Shneiderman, B., 2001, Universal usability statements: marking the trail for all users. In *ACM Interactions*, 8(2), March-April 2001, pp.16-18.

Intergraph Corp. 2002, GeoMedia Web Map. www.intergraph.com

Kahkonen, J., Lehto, L., Kilpelainen, T., and Sarjakoski, T., 1999, Interactive visualization of geographical objects on the Internet. In *International Journal of Geographical Information Science*, vol. 13 (4), June 1999, pp.429-438.

Kendall, G., 1999, A guide to Internet mapping products and pricing, *Mapping Awareness*, Vol 12 No 7, August 1999, pp. 34-39

MacEachren, A., and Kraak, M., 1997, Exploratory cartographic visualization: advancing the agenda, *Computers and Geosciences*, 23, 335-344, 1997.

Monmonier, M., 1989, Geographic brushing: enhancing exploratory analysis of the scatterplot matrix, *Geographical Analysis*, 21(1), pp. 81-84, 1989.

Nielsen//NetRatings, 2002, http://www.nielsen-netratings.com

Romero, S., 2002, Price is limiting demand for broadband, *The New York Times*, December 5, 2002.

Shneiderman B., 1998, *Designing the User Interface,* 3rd Edition, Addison-Wesley Longman, Inc., 1998.

Sorokine, A., Merzliakova, I., 1998, Interactive map applet for illustrative purpose. *Proceedings of 6th International Symposium on Advances in Geographic Information Systems*, Nov. 6-7, 1998, Washington DC, ACM Press, 1998, pp. 46-51.

Steiner, E.B., MacEachren, A.M., and Guo, D., 2001, Developing and assessing light-weight data-driven exploratory geovisualization tools for the Web. *Proceedings of the 20th International Cartographic Conference*, Aug. 6-10, 2001, Beijing, China.

Symanzik, J., Majure, J., Cook, D., 1996, Dynamic graphics in a GIS: a biderectional link between ArcView 2.0 and Xgobi. *Computing Science and Statistics*, 27, 1996, pp. 299-303,

Zaslavsky, I., 2000, A new technology for interactive online mapping with vector markup and XML, *Proceedings of 8th Internatinal Symposium on Advances in Geographic Information Systems*, 2000.