Efficient Filtering and Clustering Mechanism for Google Maps

Gayratjon Kholmuradov Habibullayevich, Xian Chen, and Hyoseop Shin
Web Intelligence Laboratory, Konkuk University
E-mail: khgaryjon@gmail.com, {chenxian, hsshin}@konkuk.ac.kr

***Abstract*—Map Services like Google Maps provide visual positioning information of streets, geographic places and business objects through the web. However, straightforward implementation of map applications by using these APIs often fails in achieving effective presentation and user interaction against a large number of geospatial entities; too many markers on the map may cause both visual overload and sluggish interaction with the map. In order to address these issues, this paper proposes grid-based filtering and density-based clustering schemes for efficient layout and handling of over thousands of geographical entities that are returned from Map Services. The experimental results with Google Maps show that our proposed schemes are quite effective at rendering and user interaction against a large number of map entities.**

***Index Terms*—grid-based filtering, density-based clustering, geospatial data.**

I. Introduction

Along with the wide use of information technology, more and more attention is being paid on online map services. Nowadays several map services are quite popular, such as Google Maps, Bing Maps, Yahoo Maps, etc. These map services have been broadly used over worldwide on the Internet to get positioning or other information about some locations, streets, etc. These services should be required to become more and more standard, which can recommend what users expect for online map experience [1], [2] and [3]. They do not only allow user to have good interactivity with these map services, but also publicly provide their own map specification and APIs for developers to integrate web and mobile applications with above mentioned maps. However, some applications are required to display a large number of locations or markers on the map. Despite the most significant convenience of these APIs, naively plotting thousands of markers on the map might quickly lead to degraded user experience. Too many markers on the map cause both visual overload and sluggish interaction with the map as shown in Fig. 1.

In this paper, to overcome above mentioned issues, we have proposed Grid-based filtering algorithm and implemented DBSCAN density-based clustering algorithm [4] for displaying large-scale geospatial data on the map.

Manuscript received November 15, 2012; revised December 28, 2012.

The paper is organized as follows. In Section II, we give an overview of the related work. In Section III, we explain the two main algorithms. In Section IV, we apply our algorithms into restaurants dataset and show experimental results to estimate efficiency and effectiveness of algorithms.

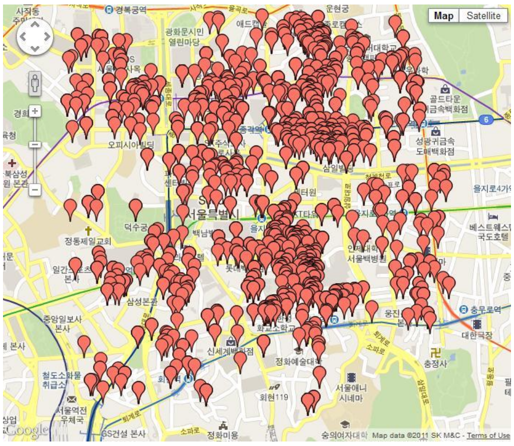

Figure 1. Naively plotting thousands of markers on a map

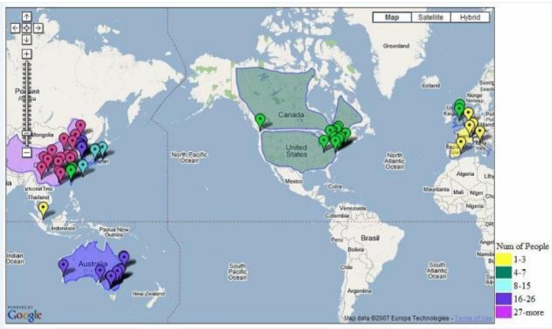

Figure 2. APWeb05 presenters' mapping on Google maps by city using pacemaker

II. Related Work

Zhang and Shi [1] presented a web mapping application based on geographic locations of conference presenters which user Google Maps. It aims to demonstrate advanced features that current web mapping services can provide for location-based decision making. The demonstration is shown in Fig. 2. For the case study of APWEB05 (Asia Pacific Web Conference, 2005) program committee (PC) presenters are used in this paper. Mohan et al. [3] proposed a new data hiding technique for embedding text data in image using conturlet transform.

doi: 10.12720/joams.1.1.107-111

The image is first contourlet transformed and then text is embedded. The result is also compared with LSB manipulation method. Vennetis *et al.* [5] tried to detect the importance of points for places. However, image features like Peak Signal to Noise Ratio (PSNR), variance, skewness [6], [7] and [8] etc. are analyzed and found that the proposed method outperforms the existing methods. The drawback of these papers is that they did not concern about the overlapped markers on the map.

III. Proposed Algorithms

Every day, millions of people around the world use Location-based Services (LBS), which rely on a user's estimated location to provide a better product experience. Google Maps for Web and Mobile is a web map service API [9] which can help users to find their location on a map and then locate places on it according to user's nearby places by request.

In fact, nowadays Google Maps is becoming very popular for providing a great web map services such as local search feature [10] and [11], street maps, a route planner for traveling by foot, car, bike or public transport and an urban business locator. However, a rough usage of Google Maps API features and opportunities cannot be so much helpful when the time comes to display a large number of locations on the map. Naively plotting thousands of markers on a map can quickly lead to degraded user experience. As a result, too many points on the map cause both visual overload and sluggish interaction with the map. To overcome this poor performance, the information displayed on the map needs to be simplified. In this chapter we address each issue one by one and give our appropriate solutions and methods for those problems.

A. Grid-Based Filtering

When creating maps with hundreds or thousands of points, it can be difficult to make it easy for the user to read and understand. Too many icons or markers on the map and they are bound to overlap each other, users won't be able rollover the marker and big chunks of the map will be completely obscured. Also it causes both visual overload and sluggish interaction with the map. To overcome this problem, the information displayed on the map needs to be evenly distributed and clear. In addition, we should make a decision based on user's desire what they want to see on the map when they search nearby places. However, if we display thousands of results on the map based on user's request, it is difficult for users to feel satisfied experience with that map. Fig. 3-1 illustrates large number of markers on the map which might cause to sluggish interaction with the map.

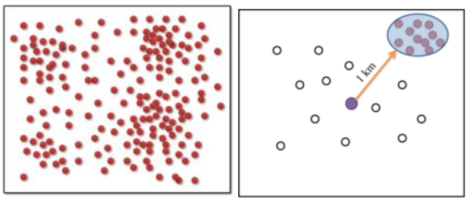

Figure 3. 1. (left) Too many markers on the map; 2. (right) Problems with displaying top k place

First emergency solution could be reducing the number of markers within a certain distance on the map, to provide user a better interaction and visualization. But in this case we cannot just show some results by decreasing their numbers. This should be also based on user's desire. Surely when users search some places within a certain distance, for instance the restaurants or shops, then they want to see places as they expected. From this requirement we learn that we can only display top k places, based on rankings of those places, instead of showing all places.

However, even after showing only top k ranked places, we still have another issue, for example, if we only show top k ranked places, those places might gather in a particular hotspot where are quite far from user's current location. Moreover, some desired places might be missed to display in other areas within a given distances which are located nearby user's current location. Fig. 3-2 indicates that issue with displaying only top k ranked places.

To solve above mentioned issue, we propose our Grid-based filtering algorithm. When we have large number of points on the map, there will be difficulties with interaction and visualization. Grid-based filtering algorithm is supposed to solve above mentioned issues. The basis of grid-based filtering algorithm implies evenly distributions of top k ranked places on the map. By this way we guarantee balanced filling of the map with top k ranked places.

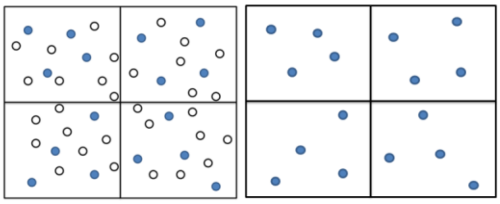

Figure 4. 1. (left) Dividing viewpoint into grid cells; 2. (right) Getting top k ranked places from each cell

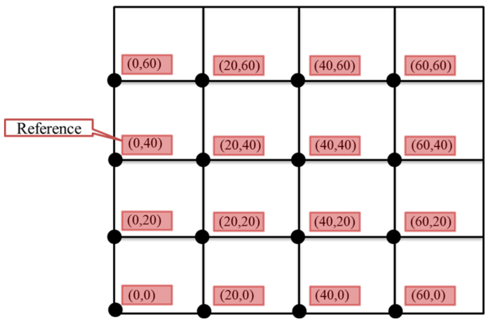

Figure 5. Cell references for mapping places

At first, we divide the current viewport into several gird cells for the effective filtering (see Fig. 4-1). After that, we will get top k places from each grid cell which is shown in Fig. 4-2. To perform Grid-based filtering algorithm effectively, we should go through some processes. Initially, we should define that each place is belonged to which grid cell. In order to mapping all of the points to the belonging grid cells, we will need a reference for each grid cell. A reference should be a unique key for its cell. As reference for a single cell, that cell coordinates which measured according to Cartesian coordinate systems is used for mapping places (see Fig. 5).

To define the reference for each cell below equation (1) are used.

$$Reference(lng_i, lat_i) = \begin{cases} \lfloor lng_i/\Delta x \rfloor * \Delta x \\ \lfloor lat_i/\Delta y \rfloor * \Delta y \end{cases} \quad (1)$$

in this equation, $lng_i$and lat_i are each place's longitude and latitude, Δx and Δy are grid cell's size in X and Y axis respectively.

One of advantages of using the equation (1) is to save computation time for checking each point which reference of cell is belonged to. In other words, we define the reference from each point's longitude and latitude using the equation (1).

Fig. 3-6 shows the processes of mapping the places to the belonging cell reference. Assuming, we have set of points, we need to map each point to a belonging cell reference. At first, we will give each point's location, longitude and latitude to equation (1). This equation returns us a cell reference for each point then a given point efficiently will be mapped to that cell reference.

After a completion of mapping process we will obtain sorted list of mapped places for each reference point. Sorting order of the list is in descending order of ranks of each place. Apparently, we can easily select top k ranked places from each list for each reference.

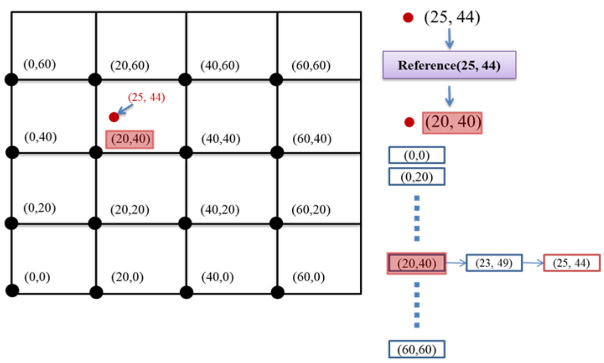

Figure 6. Mapping places to the belonging cell references

B. Clustering Geographically Overlapped Places

Sometimes it is possible that locations on the map are so close to each other which results in blocking each other when displaying with markers. Such places are considered overlapped places on the map. This makes user unable to choose one of overlapped place with clicking on single marker. Fig. 7 indicates the issues we have mentioned, the points within circle are found as overlapped points.

To solve this problem, we propose the clustering overlapped places. In order to perform clustering efficiently we use DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [4] density based clustering algorithm [12], [13], [14] and [15]. DBSCAN algorithm is known for good performance when working with large number of objects.

Clusters are identified by looking at the density of points. Regions with a high density of points depict the existence of clusters whereas regions with a low density of points indicate clusters of noise or clusters of outliers. This algorithm is particularly suited to deal with large datasets, with noise, and is able to identify clusters with different sizes and shapes.

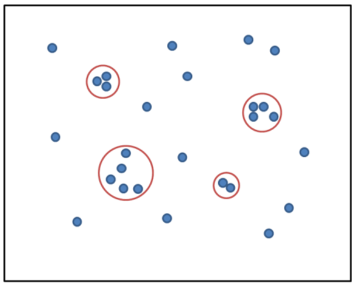

Figure 7. Overlapped places on the map

The key idea of the DBSCAN algorithm is that, for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points, that is, the density in the neighborhood has to exceed some predefined threshold.

This algorithm needs three input parameters:

- k, the neighbor list size;
- Eps, the radius that delimitate the neighborhood area of a point;
- MinPts, the minimum number of points that must exist in the Eps-neighborhood.

The clustering process is based on the classification of the points in the dataset as core points, border points and noise points:

- A point is a core point if it has more than a specified number of points (MinPts) within Eps;
- A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point;
- A noise point is any point that is not a core point or a border point.

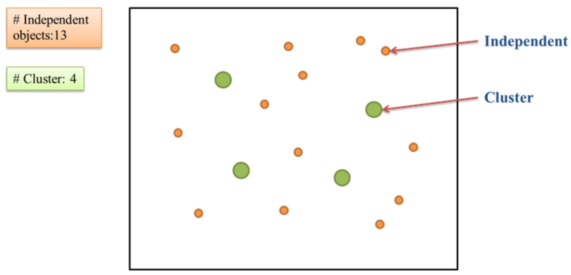

Figure 8. Clustering overlapped places

Using DBSCAN we can specify points which are too close to one another as overlapping on the map. Defined overlapped points will be clustered into a single point. As a result, we can only display one clustered point instead of showing all overlapped points. In Fig. 8 the result of DBSCAN is shown.

IV. Experimental Results

For our experiment we needed a massive spatial dataset. Therefore we collected a large number of restaurants' address and other information in whole South Korea. To identify each Restaurant's location on the map, we converted all addresses into geographic coordinates using Google Goecoding API which provides a direct way to access a goecoder via HTTP request. Thus, we have about 74000 locations information of all restaurants. For following experiments we will use those spatial dataset to test our approach and algorithm.

To get top k ranked places in our algorithm, we gave a random rank value to each restaurant for the experiment. Also we made a use of Google Maps API to represents our experimental results. We have trained and tested our proposed algorithms according to restaurant locations in different areas in South Korea as Table I. shows.

TABLE I. DATASET

Area	Restaurants
Kangnam Station	5,371
Hongdae Station	3,624
Pundang Station	2,922
Myeongdong	2,761
Sinchon Station	2,559
Daehak-ro	2,370
Jongno	2,190
Apgujeong	2,058
…	…
Total	74,000

At first, as hotspot we chose area around Seoul Subway Station. Within 1 km distance, the map showed about 8500 places in Fig. 4-1 and Table II. There are too many restaurants. We can see from the result that visualization is not good, besides, the interaction is slower. Then we displayed only top 125 places to cut the number of places. The results are displayed on the second map. We can see that some places are gathered in particular area and not evenly distributed. It seems that there is less number of restaurants because of overlapped places. After we applied proposed algorithms, we achieved a good result of visualization on the map. They are also evenly distributed within distance.

For the next experiment we chose the area around Jonggak station. At this time we displayed all places within 300 m distance. According to result data in Table III, in Fig. 10, on the first map, about 1500 restaurants were loaded which is still a lot. When we get top 125 places, the result is better than before. But some restaurants were missed in some areas which are indicated by ellipse line on the second map, whereas the result of proposed algorithm showed 115 restaurants all parts of viewport.

The last experiment was conducted around Busan area. In Fig. 11, it seems that even within 5 km distance there are less number of restaurants in contrast previous results. When we used naive top k to show restaurants, some overlapped places were existed on the map. After we applied clustering algorithm, it is found 6 clusters which include the overlapped places.

Figure 9. Results around seoul station area

TABLE II. RESULT DATA OF SEOUL AREA

	All	Na ïve Top-k	Proposed algorithms
#markers	8,500	125	116
#clusters	-	-	7
Distance	1 km	1 km	1 km
Zoom level	15	15	15

Figure 10. Results around Jonggak station area

TABLE III. RESULT DATA OF JONGGAK AREA

	All	Na ïve Top-k	Proposed algorithms
#markers	1,500	125	115
#clusters	-	-	0
Distance	300 m	300 m	300 m
Zoom level	17	17	17

Figure 11. Results around Busan station area

TABLE IV. RESULT DATA OF BUSAN AREA

	All	Na ïve Top-k	Proposed algorithms
#markers	370	125	77
#clusters	-	-	6
Distance	5 km	5 km	5 km
Zoom level	13	13	13

As an addition to experimental results, we want to show changes in response time between 3 approaches as depicted in Fig. 12. The graph represents three curve lines for three methods that are Naive Top-k places, Grid-based filtering and Grid-based Filtering & Clustering respectively. From the lines we can find that the response times of Naive Top-k and Grid-based filtering is nearly equal and also at first they are fluctuated then increased slightly. The blue line indicates response time of algorithm when Grid-based filtering and Clustering are combined. This method takes 20ms more time than others. Heuristically, the difference is a little thus acceptable to consider that method as an efficient way of displaying large number of places on the map.

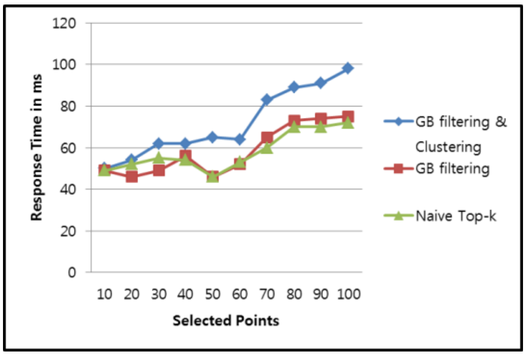

Figure 12. Response time for selected points

V. Conclusion

In this paper it is claimed that straightforward usage of Google Maps API to display a large number of objects in a map results in poor user experience and lesser performance due to overlapped or hidden locations. As a solution for this issue, it is proposed algorithms Grid-based filtering and DBSCAN density-based clustering. Applying proposed algorithms on training data resulted in effective and efficient display of overlapped and over approximated locations in the map.

Acknowledgment

This research is partially supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency)" (NIPA-2012-H0301-12-4014), and supported by Mid-career Researcher Program through NRF grant funded by the MEST (2011-0029729).

References

[1] Z. Zhang and H. Shi, "Geospatial visualization using google maps: A case study on conference presenters," in *Proc. Second International Multi-symposium on Computer and Computational Science*, Lowa, Sep. 2007, pp. 472-476.

[2] H. Hotta and M. Hagiwara, "Online geovisualization with fast kernel density estimator," in *Proc. IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, Milano , 2009, pp. 622-625.

[3] M. Mohan and Anurenjan P. R, "A new algorithm for data hiding in images using contourlet transform," in *Proc. Recent Advances in Intelligent Computational Systems*, Trivandrum, Sep. 2011, pp. 411-415.

[4] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Computer 1996 Journal*, pp. 226–231, 1996.

[5] P. Venetis, H. Gonzalez, Ch. S. Jensen, and A. Halevy, "Hyper-local, directions-based ranking of places," *VLDB Endowment*, vol. 4, no. 5, pp. 290-301, Feb. 2011.

[6] T. Kim and H. White, "On more robust estimation of skewness and kurtosis," *Finance Research Letters*, vol. 1, no. 1, pp. 56-73, 2004.

[7] P. T. von Hippel, "Mean, median, and skew: Correcting a textbook rule," *Journal of Statistics Education*, vol. 13, no. 2, 2005.

[8] Wikipedia. Skewness. [Online]. Available: http://en.wikipedia.org/wiki/Skewness.

[9] Google Maps API. [Online]. Available: http://code.google.com/apis/maps/index.html.

[10] R. Silva, J. M. Pires, and M. Y. Santos, "Spatial clustering to uncluttering map visualization in SOLAP," in *Proc. 2011 International Conference on Computational Science and Its Applications*, Ho Chi Minh, 2001, pp. 253-268.

[11] M. Vandaniker, J. Lee, K. Wongsuphasawat, and A. Olea, "Exploring clusters in geospatial datasets," *SINGGAPH Course Notes*, Los Angeles, 2004.

[12] Z. Zhang and H. Shi, "Geo-visualization and clustering to support epidemiology surveillance exploration," in *Proc. 2010 International Conference on Digital Image Computing: Techniques and Applications*, Sydney, 2010, pp. 381-386.

[13] Z. Chen, L. Wu, and D. Zhang, "Spatial data partitioning based on the clustering of minimum distance criterion," in *Proc. International Conference on Computer Science and Service System*, Nanjing, Jun, 2011, pp. 2583-2586.

[14] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen, "Discovering personally meaningful places: An interactive clustering approach," *ACM Transactions on Information Systems*, vol. 25, no. 3, Jul, 2007.

[15] K. Mumtaz and Dr. K. Duraiswamy, "An analysis on density based clustering of multi dimensional spatial data," *Indian Journal of Computer Science and Engineering*, vol. 1, no. 1, pp. 8-12, 2010.

Gayratjon Kholmuradov was a master student of the Department of Advanced Technology Fusion at Konkuk University in Seoul, Korea, since March 2010. He received the Bachelor degree in Computer Science at Tashkent University of Information Technology, in 2009. And he finished Master course in Konkuk University, in 2012. His research interests are database system, web search engine, data mining and information retrieval.

Xian Chen is a Ph.D. Candidate in the Department of Advanced Technology Fusion at Konkuk University in Seoul, Korea, since September 2008. She received a Bachelor degree from the Department of Software Engineering at Chongqing University of Post and Telecommunication in 2007 and a Master degree from Konkuk University in 2010. Her current research interests are social network analysis and data mining.

Hyoseop Shin is an Associate Professor in the Department of Internet and Multimedia Engineering at Konkuk University in Seoul, Korea, since March 2005. He received his Bachelor, Master, and Ph.D. degrees from the Department of Computer Science and Engineering at Seoul National University, in 1994, 1996, and 2002, respectively. In 2002, he won the best database paper award from the Korea Information Science Society. He serves as a program committee member for several international conferences including DASFAA and APWEB. He is also a member of the standing committee of KIISE Database Society of Korea. His current major research interests are web search and data mining.