Heattile, a New Method for Heatmap Implementations for Mobile Web-based Cartographic Applications

Sebastian Meier, Frank Heidmann, Andreas Thom[1]

Abstract Mobile handheld computing devices are becoming a more important part of our digital infrastructure. Web-based map services and visualisations for spatial data are being optimised for use on these new mobile devices. Despite the fast development of mobile devices, compared to laptop and desktop computers they remain slower in terms of processing power and have limited memory capacity. Bandwidth in most areas is still limited, regardless of the fast expansion of high-speed internet for mobile devices. To overcome these two limitations an improvement in existing spatial data visualisations is necessary. This chapter compares existing methods and presents a new approach to generating and delivering web-based heat map visualisations for spatial data that is optimised for mobile devices.

1 Introduction

Heat maps, also known as density maps, density surfaces and shaded isarithmic maps, are usually used for visualising evenly distributed spatial data points with varying values (e.g. weather maps with varying temperatures) or spatial data-point clusters representing density (e.g. network-coverage-maps with one data point per antenna). With the help of heat maps a user can identify hotspots, distribution, and correlations within a spatial dataset (Pettit et al. 2012). Through further development of web technologies these visualisations have also become of interest for web-based geo-visualisations.

Since the introduction of the iPhone and the beginning of the still growing smartphone segment, maps, or rather navigation, has been an essential part of the applications (apps) available. Today, there are numerous apps, map providers, navigation providers and many commercial and open source tools, which allow

S. Meier, F. Heidmann, A. Thom
Interaction Design Laboratories
University for Applied Science
Potsdam, Germany
e-mail: meier@fh-potsdam.de

T. Bandrova et al. (eds.), *Thematic Cartography for the Society*, Lecture Notes in Geoinformation and Cartography, DOI: 10.1007/978-3-319-08180-9_4,

users and developers to bring maps to mobile devices. In addition to maps and navigation, geovisualisations (visualisations of spatial data), are also starting to appear on mobile devices. Although they have been present on the internet for years, the existing approaches to geovisualisations have to overcome a number of technological and conceptual obstacles until they can fully embrace the new mobile medium. This chapter concentrates on the *heat map* geovisualisation, and a new method to overcome the obstacles described above.

2 Existing Methods

The existing approaches to generating and delivering heat maps for web-based map systems can be categorised into two methods. The first method, referred to here as the *client-side method,* delivers raw spatial data, depending on the technology used, in JSON, XML, CSV or other similar formats, to the user's browsers. The browser then transforms this data, using client-side technology such as JavaScript, into the actual heat maps (D3 2014, Heatmapjs 2014, Heatcanvas 2014, WebGL 2014, Google1 2014).

The second method, referred to here as the *tile-based method*, involves pre-rendering the heat maps and then storing the visualisation data in tiled images on a server. The image-data is then delivered to the user's browser in a standard tile-based system used by most common web map systems (CartoDB 2014, Nokia 2007).

3 Problem Definition

The existing approaches work very well in stationary browser applications and currently, in 2013, there are already various commercial and open source implementations of these methods. When we try to use the existing methods on mobile handheld web browsers, however, there are two major difficulties. One is the *limited performance factor* of mobile handheld devices. Even though the latest developments have turned what were not long ago still mostly text-input devices into small powerful computers, they are still not comparable to laptops or desktop computers. This limiting factor is a problem more relevant for the *client-side method*. Storing a large amount of data for real time use in the browser can be difficult on mobile devices and is limited, depending on the device. Rather seriously, this involves generating heat maps in real time, because this process is computationally intensive and also limited.

The second major limitation is the *bandwidth*. Even with the implementation of the latest network technology the bandwidth in rural areas and densely populated areas is still limited. As a result, the *tile-based method* is problematic because the delivery of heat maps through the image data used by the *tile-based method* is

very bandwidth intensive.

In addition to these two major limiting factors there are two additional factors involving the technological features of modern handheld devices. On the one hand, we have more and more *high-resolution displays* being built into modern devices. These high-resolution displays require the *tile-based method* to deliver even bigger images, making it even more bandwidth-intensive. On the other hand, multitouch displays and modern web-map systems allow a user to smoothly zoom in and out as well as interact with the data visualised on screen. While zooming is no problem for the *tile-based method,* further interaction is limited. In the *client-side method* interaction is possible, as is zooming, even though zooming can be very resource consuming.

Table 4.1: Comparison of existing heat map methods

	Tile-based Method	Client-Side Method
Computationally intensive	Low	High
Bandwidth intensive	High	Medium
Interactivity	Low	High
High-Resolution	Yes, but requires more bandwidth intensive images	Yes, Vector-Data
Real time	No, due to pre-rendering	Yes

As shown above, according to the comparison of existing approaches on mobile web browsers, a new method should have: 1. A *small data footprint* regarding bandwidth, 2. require *as little client-side calculation as possible*, as well as being 3. a *high-resolution visualisation* and it should also enable, *4. interaction with the visualised data* itself.

4 The New Heattile Approach

The new approach consists of three steps. The first step is a raster-based clustering and generalising of the data, which is then rendered on a request-basis into GeoJSONs (GeoJSON 2014), delivered to the browser and then visualised client-side for further interaction.

4.1 Server-Side Pre-Clustering

For the following steps we assume that the data is stored in a database. The sample

queries are MySQL-Queries. The first step is clustering the spatial data. We use the tiling approach, which was initially introduced through the WMS Specification (Opengis1 2014) by the Open Geospatial Consortium (OGC 2014) in 2000. It became popular through modern map providers and their implementation of the Web Map Tile Service Implementation Standard (Opengis2 2014), such as that of Google (Google2 2014), Bing (Bing 2014), OpenStreetMap (OpenStreetMap 2014) and Mapbox (Mapbox1 2014). The tiling approach is using a web-Mercator projection, which results in an orthogonal and evenly distributed coordinate system (see Fig. 4.1 right). This coordinate system is then split into smaller squares, using a Quad-Tree-Method (Potmesil 1997), starting at one square at zoom-level x, four squares at zoom-level x+1, sixteen squares at zoom-level x+2 resulting in zoom-level x, tile number t = 2^((x-1)*2). In each zoom-level every square has a unique identifier. Starting at zero in the upper left corner, row after row, every column receives the consecutive number as an identifier (see Fig. 4.2).

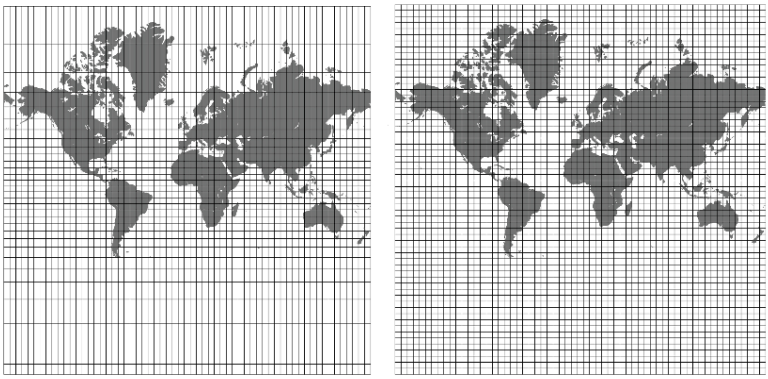

Fig. 4.1 Coordinate-system comparison

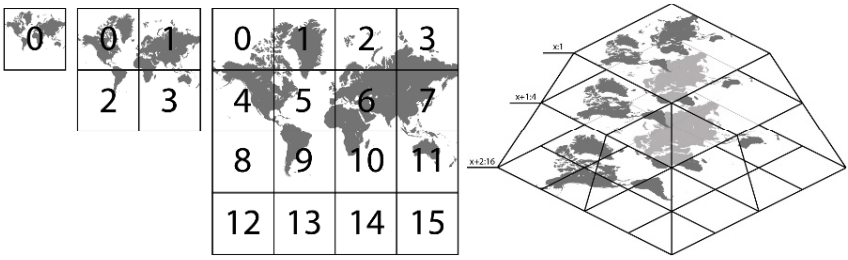

Fig. 4.2 Tiling-System for Spatial Data

This process will be applied to our spatial data; every data point receives an id for every zoom-level, allowing us to receive all data points for a specific tile-id at a specific zoom-level through a very simple query. This technique is inspired by the hierarchical clustering process (Delort 2010).

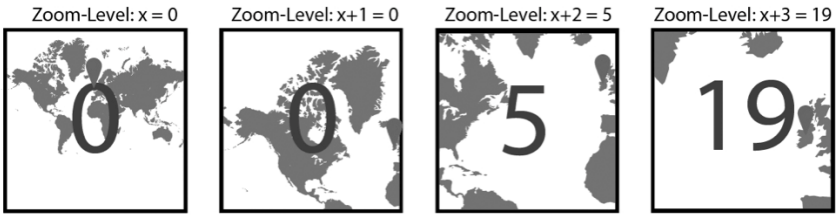

Fig. 4.3 Grouping data points on a per-tile basis: Tile-ID per zoom-level

1. [SELECT GROUP(*), * FROM geo_data WHERE z_(ZOOM_ID) = (TILE_ID) GROUP BY z_(ZOOM_ID)]

After the initial conversion process, the same process of generating tile ids for spatial data points can be applied to new points whenever they are added without touching the already clustered data.

4.2 Server-Side GeoJSON

In order to keep a small data footprint, this method pre-clusters the data into the tiles generated in the step above. Instead of delivering all data points, the method only delivers a number of data points per tile. Instead of returning all tiles for the current zoom level, the method only returns the tiles for the currently visible area on the mobile device plus an additional margin around the requested area. The extra margin gives the user smoother interaction when panning the "slippy map" (Slippy Map 2014). Using this approach, developers can decide how much data they want to deliver to the user's device, either by shrinking or expanding the margin, and by defining the size of the tiles being displayed at each zoom-level.

To keep the amount of rendering as small as possible, the method does not deliver raw data that needs to be turned into a visualisation on the client's side, but instead delivers GeoJSONs. The method delivers a range of GeoJSONs, which can be defined by the developer. In the example visualisations in Figures 4.4 and 4.5 we used an evenly distributed range from 0 to 9, 0 holding the tiles with the least data points and 9 the tiles with the most data points. For faster range-computation we recommend caching the maximum value of data points per tile for every zoom-level.

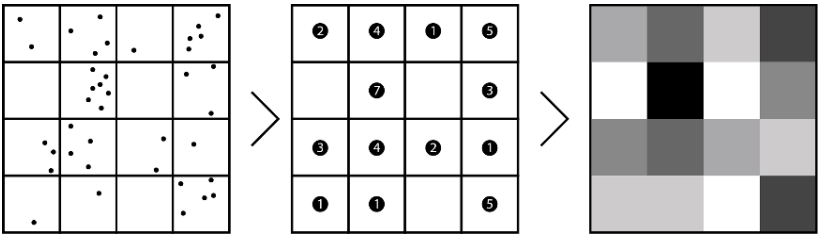

Fig. 4.4 Server-side rendered GeoJSON-Layers

2. [SELECT GROUP(*), * FROM geo_data WHERE z_(ZOOM_ID) = (TILE_ID) WHERE latitude < (LATITUDE_MAX+MARGIN) AND latitude > (LATITUDE_MIN-MARGIN) AND longitude < (LONGITUDE_MAX+MARGIN) AND longitude > (LONGITUDE_MIN-MARGIN) GROUP BY z_(ZOOM_ID)]

4.3 Client-Side Rendering & Interaction

We used Leaflet (2014), a common web-mapping framework, in the user's browser for displaying a map and requesting the GeoJSON-layer with the heat map for the currently visible area. On the client-side we can decide on visual features of the GeoJSON, manipulate those features, and add interactions for each range step (e.g. click or double-click). The colours can, for instance, be modified on the client: in the example below we used higher opacity for layers with higher density. The mono-colour theme was used instead of the common blue to red colour range for heat maps, as the mono-colour range performs better in geovisualisations (Borland 2007, Harrower and Brewer 2003).

Fig. 4.5 Two different heat map visualisations on top of maps

5 Advantages and Disadvantages

The data footprint for the tiled GeoJSONs is very small (see Table 4.2), the processing-power required is little (see Table 4.2) and all processes can be done in real time. The GeoJSONs are visualised as vectors, which gives us high-resolution visualisations. Interaction in our method is limited; we can differentiate between interactions on the layers below the heat map and interactions on each range step, but what we cannot do is receive interactions on a specific tile, which would require splitting the GeoJSON into one GeoJSON per tile, which would result in a higher data footprint as well as more processing. As a work-around we can translate the latitude and longitude values of the interaction into a tile-id and request further information from the server. Another disadvantage is the limitation in terms of visualisation styles. Due to the clustering method used, we are limited to square-based visualisations. For additional visualisations we have created a slightly altered method: every even row is offset by half the square size to the right, allowing us to create other types of visualisations, (Fig. 4.6).

Fig. 4.6 Alternative offset grid with hexagon heat map visualisation

6 Comparison

It must be noted that the amount of data used influences the performance of the three methods. In this chapter we used datasets including from 200 to 200,000 data points for the central area of Berlin, for the testing described below (Table 4.2-4.4), and we used a subset of 14,864 data points. The testing device was an iPhone 5s.

In a detailed comparison (Table 4.2) we can see that the GeoJSONs from our new method are smaller than in the other two data formats. For the image tiles,

there is no information available for the uncompressed size as the tiles are compressed on the server. Execution time does not include loading time. The time for the raw data approach can be split into 20-30ms for parsing the data into the visualisation system and an additional 480-570ms for the rendering process. Parsing time is time required for JavaScript to go through the received JSON file and pass it on to the visualisation functions. The time for the raw data was calculated using the Heatcanvas (2014) library. The other libraries mentioned in Section 2 (D3 2014, Heatmapjs 2014, WebGLHeatmap 2014, Google1, 2014) were also tested, and we received varying results (all >200ms) for the rendering time; the parsing time in all the libraries used was nearly the same (20ms-40ms).

While the tile-based method is only limited by the capacity of the pre-rendering process and the new method is only limited by the performance of the database that holds the data, the performance of the client-side method, as described in Section 3, is extremely dependent on the size of the dataset. While performing well on very small datasets, the bigger the datasets become, the worse the performance (see Table 4.4), which at some point makes the client-side method unusable.

Table 4.2 Filesize and execution time of GeoJSONs, varying in tile size, compared to image tiles and raw data for a 500x500px region

Size in Kilobytes	GeoJSON (x)	GeoJSON (x+1)	GeoJSON (x+2)	Image Tiles (lowres)	Image Tiles (highres)	Raw Data
Size	14,4	35	92,1	n/a	n/a	544
Size GZIP-compressed	1,3	5,1	13,5	159,1	430,7	51,8
Execution time in ms	7-9	8-10	9-11	10	15	20-30 (500-600)

Table 4.3 Comparison of the existing methods with the new method

	Heat Tile Method	Tile-based-Method	Client-Side-Method
Computationally intensive	Low	Low	High
Compare to Table 4.2			
Bandwidth intensive	Low	High	Medium
Compare to Table 4.2			
Interactivity	Medium	Low	High
High-resolution	Yes, vector data	Yes, but requires more bandwidth intensive images	Yes, vector data
Real time	Yes	No, due to pre-rendering	Yes

Table 4.4 Execution Time for the client-side method with raw data for a 500x500px region

Data points	14864	29728	44592
Execution time in ms	20-30 (500-600)	44-50 (950-1050)	65-80 (1430-1550)

If we compare the new method with the existing methods, the new method performs better or equally as well in every category, except interactivity. The only disadvantage that remains is the variety in terms of visualisation and interaction.

7 Application and Future Works

There is a wide field of applications for the new method. In the MSNI research project (MSNI 2014), we used the method for displaying the density of different types of locations within a city (e.g. restaurants, bars, museums, etc.) which allows the user to explore a city while being on the road (similar attempts have been made by Nokia (2007)). In a similar use case researchers developed a visualisation for social-network data from foursquare to inform tourists about nearby locations and activities (Komminos et al. 2013). Another possible use case is the visualisation of environmental data on mobile devices to allow users to explore their environment while on the move.

To overcome the persistent obstacles we tried using the clustered data as raw data for existing JavaScript-based heat map libraries (Fig. 4.7). This resulted in even smaller data footprints, as we were able to strip the GeoJSON data and turn it into simple JSON (JSON2 2014) files. On the other hand it also resulted in more client-side processing, which slowed the visualisation.

In the future we plan to apply new methods of shrinking the file-size of the GeoJSONs even further through compression (Hanov 2010, json.hpack 2014, JSON1 2014) or the BSON (2014) method. Additionally we need to look into caching systems for further optimisation and better scalability of our approach, similar to the techniques already used in the tile-based method (Liu et al. 2007).

The latest work on vector-tiles, for example that by Google Maps (Google2 2014) or MapBox (MapBox2 2014), have improved the data footprint of the tile-based-approach. If the new vector method by Google and MapBox can be applied to the tile-based method for heat maps this could improve the limitation in terms of bandwidth, although it will not improve the ways a user can interact with the visualisations generated by this method.

Fig. 4.7 Using the method to deliver clustered raw data for client-side heat map libraries

8 Conclusion

The method described in this chapter presents a useful alternative to existing approaches for visualising heat maps on mobile handheld devices. The new approach performs better in the major categories we identified as obstacles for geovisualisations on mobile devices. The fact that the generation of heat maps can be done in real time is a particularly interesting new possibility for applications. The approach also performs well in the other categories. The only downside is the variety of visual representations that it is possible to achieve, due to the raster-based clustering.

The code required for using the method is available on GitHub under GPL/MIT: https://github.com/sebastian-meier/HeatTiles.

References

Bing (2014) http://maps.bing.com, Accessed: 10.01.2014

Borland D, & Taylor MR (2007) Rainbow color map (still) considered harmful. IEEE Comput Graph Applications, 27(2), 14-17. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/17388198, Accessed: 10.01.2014

BSON (2014) BSON specification. http://bsonspec.org Accessed: 10.01.2014

CartoDB. http://cartodb.com/visualise Accessed: 10.01.2014

D3 (2014) D3 – Hexbinding. http://bl.ocks.org/mbostock/4330486 & http://bl.ocks.org/mbostock/4248145 Accessed: 10.01.2014

Delort JY (2010) Hierarchical cluster visualization in web mapping systems. In Proceedings of the 19th international conference on World wide web (WWW '10). ACM, New York, NY, USA, 1241-1244. DOI=10.1145/1772690.1772892

GeoJSON (2014) http://www.geojson.org Accessed: 10.01.2014

Google1 Heatmaps. https://developers.google.com/maps/documentation/javascript/examples/layer-heatmap, Accessed: 10.01.2014

Google2 (2014) Maps. http://maps.google.com, Accessed: 10.01.2014

Hanov S (2010) Compress JSON with automatic type extraction. URL http://stevehanov.ca/blog/index.php?id=104 Accessed: 10.01.2014

Harrower M, Brewer CA (2003) ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps. Cartographic Journal, 40(1), 27-37. doi:10.1179/000870403235002042

Heatcanvas (2014) https://github.com/sunng87/heatcanvas Accessed: 10.01.2014

Heatmapjs (2014) http://www.patrick-wied.at/static/heatmapjs/ Accessed: 10.01.2014

JSON1 (2014) Compression Algorithms. http://web-resource-optimisation.blogspot.com/2011/06/json-compression-algorithms.html Accessed: 10.01.2014

JSON2 (2014) http://json.org/ Accessed: 10.01.2014

json.hpack (2014) https://github.com/WebReflection/json.hpack/wiki Accessed: 10.01.2014

Komninos A, Besharat J, Ferreira D, Garofalakis J (2013) HotCity: enhancing ubiquitous maps with social context heatmaps. In Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia (MUM '13). ACM, New York, NY, USA, , Article 52 , 10 pages. DOI=10.1145/2541831.2543694

Leaflet (2014) http://www.leafletjs.com Accessed: 10.01.2014

Liu Z, Pierce ME, Fox GC, Devadasan N. (2007) Implementing a caching and tiling map server: a Web 2.0 case study (pp. 247–256). Presented at the 2007 International Symposium on Collaborative Technologies and Systems (CTS), IEEE. doi:10.1109/CTS.2007.4621762

Mapbox (2014) http://www.mapbox.com Accessed: 10.01.2014

Mapbox (2013) Vector Tiles for Mapbox Streets. https://www.mapbox.com/blog/vector-tiles, Accessed: 10.01.2014

MSNI (2014) http://www.m-s-n-i.de, Accessed: 10.01.2014

Nokia (2007) Identifying interesting locations based on commonalities in location based postings. http://www.google.com/patents?vid=USPATAPP11784086, Accessed: 10.01.2014

Open Geospatial Consortium (2014) http://www.ogc.org Accessed: 10.01.2014

Opengis1 (2014) Web map service (WMS) implementation specification. http://www. opengeospatial.org/standards/wms Accessed: 10.01.2014

Opengis2 (2014) Web Map Tile Service Implementation Standard: http://www.opengeospatial.org/standards/wmts Accessed: 10.01.2014

OpenStreetMap (2014) http://www.openstreetmap.org Accessed: 10.01.2014

Pettit C, Widjaja I, Russo P, Sinnott R, Stimson R, Tomko M (2012) Visualisation support for exploring urban space and place. In XXII Congress of the International Society for Photogrammetry and Remote Sensing, Melbourne, Australia

Potmesil M, Bell Laboratories, Lucent Technologies (1997) Maps Alive: Viewing Geospatial Information on the WWW. Sixth International World Wide Web Conference. April 7-11, 1997 Santa Clara, California USA.

Slippy Map (2014) http://wiki.openstreetmap.org/wiki/Slippy_Map Accessed: 10.01.2014

WebGLHeatmap (2014) http://ursudio.com/webgl-heatmap-leaflet/ Accessed: 10.01.2014