HEAT MAP SEGMENTATION

Gil Wolff
Amazon Last Mile
Bellevue, WA
wolffg@amazon.com

ABSTRACT

Many geospatial datasets can be represented as a heat map, such as rainfall, population density, terrain elevation, and others. These heat maps tend to form clusters of high density areas among a background of low density areas. This gem presents an automatic way to detect such clusters, and segment the heat map into areas. Experiments are conducted for two datasets which correlate to population density and show that the segmentation aligns with metropolitan areas and is stable to the choice of dataset. The segmentation described in this gem can potentially aid geospatial algorithms by supplying a smart divide-and-conquer strategy, such that the algorithm does not need to run for the entire Earth, but rather there can be a fine-grained model for each area.

1 Introduction

Earth is heterogeneous. Climate varies non-monotonically across Earth's surface, human population forms urban clusters, and mountains form ranges. Geospatial algorithms can be slow to run at the scale of the entire Earth. In meteorology, for example, it is common to use local models to describe local effects in fine-grained detail [1].

This gem presents a segmentation algorithm that partitions Earth into areas, such that each can be described by a local model. The segmentation algorithm operates on heat maps, and it will tend to divide between areas along contour lines which have minimal density. Segmentation is a distinct problem from clustering, in that gaps between clusters are allowed, but gaps between segmented areas are disallowed.

The choice of the variable that is used to generate the heat map determines the meaning of the output. Some examples are presented on heat maps that describe human density. Within this domain, clusters of human density correspond to metropolitan areas.

Some problems that can benefit from having a local fine-grained model include (a) Climate forecasting, such as predicting next year's rainfall (b) Recommendation problems, such as ranking the most popular tourism points of interest [2], and (c) Estimation algorithms, such as predicting real estate prices [3].

It is intuitive to wish to cut between areas along lines which will least interfere with the problem of interest. For the rainfall example, this might be interpreted as a cut between areas along a contour line that has minimum average rainfall. For the recommendation example, this might be interpreted as a cut along a contour line that is farthest from tourist attractions or from hotels. For the prediction example, this might be interpreted as a cut along a contour line where building density is minimal.

For some problems, a proxy variable may be chosen, such as population density as a proxy for building density. However, it is best to choose a variable that best describes the problem we are trying to solve, and then find the contour lines that delineate areas with respect to that variable.

Once the variable of interest is chosen, there is left to generate a heatmap, and find the contour lines alluded to above. The algorithm which is presented in this gem is based on Lindeberg's watershed algorithm, which was originally invented to detect blobs in images [4, 5]. The algorithm is described in detail in Section 2. Smoothing is used to introduce a minimal size for a blob to be considered a detected area.

2 Heat map segmentation algorithm

This section describes the heat map segmentation algorithm. The input is a list of latitudes and longitudes. We will explore in this section a specific example: a list of locations of place names [6] in the UK maintained by the office for national statistics. First, the points are projected into a local Cartesian coordinate system using the equidistant cylindrical projection [7] given by

$$x = (\quad - \quad_0)\cos\ \_0$$
$$y = \quad - \quad_0$$

centered around a point $(\ \_0;\ \_0) = (53^\circ N; 2^\circ W)$ which is roughly in middle of the UK. Here, denotes longitude and denotes latitude. This is done so to enable image processing techniques, such as Gaussian smoothing [8], in the Euclidean plane. It is possible to do Gaussian smoothing on a sphere [9] but when dealing with problems that are smaller than a large continent, a local Cartesian approximation works fine.

After projecting the points to a local Cartesian coordinate system, the UK is pixelated using a grid spacing of 1 km. This is measured at $(\ \_0;\ \_0)$ and varies with latitude on a sphere, but is fixed in the Cartesian approximation. The value of each pixel is the count of the number of place names that fall in that grid cell. This image is smoothed using a Gaussian filter of radius . If two blobs are separated by a distance much smaller than this, they will be smoothed into the same blob.

Lindeberg's algorithm is run on this image to enumerate the blobs and mark the contour lines that delineate a blob from its neighbors. The algorithm is to iterate on pixels in decreasing order of density. If a pixel has no higher-density neighbors, it is assigned a new "area index." If the pixel has a neighbor with higher density, it is assigned the area index of the neighboring pixel with highest density. Some care must be taken when there is a plateau of equal density: first expand to fill the plateau, then the entire plateau obtains a new index or copies the index of its highest-density neighboring pixel. This is made explicit in Algorithm 1.

The algorithm can be understood by the watershed analogy. Water flows downhill, starting from the highest hill (the pixel of highest density). It stops at the line of minimal height, and the area that is covered is assigned an area index of 0. Repeat for other hills in decreasing order of height, and assign area indices $1; 2; \ldots$.

Algorithm 1: Lindeberg's watershed algorithm

Input : d, an N × N array describing pixel density ($d_p \geq 0$)

Output : w, an N × N array where $w_p = 0; 1; \ldots$ is a watershed index.

```
w_p ← uninitialized ∀p
max-watershed-index ← 0
for pixel in arg-sort-descending(d) do
    if w_pixel ≠ uninitialized then
        continue
    end
    plateau ← flood-fill(d, pixel)
    neighbor-pixels← neighbors(plateau)
    highest-neighbor ← arg min_{p∈neighbor-pixels}(d_p)
    if d_pixel < d_highest-neighbor then
        w_p ← w_highest-neighbor; ∀p ∈ plateau
    else
        w_p ← max-watershed-index; ∀p ∈ plateau
        max-watershed-index++
    end
end
```

An example is shown in Figure 1. The dataset is a downloaded snapshot of the UK index of place names [6] from July 2016. There were 87,000 place names with valid location. As points, they are shown in Figure 1(a). As a smoothed heat map they are shown in Figure 1(b). Finally, Lindeberg's algorithm segments the heatmap into areas, and the boundaries between areas are shown in Figure 1(c).

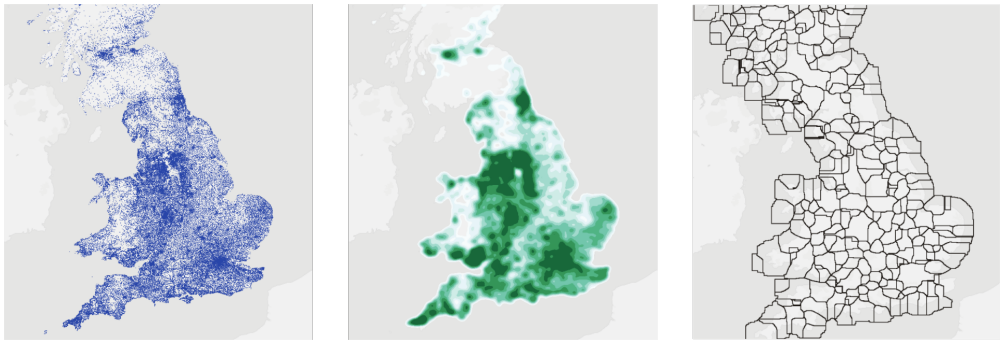

Figure 1: Heat map segmentation in the UK. (a) Locations of 87,000 place names. (b) Smoothed heat map, using Gaussian smoothing of radius = 5 km. (c) Areas detected by Lindeberg's blob-detection algorithm. Lines are plotted at the boundary between pixels which do not share the same area index.

An alternative to this algorithm is to use DBSCAN [10]. However, DBSCAN does not scale well to large datasets, and it performs best on datasets where the clusters have similar densities, which is not the case for the examples presented here.

3 Comparison across datasets and smoothing parameters

Notice that Figure 1(c) metropolitan areas such as London are clearly visible as large polygons. In rural areas, some polygons may be large despite describing a low-density region. Heat map segmentation on a dataset of place names detects metropolitan areas because there are more place names in urban areas than in rural areas. There are other datasets which correlate to population, and comparing them will give insight into the stability of this algorithm with respect to the choice of dataset. The experiment presented in this section compares the output of the algorithm across different datasets, and with different smoothing parameters. Two open datasets are presented: The index of place names in the UK presented in the previous section, and a list of all highway nodes in Open Street Maps [11] for the UK [12].

The term "metropolitan area" also depends on the smoothing parameter . Increasing will decrease the number of metropolitan areas detected, and will increase the average area of each metropolitan area. Two values of are presented: 5 km and 10 km.

There is a third parameter discussed in Section 2, the grid size. Here, smaller is better, with the only concern being that smaller grid sizes increase the computational time needed for the algorithm to run. For the visualizations created in this paper, a grid size of 1 km provided sufficient detail, and the algorithm ran in a few minutes for each of the three datasets on an Intel i7 laptop. The algorithm was implemented in python and no attempt was made to optimize it.

The result of the comparison is shown in Figure 2. The two datasets are labeled "places", and "roads", respectively. The boundaries between metropolitan areas are shown for each dataset and for each value of . An additional plot overlays the boundaries across datasets for each value of to show that the metropolitan areas are stable across datasets. The places dataset had 87 thousand rows, and the roads dataset had 28 million rows.

References

[1] Zev Levin, Shimon O Krichak, and Tamir Reisin. Numerical simulation of dispersal of inert seeding material in israel using a three-dimensional mesoscale model. *Journal of Applied Meteorology*, 36(5):474–484, 1997.

[2] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[3] Steven C Bourassa, Eva Cantoni, and Martin Hoesli. Spatial dependence, housing submarkets, and house price prediction. *The Journal of Real Estate Finance and Economics*, 35(2):143–160, 2007.

[4] Tony Lindeberg. *Scale-space theory in computer vision*, volume 256. Springer Science & Business Media, 2013.

[5] Tony Lindeberg. *Discrete scale-space theory and the scale-space primal sketch*. PhD thesis, KTH Royal Institute of Technology, 1991.

[6] UK Office for National Statistics. Index of place names. `www.ons.gov.uk/methodology/geography/geographicalproducts/otherproducts/indexofplacenamesipn`. Accessed: July 2019.

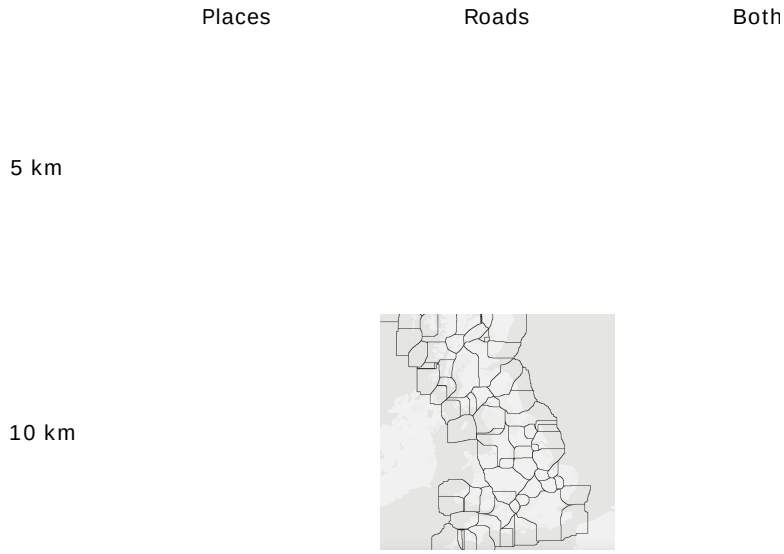

Figure 2: Comparison of the heat map segmentation algorithm across different datasets which correlate to population density and across smoothing parameters. The boundaries between areas are shown for each of two datasets discussed in the text, and for each of = 5; 10 km. The plots labeled "both" represent an overlay of the plots across datasets for each value of , and show the stability of areas with respect to the dataset. In both cases the areas roughly correspond to metropolitan areas

[7] John Parr Snyder. *Map projections–A working manual*, volume 1395. US Government Printing Office, 1987.

[8] Mark Nixon and Alberto S Aguado. *Feature extraction and image processing for computer vision*. Academic Press, 2012.

[9] KI Seon. Smoothing of an all-sky survey map with a fisher-von mises function. *J. Korean Phys. Soc.*, 48(astro-ph/0703168):L331, 2007.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[11] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[12] Open Street Maps. `download.geofabrik.de/europe/great-britain.html`. Accessed: July 2019.