An Efficient Layout Method for a Large Collection of Geographic Data Entries [*]

Sarana Nutanong Marco D. Adelfio Hanan Samet

Center for Automation Research, Institute for Advanced Computer Studies,
Department of Computer Science, University of Maryland
College Park, MD 20742, USA
{nutanong, marco, hjs}@cs.umd.edu

ABSTRACT

Many spatial applications require the ability to display locations of geographic data entries on an online map. For example, an online photo-sharing service may wish to display photos (as thumbnails) according to where they were taken. Since displaying geographic data entries as thumbnails or icons on a map requires some amount of space, displayed entries can overlap each other. As a result, we may wish to discard less popular or older entries (based on a given measure of importance) so that these more popular or newer entries become more distinct. A straightforward solution is to apply a spatial database extension such as PostGIS (i) to retrieve entries within a given display window; (ii) to discard entries in proximity of a more important one. In this paper, we demonstrate our method for efficiently selecting distinct entries from a large geographical point set. Specifically, our demonstration software presents a voting system built upon an ensemble of interrelated indexes, which is the main novelty of our query processing method. This allows us to efficiently determine the degree of distinctiveness of all entries within a query window using simple index traversal operations rather than expensive spatial operations. The effectiveness of our method in comparison to a traditional spatial query is shown by our experimental results using a real dataset of over 9 million locations. These experimental results show that our proposed method is capable of consistently producing subsecond response times, while the spatial query-based method takes more than 10 seconds on average in a low spatial selectivity setting.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*; H.2.4 [**Database Management**]: Systems—*Query Processing*

General Terms

Algorithms, Design

Keywords

Spatial databases, Query processing.

[*]This work was supported in part by the National Science Foundation under Grants IIS-10-18475 and IIS-12-19023 and by Google Research.

1. INTRODUCTION

In this paper we demonstrate our solution [8] for displaying locations of data entries on an online map where a user manipulates the display window and the zoom level to specify an area of interest. This task arises in many applications built by us that use a map query interface to access spatial and nonspatial data (e.g., NewsStand [11, 14, 19], TwitterStand [16], QUILT [13, 18], and the SAND Browser [3, 12]) for both location-based and feature-based queries [1].

This paper presents a demonstration of our efficient solution [8] for displaying the locations of data entries on an online map where a user manipulates the display window and the zoom level. One example application can be given as: *an online photo-sharing service displaying photos (as thumbnails) according to where they were taken and to allow its users to browse these images through an online mapping interface.* Since many photo entries can occupy the same area in a display window, we may choose to display only a representative subset instead. These entries can be laid out based on a scoring system denoting their importance, which can be derived, for example, from how recently each image was taken and the number times it has been viewed. We present a demonstration application which showcases the usefulness of our layout solution and exposes its internal logic.

The structure of this paper is given as follows. Section 2 presents a background discussion. Section 3 provides a query definition. Section 4 explains the proposed demonstration application. Section 5 concludes the paper.

2. BACKGROUND

Our problem can be formally described as finding an appropriate layout for a collection of spatial data entries in a display window (could result from a spatial join [5]). Specifically, given a large geographic dataset S and a geographic query window W, select a subset T of data entries from S that fall within W. Figure 1(a) shows the locations of all data entries from an example database that fall within a sample query window W. Only the outlines of image thumbnails are displayed, in order to expose the differences in density of images around the map. Figure 1(b) shows a set T of entries selected using our method on a geotagged collection of images [4, 15]. Notice that the selected entries do not result in large overlaps, and are distributed throughout the query window. In particular, we want the way in which the data entries (e.g., image thumbnails or icons) in T are selected to satisfy the following design objectives.

(i) *Minimize overlaps.* Displaying or labeling any entry from S on a map requires some amount of display space. If multiple proximate data entries are selected, their representations or labels may overlap, resulting in reduced legibility and less data clarity than is desired. In some spatial sampling applications, overlap between entries of T must be avoided completely. Assume that each data entry is represented as an image thumbnail with a size of $(\epsilon \times \epsilon)$. We can completely avoid overlaps by ensuring that no two data entries in

T have a chessboard distance smaller than a proximity threshold ϵ. In applications where a slight overlap between pairs of entries is acceptable, this proximity threshold can be relaxed.

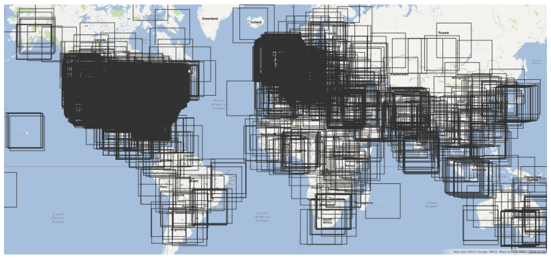
(ii) *Respect relative importance of entries.* Consider geographic datasets that include an importance measure for each entry (e.g., [6]). When two data entries are in proximity and one of them must be discarded, we should keep the more important one and discard the other entry. In other words, T should include the most important entry for different regions in W.

(iii) *Maximize spatial fullness.* The subset T should cover most of the area that is covered by all entries of S within W. That is, if one geographic region within W contains far more entries than others, then we still expect to see entries in all regions of W that contain entries in S.

(iv) *Provide panning/zooming consistency.* If a geographic window W contains a region R, and another geographic window W', obtained by panning the map, also contains R, then the spatial sample within R should be identical in both cases. Similarly, zooming consistency requires that entries selected within W should also be selected within W', when W' is obtained by zooming in from W.

(v) *Enable efficient sampling.* Even with millions of candidate entries of S within W, we still desire a fast response time to select the subset T. Note that this property restricts us from using standard map labeling algorithms that provide super-linear running time (with respect to the number of entries in W) for the selection and placement of non-overlapping map labels. This goal precludes the use of traditional map labeling solutions [2, 20] which require considering all candidate entries.

(vi) *Support filtering conditions.* Allow feature-based filters to guide the spatial sampling process, so that T does not contain any data entries that are filtered out. The filters may vary in their selectivity, in the sense that some filters could remove many data entries from being selected into T, while others may have little effect on T because they do not exclude many of the entries. This goal (in combination with the spatial fullness property) precludes the use of some precomputation-based approaches [17].


(a) Outlines of all images

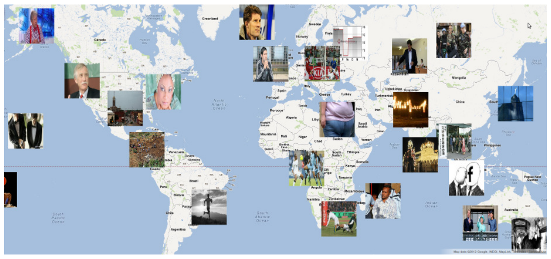

(b) Distinct images selected using the MRSD query

Figure 1: Selecting images from a set of overlapping thumbnails obtained from a geotagged collection of images.

In order to fully appreciate the challenges of designing a solution to satisfy these properties, we describe three sample approaches (based on the spatial window query) and their drawbacks. One approach is to randomly sample a subset of n entries within W, and iteratively select elements from the subset that do not overlap previously selected entries. This approach can be implemented efficiently (using a spatial index) and avoids overlaps, but does not fulfill the spatial fullness or panning/zooming consistency properties, nor does it respect entry relevance. A second approach is to select the most important n entries within W, and then to iteratively select elements from this subset that do not overlap with previously selected entries. This can be done efficiently and avoids overlaps, but again does not fulfill the spatial fullness or panning/zooming consistency properties. A third approach is to start with all entries within W, and then iteratively select entries that do not overlap with more important ones in W. This is optimal for minimizing overlaps, achieves spatial fullness, and respects relative importance of entries. However, it can be very inefficient to consider all elements in W (when W is large or S is dense within W) and panning/zooming consistency will be violated in border areas.

We model this layout problem as a problem of selecting "distinct" data entries based on a measure of importance where a data point is considered distinct if there are no other nearby data entries with a greater importance. Our solution is motivated by the `SELECT DISTINCT` query used in relational database management systems (RDBMS) which provides a way of selecting representative data entries by (i) grouping identical values from a given column; (ii) selecting one representative/distinct data entry from each group based on a set of ordering criteria. [1] This query can be efficiently processed using an appropriate index. Since grouping works for only discretized attributes, one way to adapt the `SELECT DISTINCT` query to our layout problem is to partition the data space into a grid where the grid cell size is given by a proximity threshold ϵ. We can then group data entries according to the grid cells in which they reside. However, when two data entries are separated by a grid boundary, this method considers them as being in two different groups regardless of how close they are. If these two data entries are represented as image thumbnails with a geographic size of $\epsilon \times \epsilon$, then they can overlap significantly. It is also important to point out that a user may change the zoom level while the pixel size of thumbnails remain unchanged, which effectively changes the proximity threshold relative to the map area. Hence, our solution needs to be able to handle queries when the proximity threshold ϵ changes according to the zoom level.

The demonstration application discussed in this paper illustrates the usefulness of our recently proposed query which we term the *Multiresolution Select-Distinct Query (MRSD)* [8] and an associated query processing method, which satisfy the stated design objectives. The basis of our method is a quadtree-like decomposition of space where all objects are treated on the display screen as having the same size while internally they are treated as points identified by their geographic location (somewhat analogous to the relationship between a medial axis transform and a skeleton [9]). In our experimental studies (Figure 2), we compare MRSD with a window query-based (WQB) method, which embodies the third approach described earlier, since it produces results comparable to MRSD (unlike the other two approaches). Our experiments [8] show that MRSD significantly outperforms the window query-based (WQB) method in a low spatial selectivity setting (a large query area).

3. QUERY DEFINITION

We can view the layout problem as one of selecting a set of spatial representatives from a collection of geographic data entries based on a given measure of importance. In other words, we want to select spatially distinct entries $\boldsymbol{p}$ such that $\boldsymbol{p}$ is not within a proximity threshold of more important entries. We formulate the MRSD query as providing the extent of distinctiveness of each data entry

[1]The syntax for selecting a single representative from each row group identified by a `SELECT DISTINCT` query varies between SQL implementations. In this paper, we use PostgreSQL's `SELECT DISTINCT ON (attribute)` syntax in order to emphasize that we are selecting "distinctive" data entries, however, similar results can also be obtained using standard SQL:2008 syntax.

Table 1: Data entries (from Figure 3) where each point is associated with an importance score (imp) and Morton codes from 9 different translations. The resultant distinctiveness score (ds) for each data point is given in the right most column.

Object Identifier (id)	Importance Score (imp)	Morton Codes in Different Translations									Distinctive-ness Score (ds)
		$(0,0)$ (t0)	$(\frac{w}{3},0)$ (t1)	$(\frac{2\cdot w}{3},0)$ (t2)	$(0,\frac{w}{3})$ (t3)	$(\frac{w}{3},\frac{w}{3})$ (t4)	$(\frac{2\cdot w}{3},\frac{w}{3})$ (t5)	$(\frac{0,2\cdot w}{3})$ (t6)	$(\frac{w}{3},\frac{2\cdot w}{3})$ (t7)	$(\frac{2\cdot w}{3},\frac{2\cdot w}{3})$ (t8)	
$\boldsymbol{a}$	0.95	1000	1000	1001	1000	1000	1001	1000	1000	1001	9
$\boldsymbol{b}$	0.52	1001	1001	1100	1001	1001	1100	1001	1001	1100	3
$\boldsymbol{c}$	0.47	0010	0010	0011	1000	1000	1001	1000	1000	1001	3
$\boldsymbol{d}$	0.80	0011	0011	0110	1001	1001	1100	1001	1001	1100	9
$\boldsymbol{e}$	0.65	0011	0011	0110	0011	0011	0110	1001	1001	1100	1
$\boldsymbol{f}$	0.45	0011	0011	0110	0011	0011	0110	1011	0011	0110	2
$\boldsymbol{g}$	0.66	0011	0110	0110	0011	0110	0110	0011	0110	0110	7
$\boldsymbol{h}$	0.14	0000	0001	0001	0000	0001	0001	0010	0011	0011	8
$\boldsymbol{i}$	0.90	0001	0100	0100	0001	0100	0100	0011	0110	0110	9

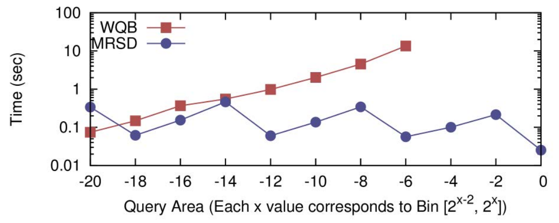


Figure 2: Total response time for selecting distinct entries from 9.96 million POIs using WQB and MRSD. Note that WQB failed to produce results for the last three bins due to excessive memory consumption.

(in the query window) according to its relative importance to the surrounding entries. Figure 3 illustrates our concept of distinctiveness using 4 data entries $\{\boldsymbol{w},\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\}$ with importance scores of $\{45, 95, 85, 10\}$, respectively. In this example, entries $\boldsymbol{x}$ and $\boldsymbol{z}$ are considered distinct since $\boldsymbol{x}$ has a higher score than $\boldsymbol{w}$ and y, and $\boldsymbol{z}$ is the only data entry in its proximity (although its importance score is much lower than $\boldsymbol{w}$, $\boldsymbol{x}$ and $\boldsymbol{y}$). It is also important to note that $\boldsymbol{y}$ is considered "less distinct" than $\boldsymbol{w}$ due to its proximity to $\boldsymbol{x}$, which is a more important entry.

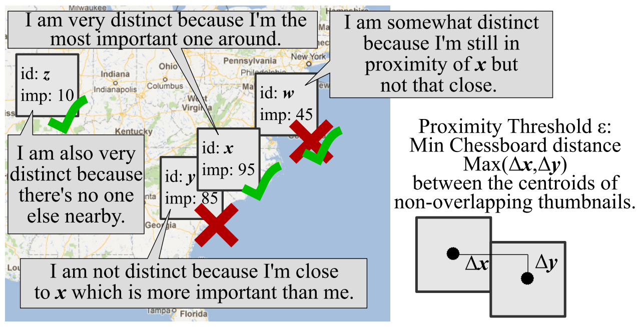

I am very distinct because I'm the most important one around.
I am somewhat distinct because I'm still in proximity of $\boldsymbol{x}$ but not that close.
id: z imp: 10
id: w imp: 45
id: x imp: 95
id: y imp: 85
I am also very distinct because there's no one else nearby.
I am not distinct because I'm close to $\boldsymbol{x}$ which is more important than me.
Proximity Threshold ε: Min Chessboard distance Max(Δx,Δy) between the centroids of non-overlapping thumbnails.
Δx
Δy

Figure 3: Distinctiveness as relative importance

The MRSD query accepts a proximity threshold ϵ (e.g., the width of each image thumbnail) and a query window (x_1, y_1, x_2, y_2) and returns a result set containing data entries residing in the query window with their respective distinctiveness scores ranging from 1 to 9. The maximum score of 9 guarantees that the corresponding entry E does not have any other object with a greater importance score within a chessboard distance of ϵ. In other words, if each entry is represented as an $\epsilon \times \epsilon$ thumbnail, then E cannot overlap with another entry with a greater importance score. If an entry $\boldsymbol{p}$ has the minimum the minimum score of 1, then $\boldsymbol{p}$ is severely overlapped by other entries with a greater importance score.

As shown in Figure 3, the value of ϵ depends on the size of each thumbnail on the map, which is in turn determined by the pixel size of each thumbnail and the zoom level. As the zoom level increases, the size of each thumbnail, where the pixel size is fixed, relative to the total map area decreases exponentially. Most online mapping applications use a constant magnification factor of 2 for successive zoom levels. As a result, in our demonstration application, the proximity threshold ϵ is reduced by 50% for every zoom level increment.

The crux of our method is a voting system built upon an ensemble of interrelated indexes, which allows us to efficiently determine the degree of distinctiveness of all entries within a query window. Specifically, our indexes are derived from the Morton code representations [10] of data entries under multiple translations. For each data point, we apply x-axis translations of 3 types and y-axis translations of 3 types providing a total of 9 translation combinations. The data entries of each translation combination are indexed using the Morton order.

We treat the distinctiveness score of a data entry as a vote count, where each vote indicates whether the data entry is considered distinct in a particular translation. Table 1 shows the results of all translations of 9 data entries (labeled $\boldsymbol{a}$-$\boldsymbol{i}$) in terms of the Morton code representations of the blocks containing them. The actual results of two translations of these entries are shown in Figure 3.

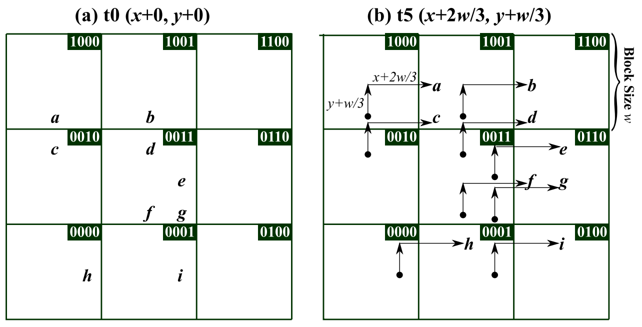
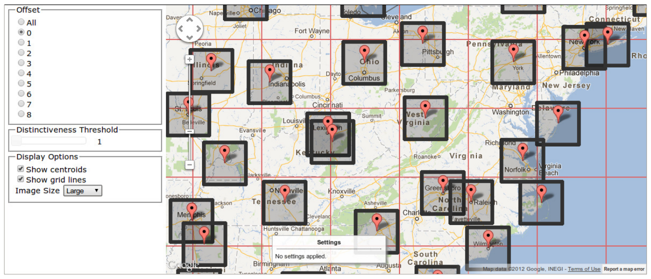


Figure 4: Data entries $\{\boldsymbol{a}, ..., \boldsymbol{i}\}$ under translations t0 and t5 (the other translations can be computed in a similar manner).

The main MRSD query consists of 9 subqueries—one per translation—that each perform an independent `SELECT DISTINCT` operation on one of the translations (ordered by the data entry's importance score (IMP)). The result set consists of distinctiveness scores, which represent the number of translations under which that data entry is the most important (of all entries with the identical Morton code). Assuming that the columns `id`, `imp`, and `t0` to `t8` from Table 1 are stored in a relational database table called `table_1`, a simple SQL statement for evaluating a MRSD query is given as follows (where the result is given as the column `ds` in Table 1). [2]

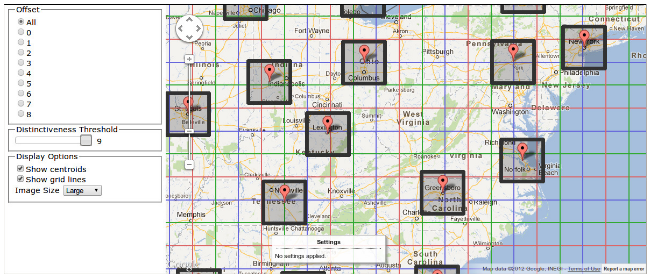
[2]Please refer to our earlier work [8] for actual SQL statements used in our implementation which support queries in multiple resolutions, subregion selection, and efficient indexing and querying.


Offset: All [✓ 0] 1 2 3 4 5 6 7 8

Distinctiveness Threshold: 1

Display Options: [✓ Show centroids] [✓ Show grid lines] Image Size: Large

(a) MRSD query with only one translation.


Offset: [✓ All] 0 1 2 3 4 5 6 7 8

Distinctiveness Threshold: 9

Display Options: [✓ Show centroids] [✓ Show grid lines] Image Size: Large

(b) MRSD query with an ensemble of 9 translations.

Figure 5: MRSD query examples.

```
SELECT id, COUNT(*) AS ds FROM (
   SELECT DISTINCT ON (t0) id FROM table_1
     ORDER BY t0 DESC, imp DESC
   UNION ALL
   ...
   UNION ALL
   SELECT DISTINCT ON (t8) id FROM table_1
     ORDER BY t8 DESC, imp DESC
) temp GROUP BY id;
```

Again, the output of this statement is a list of pairs (id, ds), i.e., the first and last columns in Table 1. Only the data entries that appear at least once in the result are returned by the query. The distinctiveness score ds of an entry x indicates the number of times x appears in the SELECT DISTINCT results from the subqueries. Hence the maximum value of ds is 9. Our demonstration application discussed in the next section uses this distinctiveness scoring system to determine whether a data entry should be displayed as well as the size in which it is displayed.

4. DEMONSTRATION APPLICATION

We have implemented a demonstration application which can be used to select distinct entries from a database of geographical data entries ranked according to their importance scores. In addition to providing a layout for geographic data entries, our demonstration also exposes the internal logic of our MRSD query processing method and the indexes.

Users of our system manipulate the mapping interface to specify an area of interest and a zoom level to see displayed results updated accordingly. In addition to the standard mapping interface, a user can run an MRSD query using one of the 9 translations or using an ensemble of all 9 translations. Note that instead of shifting the locations of the data entries, we shift the boundaries so that each data entry is always at its correct location on the map.

Figure 5(a) shows a query example which use translation t0. The red lines in the figure mark the boundaries corresponding to the translation. We can see that this query produces thumbnails that overlap each other.

When using all 9 translations, a user is also able to change the distinctiveness thresholds to specify the minimum distinctiveness score of entries that are displayed. Figure 5(b) shows results from an MRSD query which uses all 9 translations, as a result the maximum distinctiveness score in this case is 9. In this particular example, we illustrate a case where we choose to display only entries with a distinctiveness score of 9. We can see that there is no overlap between thumbnails.

5. CONCLUDING REMARKS

We have described our demonstration application for selecting spatially distinct data entries from a large geographic point set [8]. Our demonstration application illustrates the main novelty of our method which is an ensemble of interrelated indexes, and a query algorithm which performs separate SELECT DISTINCT subqueries on these indexes and combines the results of these subqueries to determine the degree of spatial distinctiveness of each entry in a query window.

6. REFERENCES

[1] W. G. Aref and H. Samet. Efficient processing of window queries in the pyramid data structure. In *PODS'90*, pp. 265–272, Nashville, TN, Apr. 1990.

[2] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. E. Moret, and B. Zhu. Map labeling and its generalizations. In *SODA'98*, pp. 148–157, New Orleans, LA, Jan. 1998.

[3] C. Esperança and H. Samet. Experience with SAND/Tcl: a scripting tool for spatial databases. *JVLC*, 13(2):229–255, Apr. 2002.

[4] B. C. Fruin, H. Samet, and J. Sankaranarayanan. Tweetphoto: photos from news tweets. In *GIS'12*, pp. 582–585, Redondo Beach, CA, Nov. 2012.

[5] E. Jacox and H. Samet. Spatial join techniques. *TODS*, 32(1):7, Mar. 2007.

[6] M. Van Kreveld, I. Reinbacher, A. Arampatzis, and R Van Zwol. Multi-dimensional scattered ranking methods for geographic information retrieval. *GeoInformatica*, 9(1):61–84, Mar. 2005.

[7] M. D. Lieberman, H. Samet, J. Sankaranarayanan, and J. Sperling. STEWARD: architecture of a spatio-textual search engine. In *GIS'07*, pp. 186–193, Seattle, WA, Nov. 2007.

[8] S. Nutanong, M. D. Adelfio, and H. Samet. Multiresolution select-distinct queries on large geographic point sets. In *GIS'12*, pp. 159–168, Redondo Beach, CA, Nov. 2012.

[9] H. Samet. A quadtree medial axis transform. *CACM*, 26(9):680–693, Sept. 1983.

[10] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, 2006.

[11] H. Samet, M. D. Adelfio, B. C. Fruin, M. D. Lieberman, and B. E. Teitler. Porting a web-based mapping application to a smartphone app. In *GIS'11*, pp. 525–528, Chicago, Nov. 2011.

[12] H. Samet, H. Alborzi, F. Brabec, C. Esperança, G. R. Hjaltason, F. Morgan, and E. Tanin. Use of the SAND spatial browser for digital government applications. *CACM*, 46(1):63–66, Jan. 2003.

[13] H. Samet, A. Rosenfeld, C. A. Shaffer, and R. E. Webber. A geographic information system using quadtrees. *Pattern Recognition*, 17(6):647–656, Nov/Dec 1984.

[14] H. Samet, B. E. Teitler, M. D. Adelfio, and M. D. Lieberman. Adapting a map query interface for a gesturing touch screen interface. In *WWW'11 (Companion Volume)*, pp. 257–260, Hyderabad, India, Mar. 2011.

[15] J. Sankaranarayanan and H. Samet. Images in news. In *ICPR*, pp. 3240–3243, Istanbul, Turkey, Aug. 2010.

[16] J. Sankaranarayanan, H. Samet, B. Teitler, M. D. Lieberman, and J. Sperling. TwitterStand: News in tweets. In *GIS'09*, pp. 42–51, Seattle, WA, Nov. 2009.

[17] A. D. Sarma, H. Lee, H. Gonzalez, J. Madhavan, and A. Y. Halevy. Efficient spatial sampling of large geographical tables. In *SIGMOD*, pp. 193–204, Scottsdale, AZ, Jun. 2012.

[18] C. A. Shaffer, H. Samet, and R. C. Nelson. QUILT: a geographic information system based on quadtrees. *IJGIS*, 4(2):103–131, Apr.–June 1990.

[19] B. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A new view on news. In *GIS'08*, pp. 144–153, Irvine, CA, Nov. 2008.

[20] F. Wagner and A. Wolff. An efficient and effective approximation algorithm for the map labeling problem. In *ESA'00*, vol. 979 of Springer-Verlag Lecture Notes in Computer Science, pp. 420–433, Corfu, Greece, Sep. 2000.