# Visualization of Geo-spatial Point Sets via Global Shape Transformation and Local Pixel Placement

Christian Panse, *Member, IEEE* Mike Sips, *Member, IEEE* Daniel A. Keim, *Member, IEEE*
Stephen C. North, *Senior Member, IEEE*

**Abstract**— In many applications, data is collected and indexed by geo-spatial location. Discovering interesting patterns through visualization is an important way of gaining insight about such data. A previously proposed approach is to apply local placement functions such as PixelMaps that transform the input data set into a solution set that preserves certain constraints while making interesting patterns more obvious and avoid data loss from overplotting. In experience, this family of spatial transformations can reveal fine structures in large point sets, but it is sometimes difficult to relate those structures to basic geographic features such as cities and regional boundaries. Recent information visualization research has addressed other types of transformation functions that make spatially-transformed maps with recognizable shapes. These types of spatial-transformation are called global shape functions. In particular, cartogram-based map distortion has been studied. On the other hand, cartogram-based distortion does not handle point sets readily. In this study, we present a framework that allows the user to specify a global shape function and a local placement function. We combine cartogram-based layout (global shape) with PixelMaps (local placement), obtaining some of the benefits of each toward improved exploration of dense geo-spatial data sets.

**Index Terms**—Geo-spatial Data, Shape Transformation, Cartogram, Pixel Visualization

## 1 Introduction

Many existing and emergent applications collect and reference data by geo-spatial location. Most electronic transactions of daily life, such as purchasing goods by credit card or making phone calls, are recorded for subsequent data analysis. Such event records usually include geographic locations and other attributes. For example, records of credit card transactions specify a purchaser (with an associated name and address), the point of sale, total amount, and possibly items and prices. Telephone call records also include locations of communication endpoints, billing accounts, and sometimes cell phone zones and other geo-coordinates. Census tables are another familiar example that incorporates geographic and statistical attributes.

Large data sets, containing millions of records or more, are nearly impossible for people to understand quickly by inspecting the raw data. Visualization is essential to surveying and exploring them. Although geographic and statistical visualization have been studied for many decades, the scale of the data we have now presents new challenges. Displaying large point sets on conventional maps is problematic. Overplotting obscures data points in densely populated areas, while sparsely populated areas waste space and convey scant detailed information. Small clusters are difficult to find – they are not noticeable enough, and are sometimes even occluded by large clusters. Figure 2 illustrates these two problems on a traditional map.

A common approach to visualization is to apply local placement functions that transform the input data set into a solution set that preserves certain constraints while making certain patterns more obvious. Previously we proposed a pixel-oriented method called PixelMaps for visualizing large spatial datasets [8]. This approach combines kernel density-based clustering with a point relocation technique that preserves local clusters and avoids overplotting. It assigns each input data point to a unique 2–D screen pixel, trading off absolute and relative position against clustering to achieve pixel coherence. In practice, we noticed that PixelMaps can reveal fine structures, but it may be difficult to relate them to geographic features such as locations of cities or regional boundaries.

Recent research has also addressed layout functions that optimize visualization constraints to preserve recognizable features in visualizations. In particular, cartograms are map transformations that preserve shapes and relationships between map regions [3]. There exist several cartogram algorithms, see [11] for an overview see. Classic cartograms preserve an input map's topology, while scaling polygonal elements according to an external parameter vector [6]. Cartograms seem more easily interpreted than PixelMaps, though they do not address overlap problems or pixel coherence.

In this study, we demonstrate how PixelMaps and cartogram layout may be composed to meet some of the challenges of large-scale geovisualization.

- *Christian Panse is now with the Functional Genomics Center - Uni|ETH Zurich, Switzerland, E-mail: cp@fgcz.ethz.ch.*
- *Mike Sips is now with the Max Planck Center for Visual Computing and Communication, Stanford University, USA, E-mail: ms@pixelmap.org*
- *Daniel A. Keim is with University of Konstanz, Germany, E-mail: Daniel.Keim@uni-konstanz.de.*
- *Stephen C. North is with AT&T Research Labs, NJ, USA, E-mail: north@research.att.com.*

*Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.*

## 2 Problem Definition

We consider the display of point sets on maps. A map is represented by a polygonal mesh. The points of the input set are assumed to have one or more associated statistical attributes. Informally, our goal is to show clusters and other relationships between points, determined by both locations and statistical values.

By considering just one statistical attribute at a time, we can interpret geo-spatial data sets as points in 3–D: the two geo-spatial dimensions and a third statistical dimension. We note that real-world data set distributions are often highly nonuniform, and data points form readily-identifiable 3–D point clouds. For example, figure 1 shows a household income distribution data set in the 3–D space spanned by longitude, latitude, and median household income.

In this paper, we assume that every data point is important and does not replace, summarize, aggregate or overplot individual data points. For example it is of major importance in the detection of fraud where a data point represents a customer or in the analysis of networks where a data points is a server/router. In both cases, conclusions may be invalid if we overplot individual data points. The result could be that we suspect our best customers of fraud and may lose them, or we identify the wrong server as out of order and then shut down the whole network.

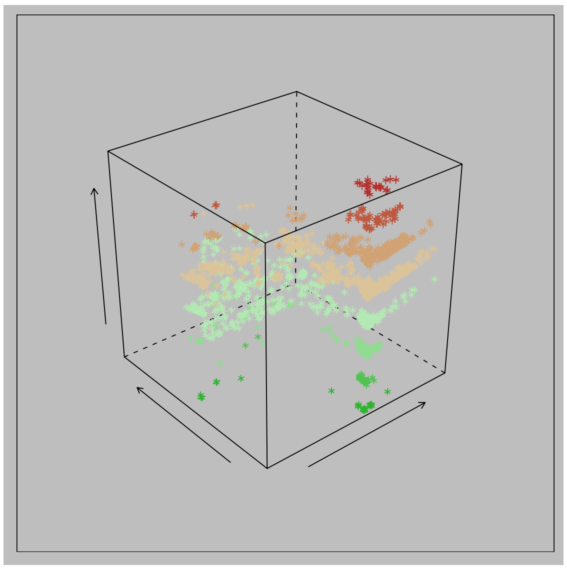


Fig. 1. Input $XY = \{xy_0, \ldots, xy_{N-1}\}$ often exhibits 3-D point clouds even in small real-world examples. This example shows 3000 data points (1%) from the U.S. Year 2000 Median Household Income [12])

## 2.1 Synthesis of Global Shape and Local Placement

These issues identified in the previous section lead us to propose three constraints for point set visualization:

- Overlap elimination
- Preservation of point positions
- Clustering of similar values
- Preservation of map shapes and their relationships

(see section 3.1.1 and section 3.2.1 for more details).
In practice, simple heuristic placement of data points on land-covering maps that preserves positions and emphasizes clustering typically also exhibits artifacts, such as spiral effects, especially in dense areas (figure 2).
Arbitrarily-distorted maps tend to hide geo-spatial relationships between clusters and other structures. Such visualizations show too much elementary information within an unusual geometry, and can potentially encourage wrong conclusions. The practical benefit of such visualizations seems low [9].
Global properties can contribute toward accuracy and readability or interpretability of plots of large point sets. Our goal is to (a) represent dense areas so as to preserve some of the important structure of the original geographical space (global shape) and (b) allocate all data points to unique display pixels, even in dense regions (local placement). In other words, we seek to show as many data points as possible, close to their original positions, while determining a good trade-off between shape distortion and the degree of overlap.
We approach this by decomposing the problem into separate map transformation and point relocation problems. The solution assigns each point to a unique pixel location, with the objective of showing all the points, but making it possible to relate features to the original map.

## 2.2 Input

The input is a point set, a global map, and the pixilated display space:

- Set of input points $XY = \{xy_0, \ldots, xy_{N-1}\}$
  $xy_i = (xy_i^x, xy_i^y)$ is the original position of each point and $S_1(xy_i), \ldots, S_k(xy_i)$ are its associated statistical parameters. It is likely that we have many data points $i$ and $j$ whose positions are very close or even identical, *i.e.* $xy_i \approx xy_j$ if $XY$ is large (see figure 1).
- Global Map $M = \{p_1, \ldots, p_k\}$
  defined by a set of connected simple polygons $p_1, \ldots, p_k$ (a polygonal mesh)
- Display space $DS \subset \mathbf{Z}^2$
  $DS = \{0, \ldots, x_{max} - 1\} \times \{0, \ldots, y_{max} - 1\}$, where $x_{max}$ and $y_{max}$ are the extents of the pixel display window.

## 2.3 Output

The output contains the local placement of the data set $XY'$, the global shape $M'$ and the synthesis of the global shape and local placement.

- Set of points $XY' = \{xy'_0, \ldots, xy'_{N-1}\}$
  $xy_i = (xy_i^x, xy_i^y)$ is the new position of each point and $S_1(xy'_i), \ldots, S_k(xy'_i)$ are the associated statistical parameters. All data points are visible, which means each is assigned to a unique pixel position.
- $M' = \{p'_1, \ldots, p'_k\}$ defined by a set of connected simple polygons $p'_1, \ldots, p'_k$ (a polygonal mesh)
- Synthesis of the global shape transformation $DS(XY', M') = \rho(\pi(XY, M))$
  $\pi = gs(XY, M)$ defines the global shape and $\rho = lp(\pi(XY, M))$ provides the local pixel placement.

## 3 Global Shape Transformation $\pi$ and Local Placement $\rho$

The selection of a global shape function $gs$ and a local placement function $lp$ defines a generic transformation for geo-spatial point set visualization.

## 3.1 Local Placement Function $lp$

Local Placement functions play an important role in visualizing large data sets. In general, a local placement function $f$ transforms an input data set $A$ into a solution set $B$, preserving important characteristics such as position, clustering and, in the case of multidimensional data, shapes of sub-windows. A reference explains further details [5].

### 3.1.1 Local Point Placement Constraints

- Overlap Elimination Constraint The highest priority constraint is that all data points be visible, which means assigning each to a unique output pixel position. Formally, this can be expressed

$$i \neq j \;\Rightarrow\; b_i \neq b_j \quad \forall i, j \in \{1, \ldots, N-1\}$$

- Position Preservation Constraint Another constraint is that the output positions should be close to the original ones. That distance can be measured by taking the absolute displacement of points from their original positions, or by finding their displacement relative to each other.
  - absolute position preservation

$$\min \sum_{i=0}^{N-1} d(a_i, b_i)$$

  - relative position preservation

$$\min \sum_{i=0}^{N-1} \sum_{j=0, i \neq j}^{N-1} (d(b_i, b_j) - d(a_i, a_j))^2$$

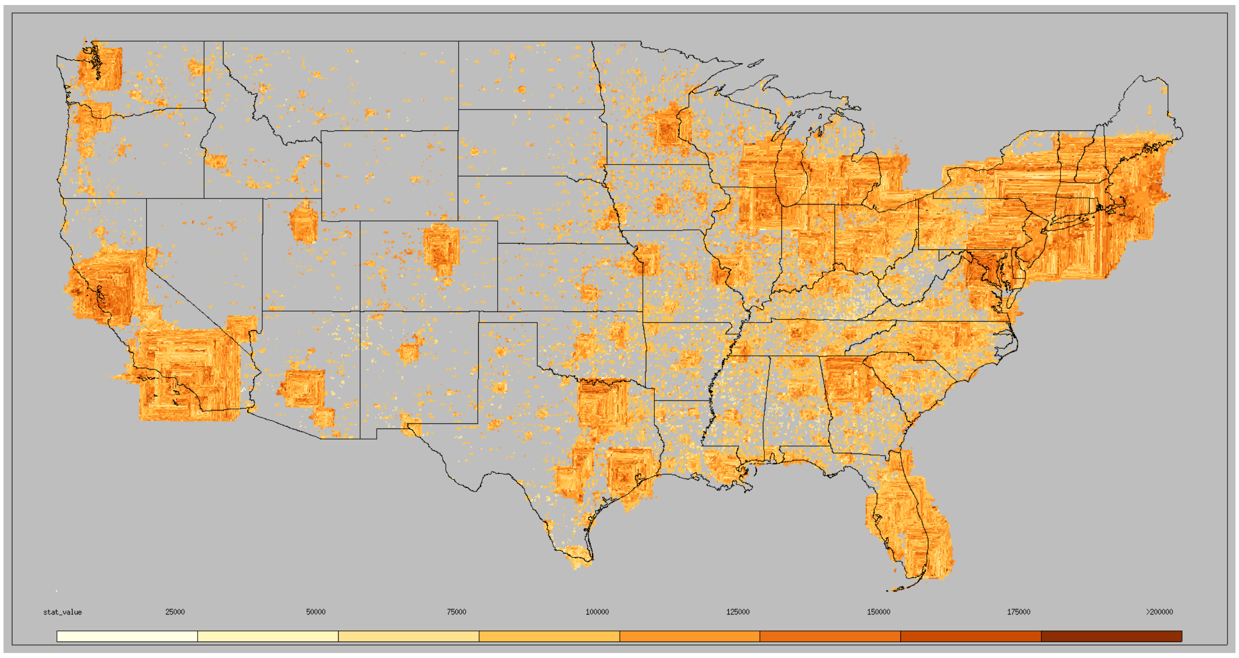


Fig. 2. **Traditional map with simple repositioning** – This dotplot map shows artifacts, such as spiral effects, especially in dense areas such as Los Angeles County, Cook County and Manhattan. The artifacts are caused by using a simple find next free pixel heuristics on conventional maps. The data set is U.S. Year 2000 Median Household Income [12]. Visualization transformation $\pi(XY,M) = gs(XY,M) = identity(XY,M)$ and $\rho(\pi(XY,M)) = lp(\pi(XY,M)) = find\_next\_free\_pixel(\pi(XY,M)))$

The choice between relative and absolute position preservation may depend on the application. The distance function $d$ can be defined by an $L^m$-norm ($m = 1$ or 2)

$$d(b_i,b_j) = \sqrt[m]{(b_i^x - b_j^x)^m + (b_i^y - b_j^y)^m}$$

.

- Clustering Constraint The third constraint involves clustering on one of the statistical attributes $S_i, i \in \{0,\ldots,k\}$. The idea is to reposition data points so that those with high similarity in $S_i$ are near each other. In other words, points in the neighborhood of any given data point should have similar values, yielding pixel coherence. Formally, we need to define the neighborhood $\mathcal{NH}$ of a data point $a_i$, and a distance function $d_S$ on the statistical attribute $S$.

$$\min \sum_{i=0}^{N-1} \sum_{b_j \in \mathcal{NH}(b_i)} d_S(S(b_i),S(b_j))$$

This neighborhood function sums up the differences in $S$ between points and their neighbors, and may be defined

$$\mathcal{NH}(b_i) = \{b_j | d(b_i,b_j) < \varepsilon\}.$$

Because $S_i$ may have a very non-uniform distribution, it may also be appropriate to apply non-linear scaling to $S$ before computing distances $d_S$.

### 3.1.2 Local Placement and PixelMaps

The local placement $\rho$ can be seen as the yield of a local placement function $lp$ that takes an input point set $XY$ to unique positions (while fulfilling the local placement constraints defined above). Formally, we can express $lp$ as follows:

$$lp : Point \rightarrow Point \text{ with } \rho(XY) = lp(XY,\pi(XY,M))$$

In this study, we focus on PixelMaps because they are intended for large data sets. PixelMaps assign input points to unique display pixels, mapping them to highlight clusters and avoid data loss from overplotting.

## 3.2 Global Shape Function $gs$

Recent research has contributed new global shape methods for information visualization on maps. These methods optimize properties such as stability, preservation of ordering, and aspect ratio of shapes (see work on TreeMaps [4] for further readings). We define a global shape function $\pi$ as follows. The input to the shape function $\pi$ is a vector of $k$ non-negative real numbers $(l_1,\ldots,l_k)$, and the output is a corresponding display space partitions $(p_1,\ldots,p_k)$ where $Area(p_i) = l_i$ $\forall i \in \{1,\ldots,k\}$. Good choices of $\pi$ scale well to show large data sets.

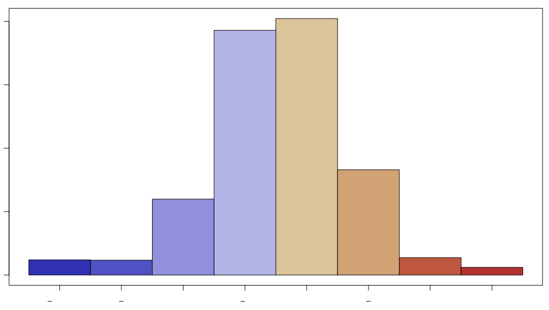
### 3.2.1 Global Shape Properties

PixelMaps avoid overplotting and the consequent loss of interesting patterns and other information. As mentioned, PixelMap transformations on typical land-covering maps often exhibit unwanted artifacts in dense areas. Potentially, pre-distortion of map regions to better fit 3–D point clouds to display space would reduce overlap without such drawbacks.
The challenge is to find a layout function that preserves recognizable shapes of the input map, while reducing pixel overlap. Note that map distortion by itself does not preclude overplotting, so local placement methods are still needed. On the other hand, if we choose an arbitrary map distortion that is sufficient to avoid all pixel overlaps, it may be difficult for humans to easily comprehend the result (compare examples in references [9]).

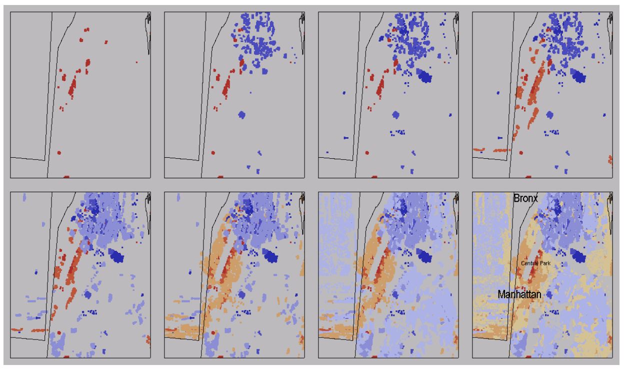
### 3.2.2 Global Shape and Cartograms

The global shape $\pi$ can be seen as the result of a global shape function $gs$ applied to an input set $XY$, taking it to a new set $XY'$ of positions on the input map $M$. Formally, we can express $gs$ as follows

$$gs : (Point,Polygon) \rightarrow Point \text{ with } \pi(XY,M) = gs(XY,M)$$



(a) Histogram of eight median household income classes.



(b) Pixel placement step, left to right starting with the smallest cluster. Color scale is the same as figure 3(a)

Fig. 3. **Local Placement** $\rho$ – PixelMap Placement Heuristic is starting from left to right with the smallest cluster. (New York Median Household Income dataset)

In this study, we focus on cartograms because they are intended to maintain recognizable global shapes in distortions of geo-spatial maps. In particular, we use the CartoDraw framework as the global shape function.

### 3.3 Generic Framework

The selection of a global shape function $gs$ and a local placement function $lp$ defines a generic transformation for geo-spatial point set visualization. If we do not take shape into account, and allow only local point placement, we get a classical PixelMap visualization. We can express PixelMap $PixelMap(XY,M)$ visualizations as follows:

$$\begin{aligned} PixelMap(XY) &= lp(gs(XY,M)) \\ gs(XY,M) &= identity(XY,M) \end{aligned}$$

A simple dot plot function $DotPlot(XY,M)$ that handles neither global shape nor local placement and can be expressed as follows:

$$\begin{aligned} DotPlot(XY,M) &= lp(gs(XY,M)) \\ gs(XY,M) = lp(XY,M) &= identity(XY,M) \end{aligned}$$

If we do not handle point set transformation, then we get classical cartogram visualizations. We can express cartogram visualizations $Carto(XY,M)$ as follows:

$$\begin{aligned} Carto(M) &= lp(gs(XY,M)) \\ XY, lp &= \emptyset \end{aligned}$$

The synthesis of both methods can be expressed as

$$\begin{aligned} \pi(XY,M) &= gs(XY,M) \\ \rho(XY,\pi(XY,M)) &= lp(XY,\pi(XY,M)) \end{aligned}$$

Obviously this framework encodes a vast range of possible geo-spatial visualization methods. In this study, $gs$ is a cartogram transformation and $lp$ is the PixelMap placement function.

## 4 Synthesis of Cartograms and PixelMaps

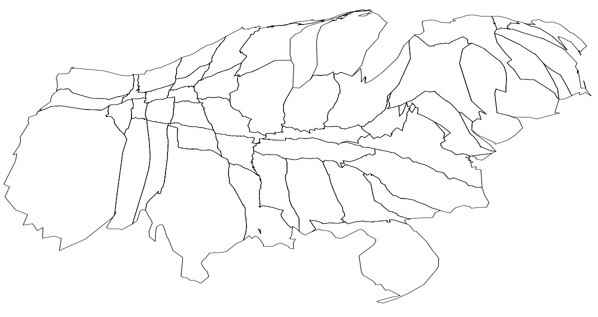
Now we can give a detailed definition of the proposed algorithm. The overall approach is to compute a global shape using CartoDraw or RecMap, then place the data points into the global shape using PixelMap.

### 4.1 CartoDraw

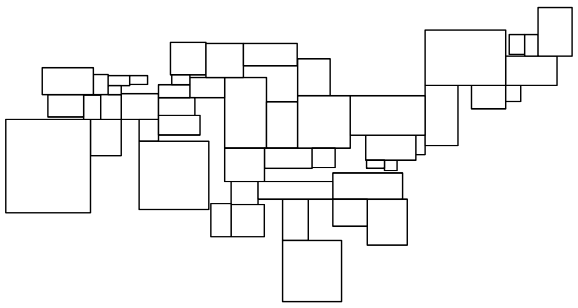
The CartoDraw heuristic [6] incrementally repositions a map's vertices along a series of scan lines. A scan line is an arbitrary line segment. A scan line defines a set of *scan sections* orthogonal to the scan line, where the map's polygons are intersected by the scan line. For each section, a target scaling factor is determined according to the *area error* factors of the corresponding polygon. In each step of the heuristic, a scan line and scan sections are determined. Vertices are then repositioned according to the scaling factors and distances to the scan line. The repositioning may be parallel or orthogonal to the scan lines. If the *shape error* introduced by applying a scan line exceeds some threshold, the repositioning of its candidate vertices is discarded.
To improve convergence toward a solution, scan lines should be applied to dense map regions. A simple approach to scan line generation is to use horizontal and vertical line segments positioned on a regular grid. Significantly better results, though, can be obtained with manual scan line placement guided by the shape of the input polygons and the local potential for improvement. Note that the incremental repositioning of vertices per scan line processing step is intentionally kept small, compared with the expected change in area. This means the same scan line may need to be applied many times to make large adjustments in a given part of the map. Figure 4(a) is an example.
As an alternative to generation of cartograms through scanlines on a regular grid, *CartoDraw* can also compute cartograms iteratively through a modified medial axes transformation [7, 10].

### 4.2 RecMap

Contiguous cartograms are made with the objective of minimizing the error in desired area while maintaining the input map's topology and preserving the shapes of its faces. Nevertheless, there exist combinations of maps and input parameter values such that it is impossible to eliminate area error [6]. Also, despite the attempt to preserve shapes, some are necessarily distorted, and the resulting irregular shapes may be difficult to find and to compare with each other.
As an alternative to contiguous cartograms, *RecMap* [2] approximates familiar land covering map regions by rectangles. The areas of its rectangles are *exactly* proportional to given statistical values. To support the understanding of information represented by a cartogram, RecMap places the rectangles close to their original positions, and seeks to preserve their adjacencies.
The RecMap construction algorithm works as follow. In the initialization step, we choose a specific polygon, called the core polygon, to be the center of the layout or cartogram. Then, in the main step, we construct a sequence of *partial layouts* or *partial cartograms*, i.e. starting with the core polygon, the remaining $n-1$ polygons are placed around



(a) CartoDraw



(b) RecMap

Fig. 4. **Global Shape** $\pi$ – The graphics display two possible cartogram methods. In these particular visualizations, the areas of each region correspond to the U.S. state census population data.

it one after the other until we have found the complete cartogram. An example of *RecMap*'s output can be seen in figure 4(b).

### 4.3 PixelMap Placement

The PixelMap algorithm runs in three phases. First, it performs an array-based clustering of the input points, partitioning them into a fixed number of bins based on a statistical attribute. The number of bins depends on the application scenario.
Second, a point placement algorithm seeks to place all cluster elements at free positions near cluster centroids, without overwriting already-occupied pixels. To solve the pixel coherence problem and make small clusters visible, all cluster members are placed close to their centroids. This step is performed one cluster at a time, starting with the smallest. Small clusters need the fewest free positions, and in practice can often be placed optimally. If data points cannot be placed without overwriting already occupied pixels, the placement algorithm searches for another free region near the centroid where most of the data points can be placed.
Finally, the placement algorithm continues with the smallest of the non-cluster bins, assigning the data points to free pixels. Figure 3 shows a sketch of the placement step. Figure 3(b) is an example from the Manhattan, New York area. The red-blue bipolar colormap encodes income classes (blue is low). Color saturation encodes the number of class members. The color usage described above can be seen in figure 3(a).

### 4.4 Synthesis Algorithm

The synthesis (algorithm 1) has the following steps.

- **Allocation and Scaling**
  - Find global shape $\pi$ using cartogram transformation to equalize density across map regions
  - Find new coordinates of data points $a_i \in A$ using a bilinear filter
- **Cluster and Pixel Placement** – Find PixelMap placement $\rho$ to position data points as clusters within the cartogram layout
- **Output** to the screen space using WALDO [9]

In the following we describe these steps in detail.

**Algorithm 1**: synthesis algorithm

```
input  : PointSet $XY$, DisplaySpace $DS$, Map $M$,
         cartogramMethode cm;
output: PointSet $XY''$
1  hashlist hl($M$.numberOfRegions(),NULL);
2  array $\lambda$($M$.numberOfRegions(), 0);
3  // Step 1: find global shape
4  // compute statistical value for each map region
5  for each region ($r \in M$) do
6      for each ($xy \in XY$) do
7          if $r.boundedregion(xy)$ then
8              $\lambda[r]$++;
9              hl[$r$].append($xy$);
10         end
11     end
12 end
13 // compute cartogram
14 $\pi(M)$=carto($M$, $\lambda$, cm);
15 // find new coords for each data point
16 $XY'$=$bf$($M$,$\pi(M)$,hl);
17 // Step 2: find local pixel placement
18 $\pi \circ f(XY')$=PixelMap($XY'$, $DS$);
19 // Step 3: output to the user
20 $XY''$ =WALDO($\pi \circ f(XY')$,$DS$);
```

### 4.5 Step 1: Allocation and Scaling

#### 4.5.1 Step 1.1: Find Global Shape $\pi(M)$

First, we approximate the overall density $\lambda$ in the two geographical dimensions $(a_i^x, a_i^y)$ by measuring the local density $\lambda(r)$ in each map region $r$. The approximated overall density $\lambda(r)$ is a second order property that identifies regional, or neighborhood patterns within the dense point distribution. The potential benefit is to use them in the identification and visualization of local patterns in the placement step. In our setting, we can efficiently approximate $\lambda(r)$ by counting the number of data points in each map region $r$. This can be done with a simple point-in-polygon test, or a more efficient data structure to search planar subdivisions.
Then, we scale the map's regions to help fit dense, non-uniformly distributed point sets to unique positions. The idea is scale each region so its area is proportional to its number of data points. Sparse regions will shrink, and dense areas will expand to enable point assignment with pixel coherence.
The use of local densities $\lambda(r)$ and *CartoDraw* [7] enables the identification of location-based point patterns. The next step describes how PixelMap uses the global shape $\pi(M)$.

#### 4.5.2 Step 1.2: Mapping to New Coordinates

Next, all data points within the original regions of $M$ are relocated to new positions in the global shape $\pi(M)$. We apply bilinear filtering, a technique previously studied to make photorealistic images with texture maps. Our bilinear filter $bf$ maps a data point within a map region $r$ to a corresponding point on $\pi(M(r))$. This is applied to each data point of each region $a_i^r \in hl[r]$. we combine the bilinear filter with a member index $hl$. More specifically, we determine the map region in which the data point is located, and then apply the bilinear filter to the original boundary $M(r)$ and the new boundary $\pi(M(r))$. The member index $hl$ can be implemented efficiently as a hash list or as records in a table.

### 4.6 Step 2.1: Cluster and Pixel Placement

Finally, after rescaling data points to new positions, PixelMap categorizes each point as either an element of a cluster or noise. It then iteratively assigns data points to display space pixels. It processes the points from densest to least dense region. Within regions, it places clusters from smallest to largest [8]. This yields the final assignment of points to display pixels.

### 4.7 Step 2.2: Label Placement

Text labels are important in readable maps [1]. We assume labels are text boxes placed with respect to a reference point. (If a reference point is not provided, there are strategies for inferring one, such as taking the centroid of the nearest cluster in the input point space.) Labels are placed on the output map by aligning them to the transformed reference point. For examples, see the figures in the following section. Although we do not deal with the problem of eliminating overlapping labels, this could be handled by applying a standard technique to the transformed labels. In the pixel plots of that paper we scaled the top most pixels of a county by a squar-root function.

### 4.8 Step 3: Presentation

We use our *Wide Area Layout Data Observer (WALDO)*, a pixel-based visual exploration system that combines several relevant interaction techniques, to show the result to the data analyst. *WALDO* allows a data analyst to adjust visualizations interactively to satisfy data exploration objectives.

## 5 Application Examples

Some examples of the synthesis of cartograms and PixelMaps using application data help to illustrate its properties. Figure 2 shows the United States Year 2000 Median Household Income database. A conventional map has many areas with a high degree of overlap. The output of the synthesis algorithm, see figure 5, shows its advantages. We can see detailed structures in Manhattan area and Los Angeles County not visible in the conventional map. Global shape preservation by cartogram preprocessing enhances visualization of these areas.

### 5.1 Census Demographics Analysis

The examples are based on U.S. Census Bureau demographics [12]. These datasets are available for multiple census levels: states, counties, and blocks. For every census block, the total number and locations of households, median income, median gross rent and price index of vacant homes are listed.

#### 5.1.1 Median Household Income on USA National Level

The first examples, figure 5 and figure 6, shows U.S. Year 2000 Median Household Income. The plot shows that New York City and Los Angeles County are areas of high population. The densest regions are allocated enough space to place all data points without occlusion and to show clusters. Some details of the distribution of median incomes can be identified. For example, one can locate features such as Central Park in Manhattan. We observe on the U.S. National Level plot that there are high income clusters on the East Side of Central Park, and in suburbs of Chicago but not its downtown neighborhood. In the San Francisco area we can identify Silicon Valley; the income in this small area is significantly greater than average.

#### 5.1.2 Housing on State Level

Figures 8 is a view of the State of New York. We can notice high gross rents ($1500-2000 USD per month) in areas of Manhattan and Queens, and low rents in the Bronx and Brooklyn. The gross rent is higher (around $1000 USD) than the rest of the state. Particularly, high income households are found on the east side of Central Park (recognizable as a rectangular void in the plot). The house price index indicates that it is expensive to buy a home in Manhattan, and it is slightly less expensive near Central Park than in the SoHo district (to its south).

The example, figure 7, shows the Median Household Income of the State of New York. We observe that there are high income clusters on the East Side of Central Park.

## 6 Conclusion & Future Work

In this study we proposed and demonstrated a method of visualizing large geo-spatial point sets. The method combines a global shape transformation function $gs$ with a local point relocation function $lp$. Each of these functions can be selected to favor a particular set of constraints and objectives.

Our study focused on the composition of cartograms (which reallocate area so as to preserve recognizable shapes) with PixelMaps that handle the assignment of individual points to unique pixels in 2–D screen space, trading off absolute and relative position preservation and clustering to achieve pixel coherence. This composition provides a capability that neither cartograms nor PixelMaps alone provide.

Initial experiments show that a synthesis of cartograms with PixelMaps offers an improvement over standalone PixelMaps in avoiding artifacts and preserving recognizable features of the input map.

The framework of global shape and local placement can be applied to classical infovis techniques. The aim is to solve some of the major problems such as overplotting in parallel coordinates.

## 7 Acknowledgments

The authors thank the anonymous referees for their comments toward the improvement of this report. This work was supported by the Max Planck Center for Visual Computing and Communication.

## References

[1] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

[2] R. Heilmann, D. A. Keim, C. Panse, and M. Sips. RecMap: Rectangular Map Approximations. In *InfoVis 2004, IEEE Symposium on Information Visualization, Austin, Texas*, pages 33–40, October 2004.

[3] D. H. House and C. J. Kocmoud. Continuous cartogram construction. In *VIS '98: Proceedings of the Conference on Visualization '98*, pages 197–204, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[4] Human-Computer Interaction Lab – University of Maryland. TreeMap home website, 2006. `http://www.cs.umd.edu/hcil/treemap/`, March 2006.

[5] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 6(1):59–78, January–March 2000.

[6] D. A. Keim, S. C. North, and C. Panse. Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 10(1):95–110, 2004.

[7] D. A. Keim, S. C. North, and C. Panse. Medial-Axis-based Cartograms. *IEEE Computer Graphics and Applications*, 25(3):60–68, May/June 2005.

[8] M. Sips, D. A. Keim, S. C. North, and C. Panse. Pixel based visual mining of geo-spatial data. *Computers & Graphics (CAG)*, 28(3):327–344, June 2004.

[9] M. Sips, C. Panse, D. A. Keim, and S. C. North. Visual data mining in large geo-spatial point sets. *IEEE Computer Graphics and Applications (CG&A), September-October 2004*, 24(5):36–44, September 2004.

[10] R. Tam and W. Heidrich. Shape simplification based on the medial axis transform. In *VIS '03: Proceedings of the Conference on Visualization '03*, pages 481–488, Washington, DC, USA, 2003. IEEE Computer Society.

[11] W. R. Tobler. Thirty five years of computer cartograms. *Annals, Assoc. Am. Geographers*, 94(1):58–73, March 2004.

[12] United States Department of Commerce. Census Bureau website, 2006. `http://www.census.gov/`, March 2006.

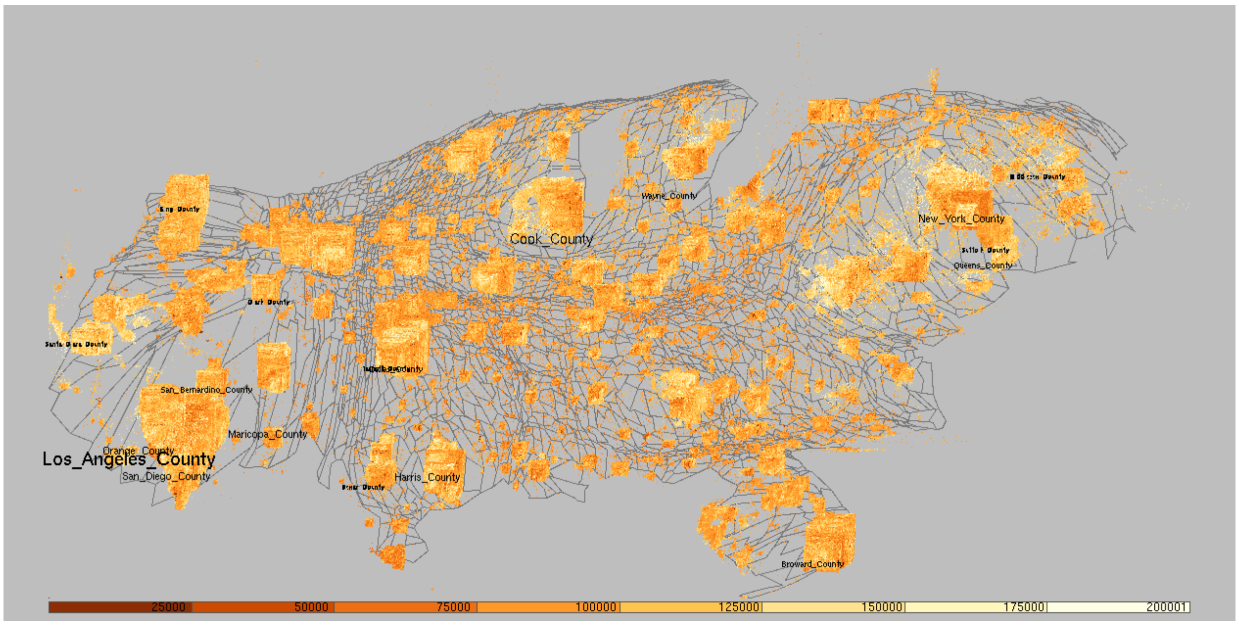


Fig. 5. Census Demographic Analysis. United States, Year 2000 Median Household Income. $\pi(XY,M) = gs(XY,M) = CartoDraw(XY,M)$

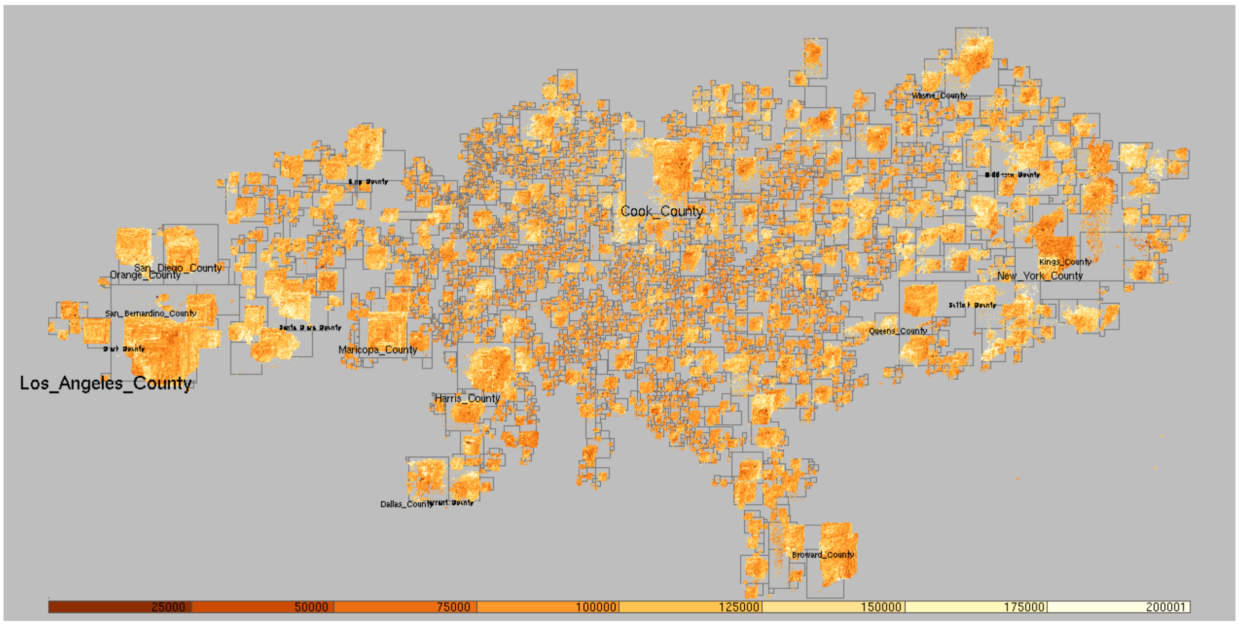


Fig. 6. *Census Demographic Analysis* – United States, Year 2000 Median Household Income. ($\pi(XY,M) = gs(XY,M) = RecMap(XY,M)$)

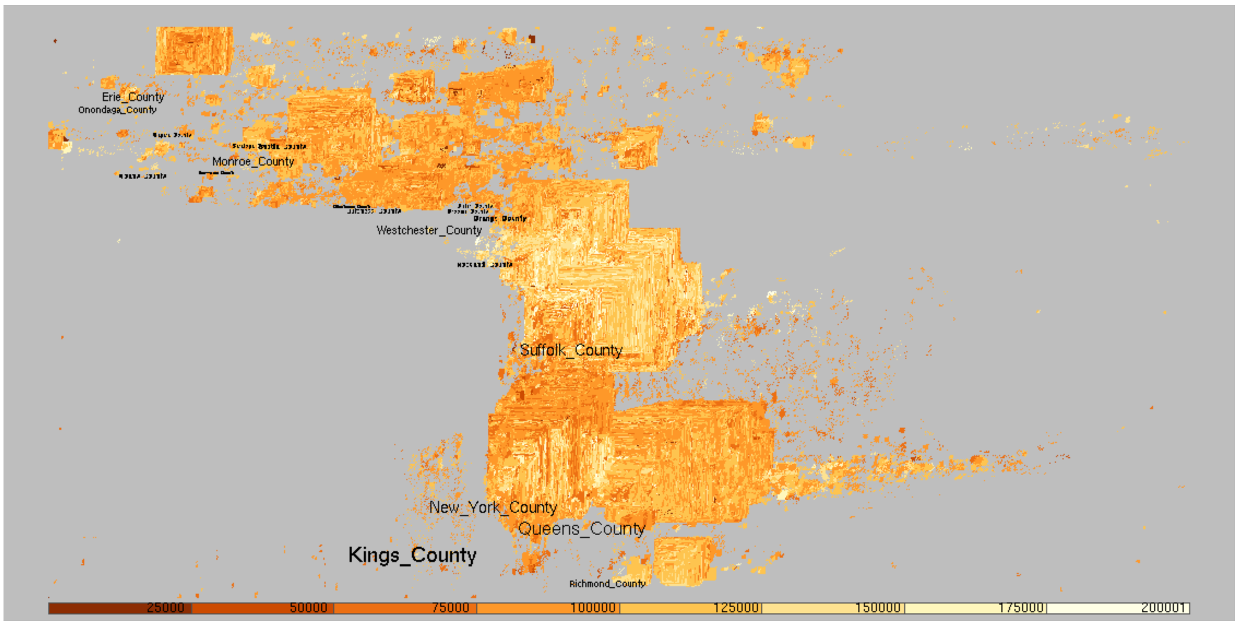


Fig. 7. Census Demographic Analysis. New York State, Year 2000 Median Household Income. $\pi(XY,M) = gs(XY,M) = RecMap(XY,M)$

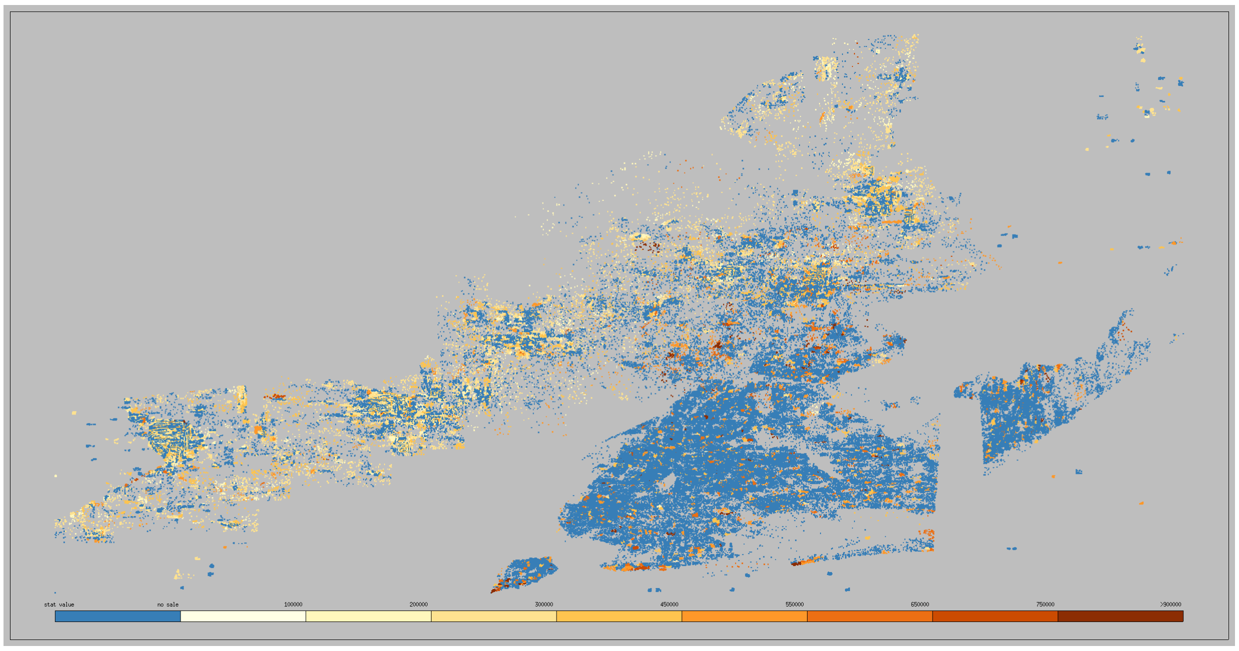


Fig. 8. Census Demographic Analysis. New York State, Year 2000 House Price Index. $\pi(XY,M) = gs(XY,M) = CartoDraw(XY,M)$ (without labeling). The labeling can be switched off by the user.