Improved Grid Map Layout by Point Set Matching

David Eppstein
Bettina Speckmann

Marc van Kreveld
Frank Staals

University of California, Irvine, Utrecht University, TU Eindhoven
Visualising Geographic Data

Given a map with $n$ regions we want to visualise some data for each region.
Visualising Geographic Data
Given a map with $n$ regions we want to visualise some data for each region. e.g. US Presidential Elections

Problem: Visual Clutter
Visualising Geographic Data

Given a map with $n$ regions we want to visualise some data for each region.

Idea: Use a Grid Map
Visualising Geographic Data

Given a map with $n$ regions we want to visualise some data for each region.

**Idea:** Use a Grid Map

- London BikeGrid: gicentre.org/bikegrid
Visualising Geographic Data

Given a map with $n$ regions we want to visualise some data for each region.

**Idea:** Use a **Grid Map**

- London BikeGrid: gicentre.org/bikegrid
- OD Maps [Slingsby, Kelly, Dykes, Wood]

based on Spatial Tree Maps [Dykes, Wood]
Assigning Cells to Regions

How do we assign the grid cells to the regions?

Tasks:
- Locate a cell
- Compare different cells
- Look for spatial patterns
Assigning Cells to Regions

How do we assign the grid cells to the regions?

Tasks:
- Locate a cell
- Compare different cells
- Look for spatial patterns

Optimisation criteria:
- Location
- Adjacency
- Relative orientation
Assigning Cells to Regions

How do we assign the grid cells to the regions?

Tasks:
- Locate a cell
- Compare different cells
- Look for spatial patterns

Optimisation criteria:
- Location
- Adjacency
- Relative orientation

NP-Hard
Assigning Cells to Regions

How do we assign the grid cells to the regions?

1-to-1 Point Set Matching Problem

 Regions $\leadsto$ set of blue points $A$.
 Grid cells $\leadsto$ set of red points $B$.

Goal: find the best matching $\phi : A \rightarrow B$
Optimising Location

Minimize the sum of the $L_1$-distances between matched points

We want to find a matching $\phi^*$ that minimises

$$D_1(\phi) = \sum_{a \in A} d(a, \phi(a))$$

where $d(a, b) = |a_x - b_x| + |a_y - b_y|$.
Optimising Location

We want to find a matching $\phi^*$, translation $t^*$, and scaling $\lambda^*$ that minimise

$$D(\phi, t, \lambda) = \sum_{a \in A} d(\lambda a + t, \phi(a)).$$

where $d(a, b) = |a_x - b_x| + |a_y - b_y|$. Minimize the sum of the $L_1$-distances between matched points under translation and scaling.
Lemma 1. For any matching \( \phi \), there is a \( t \) that \( x \)-aligns \( A \) and \( B \) and minimises \( D_T(\phi, \cdot) \).

Aligning \( A \) and \( B \) decreases \( D_T \)
Minimising $D_T$

There is an optimal matching at an $x$-alignment.

Same trick for $y$-alignment.
Minimising $D_T$

There is an optimal matching at an $x$-alignment.

Same trick for $y$-alignment.

There is an optimal matching at an $x$- and $y$-alignment.

$\implies$ There are at most $n^4$ such alignments.
Minimising $D_T$

There is an optimal matching at an $x$-alignment.

Same trick for $y$-alignment.

There is an optimal matching at an $x$- and $y$-alignment.

$\implies$ There are at most $n^4$ such alignments.

**Theorem 1.** A $\phi^*$ and $t^*$ that minimise $D_T$ can be computed in $O(n^4 \cdot n^2 \log^3 n) = O(n^6 \log^3 n)$ time.

Uses the matching algorithm by Vaidya (1988)
Minimising $D_{\Lambda}$ and $D$

Minimum distance matching under scaling?

Use exactly the same approach.
Minimising $D_\Lambda$ and $D$

Minimum distance matching under scaling?

Use exactly the same approach.

Minimum distance matching under translation and scaling?
Minimising $D_\Lambda$ and $D$

Minimum distance matching under scaling?

Use exactly the same approach.

Minimum distance matching under translation and scaling?

Same idea: $x$-align ($y$-align) two pairs of points.
Minimising $D_\Lambda$ and $D$

Minimum distance matching under scaling?
  Use exactly the same approach.

Minimum distance matching under translation and scaling?
  Same idea: $x$-align ($y$-align) two pairs of points.

**Theorem 2.** A $\phi^*$, $t^*$, and $\lambda^*$ that minimise $D$ can be computed in $O(n^8 \cdot n^2 \log^3 n) = O(n^{10} \log^3 n)$ time.
Preserving directional relations

NW  |  NE
---  |  ---
SW   |  SE

\(a_1\)  \(a_2\)
Preserving directional relations

\[ \phi(a_2) \quad \text{NE} \]

\[ \phi(a_1) \]

\[ a_1 \]

\[ a_2 \]
Preserving directional relations

Maximize the number of pairs with the right orientation.
Preserving directional relations

Minimize the number of pairs with the **wrong** orientation.

\[ W(\phi) = |\{(a_1, a_2) \mid (a_1, a_2) \in A \times A \land \text{dir}(a_1, a_2) \neq \text{dir}(\phi(a_1), \phi(a_2))\}|. \]
Preserving directional relations

Minimize the number of pairs with the wrong orientation.

\[ W(\phi) = |\{(a_1, a_2) \mid (a_1, a_2) \in A \times A \land \text{dir}(a_1, a_2) \neq \text{dir}(\phi(a_1), \phi(a_2))\}|. \]

Translation and scaling do not influence \( W \).
A 4-approximation algorithm

Compute a minimum distance matching with distance measure

\[ w(a, b) = |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + |y\text{-}rank_A(a) - y\text{-}rank_B(b)|. \]
A 4-approximation algorithm

Compute a minimum distance matching with distance measure

\[ w(a, b) = |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + |y\text{-}rank_A(a) - y\text{-}rank_B(b)|. \]

\[ x\text{-}rank_P(p) = 3 \]
\[ y\text{-}rank_P(p) = 4 \]
A 4-approximation algorithm

Compute a minimum distance matching with distance measure

\[ w(a, b) = |x\text{-rank}_A(a) - x\text{-rank}_B(b)| + |y\text{-rank}_A(a) - y\text{-rank}_B(b)|. \]

\( w(a, b) \) is the \( L_1 \)-distance in terms of ranks.
A 4-approximation algorithm

Compute a minimum distance matching with distance measure

\[ w(a, b) = |x - \text{rank}_A(a) - x - \text{rank}_B(b)| + |y - \text{rank}_A(a) - y - \text{rank}_B(b)|. \]

\( w(a, b) \) is the \( L_1 \)-distance in terms of ranks.

So compute an optimal matching using Vaidya’s Algorithm.

**Theorem 3.** A 4-approximation for minimising \( W \) can be computed in \( O(n^2 \log^3 n) \).
Implementation & Evaluation

**Implementation**: Easy; we only need an LP-solver.

**Evaluation**: We compare with spatial tree maps [Wood & Dykes], and minimizing the $L^2_2$ distance [Cohen & Guibas].
Implementation & Evaluation

**Implementation**: Easy; we only need an LP-solver.

**Evaluation**: We compare with spatial tree maps [Wood & Dykes], and minimizing the $L^2_2$ distance [Cohen & Guibas]

- **Quantitative**
  - distance
  - # and % preserved directional relations
  - # and % preserved adjacencies

- **Qualitative**
Implementation & Evaluation

**Implementation**: Easy; we only need an LP-solver.

**Evaluation**: We compare with spatial tree maps [Wood & Dykes], and minimizing the $L^2_2$ distance [Cohen & Guibas]

- **Quantitative**
  - distance
  - # and % preserved directional relations
  - # and % preserved adjacencies

- **Qualitative**
## Results

<table>
<thead>
<tr>
<th>Dir. Rel.</th>
<th>Adj.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SG</td>
<td>88%</td>
</tr>
<tr>
<td>$L_1$</td>
<td>96%</td>
</tr>
<tr>
<td>$W$</td>
<td>97%</td>
</tr>
<tr>
<td>$L_2$</td>
<td>98%</td>
</tr>
</tbody>
</table>

The results are visualized in a map of France, with different regions represented by different colors. The map includes two main sections, with the left side showing the results for the method `[Wood and Dykes]` (SG) and the right side showing the results for the method $L_2$. The bottom row of the map presents another method represented by $L_1$, while the top row shows yet another method represented by $W$. Each region in the map is color-coded based on the accuracy of the method applied to that region.
Results for the United States and the London Boroughs are in the paper.
Concluding Remarks & Future Work

Our method works for arbitrary point sets.
Concluding Remarks & Future Work

Our method works for arbitrary point sets.
Concluding Remarks & Future Work

Our method works for arbitrary point sets.

Future work: How to find cells (not) to use?
Concluding Remarks & Future Work

Our method works for arbitrary point sets.

Future work: How to find cells (not) to use?

Thank you!
Inversions vs Directions

\[ x\text{-}\text{rank}_A(a_1) < x\text{-}\text{rank}_A(a_2) \text{ and } x\text{-}\text{rank}_B(b_1) > x\text{-}\text{rank}_B(b_2) \]

\((a_1, a_2)\) is an inversion.
Inversions vs Directions

$x\text{-rank}_A(a_1) < x\text{-rank}_A(a_2)$ and $x\text{-rank}_B(b_1) > x\text{-rank}_B(b_2)$

$(a_1, a_2)$ is an inversion.

$(a_1, a_2)$ is an out-of-order pair
Inversions vs Directions

\[ x\text{-}rank_A(a_1) < x\text{-}rank_A(a_2) \quad \text{and} \quad x\text{-}rank_B(b_1) > x\text{-}rank_B(b_2) \]

\((a_1, a_2)\) is an inversion.

\[ \updownarrow \quad \leftrightarrow \]

\((a_1, a_2)\) is an out-of-order pair.

So \( W(\phi) = \# \text{inversions} = I(\phi) \).