Spatial Generalization and Aggregation of Massive Movement Data

Natalia Andrienko and Gennady Andrienko

Abstract—Movement data (trajectories of moving agents) are hard to visualize: numerous intersections and overlapping between trajectories make the display heavily cluttered and illegible. It is necessary to use appropriate data abstraction methods. We suggest a method for spatial generalization and aggregation of movement data, which transforms trajectories into aggregate flows between areas. It is assumed that no predefined areas are given. We have devised a special method for partitioning the underlying territory into appropriate areas. The method is based on extracting significant points from the trajectories. The resulting abstraction conveys essential characteristics of the movement. The degree of abstraction can be controlled through the parameters of the method. We introduce local and global numeric measures of the quality of the generalization, and suggest an approach to improve the quality in selected parts of the territory where this is deemed necessary. The suggested method can be used in interactive visual exploration of movement data and for creating legible flow maps for presentation purposes.

Index Terms—Movement, generalization, aggregation, information visualization, geovisualization, visual analytics.

1 Problem Statement

This paper addresses the issues of visualization of large amounts of movement data, that is, records about the spatial positions of some moving agents (such as people, vehicles, or animals) at different time moments. The main components of a position record are <agent identifier, time, spatial position>. A time-ordered sequence of positions of one agent is called *trajectory*.

Movement data are generated in vast amounts by means of current tracking technologies. A straightforward visualization of trajectories as lines or traces on a map or in a space-time cube [18], [19] is not appropriate for massive movement data: such a display is illegible because of cluttering and overlapping of the symbols. It is necessary to apply *data abstraction*, which is defined in [6] as the process of hiding detail of data while maintaining their essential characteristics. An analogous concept, called *cartographic generalization* [21], [25], exists in cartography and geographic visualization. From the existing methods of cartographic generalization, the method most appropriate for our purposes is *aggregation*, when several items are put together and represented as a single unit. In our case, the items would be trajectories or fragments of trajectories. Aggregation reduces the number of items, which is very helpful in case of numerous trajectories. At the same time, it does not just omit some items, but transforms the original items into a smaller number of constructs that summarize the properties of the original items.

Widely known examples of cartographic representation of aggregated movements of multiple agents are historical maps showing the movements of tribes and armies by arrow symbols. This kind of map is called *flow map* [23], [25]. The most famous flow map is the representation of Napoleon's Russian campaign of 1812 by Minard [28]. The map portrays several characteristics of the movement: the route, the locations of the army at different times, the directions of the movement, the change of the size of the army (i.e., the number of moving agents), splitting into parts, and rejoining of the parts. However, this and similar representations deal with *coherent* movement of multiple agents, i.e., the case when the agents are moving together as a single unit. This technique is not applicable to independently moving agents, such as cars or pedestrians moving over a city.

The term *flow map* is also used for maps where arrow symbols represent only the numbers of items or amounts of goods moving between some places, but not the routes of the movement [26], [27]. Given a set of predefined places, a flow map is built as follows: For each pair of places, the number of trajectories originating in the first place and ending in the second one is counted. The count so obtained is represented by an arrow symbol directed from the first to the second place with the thickness or brightness proportional to the count. To improve the legibility of the resulting display, the symbols representing minor amounts are typically omitted. The algorithm for automated design of flow maps suggested in [23] minimizes crossings between symbols.

The idea of flow map can be extended to take into account not only the starts and ends of the trajectories, but also the intermediate positions along the routes. For this purpose, the trajectories are divided into segments. Two methods for division are possible: by time intervals and by visits of the places. An example of time-based division can be seen in the representation of the movements of tourists in New Zealand by Drecki and Forer [8] (reproduced in [2]). The trajectories of the tourists were divided into segments corresponding to the days of visit, starting from the arrival

• *The authors are with Fraunhofer Institute IAIS—Intelligent Analysis and Information Systems, Schloss Birlinghoven, Sankt-Augustin D-53754, Germany. E-mail: gennady.andrienko@iais.fraunhofer.de.*

Manuscript received 24 June 2009; revised 21 Sept. 2009; accepted 4 Oct. 2009; published online 16 Feb. 2010.
Recommended for acceptance by A. MacEachren.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2009-06-0127.
Digital Object Identifier no. 10.1109/TVCG.2010.44.

to New Zealand. The representation consists of six parallel planes, shown in a perspective view, with a map of New Zealand depicted on each plane. The planes correspond to the days of the tourists' travel. The movements of the tourists are represented as lines connecting the locations of the major tourist destinations on successive planes. The brightness of a line corresponds to the number of people that moved from its origin location (on the upper plane) to the destination location (on the lower plane) between the days corresponding to the upper and lower planes.

When time-based division is used, the results of the aggregation should be visualized in such a way that the aggregate moves corresponding to different time intervals are differentiated. Drecki and Forer do this by involving an additional display dimension. Another possible approach is a series of maps ("small multiples"), each showing the movements during one time interval.

In the place-based division, the sequence of visited places is found for each trajectory, and the trajectory is divided into segments going between successive places. Then, the segments with coinciding places of starts and places of ends are put together and represented as aggregate moves between these places.

All these approaches assume that there is a *predefined set of relevant places*. In our research, we have been looking for a way to *define suitable places* for aggregating movement data and building flow maps in cases when positions of moving agents are specified only by numeric coordinates lacking any semantics. We have developed a computational method that uses these coordinates to partition the territory into suitable areas, so that flow maps built on the basis of these areas can adequately portray essential spatial characteristics of the movement. The design of flow maps, however, is not our focus; our method is meant to prepare input data for flow maps.

In Section 2, we shall demonstrate several examples of data generalizations obtained by means of the method we have developed. The method itself will be described in detail in Section 3. In Section 4, we shall talk about the quality of the generalization: how it can be measured and improved. In Section 5, we shall briefly discuss the possible uses of our method, which are not limited to building static flow maps. In Section 6, we shall give an overview of the related literature and then conclude in Section 7.

2 Examples of Movement Generalization

In brief, we suggest a method that extracts specific points from the trajectories, groups them by spatial proximity, and uses the centers of the groups as generating points for Voronoi tessellation of the territory. The resulting Voronoi cells are used as the places for aggregating movement data and building flow maps. The degree of the generalization depends on the sizes of the cells which, in turn, depend on the spatial extents of the point groups. The desired spatial extent (radius) is a parameter of the method.

2.1 Example 1: Cars in Milan

In this example, we use a data set collected by GPS-tracking of 17,241 cars in Milan (Italy) during 1 week. The data set consists of more than 2 million records, each including car identifier, time stamp (date and time of the day), and geographical coordinates. The time intervals between the records of the same car are irregular, mostly ranging from 30 to 45 seconds. The data have been kindly provided by Comune di Milano (Municipality of Milan). The whole data set is too big for processing in RAM; therefore, we shall use a subset consisting of about 6,200 trajectories from a 4-hour time interval. Fig. 1a demonstrates this set of trajectories represented on a map by linear symbols with special markers for the start and end positions of the trajectories: tiny hollow and filled squares, respectively. The lines are drawn on the map with 10 percent opacity, which gives an idea about the relative density of the movement in different places and about the topology of the road network in Milan.

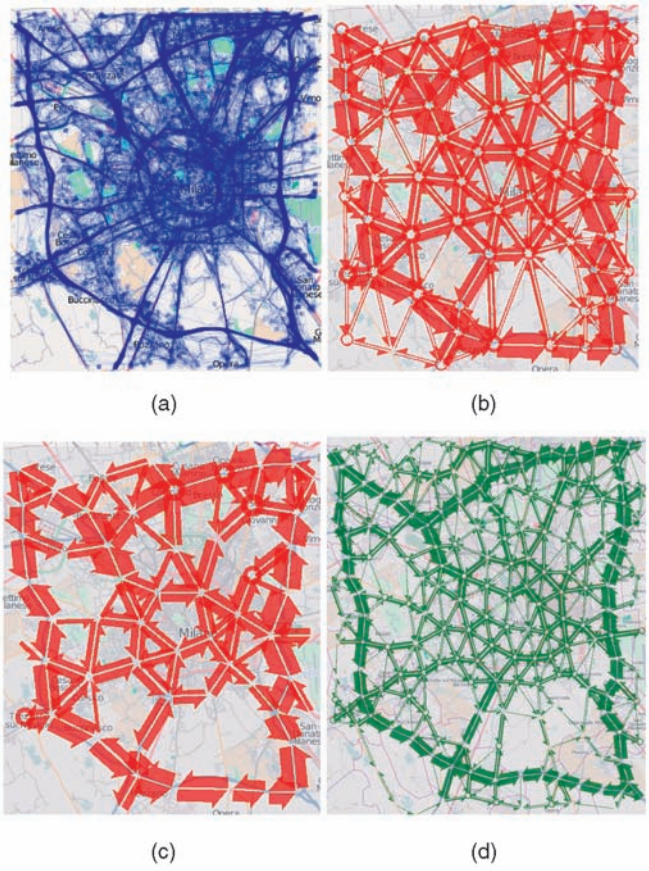

Fig. 1. (a) The original 6,187 car trajectories are drawn with 10 percent opacity. (b) The trajectories have been aggregated; all flows are visible. The maximum arrow width corresponds to 253 trajectory segments. (c) The same as Fig. 1b, but only the flow symbols representing 50 or more trajectory segments are visible. (d) A lower degree of abstraction of the same data. The maximum arrow width corresponds to 255 trajectory segments.

Fig. 1b represents the result of aggregating the trajectories by our method (radius = 3,000 m). For the sake of legibility, the screenshots do not include the boundaries of the Voronoi cells that have been used for the aggregation. The special "half-arrow" symbols represent movements between the cells in two opposite directions. This technique has been borrowed from Tobler's maps [26], [27]. The starts and ends of the symbols lie on the lines connecting the generating points of the cells. The symbols are slightly shorter than the lines to reduce intersections. The widths of the symbols are proportional to the numbers of the

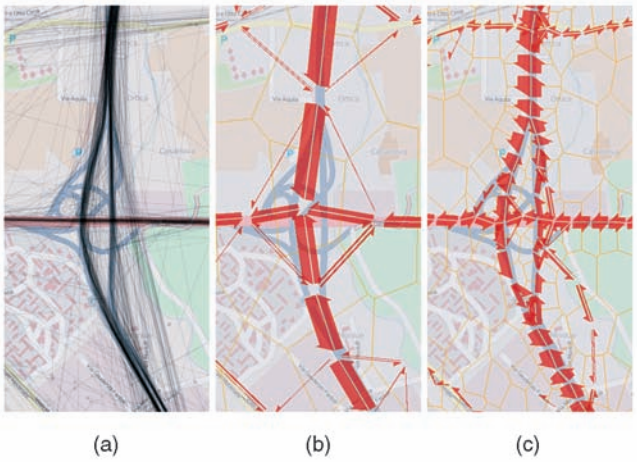

Fig. 2. (a) An enlarged fragment of the map of Milan. The trajectories are shown in black with 10 percent opacity. (b) and (c) Generalized representations of the trajectories generated with the values of the parameter 500 and 100 m, respectively.

trajectory segments they represent. The circular symbols stand for trajectories that fully fit inside one cell. In Fig. 1c, the symbols corresponding to less than 50 trajectory segments are hidden; hence, only the major flows are visible. The background map images in all maps in the paper have been taken from the Open Street Map Web server (www.OpenStreetMap.org).

A good correspondence between the original data and the generalized representation can be noted. Thus, Fig. 1a displays a high intensity of the movement on the peripheral belt roads. The same information can be gained from the generalized view (Figs. 1b and 1c), which contains thick arrows (meaning intensive movement) positioned closely to the belt roads. The shapes formed by these arrows even approximate the shapes of the roads. The radial flows in the aggregated representation are positioned closely to the busiest roads connecting the central part with the periphery. In comparison to Fig. 1a, the generalized view conveys additional information about the relative intensities of movements in opposite directions.

A comment should be made that our automatically generated flow maps are by no means perfect in terms of map design. Thus, a more sophisticated approach presented in [23] produces esthetically more pleasing maps. Unfortunately, it is limited to showing movements originating in one or two places. We do not know any good automatic method for flow map design that would cope with numerous origins and numerous destinations. The topic of our work is not the design of flow maps, but partitioning of the territory and aggregation of movement data, which precede the map design. The simple visualization technique that we use is sufficient for demonstrating the feasibility of our approach.

Fig. 1d demonstrates the possibility of varying the level of data abstraction. Here, the territory of Milan has been divided into smaller compartments than in Figs. 1b and 1c ($\mathrm{radius} = 1{,}500$ m). The resulting generalization is finer when consistent with the previous, coarser generalization.

Fig. 2 demonstrates very low degrees of data abstraction, which are achieved with the parameter values 500 m (Fig. 2b) and 100 m (Fig. 2c). These degrees of abstraction are suitable for large-scale maps, from which only small fragments can be used in the paper. On the contrary, Fig. 3 demonstrates very high degrees of abstraction achieved with parameter values 6 km (Fig. 3a) and 10 km (Fig. 3b). Although the generalization shown in Fig. 3b may seem excessive, the one shown in Fig. 3a can still be quite useful. The information conveyed is consistent with Fig. 1c.

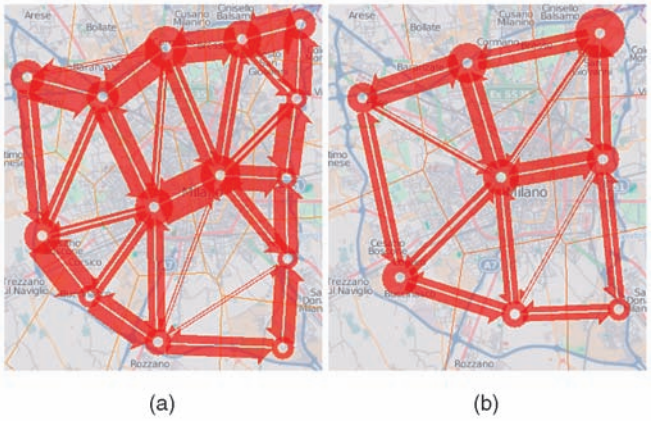

Fig. 3. The Milan car trajectories have been generalized using the parameter values (a) 6 and (b) 10 km.

2.2 Example 2: Roe Deer

In this example, we use a data set with positions of 74 roe deer collected over a period of about 5 years. The track durations range from 5 to 1,077 days. The data have been kindly given to us by the scientists from the Bavarian National Park. From the representation of the original trajectories of the animals shown in Fig. 4, it can be seen that the movements mostly occurred within relatively small areas. There was not much movement between the areas. The aggregated representation of the data, which is overlaid on the lines of the trajectories, reflects these properties of the data. This example shows that the suggested method of generalization is applicable not only to movements constrained by a street network, but also to network-free movements such as movements of wild animals.

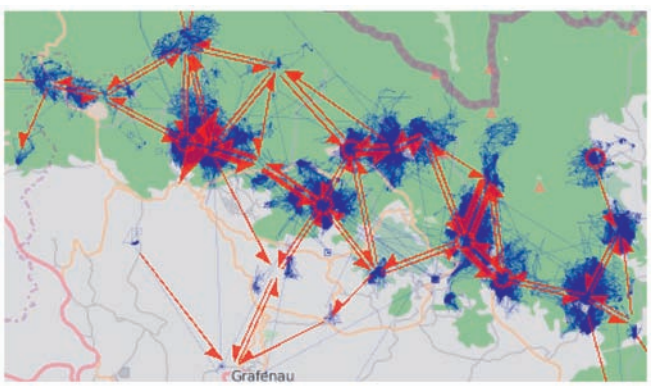

Fig. 4. Blue: the original trajectories of 74 roe deer, shown with 10 percent opacity. Red: the trajectories have been generalized to flows, which are superimposed on the trajectories. The maximum arrow width corresponds to 607 trajectory segments.

3 Description of the Method

Our method for generalization of movement data consists of the following major steps:

1. extract characteristic points from the trajectories;
2. group the extracted points by spatial proximity;
3. extract the centroids (average points) of the point groups and use them as generating points for Voronoi tessellation;
4. use the resulting Voronoi cells as places for place-based division of the trajectories into segments;
5. for each ordered pair of places, aggregate the trajectory segments starting in the first place and ending in the second place;
6. measure the quality of the generalization and improve it if necessary.

The first five steps are described in this section, while the final step will be covered in Section 4.

3.1 Extracting Characteristic Points of Trajectories

Characteristic points of trajectories include their start and end points, the points of significant turns, and the points of significant stops (pauses in the movement). If a trajectory has long straight segments, it is also necessary to take representative points from these segments. Otherwise, straight segments will not be taken into account in choosing generating points for Voronoi cells and, as a result, may be inadequately represented by flows (i.e., the flows representing these segments may deviate too much from the directions of the segments).

We use the following algorithm to extract characteristic points from a trajectory:

Algorithm 1. Extracting characteristic points from a trajectory

Given:

- Trajectory $\mathrm{T} = (\mathrm{x}_i, \mathrm{y}_i, \mathrm{t}_i), 1 \le \mathrm{i} \le \mathrm{n}, \mathrm{n} \ge 2, \mathrm{t_i} < \mathrm{t_{i+1}}$ for $\forall \mathrm{i} < \mathrm{n}$ ($\mathrm{x_i}$ and $\mathrm{y_i}$ are spatial coordinates and $\mathrm{t_i}$ is time);
- MinAngle—the minimum angle between the directions of consecutive trajectory segments to be considered as a significant turn;
- MinStopDuration—the minimum time spent in approximately the same position to be treated as a significant stop;
- MinDistance—when the distance between two consecutive points is below this value, the points are treated as approximately the same position;
- MaxDistance—the maximum allowed distance between consecutive characteristic points extracted from the trajectory (i.e., if the trajectory has a straight segment with the length more than this value, representative points must be taken such that the distances between them do not exceed this value).

Description of the algorithm:

1: **let** $\mathrm{C} = \{(\mathrm{x_1}, \mathrm{y_1}, \mathrm{t_1})\}$; **let** $\mathrm{i} = 1$;
2: **let** $\mathrm{j} = \mathrm{i} + 1$;
3: **if** $\mathrm{j} \ge \mathrm{n}$ **then go_to** 9; **end_if**;
4: compute $\mathrm{dSpace_{i,j}} = \mathit{spatial\_distance}((\mathrm{x_i}, \mathrm{y_i}), (\mathrm{x_j}, \mathrm{y_j}))$;
5: **if** $\mathrm{dSpace_{i,j}} \ge \mathrm{MaxDistance}$ **then**
/* the jth point is a representative point from a long segment */
let $\mathrm{C} = \mathrm{C} \cup \{(\mathrm{x_j}, \mathrm{y_j}, \mathrm{t_j})\}$; **let** $\mathrm{i} = \mathrm{j}$; **go_to 2;**
end_if;
6: **for** $\mathrm{k} = \mathrm{j} + 1$ **to** n
compute $\mathrm{dSpace_{j,k}} = \mathit{spatial\_distance}((\mathrm{x_j}, \mathrm{y_j}), (\mathrm{x_k}, \mathrm{y_k}))$;
if $\mathrm{dSpace_{j,k}} \ge \mathrm{MinDistance}$ **then go_to** 7; **end_if**;
end_for;
go_to 9; /* no points far enough from the jth point */
7: **If** $\mathrm{k} > \mathrm{j} + 1$
/* there are points between the jth and kth points whose spatial positions are close to the jth point */
then
compute $\mathrm{dTime} = \mathrm{t_{k-1}} - \mathrm{t_j}$
if $\mathrm{dTime} \ge \mathrm{MinStopDuration}$ **then**
/* the j-th point is a significant stop point */
let $\mathrm{C} = \mathrm{C} \cup \{(\mathrm{x_j}, \mathrm{y_j}, \mathrm{t_j})\}$; **let** $\mathrm{i} = \mathrm{j}$; **let** $\mathrm{j} = \mathrm{k}$; **go_to** 3;
else
/* compute the average spatial position */
compute $(\mathrm{x_{ave}}, \mathrm{y_{ave}})$:
$\mathrm{x_{ave}} = \mathit{get\_mean}(\mathrm{x_p}), \mathrm{y_{ave}} = \mathit{get\_mean}(\mathrm{y_p})$, $\mathrm{j} \le \mathrm{p} \le \mathrm{k} - 1$;
/* find the closest point to the average position */
find $\mathrm{m}, \mathrm{j} \le \mathrm{m} \le \mathrm{k} - 1 : \mathit{spatial\_distance}((\mathrm{x_m}, \mathrm{y_m}), (\mathrm{x_{ave}}, \mathrm{y_{ave}})) \le \mathit{spatial\_distance}((\mathrm{x_p}, \mathrm{y_p}), (\mathrm{x_{ave}}, \mathrm{y_{ave}})))$, $\mathrm{j} \le \mathrm{p} \le \mathrm{k} - 1$;
/* this will be a representative point among the points from jth to $(\mathrm{k} - 1)$th */
let $\mathrm{j} = \mathrm{m}$;
end_if;
end_if;
8: compute
$\mathrm{aTurn} = \mathit{angle\_between\_vectors}(<(\mathrm{x_i}, \mathrm{y_i}), (\mathrm{x_j}, \mathrm{y_j})>, <(\mathrm{x_j}, \mathrm{y_j}), (\mathrm{x_k}, \mathrm{y_k})>)$;
if $\mathrm{aTurn} \ge \mathrm{MinAngle}$ **then**
/* the j-th point is a significant turning point */
let $\mathrm{C} = \mathrm{C} \cup (x_j, y_j, t_j)$; **let** $\mathrm{i} = \mathrm{j}$; **let** $\mathrm{j} = \mathrm{k}$;
else
let $\mathrm{j} = \mathrm{j} + 1$; /* skip the jth point */
end_if;
go_to 3;
9: **let** $\mathrm{C} = \mathrm{C} \cup \{(\mathrm{x_n}, \mathrm{y_n}, \mathrm{t_n})\}$; /* add the end point to C */
10: **return** C;

In Fig. 5, Algorithm 1 has been applied to the same trajectory with three different settings of the parameters MinDistance and MaxDistance. Each of the horizontal sections labeled A, B, and C contains three panels. The larger panel on the left shows the whole trajectory as a line and the extracted characteristic points as small semitransparent circles. The small panels on the right show two fragments of the image enlarged; these are marked in the larger panel by boxes. The fragments demonstrate the effect of the parameter MinDistance. The role of this parameter is to omit minor fluctuations of a position. The parameter MaxDistance affects the positions and numbers of representative points taken from trajectory segments with no significant turns, as can be seen from comparing the big image in panel C with those in panels A and B.

Fig. 6 shows the characteristic points extracted from the 6,187 trajectories of cars in Milan presented in Fig. 1 and from the 74 trajectories of roe deer presented in Fig. 4. A total

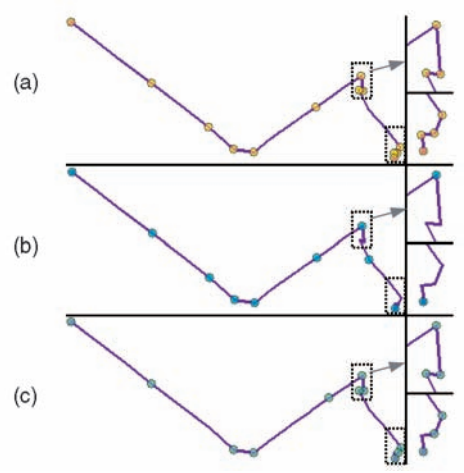

Fig. 5. Characteristic points extracted from the same trajectory by Algorithm 1 with different settings of the parameters MinDistance and MaxDistance. (a) $\mathrm{MinDistance} = 100$ m, $\mathrm{MaxDistance} = 2{,}000$ m. (b) $\mathrm{MinDistance} = 500$ m, $\mathrm{MaxDistance} = 2{,}000$ m. (c) $\mathrm{MinDistance} = 250$ m, $\mathrm{MaxDistance} = 2{,}500$ m. In all cases, $\mathrm{MinAngle} = 30$ degree and $\mathrm{MinStopDuration} = 300$ seconds.

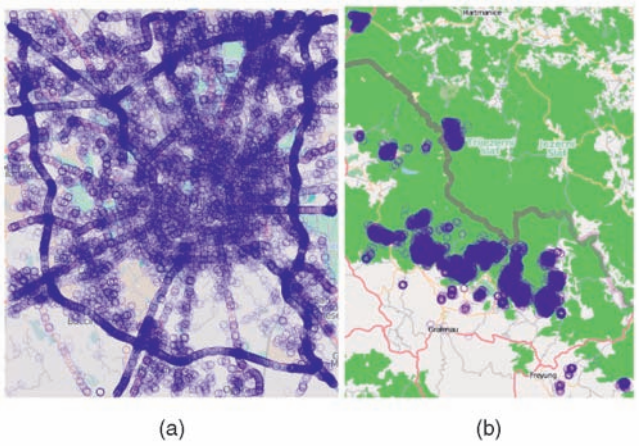

Fig. 6. The characteristic points extracted from (a) the 6,187 trajectories of cars in Milan and (b) the 74 trajectories of roe deer. The parameters in both cases were $\mathrm{MinAngle} = 30°$, $\mathrm{MinStopDuration} = 300$ seconds, $\mathrm{MinDistance} = 100$ m, and $\mathrm{MaxDistance} = 3{,}000$ m. The points are represented by circles drawn with 10 percent opacity.

of 36,328 points were extracted from the Milan data and 35,113 points from the roe deer data. In both cases, the computing time was 93 milliseconds. All computing times mentioned in this paper have been measured on a personal computer with 2.4 GHz clock rate and 3.25 GB of RAM under the operating system Windows XP. The computational complexity of Algorithm 1 is linear with respect to the number of points in a trajectory. The upper limit of the computation time for a trajectory with N points is proportional to M*N, where M is the maximum number of consecutive trajectory points fitting in a circle with the diameter MinDistance.

The next step after extracting the characteristic points from all trajectories is to group the points in space, so that the spatial extents of the groups approximate the desired sizes of the cells (places) to be used for the generalization.

3.2 Grouping Characteristic Points in Space

A natural approach to group points in space is the use of some clustering method. However, the existing clustering algorithms are not well suited to our needs. Thus, the popular clustering algorithms "k-means" and "k-medoids" [17], [30] require that the desired number of clusters be specified as a parameter. In our case, however, the number of clusters is not known in advance. The density-based clustering algorithms [10] can produce clusters of arbitrary shapes and spatial extents whereas we need such groups of points that can be enclosed by convex polygons of a certain size, depending on the desired level of the generalization. We did not find any clustering algorithm capable of producing convex spatial clusters with desired spatial extents. Therefore, we have devised and implemented the following algorithm:

Algorithm 2. Grouping points in space
Given:
- Set of points $P = (\mathrm{x_i}, \mathrm{y_i}), 1 \le \mathrm{i} \le \mathrm{n}$.
- MaxRadius—the desired radius of a group, i.e., the radius of the circumferential circle of all its points.

Description of the algorithm:
1. find xMin, xMax, yMin, yMax:
   $\forall \mathrm{p} \in \mathrm{P}: \mathrm{xMin} \le \mathrm{p.x} \le \mathrm{xMax}$ & $\mathrm{yMin} \le \mathrm{p.y} \le \mathrm{yMax}$.
   /* the bounding rectangle of P */
2. $\mathrm{buildG} = \mathrm{Grid}(\mathrm{xMin}, \mathrm{xMax}, \mathrm{yMin}, \mathrm{yMax}, \mathrm{MaxRadius}, \mathrm{MaxRadius})$;
   /* G is a grid with square cells covering the bounding rectangle of P. The size of a grid cell is MaxRadius × MaxRadius. The grid is used as a spatial index for a better efficiency of the algorithm. As point groups are built, their centroids (average points) are put in the grid cells according to their coordinates. */
3. **let** $R = \oslash$; /* this will be the resulting set of groups */
4. **for_each** $\mathrm{p} \in \mathrm{P}$ $\mathrm{put\_in\_proper\_group}(\mathrm{p}, \mathrm{R}, \mathrm{G})$;
   end_for_each;
5. redistribute_points(P,R,G);
6. **return** R;

procedure put_in_proper_group (Point p, Set R, Grid G):
1. **let** $\mathrm{c} = \mathrm{get\_closest\_centroid}(\mathrm{p}, \mathrm{G})$;
2. **if** $\mathrm{c} = \mathrm{null}$ **then**
   build $\mathrm{g} = \mathrm{Group}(\mathrm{p}, \mathrm{p})$;
   /* the new group g consists of a single point p, which is the centroid of the group */
   let $\mathrm{R} = \mathrm{R} \cup \{\mathrm{g}\}$;
   else
   get $\mathrm{g} \in \mathrm{R}$: g.centroid = c;
   /* g is the group with the centroid c */
   let $\mathrm{g.members} = \mathrm{g.members} \cup \{\mathrm{p}\}$;
   remove c from G;
   compute g.centroid = *get_centroid*(g.members);
   end_if;
3. **let** $(\mathrm{i}, \mathrm{j}) = \mathrm{get\_grid\_position}(\mathrm{g.centroid}, \mathrm{G})$;
4. put g.centroid in G.cell[i,j];

procedure get_closest_centroid (Point p, Grid G):
1. **let** $(\mathrm{i}, \mathrm{j}) = \mathrm{get\_grid\_position}(\mathrm{p})$;
2. /* take from the cell (i,j) and the neighboring cells all centroids whose distances to p do not exceed MaxRadius */

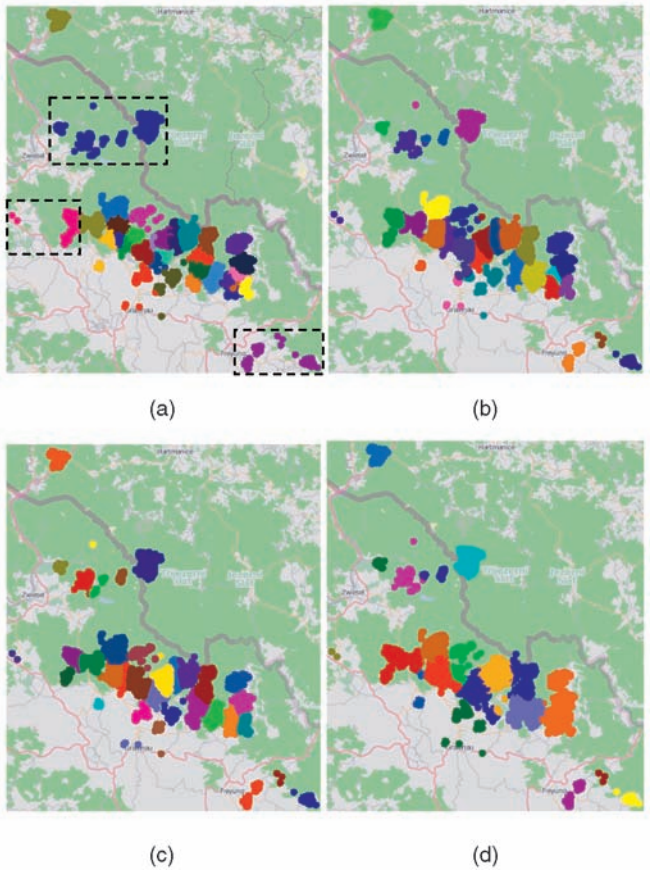

Fig. 7. (a) The characteristic points extracted from the trajectories of the roe deer have been clustered by the "simple k-means" method. (b) The same points have been grouped by Algorithm 2; $\mathrm{MaxRadius} = 3{,}000$ m. (c) The groups resulting from Algorithm 2 have been optimized using Algorithm 3. (d) The points have been grouped by Algorithms 2 and 3 with $\mathrm{MaxRadius} = 5{,}000$ m. In all cases, the groups are represented by colors of the circles representing the points.

let $\mathrm{C} = \emptyset$;
for $\mathrm{k} =$ *max*$(\mathrm{i}-1{,}1)$ **to** *min*$(\mathrm{i}+1{,}\mathrm{G.nRows})$
for $\mathrm{m} =$ *max*$(\mathrm{j}-1{,}1)$ **to** *min*$(\mathrm{j}+1{,}\mathrm{G.nColumns})$
for_each $\mathrm{c} \in \mathrm{G.cell[k,m]}$
if $spatial\_distance(\mathrm{p,c}) \leq \mathrm{MaxRadius}$ **then**
let $\mathrm{C} = \mathrm{C} \cup \{\mathrm{c}\}$;
end_if;
end_for_each;
end_for;
end_for;

3. **if** $\mathrm{C} = \emptyset$ **then return** null; **end_if**;
4. /* if C contains several centroids, take the one that is the closest to p */
find $\mathrm{c_k} \in \mathrm{C} : \forall \mathrm{c_m} \in \mathrm{C}\ spatial\_distance(\mathrm{c_k,p}) \leq spatial\_distance(\mathrm{c_m,p})$;
5. **return** $\mathrm{c_k}$;

procedure get_grid_position (Point p, Grid G):
/* compute the grid cell in which the point fits */
1. **let** $\mathrm{i} =$ *floor*((p.x-xMin)/MaxRadius);
2. **let** $\mathrm{j} =$ *floor*((p.y-yMin)/MaxRadius);
3. **return** (i,j);

TABLE 1
Characteristics of the Clusters Generated by Simple K-Means and by Algorithm 2

	Simple k-means			Algorithm 2		
Metrics	max.	mean	st.dev.	max.	mean	st.dev.
– radius, m	7227	2237	1656	3372	1835	738
– mean distance to centroid, m	3819	648	666	1344	605	302
– median distance to centroid, m	3883	571	635	1285	531	278
– density, points/km^2	1548	174	285	6059	276	1029

procedure redistribute_points (Set P, Set R, grid G):
/* redistribute the points among the groups */
1. /* remove all points from all groups but keep the group centroids */
for_each $\mathrm{g} \in \mathrm{R}$ **let** $\mathrm{g.members} = \emptyset$; **end_for_each**;
2. /* for each point, find the closest centroid and put the point in the respective group without changing the centroid */
for_each $\mathrm{p} \in \mathrm{P}$
let $\mathrm{c} = \mathrm{get\_closest\_centroid(p,G)}$;
get $\mathrm{g} \in \mathrm{R}$: $\mathrm{g.centroid} = \mathrm{c}$;
let $\mathrm{g.points} = \mathrm{g.points} \cup \{\mathrm{p}\}$;
end_for_each;

The computational complexity of this algorithm is linear with respect to the number of points. To place a point in the right group, it is necessary to compute its distances to the group centroids located in at most nine grid cells (see procedure get_closest_centroid). If K is the maximum number of centroids fitting in a grid cell, distances to at most K*9 centroids need to be computed. Since the sizes of the grid cells are determined by the value MaxRadius, which is also the maximum radius of a group, a cell may contain at most four group centroids (this may happen in a particular case when the coordinates of the centroids coincide with the corners of the cell). In our experiments, the computing time was 265 milliseconds for the 36,328 points extracted from the Milan data set and 266 milliseconds for the 35,113 points extracted from the roe deer data set (with $\mathrm{MaxRadius} = 3{,}000$ m in both cases). In the first case, we obtained 54 groups, and in the second case, 33 groups. For comparison, we applied the algorithm "simple k-means" available in the Weka data mining library [30] to the same sets of points. Producing 54 clusters from the Milan set took 79.59 seconds, and producing 33 clusters from the roe deer set took 29.84 seconds.

Our method differs from simple k-means not only in the computing time but also in the results produced. As can be seen from Figs. 7a and 7b, the groups of points produced by Algorithm 2 (Fig. 7b) are spatially more compact and do not differ so much in spatial extents as the clusters produced by simple k-means (Fig. 7a). The boxes in Fig. 7a enclose three groups of spatially scattered points that have been treated as clusters by the simple k-means method. Algorithm 2 divided these into smaller groups. Table 1 contains statistics of the sizes and densities of the point groups generated by simple k-means and by Algorithm 2. The three numbers given per

method and metrics are the maximum, mean, and standard deviation, respectively. It can be seen that the radii of the clusters by simple k-means vary greatly and reach much larger values than for the groups by Algorithm 2. The mean and median distances from the points to the centroids of their groups do not differ much on average, while the maximum values are much larger for simple k-means. The maximum and average densities of the groups by Algorithm 2 are much higher than those for simple k-means.

Hence, Algorithm 2 suits much better to our needs. Still, we would like to decrease the sensitivity of the results to the order in which the points are processed and to improve the correspondence between the generated groups of points and the "natural" clusters, i.e., dense concentrations of points. For this purpose, we have devised a method that optimizes the groups generated by Algorithm 2. The idea is to regroup the points around the centers of dense regions.

Algorithm 3. Optimization of point groups with respect to the densities of the points

Given:

- the set of point groups produced by Algorithm 2: $R = \{g_k\}, 1 \le k \le N$;
- the grid G built in the same way as in Algorithm 2 (this may be the same grid but all centroids must be removed from the cells).

Description of the algorithm:

1. **for_each** $g_k \in R$:
   1.1 compute $\mathrm{medXY_k} = \mathrm{Point(x, y)}$ where
   $x = get\_median\_x(g_k.\mathrm{members})$;
   $y = get\_median\_y(g_k.\mathrm{members})$;
   /* x is the median of all x-coordinates and y is the median of all y-coordinates */
   1.2. compute $\mathrm{mDist_k} = get\_mean\_distance(g_k.\mathrm{members}, \mathrm{medXY_k})$;
   1.3. compute $\mathrm{dens_k} = count(g_k.\mathrm{members})/\mathrm{mDist_k^2}$;
   /* an estimation of the density: the number of points divided by the squared mean distance to $\mathrm{medXY_k}$ */
   end_for_each;
2. compute $\mathrm{mDens} = get\_median(\mathrm{dens_k}|1 \le k \le N)$;
3. build $\mathrm{oList} = \mathrm{OrderedList}(\{g_i\}|1 \le i \le N\}$ where $\forall i, i > 1 : \mathrm{dens_i} \le \mathrm{dens_{i-1}}$;
   /* oList contains the groups in the order of decreasing densities */
4. **let** $R' = \varnothing$;
5. **for** i = 1 **to** N **while** $\mathrm{dens_i} \ge \mathrm{mDens}$
   let $g_i = \mathrm{oList.element[i]}$;
   find $\mathrm{pMed} \in g_i.\mathrm{members} : \forall p \in g_i.\mathrm{members}$ $\mathrm{spatial\_distance(p, medXY_i)} \ge \mathrm{spatial\_distance(pMed, medXY_i)}$;
   /* pMed is the group member that is the closest to $\mathrm{medXY_i}$; it will be a seed of a new group */
   build $g' := \mathrm{Group}(\varnothing, \mathrm{pMed})$;
   let $R' = R' \cup \{g'\}$;
   let $(j, k) = \mathrm{get\_grid\_position(pMed, G)}$;
   put pMed in G.cell[j,k];
   end_for;
6. **for** $i = 1$ **to** N
   let $g = \mathrm{oList.element[i]}$;
   for_each $p \in g$ put_in_proper_group(p, R', G);
   end_for_each;
   end_for;
7. $\mathrm{redistribute\_points}(P, R', G)$;
8. **return** R';

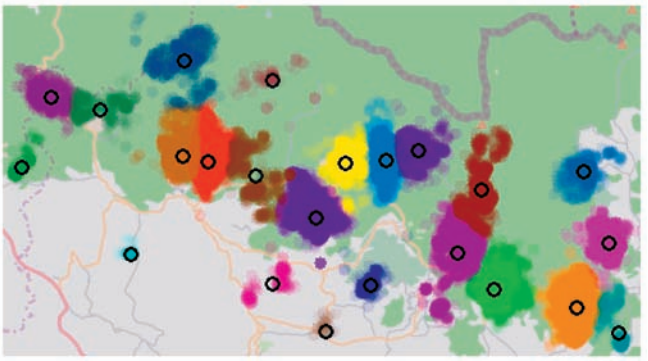

Fig. 8. The area containing the majority of the groups shown in Fig. 7c is enlarged. The points are drawn with 10 percent opacity. The hollow black circles represent the centroids of the groups. A good correspondence between the groups produced by Algorithms 2 and 3 and the existing dense concentrations of points can be observed.

The results of Algorithm 3 for the points extracted from the trajectories of the roe deer are presented in Fig. 7c. By comparing it with Fig. 7b, we see that Algorithm 3, indeed, improves the correspondence of the computed groups to the natural clusters. This can be better seen in Fig. 8, where the area containing the majority of the groups is enlarged. The points are drawn with 10 percent opacity to enable the estimation of the densities. The hollow black circles represent the centroids of the groups.

Fig. 7d demonstrates the impact of the parameter MaxRadius. The groups in Figs. 7b and 7c have been produced with $\mathrm{MaxRadius} = 3{,}000$ m, and the groups shown in Fig. 7d with $\mathrm{MaxRadius} = 5{,}000$ m. We judge the results of the grouping to be quite good for different values of MaxRadius.

The estimation of the point density in a group (step 1 of Algorithm 3) requires explanation. Point density could be computed as the number of points divided by the spatial extent of the group, which can be approximated by the area of the circumferential circle or bounding rectangle. However, if a group consists of a compact dense cloud of points and one or a few outliers located far from this cloud, the computed density may be rather low due to the large size of the enclosing shape. If a group contains a dense cloud comprising the bulk of the points, the point medXY whose x and y-coordinates are the medians of the x and y-coordinates of the group members is likely to be located inside this cloud. We use the mean distance of the points to medXY as an estimation of the size of the dense cloud. This allows us to give proper attention to groups where points are densely concentrated irrespective of occasional outliers. Furthermore, in step 5, we find the group member that is the closest to medXY and use it as a seed for a new group.

The computational complexity of the optimization phase (Algorithm 3) is the same as for Algorithm 2. In fact, this is a reapplication of Algorithm 2 after some preparatory operations (steps 1-5), which do not depend on the number of points, but only on the number of groups. In our experiments, the optimization phase takes less time than the initial grouping (e.g., 203 versus 266 milliseconds for the roe deer data and 235 versus 265 milliseconds for the Milan cars

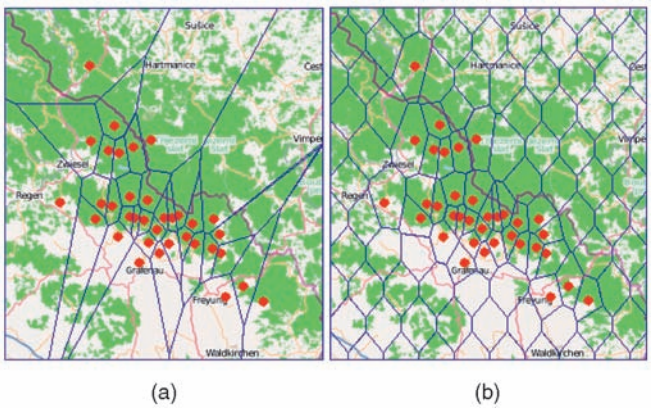

Fig. 9. (a) Voronoi polygons generated using only the group centroids. (b) Voronoi polygons generated using the group centroids and additional points in empty areas and around the bounding rectangle of the trajectories.

data). The reason is that the points in the optimization phase do not come in a random order, but are taken from existing groups, and there are fewer groups to check in order to find an appropriate group for each coming point.

3.3 Partitioning the Territory

To divide the territory into appropriate compartments, we use the centroids of the point groups as the generating points for Voronoi cells. We also introduce additional generating points around the boundaries of the territory and in the areas where there are no characteristic points from the trajectories. This allows us to obtain cells of more even sizes and shapes. Thus, in the data about the roe deer, the points are concentrated in limited areas. If only the group centroids are used for the partitioning, some of the resulting cells have very elongated shapes and are much larger than the others, as illustrated in Fig. 9a. In Fig. 9b, the division has been done with the use of additional points. The cells are much more even in size and shape. The additional points are generated in a regular manner. A new point is added only if it is sufficiently far from all group centroids, which means that the distance is more than the doubled MaxRadius. The use of additional generating points is not absolutely necessary, but it can improve the appearance of the resulting flow maps.

The Voronoi cells are built on the basis of Delaunay triangulation, for which we use the Java code developed by Chew [5].

3.4 Dividing Trajectories into Segments

As said before, we apply place-based division of trajectories into segments. This is done as follows: For each trajectory, the cell c_1 containing its first point p_1 is found. Then, the second and following points of the trajectory are checked for being inside c_1 until finding a point p_i not contained in c_1. For this point p_i, the containing cell c_2 is found. The trajectory segment from the first point to the ith point is represented by the vector (c_1, c_2). Then, the procedure is repeated: the points starting from p_{i+1} are checked for containment in c_2 until finding a point p_k outside c_2, a cell c_3 containing p_k is found, and so forth up to the last point of the trajectory. In the result, the trajectory is represented by a sequence of cells $\{c_1, c_2, \ldots, c_n\}$. There may be also a case when all points of a trajectory are contained in one and the same cell c_1. Then, the whole trajectory is represented by the vector (c_1, c_1).

Let us consider the more general case when a trajectory is represented by a sequence of at least two cells. For two consecutive cells c_i and c_{i+1}, two possibilities exist: 1) c_i and c_{i+1} are adjoining cells, i.e., having a common edge and 2) c_i and c_{i+1} are not adjoining. The second case may need a special treatment. In a visual representation, a vector connecting nonadjoining cells intersects other vectors, which decreases the legibility of the display. It may be preferable to avoid such vectors. This can be done by inserting intermediate cells between c_i and c_{i+1}, such that any two consecutive cells in the resulting sequence are adjoining. We do this by means of linear interpolation.

Let p_m be the last point of the trajectory contained in c_i. We build a straight line between p_m and p_{m+1}, which is contained in c_{i+1}, and find all cells intersected by this line. These cells are inserted in the sequence between c_i and c_{i+1}. The corresponding intermediate points of the trajectory are computed as the points of the crossing line having the minimum distances to the generating points of the cells.

However, not any movement data permit interpolation between known positions. In particular, interpolation may be inappropriate in a case of large time intervals between the position records in the data. For example, in a data set with positions of mobile phone users, the position records exist only for the time moments when the users make calls. It would be inappropriate to make any assumptions concerning their positions between the calls. The set of records about georeferenced photographs published in flickr and/or Panoramio (see Section 5.3) has similar properties. This data set contains sequences of positions of people who made photographs in different places. The positions of these people between the moments of taking the photographs are unknown and cannot be estimated by means of interpolation.

Our generalization tool allows the user to choose whether to apply interpolation or not, on the basis of user's knowledge about the nature of the data.

Irrespective of the use of interpolation, the tool finds, for each trajectory, a sequence of cells $\{c_1, c_2, \ldots, c_n\}$. A cell may appear in this sequence more than once, but there is always at least one other cell between two occurrences. For each cell c_i in the sequence, there is a corresponding segment of the trajectory $[p_1^i, p_2^i]$, where p_1^i is the first trajectory point inside c_i and p_2^i is the last trajectory point inside c_i. In particular, the segment inside the cell may consist of a single point, i.e., p_1^i and p_2^i may coincide.

Accordingly, each trajectory is represented by a sequence of visits $\{v_1, v_2, \ldots, v_n\}$ of the cells $\{c_1, c_2, \ldots, c_n\}$. A visit v_i is a spatiotemporal object $<c_i, t_{start}, t_{end}>$, where c_i is a cell, t_{start} is the starting time of the visit, and t_{end} is the ending time. The times t_{start} and t_{end} equal the temporal references of the points p_1^i and p_2^i, respectively. Besides, the following characteristics of each visit are computed:

- the estimated duration of staying inside the cell (i.e., the difference between t_{end} and t_{start});
- the estimated length of the internal path inside the cell, which is computed on the basis of the spatial coordinates of p_1^i, p_2^i, and intermediate points, if any;
- the average speed of the movement, which is the ratio between the path length and the duration of staying inside the cell.

At the same time, each trajectory is also represented by a sequence of *moves* $\{m_1, m_2, \ldots, m_{n-1}\}$, where a *move* m_i is a spatiotemporal object $<c_i, c_{i+1}, t_0, t_{fin}>$ describing the transition from cell c_i to cell c_{i+1}. Here, t_0 is the time moment when the move began and t_{fin} is the time moment when the move ended. We take t_0 to be the same as t_{end} of the visit v_i of the cell c_i; t_{fin} is the same as t_{start} of the visit v_{i+1} of the cell c_{i+1}. For each move, like for a visit, it is also possible to compute the duration, length, and average speed.

3.5 Aggregation of the Data

Hence, we have generated a dual representation of each trajectory: as a sequence of visits and as a sequence of moves. On this basis, we aggregate the data in two complementary ways. First, for each cell of the territory division, we aggregate the visits of this cell. We count the number of visits and compute the statistics of durations, path lengths, and average speeds (minimum, maximum, mean, median, etc.). Second, for each pair of cells (c_j, c_k), we aggregate the moves from c_j to c_k. Note that the pairs (c_j, c_k) and (c_k, c_j) are different. For each aggregate move, we count the number of elementary moves and compute the statistics of durations, lengths, and average speeds.

In both cases, the counts and statistics may be computed not only for the whole time span of the data, but also by arbitrarily defined time intervals. In the latter case, each cell and each aggregate move will be characterized by time series of aggregate attribute values. It is also possible to do the aggregation in a dynamic way, as described in [24]. The idea is that the aggregate attributes are recomputed in response to changes of any filters applied to the trajectories, including temporal, spatial, and attribute filter.

In our examples of aggregated representation of movement data, the aggregate moves are represented by arrows (vectors) with the widths proportional to the counts of the elementary moves. However, it is also possible to visualize any other attribute of the aggregate moves by means of vector widths and/or coloring. Besides, it is possible to visualize the statistics of the visits of the cells by means of coloring, symbols, or diagrams.

4 The Quality of the Generalization

4.1 Measuring the Quality

In the visual representation of aggregate moves by vectors, the starts and ends of the vectors are positioned near the generating points of the Voronoi cells. Each vector represents a certain set of trajectory segments. The generalization may be deemed good if each vector closely approximates the respective trajectory segments. This refers to the start and end points of the trajectory segments, but not to their geometric shapes, which are not taken into account in the aggregation. Hence, the quality of the generalization can be expressed in terms of the displacements, i.e., the distances between the start and end points of the vectors, and the start and end points of the respective trajectory segments: the smaller the displacements, the better the quality.

It is also possible to look at the quality of the generalization from a different perspective, taking into account the dual representation of the trajectories in the generalization. Each trajectory is transformed into a sequence of visits of some areas. In each area, there is at least one point of the trajectory. In the visualization of the aggregated data, the areas are represented by certain points, which serve as starts and ends of the vector symbols. The quality of the generalization can be measured in terms of the distances between the representative points of the areas and the original trajectory points inside the areas: the smaller the distances, the better the quality.

These two approaches to the assessment of the quality are equivalent, both being based on computing the distances between the representative points of the areas (i.e., the generating points of the Voronoi cells in our case) and the points of the trajectories contained in the areas. In a case when there are two or more points of a trajectory in one visit, the quality measure for this visit is derived from the distances of these points to the representative point by applying an appropriate statistical operator such as minimum, mean, or median. Our choice is the minimum distance. The number of trajectory points in an area is mainly the function of the frequency of position measurements. This factor is not very relevant to the generalization quality assessment. The use of the minimum distance reduces the dependence of the quality measure on the positioning frequency.

We shall call the quality measure for a single visit of an area *elementary displacement*. The *local quality* of the generalization in an area is expressed in terms of the *mean displacement* and *total displacement*, which are, respectively, the mean and sum of the elementary displacements. The use of only one measure would be insufficient. When an area contains only a few dispersed visits, the mean displacement is high. However, since the visits are few, the importance of this area from the perspective of the overall quality is low. This corresponds to the low value of the total displacement in this area. On the other hand, the total displacement in an area containing very many visits may be high due to the sheer number of visits while all elementary displacements may be quite low. In this case, the mean displacement is low, which indicates a high local quality.

The overall generalization quality for the whole territory may be assessed in terms of the summary statistics of the local quality measures: minimum, maximum, mean, median, quartiles, etc.

The local quality measures can be visualized and examined. An example is presented in Fig. 10. The map contains the areas (Voronoi cells) and the aggregate moves derived from the same subset of the Milan car trajectories, as shown in Fig. 1. The generalization has been done with $\mathrm{MaxRadius} = 2{,}500$ m. The local quality measures are visualized on the map by means of radial diagrams where the radii of the yellow segments are proportional to the total local displacements and the radii of the blue segments to the mean local displacements. The gray circles around the segments correspond to the maximum values of both measures. Fig. 14a shows an enlarged map fragment, which also includes the layer with the original trajectory lines. The trajectory lines are drawn with 10 percent opacity. As a result, the visible degree of darkness of the lines corresponds to the density of the trajectories: the more trajectories overlap, the darker the lines. It may be seen that the mean local displacements are low in the places

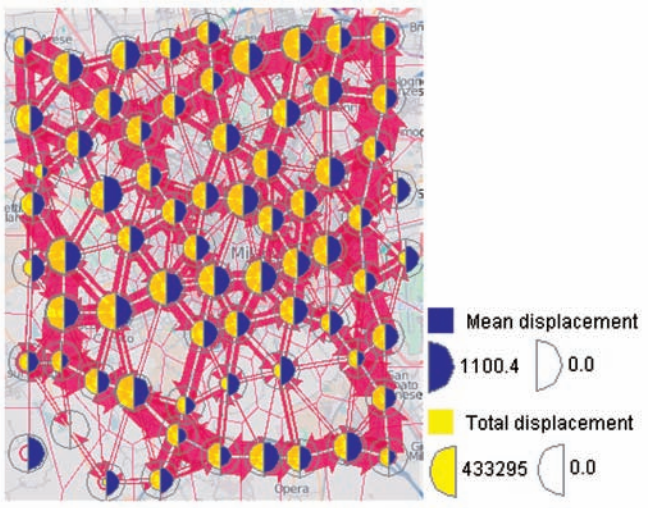

Fig. 10. The local quality measures of the generalization of the Milan car trajectories with $\mathrm{MaxRadius} = 2{,}500$ m.

where aggregate moves are close to dark lines, i.e., densely concentrated trajectories. In the places where aggregate moves substantially deviate from dense bundles of trajectory lines, the local displacements are higher. Hence, the numeric measures correspond quite well to the intuitive understanding of the quality of the generalization.

4.2 Improving the Quality

In some cases, it may be desirable to improve the quality of the generalization, for example, when it is necessary to produce a flow map for presentation purposes. In principle, the generalization quality may be substantially improved on the whole territory by decreasing the value of MaxRadius in Algorithms 2 and 3. As a result, the territory will be partitioned into smaller areas. The displacements in small areas are also small; hence, the quality is higher. However, smaller areas mean lower degree of generalization, which may be unsuitable for the required map scale. It is desirable to have a way to improve the generalization quality locally, i.e., only in the places where it is deemed necessary.

We suggest Algorithm 4 to refine the territory division in selected places. The refined division is used for reaggregating the data.

Algorithm 4. Local refinement of the territory division for improving the generalization quality

Given:

- territory division $\mathrm{D} = \{\mathrm{c_i}\}, 1 \le \mathrm{i} \le \mathrm{nCells}$, where each compartment $\mathrm{c_i}$ is a Voronoi cell having its representative (generating) point, which is denoted rPoint;
- $\mathrm{V} = \mathrm{V}(\mathrm{c_1}) \cup \mathrm{V}(\mathrm{c_2}) \cup \cdots \cup \mathrm{V}(\mathrm{c_{nCells}})$, where $\mathrm{V}(\mathrm{c_k})$ is the set of all visits of the cell $\mathrm{c_k}$;
- C - a selected subset of cells where quality improvement is needed;
- distThreshold - the distance threshold for treating a visit of a cell as being too far from its representative point. The threshold may be defined as a constant value or as a proportion of the mean, median, or maximum displacement in a cell.

Description of the algorithm:

1. **let** $\mathrm{P} = \varnothing$;
2. **for_each** $\mathrm{c} \in \mathrm{C}$
   /* extract the positions of the visits that are too far from the representative point of the cell c */
   for_each $\mathrm{v} \in \mathrm{V(c)}$
   let p = v.position;
   if *spatial_distance*(p,c.rPoint)>distThreshold **then**
   let $\mathrm{P} = \mathrm{P} \cup \{\mathrm{p}\}$;
   end_if;
   end_for_each;
   end_for_each;
3. compute R = get_spatial_groups(P);
   /* the extracted points are spatially grouped using Algorithms 2 and 3 */
4. **let** $\mathrm{GP} = \varnothing$; /* this will be a set of generating points for a new Voronoi tessellation */
5. **for_each** $\mathrm{cl} \in \mathrm{R}$ **let** $\mathrm{GP} = \mathrm{GP} \cup \{\mathrm{cl.centroid}\}$;
   end_for_each;
6. **for_each** $\mathrm{c} \in \mathrm{C}$
   /* extract the positions of the visits that are closer to the representative point of the cell c than to any generating point in GP */
   let $\mathrm{CP} = \varnothing$;
   for_each $\mathrm{v} \in \mathrm{V(c)}$
   let p = v.position;
   if $\forall \mathrm{g} \in \mathrm{GP}$
   spatial_distance(p,g) > *spatial_distance*
   (p,c.rPoint)
   then
   let $\mathrm{CP} = \mathrm{CP} \cup \{\mathrm{p}\}$;
   end_if;
   end_for_each;
   /* the centroid of the extracted set of points will be used as a generating point for a new Voronoi tessellation instead of the old generating point c.rPoint */
   compute q = *get_centroid*(CP);
   let $\mathrm{GP} = \mathrm{GP} \cup \mathrm{q}$;
   end_for_each;
7. /* combine the new generating points with the representative points of the cells that were not selected for the quality improvement */
   for_each $\mathrm{c_i} \in \mathrm{D}$
   if $\mathrm{c_i} \notin \mathrm{C}$ **then let** $\mathrm{GP} = \mathrm{GP} \cup \{\mathrm{c_i}.rPoint\}$; **end_if**;
   end_for_each;
8. compute $D' = get\_Voronoi\_tessellation(\mathrm{GP})$;
9. **return** D';

We have implemented two procedures for improving the generalization quality, automated and interactive. The automated procedure works in an iterative way. In each step, it selects the cells where the values of the total and mean displacement exceed the user-specified thresholds, applies Algorithm 4, and reaggregates the data using the new territory division. The thresholds for selecting the cells are specified as percentages of the current maximum (see Fig. 11). The user also specifies the maximum number of cells to select in each step and the termination conditions, which include the minimum quality improvement in comparison to the previous state, the maximum number of

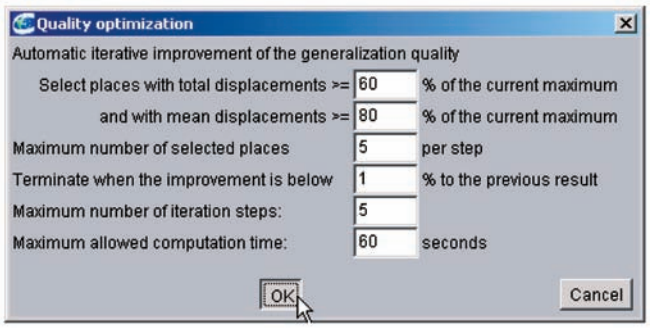

Fig. 11. The dialog to start the automated procedure for improving the generalization quality.

iteration steps, and the maximum allowed computation time. The result of the automated procedure is two map layers: one with the refined territory division (Voronoi cells) and the other with the corresponding aggregate moves. The procedure also produces a report in the form of table, which shows how many steps have been made, how many areas added, the statistics of the quality measures for the resulting aggregations, and the percentages of improvement in comparison to the previous states. For example, from the report shown in Fig. 12, we can learn that three iteration steps were made, but the third step was not successful since only a slight improvement was gained in the overall mean displacement, while the overall total displacement increased. Therefore, the result of the second step has been taken as final.

Fig. 13 demonstrates the result of the automatic quality improvement for the generalization presented in Fig. 10. The parameters shown in Fig. 11 have been used. The changes have been made in the western part of the inner city. This area is enlarged in Fig. 14, where the new generalization can be compared with the original one in more detail. In the new generalization, the aggregate moves are closer to the dense bundles of trajectory lines than in the original generalization. At the same time, the segments in the diagrams representing the local quality measures are smaller, i.e., the values are lower. Hence, the improvement of the numeric quality indicators corresponds to the perceived improvement of the quality.

In the interactive procedure, the areas where the generalization quality must be improved are explicitly selected by the user. The system supports the selection by automatic visualization of the local quality measures and providing tools for interactive filtering (dynamic query) and marking areas on the map. After the user has selected one or more areas either by filtering or marking, the system

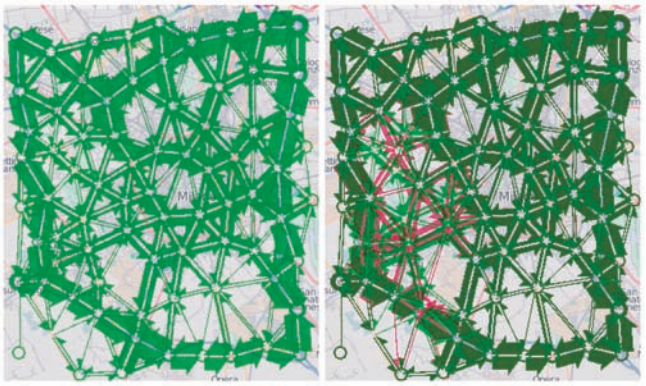

Fig. 13. (a) The quality of the generalization presented in Fig. 10 has been improved by means of the automated procedure. (b) The result of the improvement is overlaid on the original generalization, to see where the differences are.

applies Algorithm 4, reaggregates the data, and visualizes the results: the modified map layers, the statistics of the quality measures, and the percentages of improvement. The user has the choice to continue the procedure by selecting other areas for quality improvement, to finish the process, or to return to the previous state if the results are not satisfactory.

5 Applications of the Generalization Method

5.1 Summarization of Clusters of Trajectories

One of the big problems in using clustering techniques for analysis of large sets of trajectories is how to visualize clusters of trajectories so as to have an overview of all clusters obtained and to be able to compare different clusters. This is a problem because trajectories are not disjoint in space. They may intersect and partly overlap, and so do the clusters. Hence, there is no way to show all clusters in one map in a comprehensible way. A suitable approach is to generate multiple small maps, each presenting a single cluster. Since the maps have to be small, the clusters need to be shown in a highly generalized manner. The generalization method presented in this paper is well suited to this purpose. An example is shown in Fig. 15.

There are two possibilities in applying the method to clusters of trajectories. One is to do the tessellation of the territory once using the trajectories from all clusters. Then, the aggregation is done separately for each cluster. The other possibility is to do also the tessellation separately for each cluster. This is quite feasible even for a large number of

Table view: Iterative automatic refinement of Aggregated moves from Trajectories (angle 30; stop time 300; radius 100-2500)

	Iteration step N	N of areas	Max of mean local displacements	Max of total local displacements	Overall mean displacement	Overall total displacement	Improvement max mean %	Improvement max total %	Improvement overall mean %	Improvement overall total %
0	0	101	1100.38	433295.31	549.09	15881296.00	0.00	0.00	0.00	0.00
1	1	104	877.92	413134.16	534.21	15539742.00	20.22	4.65	2.71	2.15
2	2	108	877.92	413134.16	510.56	14978827.00	0.00	0.00	4.43	3.61
3	3	109	877.92	413134.16	508.88	14997697.00	0.00	0.00	0.33	-0.13
4	2	108	877.92	413134.16	510.56	14978827.00	20.22	4.65	7.02	5.68

Fig. 12. A report about the work of the automated procedure for the generalization quality improvement.

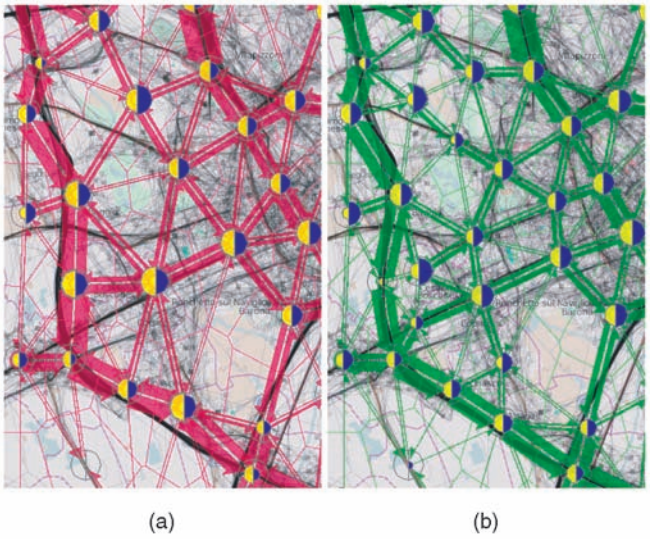

Fig. 14. The part of the territory where the changes have been made by the automated procedure for generalization quality improvement is enlarged. (a) The original generalization (see Fig. 10). (b) The modified generalization (see Fig. 13). The aggregate moves and the diagrams representing the local quality measures are overlaid on the original trajectory lines shown in black with 10 percent opacity.

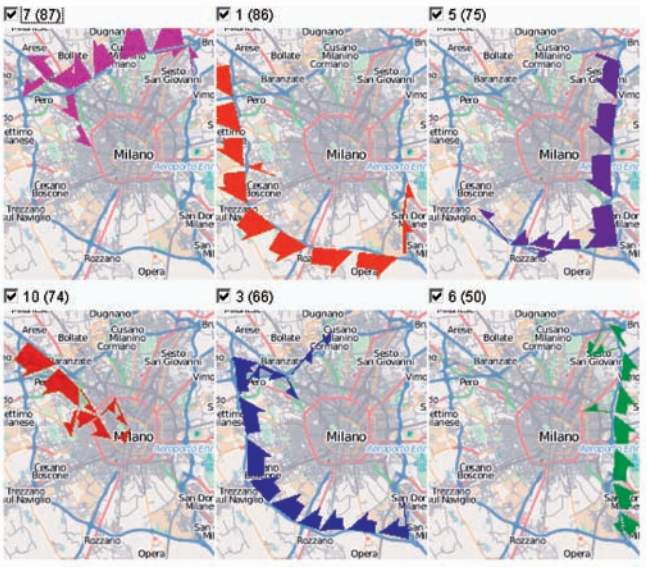

Fig. 15. A fragment of a display with summarized representations of clusters of trajectories. Each cluster consists of trajectories with similar routes.

clusters because the method works very fast. Thus, in Fig. 15, both the tessellation and the aggregation have been done individually for each cluster. In this case, the quality of the generalization is higher than when a common tessellation is used for all clusters.

5.2 Summarization of a Large Database

A set of movement data may be too big for loading into RAM. Still, it is possible to get an overview of the data in a summarized form. For this purpose, we first extract a sample of trajectories from the database. The size of the sample must be suitable for loading in the main memory. From this sample, we extract the characteristic points of the trajectories and divide the territory according to the spatial distribution of these points. Assuming that the sampling is done sufficiently well, i.e., the statistical and spatial distribution properties of the whole data set are preserved in a sample, we can use the territory division so obtained to summarize the data not only from the sample, but also from the whole database. To do this, we load the data in the main memory by suitable portions and do the summarization incrementally: the data from each new portion are added to the previously summarized data. After a portion is processed, it can be removed from the main memory, and the next portion can be loaded instead. The results of the summarization do not depend on how the database is divided into portions because the division of the territory does not change at this stage, but only the statistics of the visits of the cells and moves between them are updated.

Incremental summarization of a large database can also be done in combination with clustering. We first find clusters in a subset of trajectories and generate a classifier, which can attach new trajectories to the existing clusters on the basis of their similarity to the trajectories of the clusters [1]. For each cluster, we generate a suitable division of the territory. Then, we load the trajectories from the database into RAM by portions. The classifier compares each trajectory to the definitions of the clusters. If it finds a cluster in which this trajectory fits, the data from the trajectory are summed up with the previously summarized data for this cluster. At the end of the process, we obtain a display, like in Fig. 15, representing in a summarized form the clusters of trajectories existing in the whole database.

5.3 Multiscale Analysis

By varying the parameter MaxRadius in our method, we can divide the territory into larger or smaller compartments. This allows us to explore movement data at multiple spatial scales. This is especially appropriate in a case when the data represent movements differing very much in their spatial extent, such as movements within cities or small areas and movements between the cities or areas. An example is the data about the georeferenced photographs published by various people in Panoramio (www.panoramio.com) and flickr (www.flickr.com). The photographs are linked to the places where they were taken and supplied with the dates and times of the shots. The positions of the photographs made by the same person can be considered as a trajectory of this person in the geographical space. We have investigated the data about the subset of 590,000 Panoramio photographs referring to the territory of Germany and made in the period from January 1, 2005 to March 30, 2009 (the data have been acquired and prepared by Kisilevich, Univ. Konstanz). We found that people typically made multiple photographs of different objects and places in one city or tourist area. The data about the positions of these photographs represent the movements of the photographers within the city or area. Besides, many of the Panoramio users visited different cities and made photographs there. Hence, apart from the small-scale movements within cities/areas, there are also large-scale movements between the cities/areas.

Our summarization method allows us to extract and explore movements at different spatial scales. Thus, Fig. 16a displays the major flows of the Panoramio users between the cities and large areas in Germany. This flow map was produced with $\mathrm{MaxRadius} = 100\,\mathrm{km}$. Only the flows aggregating 100 or more moves are visible. In Fig. 16b, the map with the flows inside the city of Düsseldorf has been generated with $\mathrm{MaxRadius} = 1\,\mathrm{km}$ (only the flows

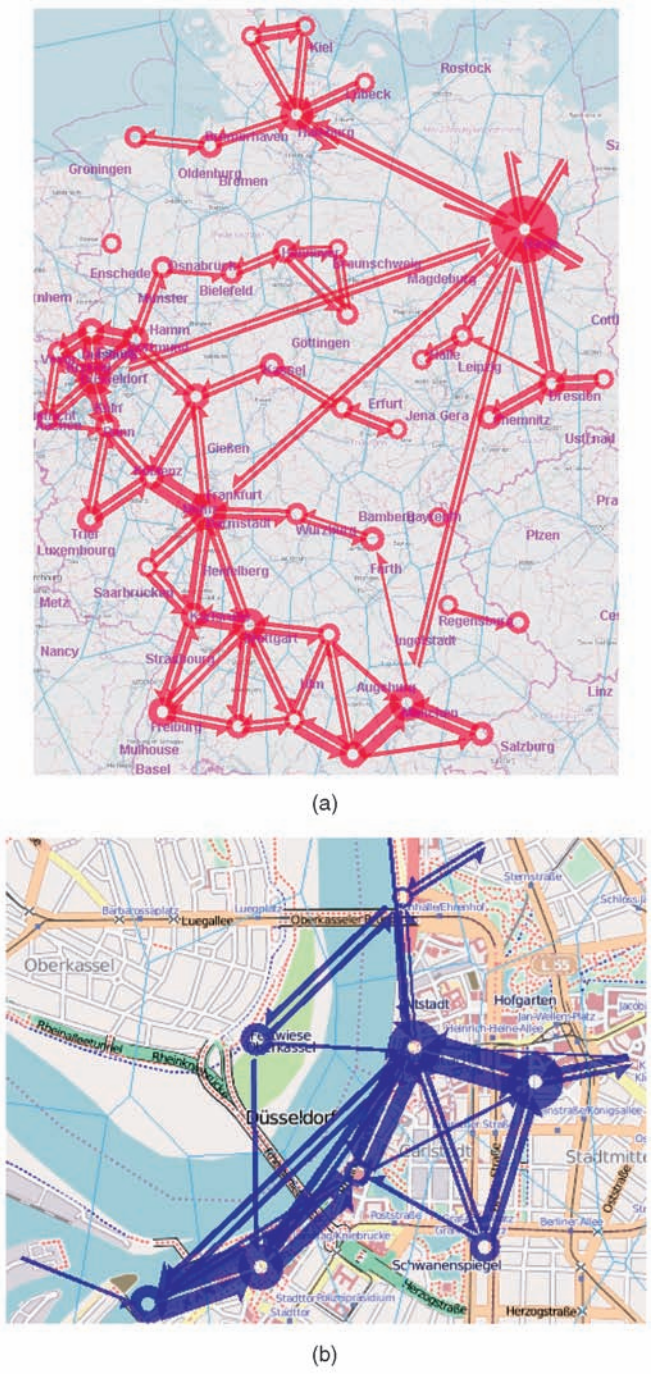

Fig. 16. Generalization of the data about Panoramio photos at two different scales.

standing for at least five moves are visible). Note that, in this case, we do not apply interpolation when consecutive points of a trajectory lie in different nonadjoining cells (see Section 3.4). The reason is that the position records are temporally very sparse, and it would be wrong to make any assumptions about the paths of the photographers between the places where they took the photographs.

5.4 Anonymization of Movement Data

The possibility to collect, store, disseminate, and analyze data about movements of people raises very serious privacy concerns, given the sensitivity of the information about personal positions [13]. Therefore, an important problem is how to anonymize movement data so that no information about any particular person can be disclosed, but the data are still suitable for analysis and can be a source of useful information about the mobility of the population as a whole or about large subsets of the population. One of the possible approaches to hide identifiable personal information is generalization of movement data by replacing exact positions in the trajectories by approximate positions, i.e., points by areas. Our method for territory division can be used for generating suitable areas. However, some extensions are required. Thus, it is necessary to ensure that each area contains positions from the trajectories of a sufficient number of different people. The desired minimum number of different people would be a parameter; let us call it *anonymity threshold*. Hence, after having generated a version of the territory division, the system must count the number of different people whose positions are contained in each cell and check whether this number reaches the specified anonymity threshold. The cells where the threshold is not reached must be enlarged to include positions of more people. This can be done, in principle, by merging these cells with neighboring cells. A disadvantage is that the resulting areas will not be convex, which brings inconveniences for the visualization and analysis. Another approach, which may be more suitable, is to generate a new Voronoi tessellation after excluding the generating points of the cells where the anonymity threshold is not reached.

An important property of this method for protecting personal data is that the resulting transformed data are suitable for various kinds of analysis. Thus, it is possible to analyze the flows between the areas and statistics of the visits of the areas. One may also analyze the statistics of the travel times between different pairs of areas, not only neighboring. Frequently occurring sequences of visited areas can be discovered by means of data mining techniques. It is also possible to apply cluster analysis to the modified trajectories. Thus, we have performed several experiments with clustering of the original car trajectories from Milan and generalized versions of these trajectories using the generic density-based clustering algorithm OPTICS with a suitable distance function [1], [24]. We found that the results of clustering the original and the generalized trajectories are very similar when the distance threshold (the parameter of the clustering algorithm) for the generalized trajectories is about one half of the distance threshold for the original trajectories. This allows us to believe that the idea has a good potential. However, further investigations are required for checking whether any risks to personal privacy are, indeed, precluded when trajectories are anonymized in this way.

6 Related Works

Fredrikson et al. [12] described the use of spatial, temporal, and categorical aggregation for visualization of discrete events such as traffic accidents. Movement data are often aggregated in the same ways [9], [11], [22], [31], i.e., the position records of the data are treated as independent discrete events. Although this approach can be suitable for certain analysis tasks [3], it disregards the very essence of movement as change of spatial position. Willems et al. [29]

build a density surface from movement data, taking into account the spatial and temporal dependencies between position records. However, such a surface does not convey information about the movement directions.

Movement data can also be aggregated in the form of vector fields (e.g., [7]). The underlying area is partitioned by means of a regular grid. In each grid cell, a vector is built with the angle corresponding to the prevailing movement direction and length and width proportional to the average speed and the amount of movement, respectively. This approach, however, does not work well when the movement directions inside the cells vary greatly and no dominant directions exist. This is the case of city traffic, for instance.

There are also works where movement data are considered as moves between predefined places. Aggregated moves are visualized by means of flow maps [26], [27] and transition matrices [15], where the rows and columns correspond to the places and symbols in the cells or cell coloring or shading encode aggregated values, in particular, the counts of the moves. Guo et al. [15] also use a multimap display, where each map represents the aggregated moves from/to a single place by colors or shades of the places.

Our paper suggests a method for territory partitioning, which can be used to define suitable places for the aggregation of movement data when there are no predefined places. The partitioning is done on the basis of the spatial distribution of points extracted from movement data. In geographical research, the procedure of building regions by grouping spatial objects is called regionalization [16]. The approaches are based on various clustering techniques [4]. These approaches are computationally complex as they account not only for the spatial neighborhood of the objects, but also for their similarity in terms of nonspatial attributes. For this reason, they do not scale to very large sets of objects. Such complex algorithms are, on the one hand, not necessary for our purposes since we do not use any nonspatial attributes of the points. On the other hand, these algorithms are not suitable for us since we have to deal with very large numbers of points.

Guo [14] applies a graph-based clustering algorithm to aggregate about 200,000 different locations defined in synthetic movement data into a smaller number of larger places. The algorithm is suited to a specific property of this data set: the locations are fixed and can appear in trajectories of different moving agents and/or reappear in the trajectory of the same agent. On this basis, Guo builds a spatial interaction graph where two locations are linked if they share at least one visitor. The algorithm divides this graph into subgraphs containing approximately equal numbers of strongly connected locations. This approach is not directly applicable to real movement data. Due to temporal gaps and unavoidable errors in measuring positions, it cannot be guaranteed that the recorded tracks of two or more moving agents who visited the same point in space will contain position records with exact coordinates of this point. Hence, the links between all locations are hard to reconstruct. Our method for grouping points, which is based on computing distances between the points and densities of the points, is suitable for real movement data. Our method also differs from graph-based methods in generating regions of desired size. The regions resulting from graph-based methods are small in dense areas and large in areas where the original locations are sparsely distributed.

Cui et al. [6] argue for the necessity of assessing the quality of data abstraction used in visualization and suggest two possible quality measures. The histogram difference measure (HDM) is applicable to data abstraction by means of sampling. It compares the frequency distributions of attribute values in a sample and in the whole data set. For the nearest neighbor measure (NNM), it is assumed that each record in the original data set has a nearest neighbor (the most similar object) in the abstracted data set, called its representative. The NNM is the normalized average of the distances between every record of the original data set to its representative. There is a similarity between the NNM and our quality measures. In our case, the positions from the original data are represented by the generating points of the Voronoi cells. We measure the distances between the original positions and their representatives. The local quality measures are the mean and sum of these distances computed for each cell. The global quality measures are the mean and sum of the distances computed for the whole set of positions.

Cui et al. also argue that information about data abstraction quality should be explicitly presented to analysts, and that the latter should have a possibility to adjust the data abstraction level by trading off the accuracy of representing the data and the degree of visual clutter. Our system visualizes the quality measures and provides tools for adjusting the data abstraction level in spatial aggregation of movement data. It is possible to increase or decrease the abstraction level on the whole territory and to do local adjustments in selected parts of the territory.

7 Conclusion

We have described an approach to spatial generalization and aggregation of movement data. In this approach, the space is divided into compartments, the trajectories are transformed into moves between the compartments, and the moves with common origins and common destinations are aggregated. The results can be visualized by means of a flow map or transition matrix. The major part of the contribution is a method for space partitioning based on spatial grouping of characteristic points extracted from the trajectories. We suggest a simple and fast algorithm for spatial grouping of points. The desired spatial extent of a group is a parameter of the algorithm. Through this parameter, the user can control the overall level of data abstraction. We also suggest a set of suitable local and global measures of the data abstraction quality and techniques supporting local adjustments of the quality and abstraction level in selected parts of the territory.

We have outlined possible applications of our generalization method. We have tested the method on essentially different examples of real movement data, including trajectories of vehicles, pedestrians, animals (roe deer and white storks), and data consisting of positions of photographs published in the Web. In all cases, the method allowed us to obtain interpretable data abstractions. The method is very fast and therefore well suited to the use in interactive visual exploration of movement data. At the same time, the method can be used for creating legible flow maps for presentation purposes.

ACKNOWLEDGMENTS

The work has been supported by the DFG—Deutsche Forschungsgemeinschaft (German Research Foundation) within the research project ViAMoD—Visual Spatiotemporal Pattern Analysis of Movement and Event Data.

REFERENCES

[1] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti, "Interactive Visual Clustering of Large Collections of Trajectories," *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST),* pp. 3-10, 2009.

[2] N. Andrienko and G. Andrienko, "Designing Visual Analytics Methods for Massive Collections of Movement Data," *Cartographica,* vol. 42, no. 2, pp. 117-138, 2007.

[3] G. Andrienko and N. Andrienko, "Spatio-Temporal Aggregation for Visual Analysis of Movements," *Proc. IEEE Symp. Visual Analytics Science and Technology (VAST '08),* pp. 51-58. 2008.

[4] R.M. Assunção, M.C. Neves, G. Câmara, and C.D.C. Freitas, "Efficient Regionalization Techniques for Socio-Economic Geographical Units Using Minimum Spanning Trees," *Int'l J. Geographical Information Science,* vol. 20, no. 7, pp. 797-811, 2006.

[5] L.P. Chew, Java Code for Delaunay Triangulation, http://www.cs.cornell.edu/Info/People/chew/Delaunay.html, 2009.

[6] Q. Cui, M.O. Ward, E. Rundensteiner, and J. Yang, "Measuring Data Abstraction Quality in Multiresolution Visualizations," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 709-716, Sept./Oct. 2006.

[7] D.R. Brillinger, H.K. Preisler, A.A. Ager, and J.G. Kie, "An Exploratory Data Analysis (EDA) of the Paths of Moving Animals," *J. Statistical Planning and Inference,* vol. 122, no. 2, pp. 43-63, 2004.

[8] I. Drecki and P. Forer, "Tourism in New Zealand—International Visitors on the Move (A1 Cartographic Plate)," Tourism, Recreation Research and Education Center (TRREC), Lincoln Univ., 2000.

[9] J.A. Dykes and D.M. Mountain, "Seeking Structure in Records of Spatio-Temporal Behavior: Visualization Issues, Efforts and Applications," *Computational Statistics and Data Analysis,* vol. 43, pp. 581-603, 2003.

[10] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. Second Int'l Conf. Knowledge Discovery and Data Mining,* pp. 226-231, 1996.

[11] P. Forer and O. Huisman, "Space, Time and Sequencing: Substitution at the Physical/Virtual Interface," *Information, Place and Cyberspace: Issues in Accessibility,* D.G. Janelle and D.C. Hodge, eds., Springer-Verlag, pp. 73-90, 2000.

[12] A. Fredrikson, C. North, C. Plaisant, and B. Shneiderman, "Temporal, Geographical and Categorical Aggregations Viewed through Coordinated Displays: A Case Study with Highway Incident Data," *Proc. Workshop New Paradigms in Information Visualization and Manipulation,* pp. 26-34, Nov. 1999.

[13] *Mobility, Data Mining and Privacy—Geographic Knowledge Discovery,* F. Giannotti and D. Pedreschi, eds. Springer, 2007.

[14] D. Guo, "Visual Analytics of Spatial Interaction Patterns for Pandemic Decision Support," *Int'l J. Geographical Information Science,* vol. 21, no. 8, pp. 859-877, 2007.

[15] D. Guo, J. Chen, A.M. MacEachren, and K. Liao, "A Visual Inquiry System for Spatio-Temporal and Multivariate Patterns (VIS-STAMP)," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 6, pp. 1461-1474, Nov./Dec. 2006.

[16] P. Haggett, A.D. Cliff, and A. Frey, *Locational Analysis in Human Geography.* Arnold, 1977.

[17] J. Han and M. Kamber, "Data Mining," *Concepts and Techniques,* Morgan Kaufmann, 2006.

[18] T. Kapler and W. Wright, "GeoTime Information Visualization," *Information Visualization,* vol. 4, no. 2, pp. 136-146, 2005.

[19] M.-J. Kraak, "The Space-Time Cube Revisited from a Geovisualization Perspective," *Proc. 21st Int'l Cartographic Conf.,* pp. 1988-1995, Aug. 2003.

[20] M.-J. Kraak and F. Ormeling, *Cartography: Visualization of Spatial Data,* second ed., Pearson Education Limited, 2003.

[21] *Generalization of Geographic Information: Cartographic Modelling and Applications,* W.A. Mackaness, A. Ruas, and L.T. Sarjakoski, eds. Elsevier, 2007.

[22] D.M. Mountain, "Visualizing, Querying and Summarizing Individual Spatio-Temporal Behavior," *Exploring Geovisualization,* J.A. Dykes, M.J. Kraak, and A.M. MacEachren, eds., pp. 181-200, Elsevier, 2005.

[23] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd, "Flow Map Layout," *Proc. IEEE Symp. Information Visualization (InfoVis),* pp. 219-224, Oct. 2005.

[24] S. Rinzivillo, D. Pedreschi, M. Nanni, F. Giannotti, N. Andrienko, and G. Andrienko, "Visually-Driven Analysis of Movement Data by Progressive Clustering," *Information Visualization,* vol. 7, nos. 3/4, pp. 225-239, 2008.

[25] T.A. Slocum, R.B. McMaster, F.C. Kessler, and H.H. Howard, *Thematic Cartography and Geovisualization,* third ed. Pearson Prentice Hall, 2009.

[26] W. Tobler, "Experiments in Migration Mapping by Computer," *The Am. Cartographer,* vol. 14, no. 2, pp. 155-163, 1987.

[27] W. Tobler, Display and Analysis of Migration Tables, http://www.geog.ucsb.edu/~tobler/presentations/shows/A_Flow_talk.htm, 2005.

[28] E.R. Tufte, *The Visual Display of Quantitative Information.* Graphics Press, 1986.

[29] N. Willems, H. van de Wetering, and J.J. van Wijk, "Visualization of Vessel Movements," *Computer Graphics Forum,* vol. 28, no. 3, pp. 959-966, 2009.

[30] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques,* second ed. Morgan Kaufmann, 2005.

[31] J. Wood, A. Slingsby, and J. Dykes, "Using Treemaps for Variable Selection in Spatio-Temporal Visualization," *Information Visualization,* vol. 7, nos. 3/4, pp. 210-224, 2008.

Natalia Andrienko received the master's degree in computer science from Kiev State University in 1985 and the PhD degree in computer science from Moscow State University in 1993. She undertook research on knowledge-based systems at the Institute for Mathematics of Moldavian Academy of Sciences (Kishinev), then at the Institute for Mathematical Problems of Biology of Russian Academy of Sciences (Pushchino Research Center). Since 1997, she has held a research position at GMD, now Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS). Since 2007, she has been the lead scientist of the institute responsible for the visual analytics research. She authored the monograph *Exploratory Analysis of Spatial and Temporal Data* (Springer, December 2005), more than 40 peer-reviewed journal papers, more than 20 book chapters, and more than 100 papers in conference proceedings. Dr. Natalia Andrienko has been involved in numerous international research projects. Her research interests include geovisualization, information visualization with a focus on spatial and temporal data, visual analytics, interactive knowledge discovery and data mining, and spatial decision support and optimization.

Gennady Andrienko received the master's degree in computer science from Kiev State University in 1986 and the PhD in computer science from Moscow State University in 1992. He undertook research on knowledge-based systems at the Institute for Mathematics of Moldavian Academy of Sciences (Kishinev), then at the Institute for Mathematical Problems of Biology of Russian Academy of Sciences (Pushchino Research Center). Since 1997, he has held a research position at GMD, now Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS). Since 2007, he has been a lead scientist of the institute responsible for the visual analytics research. His research interests include geovisualization, information visualization with a focus on spatial and temporal data, visual analytics, interactive knowledge discovery and data mining, and spatial decision support and optimization. Since 2007, he has been chairing the ICA Commission on GeoVisualization. He organized several scientific events on visual analytics, geovisualization, and visual data mining, and edited several special issues of journals and proceedings volumes.