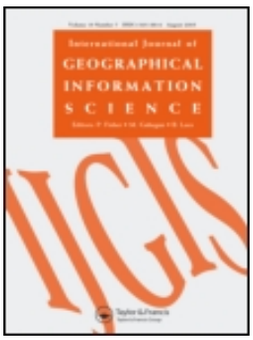

International Journal of Geographical Information Science

ISSN: 1365-8816 (Print) 1362-3087 (Online) Journal homepage: https://www.tandfonline.com/loi/tgis20

Space–time density of trajectories: exploring spatio-temporal patterns in movement data

Urška Demšar & Kirsi Virrantaus

To cite this article: Urška Demšar & Kirsi Virrantaus (2010) Space–time density of trajectories: exploring spatio-temporal patterns in movement data, International Journal of Geographical Information Science, 24:10, 1527-1542, DOI: 10.1080/13658816.2010.511223

To link to this article: https://doi.org/10.1080/13658816.2010.511223

Published online: 06 Oct 2010.

Submit your article to this journal

Article views: 3373

View related articles

Citing articles: 27 View citing articles

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tgis20

International Journal of Geographical Information Science
Vol. 24, No. 10, October 2010, 1527–1542

Taylor & Francis
Taylor & Francis Group

Space–time density of trajectories: exploring spatio-temporal patterns in movement data

Urška Demšar[a]* and Kirsi Virrantaus[b]

[a]*National Centre for Geocomputation, National University of Ireland Maynooth, Maynooth, Ireland;* [b]*Geoinformatics and Cartography, Department of Surveying, School of Science and Technology, Aalto University, Espoo, Finland*

(*Received 27 April 2010; final version received 9 July 2010*)

Modern positioning and identification technologies enable tracking of almost any type of moving object. A remarkable amount of new trajectory data is thus available for the analysis of various phenomena. In cartography, a typical way to visualise and explore such data is to use a space–time cube, where trajectories are shown as 3D polylines through space and time. With increasingly large movement datasets becoming available, this type of display quickly becomes cluttered and unclear. In this article, we introduce the concept of 3D space–time density of trajectories to solve the problem of cluttering in the space–time cube. The space–time density is a generalisation of standard 2D kernel density around 2D point data into 3D density around 3D polyline data (i.e. trajectories). We present the algorithm for space–time density, test it on simulated data, show some basic visualisations of the resulting density volume and observe particular types of spatio-temporal patterns in the density that are specific to trajectory data. We also present an application to real-time movement data, that is, vessel movement trajectories acquired using the Automatic Identification System (AIS) equipment on ships in the Gulf of Finland. Finally, we consider the wider ramifications to spatial analysis of using this novel type of spatio-temporal visualisation.

Keywords: space–time density; space–time cube; spatio-temporal visual data exploration; movement data

1. Introduction

Data about movements of objects are often collected as trajectories in space and time, that is, the movement of each object is recorded as a series of geographic locations with respective time stamps, when the object moves in the basic 3D framework of our physical world, defined by geographic space and time. Exploring and analysing large datasets of movement data is important in many disciplines, such as human geography, spatial epidemiology, location-based services, transportation modelling, monitoring of maritime and air traffic, urban planning, tourism and wildlife ecology. However, to be able to perform statistical or computational analysis or derive mathematical models of movement, it is first necessary to visualise these data and visually explore their properties. This problem is non-trivial, especially when large amounts of data are present, since movement trajectories often overlap.

*Corresponding author. Email: urska.demsar@nuim.ie

ISSN 1365-8816 print/ISSN 1362-3087 online
© 2010 Taylor & Francis
DOI: 10.1080/13658816.2010.511223
http://www.informaworld.com

In cartographic generalisation, the problem of overlapping data elements is traditionally solved by applying an operator of *cartographic aggregation* (Slocum et al. 2010). This operator combines several data elements (typically point objects) based on their spatial interrelationships into a single higher-dimensional spatial unit (typically an area). The result is a generalised and abstracted representation of a set of close-by objects, which would otherwise obstruct each other in the map display. Although some information might be lost in the process, the generalised pattern is easier to perceive. Analogously, aggregation of movement data can be used to produce spatio-temporal visualisations that display movement patterns in a clearer way (Andrienko and Andrienko 2010a).

There are three main approaches for aggregation of movement data (Andrienko and Andrienko 2010a). The first approach is based on spatial (S-), temporal (T-) or attribute (A-) proximity: that is, the space/time or attribute space are divided into compartments, into which the trajectories (viewed as a set of discrete movement events, i.e. geographic locations with respective time stamps) are projected. For example, 2D kernel density of polylines in a regular grid is a S-aggregation. Such aggregation approaches are very common and are often seen in combinations, for example, S × T- or S × T × A-aggregations (Dykes and Mountain 2003, Laube et al. 2005, Mountain 2005, Slingsby et al. 2008, Zhao et al. 2008).

The second approach is also trajectory-based: here trajectories are aggregated in their entirety based on their similarity in geographic, temporal or attribute space (or a combination thereof) (Andrienko and Andrienko 2010a) – this is route-based (R-) aggregation, often performed by clustering in the data space or in an abstract projection thereof (Skupin and Hagelman 2005, Rinzivillo et al. 2008, Wilson 2008).

The third approach aggregates movement data based on their origin and destination and ignores the route between these two spatial locations, so that the movement is seen as a vector between the two locations, not as a set of recorded positions on a trajectory (Andrienko and Andrienko 2010a). This type of aggregation is either S × S-based or S × S × T × T-based (Guo 2007, Andrienko and Andrienko 2010b).

Which type of aggregation is most suited for visualisation of a particular movement dataset depends on the conceptual model of movement and respective analysis task. Andrienko and Andrienko (2010a) introduce two types of movement conceptualisation. Given a set of moving objects (and trajectory data that describe their movement), movement can either be viewed from a situation-oriented point of view or from a trajectory-oriented point of view. In a *situation-oriented point of view*, we are interested in the spatial distribution of objects at each point in time – this can be illustrated as a series of snapshot maps, each for a different moment in time, where there is a number of point objects with respective motion vectors (direction, speed) on each map and the movement is only shown for the next time step. In contrast to this, a *trajectory-oriented point of view* looks at each separate moving object throughout the entire period – that is, at the entire trajectory of each individual object.

In each of these two views of movement, we can further distinguish between two different types of analysis tasks (Andrienko and Andrienko 2010a): we see the movement either as a property within space (*the space-centred approach*) or as a property of an individual moving entity (*the entity-based approach*). For this article, it is the space-centred approach with respective specific analysis tasks and aggregation methods that is of interest. Tasks of the *situation-oriented space-centred approach* include exploration of space use, accessibility and permeability. This approach uses S-, T-, A-aggregations and their combinations (the first aggregation approach above). The *trajectory-oriented space-centred approach* on the other hand looks at space connectivity, major flows, routes and use of pathways and involves aggregations from the third group (S × S-, S × S × T × T-). In this

article, we look at a commonly used cartographic method, a space–time cube, which represents movement from a trajectory-oriented space-centred point of view. We introduce a S × T-aggregation (the 3D kernel density) of trajectories in the space–time cube to improve the visualisation.

A *space–time cube* (or a space–time aquarium) is a concept from time geography first introduced by Hägerstrand in the 1960s. It is a graphic representation of the idea that space and time are inseparable. This concept is represented as a cube with a geographic map on the *x*–*y* plane and the time on the *z*-axis. Trajectories of objects are shown as 3D polylines in the cube. Space–time cubes have been widely used for visualisation and visual exploration of trajectories, particularly in time geography for human activity patterns (Kwan 2000, Kraak 2003, Kapler and Wright 2005, Kraak 2008). As a visualisation method, the space–time cube conceptualises the trajectory-oriented view of movement, since each separate trajectory is shown. The tracks of the trajectories through the cube provide information on general flow and routes through space and illustrate how objects move from one location to another within space. It can therefore be considered as a tool for space-centred analysis. An important additional property of the space–time cube is that it visually illustrates continuity of movement in both space and time – that is, an observer perceives that the displayed trajectories are continuous (even though they are not continuously smooth, but usually linearly interpolated between discretely sampled location points), which gives an impression that the movement is continuous in both space and time.

With the increased availability of larger movement datasets, the space–time cube display becomes unsatisfactory, particularly when there are many trajectories present. The main issues are overprinting and cluttering, both of which obscure the view of and prevent reliable visual identification of spatio-temporal patterns. A common solution to this is that a space–time cube is linked to other visual and/or computational data mining methods in multiple displays (Andrienko *et al.* 2007, Crnovrsanin *et al.* 2008, Eccles *et al.* 2008, Kraak and Huisman 2009, Vrotsou *et al.* 2009, Zhao *et al.* 2008).

We suggest an alternative solution to the problem of the cluttered space–time cube by aggregating the trajectories via calculation of their 3D space–time density. This is a S × T-aggregation into a volumetric dataset through a regular division of geographic space and time into voxels. This aggregation preserves properties of the original space–time cube: the resulting density volume allows a trajectory-oriented space-centred analysis of movement and at the same time provides a visual illustration of continuity of movement in space and time.

Specifically, the 3D space–time density is a generalisation of the standard 2D kernel density where kernels are positioned on data points in 2D geographic space into a 3D density, where kernels are situated around each trajectory, that is, a 3D polyline, in 3D space. These kernels describe the level of presence of the trajectory in geographic space as well as time. A similar 3D space–time density has recently been used on 3D point data describing crime patterns (Brunsdon *et al.* 2007, Nakaya and Yano 2009), but to the best of our knowledge this is the first extension to 3D polyline data. In this article, we introduce the 3D density algorithm and show some basic visualisations of the density for simulated trajectories.

We also present a real-life application using vessel movement data from the Gulf of Finland. Most of the professional vessels – those that are larger than a certain specified size (in tons) – are equipped with the Automatic Identification System (AIS), which uses global satellite positioning to measure the location of the vessel at specified time intervals and records this information together with information about the ship. We suggest that the 3D space–time density calculated from AIS data collected over a long period could facilitate

identification of the spatio-temporal patterns in the behaviour of different types of vessels. The 2D kernel density of AIS vessel movement trajectories has been recently used for similar purposes for Rotterdam (Willems *et al.* 2009) and Helsinki (Legouge *et al.* 2010), but in contrast with our approach, these two studies consider vessel trajectories only in geographic space and not simultaneously in space and time as we do.

2. Space–time density of trajectories

This section presents our algorithm for calculation of the space–time density of trajectories and some visualisation possibilities for the resulting density volume.

2.1. Calculating the density – the algorithm

The algorithm for 3D space–time density is a generalisation of the widely used 2D kernel density on point data (Silverman 1986) into 3D kernel density on polyline data (trajectories). We calculate density for each trajectory separately. This density represents the influence of that particular trajectory to the nearest neighbourhood of the trajectory in both space and time. Then, separate densities are added up to obtain the total density for the entire dataset of trajectories.

The density around each trajectory (polyline) in space and time is calculated as a volume (i.e. a 3D raster or a 3D grid). The value is assigned to each voxel according to the distance of the central point of the respective voxel to the trajectory. This distance is normalised using the user-specified kernel size, so that voxels on the trajectory are assigned value 1 and the density value decreases linearly with distance from the trajectory and reaches 0 at the limit of the kernel. The kernel function is presently linear – other types of kernel functions will be considered later. In the final step, the densities for each separate trajectory are added up to a total combined density. This operation is done using 3D map algebra (i.e. voxel by voxel) and the sum is then normalised to the [0,1] range. In the following, we present a more detailed description of the algorithm. The pseudocode for the algorithm is as follows:

```
TotalDensity = 0;
for each trajectory
  TrajectoryDensity = 0;
  calculate KernelArea around the trajectory;
  for each voxel in the KernelArea
    calculate DistanceToTrajectory;
    TrajectoryDensity = normalised DistanceToTrajectory
  end
  TotalDensity = TotalDensity + TrajectoryDensity;
end
Normalise TotalDensity with number of trajectories;
```

In this procedure, TotalDensity and TrajectoryDensity are volumes with voxel size defined according to pre-specified geographic and temporal resolutions. Each trajectory is given as a list of points in 3D space, that is, a 3D polyline consisting of points $P_1 - P_2 - \cdots - P_n - P_{n+1} - \cdots - P_m$, where $P_n = (x_n, y_n, z_n)$, x_n and y_n are geographic coordinates and z_n represents the time when the object was in location (x_n, y_n). Although a trajectory of a moving object in real world is always a continuous 3D line, its representation for computational purposes has to be discretised in order to be stored and analysed in an

information system. The trajectory of an object is therefore represented as a collection of discrete 3D points with coordinates in geographic space and time, that is, a polyline as described above. The original continuous track is then normally approximated as a series of straight-line segments between sampled points (Laube *et al.* 2005) and we adopt this standard assumption for our algorithm. Not all trajectories necessarily consist of the same number of points (i.e. m can vary from trajectory to trajectory); therefore the KernelArea and DistanceToTrajectory functions read each trajectory sequentially, line segment after line segment.

KernelArea approximates a 3D buffer around a trajectory as a union of rectangular prisms, one around each line segment P_nP_{n+1} in the trajectory. This is a well-known shortcut from computational geometry (Schneider and Eberly 2003) to speed up the algorithm and ensure that distance calculations in the next step of the algorithm are only calculated for those voxels that are expected to fall within the pre-defined distance from the trajectory – this distance is given by the pre-defined kernel size. To build up the KernelArea, we calculate a prism around each segment of the polyline, so that the segment is located on the main diagonal of the prism, but the prism extends for a distance equal to the extent of the kernel in all directions (Figure 1). The KernelArea for the trajectory is a union of such prisms for all line segments of the polyline. Only voxels inside this area are considered in the next step of the calculation. The TrajectoryDensity of voxels outside KernelArea is set to 0.

The next step is to proceed through all voxels in the KernelArea, one by one, and for each voxel calculate the distance between the central point of the voxel (x_v, y_v, z_v) and the trajectory – DistanceToTrajectory. This is done by using a computational geometry algorithm for the distance between a 3D point and a 3D polyline (Schneider and Eberly 2003) in the following manner: go through all the line segments P_nP_{n+1} of the given trajectory and calculate the distance between the point (x_v, y_v, z_v) and the line segment P_nP_{n+1}. This is either the distance between the point (x_v, y_v, z_v) and the infinite line in 3D space through points P_n and P_{n+1}, or the distance between (x_v, y_v, z_v) and one of the points P_n or P_{n+1}, depending on the relative location of the point (x_v, y_v, z_v) to the line segment P_nP_{n+1}. The DistanceToTrajectory, that is distance between the point (x_v, y_v, z_v) and the polyline

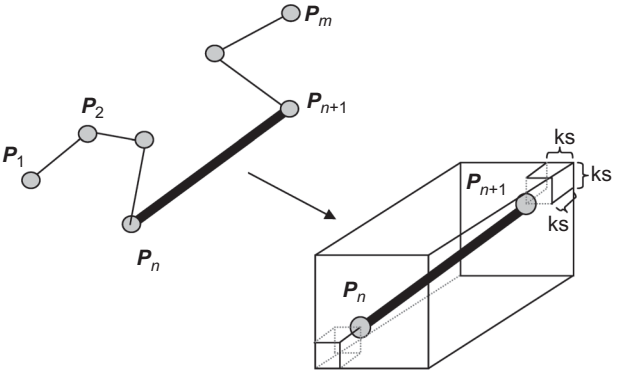

Figure 1. Construction of KernelArea for one line segment P_n–P_{n+1} in the polyline $P_1 - P_2 - \cdots - P_n - P_{n+1} - \cdots - P_m$. KernelArea for one line segment is a prism, such that the segment P_n–P_{n+1} is located on the main diagonal of the prism and the prism extends for kernel size (ks) from the segment in all directions. KernelArea for the entire polyline is a union of prisms for all line segments in the polyline.

$P_1 - P_2 - \cdots - P_n - P_{n+1} - \cdots - P_m$ is then calculated as the minimum of all distances from point (x_v, y_v, z_v) to line segment P_nP_{n+1} for $n = 1,\ldots, m–1$.

In the next step, the DistanceToTrajectory is used to calculate TrajectoryDensity at voxel (x_v, y_v, z_v). We normalise this distance using a linear function, so that TrajectoryDensity is equal to 1 on voxels on the trajectory, it then linearly decreases with distance from trajectory and reaches value 0 at distance from the trajectory that equals kernel size. All voxels in the KernelArea whose DistanceToTrajectory is larger than kernel size are also assigned value of 0. All voxels outside KernelArea have not had their TrajectoryDensity changed from the initial 0. The result is a 3D buffer of voxels around the trajectory, with value 1 on trajectory, value 0 on the surface of the buffer and outside the buffer and linearly decreasing values between the centre of the buffer (on the trajectory) and the surface. This buffer is a 3D analogy of the 2D kernel function in standard 2D kernel density. Since this is the first implementation of this algorithm, we used the simplest kernel function – a linear kernel (i.e. a linear decrease of density value with distance from trajectory as described above). Other kernel functions could be used – in 2D kernel density calculation, the most commonly used kernels are Gaussian and quadratic functions (O'Sullivan and Unwin 2003). These could be generalised into 3D by applying an appropriate normalisation function (Gaussian or quadratic) in the calculation of TrajectoryDensity from DistanceToTrajectory.

Figure 2 shows TrajectoryDensities (i.e. 3D buffers as described above) for eight simulated trajectories, which we used to test the algorithm. Results are shown using

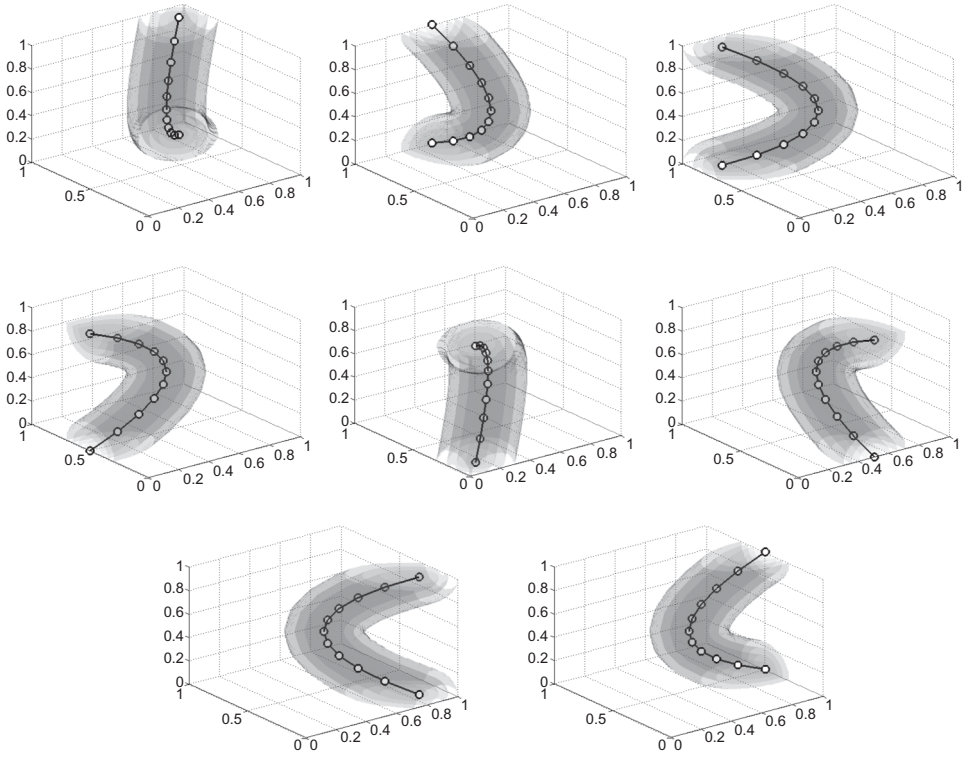

Figure 2. Space–time kernel densities around eight artificial trajectories. Each image shows density values around a separate trajectory, where density values are shown with five isosurfaces, for density values 0, 0.2, 0.4, 0.6 and 0.8.

isosurfaces (we explore other possibilities for visualisation of density volumes in the next section), which are 2D surfaces in 3D space that connect all voxels with the same density value. In Figure 2 there are five isosurfaces for each trajectory, one for each of the density values 0, 0.2, 0.4, 0.6 and 0.8. All isosurfaces are shown in the same colour. Density value on each surface is visualised with transparency: the higher the density value, the less transparent the respective isosurface.

When the algorithm proceeds through all the trajectories, their corresponding densities are added up to TotalDensity – the space–time density for the entire set of trajectories. In the final step, the resulting TotalDensity volume is normalised with the number of trajectories so that the range of the total density is [0,1]. Figure 3 shows the TotalDensity for the eight

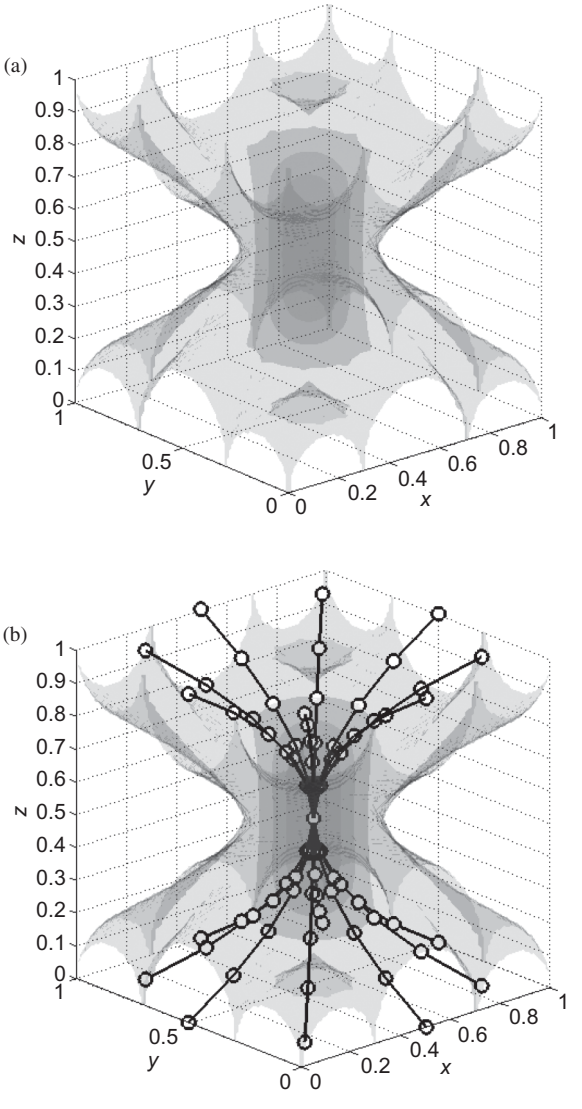

Figure 3. Combined space–time density for eight trajectories from Figure 1: (a) total density shown as five isosurfaces; (b) total density shown as five isosurfaces with superimposed simulated trajectories.

simulated trajectories: in Figure 3a the total density volume is shown with five isosurfaces for the same values as in Figure 2. In Figure 3b, trajectories are superimposed on the density volume.

The algorithm was implemented in Matlab.

2.2. Visualising the density volume

The space–time density of trajectories is a volume, that is, a 3D grid, where we have a value of a scalar field measured or calculated in the centre of each voxel. It is therefore not very easy to visualise on the two display dimensions of the computer screen. Volumetric data are common in medicine (data from scanning devices such as CT and MRI scanners and recently the 3D ultrasound) and in some scientific fields, such as computational fluid dynamics, geophysics and biomedical computing. Such data are measured in regular (or sometimes irregular) 3D grids, where measurements represent values of a scalar field in a 3D context space (Callahan *et al.* 2008, Lederberger *et al.* 2008). Volume visualisation is a major research field in information visualisation, particularly in medical visualisation – some recent examples include 3D ultrasound visualisations (Karadayi *et al.* 2009), visualisations of CT and MRI scans (Preim *et al.* 2009) and visual computing system for surgical training (Mühler *et al.* 2010). The three main approaches to volume visualisation are direct volume rendering, isosurfaces and volume slicing with clipping planes.

Direct volume rendering assigns optical properties (colour and transparency) to voxels according to the scalar field values. The volume is considered as a medium of semi-transparent material, which enables us to see inside the data. By changing the optical properties of voxels (i.e. they can emit, absorb or transmit light) and using different projections of the volume to the 2D display from different points of view, we can achieve different lighting effects that facilitate different insights into the data volume. Different display methods produce different visualisations: the most common methods are ray casting, texture slicing, shading and shear-warp displays (Callahan *et al.* 2008, de Pinto and Freitas 2008, Lederberger 2008).

Isosurfaces are implicitly defined 2D surfaces in the 3D scalar field such that all the points on the isosurface share the same scalar field value. These surfaces are calculated from the volume data for different values of the scalar field that the data represent. Surfaces are then displayed with different optical properties (colour, transparency) to give an impression of the distribution of one particular value inside the volume. They can be shown using an array of different display methods, such as ray-casting or shading (Hadwiger *et al.* 2005, Jang and Varetto 2009).

Volume slicing, *volume clipping* or visualisation with *clipping planes* show a cross-section of the volume which has been cut out from within the 3D space by either a plane or some other type of 2D surface. In this way, unimportant parts of the volume can be cut away to expose internal patterns and uncover important, but otherwise hidden, details in the dataset (Weiskopf *et al.* 2003). This methodology is particularly common in medical visualisation of sequential scans (CT and MRI data) (Preim *et al.* 2009, Mühler *et al.* 2010), to display a cross-section of the patient's body at a particular horizontal or vertical plane.

Volume visualisation is not very common in GIScience – the closest applications are in geophysics and geology, where volumetric data are very common. Some recent examples of these include the following: Hsieh *et al.* (2010) use all three methods (direct volume rendering, isosurface and volume slicing) to visually explore ground-motion data from an earthquake; Gao (2009) displays seismic volume data with direct volume rendering and

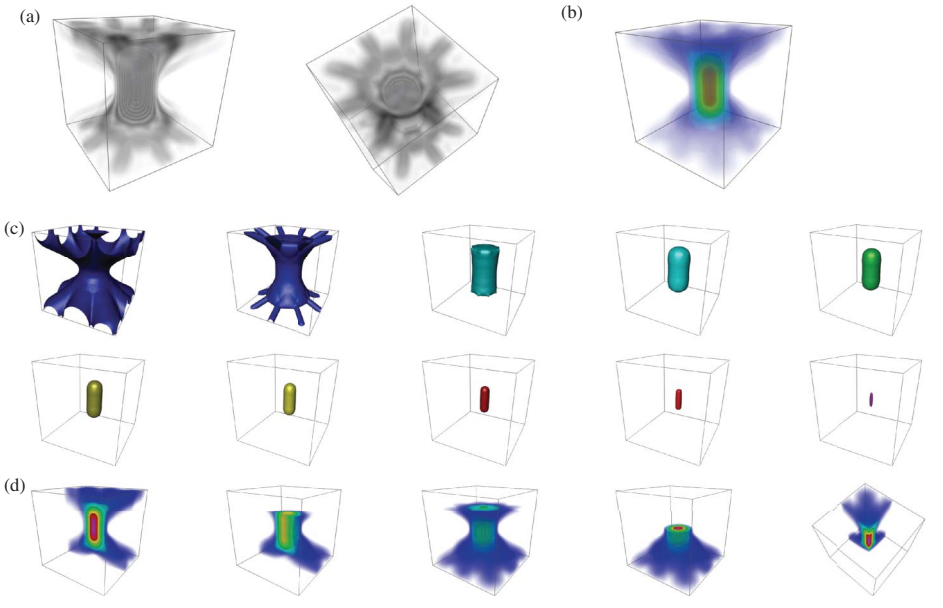

Figure 4. Visualising the space–time density volume of the simulated dataset by (a) volume rendering with zebra colour scheme, (b) volume rendering with rainbow colour scheme, (c) isosurfaces for density values 0, 0.1, . . . , 0.9 and (d) with clipping planes in different orientations.

vertical and stratal slicing; Chen *et al.* (2008) display 3D spherical mantle convection with direct volume rendering. A recent GIScience application are space–time accessibility volumes (Forer and Huisman 2000).

For our space–time density volume, we used all three volume visualisation methods listed above. Figure 4 illustrates these display methods on the space–time density of the simulated dataset: panels 4a and 4b show the density volume using direct volume rendering with two different colour schemes. Figure 4c shows isosurfaces for 10 different density values (values 0, 0.1, 0.2, . . . , 0.9), whereas in Figure 4d the density volume is sliced with clipping planes in 5 different directions. Some of these cross sections in Figure 4d are produced using two clipping planes in two different directions. We used Voxler software for these visualisations (produced by Golden Software), which is a specialised system for visualising volumetric data.

Different volume visualisations in Figure 4 can aid visual identification of spatio-temporal patterns in the density volume. For example, there is a hot spot in both space and time in the central part of the cube in all panels of Figure 4 – in this area many trajectories simultaneously come in proximity of each other, that is, we have a *spatio-temporal convergence of trajectories*, which means that there are many objects moving at the same time in close proximity to each other.

Two other types of patterns that could be identified visually using volume visualisations of density volume are temporal bridges and temporal towers. *Temporal bridges* are areas where there is a discrete temporal pattern to high density of trajectories in these areas, that is, there are many objects in these areas at certain times and no objects at other times. *Temporal towers* are columns of consistently high density, which indicate areas where there are many objects in proximity at all times, so density is always high, regardless of time.

Identifying such patterns and their interpretation could be important for certain applications that use movement data – for example when looking at movement of vessels – as we present in the next section.

3. Application: space–time density of vessel movement trajectories in the Gulf of Finland

This section presents an application of the space–time density on a real dataset that describes movements of vessels in the Gulf of Finland. We look at two different types of vessels: passenger ships and tankers and demonstrate how the new spatio-temporal visualisation method contributes to identification of spatio-temporal patterns in movements of ships.

We used 1 month of AIS data for the Gulf of Finland for this experiment. Data were from January 2008. According to the data provider, during this month, the sea was ice-free and so there were no particular restrictions on movement paths of vessels. The original dataset included tracks of 23 different types of vessels, but we used only a subset of trajectories of the two most common vessel types, passenger ships and tankers. Data were split into separate subsets per vessel type and per day so that we could calculate daily densities for each day in January 2008 for each of the two vessel types. The number of passenger ships per day was fairly constant during this month, around 30 per day (mean 28.58, SD 1.28). The number of tankers varied more, 38–60 tankers per day (mean 49.06, SD 5.44).

We calculated daily densities for each of the two ship types for spatial resolution of 2 km in both north-south and east-west directions and temporal resolution of 2000 seconds in a 24-hour interval. This resulted in a density volume of $322 \times 104 \times 44$ voxels. We visualised the daily densities in the same three ways as in the example with simulated data, that is, with direct volume rendering, isosurfaces and clipping planes. We then looked for patterns and similarities in these daily densities. We also calculated a total monthly density for both ship types. The rest of this section illustrates how these densities can be visualised with the above-mentioned volume visualisation methods and lists some examples of spatio-temporal patterns that we could identify.

Figure 5 shows examples of the three different volume visualisation methods for density of real vessel movement data. Figure 5a is a traditional space–time cube for 1 day of data (26 January 2008) and one ship type – tankers. There were 51 tankers in the Gulf of Finland on that day and consequently there are 51 trajectories shown in this space–time cube. Each trajectory is shown in a different shade of purple. There is also a map of the Gulf of Finland draped on the x–y plane of the cube – the map shows the main ports, making it easier to identify origins and destinations of vessels. The display is cluttered with a high amount of overprinting present, particularly in ports.

Figure 5b and c shows the space–time density for this daily dataset, that is, density of tankers on 26 January 2008. The density volume is shown as an isosurface in Figure 5b and with direct volume rendering in Figure 5c. Both these displays are clearer than the space–time cube in Figure 5a, and spatio-temporal patterns are easier to identify visually.

Another example of direct volume rendering is in Figure 5d, which shows the aggregated monthly space–time density of passenger ships in the Gulf of Finland. This same density volume is shown in Figure 5e with a vertical clipping plane aligned to the line between Helsinki and Tallinn.

The main spatio-temporal pattern in the density of passenger ships consists of a very dense traffic line north–south between Helsinki and Tallinn. There, traffic is dense throughout the day, but there are two denser passes in the morning and the evening (shown as spatio-temporal hot spots of high density in the volume rendering in Figure 5d) – that is, a

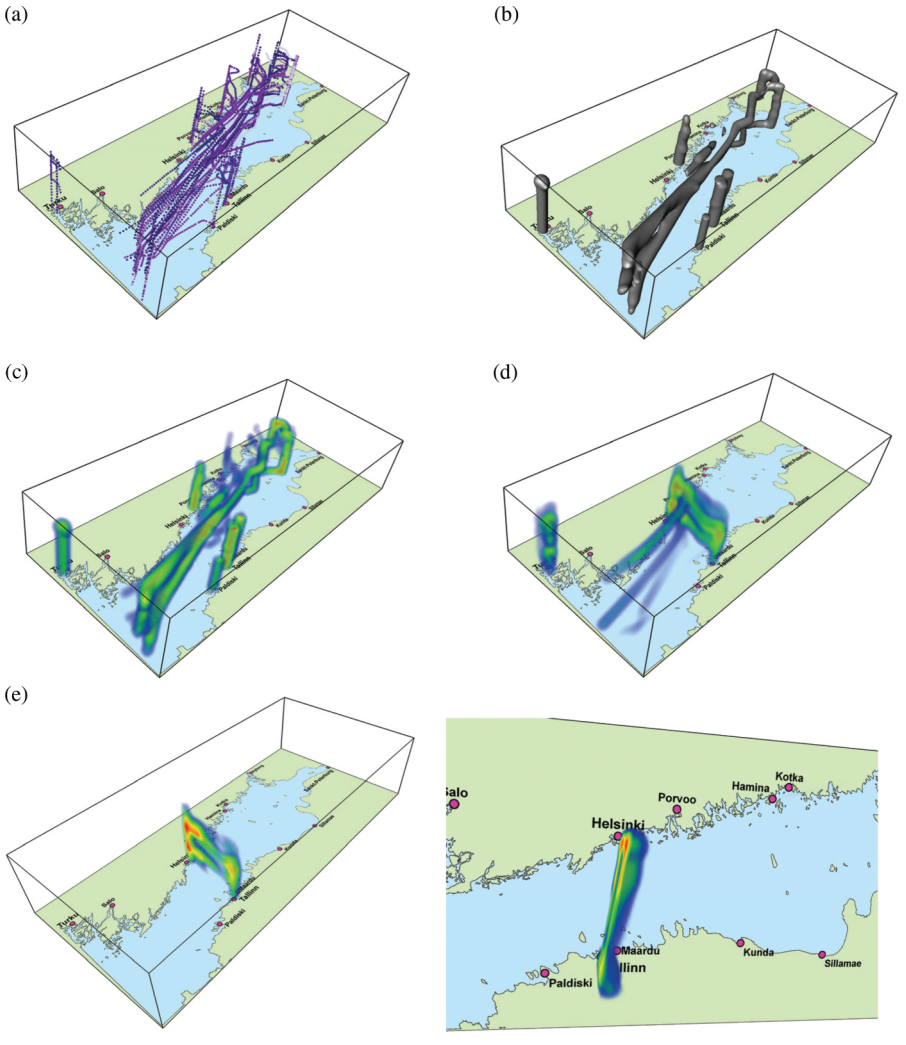

Figure 5. Visualising space–time density volumes of a real movement dataset: (a) trajectories of tankers on 26 January 2008 shown in a traditional space–time cube, (b) space–time density of tankers on 26 January 2008 shown with an isosurface and (c) with volume rendering. Monthly space–time density of passenger ships in the Gulf of Finland shown with (d) volume rendering and (e) using a vertical clipping plane aligned with the line between Tallinn and Helsinki and shown from two different points of view.

spatio-temporal convergence of passenger ship trajectories. These spatio-temporal hot spots are even more eye-catching when the volume is displayed using a vertical clipping plane aligned through Helsinki and Tallinn (Figure 5e). There are very few passenger ships sailing east from the Helsinki–Tallinn line. Furthermore, there are three daily passes from Helsinki to Sweden – these are clearly visible in Figure 5d as the three separated temporal bridges in the central part of the Gulf of Finland, pointing west from the Helsinki–Tallinn line. When interactively exploring these visualisation in Voxler, it is also clear that the passenger ships arrive in Helsinki from Sweden in the morning and return to Sweden in the early evening

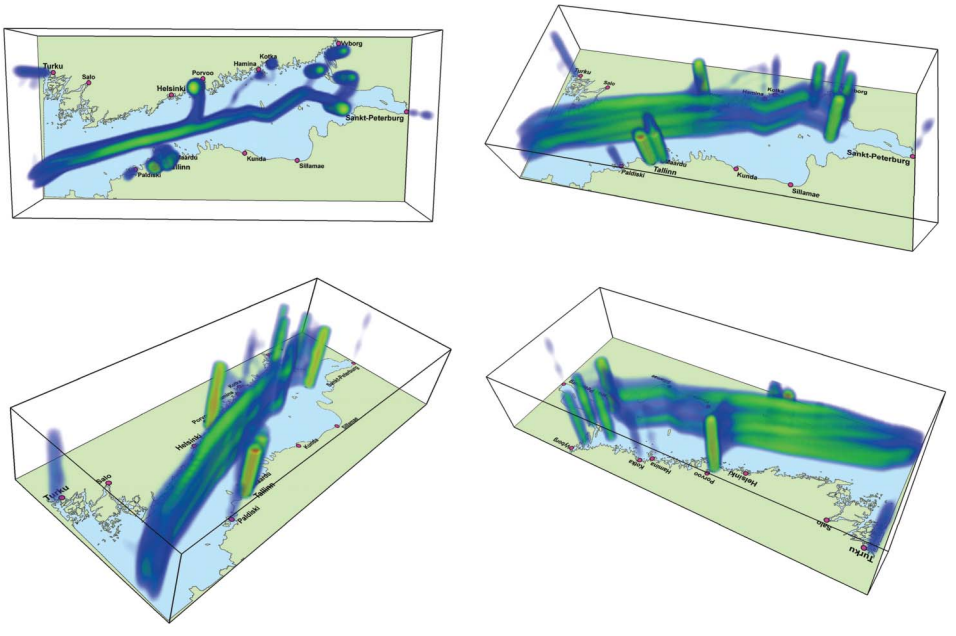

Figure 6. Monthly space–time density of tankers. All four panels show the same density volume, shown from different directions using volume rendering.

(which will be familiar to anyone who has visited Helsinki – these are likely the daily Viking line, Tallinnk and Silja line ferries between Stockholm and Helsinki). Perhaps surprisingly, there are no passenger ships going to Saint Petersburg.

Figure 6 shows the aggregated monthly density of tankers from four different points of view. It is obvious that the pattern of the tankers' density (Figure 6) is very different from the density of passenger ships (Figure 5d). Their main track is east–west, they come in from Baltic sea and sail towards ports in Finland, Estonia and Russia – this pattern is shown with the 'curtain' of high density, stretching in the east–west direction in Figure 6. There are several resting points for tankers in Figure 6 – these are shown as temporal towers of high density, which stretch vertically throughout the day: three such points are around Vyborg, one around Tallinn–Maardu, another large one in the middle of the Gulf of Finland just west of Saint Petersburg and one near Porvoo.

4. Conclusions and discussion

In this article, we present an algorithm for the calculation of 3D space–time density of trajectories and visualise the resulting density volume using several types of volume visualisations. This is a new spatio-temporal visualisation method that attempts to solve the problem of cluttering and overprinting in the space–time cube used for large datasets. The S × T-aggregation of trajectories into a volumetric dataset contributes to easier legibility of the display and facilitates identification of spatio-temporal patterns, which might have gone unnoticed in the original trajectory display. The new methodology could therefore bring significant advantage to any discipline where exploring and analysing large datasets on moving objects is important.

The space–time density preserves the main characteristic of the space–time cube – the inseparability of geographic space and time, yet it has certain limitations. It reveals presence of moving objects in space and time and gives information on flows and routes, but it does not reveal information on other movement attributes, such as direction and speed. The method was designed for objects and phenomena moving in general unconstrained space. It is doubtful how suitable it would be for movement of network-constrained objects (e.g. car trajectories), as areas with high density would be limited to voxels situated directly over the network and therefore any observed patterns would not bring any new information about spatial distribution of movement.

In the context of placing the space–time density into the framework for aggregation for visual exploration of movement data, it should be considered how our methodology compares with similar 2D and 3D aggregations. One similar S × T-aggregation is a series of stacked 2D densities, calculated from situation snapshots of objects at each moment in time. That is, we can look at the distribution of moving objects at times $t_1, t_2, \ldots, t_p$ and calculate the respective (spatial only) 2D kernel density of these objects to produce a snapshot density map for each moment in time. Another similar approach (Andrienko and Andrienko 2010a) introduces a discrete S × T-aggregation into spatio-temporal compartments, where there are certain properties calculated for each compartment based on all the trajectories that have one measured location in that particular compartment. These properties are shown as snapshot maps for each moment in time and linked to other visualisations, such as temporal histograms or mosaic diagrams. Both of these aggregations are algorithmically similar to our space–time density calculation, yet they are different from the conceptual point of view. They consist of a series of snapshot visualisations that shows the use of space at each particular moment in time and does not directly illustrate the continuity of flows and routes through time. As such, the two snapshot aggregation methodologies are suitable for situation-centred space-centred analysis. This is in contrast with our density volume, which, as discussed in the introduction, is a trajectory-based space-centred approach, provides an illusion of continuous movement through time and thereby visually illustrates the inseparability of geographic space and time in the same way as the space–time cube does. Additionally, while the S × T-aggregation by Andrienko and Andrienko (2010a) is suited for analysis of other data attributes (direction and speed) and for network-constrained trajectory data, our space–time density is best suited for unconstrained data and only illustrates location and time. Therefore, these approaches, while similar algorithmically, are conceptually complementary to each other and suitable for different analysis tasks.

To demonstrate the utility of the new visualisation method, we tested our new approach on a real dataset of movement data, representing maritime traffic in the Gulf of Finland. As expected, we were able to visually identify specific spatio-temporal patterns in the density volume, which describe characteristics of the traffic of two ship types in this area, passenger ships and tankers.

There are several issues worth further consideration and these can be grouped into three categories:

(1) *Algorithmical issues*. At present, our algorithm uses a linear kernel around each trajectory, which is based on the distance from each voxel to the 3D polyline. As with standard 2D kernel density, other kernel functions could be implemented and consideration should be given to kernel function type and kernel size (bandwidth).
Another algorithmical issue is computational complexity: this first version of the algorithm requires a calculation of the distance between each 3D polyline and each voxel in its nearest neighbourhood (defined by KernelArea). This results in a

time-consuming process of $O(n \times m \times v)$, where n is the number of trajectories in the dataset, m the maximum length of trajectories (maximum number of points in any trajectory) and v the number of voxels in the density volume, which depends on the spatial extent of the area and pre-defined spatial and temporal resolution. As an example, to calculate density of a dataset of 60 trajectories each with 240 points and in a volume of $322 \times 104 \times 44 = 1473472$ voxels took roughly 1.5 hour on a standard desktop computer with 2.2 GHz processor and 2 GB of RAM – this was for one daily density for one type of ships (tankers). Total processing time for 1 month of tankers' data was 31 times of that, so around 46.5 hours. Total processing time for 1 month of data on passenger ships was roughly half of this, as there were roughly half as many passenger ships each day in the Gulf of Finland as there were tankers. This is of course a very long time, and under such conditions it is impossible to use the method for real-time visual analytics. However, it could be used for analysis of patterns in historic data, where the density is calculated only once for a certain historic dataset and then the real-time trajectories are superimposed over the historic density volume for comparison. How to improve the complexity and adapt the algorithm to work for real-time streaming data warrants further consideration. For example, the algorithm could be made faster using alternative shortcut methods from computational geometry and computer graphics (Schneider and Eberly 2003).

(2) *Visualisation issues*. Space–time density is a volume, which is a difficult data type to visualise and explore visually. We are currently considering other volumetric visualisation methods than the ones presented in the previous sections with the goal to improve clarity of the presentation and facilitate visual pattern identification. In medical visualisation there are a number of computational and visual approaches for classification and querying of volume data (e.g. using methods such as clipping based on volumetric textures, occlusion spectrum, nonphotorealistic rendering, hue-preserving colour blending – see Rheingans and Ebert 2001, Weiskopf *et al.* 2003, Chuang *et al.* 2009, Correa and Ma 2009). Such methods produce volume visualisations where patterns are more pronounced and easier to spot and could therefore be used for exploration of our density volume.
Another idea to consider is to combine volume visualisation of the density volume with dense line visualisations of the trajectories that were used to calculate the density. Dense line data are common in physics (e.g. 3D lines showing electromagnetic fields, streamline data showing particle traces in storms or in fluid flow simulations) and are visualised using methods such as direct 3D line rendering, 3D anaglyphs and depth-dependent halos (Ma *et al.* 2002, Everts *et al.* 2009). It would be interesting to investigate if such dense line visualisations are appropriate for movement data and if they can be displayed in conjunction with space–time density, either in the same 3D visualisation or in two interactively connected displays.

(3) *Application and user-related issues*. Last but not least, all these or other visualisation methods should also be developed with the users in mind and tested for usability and efficiency. Topics here include issues that are important generally (i.e. how humans perceive different types of volume visualisations, what works for visual pattern identification in 3D and what does not), but also specific issues for a particular application. Usability testing and the development of effective visualisations of the maritime situation picture are ongoing research topics in parallel projects (Multimäki *et al.* 2010). In conjunction with our demonstration example – using the novel spatio-temporal visualisation methodology on maritime movement dataset – these issues would have to be considered in close collaboration with intended users – the analysts at the Vessel Traffic Service (VTS) stations at the Finnish Maritime authorities.

5. Acknowledgements

The work of the first author is supported by a Strategic Research Cluster grant (07/SRC/I1168) awarded to the National Centre for Geocomputation by Science Foundation Ireland under the National Development Plan. The work of the second author is supported by The Finnish Maritime Administration (from 1 January 2010 Maritime Department in Finnish Transport Agency), which also kindly provided the AIS data. Research presented in this article is part of the collaboration under the EU COST Action IC0903, 'Knowledge Discovery from Moving Objects (MOVE)'.

References

Andrienko, G. and Andrienko, N., 2010a. A general framework for using aggregation in visual exploration of movement data. *Cartographic Journal*, 47 (1), 22–40.

Andrienko, N. and Andrienko, G., 2010b. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics*, 16, in press.

Andrienko, G., Andrienko, N. and Wrobel, S., 2007. Visual analytics tools for analysis of movement data. *SIGKDD Explorations Newsletter*, 9 (2), 38–46.

Brunsdon, C., Corcoran, J. and Higgs, G., 2007. Visualising space and time in crime patterns: a comparison of methods. *Computers, Environment and Urban Systems*, 31, 52–75.

Callahan, S.P., *et al*., 2008. Direct volume rendering. *Computing in Science & Engineering*, 10 (1), 88–92.

Chen, S., *et al*., 2008. Volume rendering visualization of 3D spherical mantle convection with an unstructured mesh. *Visual Geosciences*, 13, 97–104.

Chuang, J., Weiskopf, D. and Möller, T., 2009. Hue-preserving color blending. *IEEE Transactions on Visualization and Computer Graphics*, 15 (6), 1275–1282.

Correa, C.D. and Ma, K.-L., 2009. The occlusion spectrum for volume classification and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15 (6), 1465–1472.

Crnovrsanin, T., *et al*., 2008. Proximity-based visualization of movement trace data. *VAST Symposium 2008*, October 2008. Columbus, OH, USA.

de Pinto, F.M. and Freitas, C.M.D.S., 2008. Volume visualization and exploration through flexible transfer function design. *Computers & Graphics*, 32, 420–429.

Dykes, J. and Mountain, D.M., 2003. Seeking structure in records of spatio-temporal behaviour: visualization issues, efforts and applications. *Computational Statistics & Data Analysis*, 43, 581–603.

Eccles, R., *et al*., 2008. Stories in GeoTime. *Information Visualization*, 7, 3–17.

Everts, M.H., *et al*., 2009. Depth-dependent halos: illustrative rendering of dense line data. *IEEE Transactions on Visualization and Computer Graphics*, 15 (6), 1299–1306.

Forer, P.C. and Huisman, O., 2000. Space, time and sequencing: substitution at the physical/virtual interface. *In*: D.G. Janelle and D. Hodge, eds. *Information, place and cyberspace: issues in accessibility.* Berlin-Heidelberg: Springer Verlag, 73–90.

Gao, D., 2009. 3D seismic volume visualization and interpretation: An integrated workflow with case studies. *Geophysics*, 74 (1), W1–W12.

Guo, D., 2007. Visual analytics of spatial interaction patterns for pandemic decision support. *International Journal of Geographical Information Science*, 21 (8), 859–877.

Hadwiger, M., *et al*., 2005. Real-time ray-casting and advanced shading of discrete isosurfaces. *Eurographics*, 24 (3), 303–312.

Hsieh, T.-J., Chen, C.-K. and Ma, K.-L., 2010. Visualizing field-measured seismic data. Proceedings of the IEEE Pacific visualization symposium, Taipei: Taiwan, 2–5 March 2010, 65–72.

Jang, Y. and Varetto, U., 2009. Interactive volume rendering of functional representations in quantum chemistry. *IEEE Transactions on Visualization and Computer Graphics*, 15 (6), 1579–1586.

Kapler, T. and Wright, W., 2005. GeoTime information visualization. *Information Visualization*, 4, 136–146.

Karadayi, K., Managuli, R. and Kim, Y., 2009. Three-dimensional ultrasound: from acquisition to visualization and from algorithms to systems. *IEEE Reviews in Biomedical Engineering*, 2, 23–39.

Kraak, M.-J., 2003. The space-time cube revisited from a geovisualisation perspective. *21st international cartographic conference*, August 2003. Durban, South Africa.

Kraak, M.-J., 2008. Geovisualization and time – new opportunities for the space-time cube. *In*: M. Dodge, M. McDerby and M. Turner, eds. *Geographic visualization: concepts, tools and applications*. Chichester, UK: John Wiley & Sons, 293–306.

Kraak, M.-J. and Huisman, O., 2009. Beyond exploratory visualization of space-time paths. *In*: H.J. Miller and J. Han, eds. *Geographic data mining and knowledge discovery.* 2nd ed. London: Chapman & Hall/CRC, 431–443.

Kwan, M.-P., 2000. Interactive geovisualization of activity-travel patterns using three-dimensional geographical information systems: a methodological exploration with a large data set. *Transportation Research Part C*, 8, 185–203.

Laube, P., Imfeld, S. and Weibel, R., 2005. Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19 (6), 639–668.

Lederberger, C., *et al*., 2008. Volume MLS ray casting. *IEEE Transactions on Visualization and Computer Graphics*, 14 (6), 1372–1379.

Legouge, R., *et al*., 2010. Risk and vulnerability analysis in the Gulf of Finland. *Proceedings of the XXIV FIG international congress*, April 2010. Sydney, NSW, Australia.

Ma, K.-L., Schussman, G. and Wilson, B., 2002. Advanced visualization technology for terascale particle accelerator simulations. *Proceedings of the IEEE/ACM SC2002 conference* (SC'02), 16–22 November 2002 Baltimore, MD. Los Alamitos, CA, USA: IEEE Computer Society Press.

Mountain, D.M., 2005. Visualizing, querying and summarizing individual spatio-temporal behaviour. *In*: J. Dykes, A.M. MacEachren and M.-J. Kraak, eds. *Exploring geovisualisation*. Amsterdam: Elsevier, 181–200.

Mühler, K., *et al*., 2010. The medical exploration toolkit: an efficient support for visual computing in surgical planning and training. *IEEE Transactions on Visualization and Computer Graphics*, 16 (1), 133–146.

Multimäki, S., Seppänen, H. and Ahonen-Rainio, P., 2010. User profiling for the Maritime Situational Picture. ICA Geovisualisation Workshop '*GeoSpatial Visual Analytics: Focus on Time*' at the AGILE 2010, May 2010. Guimaraes, Portugal.

Nakaya, T. and Yano, K., 2009. Visualising spatio-temporal crime clusters in a space-time cube. *Proceedings of the 17th GIS research UK conference, GISRUK 2009*, April 2009. Durham, UK.

O'Sullivan, D. and Unwin, D.J., 2003. *Geographic information analysis*. Hoboken, NJ, USA: John Wiley & Sons.

Preim, B., *et al*., 2009. Survey of the visual exploration and analysis of perfusion data. *IEEE Transactions on Visualization and Computer Graphics*, 15 (2), 205–220.

Rheingans, P. and Ebert, D., 2001. Volume illustration: nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, 7 (3), 253–264.

Rinzivillo, S., *et al*., 2008. Visually driven analysis of movement data by progressive clustering. *Information Visualization*, 7, 225–239.

Schneider, P.J. and Eberly, D.H., 2003. *Geometric tools for computer graphics*. San Francisco, CA: Morgan Kaufmann Publishers.

Silverman, B.W., 1986. *Density estimation for statistics and data analysis*. New York: Chapman and Hall.

Skupin, A. and Hagelman, R., 2005. Visualizing demographic trajectories with self-organizing maps. *GeoInformatica*, 9 (2), 159–179.

Slingsby, A., Dykes, J. and Wood, J., 2008. Using treemaps for variable selection in spatio-temporal visualization. *Information Visualization*, 7, 210–224.

Slocum, T.A., *et al*., 2010. *Thematic cartography and geovisualization*. 3rd ed. Upper Saddle River, NJ: Pearson/Prentice Hall.

Vrotsou, K., Forsell, C. and Cooper, M., 2009. 2D and 3D representation for feature recognition in time geographical diary data. *Information Visualization*, 8, 1–14.

Weiskopf, D., Engel, K. and Ertl, T., 2003. Interactive clipping techniques for texture-based volume visualization and volume shading. *IEEE Transactions on Visualization and Computer Graphics*, 9 (3), 298–312.

Willems, N., van de Wetering, H. and van Wijk, J.J., 2009. Visualization of vessel movements. *Eurographics/IEEE-VGTC Symposium on Visualization 2009*, 28 (3), 959–966.

Wilson, C., 2008. Activity patterns in space and time: calculating representative Hägerstrand trajectories. *Transportation*, 35, 485–499.

Zhao, J., Forer, P. and Harvey, A.S., 2008. Activities, ringmaps and geovisualization of large human movement fields. *Information Visualization*, 7, 198–209.