Generating Dynamic Raster Data

Theodoros Tzouramanis[1] Michael Vassilakopoulos[2] Yannis Manolopoulos[1]

[1]Dept. of Informatics, Aristotle University of Thessaloniki
541 24 Thessaloniki, Greece E-mails:{theo,manolopo}@delab.csd.auth.gr

[2]Dept. of Informatics, Technological Educational Institute of Thessaloniki
P.O. Box 14561, 541 01 Thessaloniki, Greece E-mail:vasilako@it.teithe.gr

Abstract

Benchmarking of spatio-temporal databases is an issue of growing importance. In case large real data sets are not available, benchmarking requires the generation of artificial data sets following the real-world behavior of spatial objects that change their locations, shapes and sizes over time. In this report, a generator for changing raster data, called "Generator of Time-Evolving Regional Data" (G-TERD), is presented. The basic concepts that determine the function of G-TERD are the structure of complex two-dimensional raster objects, their color, maximum speed, zoom and rotation-angle per time slot, the influence of other moving or static objects on the speed and on the moving direction of an object, the position and movement of the scene-observer, the statistical distribution of each changing factor and finally, time. In the framework developed, the user can control the generator response by setting several parameters values.

1 Introduction

Spatio-temporal databases (STDBs) provide a framework for the efficient storage and retrieval of all states of a spatial database over time. This includes the current and past states and the support of spatial queries that refer to present and past time points, as well. During the last years many efforts have focused on spatio-temporal formalism, data models, query languages, visualization and access methods. However, little work has appeared on benchmarks for STDBs.

The goal of benchmarks in STDBs is to compare the performance of different implementation alternatives. An example of a benchmark is the comparison of space requirements and query execution time of spatio-temporal access methods (STAMs). In order to evaluate such STAMs, extensive experimentation using real and synthetic data is required. A good benchmark must correspond to a recognizable, comprehensible real-life situation. It is important that the results hold not only for a specific environment but in more general settings, as well. Thus, the user is able to repeat the experiments and come to similar conclusions. Very often, either real data sets are not available or they cannot be useful for testing extreme conditions. In both cases synthetic data sets can be generated by some artificial specifications rather than by obeying a real-world behavior.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Much work has been done towards data generation for benchmarks in prototype and commercial non-spatio-temporal databases (for example, [4, 5]). These generated data can be used to test different implementations under various operating conditions. In STDBs, the work on the generation of test data is limited and only a few innovative papers have appeared in the literature. In [9] the authors present a general approach for the generation of synthetic scenes of moving points, or rectangular objects. This approach is parameter-driven and does not support the interaction between objects. In [6], the previous work is extended by introducing new features to the generation process. In [7] a specialized spatio-temporal data generator, motivated by an application modeling fishing boats, is proposed, where there exist no, or very few restrictions for the motion: e.g., objects of one type may be attracted, or repulsed by objects of other classes. Recently, another approach for the generation of test data, which is motivated from applications in the context of traffic telematics, has been presented [1, 2].

None of these approaches is suitable for benchmarking STAMs for raster data (especially quadtree-based STAMs, for example, [11, 12]). In this report we review a novel approach for data generation, which is specifically designed for applications stemming from the field of dynamic (time-evolving) raster data [13]. The generator presented in [13] is called *"Generator of Time-Evolving Regional Data"* (G-TERD). This software tool is highly parameterized so that different parameter values may produce spatio-temporal data set distributions with different characteristics. In this framework, the user can control the behavior of the generator by defining parameters and statistical models. Such an approach considers the characteristics of a wide range of applications.

In order to make G-TERD available to the user for experimentation purposes, a Web site has been created, which allows downloading the MS-DOS executable file of the generator, the user documentation and the source code in C language. The Web site is at `http://delab.csd.auth.gr/stdbs/g-terd.html` and provides, also, a visualization tool (based on Gnuplot 3.7) that runs under MS-Windows. It is offered with sample data sets obeying different user-defined settings.

The remaining part of the report is organized as follows. Section 2 describes the basic concepts which were considered during the generator design for continuously changing synthetic raster data. Section 3 describes the operation of the new generator and some basic programming issues. Section 4 presents example setups and the sequences of scenes generated. Finally, the last section concludes the report and discusses future work issues.

2 Fundamental Concepts

We assume a two-dimensional workspace, where real coordinates are used (float numbers in C) and that the space extent on the x- and the y-axis is set by the user. In the sequel, in order to simplify presentation, the values of coordinates are counted in "units". With respect to the time domain, we assume that the changing scene lasts for a period of time $T = [0, t_{max})$. This period is divided in time slots. We further assume that the scene remains unchanged during a time slot. In other words, time is digitized.

2.1 Objects and sub-objects

The basic data structure of G-TERD is the *time-evolving two-dimensional regional object*. Each object consists of a group of sub-objects which are quadrangles of the same, or different colors. This group of sub-objects, in general, changes at each time slot. An object, or sub-object may have a static shape, or it can change its spatial extension with time. The spatial extension of an object is determined by the smallest possible two-dimensional rectangle, called *Minimum Bounding Rectangle* (MBR), that encloses all its sub-objects. The user sets the maximum size of the object MBR and the maximum size of the sub-object quadrangle.

In order to simulate real-world complex non-rectangular objects (e.g. cars, animals, airplanes, clouds, islands) by an object, some, or all of its sub-objects may be connected and remain connected, for the whole lifetime of the object.

The number of objects appearing in the scene and the number of sub-objects in an object are dynamic. The objects may be static, or moving towards any spatial direction. The square sub-objects may, also, expand, or shrink and rotate around their center. For the sake of simplicity, both elastic shocks and plastic crushes between objects, are not allowed. Instead, there is a possibility that the one object will pass over the other, as if there were moving in different heights, or there is a possibility that both objects will change moving directions in order to avoid the crush.

2.2 The Scene-Observer

The *scene-observer* plays a central role in a time-evolving scene. In real-life, the observer may be a satellite monitoring the earth surface, a telescope watching the space universe, the field of vision of a video or a photo camera, the eyes of an onlooker, etc. The scene-observer in G-TERD is a virtual two-dimensional rectangular window that shifts, zooms and rotates over the scene and films it, by printing one snapshot image per time slot, for the whole lifetime of the evolution.

The observer's window side length is user-controlled and it digitizes a workspace portion. The dimensions of a quadrangle of the workspace that is represented by a pixel of the observer's window, or in other words, the size of such a pixel varies according to the zoom-in, or zoom-out factor. The scene-observer may move towards any spatial direction, or he/she may be static for some periods of time or for the whole scene lifetime. The observer may shift ("travel") in a random way over the surface of the predefined workspace, or may follow the movement of a specific "live" object. Other functions supported are the zoom in-out and the rotation around the center point of the observer's window. Using the concept of the scene-observer's window, a sequence of multicolored images that can be saved in the disk are produced.

2.3 Change of Spatial Locations, Size, Shape and Color of the Objects

Data objects may change their spatial location, size and/or their shape at different time intervals, according to the values of their speed, zoom and rotation-angle fields. These fields are measured in units of the workspace coordinate system per time slot and their domains are bounded by minimum and maximum values set by the user. There are static, slow and fast moving objects. When an object is static (in slow movement) the speed, zoom and rotation of all its sub-objects is set at zero (at half the value of the speed, zoom and rotation of the object). In real-life examples, objects appear gradually in our field of vision. This also holds for objects and sub-objects in G-TERD. Moreover, sub-ojects may have the same, different, or slightly different colors (following changes of the intensity of light).

3 The operation of G-TERD

3.1 User-Defined Parameters and Distributions

In G-TERD several parameters may be user-defined in order to let the user control the behavior of the generator (Table 1). The appropriate definition of these parameters is the simplest technique to control the properties of the resulting data sets. For instance, by setting $speed\_min = speed\_max$ for both the x- and y- axes, then all the objects are forced to move in a parallel fashion at the same speed as if they were one object. This is similar to the movement of birds or military aircrafts flying together, or to the movement of a group of soldiers.

In order to have a generalized tool for benchmarking that simulates classes of real-life applications, several quantities that determine the operation of this tool must be random variables that obey a specified distribution among the ones mentioned above. For G-TERD, these random variables are presented in Table 2.

Through careful specification of different distributions for the variables of Table 2, the user can simulate several interesting scenarios. For instance, by using the exponential distribution with small mean for the speed

Parameter	Explanation
X_{max}	workspace length on the x-axis
Y_{max}	workspace length on the y-axis
$num\_timeslots$	time duration of the changing scene ($T = [0, num\_timeslots)$)
max_num_objs	maximum number of "live" objects per time slot
$max\_num\_subobjs$	maximum number of "live" sub-objects in an object per time slot
num_colors	number of colors of the color palette
$observer\_side$	length of the observer's window side
obj_side_max	maximum side length of object MBR
$subobj\_side\_max$	maximum side length of the rectangular sub-object
$percent\_objs\_static$	percentage of static objects per time slot
$percent\_objs\_slow$	percentage of objects in slow movement, zoom and rotation per time slot
$percent\_objs\_fast$	percentage of objects in fast movement, zoom and rotation per time slot ($percent\_objs\_fast = 100 - percent\_objs\_static - percent\_objs\_slow$)
$speed\_min[]$	minimum object speed per time slot positive value on x- (y-) axis: movement to the East (North)
$speed\_max[]$	maximum object speed per time slot positive value on x- (y-) axis: movement to the East (North)
$zoom\_min$	minimum object in-out zoom per time slot ($-1 \leq zoom\_min \leq 1$)
$zoom\_max$	maximum object in-out zoom per time slot ($-1 \leq zoom\_max \leq 1$)
$rotation\_min$	minimum object rotation-angle per time slot
$rotation\_max$	maximum object rotation-angle per time slot
$t_{max\_duration}$	maximum number of time slots that have to elapse before the next computation of a field value of an object, a sub-object or of scene-observer
$live\_objects$ at $t = 0$	number of "live" objects at time slot $t = 0$

Table 1: The user-defined parameters of G-TERD.

and the period of time before the re-computation of the object speed, most of the objects would move slowly and "nervously" on the workspace, since their speed would change direction very frequently.

Evidently, by properly adjusting the domain value of each variable of Table 2, the user may limit the data generated from the chosen distribution. For instance, we can consider setting the domain of the zoom value equal to [-1,0]. This will lead to a scenario where every created object would expand for a while, during the initialization phase of its creation, and afterwards it will be deleted ("die") in a very short time. G-TERD can generate benchmark data for many application domains, given that their data distributions are known. It currently supports the Uniform, Triangular, Normal, Exponential, Zipf and Poisson distributions.

3.2 Creation and Update of an Object

Initially, the new object location is selected so that its center point is randomly placed in the workspace, according to a predefined statistical distribution. The acceptable placement of an object is controlled by a function, which checks if the selected workspace area is occupied by another object and if the two objects are allowed to overlap. Afterwards, a decision is made about the number of sub-objects that each object will initially have, about their color (the same for all sub-objects, or not) and about their speed and rotation-angle per time slot. The speed and rotation-angle of each sub-object are set randomly, following their domain value and the properties of the related user-defined distributions. The instance of a sub-object for the next time slot is calculated, for each speed, zoom and rotation-angle candidate value.

The scenario is not very realistic at the beginning of the data generation. All the sub-objects of the newly created "live" objects cover a surface of zero size and expand by 1 square unit per time slot. Therefore, to obtain satisfactory results, the generated data should not be used during a *warm-up phase* [7]. More details on how G-TERD implements the warm-up phase and the creation of obejcts in general can be found in [13].

The update procedure of an object starts with the calculation of the new object location and MBR. If all the sub-objects have the same color and the period of time that this color remains unchanged has expired, a new

Variable	Distribution	Domain
number of "live" sub-objects $obj.live\_subobjs$ in an object obj at the time slot of its creation	user-defined	[1, $max\_num\_subobjs$]
initial distribution of the center point of a new object	user-defined	the workspace
number of new objects per time slot	user-defined	$[0, max\_num\_objs - live\_objects]$
number of new sub-objects in an object per time slot	user-defined	$[0, max\_num\_subobjs - obj.live\_subobjs]$
deletion time slot $obj.endtime$ of an object	user-defined	[birth time of the object + 1, $num\_timeslots$]
deletion time slot $subobj.endtime$ of a sub-object	user-defined	[birth time of the object in which it belongs + 1, $obj.endtime$]
color $.color$ of an object or a sub-object	user-defined	[1, num_colors]
speed $subobj.speed$ of a moving sub-object per time slot and spatial axis	user-defined	$[speed\_min, speed\_max]$
zoom $subobj.zoom$ of a moving sub-object per time slot	user-defined	$[zoom\_min, zoom\_max]$ with $-1 \leq zoom\_min \leq zoom\_max \leq 1$
rotation-angle $subobj.rotation$ of a moving sub-object per time slot	user-defined	$[rotation\_min, rotation\_max]$
number of time slots that must elapse before the next computation of an attribute such as the speed, zoom, color, etc.	user-defined	[1, $t_{max\_duration}$]

Table 2: Variables that are controlled by statistical distributions.

color for all its sub-objects is selected. Otherwise, each sub-object them may change its color, independently. If the period of time during which the speed, zoom and/or rotation-angle of each sub-object has expired, a new value is set to the corresponding field. The instance of the sub-object for the next time slot is calculated, for each speed, zoom and rotation-angle value and a procedure is followed to avoid an undesirable crush.

3.3 Positioning and Output of the Scene-Observer

The shift, zoom and rotation functions of the scene-observer are similar to the corresponding functions of an object. If the scene-observer follows the evolution of a specific object, then the observer's speed is proportional to the distance of the center point of its window from the center point of the object followed. In this case, the zoom in-out function of the observer takes care of keeping the whole object inside the observer's window.

The output of the scene-observer is a sequence of multicolored images that can be saved in the disk. For each pixel in the observer's window, the algorithm checks if there is any sub-object covering the pixel. In this case, the sub-object color is recorded in the output. If sub-objects of different objects cover the same pixel of the observer's window, then the color of the object, which can pass over any other of the involved objects, appears in the output image.

3.4 The Main Routine

The input of the algorithm consists of the values and statistical distributions of all the parameters and variables that appear in Table 1 and Table 2, respectively. During the scene initialization phase at time slot $t = 0$, a user-defined number of "live" objects is created and located in the workspace. The initialization of the observer's window and the output of the first snapshot image of the time-evolving scene follow.

During the main loop phase new object instances are generated. If the deletion time slot of an object has already been reached and the surface of each of its sub-objects is less than 1 square unit, the object is deleted from the scene. A random number of new objects at each time slot is also created and placed in the workspace. The random number of new objects follows a predefined statistical distribution, such as the ones discussed in subsection 3.1. Finally, the observer's window is located at another position over the workspace and the output function prints the snapshot image of the current state of the continuously changing scene.

4 A Sample Run

In the following, we give an example of the use of G-TERD. The side length of the scene-observer's window is set at 1024 units, the maximum number of live objects per time slot is 70 and the lifetime of the evolution is 101 time slots. The random number of new sub-objects per time slot and their deletion time slot are Uniform. The same holds for the speed, the color and the period of time that must elapse before the next computation of an attribute such as the speed, zoom, etc. Finally, the change of the size and shape of the objects/sub-objects are controlled by the zoom and rotation-angle values per time slot, which are generated by the Triangular distribution. The scenario in Figure 1 illustrates static objects that are uniformly distributed in a workspace of

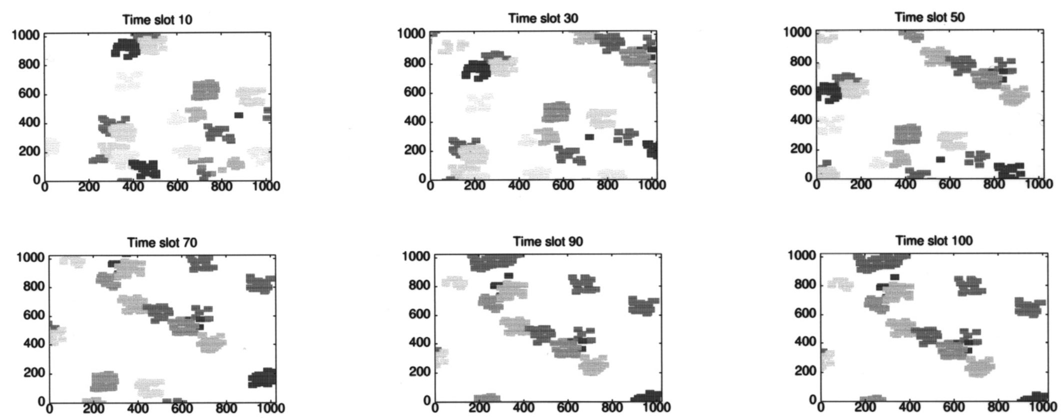

Figure 1: Static objects and a scene-observer moving from South-West to North-East.

2000$\times$ 2000 units (color versions of these greyscale figures can be obtained via the G-TERD site). The speed domain value is [10,10] for each axis. Therefore, the scene-observer shifts over the workspace surface at a constant speed and diagonal orientation, from South-West to North-East. The objects are all created at the initial time slot, whereas in the sequel the creation of a new object is not allowed, since the distribution of new objects per time slot was Normal with mean μ=0 and mean square deviation σ=0. No objects are deleted during the scene lifetime, since the distribution of their deletion time was also Normal with mean μ=100 and mean square deviation σ=0.

5 Conclusion and Future Work

In this report we reviewed the first approach, the G-TERD generator, for data generation specifically designed for applications stemming from the field of dynamic (time-evolving) raster data. The basic concepts involved in the development, operation and use of G-TERD were examined. G-TERD is highly parameterized and flexible, thus supporting the simulation of a variety of real-world scenarios. The Web site of G-TERD provides access to the generator, its source code and some illustrative examples. G-TERD offers a framework for creating user-defined synthetic time-evolving raster data sets that can be used for the experimental comparison of different STAMs.

In the future, we plan enhancing G-TERD to support a greater variety of distributions, such as skewed distributions or correlative two-dimensional distributions. Besides, the Web-based interface will be developed in order to both generate and make it possible to visualize time-evolving synthetic raster data.

References

[1] T. Brinkhoff: "Generating Network-Based Moving Objects," *Proc. 12th Int. Conf. on Scientific and Statistical Database Management*, pp.253-255, Berlin, Germany, 2000,

[2] T. Brinkhoff: "A Framework for Generating Network-Based Moving Objects," *Geoinformatica*, Vol.6, No.2, pp.153-180, 2002

[3] O. Guenther, P. Picouet, J.-M. Saglio, M. Scholl and V. Oria: "Benchmarking Spatial Joins A La Carte," *Int. Journal of Geographical Information Science*, Vol.13, No.7, pp.639-655, 1999.

[4] C. Gurret, Y. Manolopoulos, A. Papadopoulos and P. Rigaux: "BASIS: a Benchmarking Approach for Spatial Index Structures," *Proc. Workshop on Spatiotemporal Database Management*, pp.152-170, Edinburgh, Scotland, 1999.

[5] J. Pei, R. Mao, K. Hu and H. Zhu: "Towards Data Mining Benchmarking: a Testbed for Performance Study of Frequent Pattern Mining," *Proc. 2000 ACM SIGMOD Conf.*, pp.592, Dallas, TX, 2000.

[6] D. Pfoser and Y. Theodoridis: "Generating Semantics-Based Trajectories of Moving Objects," *Proc. Workshop on Emerging Technologies for Geo-Based Applications*, Ascona, Italy, 2000.

[7] J.-M. Saglio and J. Moreira: "Oporto: a Realistic Scenario Generator for Moving Objects," *Geoinformatica*, Vol.5, No.1, pp.71-93, 2001.

[8] Y. Theodoridis and M.A. Nascimento: "Generating Spatiotemporal Datasets on the WWW," *ACM SIGMOD Record*, Vol.29, No.3, pp.39-43, 2000.

[9] Y. Theodoridis, J.R.O. Silva and M.A. Nascimento: "On the Generation of Spatiotemporal Datasets," *Proc. 6th Symp. on Spatial Databases*, pp.147-164, Hong Kong, China, 1999.

[10] Y. Theodoridis, T. Sellis, A. Papadopoulos and Y. Manolopoulos: "Specifications for Efficient Indexing in Spatiotemporal Databases," *Proc. 7th Conf. on Statistical and Scientific Database Management Systems*, pp.123-132, Capri, Italy, 1998.

[11] T. Tzouramanis, M. Vassilakopoulos and Y. Manolopoulos: "Multiversion Linear Quadtree for Spatio-Temporal Data," *Proc. 4th East-European Conf. on Advanced Databases and Information Systems*, pp.279-292, Prague, Czech Republic, 2000.

[12] T. Tzouramanis, M. Vassilakopoulos and Y. Manolopoulos: "Overlapping Linear Quadtrees and Spatio-Temporal Query Processing," *The Computer Journal*, Vol.43, No.4, pp.325–343, 2000.

[13] T. Tzouramanis, M. Vassilakopoulos and Y. Manolopoulos: "On the Generation of Time-evolving Regional Data," *GeoInformatica*, Vol.6, No.3, pp.207–231, 2002.