MethodsX 3 (2016) 69–86

Contents lists available at ScienceDirect

MethodsX

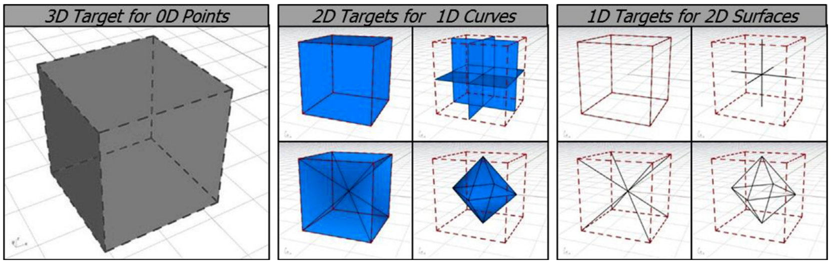
journal homepage: www.elsevier.com/locate/mex

Voxelization algorithms for geospatial applications
Computational methods for voxelating spatial datasets of 3D city models containing 3D surface, curve and point data models

Pirouz Nourian [a,*], Romulo Gonçalves [b], Sisi Zlatanova [c], Ken Arroyo Ohori [c], Anh Vu Vo [d]

[a] TU Delft, Faculty of Architecture and the Built Environment, Department of Architectural Engineering + Technology, Design Informatics, Netherlands
[b] eScience Researcher at Netherlands eScience Center (NLeSC), Netherlands
[c] TU Delft, Faculty of Architecture and the Built Environment, Department of Urbanism, 3D Geo Information, Netherlands
[d] University College Dublin, Environmental Modelling Group, Ireland

GRAPHICAL ABSTRACT



ABSTRACT

Voxel representations have been used for years in scientific computation and medical imaging. The main focus of our research is to provide easy access to methods for making large-scale voxel models of built environment for

* Corresponding author.
E-mail addresses: P.Nourian@tudelft.nl (P. Nourian), R.Goncalves@esciencecenter.nl (R. Gonçalves), S.Zlatanova@tudelft.nl (S. Zlatanova), G.A.K.ArroyoOhori@tudelft.nl (K.A. Ohori), Anh-Vu.Vo@ucdconnect.ie (A. Vu Vo).

http://dx.doi.org/10.1016/j.mex.2016.01.001

environmental modelling studies while ensuring they are spatially correct, meaning they correctly represent topological and semantic relations among objects.
In this article, we present algorithms that generate voxels (volumetric pixels) out of point cloud, curve, or surface objects. The algorithms for voxelization of surfaces and curves are a customization of the *topological voxelization* approach [1]; we additionally provide an extension of this method for voxelization of point clouds.
The developed software has the following advantages:

- It provides easy management of connectivity levels in the resulting voxels.
- It is not dependant on any external library except for primitive types and constructs; therefore, it is easy to integrate them in any application.
- One of the algorithms is implemented in C++ and C for platform independence and efficiency.

A R T I C L E I N F O
Method name: Topological voxelization algorithms
Keywords: Topological voxelization, Environmental modelling, 3D city models, Point cloud voxelization, Geo-spatial database
Article history: Received 21 September 2015; Accepted 1 January 2016

Method details

Requirements

- The C++ and C version of our libraries requires the input to be stored in OBJ format, some examples are provided under: https://github.com/NLeSC/geospatial-voxels/tree/master/software/voxelGen/data.
- Rhinoceros3D® and Grasshopper©, NURBS Modelling for Windows (Windows 7 or higher) with Dot NET framework (version 3.5 or higher) to run C# version.
- The codes have been tested with PC 64-bit machines with Intel (R) i7 Core(TM) CPU@ 2.50 GHz and installed memory (RAM) of 8 GB.

Software

- Link: https://github.com/NLeSC/geospatial-voxels/tree/master/software/voxelGen.
- License: Our *software is licensed under the* Apache License Version 2.0 (APLv2).

Reuse potential

The methods and algorithms can be reused, adapted and integrated with software applications that require voxelization. The installation steps are described in GitHub.

Introduction

We present three methods for voxelating point, curve, and surface objects. For curve (1D) and surface (2D) objects we present algorithms for the methods mathematically described by Laine [1] and for voxelizing points (0D) we devise our own algorithms. Note that 0D, 1D, and 2D refer to the topological dimension of inputs; this, however does not contradict the fact that all inputs are embedded in three-dimensional Euclidean space of $\mathbb{R}^3$, i.e. a Cartesian product of X, Y and Z coordinates represented in the domain of real numbers $\mathbb{R}$. Full scripts of algorithms are available in the abovementioned repository. Laine [1] mathematically proves that using the so-called intersection targets desired connectivity levels such as 6 or 26 connected voxel results could be achieved. He does not present an algorithm in detail though. In developing an algorithm for dealing with large datasets, we had to come up with an efficient way of iteration. In general, one can either iterate over all possible