Real Time Processing of Large Data Sets from Built Infrastructure

Vasco VARDUHN [1], Ralf-Peter MUNDANI, and Ernst RANK
Computation in Engineering, Technische Universität München, Germany
[1] corresponding author address: *varduhn@tum.de*

ABSTRACT

In this paper, we present a framework that gives the user a tool at hand to explore large data sets from built infrastructure.
In a first step we describe the integration of fully detailed product models of constructions delivering the geometric and auxiliary information we build up our data exploration from.
The main part of this paper follows by presenting the application of a hierarchical data structure, an octree, that is capable of holding building information for a whole region or even a country and the development of complexity reduction algorithms that allow the visualisation of those data. To exploit the full performance of modern hardware platforms, the application of parallelisation techniques is inevitable and we present the implementation of these techniques to the data processing steps performed in the framework.
After introducing the framework, we show possible applications to various disciplines such as environmental and civil engineering, architecture, or disaster management.

Keywords: Hierarchical Data Structures, Built Infrastructure (BIM/IFC), Parallel Computing, Interactivity, Algorithm Optimisation, Visualisation

1. INTRODUCTION

Nowadays, within civil and environmental engineering built infrastructure data becomes more and more important for planning and maintaining purposes [7, 10]. Therefore, the data must be available on all scales from entire buildings down to single entities such as walls, heating / cooling devices or even screws and sockets as well as any available auxiliary information. Obviously this entails a huge data advent, especially when considering a whole city or region, thus, sophisticated techniques are inevitable to process the data in real time as needed for interactive applications.
In this paper, we describe a hierarchical approach to access and process huge infrastructure data sets in real time stored to (distributed) databases using the Industry Foundation Classes (IFC) [9]. The objective is to give the user a tool at hand which lets him freely navigate through the data in order to gain insight and explore details. Therefore, a scale-dependent filtering is necessary to provide the appropriate presentation (region → city → building → floor → room → window → ...) and to hide unwanted details. Since the IFC provides the data in the highest level of detail, hierarchical structures such as octrees are advantageous to restrict the access to the relevant data only and, thus, to prevent unnecessary computations.
In our approach, we use a dynamic two-layer octree structure to handle memory and bandwidth limitations. The first layer is being pre-computed to store the assembly of the infrastructure objects on the global scale while the second layer is being computed on-the-fly giving fully detailed information of single infrastructure objects. In order to guarantee the processing of the data in real time, pre-fetching of the relevant information from the database is applied to reduce latency. Nevertheless, due to the interactive user behavior this is a highly dynamical process, hence predictions are necessary for pre-fetching proper data. Those predictions are based on a proximity analysis coupled with heuristics (for possible user movements) to select the most probable alternatives for fetching and processing data in parallel to cover the user's next step.
To fulfill also interactive visualisation requirements, level-of-detail considerations depending on the relative position of objects and the user have to be applied to the different infrastructure entities during run time. Due to the usage of the IFC any geometric and auxiliary information can be easily displayed at the highest level. On coarser levels, the visualisation of (large) objects such as bridges is performed by displaying textured boundary boxes for instance or just lines on even coarser levels.
This paper is organised as follows: In section 2 we introduce the format used for storing construction and built infrastructure models, section 3 is the main part of this paper as it presents the developed data structure, section 4 introduces the steps of complexity reduction by applying levels of detail and algorithms for enhancing visualisation. Section 5 introduces the design of parallelisation techniques and in section 6 some first results of our work are shown. Our paper closes with the conclusions and the outlook given in section 7.

2. EXCHANGE FORMAT FOR CONSTRUCTIONS AND BUILT INFRASTRUCTURE

In this section, we focus on the format for accessing Built-Infrastructure-Models (BIM) – i. e. product models that store relevant information exceeding the geometric representation of constructions and built infrastructure. It turned out that Industry-Foundation-Classes fulfill our demands and therefore have been selected and integrated.
In the process of selecting a file format for constructions and built infrastructure three criteria where mandatory for a format to be suitable. First the format has to be capable of storing information about constructions in fully detail. As we will show in section 4, the generation of different levels of detail is a crucial task for our project and will be put into practice by a step-wise coarsening of fully detailed representations. Second the format not only has to provide geometric information but auxiliary information as well. These auxiliary information have to provide a certain ordering of the data in a way that for example for a building all its parts such as windows, doors, walls, or the roof can be identified. Furthermore it has to be mapped which parts of the geometric representation belong to which element, for instance a door, of the building and vice-versa. Additionally for every element miscellaneous information such as the IP address of a network port or the type of glass a window is manufactured of have to be stored and be editable to the user. Third the format to be chosen has to be an industry standard to ensure sufficient exchange with existing file formats and make this framework open to most of existing data of constructions and built infrastructure.
All three criteria should be self explaining but we want to point out the importance of the second criterion, the demand for storing auxiliary information. This gives us the possibility to provide information to the user beyond the scope of tools such as Google Earth. To explore a whole city and being able to get into every detail of a construction or built infrastructure opens the door to various imaginable user scenarios from civil and environmental engineering [1–3, 6], to architecture [4] or disaster management [13], just to name a few.
Als already mentioned it turned out that Industry-Foundation-Classes [9] are completely suitable for our purposes, although there are known difficulties about IFC. IFC don't provide information such as the exact intersection of walls, but since this does not affect our application it can be neglected. Furthermore the IFCs are maintained by a large community and due to this position on the market of BIM formats, there are several tools and APIs for accessing IFC files. We decided to use the IFCEngine [12] to extract geometric and auxiliary information from IFC files during loading to the framework. Geometric information are received in form of a Vertex, Edge, Face stream and auxiliary information as a mapping between elements such

as doors or walls and their geometric description. Further auxiliary information is mapped as strings to single elements. In a first step of preprocessing basic information such as the bounding boxes of the constructions are computed for the buildup of the hier-archical data-structure as described in section 3.

3. APPLICATION OF A HIERARCHICAL DATA-STRUCTURE

In this section, we discuss the main part of our work, namely the data structure of the framework, which is crucial for fulfilling the demands of the project. Since it is our goal to create a framework which lets the user freely and seamlessly navigate through large sets of construction and built infrastructure data, the development of the data-structure is a central point. On the one hand, we have to make sure that this data-structure is capable of holding construction data of a whole region without exceeding memory limitations, on the other hand, the access to those information has to be efficient such that our claim for seamless data access is fulfilled. We figured out that following a hierarchical approach and implementing an octree structure is well suited for our purposes due to multiple reasons. The octree is based on spatial decomposition of a given domain, it divides the domain along every axis in two equal parts resulting in eight sons (octants), each of these octants will be recursively subdivided until it lies complete inside (black node) or outside (white node) the given object or the maximum depth of the tree is reached (grey-leaf node). Therefore it even adds spatial ordering to the distinct data of constructions and built infrastructure originating from IFC and enables us to determine spatial relations as described in the following sections.

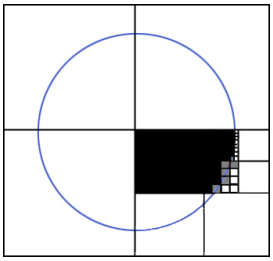

Figure 1: *Application of a quadtree, the 2D version of an octree, to the section of a circular geometry.*

There exist well known and efficient algorithms to provide location awareness between objects stored in an octree, which is essential for the integration to our framework. Due to its spatial decomposition the octree natively features different levels of detail by varying the depth of the tree.
We designed the data structure to hold a precomputed first-layer octree and a set of dynamically generated second-layer octrees in memory.

First Layer Octree

The first layer octree is generated based on the bounding-boxes of all IFC objects. This octree is static and kept in memory during run-time, since it is the central point of our framework. The octree holds the complete assembly of constructions but is "small" enough to be kept in memory. All decisions on which objects to load, when to load them or discard them are made by evaluating this first-layer octree. In this octree we hold storage information for the single BIM models, which we use to estimate parameters such as latency before the loading process. We use this to load these IFC data from different (distributed) databases at the right time.
When building up the first-layer octree we follow natively the recursive definition of spatial decomposition of the octree. In the first step one root octant is created covering the whole domain and containing references to the storage information and the bounding box of every single IFC object, then the whole domain is equally subdivided along the three axes and the eight son-octants are created. This recursive process is repeated for each octant until the maximum depth of the tree is reached or further refining would lead to the distribution of a single bounding box to multiple octants.

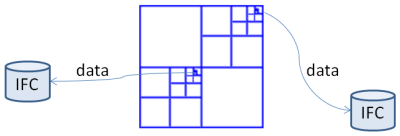

Figure 2: *Data access to single IFC models is organised by querying the first layer octree.*

With respect to the storage information at the time being we only take files of storage-type localy-stored into account but this will be expanded to network-stored files and is open to integrate further distributed repositories. To guarantee processing in real time we integrated pre-fetching of data to ensure seamless data-delivery. Pre-fetching is the process of loading data that are not yet requested for visualisation or processing in general but are already loaded in order to reduce latencies when they are requested. This is a highly dynamical process due to interactive user behavior and we have to define criteria for data to be requested by the user most likely and therefore pre-fetch them in advance. To fulfill this request we define the pre-fetching distance p_i, which describes the upper bound of the distance of an object to the user where it is prepared for visualisation. At a distance below p_i an object will be pre-fetched from the data storage, the appropriate level of detail is applied and its representation is cached. This will be further discussed in section 4.
Depending on the parameters of a scenario such as size of the BIM data and the underlying hardware, these values p_i have to be adjusted. In order to extend the efficiency of pre-fetching, we design the integration of heuristics for the analysis of user behaviour to our framework. Right now we are not taking the history of user-movements into account, instead decisions about pre-fetching are made solely based on the distance between the user and an object, whereas an probabilistic analysis of user behavior could lower the miss rate of objects that are not yet pre-fetched at the time they are requested.
As it is useful for the structure of this paper, aspects concerning the integration of parallelisation techniques are shifted to section 5.

Second Layer Octree

The second-layer octree stores a single construction or built infrastructure extracted from an IFC file at full detail and is computed on-the-fly for a construction or built infrastructure as soon as the distance between the construction and the user falls below p_i. Building up the second-layer octree is completely analogous to the buildup of the first-layer octree following the recursive process of spatial decomposition as described in the beginning of this section, since every triangle originating from the extraction of IFC objects can be associated with its bounding box which brings us back to the setting of the generation of the first-layer octree.
Since a building easily can consist of a huge number of triangles, further algorithms for reducing the amount of prmitives to be rendered are applied as shown in section 4. As described before each second-level octree holds all information about a single construction, therefore we can transfer our setting to a bride set of applications such as graph-theory and the computation of possible user movement graphs within the building to determine evacuation scenarios or solve shortest path problems [8], just to name a few.

4. LEVELS OF DETAIL (LOD) AND VISUALISATION TECHNIQUES

In this section, we describe the application of levels of detail (LoD) to single construction entities such as buildings, bridges, or streets. Depending on the distance of an object to the user different levels of detail are applied as the representation of a bridge, e. g., can be sufficient by displaying its bounding box at a high distance and varies to a representation with all details for a exploration at a very close distance. The application of LoDs is one necessary task for interactive visualisation, since it reduces the amount of objects to be rendered without an observable or at least acceptable loss of information and rendering all objects in fully detail would exceed the performance of any graphics hardware when it comes to certain ranges.

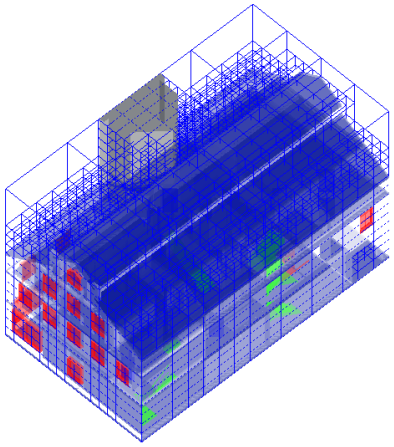

Figure 3: *Second-layer octree covering a single building.*

LoDs are applied to entities during run time and depend only on the distance between the user and the entity. We designed four LoDs and for every level $i = 0..3$ we define d_i as the distance of the level to become valid. Below this distance of a construction to the user the level of detail i becomes the level to be applied.
In level $i = 0$ information is delivered at full detail. Every aspect of an entity such as a window or even a screw is displayed or provided, this refers to the geometric representation as well as to auxiliary information. Since we integrated IFC as data storage format, this can be provided directly.
In level $i = 1$ we introduce a first level of approximation. At this level, single entities such as windows or doors are represented via their bounding-boxes. This can be applied, since from a certain distance the representation of the edging of a window for instance can't be distinguished from the approximation with its bounding box. This can be done easily by exploiting one of the strengths of IFC as described in [9] as follows. IFC doesn't only provide the geometric representation of a building and all its parts in full detail but also stores the mapping between geometry and function. This implies we directly query IFC to get the sets of windows, walls, or doors (with their exact geometric representation) of a building and then draw the bounding boxes for every single object.
In level $i = 2$ a construction is represented by a textured bounding box. At the time of reading an IFC file to our framework, the bounding box of the construction is computed and stored, such that a texture provided with the IFC file can be mapped onto the given bounding box. One of the future tasks in the project can be to automatically generate the texture of a construction by evaluating the IFC directly when the file is loaded into the framework.
In level $i = 3$ the bounding box of the whole construction is used and sufficient for approximating the construction or built infrastructure. Entities such as bridges, pipelines, or streets are represented by a single line.

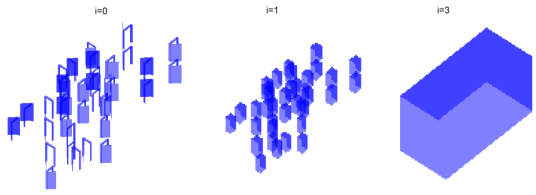

Figure 4: *Extracted doors as an example for step-wise coarsening along different levels of detail.*

After the application of levels of detail, the amount of objects to be rendered can be reduced further by applying algorithms that remove objects that are invisible to the user. For a deeper look into detail we refer the reader to [5]. In a first step we limit the set of objects to be considered to those lying in the visible area of the user. This property can be evaluated efficiently by iterating through the nodes of the octree (see section 3). Hence objects are excluded that are lying behind the user or objects that would be projected to points outside the clipping area.

The next algorithm applied is back face-culling for solids, which we can rely on since all geometric objects retrieved from IFC are solids. Back face-culling removes all faces of a solid which are invisible to the user since they are located at the backside of a solid.
After these steps the Z-buffer algorithm is applied to the remaining objects in order to remove faces that are hidden by other faces.

5. INTEGRATION OF PARALLELISATION TECHNIQUES

In this section, we describe the integration of parallelisation techniques, which are crucial for fulfilling our demands of seamless data delivery. Since all reasonable hardware platforms these days are multi-core systems the distribution of the workload over several processes has to be implemented in order to exploit the full performance of the underlying hardware.
Our application is intended to be run on a multi-sided CAVE, a fully immersive visualisation environment providing stereo-enabled viewing on multiple projection planes. The underlying rendering hardware is implemented as a multi-core shared-memory system and therefore we decided to use OpenMP [11] for parallelising computations. OpenMP is a memory-coupled interface providing a set of compiler directives and functions delivered within a runtime library for parallelising applications by distributing portions of the code to be executed in different threads.
We integrate parallelisation to our framework using a function-parallelism approach to separate the visualisation from the data processing and enable the parallel processing of large data-sets. About the distribution of threads we designed the system to use one visualisation thread, as we are using OpenGL which doesn't support the direct parallel rendering of primitives nor it would lead to an enhancement of performance. Considering a total amount of N threads or processor-cores to be available, the rest, i. e. $N-1$, of them are spent for data-processing.
Decoupling the visualisation and data-processing ensures the seamless delivery of information even if information needed is not processed yet when for example switching an object from a coarser to a finer level of detail, the coarser level is displayed until the finer level of detail is processed. This prevents the system from insufficient frame rates. As soon as the data is processed at the finer level of detail it is displayed in the next iteration of visualisation.
To synchronise the single processes we designed a data structure to which the data-processing threads write the data prepared for displaying and from which the visualisation thread reads the data and renders it.
The data structure for the distribution of the data-processing is designed as a first-in-first-out (FIFO) queue whereas the accesses to the queue is synchronised using a critical section. When the user changes the position or the viewing angle, these new parameters are written to the queue such that the data-processing threads can update the geometric representation to be rendered by the visualisation thread. As soon as the change of the viewing angle or the position of the user is entered to the queue, the next free data-processing thread pops the information about the change of the viewing parameters from the queue, checks for the IFC objects that have to be added for visualisation, removed from visualisation or whose levels of detail have to be changed and writes those tasks individually back to the queue whereas those tasks are ordered ascending in the distance of the IFC object to the user such that closer objects are updated first. The tasks of changing the representation of a single IFC object are then executed by the data-processing threads.
For the application of a data-parallel approach, the synchronisation and organisation of different threads executing an equal set of instructions to a decomposition of the data has to be taken into account. Due to its structure the octree directly delivers a decomposition of the data to distribute them to the single threads. When a task for fulfilling the processing of an IFC object is received by the next free data thread, it decomposes the domain for the task into chunks of size c and enters the tasks for the data-processing back to the FIFO queue. Those tasks will then be finally processed sequentially by the data-processing threads. Using this decomposition the generation of the first-layer and second-layer octrees described in section 3 can be data-parallelised over the generation of the octants, analogously the second-layer octree can be parallelised when applying geometry reduction algorithms described in section 4.
The data structure for the storage of the geometric representation of the IFC objects is implemented as a set of triples, containing the identification of the

IFC object, the actual level of detail and a pointer to the geometric representation whereas the access to this set is synchronised using a critical section. If an IFC object is to be removed from visualisation, first the entry from the set is removed and then the memory allocation holding the geometric representation is freed. If an IFC object is to be added to the visualisation, first the geometric representation is stored and then the pointer to the representation is added to the set. If the level of detail for an IFC object is to be changed, the representation for the new level of detail is generated. Afterward the pointer to the geometric representation and the value indicating the actual level of detail is updated and the memory allocated for the geometric representation of the former level of detail is freed.
Using this data structure we can ensure a seamless visualisation with sufficient frame rates.
At the time being we focuse on running our framework on a four-core Intel-based hardware and distribute those 4 threads to work with one visualisation thread and three data-processing threads.
The structure of the given parallelisation concept is given in the following figure.

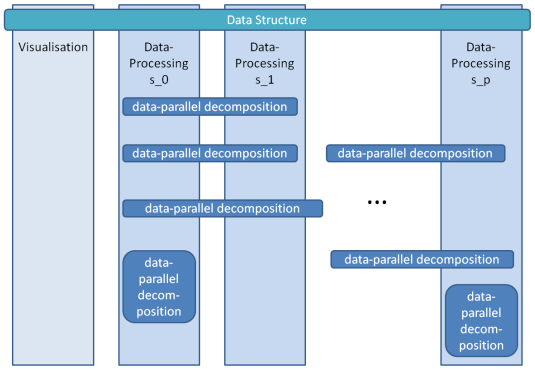

Figure 5: *Parallelisation diagram following a function-parallel approach for the separation of the visualisation thread from the data-processing threads and a data-parallel approach for the distribution of decomposed data over multiple data-processing threads.*

6. RESULTS

In order to show the benefits of our approach, a prototypical implementation was done. First results, still obtained with the sequential algorithm, are already very promising and reveal the potential of the final fully parallelised code.
To set up a benchmark, for our framework a scenario that covers a highly detailed textured height map [1] of the country side area close to Vorarlberg in Austria with a resolution of 2048 x 2048 pixels for the texture and a resolution of 512 x 512 pixels for the height map was defined. The first-layer octree is computed up to a maximum height of eight levels, whereas the second-layer octree is computed up to a maximum height of four levels. As IFC models we use different models of a complexity varying from 53,000 to 159,000 vertices.
The given results have been achieved on an Intel Q9550 Architecture with 4GB main memory and an NVIDIA Quadro FX 570 GPU installed.
As we are interested in the development of a framework that gives the user a tool at hand for seamless data exploration, we focus on the resulting frame rate achieved by the framework. For an exploration of the terrain where most of the parts of the building are not lying in the focus of the user we achieve a frame rate of approximately 60fps, during the exploration of a building and its details, the frame rate drops down to 20-30fps as the data-accessing and -processing is a highly dynamically process, nevertheless still high enough for an interactive application.

[1] provided by the Chair for Geoinformation Systems at Technische Universität München

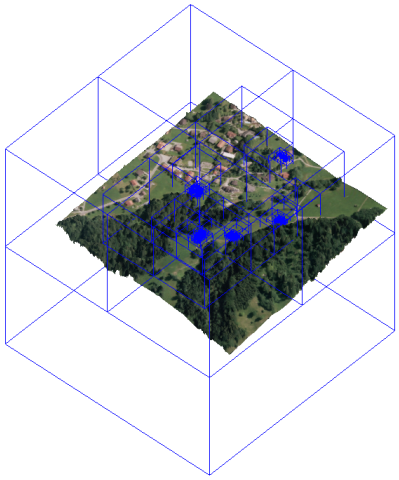

Figure 6: *The applied terrain loaded with IFC objects arranged with the octree.*

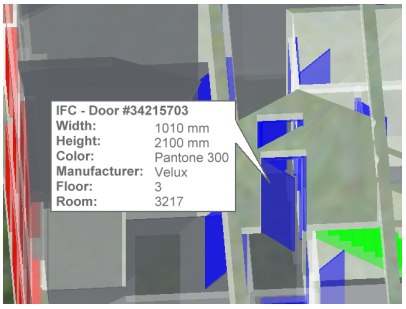

Figure 7: *Exploring details of a building.*

7. CONCLUSIONS AND OUTLOOK

In this paper, we presented an approach for a framework to give the user a tool at hand which combines the handling of vast amounts of construction and built infrastructure data with the seamless delivery of information and provision of immersive data exploration. This gives the user the possibility to gain insight about information ranging from the scale of global overview to highly specific information of single construction entities on the level of screws. All this information delivery had to be kept in the balance between a high grade of detail and a perceptible flow of information.
This framework uses mostly well known techniques of information processing and combines them in a new way of handling data of constructions and built infrastructure beyond today established frameworks.
Future work will comprise the integrating of heuristics for pre-fetching and the calibration of data-parallelisation. Hence, this framework gives the possibility of evaluating queries of the type "Which building within 20 miles from my position has an air condition system installed for a room of more than 1000sf usable space?", which are not possible with nowadays systems.

8. ACKNOWLEDGEMENTS

This publication is based on work supported by Award No. UK-c0020, made by King Abdullah University of Science and Technology (KAUST). Furthermore the work in this project is supported by the International Graduate School of Science and Engineering of the Technische Universität München.

9. REFERENCES

[1] M. Alshawi, G. Aouad, I. Faraj, T. Child, and J. Underwood. **The implementation of the industry foundation classes in integrated environments**. *CIB W78 Conference*, pages 55–66, 1998.

[2] J. Benner, A. Geiger, and K. Leinemann. **Flexible Generation of Semantic 3D Building Models**. *Proc. of the 1st Intern. Workshop on Next Generation 3D City Models*, 2005.

[3] B. Björk. **The RATAS project — an example of co-operation between industry and research toward computer integrated construction.** *Journal of Computing in Civil Engineering, ASCE*, page 401–419, 1994.

[4] I. Faraja, M. Alshawi, G. Aouad, T. Child, and J. Underwood. **An industry foundation classes Web-based collaborative construction computer environment: WISPER** . *Automation in Construction*, 10:79–99, 2000.

[5] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. **Computer Graphics: Principles and Practice in C, 2nd Edition**. *Addison-Wesley Professional*, 1995.

[6] T. Froese and K. Yu. **Industry Foundation Class Modeling for Estimating and Scheduling** . *8th International Conference on Durability of Building Materials and Components*, 1999.

[7] A. Hammad, C. Zhang, and Y. Hu. **Mobile Model-Based Bridge Life Cycle Management**. *Proceedings of the 8th International Bridge Management Conference*, page I–5/1–13, 2006.

[8] C. S. Han, J. C. Kunz, and K. H. Law. **Building Design Services in a Distributed Service Architecture**. *Journal of Computing in Civil Engineering, ASCE*, 13, 1999.

[9] IFC2x3 TC1 Release. **IFC Specification**. *http://www.iai-tech.org/products/ifc_specification/ifc-releases/ifc2x3-tc1-release*.

[10] M. Kluth, A. Borrmann, E. Rank, T. Mayer, and P. Schiessl. **3D Building Model-Based Life-Cycle Management of Reinforced Concrete Bridges**. *Proc. of the 7th European Conference on Product and Process Modelling (ECPPM'08). Sophia-Antipolis, France*, 2008.

[11] OpenMP. **The OpenMP API specification for parallel programming**. *http://openmp.org/*.

[12] TNO Building and Construction. **IFC Engine** . *http://www.ifcbrowser.com/*.

[13] P. van Oosterom, S. Zlatanova, and E. M. Fendel. **Geo-information for Disaster Management**. *Springer Berlin Heidelberg*, 2005.