Visualisation and Generalisation of 3D City Models

Bo Mao

Doctoral Thesis
Abstract

3D city models have been widely used in various applications such as urban planning, traffic control, disaster management etc. Efficient visualisation of 3D city models in different levels of detail (LODs) is one of the pivotal technologies to support these applications. In this thesis, a framework is proposed to visualise the 3D city models online. Then, generalisation methods are studied and tailored to create 3D city scenes in different scales dynamically. Multiple representation structures are designed to preserve the generalisation results on different level. Finally, the quality of the generalised 3D city models is evaluated by measuring the visual similarity with the original models.

In the proposed online visualisation framework, City Geography Makeup Language (CityGML) is used to represent city models, then 3D scenes in Extensible 3D (X3D) are generated from the CityGML data and dynamically updated to the user side for visualisation in the Web-based Graphics Library (WebGL) supported browsers with X3D Document Object Model (X3DOM) technique. The proposed framework can be implemented at the mainstream browsers without specific plugins, but it can only support online 3D city model visualisation in small area. For visualisation of large data volumes, generalisation methods and multiple representation structures are required.

To reduce the 3D data volume, various generalisation methods are investigated to increase the visualisation efficiency. On the city block level, the aggregation and typification methods are improved to simplify the 3D city models. On the street level, buildings are selected according to their visual importance and the results are stored in the indexes for dynamic visualisation. On the building level, a new LOD, shell model, is introduced. It is the exterior shell of LOD3 model, in which the objects such as windows, doors and smaller facilities are projected onto walls. On the facade level, especially for textured 3D buildings, image processing and analysis methods are employed to compress the texture.

After the generalisation processes on different levels, multiple representation data structures are required to store the generalised models for dynamic visualisation. On the city block level the CityTree, a novel structure to represent group of buildings, is tested for building aggregation. According to the results, the generalised 3D city model creation time is reduced by more than 50% by using the CityTree. Meanwhile, a Minimum Spanning Tree (MST) is employed to detect the linear building group structures in the city models and they are typified with different strategies. On the building level and the street level, the visible building index is created along the road to support building selection. On facade level the TextureTree, a structure to represent building facade texture, is created based on the texture segmentation.
Different generalisation strategies lead to different outcomes. It is critical to evaluate the quality of the generalised models. Visually salient features of the textured building models such as size, colour, height, etc. are employed to calculate the visual difference between the original and the generalised models. Visual similarity is the criterion in the street view level building selection. In this thesis, the visual similarity is evaluated locally and globally. On the local level, the projection area and the colour difference between the original and the generalised models are considered. On the global level, the visual features of the 3D city models are represented by Attributed Relation Graphs (ARG) and their similarity distances are calculated with the Nested Earth Mover’s Distance (NEMD) algorithm.

The overall contribution of this thesis is that 3D city models are generalised in different scales (block, street, building and facade) and the results are stored in multiple representation structures for efficient dynamic visualisation, especially for online visualisation.

**Keywords:** 3D city models, visualisation, generalisation, multiple representation structure, similarity evaluation, aggregation, typification, shell model, street index, texture compression, texture segmentation.
List of papers

I: Bo Mao, Yifang Ban, 2011. Online Visualisation of a 3D City Model Using CityGML and X3DOM, Cartographica, Vol. 46, No. 2, pp. 109-114


# Table of contents

1. Introduction .................................................................................................................. 1
   1.1 Background .............................................................................................................. 1
      1.1.1 Applications of 3D city models ................................................................. 1
      1.1.2 3D city modelling and management ....................................................... 4
      1.1.3 Visualisation of 3D city models ............................................................... 6
      1.1.4 Generalisation of 3D city models ............................................................ 7
   1.2 Research objectives ............................................................................................... 8
   1.3 Thesis organisation .............................................................................................. 9

2. Literature review ........................................................................................................... 11
   2.1 Representation of 3D city models ................................................................. 11
      2.1.1 CityGML .................................................................................................... 11
      2.1.2 Other standards ........................................................................................ 15
   2.2 Visualisation of 3D city models ................................................................. 17
      2.2.1 Standards for 3D visualisation .............................................................. 17
      2.2.2 Visualisation frameworks .................................................................... 21
   2.3 Generalisation of 3D city models ................................................................. 32
      2.3.1 Geometry generalisation ..................................................................... 32
      2.3.2 Texture generalisation ........................................................................ 39
      2.3.3 Generalisation evaluation .................................................................. 40
      2.3.4 Multiple representations of 3D city models .................................... 44

3. Methodology ................................................................................................................. 47
   3.1 Overall structure ............................................................................................... 47
   3.2 Data integration ................................................................................................. 49
      3.2.1 Integration strategy .............................................................................. 49
      3.2.2 CityGML ................................................................................................ 50
      3.2.3 X3D ......................................................................................................... 52
      3.2.4 KML/COLLADA .................................................................................. 55
   3.3 Online visualisation .......................................................................................... 56
   3.4 Block level generalisation .............................................................................. 58
      3.4.1 Neighbourhood structure ................................................................. 58
### List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DMLW</td>
<td>3D Markup Language for Web</td>
</tr>
<tr>
<td>ADE</td>
<td>Application Domain Extensions</td>
</tr>
<tr>
<td>AJAX</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ARG</td>
<td>Attributed Relation Graph</td>
</tr>
<tr>
<td>BIM</td>
<td>Building Information Modelling</td>
</tr>
<tr>
<td>BLG-tree</td>
<td>Binary Line Generalisation tree</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CityGML</td>
<td>City Geography Makeup Language</td>
</tr>
<tr>
<td>COLLADA</td>
<td>COLLaorative Design Activity</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>EMD</td>
<td>Earth Mover’s Distance</td>
</tr>
<tr>
<td>GAP-tree</td>
<td>Generalised Area Partitioning tree</td>
</tr>
<tr>
<td>GML</td>
<td>Geography Markup Language</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Process Units</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>IFC</td>
<td>Industry Foundation Classes</td>
</tr>
<tr>
<td>ISO</td>
<td>International Organisation for Standardisation</td>
</tr>
<tr>
<td>KML</td>
<td>Keyhole Markup Language</td>
</tr>
<tr>
<td>LOD</td>
<td>Level of detail</td>
</tr>
<tr>
<td>MBR</td>
<td>Minimum Boundary Rectangle</td>
</tr>
<tr>
<td>MST</td>
<td>Minimum Spanning Tree</td>
</tr>
<tr>
<td>NEMD</td>
<td>Nested Earth Mover’s Distance</td>
</tr>
<tr>
<td>O3D</td>
<td>Open 3D</td>
</tr>
<tr>
<td>OGC</td>
<td>Open Geospatial Consortium</td>
</tr>
<tr>
<td>OpenGL</td>
<td>Open Graphics Library</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer to Peer</td>
</tr>
<tr>
<td>U3D</td>
<td>Universal 3D</td>
</tr>
<tr>
<td>VRML</td>
<td>Virtual Reality Modelling Language</td>
</tr>
<tr>
<td>W3DS</td>
<td>Web 3D Service</td>
</tr>
<tr>
<td>WebGL</td>
<td>Web Graphics Library</td>
</tr>
<tr>
<td>WPVS</td>
<td>Web Perspective View Service</td>
</tr>
<tr>
<td>X3D</td>
<td>Extensible 3D</td>
</tr>
<tr>
<td>X3DOM</td>
<td>Extensible 3D Document Object Model</td>
</tr>
<tr>
<td>XSLT</td>
<td>Extensible Stylesheet Language Transformations</td>
</tr>
</tbody>
</table>
List of figures

Figure 1.1: 3D city model related applications ................................................ 3
Figure 2.1: The five LODs defined by CityGML ................................................. 14
Figure 2.2: Relationship of VDI 3805, IFC and CityGML ................................. 16
Figure 2.3: Rendering phases and graphics resources ........................................ 21
Figure 2.4: Test topographic map of Shah Alam city for 3D building generation .................................................................................................................. 23
Figure 2.5: Workflow scheme of X3D model generation .................................... 24
Figure 2.6: The system architecture of the virtual 3D city model of Berlin ...... 25
Figure 2.7: The architecture of a web 3D service ................................................ 26
Figure 2.8: 3D city data exchange framework .................................................... 27
Figure 2.9: Overall architecture of the CityServer3D system ............................. 29
Figure 2.10: Generalised 3D building model ...................................................... 33
Figure 2.11: Direct merge ..................................................................................... 34
Figure 2.12: Snap merge ..................................................................................... 34
Figure 2.13: 2D building projections .................................................................. 35
Figure 2.14: Aggregation of 3D buildings ......................................................... 36
Figure 2.15: Typification based on Kohonen Net .............................................. 37
Figure 2.16: Example of P-tree ......................................................................... 44
Figure 2.17: Hierarchical structure of 3D urban block geometry ...................... 45
Figure 2.18: Structure of the texture-atlas tree ................................................. 46
Figure 2.19: Structure of the texture quad-tree ............................................... 46
Figure 3.1: Overall structure of the thesis .......................................................... 47
Figure 3.2: Data integration framework ............................................................. 49
Figure 3.3: CityGML surface example ............................................................... 50
Figure 3.4: CityGML texture ............................................................................. 51
Figure 3.5: Positioning of textures using texture coordinates .......................... 51
Figure 3.6: Elevationgrid Node in X3D .............................................................. 52
Figure 3.7: IndexedFaceSet Node in X3D ........................................................... 53
Figure 3.8: IndexedFaceSet node with texture .................................................. 54
Figure 3.9: An example of the file structure used in Blom 3D city models ...... 55
Figure 3.10: Structure of a Blom 3D model ................................................................. 55
Figure 3.11: Moving from a loose plugin-based Scene-Access-Interface (SAI) to the tightly integrated X3DOM model ................................................................. 56
Figure 3.12: X3D in HTML5 ..................................................................................... 57
Figure 3.13: Texture images of 3D building models .................................................. 64
Figure 3.14: Two-step adjustment of irregular-shaped facade images ...................... 65
Figure 3.15: Original texture image for an irregular-shaped facade ....................... 65
Figure 3.16: Visualisation of mean shift-based filtering and segmentation for grey-level data .............................................................................................................. 66
Figure 3.17: Texture image segmentation using Mean shift algorithm1 ................ 67
Figure 3.18: 3D Pavement art .................................................................................... 68
Figure 3.19: Estimated probability of correct node-to-node matching as noise increases ......................................................................................................................... 70
Figure 4.1: The framework for online visualisation of 3D city models (Paper I) .... 72
Figure 4.2: CityTree model (Paper II) ..................................................................... 74
Figure 4.3: Work flow of our generalisation strategy in Paper II .......................... 75
Figure 4.4: Work flow chat of dynamic typification .............................................. 76
Acknowledgements

I am grateful to my supervisors, Prof. Yifang Ban, KTH and Dr. Lars Harrie, Lund University, for their constant support, instructions, guidance and encouragement during the research and daily life during my studies in Sweden.

Thanks to all of the staff and PhD students at the Geoinformatics Division, KTH. The united international team has given me inspiration and feelings of home. Thanks to Hongchao Fan for the fruitful collaboration during the thesis research.

I would like to thank the China Scholarship Council and the School of Architecture and Built Environment at KTH for providing the scholarships. I also thank the school of Computer Science and Engineering, Southeast University, China for the chance to study abroad.

This work has been carried out as part of the ViSuCity research project (PI: Prof. Yifang Ban) funded by the Knowledge Foundation, Swedish Foundation for Strategic Research, Vinnova, the Swedish Foundation for Health Care Sciences and Allergy Research, and Invest in Sweden.

Finally, I would like to thank my family and my friends for their support all along.

Bo Mao
Stockholm, October 2011
1. Introduction

1.1 Background

An increasing amount of people are living in cities, the home of more than half of the world’s population. By 2030, the number will be close to 5 billion (United Nations 2008). Therefore, it is essential to develop efficient techniques to assist the management of modern cities.

3D city models are the basis of many applications and are the platform for integrating city information from different resources. 3D city models make it easier for people to understand the spatial properties of urban objects, since the real world we live is in 3D and it is natural for the human brain to interpret 3D scenes. Many companies utilise 3D city models to attract more users by enhancing expressiveness and simplifying the operations of their applications. Research on 3D city models is becoming a hotter topic in both academia and industry. Many algorithms and methods are developed to deal with challenges of 3D city models, and efficient visualisation is one of the most critical issues. Generalisation methods can be applied to 3D city models not only to improve the visualisation efficiency by reducing the data volume but also to emphasize the objects users are interested in.

This section contains three parts about the 3D city models: applications, modelling and management, the state-of-art of visualisation and generalisation methods.

1.1.1 Applications of 3D city models

3D city models have already been widely employed to assist different applications such as urban planning, traffic control, mobile telecommunication design, etc. Therefore, many cities are creating and releasing their official 3D models. For example, 3D Berlin is available to the public since March 2009. It includes a total of 474,000 fully textured buildings spanning an area of 857 km$^2$. This official Berlin model can be viewed through Google Earth and is available in CityGML format (Kada 2009). The official Berlin 3D city model creates a basis to gather, integrate and publish existing geoinformation and a framework for specialist and non-specialist members of the public to take part in planning and decision-making processes. Based on these 3D city models, some related application areas are listed as follows.

Urban planning: Landscape planners need 3D city models to visualise the impact of newly proposed projects on the city environment (Sadek et al. 2002, Chen, 2011). Murata (2004) developed a 3D-GIS application prototype to
visualise the existing state of cities and to perform simulations of district development plan. In their application, the high-resolution 3D city model is combined with the 2D urban planning database for spatial analysis. The ViSuCity project (Ban et al. 2008) developed an effective web-based, interactive 5D (3D+time+sustainability) visualisation demonstrator, ViSuCity, to support sustainable city planning in terms of information sharing, analysis, development, presentation and communication of ideas and proposals throughout the city planning processes. Parameters relevant for an integrated sustainable city planning, such as transportation system, energy, water and waste management, green structure, etc will be integrated to enhance the quality of both the planning process and the planning results. Lamberti et al. (2011) use 3D city models to improve the street lighting managers in the identification of cost-effective illumination strategies complying with safety and security constraints.

**Disaster management:** Public safety agencies such as fire department and emergency medical services need 3D city models to help them locate the available fire response equipment, manage the local transportation infrastructure in the case of an emergency, and settle down the evacuated peoples, etc. (Tang et al. 2006). Another example is Schulte and Coors (2008) who implemented a flood visualisation system based on 3D city models to illustrate the flood emergency.

**Spatial analysis:** Space managers are interested in understanding the form, function, assignment, and availability characteristics of their space in 3D. Dehghan and Steele (1997) discussed the necessity of using 3D city buildings to act as electromagnetic models that define the size and shape of each 3D cell to form a mobile radio network. Ming et al. (2002) employed 3D city models to design a closed-circuit television monitoring system. Hermsmeyer et al. (2005) also used 3D city models to plan telecommunication antenna networks for highly developed inner-city areas of Kuwait. The commercial Real Estate community needs 3D city models to demonstrate their product not only at individual room level but also at the suite, building or surrounding area level. Researchers from Germany applied 3D city models to analyse and visualise the noise in North Rhine-Westphalia area (Czerwinski et al. 2007). Peng et al. (2009) and Dawood et al. (2011) made use of 3D city models for vehicle geolocalisation when the GPS information is not available or not accurate enough.

**Public affairs:** Public administration agencies need 3D city models to regulate land use entitlement or to administer local taxation policy. For example, the local taxes that are paid in Singapore are partly determined by the availability of sunlight in the occupied space and the impact of sunlight availability on the surrounding area (Rich 2009). The facilities, energy and public health management agencies could do a better job with the help of 3D city models. 3D
city models are also widely used in location marketing, tourism and etc. (Schulze-Horsel 2007). Klimke and Döllner (2010) presented an approach for combining spatially distributed synchronous and asynchronous collaboration within 3D city models. Figure 1.1 demonstrates a snapshot about some examples of existing or on-going applications with the 3D city models mentioned above.

Figure 1.1: 3D city model related applications (Source: Kolbe and Rönsdorf 2008, pp. 8).

In the foreseeable future, 3D city models will play an increasingly important role in our daily lives and become an essential part of the modern city information infrastructure (Spatial Data Infrastructure). Similar to the 2D cartographic maps, the 3D city models will be used to integrate various data from different sources for public accessible visualisation and many other applications.
1.1.2 3D city modelling

The implementation of 3D city models is based on many techniques that can be classified into two categories: general computer technologies and 3D city modelling

Related computer technology

With the rapid development in computational capacities, computer graphics, and web technologies, it is possible to create detailed 3D city models that can be simultaneously accessed by thousands of people from all over the world.

First, computer devices become increasingly powerful. The famous Moore’s Law (Moore 1965) indicates that the number of transistors that can be placed inexpensively on an integrated circuit has doubled approximately every two years. This trend has continued for more than half a century and is not expected to stop until 2015 or 2020 or later (Kanellos 2005). Nowadays not only super computers, but also common PCs have enough capacity to store and process huge volumes of data. Modern Graphics Process Units (GPUs) are used to perform and accelerate calculations related to 3D computer graphics. Mobile devices, e.g. iphone, are also getting better and smarter, which makes mobile 3D city models possible (Mobile 3D City 2009).

Second, computer graphics technologies are becoming more sophisticated. From the Hollywood movies to the PS3 video games, the virtual environments created by artists with modern computer graphics technologies are of such quality that people can hardly identify them from the real world. The process of creating 3D computer graphics can be sequentially divided into three basic phases:

1. **3D modelling**: the process of forming the shape of an object. 3D models especially the city or building models are now created both by man hands with tools e.g. Computer Aided Design (CAD) software and automatically from real world objects.

2. **Layout and animation**: the motion and placement of objects within a scene. The 3D animation can be performed by motion capture system or simulated by physics engine.

3. **3D rendering**: producing an image of an object. 3D rendering can be easily performed by 3D computer graphics software or 3D Graphics Application Programming Interface (API) such as OpenGL (2011) or DirectX (2011). Meanwhile, the image processing algorithms can be applied in analysing the texture of 3D city models.
Third, the web technologies make it possible for anyone to access online 3D city models at anytime from anywhere. According to the Nielsen's Law of Internet Bandwidth (Nielsen 1998), a high-end user's connection speed grows by 50% per year. With the development of wireless networks such as 3G/4G and Wi-Fi, the Internet can be accessed from anywhere in high speed. Web services and cloud computing technologies supply access standards and computation power for web 2.0 applications such as Facebook, Twitter or YouTube which contain extremely amount of content created by huge number of users. 25.6% of the world population, 1.7 billion, is using the Internet by September 2009 (Internet World Stats 2009) and 1 trillion unique URLs were found by Google in July 2008 (Google 2008). The web is also going to the 3D. Currently, it is possible to display 3D content through common web browsers by installing certain plugins. HTML5, the next major revision of Hypertext Markup Language (HTML), will support video and 3D directly. Apple’s iPad adopted HTML5 instead of Adobe Flash for its multimedia, which shows the power and potential of HTML5. Meanwhile professional tools such as Google Earth, Virtual Earth and ArcGIS Explorer also start supporting online 3D city models.

Besides the above mentioned, other computer technologies such as databases are also useful in some applications of 3D city models. These computer technologies create the underlying platform to support 3D city models related applications. Based on that, the studies on the 3D city models are carried out.

**3D city modelling and management**

*3D city model creation:* First of all, 3D city models have to be created before we can use them. Most of current 3D city models are created manually. Although there are a lot of tools to support the process, they still can not satisfy the needs for modelling large city area. Nowadays, an increasing amount of cities employ Lidar, digital camera and photogrammetric technologies to quickly generate textured 3D city models over large areas (Deng et al. 1995, Frueh 2002, Frueh and Zakhor 2003, Xiao et al. 2009). 3D city models can also be converted from existing Building Information modelling (BIM) in CAD files (Döllner and Hagedorn 2007) or from the OpenStreetMap (Over et al. 2010).

*3D city model management:* Since the 3D city models come from several sources, a standard is required to describe, store and exchange 3D city models. That is why CityGML (Kolbe et al. 2005, OGC 2008) was designed. CityGML, issued by OGC in August 2008, is a common information model for the representation of 3D urban objects. It contains both geometric and semantic information. Because of the huge data amount of 3D city models, databases such as Oracle and PostgreSQL can be used to store and manage the data. Some databases support spatiotemporal data type or even CityGML schemas, e.g.
Oracle 11g. To deal with even larger data volume, cloud computing can be a solution.

*Semantic information of 3D city models:* Many applications such as urban planning, facility management and personal navigation require semantic information about the city objects besides the geometry models. It is essential to develop methods for 3D city model semantic information modelling, representation, discovering, management, querying and analysis (Kolbe 2008).

Other 3D city model technologies: Besides the above issues, there are several other 3D city model related technologies needed to be taken into consideration. They are:

1. **Analysis methods based on 3D city models:** For example: line of sight analysis, noise modelling and 3D addressing/routing.
2. **Mobile deployment:** Many users would benefit from the ability to access a 3D city model in the mobile environment. This capability would be particularly useful to the security, public safety, facilities management, environmental quality, and municipal inspection communities.
3. **Web publishing:** Since accessibility is very important to the 3D city model, we should utilize these models as services published over the web. Given the complexity of some 3D city models, server performance and network bandwidth constraints are significant challenges.

### 1.1.3 Visualisation of 3D city models

Most of the 3D city models related applications require efficient methods to visualise 3D models in various scales. This thesis focuses on visualisation of multi-scale 3D city models derived from generalisation methods.

Visualisation is a complex and important issue in 3D city model applications. These applications have some common visualisation requirements listed as follows.

**Visualisation of various LODs:** It is required to visualise the urban environment in different scales, e.g. from overview scale like a region down to detailed scale like a building or even a room. This kind of zooming is interesting for many applications e.g urban planning.

**Visualisation of changes over time:** It is necessary to view the status of the city models at different times to find out the changes. The temporal information of each object is stored to create a 4D (3D+time) city model which would be very useful in some applications such as urban planning (Ban *et al.* 2011).
Online visualisation of 3D city models: Internet has become a basic information infrastructure all over the world. Therefore, it is necessary to develop methods to visualise 3D city models through the Internet. Currently, many plugins of Internet browsers have been developed to display 3D scenes such as Adobe Flash, Microsoft Silverlight and Java3D etc. But sometimes 3D content can not be opened since the user did not install the right plugins. Therefore, the accessibility of 3D scenes based on specific browser plugins is not satisfactory. To deal with the problem, HTML5, the new coming basic standard of the Internet, defines X3D to present the 3D content in the web. Although X3D content is not supported by all web browsers directly right now, it is the trend that in the future X3D elements will be able to be directly manipulated by the mainstream browsers utilizing the DOM technique (Behr et al. 2009). Researchers have already started to develop methods to support X3D scene visualisation and manipulation in web browsers without plugin installation.

Challenges of 3D city model visualisation include how to create the 3D scenes for multiple platforms through internet and how to automatically generate the multiple representations for different scales. In thesis, a framework based on X3D and HTML5 is proposed to visualise the 3D city models online for multiple platforms. Further, generalisation methods are designed to create multiple representation structures of 3D city models for different scales.

1.1.4 Generalisation of 3D city models

For effective visualisation, generalisation is essential to reduce the data volume and improve the visual presentation. Generalisation stems from cartography and is the method whereby information is selected and represented on a map in a way that adapts to the scale and the purpose of the map, (Ruas 2000). Traditionally, cartographic generalisation is carried out manually. The experts from cartography create generalised maps in different scales based on their experience. However, with development of GIS related technologies such as Global Positioning System (GPS), and remote sensing, the volume of raw data especially for 3D city models is increased dramatically. These data are also required to be updated constantly or even in real-time. It is impossible to manually maintain real-time updated multi-scale representations for such amount of data for a city, not to mention for a country. Therefore, the generalised methods discussed in this thesis are always carried out by computer instead of humans.

Many researchers have paid attention to the generalisation of 3D city models. Different LODs are proposed to decrease the complexity of 3D models (OGC 2008). Automatic generalisation methods are used to create 3D city model in lower LODs from higher LODs and visualise a large region of urban with a
reasonable amount of data, which is very important for online 3D city models. Several algorithms have been proposed for generalisation of 3D city models for buildings, roads etc. These algorithms not only reduce the data amount of the models, but also try to extract and enhance the interesting parts.

However, most of the existing 3D city model generalisation methods focus on a signal building, and the generalisation of building groups are rarely studied. This thesis presents three generalisation methods for building groups: aggregation, typication and selection.

Texture is another important aspect of the 3D city models. Textured models look more realistic and give better visualisation impressions. It is necessary to use textured 3D building models especially in street level view applications such as navigation. Despite many advantages, the main problem of the textured model is the large data volume. Therefore, simplification of 3D textured building models is required for efficient and fast visualisation.

Since the generalised 3D city models are automatically created, it is essential to assess the quality of these generalised models. The quality of the generalised 3D city model is determined by the quality of both original 3D models and the generalisation process (Harrie 2001). Because the evaluation of generalisation quality mainly relies on the visual assessment of the results (Weibel and Dutton 1999), this thesis also deals with the visual similarity between generalised models and the original ones. The results are evaluated by both user survey and pattern recognition algorithms. The evaluation is not only based on the 3D city model itself but also based on its projection in 2D view.

1.2 Research objectives

The overall objective of this research is to develop a framework which can efficiently visualise multi-scale 3D city models from block level, building level to facade level. Specifically, the proposed framework should be able to deal with textured models and import the data in standard formats such as CityGML and Keyhole Markup Language (KML). It should also generate the suitable 3D city scenes in varying LODs according to the current user view point and recreate the scenes dynamically as user view point changes. The created scenes should be interactively viewed by standard web browsers without specific plugins.

The primary scientific questions in the thesis are:

(1) How to represent and visualise 3D city models efficiently?
(2) How to automatically generate 3D city models in different LODs and implement dynamic visualisation of 3D city models in multi-scale?
(3) How to assess the visual quality of generalised 3D city models?
Accordingly, the specific research objectives are:

1. Create online 3D scenes with international 3D standard X3D from the standard of the 3D city models, CityGML.
2. Study current cartographic generalisation methods and propose new algorithms to support 3D city model generalisation in block, building and textured facade levels.
3. Propose suitable structures to represent the visual feature of 3D city models for continuous scaling, and evaluate the similarity between generalised models and original ones.

This research has been carried out as part of the research project “ViSuCity” (Ban et al. 2011). This project contributes to a faster planning process with improved quality and enhanced sustainability, and is expected to introduce interactive 3D visualisation as a de facto standard tool in planning and decision processes for municipal, regional and national projects.

1.3 Thesis organisation

This thesis is structured as follows: Chapter 1 gives an overview of the research, including the background, research objectives, and organisation of the thesis. Chapter 2 reviews the related work about 3D city models, such as CityGML, 3D visualisation standards and generalisation methods. Chapter 3 describes the methodologies of the thesis including data integration, online visualisation, generalisation algorithms and evaluation methods. Thereafter, Chapter 4 presents the system implementation details and experiment results in each paper. Chapter 5 concludes the research and explores future research directions.

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.

I: Bo Mao, Yifang Ban, 2011. Online Visualisation of a 3D City Model Using CityGML and X3DOM, Cartographica, Vol. 46, No. 2, pp. 109-114


In paper I, the authors propose a framework of online visualisation of 3D city models. In this paper, CityGML is selected for 3D city models representation. The CityGML file is converted to an X3D scene which is viewed online using X3DOM (Objective 1). In paper II the authors propose a multiple representation data structure CityTree to represent building group and implemented the dynamic building aggregation (Objective 2). In paper III, a novel method for detection and typification of linear structures in the 3D city models is proposed. MST is employed to detect the linear structures, and the visual look is preserved in the typification process (Objective 2 and 3). In paper IV, a new LOD is introduced to fill the gap between LOD2 and LOD3. It is the exterior shell of the LOD3 model and the opening objects like windows, doors as well as smaller façade objects are projected onto walls. The results of a user survey show that the shell model can give users almost the same visual impression as the LOD3 model (Objective 2 and 3). In paper V, three methods are proposed to calculate the visual similarity between the original models and the generalised ones. Based on the visual similarity, the buildings in street view are selected for visualisation. The three methods are evaluated by user survey and the results shows that the local based method is better than the global method and minimum projection area in the street level view (Objective 2 and 3). In paper VI, textured 3D city models are generalised by two steps: image compression and texture facade multiple representation based on image segmentation. Wavelet is used for image compression to generate the texture for the required compression level in horizontal/vertical direction. For further generalization, the textured facade is segmented into polygons with similar colours, and then these coloured polygons are merged step by step to create a tree based multiple representation structure for the dynamic visualization (Objective 2).

The author of this thesis has made the implementation of Paper I, II, III, V, VI and shell model generation in paper IV. Co-authors have given suggestions of methodologies and have assisted in writing except Paper IV.
2. Literature review

Along with the increasing demand for 3D city models, visualisation, as one of its most important research aspect, has drawn broad attention not only from the academia but also industry and public authorities. Representation and generalisation of 3D models are the keys to efficient visualisation of multi-scale 3D city models. This chapter aims to provide a thorough literature on the state of the arts of 3D city model representation, visualisation and generalisation.

2.1 Representation of 3D city models

In order to store, manage and integrate 3D city models from different sources, a standard is required to represent 3D city models. CityGML proposed by OGC in August 2008 is the first international standard to represent 3D city models.

2.1.1 CityGML

CityGML overview

CityGML is a common information model for the representation of 3D urban objects. It is realised as an open data model and XML-based format for the storage and exchange of virtual 3D city models. As an OGC standard, CityGML plays a leading role in the modularisation of urban geospatial information.

CityGML is an application scheme for the Geography Markup Language version 3.1.1 (GML3) and represents both graphical appearance of 3D city models and their semantic properties. The current version of CityGML is 1.0 (OGC 2008). Some applications have been developed based on the standard. However, CityGML is still evolving and new attributes/contents will be added into it.

Applications of CityGML

Several cities chose CityGML for their official 3D city models such as Stuttgart, Bonn and Berlin. They use CityGML to integrate existing 3D models from different departments to support urban planning, city marketing, and recruiting of new enterprises (Döllner et al. 2006).

Meanwhile, CityGML has been used in environmental protection. Sustainable Spatial Data Infrastructure for EU Noise Mapping in North Rhine-Westphalia project is one of those (Czerwinski et al. 2006a; 2006b and 2007). The project makes use of CityGML to analyse the noise in the region. Also, CityGML is used in Homeland security and other geo-related analysis project such as Traffic
Simulation and Driving Simulation with Municipal CityGML Geodata from Dusseldorf (Pantzer 2008).

*Extensions of CityGML*

City itself is complex, and it is impossible for a standard to specify every detail of the city. Therefore, objects which are not yet explicitly modelled in the current version of CityGML, can be represented using the concept of generic objects and attributes. In addition, extensions to the CityGML data model applying to specific application fields can be implemented using the Application Domain Extensions (ADE). Generic objects/attributes allow extensions during runtime, but may cause arbitrarily and name conflicts because different user-defined objects may have the same name. The ADE can extend existing CityGML feature types, define new feature types and create its own XML schema definition. Therefore, in many cases, ADE is chosen to extend CityGML. Some examples are:

- CityGML noise ADE used in noise mapping project in North Rhine-Westphalia (Czerwinski *et al.* 2007).
- CityGML flood ADE for a web based 3D flood information (Schulte and Coors 2008).

*Problems of CityGML*

Since CityGML is a new standard, it also has some problems that should be taken care of in the future research (Kolbe 2007).

- Size. CityGML files become very large (Several GB for a large city). Even if the file sizes can be effectively reduced by gzip compression (reduced to around 10% of original size), XML validation and processing can be problematic (classical DOM parsing is not feasible due to main memory limitations) and Web Feature Service (OGC 2009) access might have to be realized in an asynchronous way in order to avoid timeouts.

- Complexity. Because the city itself is complicated CityGML is complex. Even though the CityGML specification is growing, it is still impossible to implement all details in a City. The complexity of CityGML may pose an obstacle to the understanding and application of the standard.
CityGML module and LODs

Since CityGML covers broad thematic fields of city objects, from geometrical and topological to semantic aspects, some applications may only support a subset of all thematic fields of CityGML. The logical subsets of CityGML can reduce the complexity of the models. In CityGML standard version 1.0 possible subsets are defined and embraced as CityGML modules. CityGML consists of a core module and thematic extension modules.

The CityGML core module defines the basic concepts and components of the CityGML data model (OGC 2008). The core module is the lower bound of the overall CityGML data model and a dependency of all thematic extension modules; it must be implemented by any conformant system. In the CityGML core module, the basic classes such as CityObject, Address, Geometry, Feature etc. are defined. Based on the core module, eleven thematic extension modules are implemented in version 1.0 of the CityGML standard. These modules are: Appearance, Building, CityFurniture, CityObjectGroup, GenericCityObject, LandUse, Relief, Transportation, Vegetation, WaterBody, TexturedSurface.

The extension modules can be arbitrarily combined according to the application requirements. A combination of modules is called a profile. The union of all defined modules is the CityGML base profile which forms the upper bound of the overall CityGML data model. The CityGML modules and the base profile will be extended in the future by experts in the different fields. The application data can also be incorporated within the existing modules by using the concepts of generic city objects/attributes or Application Domain Extension mechanism.

CityGML supports five LODs to reflect independent data collection processes with differing application requirements and facilitate efficient visualisation and data analysis (Figure 2.1). LOD0 is a 2.5D Digital Terrain Model. LOD1 is the blocks model comprising prismatic buildings with flat roofs. LOD2 contains differentiated roof structures and surfaces. LOD3 is the architectural models with detailed wall and roof structures. LOD4 combines a LOD3 model with interior structures. City objects can be represented in all LODs simultaneously. In this thesis, the LOD defined by CityGML is adopted.
Figure 2.1: The five LODs defined by CityGML (Source: OGC 2008, pp. 9).

CityGML resources

As an open standard, CityGML is receiving wide support from academic and industry fields. Many open source and commercial software are developed for CityGML visualisation, manipulation and management.

For visualisation, the following tools are developed.

- Aristoteles (2011), an open source CityGML viewer, is developed by the Institute for Cartography and Geoinformation, University of Bonn.
- LandXplorer Xpress Viewer (Autodesk 2011) is developed by the company Autodesk to visualise CityGML dataset.
- FZKViewer (2011), developed by the Institute for Applied Computer Science, Karlsruhe Institute of Technology, can support CityGML 0.4.0, CityGML 1.0.0, the CityGML Noise ADE, the CityGML Subsurface Structure ADE and Industry Foundation Classes (IFC) models.
- Other software such as CityVu (2011), Ptolemy3D (2011), BS Contact Geo 7.2 (2011), FME 2011 Special (2011) also support CityGML visualisation.
To manipulate CityGML the following tools can be used:

- Google Sketchup plug-in to import and export CityGML for editing is developed by researchers from University of Applied Sciences Gelsenkirchen.
- Citygml4j (2011), an open source Java class library and API for facilitating work with the CityGML, is developed by Institute for Geodesy and Geoinformation Science of the Berlin University of Technology. It can be used to read, process, and write CityGML datasets, and to develop CityGML-aware software applications.
- QS-City 3D is a free online service for checking CityGML data developed by HfT Stuttgart (2011).

For management, 3DCityDB (2011), a free and Open Source 3D geo database to store, represent, and manage virtual 3D city models on top of the Oracle 10G R2 spatial (or 11G), is developed by Institute for Geodesy and Geoinformation Science of the Berlin University of Technology. The database model contains semantically rich, hierarchically structured, multi-scale urban objects facilitating complex GIS modelling and analysis tasks.

This thesis makes use of citygml4j to implement the reading, parsing and exporting of CityGML files.

2.1.2 Other standards

Besides CityGML, there are some other standards related to 3D city models such as VDI 3805 (2011) and Industrial Foundation Class (IFC 2011).

VDI 3805 is an industry standard for product data exchange in Building Services and is used as uniform format to model all Building Services equipment (heating technology, ventilation and air conditioning technology, sanitary engineering, etc).

IFC is a neutral and open specification to facilitate interoperability in building industry and is a commonly used format for BIM (Eastman 2009). IFC is now an official International Standard ISO/IS 16739 and has been made a compulsory format for publicly aided building projects by some countries e.g. Denmark.

The relationship among VDI 3805, IFC and CityGML is shown in Figure 2.2. VDI 3805 is a standard about components in the building; IFC is focused on building modelling and CityGML is designed on the city level and contains more city objects such as vegetation, road etc.
Figure 2.2: Relationship of VDI 3805, IFC and CityGML (Source: KIT 2011).

These standards can be converted and integrated into each other (El-mekawy 2010). With the software e.g. ETU VDI 3805 Navigator (2011), models in VDI 3805 format can be saved as IFC format. IFC models can also be converted into CityGML models. The Open Source BIM Server project (BIMServer 2011) now supports the export of CityGML. Models can be converted on the fly from the stored IFC models and exported as CityGML LOD4 building models without losing their semantic information (thematic structure and attributes). For example, a complete IFC model can be exported into CityGML or contrary by using the IfcExplorer (2011). Berlo and Laat (2011) also described a CityGML extension called GeoBIM to get semantic IFC data into a GIS context and convert IFC to CityGML.

Since VDI 3805 and IFC models are designed for more detailed level, CityGML is the most suitable standard to represent a whole 3D city. Meanwhile, the data in these related formats can be converted into CityGML data and becomes an important source of 3D city objects.
2.2 Visualisation of 3D city models

This section concentrates on presentation of 3D city models—3D visualisation. CityGML is designed to represent 3D city models, but not to present or visualise 3D city models directly. As Kolbe (2008, pp. 28) points out: “CityGML is complementary to visualisation standards like X3D or KML. While these address presentation, behaviour, and interaction of 3D models, CityGML is focused on the exchange of underlying urban information behind 3D objects”. Therefore, specific 3D visualisation technologies are required for more efficient 3D city models presentation.

2.2.1 Standards for 3D visualisation

In the early stage of computer graphics, visualisation was implemented directly based on the hardware (Carlson 2003). For example, if a user wants to draw a line on the screen, he/she should not only supply the coordinations of start and end point but also consider the monitor resolution, frame buffer, and related system calls. Since the computer platforms were quite different from each other, the portability of the visualisation program at this stage was quite low.

In order to simplify the development process and increase the portability, standard APIs for writing applications that produce 2D and 3D computer graphics were proposed (DirectX 2011, OpenGL 2011). These APIs deal with the hardware, supply the function calls that can be used to draw complex 3D scenes from simple primitives. The manufactures can also upgrade their hardware without influencing the applications running on them by supporting the APIs. However, there are still problems. First, the APIs are still too complicated (OpenGL API contains 250 different function calls) and too primitive (only support atomic geometric objects e.g. point and line segment) to implement 3D visualisation directly. In order to create 3D scenes, the user is still required to calculate the projections, transformations, render the scene and so on, which is not suitable for the complex 3D models like 3D city models. Second, there are several graphic APIs such as OpenGL, DirectX, Mesa 3D (2011), VirtualGL (2011), RISpec (20110), Glide (2011) and so on. Different versions have to be created to support different APIs. For 3D city models which needed be viewed by anyone from anywhere, it is not a wise choice to employ these basic graphic APIs for online 3D city models visualisation.

Along with the development of the Internet, the requirement for sharing information is increasing. In order to represent the 3D computer graphics and to view them through Internet, 3D standards are proposed. These standards describe the 3D scenes, and supply APIs for user interactivity. The visualisation of 3D city models can be saved in these standards that are supported by various

X3D

X3D is an open standard and run-time architecture to represent and communicate 3D scenes and objects based on XML. As the successor of VRML, it encodes the 3D scene using XML syntax. It also provides an enhanced APIs. X3D is proposed and managed by Web3D Consortium, and is now an ISO ratified standard that provides a system for the storage, retrieval and playback of real time graphics content embedded in applications, all within an open architecture to support a wide array of domains and user scenarios. X3D is particular designed for visualisation 3D scenes through Internet.

X3D is now being used in many cases, such as e-learning (Thomas 2008), medication (Jung et al. 2008, Willis 2007, Hamza-Lup et al. 2006) and geo-related applications. Based on X3D, Araujo et al. (2008) proposed a framework for 3D web-based visualisation of HLA-compliant (High Level Architecture) Simulations. X3D has been applied to virtual cultural heritage (Eliens et al. 2007, Cabral et al. 2007) and BIM (Campbell 2007). In mobile GIS, 3D maps and models can also be implemented in X3D (Nurminen 2006, 2007, Alessandro et al. 2007).

Meanwhile, X3D itself is improving. Jung and Behr (2008) proposed a method to extend the animation; Buttussi et al. (2007) and Patterson (2007) focused on the search of X3D models. Semantic information is also adding into X3D (Bilasco 2007, Pittarello and Faveri 2006); Jung (2007) used X3D into Mixed Reality appliances; Weber and Parisi (2007) designed an open protocol for Wide-area Multi-user X3D.

As an open standard, there are several tools to support X3D. Many free viewer software and plugins for browsers are available for X3D visualisation, for example Bit Management Contact viewer (2011), Octaga (2011), Flux (2007) and InstantReality (2011). X3DOM (2011) is an open source framework to integrate X3D content with HTML5 and to visualise the 3D scenes through web browsers.

Editors are also developed to create X3D file. X3D-Editor (2011) is a graphics file editor for X3D that enables simple error-free editing, authoring and validation of X3D or VRML scene-graph files. It is the editor officially recommended by Web3D. Other editors such as Vivaty Studio (2011), BS Editor (2011), SwirlX3D Editor (2011), and Flux Studio (2011) are commercial
products. Some of the 3D softwares like Blender, 3DSMax, AutoCAD and Maya are also supporting any X3D file import/export.

For development, Xj3D (2011) is an open source toolkit for X3D related applications to create interactive 3D scenes. Xj3D supports two types of visualisation platform Java Application and Java Applet which could be used to deliver 3D city models through Internet. ChefX3D (2011) focuses on developing a toolkit for building 3D editors. It is expected that these editors will form part of a larger application. Typically these editors will be customized for authoring certain types of content. Rez (Thorne 2007) is an open source framework and tools for translating geospatial data to different formats including images and multi-resolution models for X3D or VRML web browsing. Jeospace (2011) is another open source java toolkit for creating geo-spatial simulations.

X3D is playing an important role in geo-visualisation applications. In Web3D Consortium, researchers are developing an X3D profile for GeoVisualisation, called X3D Earth (2011). X3D Earth is an open standards-based infrastructure for visualising information in a geospatial context. X3D Earth profile has been used in many projects and applications e.g. terrain Generator in Rez, GeoSpace globe generator, WorldWind (2008) with an X3D loader, Digital Nautical Chart (2011), Planet 9 Virtual Cities (2011), MBARI Monterey Bay operations (MBARI 2010), and NPS Savage Studio scenario creation (X3D Earth 2011). Meanwhile, the Web3D Consortium and OGC have made an agreement to integrate X3D with OGC standards such as CityGML (Havele 2010).

KML/COLLADA

KML is a file format to display geographic data in an Earth browser such as Google Earth and Google Maps. OGC has approved KML as an official standard in April 2008 and defined KML as an XML language focused on geographic visualisation, including annotation of maps and images. Although KML is not designed for 3D visualisation, it employs COLLADA for 3D model. COLLADA stands for COLLaorative Design Activity and it is an interchange file format for interactive 3D applications. COLLADA is now an open standard XML scheme for exchanging digital assets among various graphics software applications. It is mainly used in industry, especially in the game industry.

KML/COLLADA is designed for Earth browser, while X3D is a better choice to present online 3D city models for its compatibility with the HTML web and wide support from popular browsers such as Firefox or Chrome. Meanwhile, Google Earth and Google Map are increasingly popular; many people not only view 3D city models from Google Earth but also create and share 3D city models through it. The city of Berlin publishes its 3D city models through
Google Earth in KML format (Kada 2009). Many online GIS applications employ KML for visualisation. Calado et al. (2008) export hydrographic data into KML. Chen et al. (2008) visualise the earthquake impact with Google Earth. Shinozaki (2008) uses KML for virtual 3D urban design in Tokyo and Fukuoka. KML could be another important option for 3D city models visualisation and generation source of 3D city models. It is necessary to integrate KML with CityGML and X3D for better 3D city models visualisation in the future.

**Other standards**

Besides X3D and KML/COLLADA, there are several other 3D visualisation standards, e.g. 3DMLW, O3D and U3D. 3DMLW, stands for 3D Markup Language for Web, is an XML-based file format for representing 3D and 2D interactive content on the World Wide Web. Displaying 3DMLW requires the 3DMLW plug-in to be installed on the computer and it uses OpenGL for rendering. The 3DMLW plug-in is developed by 3D Technologies R&D for common web browsers (Internet Explorer, Mozilla Firefox, Opera, etc.). O3D is an open source JavaScript API created by Google for creating interactive 3D graphics applications that run in a web browser window. Universal 3D (U3D) is a compressed file format standard for 3D computer graphics data. 3D objects in U3D format can be inserted into PDF documents and interactively visualised by Acrobat Reader since version 7.

Compared with X3D and KML/COLLADA, these standards are designed for certain applications by different companies, but they are not international standards approved by OGC or ISO and they are not widely supported. However, these standards can be used for certain purposes such as visualising 3D city models in U3D to create 3D PDF files.
2.2.2 Visualisation frameworks

Many frameworks have been proposed for visualisation of 3D city models. In this section, some of the typical ones are introduced.

Direct visualisation

Beck (2003) directly employed OpenGL to visualise 3D city models in real-time. In order to achieve the high speed, this framework employed the computer graphics related technologies such as pre-stripping the triangles, frame controls, texture memory management etc. The data comes from a database in self-defined 3D format and integrated with the textures. LODs are used to display several geometric representations of the same object at different times. Fewer details are represented when the object is far away and more detailed when it is closer to the observer. They developed several web plugins for Internet browsers to facilitate the interactive visualisation of high-resolution 3D terrain and building data over the Internet. These plugins deal with TCP/IP protocols and are optimized for low-bandwidth data transfer. For security purposes all data streams are compressed and heavily encrypted.

Figure 2.3: Rendering phases and graphics resources (Source Döllner et al. 2005, pp. 46)
Döllner et al. (2005) presented a direct illustrative visualisation technique to provide expressive representations of large-scale 3D city models. Their framework was inspired by the tradition of artistic and cartographic visualisations typically found in bird’s-eye view and panoramic maps. The rendering algorithm consists of four phases (Figure 2.3): Phase 1 generates a texture encoding shadowed regions in image space; Phase 2 renders the scene with enhanced image-space edges, shaded and textured facades; Phase 3 renders stylized edges; and Phase 4 renders remaining components of the 3D city model.

Although it may increase the visualisation speed, the direct visualisation framework has to deal with basic computer graphics issues such as frame control, scene rendering etc., which may increase development difficulties and workload dramatically. Different versions of the visualisation programs need to be developed for different platforms. Too many details related with the basic operation system and graphics interfaces may highly influence the portability of the visualisation framework. Therefore, visualisation of 3D city models based on standards is proposed.

**Visualisation using 3D standards**

Several studies have applied 3D visualisation standards such as VRML, X3D and KML to their 3D city models. Lerma and Garcia (2004), for example, employed VRML to visualise 3D models of the historical centres. They developed the software TopVRML to model and join 3D virtual objects. TopVRML is written in C++ to visualise VRML files directly and interactively over the Internet or local networks. This software allows users to set the observer characteristics (flying, walking, or examine), sky and ground backgrounds; insert a 3D prototype and link it with a web page; generate 3D objects from images and create geometrical shapes such as cube, cylinder, cone etc.; and set the illumination by placing both directional and punctual colour lights in the scene.

Since Google Earth is widely applied and its visualisation language KML becoming an open standard of OGC, Google Earth has become a popular data viewer in a variety of fields including geoscience. Yamagishi et al. (2010) proposed a 3D visualisation system for geoscience data based on KML. They developed a software package, called “KML generator” to convert original data files to KML. These generators can deal with seismic tomographic models, geochemical data of rocks, and geomagnetic field models. With these KML generators, an overlapping visual presentation of different types of data can be obtained. Existing KML files of geoscientific data (e.g., earth quake hypocenters and seismograph stations) provided by various research institutions around the world (such as the U.S. Geological Survey) can also be overlaid on these data.
Eran et al. (2005) gave an example of creating and visualising 3D city models using X3D. Their main source of data is a topographic map (see Figure 2.4) and aerial photographs. Topographic maps provide geographic reference which store the positional information of terrain features such as buildings, roads, rivers, contours, land use, administrative boundaries etc. Aerial photographs provide information about the city model textures. Several different types of software are employed to create the 3D models. ArcGIS was used to develop the foundation of their 3D city models, while a 3D-modelling application, Canoma (2011), was used to construct a photorealistic building model. 3D Studio Max was used to authoring and integration of the foundation, building and other urban features (e.g. tree, lamp post and signboard). Cosmo player (2011) was used to visualise the 3D city models in X3D format.

**Figure 2.4:** Test topographic map of Shah Alam city for 3D building generation (Source: Eran et al. 2005, pp. 4).

The workflow of model creation in Eran et al. (2005) is shown in Figure 2.5 and is described as follows:

1. DEM was generated from topographic map and aerial photograph using ArcGIS software.
2. Superimpose or drag a 2D representation of topographic features, e.g. roads, rivers & land use information from an orthophoto onto the DEM.
3. Create the initial 3D buildings based on the footprints and the heights from the aerial photograph. Then merge these models with the generated DEM.
4. Reconstruct photo-textured building models with multiple view points from terrestrial images.
5. Edit the texture of the building model to eliminate the shadow effects and obstructing objects like trees and vehicles.
6. Integrate the foundation of the initial city model with the photorealistic building models and add extra accessories (landscape objects) such as trees, lamp post, sign board, bus stop etc.

![Workflow scheme of X3D model generation](image)

**Figure 2.5**: Workflow scheme of X3D model generation (Source: Eran et al. 2005, pp. 6).

The created models were saved in both X3D and VRML format and visualised with X3D viewers such as Cosmo player or Flux player.

However, these 3D standards can not be used to represent 3D city models, because the semantic information is missing. It will be impossible to integrate data from different standard formats and perform spatial analysis, if there is no semantic information. Therefore, 3D city model visualisation requires a standard that supports both geometry and semantic information. This standard is CityGML.

**Visualisation using CityGML**

CityGML, the OGC standard to represent 3D city models, can be used to integrate both geometric and semantic information of the city models. Döllner et al. (2006) introduced the concepts, implementation, and experience of the Berlin virtual 3D city models. The project is the origin of the CityGML standard. Figure 2.6 shows the overall architecture of the Berlin 3D City Model System:
3D Authoring System: It is responsible for creating, editing, and versioning of the virtual 3D city models and its components, e.g., importing, exporting, grouping, and annotating buildings, vegetation plans, landscape plans, etc. Technically, it provides an interactive access to the 3D geo-database.

3D Geo-Database System: The database for storing and managing virtual 3D city models is based on the logical structure of CityGML. From Figure 2.6, it is clear that the CityGML based 3D Geo-Database System is the core of the whole system. Geo-datasets from different resources such as Cadastral Data, Digital Terrain Model, Aerial Photography, Building Models etc. are all converted into CityGML data and are integrated in a uniform database. With the CityGML, exchange of 3D city models becomes much easier for different applications. And for visualisation, it is not necessary to consider the variety of data sources but only focus to present the CityGML.

3D Presentation System: The presentation system provides real-time visualisation of and interaction with the virtual 3D city model. The 3D city models are converted into both KML and X3D format for different viewers. It is possible to access the Virtual Berlin through a KML viewer like Google Earth and a X3D viewer like LandXplore.

Figure 2.6: The system architecture of the virtual 3D city model of Berlin (Source: Döllner et al. 2006, pp. 4).
Reitz et al. (2009) talked about the integration of CityGML and X3D in spatial data visualisation of web 3D service (W3DS). Figure 2.7 outlines the architecture of their W3DS. The geodata in CityGML format is first processed and stored in the *Data Description*, a structure containing meta-information about the object in the dataset (e.g. an object’s spatial location). Then the data is filtered according to certain rules and becomes the *Filtered Data*. Finally, the Portrayal Service creates the view, which consists of the filtered objects in their representation defined by Styled Layer Descriptors. Finally the W3DS renders this view into the exchange format requested by the client (for example VRML or X3D).

**Figure 2.7**: The architecture of a W3DS (Source: Reitz et al. 2009, pp. 140).

In Figure 2.7, geospatial data in CityGML is filtered in two steps. The view will then be created according to Styled Layer Descriptors and encoded into a scene graph format like X3D before it is rendered to the client. (Reitz et al. 2009)

Compared with other visualisation frameworks, it is reasonable to represent 3D city data in CityGML and visualise it with X3D or other 3D visualisation standards, in which both the geometry and semantic information are supported. Considering the connection between CityGML and X3D (Park 2010), X3D is selected to visualise the 3D city models in this thesis.

**Online 3D City Model Visualisation**

The Internet is a convenient and widely accessible channel to deliver information including 3D city models. Therefore, many frameworks for online 3D city models visualisation are proposed in both academic and industry fields. These frameworks can be divided into three categories: thick client-server model, thin client-server model and Peer to Peer (P2P) model.
**Thick client-Server model**

In the thick client-server model, the user has to install a specific client program to access the 3D scenes supplied by the city model server. The developer defines and controls the whole visualisation and communication process, which gives more flexibility to the developer but also increases the technical complexity. One of the most noticeable defects of the Client-Server model is the difficulty to persuade the user to install the client program, which limits its accessibility. Also, several versions of the client program have to be developed for different platforms, so the development workload is increased.

One of the most successful thick client-server programs for the 3D city models is Google Earth. It is installed in most computers and mobile devices such as iphone and ipad. According to statistics from Apple (Siegler 2011), Google Earth is the 7th and 4th top all-time download app in iphone and ipad respectively. Because of the wide availability of Google Earth KML has become the OGC standard for expressing geographic annotation and visualisation. It is popular to deliver 3D city models through the Google Earth platform. It has been used as web-based solution for architectural 3D models in Apollonio et al. (2010). However, Google Earth is still commercial software and the development based on it is constrained. Another similar commercial 3D city platform is the Bing Map 3D from Microsoft.

NASA World Wind is an open source thick client-server based virtual globe program for personal computers. It starts to support 3D content and can be applied in the visualisation of 3D city models. Conti et al. (2009) used NASA World Wind to deliver web-based 3D content to provide interoperable access to geographical information and geospatial processing services.

Agugiaro et al. (2011) developed a 3D visualisation client using the game engine Unity 3D. Unity is an integrated authoring tool for creation of 3D videogames or other interactive content such as architectural visualisations or real-time 3D animations. The PHP Internet communication library of Unity is used to access the 3D database built with PostgreSQL. The system framework is shown in Figure 2.8.

![Figure 2.8: 3D city data exchange framework (Source: Agugiaro et al. 2011).](image-url)
Thin client-Server Framework

In a thin client-server framework, the browser is usually used for retrieving, presenting, and traversing information resources on the web. It is installed in almost every computer that has access to the Internet. In fact, the browser is an essential part of a modern computer operation system and is often installed by default. Compared to the specific user clients, including Google Earth, the browser is much more widely available. A user can interact with the 3D city scenes much easier if the models are delivered through the browser. Since existing browsers do not support 3D content natively, the following methods are proposed for 3D visualisation in the browsers.

Plugin based. A plugin is a set of software components that adds specific abilities to a larger software application. To support 3D scenes, many plugins have been developed for the browsers to visualise 3D content. Several online 3D city model visualisation frameworks are based on plugins.

Some online 3D city model visualisation frameworks are based on VRML/X3D. Zhou et al. (2006) created a backend server to process the parameters from the user, generated 3D models, and sent them back to the VRML-enabled browser for display/navigation. Their display and navigation of the 3D model is implemented with Java Server Pages/Servlet technologies built under a multitier web architecture. Lerma et al. (2004) also selected VRML for their virtual museum project.

Manferdini and Remondino (2010) developed a 3D city model visualisation methodology based on O3D technologies. O3D is a JavaScript implementation to create interactive 3D applications with web browser supporting WebGL (a 3D graphics API based on OpenGL). In their visualisation pipeline, the detailed 3D building models are stored in COLLADA format which is then converted into O3D gzipped tar file (.o3dtgz) with a special file describing the scene and textures. The model is finally loaded inside the O3D viewer plugin for visualisation, interaction and database queries.

Marvie et al. (2011) proposed an approach for adaptive streaming of online multiuser virtual worlds, using generic transfer protocols and a unified representation of the worlds based on X3D. The system is based on passive servers and does not require any dedicated protocol, server software or specific proxy configuration. The geometry and textures are sent to the user browser on-demand depending on live estimates of the quality of network links and of available resources. The 3D stream can thus be visualised in any X3D supported browser.
Web services. Web services are also applied in online 3D city model visualisation. Haist and Coors (2005) developed CityServer3D, a W3DS Interface to deliver the 3D city models as well as the GIS data. The server component consists of different modules which can be categorised into the interface (IL), converter (CL), functional (FL) and database (DB) layer. The overall architecture of the CityServer3D is presented in Figure 2.9. A web browser based viewer, the WebViewer, is developed to support visualisation and integration of 3D GIS data. In their web service, the 3D city models are generated and represented in VRML and visualised with the browser plugin. Also, they supplied the CityGML and M3G (for mobile devices) output formats.

![Figure 2.9: Overall architecture of the CityServer3D system (Source: Haist and Coors 2005, pp. 2).](image)

Besides W3DS, Web Perspective View Service (WPVS) is another OGC standard under discussion for 3D online visualisation. Compared with the W3DS, WPVS only transfer the image to the browser. These images about the 3D scenes are generated in the server side, so that the plugins in the browser can be implemented easily. Willmes et al. (2010) made use of the deegree3 framework (http://www.deegree.org/) to implement the WPVS. In their framework, the 3D city models from different sources are represented in CityGML that is imported into deegree3 backend. They also developed a
WebGUI frontend (browser plugin) for navigation in a 3D scene using the WPVS interface (http://www.campusgis.de/3D).

However, as pointed out by Behr et al. (2009, pp. 128): “All plugin-based systems have two major drawbacks. First, they are plugins and not installed by default on most systems. Therefore, the user has to deal with plugin installation, security and browser or OS incompatibility issues. Second, the presented systems define an application and event model inside of the plugin, which is decoupled from the DOM content. Developers, who try to develop integrated web-applications or web-pages that use both, the DOM/browser and the plugin-model, have to deal with the small plugin-specific interface and its synchronisation capabilities.”

**X3DOM.** In the HTML5 specification, the next generation of HTML, the 3D scene will be supported by the browsers directly. Some mainstream browsers such as Firefox, Chrome and Safari are already embedded with the native 3D library, WebGL. Meanwhile, X3D is selected as the standard 3D model representation language in HTML5. Based on WebGL, X3DOM is developed to show native X3D within an HTML page. Some of the 3D city model projects are built on X3DOM.

Lornet (2011) demonstrated a showcase in the Web3D 2011 conference about the 3D city models of Dijon. The 3D textured models are exported as X3D and visualised with the browser using X3DOM. The demo is around 350 Kb XHTML page and the textures weigh 2 Mb. Even though, loading and processing are still time consuming due to limitations of the browser. Using X3DOM to visualise the 3D city models will be widely used as a default part of the HTML5, but the performance still has to be improved to deal with the limitations of the browsers.

Jung et al. (2011) described application scenarios and analysed the technological requirements for an efficient presentation and manipulation of 3D virtual heritage assets on the web. Their implementation indicates that X3DOM integrates the HTML and X3D, provides a single declarative developer interface based on current web standards, and supports various back ends through a powerful fallback model for runtime and rendering modules. The benefit of the proposed model is the tight integration of declarative 3D content directly into the HTML DOM tree without the need to forge new concepts, but by using existing standards. Similar to images or videos today, 3D objects will become just another popular medium in the Internet.
P2P 3D visualisation Framework

The previous client-server models cannot scale easily with user size, because the bandwidth or computing resources of any given server is often fixed, whereas the number of users might increase with user activities, and may in some cases overload the server capacity (Chien et al. 2009). Therefore, P2P architectures (Royan et al. 2007, Hu et al. 2008, Botev et al. 2008) are being developed to support online 3D visualisation.

Royan et al. (2007) used two different types of hierarchical LOD models for terrain and groups of buildings and implemented the online 3D visualisation using peer-to-peer technologies. They focused on the two important problems that arise when implementing a P2P overlay: Peer connectivity and Data exchange policy. Their simulation results demonstrate the self-scalability of the P2P overlay and its capacity to quickly self organise.

Hu et al. (2008) also employed P2P networks for 3D streaming and presented a structure that allows clients of 3D virtual globe or virtual environment (VE) applications to obtain relevant data from other clients while minimizing server resource usage. However, their framework can not function properly if there are not enough neighbour peers for a node. Meanwhile, the bandwidth of some users cannot be properly used when the data sources are limited to only a few nearby neighbours in the virtual space.

Chien et al. (2009) proposed Bandwidth-Aware Peer Selection (BAPS), a peer selection strategy that improves the bandwidth utilization for 3D streaming. Their framework avoids request contention and peer overloading as object and user densities increase, thus improving both bandwidth utilization and system scalability. However, this framework has to maintain a Peer List, which might be a bottleneck for the server. Also the complexity of the system is increased compared to the user-client systems.

On one hand, P2P frameworks are complex. On the other hand, recent development in cloud computing (Yang et al. 2011) increases the ability of the server dramatically, so the thin client-server framework instead of P2P architectures are employed in this thesis.
2.3 Generalisation of 3D city models

Visualisation of 3D city models in multi-scale is essential for online presentation. Zooming user interface is used to change the scale of viewed area in order to see more detail or less, and to browse through different areas (Bederson and Hollan 1994). It is important to find methods to generate lower LODs from higher LODs automatically by generalising city objects on block level, building level and facade level, because generalisation not only reduces the data volume but also hides the unnecessary details in different scales. These generalised representations in different LODs should be unified to support dynamic visualisation. In this section, some generalisation algorithms for geometry and texture are first described. Then algorithms for measurement of the difference between the original models and the generalised ones are introduced. Finally, current multi-LOD fusion methods are presented to create continuous LOD.

2.3.1 Geometry generalisation

Automatic cartographic generalisation has been acknowledged as an important research topic for a long time (Mackaness et al. 2007). An overview of generalisation methods was given by Harrie (2001). Generalisation contains several operators such as simplification, smoothing, aggregation, amalgamation, collapse, selection, typification, exaggeration, enhancement, classification, displacement (Shea and McMaster 1989). Recently, much attention is paid to 3D city models related generalisation. In this thesis, simplification and typification are mainly discussed since they provide the key aspects for generalising buildings in 3D city models.

The problem of simplifying 3D city models in higher LODs are proposed a decade ago (e.g. Köninger and Bartel 1998). Many researches have been performed on simplifying single buildings (Thiemann 2002, Forberg 2007, Kada 2002, Sester 2000, Fan et al. 2009). The CityGML standard also specifies five LODs (Figure 2.1) for efficient multi-scale visualisation. Several simplification methods have been proposed to obtain lower LODs from higher LODs automatically. Meng and Forberg (2007) summarised generalisation methods that change the representation between different LODs. Sester (2007) also overviewed 3D visualisation and generalisation. For single building model simplification, Mayer (2005) and Forberg (2007) created a scale-space technique partly based on the morphological operators opening and closing to simplify 3D building model. A half space model was used by Kada (2006) to detect the main outline of a building as shown in Figure 2.10. Then he extended the simplification method to the roof structures using pre-defined roof types (Kada 2007). As CityGML becomes the OGC standard, some researchers focus on simplifying 3D city models stored in CityGML format. For a group of buildings,
aggregation is used to simplify the models. Aggregation means to combine individual buildings of a city model so that intermediate nodes or faces can be removed. The result of the aggregation is a reduced number of building objects and building surfaces. Anders (2005) introduced an aggregation method using 2D projections of linear building groups. Glander and Döllner (2007) made use of cell-based clustering to merge building blocks. Götzelmann et al. (2009) discussed the terrain-dependent aggregation of 3D city models. Guercke et al. (2011) introduced the methods to model different aspects of building aggregation in the form of Mixed Integer Programming problems.

Typification is another effective method for 3D city model generalisation. It is the process of replacing a number of objects by a smaller number of new objects. It has been studied in 2D by Regnauld (1996), Moritz et al. (2009) and Anders and Sester (2000). Typification methods can be categorised into two types (Anders 2005): with and without structural knowledge.

Typification methods with structure knowledge try to detect the geometrical structures in the object groups and generalise the groups based on these structures. Regnauld (1996) employed MST to detect the linear structure of building groups and implemented the typification generalisation based on the MST. Christophe and Ruas (2002) proposed another straight line based approach to detect building linear structure and to group the objects directly. Besides line structure, Burghard and Cecconi (2007) made use of Delaunay triangulation to detect the neighbourhood structures. Two dimensional structures or grid structures are also used for typification (Anders and Sester 2000, Anders 2005). Most of these structure knowledge typification methods use Töpfers’s radical law (Töpfer and Pillewizer 1966) to determine the reduction factor. Töpfers’s radical law provides numeric guidelines by which to determine how much detail to retain during map compilation and reduction.

**Figure 2.10**: Generalised 3D building model in its original (left) and simplified (right) shape (Source: Kada 2006, pp. 6).
Typification methods without structural knowledge try to implement some common rules in the generalisation of the object groups. Müller and Wang (1992) used mathematical morphology to typify natural areal objects. They removed the smaller objects unless they are important. Sester and Brenner (2000) employed neural network, Kohonen Feature Maps, to preserve the original structure. The remaining objects are moved along the direction of the removed ones to minimize a certain error measure. However, these methods may destroy the dominant structures of the models (Anders 2005).

Some of the above mentioned geometry generalisation algorithms related with this thesis are described in detail as follows:

**Aggregation**

A number of aggregation methods have been developed for aggregation of 2D cartographic objects and 3D buildings. Bundy et al. (1995) introduced two types of aggregation operators: Direct-merge operator and Snap-merge operator. The direct-merge operator maintains the alignment of the objects by moving the objects together directly. The relationship between the facing edges is represented by the triangles that have an edge in one object and a point in the other. Figure 2.11 shows an example of direct-merge. Meanwhile, the snap-merge tries to align the objects’ nearest parallel edges. It can be achieved by aligning the merge vector with the shortest outer connecting edge. Figure 2.12 shows the process of snap merge.

![Figure 2.11: Direct merge (Source: Bundy et al. 1995, pp. 112).](image1)

![Figure 2.12: Snap merge (Source: Bundy et al. 1995, pp. 112).](image2)
Anders introduced an approach for 3D building aggregation using the following steps (Anders 2005).

1) Compute the minimal bounding box of the building group using all points of the building geometries.

2) The length, width, and height of the bounding box define the three projection-directions L, W and H.

3) Create the orthogonal projection along height (Figure 2.13a), width (Figure 3.7b), and length (Figure 2.13c).

4) Generalise the three projections (Figure 2.14b).

5) Extrude the 2D geometries of orthogonal projection along height, width, and length respectively (Figure 2.14c).

6) The generalised 3D building group is computed by the intersection of the extruded geometries (Figure 2.14d).

From the experiments, Anders’ 3D building aggregation method is quite suitable for buildings in a straight line with same orientation. Also their projections along height, width and length should also be in a straight line or superposition. These constraints may restrict the application of this method.

![Figure 2.13: 2D building projections (Source: Anders 2005, pp. 6).](image)
Dynamic typification

Typification is another important generalisation operation for many city areas with regular building distribution. Typification is a generalisation operation that replaces a large number of objects by a smaller number of objects while trying to preserve the typical spatial structure of the objects. Two types of typification methods, typification with and without structural knowledge are introduced as follows.

Typification without structural knowledge

Müller and Wang (1992) made use of mathematical morphology and proposed a solution to automatically generalise one particular class of geographical objects. These objects are area patches (or ground plan polygons) distributed over a two-dimensional space, and are assumed to be no semantically difference among them. Certain rules are applied to typify the objects: (1) emphasis on larger patches at the expense of smaller ones, (2) preservation of the overall group relationship, (3) partial topological integrity, and (4) differential displacement according to patch area. Even if they try to preserve the group relationship, it is a quite hard task without knowledge of the overall distributions.
Sester and Brenner (2000) described another typification algorithm without structure knowledge. Their approach was based on Kohonen Feature nets (Kohonen 1982), a neural network learning technique. The neurons of the nets have a property of keeping their spatial ordering when they are adapted to a new situation, which could be used in typification. The process is described as follows (Figure 2.15). First, determine the reduction rate using Töpfer’s law (Töpfer and Pillewizer 1966). Then, randomly select a set of objects as remain set, and create the Delaunay triangulation to represent the topology of the objects in the remain set. Next, introduce the remain objects as the output objects of the Kohonen net and go to the learning phase. In the learning phase, the original objects act as attractors that drag the objects of the output map into their direction. In the learning iterations, the output objects are iteratively adjusted according to the underlying attractor structure. Finally, the output objects are distributed across the space according to the original spatial distribution.

![Figure 2.15: Typification based on Kohonen Net (Source: Sester 2005, pp. 888).](image)

In certain cases where objects can be assumed as equal points, this approach produces quite good results. However, in the 3D city models, objects like buildings are different from each other. Also the typification algorithms without structure knowledge cannot preserve the dominant structures, like linear or grid, which are quite important in visualisation according to the Gestalt psychology. Therefore, structure based typification algorithms are proposed.
Typification with structural knowledge

To generalise a structured city model, one has to first detect the structure of the models. In the real city, there are many “regular” areas. Regular means that the buildings in these areas have some common property or structures, e.g., linear structure in which buildings are evenly distributed along a road or grid structure, which is lattice-like layout of buildings.

Regnauld used MST to detect the linear structure and recognize the building clusters for generalisation (Regnauld 1996). Each building is viewed as a forced cluster. The MST is composed with buildings as nodes and segments that represent the distance between two buildings as edges that is weighted by the minimal distance between the two building contours. The MST is built by linking each building with its closest neighbour. Then the MST is segmented according two criteria: delete the long edges and make the buildings of a well-known structure to be part of the same linear cluster.

Christophe and Ruas (2002) proposed a linear structure detection method based on straight line. First, identify the aligned buildings within a city block as the candidates for linear structure by scanning the whole space with straight lines. Second, qualify the generated candidates according to their regularities and eliminate the ones, which are not regular. The drawback of this method is the computation load. Because every possible straight line in the space has to be scanned, it takes too long to calculate if the interval is too small and some of the linear structures may be missed if the interval is too big.

Anders (2005) introduced an algorithm to detect the grid structure of 3D building groups. In a grid structure every building belongs to two linear structures which have to be preserved if possible. He uses relative neighbourhood graph (Toussaint 1980) to detect the grid structure. The detected irregular grids are abstracted to perfect regular grids with $m \times n$ members. The orientation of the grid is determined by the minimum enclosing rectangle (Freeman and Shapira 1975). Then the abstracted regular grid structure is typified by reducing the number of rows and columns. For other non-structured parts the typification methods without structure knowledge can be employed.
2.3.2 Texture generalisation

Texture is another essential part of 3D city models. Textured models give more realistic visual impression than the non-textured ones. Besides created by hands, many methods are developed to generate the textured 3D city models automatically. Frueh (2002) proposed the automated texture mapping method for 3D city models with oblique aerial imagery. Companies such as Blom (2011) and C3 (Simonite 2011) used Lidar and airborne oblique imagery to generate textured 3D city models in large areas. Xiao et al (2009) proposed an automatic approach to generate street-side 3D photo-realistic models from images captured along the streets at ground level. Nowadays, more and more textured 3D city models are available.

Usually, the texture information is stored as the image file and linked to the 3D geometry models with coordinate tags. The texture images are raster data, and always take a big proportion of the whole models. Also, the loading, processing and visualisation time required for the texture image is usually much longer than for the geometry data. Therefore, the generalisation of texture is drawing increasing attention. In order to simplify the texture and increase the visualisation efficiency, different methods are proposed to compress the texture images.

Texture image compression is different from the general image compression methods such as JPEG, and the texture compression algorithm are optimized for random access. According to Krasue (2010), the existing texture compression methods can be classified into 3 categories.

1) Vector Quantization (VQ). The initial study of VQ is indexed colour (Heckbert 1982), and then Beers et al. (1996) extended it and proposed the concept of VQ which is suitable for texture with many repeating patterns.

2) Block based compression. Early studies contain Block truncation coding (Delp and Mitchell 1979) for gray images and Colour Cell compression (Campbell et al., 1986) for colour images. Knittel et al. (1996) first implemented the block based method into hardware. Nowadays, the most common method is S3 Texture compression (Iourcha et al. 1999) that is implemented both in OpenGL and Direct3D and applied in different applications. Other block based approaches are Ericsson Texture Compression (Strom and Moller 2005) and 3Ds (ATI 2004).

3) High Dynamic Range (HDR) texture compression. The block based methods are suitable for Low Dynamic Range (LDR). Munkberg et al. (2006) and
Roimela et al. (2006, 2008) proposed a method to deal with HDR. Meanwhile, HDR also is implemented in DirectX 11 (Gee 2008).

However, these methods were developed for general textures but not optimized for the 3D building models. The building facade texture has many features such as mainly composed by the linear structure like windows or doors. These features are studied by researchers to improve the 3D city model texture generalisation.

Khan and Okuda (2006) presented a strategy for creating images of reduced details while preserving major features of 3D urban texture images. The texture images are divided into two classes, those with linear edges and others, which are detected by Hough transform. They found that the images with linear edges can be significantly compressed by dividing the images into regions with single colour. In their experiment city block, the size of texture is about 32MB, out of which about 13 MB are the images with linear edges and can be reduced to around 300 KB using their simplification procedure.

Loyaz et al. (2008) proposed a method to create the generalised representations of the textured building facade. They first identified the primary colours of the building texture, and made use of these values to create the parametric structural representation of the building facade. The parametric structural representation was defined as the Fourier Transform of “facade waveforms” generated by projecting the facade image in the horizontal and vertical directions. The limitation of the technique is that it does not address the abstraction of buildings with predominantly glass facades.

2.3.3 Generalisation evaluation

Generalisation of 3D city models could be implemented with a multitude of alternatives, and generates different results. Even for the same generalisation operator, different strategies will have different outcomes. Therefore, it is essential to evaluate these results and select the most suitable one as the final outcome.

Evaluation in cartographic generalisation

Generalisation evaluation methods have been studied by the cartographic society for a long time. Shea and McMaster (1989) tried to find out which generalisation operator is more suitable at a certain time by situation assessment of the conditions for generalisation, the measures indicating the need for generalisation and the controls on how to apply generalisation functionality. Harrie (2001) divided the generalisation quality evaluation into visual assessment, functional
assessment and quantitative assessment. Zhang et al. (2009) quantitatively analyzed the map information in different scales by making use of a landscape pattern index. Filippovska et al. (2009) introduced a method to evaluate the quality of generalised building ground plans. Zhang et al. (2008) proposed an object-oriented density measurement based on skeletonisation of gap space to evaluate the spatial distribution density in map generalisation. Some of the evaluation frameworks are discussed below. However, most of these generalisation quality evaluation methods are proposed for 2D cartographic maps instead of 3D city models.

Filippovska et al. (2009) proposed a framework to evaluate the quality of ground plan generalisation. They first classified the quality of the generalised data on four different geometric aspects: trueness of ground plan, trueness of location, trueness of extension and trueness of shape. Trueness of ground plan can be measured by the largest deviation between the contour of the original and generalised ground plan. Trueness of location can be measured by the translation of the centroid before and after generalisation. Trueness of extension, the area distribution of an object in space can be measured by area difference, oriented bounding rectangles etc. Trueness of shape is a combined property that considers both the outline of a polygon and its area. The geometric properties such as compactness, roundness and convexity are suggested to measure the trueness of shape. In implementation, the trueness of ground plan is evaluated by boundary similarity $R_{\text{boundary}}$, the percentage of the remaining part of the polygon. $R_{\text{boundary}}$ can be calculated with Formula 2.1 (Source: Filippovska et al. 2009). $\text{Buffer}(O)$ is a buffer around the original ground plan polygon which makes the matching of the two contours will not be based on strict geometric conditions, but rather tolerates small inaccuracies. $\text{Perimeter}(G)$ and $\text{Perimeter}(O)$ are the length of the contours of the generalised and the original ground plans.

$$R_{\text{boundary}} = \frac{\text{Buffer}(O) \cap \text{Perimeter}(G)}{\text{Perimeter}(O)}$$

The trueness of extension and shape are measured by the ratio of intrusions ($R_{\text{intrusion}} = \frac{\text{Area}(O \cap G)}{\text{Area}(O)}$), extrusions ($R_{\text{extrusion}} = \frac{\text{Area}(O \cap G)}{\text{Area}(G)}$) and area ($R_{\text{area}} = \frac{\text{Area}(G)}{\text{Area}(O)}$). The total quality characteristic TQ is the Euclidean distance which aggregated all quality parameter like the rate of the boundary similarity, the rate of intrusions and extrusions, and the area ratio.

Area and perimeter are also used by other studies (Podolskaya et al. 2007) to evaluate the quality of generalisation. However, these features can not reflect the similarity of city objects at certain generalisation operations like exaggeration or zooming.
Bard (2004) proposed a more general quality assessment model based on: (1) characterisation of the data in their initial and final states at different levels of analysis; (2) a data quality assessment by comparison of the two characterisations; and (3) aggregation of the various assessment results to summarize data quality. He divided the assessment functions into three parts: characterisation functions, evaluation functions and aggregation functions.

Characterisation functions try to describe the features of the object under assessment by geographic properties (geometric, semantic and topologic), such as area, or perimeter used in the ground plan evaluation method mentioned above. These geographic properties can be at several levels of analysis micro, meso and macro (Ruas 2000). The micro level represents individual features independent of other features such as size, height of a building. The meso level is made of groups of micro or meso features, such as a set of buildings (a micro feature) or a group of urban blocks (meso features). And the macro level is the overall information such as the mean height of all buildings.

The evaluation functions are used to measure the similarity of certain features between two objects. Three kinds of functions are presented: (1) threshold respect functions which check if the assessed property is over or under a threshold of legibility; (2) property evolution analysis functions which assess if a property has evolved as expected; and (3) qualification functions which aim to make evaluation results comparable.

The aggregation functions are used to get overall quality of a generalisation result. It is composed of intra-class and inter-class aggregation. Intra-class aggregation is the assessment values for features of the same class such as building, while the inter-class aggregation is the aggregation of assessment results between different classes such as road and building. Classical operators of aggregation are the mean, the median, the minimum and the maximum. Also the weight set is quit important especially in inter-class aggregation.

There are few studies on quality assessment of 3D city model generalisation. In visualisation applications, the most important criterion for the generalised models is the visual similarity to the original models, so visual similarity is the main criterion to evaluate the generalisation results. Technologies in pattern recognition can be applied to measure the visual concept distance from the original 3D city models to the generalised ones. Mathematically, features of 3D city models can be represented as ARG, whose nodes contain the feature of objects in 3D city models e.g. buildings, and edges contain the relations between nodes e.g. distance between buildings. Therefore, the problem of quality assessment of generalisation can be converted to the matching of two ARGs which represent the original 3D city models and the generalised ones.
respectively. A great deal of effort has been devoted to develop efficient ARG matching algorithms. Kim et al. (2004) proposed a method using the nested structure of Earth Mover’s Distance (EMD) to calculate the difference between ARGs. The difference can be used to represent the visual similarity between 3D city models and evaluate the generalisation results. To measure the similarity, visual salience is introduced.

**Visual salience**

The visual evaluation of generalisation is not only related to the model itself but also to the user experience. Some researchers combine the visual features with the human perception and propose the concept of visual salience that is defined as “the distinct subjective perceptual quality which makes some items in the world stand out from their neighbours and immediately grab our attention” (Itti 2007).

Scientists have shown the effect of visual salience on the human recognition process (Yantis 2005, Cole et al. 2004, Thompson and Bichot 2005). In cartographic society, it is used to detect the landmarks in a map (Elias 2003, Elias et al. 2005) and to measure how much a facade is visible/attractive when approaching a decision point (Winter 2003). In 3D city models related applications, the visual perception is mainly determined by these salience objects. Therefore, in the generalisation process, the salience values of the city objects need to be calculated and these objects with high salient values should be preserved.

In a 2D situation, different methods are developed to identify these salient objects (landmarks) in the map. Raubal and Winter (2002) proposed measures to formally specify the landmark saliency of a feature. Different individual properties for the attractiveness of a landmark were defined and then put together to form a global measure of landmark saliency for each feature in a dataset. They introduced the visual attraction (facade area, shape, colour, visibility, texture, and condition), semantic attraction (cultural importance, explicit marks) and structural attraction (nodes, boundaries). Duckham et al. (2010) applied visual salience in the routing instruction. They presented a model that implements cognitive salience in two ways: by weighting types of spatial features for their expected experiential salience; and applying their relevance in the context of a route.
2.3.4 Multiple representations of 3D city models

With the help of generalisation methods, 3D city models in different LODs will be created automatically. It is essential to integrate these data in dynamic visualisation. Therefore, multiple representation of the data structure is required. The problem of handling multiple LODs in city models is important for continuous scaling (Königer and Bartel 1998). Some multiple representation data structures of city models are discussed in this section.

For 2D objects, tree structures are employed to represent the different map data in multi-LOD or multi-scales (Oosterom and Schenkelaars 1995). The Binary Line Generalisation tree (BLG-tree) is used to represent the polyline. BLG-tree is created based on the Douglas-Peucker algorithm (Douglas and Peucker 1973). Reactive-tree (Oosterom 1991) based on R-tree is used to represent the multiple geometric types, e.g. point, polyline and polygon in the node form of \{MBR, value, id\}, where MBR is the minimal axes-parallel bounding rectangle, value is a positive integer indicating the importance level decided by type, attributes, size etc. of the object, and id contains a reference to the object. Every node can have children which represent the sub objects in its Minimum Boundary Rectangle (MBR). Generalised Area Partitioning (GAP)-tree (Oosterom 1993) is used to represent the hierarchy structure of area partitioning, which more accurately reflects the structure of the map than MBR used in reactive-tree.

For 3D city models, based on R-tree, Coors (2003) proposed Progressive Tree or P-Tree to represent the hierarchical LODs of 3D city models. Compared with R-tree, P-tree has additional elements to store graphical information in this entry. This graphical information can be used to represent all child objects. Minimum bounding box is used to present the aggregation of all children. Figure 2.16 presents a P-tree structure of 3D city models.

![Figure 2.16: Example of P-tree (Source: Coors 2003, pp. 354).](image-url)
Quadtree is also used to represent the texture of 3D city models in different resolution. Parry et al. (2002) described a scalable building organisation structure appropriate for dense 3D urban areas. It permits the handling of highly detailed building facades and the inclusion of other data collected at the same time, such as tree, sidewalk, or lamppost data. As shown in Figure 2.17, the hierarchy is embedded into a quadtree based global hierarchical structure. Each city block contains a latitude and longitude position and is inserted into the global quadtree based on this position. Each block is composed of a set of simple facades to which textures and 3D details are attached.

**Figure 2.17**: Hierarchical structure of 3D urban block geometry (Source: Parry et al. 2002, pp. 3).

Buchholz and Döllner (2005) presented texturing atlas tree (see Figure 2.18) based on quadtree to preserve the 3D city model texture images. In their pre-processing step, a hierarchical data structure is created for all textures used by scene objects, and it derives texture atlases at different resolutions. At runtime, only a small set of these texture atlases are loaded, which represent scene textures in an appropriate size depending on the current viewpoint and screen resolution. According to their experiments, the average frame rates were 17 fps for the original textures, 30 fps without texture switches and 27 fps for the atlas tree. This implementation indicates that the multiple representation structures for the textured 3D city models are necessary.
Wen et al. (2009) also divided the 3D city models into regular tiles and generated a pyramid mode for multi-resolution virtual environment is generated. In their framework the texture tiles are also organised in quadtree structure (see Figure 2.19), where each texture is linked to a unique node in the tree and each node is thus associated with a coverage area, or a tile.
3. Methodology

3.1 Overall structure

Visualisation of 3D city models is a multiple scale task which usually covers from city level, block level, building level, facade level to indoor level. These visualisation levels correspond to the five LODs defined in the CityGML standard. It is necessary to integrate the multiple representations of the 3D city models for the better visualisation such as fast rendering, continuous scaling, and target emphasising.

In this thesis, 3D generalisation methods are employed to create the multiple representation structure of the city models in different LODs (see Figure 3.1). Since the LOD0 (2.5D models) can not reflect the full properties of 3D city models and LOD4 data is not as widely acquirable as others for privacy reasons, this thesis focuses on the generalisation of models in LOD1, LOD2 and LOD3. With the proposed multiple representation structure for each LOD, the generalised 3D city models in different scales are integrated continuously. Then the dynamic visualisation strategies are discussed based on the multiple representation structures in different LODs. Finally, the generalisation results are evaluated by comparing the visual similarity between the original models and the generalised ones. The evaluation is performed both automatically and manually to test the accuracy of the proposed assessment methods. The overall structure of this thesis is shown in Figure 3.1.

Figure 3.1: Overall structure of the thesis.
First, the X3DOM based online visualisation framework is implemented by converting the 3D city models from CityGML into X3D files and updating the 3D scenes in the user browser using Asynchronous JavaScript and XML (AJAX). According to the experimental results, it is difficult for the browser to support the large data volume of 3D city models, which illustrates the necessity of the proposed generalisation methods.

Then the 3D city model generalisation methods on block level, building level and facade level are studied thoroughly. The multiple representation structures are created on each level to store the automatically generated generalisation results for dynamic visualisation. On the block level (LOD1), CityTree, a novel tree structure, is proposed to represent the models in different scales by aggregating the nearby buildings; MST is used to typify the city objects (mainly buildings) in linear structure. On building level (LOD2 and LOD3), a new LOD of the city object is created by projecting the detailed geometrical structures of the building surfaces such as windows, doors, etc. into its exterior shell and an index structure is created specified for the street level visualisation. On the facade level (LOD3), the generalisation of the texture is mainly considered. The dynamic texture compression is carried out in two directions, horizontal and vertical. And for the further texture generalisation, TextureTree is created by representing the texture with colours that is detected from segmentation of the textured geometry. For the multiple representations on block and facade level, user view point based dynamic visualisation strategies are designed accordingly.

The automatic results evaluation algorithms are developed to assess and improve the generalisation results. Visual similarity is the main criteria of the evaluation process. The ARG is used in this thesis to abstract the visual features. The salience features of 3D city models such as height, colour, and size, are taken into consideration. The difference between the 3D city models represented by ARG is computed using NEMD algorithm. The automatic evaluation is compared with the user survey results to testify the effectiveness of the algorithm.

The test data of the proposed visualisation framework is in CityGML format. The data in other format, e.g. KML, will be converted into CityGML. The test datasets in CityGML come from the web site CityGML.org, the official homepage of CityGML, and the datasets in KML come from the ViSuCity project. The rest of the chapter will discuss these parts of the proposed visualisation framework in details.
3.2 Data integration

3.2.1 Integration strategy

The data integration framework used in this thesis is shown in Figure 3.2. The input data of 3D city models in different format such as KML/COLLADA, BIM, and shapefile is converted and integrated into CityGML database. All data management, processing, generalisation and analysis of 3D city models are implemented according to the CityGML standard. The original CityGML files or generated multiple representation structures will be visualised using X3D.

![Figure 3.2: Data integration framework.](image-url)

There are three advantages to the proposed integration framework. First, CityGML supports both geometry and semantic information of the 3D city models, which makes it suitable to be the standard format to integrate data from different domains. Second, the framework provides a standard platform of 3D city model processing. Based on this framework, the developers can focus on creating methods to analyse the 3D city models instead of dealing with the different input data, and it is also to apply the existing processing or easy analysis to the new dataset in a different area or a different format. Third, this framework is extensible. Any data in the new format can be integrated into the framework by just creating the method to extract its geometry or semantic information. Then the corresponding representation form in CityGML will be generated automatically, and the existing analysis methods will be able to be applied to the new data. Since CityGML supports the ADE, any 3D city models...
related data can be supported in the framework, even if it is not defined by CityGML standard.

Next, the details about the geometry features of CityGML, X3D and KML/COLLADA will be introduced to reveal how the proposed framework works.

3.2.2 CityGML

Geometry representation

Objects of GML3 geometry model are adopted by CityGML to represent its spatial properties. The 3D geometry is recorded according to the well-known Boundary Representation (B-Rep, cf. Foley et al. 1995). For each dimension, there is a geometrical primitive: a zero-dimensional object is a Point, a one-dimensional a Curve, a two-dimensional a Surface, and a three-dimensional a Solid (OGC 2008). In CityGML, Surfaces are represented by Polygons, which define a planar geometry. The CityGML specification (OGC 2008) provides an example of CityGML file as shown in Figure 3.3: a gml:Polygon with an id wallSurface4711, which is part of the geometry property lod2Solid of a building. Although CityGML supports different types of surfaces such as OrientableSurface and TriangulatedSurface, only the basic surface object is analysed in this thesis to simplify the processing and concentrate on generalisation/visualisation.

Figure 3.3: CityGML surface example (OGC 2008, pp. 25).
Texture

Textures in CityGML are always raster-based 2D textures. The raster image is specified by imageURI using a URI and can be an arbitrary image data resource, even a preformatted request of a web service. The specification of texture wrapping comes from COLLADA. For accessing a texture outside the image raster, five types of wrapModes are defined: none, wrap, mirror, clamp and border as shown in Figure 3.4.

![Image of different texture wrap modes](image)

**Figure 3.4:** CityGML texture (a) applied to a facade using different wrap modes: (b) none, (c) wrap, (d) mirror, (e) clamp and (f) border. The border color is red. The numbers denote texture coordinates (Source: OGC 2008, pp. 33).

The texture images are projected onto the geometry surfaces by texture coordinates. According to the CityGML standard, texture coordinates are applicable only to polygonal surfaces. An explicit mapping of a surface’s vertices to points in texture space is defined in the Appearance tag. Figure 3.5 presents an example of the texture mapping.

![Image of texture mapping](image)

**Figure 3.5:** Positioning of textures using texture coordinates (Source: OGC 2008, pp. 35).
Geometry and texture analysis

To simplify the analysis of the geometry objects, all the polygons in CityGML are converted into the Polygons of JTS Topology Suite, an API of 2D spatial predicates and functions (JTS 2011). Since JTS does not support 3D analysis, we need to project the 3D coordinate into 2D. The details of projection will be introduced in section 3.4.3 on street level analysis. After the projection, the topological operations of JTS polygon class such as getting area, getting length, etc., will be easily applied on the CityGML data. The operations for generalisation like merging, movement, extension, etc. are also implemented based on the JTS polygons.

For texture analysis, the Java `BufferedImage` class is used. Since some texture can be reused many times in different surfaces, a table is created in which the content is the `BufferedImage` object and the analysis results, and the key/ID is the texture image URL and the mapping coordinates. When analysing a new textured polygon, we first check if it is already loaded by searching the table with the texture image URL and coordinates of the input polygon. It will reduce the computation overload quite much and save the system memory.

3.2.3 X3D

The geometry and texture definition in X3D are different from CityGML. In X3D specification, seven 3D geometry nodes are supported that are Box, Cone, Cylinder, Elevationgrid, Extrusion, Indexedfaceset, and Sphere. The Elevationgrid is suitable to define the DEM of the 3D city models as shown in Figure 3.6.

![Figure 3.6: Elevationgrid Node in X3D (Web3D 2008).](image-url)
The surface in CityGML is composed by the polygons, and the *IndexedFaceset* in X3D is suitable to represent the set of polygons. According to the X3D specification, *IndexedFaceSet* node represents a 3D shape formed by constructing faces (polygons). An example of the *IndexedFaceset* in VRML format (same as the X3D, but more compressed) is shown as below. The points are first defined in the *coord Coordinate* field and then the polygons are specified by the *coordIndex*. The corresponding 3D geometry is shown in Figure 3.7 (Carey and Bell 1997).

```
IndexedFaceSet {
    coord Coordinate {
        point [ 1 0 -1, -1 0 -1, -1 0 1, 1 0 1, 0 2 0 ]
    }
    coordIndex [ 0 4 3 -1 # face A, right
                   1 4 0 -1 # face B, back
                   2 4 1 -1 # face C, left
                   3 4 2 -1 # face D, front
                   0 3 2  1 ] # face E, bottom
}
```

*Figure 3.7*: IndexedFaceSet Node in X3D (Carey and Bell 1997).

*IndexedFaceset* also supports image texture by defining the mapping of texture coordinates to polygon coordinates which is the same as in CityGML. Figure 3.8 shows an example of texture mapping in X3D.
Figure 3.8: IndexedFaceSet node with texture (Source: Web3D 2008).
3.2.4 KML/COLLADA

Besides the CityGML data, the 3D city models in KML/COLLADA format are also integrated in the proposed framework. The data is supplied by Blom Company who automatically generates the 3D city models from LiDAR scanning and oblique aerial photography. Blom utilizes the KML/COLLADA format to export its 3D city models which is structured as shown in Figure 3.9:

![Figure 3.9: An example of the file structure used in Blom 3D city models (Source: Blom 2011).](image)

This structure includes the following parts:

- **KML**: One KML file which contains the geolocation information and the reference to the 3D model
- **COLLADA**: One ADE file that contains the geometries of the buildings modelled and the references to the texture imagery
- **Images**: Several JPG files which contains the texture images of the model

The COLLADA files describing 3D city models from Blom is shown in Figure 3.10

```xml
<?xml version="1.0" encoding="utf-8"?>
<COLLADA version="1.4.1" xmlns="http://www.collada.org/2005/11/COLLADASchema">
  <asset>...
  <library_images>...</library_images>
  <library_effects>...</library_effects>
  <library_materials>...</library_materials>
  <library_geometries>...</library_geometries>
  <library_visual_scenes>...</library_visual_scenes>
  <scene>...</scene>
</COLLADA>
```

**Figure 3.10**: Structure of a Blom 3D model.
The element <asset> contains general information about the COLLADA file, such as author, creation data, etc. The following three libraries which contain the elements of <image>, <effect> and <material> are used to apply colours and textures to the 3D models. The <material> element is an instance of <effort> element which could be image texture represented by <image> or colour. <library_geometries> defines the all <geometry> elements used in the COLLADA files. In the <geometry> element the triangles, texture mapping and feature (roof, facade, etc.) of the building surface are defined in the same way as CityGML. These geometries will be composed into 3D scenes of the city models. These 3D scenes are also organised as library and instances that are indicated by <library_visual_scenes> and <scene> respectively.

An open source Java COLLADA parser program (jME 2011) is used to read the Blom data. They will be converted into CityGML files with the export function in citygml4j. The geometry, texture and semantic information of the 3D city models are all preserved and integrated with the CityGML data.

### 3.3 Online visualisation

Although 3D city models can be viewed in different software, e.g. LandXplorer (Autodesk 2011) for CityGML, Google earth for KML/ COLLADA, etc., it is still important to develop the 3D visualisation methods for web browsers since they are currently the most widespread user interface application. Because the current web standard (HTML4.01) does not specifically support the 3D objects, many plugins is developed for browsers to visualise the 3D models, e.g. Adobe Flash, Microsoft Silverlight, Java3D etc. Two drawbacks of these plugin-based systems are pointed out by Behr et al. (2009). First, these 3D plugins are not installed by default on most systems, so the users have to deal with the issues like installation, security, incompatibility etc. Second, it may double or triple the workload of the developers to extend their 3D web applications to several different plugins.

Therefore, in the latest web standard HTML5, X3D is specified as the standard content of 3D scene. The Web3D Consortium has recommended X3D directly inline as Document Object Model (DOM) in HTML for web 3D visualisation. X3DOM is being considered as the potential standard implement to integrate X3D with HTML.

![Figure 3.11: Moving from a loose plugin-based Scene-Access-Interface (SAI) to the tightly integrated X3DOM model (Source: Behr et al. 2009, pp. 1).](image-url)
As shown in Figure 3.11, X3DOM model tries to directly integrate the X3D nodes into HTML5 as DOM element and visualise them through web browsers. The namespace prefix x3d is used on all nodes belonging to the X3D scene which is indicated by <x3d> tag. Figure 3.12 is an example of HTML5 file with X3D content, which will create a 3D box in the browser.

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
X3D DOM integration and manipulation
</title></head>
<body>
<h1>X3D DOM integration and manipulation</h1>
<x3d xmlns:x3d="http://www.web3d.org/specifications/x3d-3.0.xsd">
    <Scene>
        <Shape>
            <Box x3d:size="4 4 4" />
        </Shape>
    </Scene>
</x3d>
</body>
</html>
```

**Figure 3.12:** X3D in HTML5 (Source: Behr *et al.* 2009, pp. 7).

X3DOM also supports the interaction with users. There are three DOM changes that affect the X3D content in HTML: removes/inserts of elements and changes of elements attributes. User actions in the 3D scenes such as touching, clicking, moving, etc can also be detected. The user profiles in the scenes such as view point and angle are also available. Based on these user information (actions and profiles), the corresponding changes in X3D content can be triggered in the dynamic visualisation.

XMLHttpRequest (XHR) can be used to dynamically update the X3D content in the HTML files. XHR is a DOM API “that provides scripted client functionality for transferring data between a client and a server” (W3C 2011). XHR allows asynchronous data requests without page reload, so that the X3D models can be preloaded or cached into the user browser. Even though, the existing 3D city models are still too large for browser visualisation at one time. Therefore, the generalisation methods are required to simplify the 3D city models on city block, building and facade levels.
3.4 Block level generalisation

On the block level of 3D city models, the number of the objects is huge and the data volume is quite large. Generalisation is required not only to reduce the data volume and improve the visualisation efficiency, but also to emphasize the interesting/target parts such as the destination in the navigation. On the block level the overall structure is more important than the details of each object in the visualisation, because the distance from the user viewpoint to the city objects is too long to reveal all the details. There are many structures or patterns on the city block level, e.g. linear, grid, etc. In this section, Delaunay Triangulation and MST are used to detect two common structures on the block level, the neighbourhood and linear structures. For the neighbourhood structure, the aggregation operation is applied, and for the linear structure, the typification operation. Since the 3D city models on the block level are widely distributed in a relatively large area, and their distances from the current user viewpoint could be quite different. Therefore, it is necessary to create the multiple representation structures to support the different generalisations in dynamic visualisation. In this section, CityTree and linear building group are used to represent the 3D city models on block level.

3.4.1 Neighbourhood structure

A common method used in 2D cartographic generalisation is to aggregate the neighbourhood objects. This operator reduces the number of objects by aggregating objects nearby to one. This method is based on the algorithms for clustering objects (see Murray and Estivill-Castro 1998, Andres and Sester 2000). It is essential to identify the neighbourhood relationships among the city objects. Joubran and Gabay (2000) made use of the Delaunay triangulation to analyse the spatial distribution of the 2D map objects. Delaunay Triangulation is suitable for cartographic purposes, since it supplies triangles with especially short edges. According to Shewchuk (1996), Delaunay Triangulation “is based on the shortest distance criterion, defined by the condition claiming that the circumscribing circle for each triangle does not contain any point from the set of data”. Some cartographic properties of the Delaunay Triangulation are mentioned by Saalfeld (1993):

(1) The principle of the circumscribing circle produces a situation in which each edge of the triangulation triangle edges joins between the pairs of points most adjacent to each other;
(2) Delaunay Triangulation is preserved after any transformation on a set of points;
(3) Delaunay Triangulation produces the maximum value of the smallest angle of all the angles in the triangulation;
(4) Delaunay Triangulation requires local updating when adding or deleting a point.

Many algorithms are proposed to calculate the Delaunay Triangulation (Su and Drysdale 1996). Some of the algorithms can have $O(n \log n)$ time complexity in which $n$ is the number of input points.

In this thesis, 3D city objects are simplified into 2D points and the Delaunay Triangulation is generated based on these points. The JTS is used to implement the Delaunay Triangulation. According to our test, the calculation time is around 2 min in a PC for a graph with approximately 2000 points. This performance is acceptable for experimental purposes, since all these computations are performed before dynamic visualisation. The generated triangulation is further modified by removing edges crossing a road or longer than a predefined value, because it is not reasonable to aggregate the buildings which are located on different sides of a road or are too far away from each other. Based on the modified Delaunay Triangulation, the connected objects are aggregated and a multiple representation structure CityTree is created.

The 2D polygon aggregation has been intensively studied in the past. For the rigid objects such as buildings, aggregation can be implemented by closing operators (“expand and shrink” or “dilation and erosion”) in the mathematical morphology (Gonzales and Woods 1993) or by moving one of the objects (Bundy et al. 1995).

For the 3D city models, Lal and Meng (2001) explained the necessity of the 3D aggregation, e.g. reducing complexity, maintaining the structure and distribution pattern of the group, enhancing/exaggerating the impression of information density across a map. They proposed some rules for the 3D aggregation such as touching, proximity, alignment, angle, height, roof type, geometry etc. Next, the different view types like Top view, oblique view or fish-eye view and different perspective viewing heights were also taken into consideration in the 3D aggregation methods. Finally, the graphic, topological, structural and perceptual constraints for different types of city objects like road, building, etc. were defined for the 3D aggregation which will be considered in the proposed method discussed in section 4.

### 3.4.2 Linear structure

From the perception point of view, in the real city, many objects (buildings) are distributed in some patterns which are quite important characters in visualisation according to the Gestalt Theory (Rock 1983). Some of these efficient criteria are used (Regnauld 1998) in the context of cartographic generalisation, e.g.
typification. The patterns are detected based on the visual sensitivity to the group according to the following criteria (Boffet and Serra 2001):

- Proximity, when objects are close to each other.
- Similarity, when objects look similar.
- Continuation, when there is regularity between objects.
- Symmetry, when symmetry of positioning is verified.
- Enclosure, when a particular configuration of objects forms a well known shape.

For all the patterns in the city, roads/linear structures are quite common. Hangouët (1998) associated buildings to the nearest road by combining proximity and regularity parameters between buildings and roads. Regnauld (1998) applied the MST to detect the linear structures in the building area. The results showed that MST is suitable for the linear pattern. Therefore, MST is adopted and extended in this thesis for 3D typification.

A MST is one type of valued graph, in which

- All nodes are connected
- The length sum of the links is minimised.

An early MST algorithm was proposed by the Czech scientist Otakar Boruvka in 1926. The two common methods today are Kruskal’s algorithm (Kruskal 1956) and Prim’s algorithm (Prim 1957). All these three methods are greedy algorithms that run in polynomial time. In this thesis the Prim algorithm is selected to implement the MST calculation. The algorithm continuously increases the size of a tree, one edge at a time, starting with a tree consisting of a single vertex, until it spans all vertices. The time complexity of the algorithm for the adjacency matrix is $O(n^2)$ in which $n$ is the number of nodes.

To extend the 2D linear structure typification into 3D city models, the MST is generated based on the complete graph of the building nodes. The graph is weighted by their distances and cut by the roads. Then the MST is further divided into line strings to form the building chain based on which the typification is implemented as described in Paper III.
3.5 Building level generalisation

3.5.1 Shell models

In many 3D city related applications such as road navigation, urban planning, etc., the indoor details are not necessary and should not be visualised for most of the city objects due to privacy considerations. The model appearances are enough for these applications. Therefore, by creating the exterior shell representations of the detailed 3D city objects, not only the visual efficiency is increased but the user privacy is also protected.

Based on the shell models, the building objects can be further generalised by different methods. Thiemann and Sester (2004) made use of the polyhedron segmentation (Ribelles et al. 2001) to decompose building models into Constructive Solid Geometry presentation in which the solid is constructed from primitive elementary 3D-objects such as box, column, sphere, etc. using Boolean operations (set operations). The ground plan and roof type of the shell model are analysed by Lal and Meng (2003) to find recurring templates and essential parameters describing the building. Lal and Meng (2004) also implemented an automatic planar-structure building type recognition algorithm based on a hierarchical neural network. Meng and Forberg (2004) summarised the model generalisation of 3D buildings. In their context, a building is considered to have three generic surface types: a ground plan (horizontal facets), a number of walls (vertical facets), and a roof (inclined or horizontal facets). This is exactly the output of the proposed shell model generation algorithm. Therefore, the shell model is an important bridge linking the models in higher LODs into lower LODs.

To extract the exterior shell of the detailed 3D city models, four steps are employed in the proposed methods: (1) extract exterior shell of wall elements, (2) project the selected polygons of windows and doors on the corresponding walls, (3) select the polygons of windows and doors whose plane are parallel to the walls where the windows and doors are sealed, and (4) extract the exterior shell of the roof structure. The details of the algorithm implementation are described in Paper III.

3.5.2 Street index

Another common city visualisation pattern is the street level view where most of our daily actives take place. Many imagery based online street view systems are published after Google launched the Street View feature of Google Maps in May 2007 (Vincent 2007). These systems draw quite much attention and show their unique ability in navigation or surround area demonstration. However, the
imagery-based street view has many limitations such as being available only along the camera route, difficult to view a building at a certain angle, containing many unwanted objects (cars, people), etc. In addition, privacy is a problem. Although the privacy protection method is proposed for Google Street View (Frome et al. 2009), some government authorities show their concerns for potential leak of sensitive personal information.

Compared with the imagery, 3D city models have advantages in flexibility and privacy protection in street level visualisation. The user can view 3D building models in any angle/distance and the personal information can be completely removed from the 3D city models. Although it is much more difficult to create the 3D street models than just acquiring the imagery, many automatic or semi-automatic methods are proposed for 3D street-side modelling such as artificial synthesis of buildings based on grammar rules (Muller et al. 2006), 3D scanning of street facades (Fruch and Zakhor 2003), and image-based approaches (Debevec et al. 1996, Xiao et al. 2008). It is reasonable to expect the widespread use of 3D city models on street level.

One of the most challenging issues of the street level 3D model visualisation is the data volume, especially for the online application, since the 3D city models are very detailed in the street level view as the distances from the view point are small, especially for the buildings beside the road. The textures are essential for a nice visualisation in street view, so the models are often data intensive. Currently it is difficult or even impossible to visualise thousands of buildings in such details. Therefore, generalisation of the 3D city models is required. Considering the fact that not all of the 3D objects can be seen at a certain view point on street level (actually the visible objects are quite limited), the selection operator is used for street level generalisation.

In the proposed method, in a certain view point, the visible objects are first selected and then ordered by their visual importance. The visible detection is implemented by first projecting all the 3D surfaces into a 2D camera plane, and then calculating the coverage relationship between every two surfaces. If a surface is completely covered by others, it will be set as invisible. If all surfaces of an object are invisible, the object is set as invisible.

The visual importance calculation is based on the visual salience which is defined by Itti as “the distinct subjective perceptual quality which makes some items in the world stand out from their neighbours and immediately grab our attention” (Itti 2007).

In the dynamic visualisation framework proposed in Paper V, only the $n$ most visual important objects will be visualised in a certain view point. If the number
of visible object at certain points is less than $n$, then all will be visualised. The number $n$ is decided by the user context, for example, $n$ may be set to 20 for the mobile phone, while it is set to 40 for the PC client with high speed network. The visibility and visual importance computation are a time consuming task. It is inefficient to calculate the information in the dynamic visualisation stage. Therefore, the pre-processing is required to create indexes of the visible buildings, so that the results are directly available in the dynamic visualisation. However, the possible view points in the 3D space are unlimited, so it is a problem to pick up the pre-computed view points as the indexes. Since the focus is on street level, the view points along the street are selected and the view directions are limited along and against the street. In dynamic visualisation, the nearest index covers the current view point will be selected and the visible objects in this viewpoint will be loaded according to their visual importance. By replacing the real visible buildings with the nearest visual index, some buildings may be missed. The visual difference based on the visual salience is computed between the real visible objects and the indexed objects, which illustrate that these missed buildings are usually not important for the visual purposes and the proposed method is efficient on street level visualisation.

3.6 Facade level generalisation

Texture is important for 3D city models. It not only makes the 3D city objects, e.g. buildings, more realistic in visualisation, but also provides the information in intricate applications such as navigation. One main problem of the textured 3D city objects is the huge data volume. Currently, the texture of the 3D objects is usually saved as image data. Compared with the XML formatted geometry and semantic information of 3D city models, the texture images often represent a large proportion of the whole model. Besides, the XML text data can be effectively compressed by different software such as winrar, gzip, etc., while the compressed texture image files are still quite large. For some manually generated 3D city building models in CityGML, the ratio between the size of compressed xml file and image files is 1:3.92, and if the textures are optimized manually. For the automatic generated 3D city models (Blom 2011) the image files basically take over 90%-98% of the total data volume.

It is essential to generalise the texture data in 3D city models, which has not been intensively studied from the generalisation perspective before. The existing texture compression or simplification methods are mainly proposed for the general 3D objects. In Paper VI of this thesis, the features of 3D building facades will be analysed for texture image compression. For further model generalisation, the texture will be segmented into monochrome parts and represented by their domain colours.
3.6.1 Texture compressing

The 3D building facade texture has the feature that horizontal and vertical structures (e.g. rectangular windows, doors, etc.) are quite common, since walls are basically vertical to the ground for architectural reasons. These structures are also essential for visualisation. For example, if the edges of a window are slantwise, it will be easily recognized even from far away. Because people are usually look upward, downward or around, but rarely rotate the sight around the view line.

For manually generated 3D building models, the texture is usually organised as the combined rectangles from the different building parts as shown in Figure 3.13a to save storage space. But for the automatically generated models, the texture is in different orientations as shown in Figure 3.13b.

![Figure 3.13: Texture images of 3D building models.](image)

For the automatically generated irregular-shaped (non-planar) texture, Tsai and Lin (2007) proposed a two-step adjustment method as shown in Figure 3.14. The first step was to approximate the top and bottom lines of the facade with polynomial functions and then stretch the image along its vertical axis so that the top and bottom curves of the facade were pulled into horizontal lines as shown in Figure 3.14b. The second step was to stretch the image horizontally until the vertical lines were aligned correctly (Figure 3.14c). Figure 3.15 is an example of the texture adjustment.
According to the common view pattern of 3D city scenes, the horizontal and vertical structures should be preserved by creating different compression levels of the texture in these two directions. For example, if we look at a wall from the side, the vertical features should be less compressed than the horizontal ones; if we look from bottom up, the compression level in the horizontal direction should be reversed. The compression rate should also take the distance from the viewpoint into consideration. Meanwhile, since the texture images are organised in horizontal or vertical fashion, the compression in these two directions can be easily implemented as the compression in the $x$ and $y$ axis of the image files. In paper V, the proposed dynamic texture image compression method is discussed in detail.
3.6.2 Texture colouring

Although, image compression can reduce the volume of the texture files (up to 98% in this thesis), there is still room for further generalisation, especially for objects far away from the viewpoint. To represent the texture with colour will dramatically reduce the overall data volume and increase visual efficiency. It might affect the realistic level of the 3D city model, but for the background objects in a 3D city scene, the visual difference is barely noticeable.

In order to map the facade texture into colours, the image segmentation method is studied. Compared to other image segmentation methods, the mean shift algorithm (Comaniciu and Meer 2002) has better performance according to the extensive experimental results. Figure 3.16 illustrates the process of mean shift based segmentation and Figure 3.17 shows the segmentation result of the building facade.

![Figure 3.16: Visualisation of mean shift-based filtering and segmentation for gray-level data. (a) Input. (b) Mean shift paths for the pixels on the plateau and on the line. (c) Filtering result. (d) Segmentation result. (Source: Comaniciu and Meer 2002).](image)
After segmentation, TextureTree, a multiple representation structure, is generated to store the segmented texture for dynamic visualisation. To create TextureTree, the region boundaries (Figure 3.17c) of texture are first detected by Canny edge detection algorithm (Canny 1986). Since the horizontal and vertical edges are the main feature in the texture, the texture images will be iteratively divided in these two directions. This process is recorded in a segmentation graph based on which TextureTree is generated. The specific process of creating the TextureTree is described in Paper V.

### 3.7 Generalisation evaluation

In the automatic 3D city model generalisation, the results evaluation algorithm is crucial because it not only tells if the results are suitable for the visualisation but it can also be used to improve the generalisation methods. Harrie (2001) described three different approaches to assess the generalisation quality of 2D maps: visual, functional and quantitative assessment. The visual assessment is performed by comparing the generalised models with the source models. The functional assessment is to study the applicability of the generalised data set in different applications such as map readability or way-finding. The quantitative assessment is to measure the spatial accuracy. These approaches can also be applied to 3D city models. In this thesis, the evaluation is mainly performed in visual aspect by considering the spatial accuracy and the relationships between objects.

In this section, the visual features of the 3D city models will be studied and extracted to represent their characteristics. The visual difference between the original and the generalised models will be calculated based on these features to evaluate the efficiency of the generalisation.
3.7.1 Feature extraction

To calculate the visual similarity between the original and the generalised 3D city models not only the features of each object, but also the relationships between objects should be taken into consideration. Furthermore, for different viewpoints and view angles, the visual perceptions of 3D objects are quite different, so the absolute and relative similarities of the 3D city models are defined.

The absolute similarity is calculated based on the models themselves without considering user’s viewpoint. Many 3D object matching algorithm have been proposed in the computer vision field (see Osada et al. 2001). The visual features of a 3D objects are quite complex and it requires sophisticated methods to abstract these features and compute their difference. Meanwhile, 3D city models have their own characteristics compared with general 3D objects. For example, the models are usually distributed in a widely range in the horizontal while they have less variations in the vertical, except for some mountainous area or central business districts in the metropolis. And the number of objects is usually large while each model is relative simple (LOD1 or LOD2). Therefore, it is useful and simple to evaluate the generalisation results of 3D city models with the relative similarity.

The relative similarity is decided by both the 3D models and the user’s viewpoint. The idea of this method is to project the 3D objects into the 2D plane according to the user viewpoint and then calculate the visual similarity based on the projected 2D shape. The projection can reduce the complexity of the matching algorithm dramatically, but it has constraints. Figure 3.18 (Edgar Mueller 2011) shows an example of the 3D pavement art which tries to simulate the 3D realistic in certain viewpoint (Figure 3.18a) while complete unrecognisable in other viewpoints (Figure 3.18b).

![Figure 3.18](image)

(a) View in the designed angle          (b) View in a different angle

**Figure 3.18**: 3D Pavement art (Source: Edgar Mueller 2011).
In this thesis, two kinds of methods are used for certain generalisation methods. For the block level typification, the absolute similarity is employed to reflect the topology of the 3D city models, since the topology is independent from the viewpoint. The footprint area of the building and the distances between each other represent the feature and relationship respectively. For the street level generalisation, the objects are projected in different viewpoint on the street. Although the generalised model could be quite different from the original if we look from a different angle statically, the framework can dynamically update the models according to the current viewpoint. In the street level evaluation, the visual salience such as colour, size and shape of the projected objects are considered as the object features; the $x$ and $y$ differences of the projected shape in the camera screen compose the relationships.

### 3.7.2 ARG and NEMD

To represent the features and relationships of city models, a graph structure is required in which the node stands for the object with features and the edge stands for the relationships between objects. Such graph structure has been used widely in the pattern recognition field and is known as ARG (Sanfeliu and Fu 1983). In ARG, as used in this thesis, the nodes represent objects like buildings or parts of buildings, and edges represent relationship between building (parts). The visual similarity between the original and the generalised 3D city models will be calculated based on their corresponding ARGs.

Since ARG is an effective and widely used structure in object recognition, a large number of algorithms are proposed to implement the efficient and robust ARG matching. Two steps are usually contained in the ARG matching procedure. The first step is to create a distance matrix of every pair of nodes in two ARGs based on the predefined measure on unary attributes and binary relations. The second step is to establish the correspondence between the nodes based on the distance matrix and generate the overall distance of two ARGs. These two steps are similar to the point matching procedure, so the NEMD (Kim et al. 2004) is employed.

The NEMD’s robustness performance is better than other ARG matching algorithms. Kim et al. (2004) compared the NEMD with some selected higher performance methods such as the graduated assignment graph matching (GAGM, see Gold and Rangarajan 1996), Faugeras-Price relaxation labelling (FRPL, see Faugeras and Price 1981), spherical approximation graph matching (SAGM, see Wyk and Wyk 2000), and the least squares graph matching (LSGM, see Wyk and Clark 2000). Figure 3.19 shows the matching results between the random generated graph and its sub-graph as the noise (Epsilon) increases. It indicates the NEMD is more robust than others.
Figure 3.19: Estimated probability of correct node-to-node matching as noise increases (Source: Kim et al. 2004, pp. 4).

However, the computational complexity is relatively high for NEMD $O(n^2n'^2 \max(n,n'))$ in which $n$ and $n'$ are the nodes number of two ARGs. In this thesis the number of nodes is limited since the generalisation is implemented in the segmented part of the whole 3D city models (e.g. linear building groups in typification and visible polygons in the street view generalisation). Therefore, it is suitable to employ the NEMD method in the proposed generalisation results evaluation. The details of the NEMD calculation are described in Paper III and V.
4. Results and discussion

In this chapter, the main results of the listed six papers are summarised and the contribution of each study to the knowledge in the field is discussed. The connections between the papers are shown in Figure 3.1.

Paper I describes the basic framework of online 3D city model visualisation that explains the reason and application basis for the 3D city model generalisation. Based on the requirement of 3D city model visualisation especially for online visualisation, the generalisation methods on block level, building level and facade level are proposed in Paper II -VI.

On block level, Paper II introduces CityTree, a multiple 3D city data representation structure, for more efficient visualisation based on the aggregation method. Paper III utilizes the MST to detect linear structures in the city for typification. On building level, shell models are created in Paper IV to simplify the buildings in high LODs. In Paper V, the building selection is implemented for street view and the visibility index is created to support the high performance dynamic visualisation. On facade level, Paper VI describes a method that first compresses the building texture in horizontal and vertical directions; then it divides the textured polygons into rectangular segments with different colours which are preserved in TextureTree, a multiple representation data structure, for the dynamic visualisation.

To evaluate the generalisation results, the ARG/NEMD based visual similarity computing algorithm is applied in Paper III and V. In Paper IV, the user survey is conducted to evaluate the generalisation assessment methods.

4.1 Paper I: Online visualisation

The objective of the study in Paper I is to develop a framework for 3D city model visualisation. This framework should be able to integrate data from different sources, to support urban 3D analysis, to enable model generalisation, to allow user interaction etc. and to visualise 3D scenes via the Internet. Meanwhile, as a research prototype, this framework should also be easily extended and implemented. Therefore, CityGML is chosen as the data source because it is the 3D city model standard of OGC; Java is used for 3D analysis because it is widely supported by open source projects and is easily extended and implemented; X3D is selected for 3D visualisation because it is the 3D scene standard of the Internet and supports real time user interaction.

The proposed framework for online 3D visualisation in Paper I is shown in Figure 4.1. Data in CityGML format is parsed and converted into Java classes to
represent city objects such as buildings, roads etc. These classes may contain both geometry and semantic information (Citygml4j 2011). Then based on the geometry or semantic information, scenes of 3D city models are generated. The so created 3D scenes are then viewed in a standard web browser using X3DOM (Behr et al. 2009). X3DOM is a model that directly integrates X3D nodes into HTML5 DOM content based on WebGL, a JavaScript binding to OpenGL to enable rich 3D graphics within a browser.

![Figure 4.1](image.png)

**Figure 4.1**: The framework for online visualisation of 3D city models (Paper I).

The input data set is in CityGML format, and the output visualisation results are in X3D. This process is quite easy to implement and is suitable for research applications such as a study of 3D model generalisation, spatial data analysis and demonstration. Since CityGML and X3D are both XML based, it is possible to convert between them using Extensible Stylesheet Language Transformations (XSLT 2011). This framework is more suitable than XSLT based visualisation for 3D city model related applications. XSLT treats all objects separately, but our generalisation methods require the relationships between these objects. Therefore, the standard Java classes are chosen in the framework.

A key part of our framework is the transform from CityGML to X3D. All LODs in CityGML are represented by GML multi-surface set which can be directly converted into X3D surface set object. While some other X3D elements are more suitable for lower LODs (LOD0, 1). LOD0 is a two and a half dimensional digital terrain model (see Figure 2.1), which could be converted to ElevationGrid objects in X3D (see Figure 3.6). In Paper I, the visualisation of texture is also discussed and the visualisation of 3D city model in CityGML through the Internet is implemented with the X3DOM.

Visualisation of 3D city models with CityGML and X3D is not new. The contribution of this study is mainly on the implementation side. The proposed framework has five features that make the framework suitable for online
visualisation of 3D city models. First, CityGML is used for 3D city model representation that facilitates the integration of datasets from different sources. Second, based on this framework, any Java library can be integrated for city data analysis in the server side. Third, AJAX technique creates a better user experience by updating the 3D scenes without interfering with the display and behaviour of the existing page. Forth, X3DOM liberates the developers from 3D visualisation details and the users from different types of browser plugin management. Fifth, user interaction is supplied by the visualisation language X3D. Therefore, compared with the current online visualisation systems, the proposed framework is much simpler and can be easily implemented and extended while preserving efficient performance.

This paper implements a framework for online 3D city model visualisation, and demonstrates the necessity of generalisation. Compared with the data volume of 3D city models, the capacities of browser and bandwidth are limited. The experiments in this paper indicate that the 3D city models should be generalised on different levels and be organised as multiple representation structures, which are the main research objectives in the rest of papers in this thesis.

4.2 Paper II: Aggregation and multiple representation

Since the data volume for representation 3D city models usually is extensive, generalisation is necessary to reduce the data volume and highlight the interesting parts for more efficient visualisation. Meanwhile, 3D city model visualisation in different scales of the models is also required by many applications such as navigation and planning. Therefore, multiple representation data structure of the city models is necessary for continuous scaling and real time visualisation. The research objective of this study is to create a multiple representation data structure of the city models with generalisation method to support continuous scaling.

CityTree, a multiple representation data structure of the city objects, is created to effectively implement the continuous scaling and dramatically reduce the loading time of 3D models. By utilizing CityTree, it is possible to have dynamic zoom functionality in real time. When the 3D city models in different LODs are generalised, it is required to first convert the model from higher LODs to lower LODs. Then nearby buildings are aggregated together to simplify the 3D city models. In our implementation all buildings are converted to LOD1 with only ground plan and height information.

A binary tree based structure, CityTree, is proposed for representation of multiple LODs data. Figure 4.2 is a demo example of the CityTree. Figure 4.2(a) shows the distribution of the original city building group which contains 5
buildings (1-5). The rectangle areas (A-D) are created by selecting nearby buildings. The CityTree is generated as shown in Figure 4.2(b). The leaf nodes (1-5) are original objects in city model. The other nodes (A-D) are new generated middle nodes with geometry feature of aggregation of children models.

Figure 4.2: CityTree model (Paper II).

In the visualisation step, the CityTree nodes are selected based on the user’s viewpoint and the features of the node. If the parent node is selected to visualise, all his children will not be loaded, which can reduce the computational complexity dramatically.

Figure 4.3 describes the work flow of the CityTree creation. The ground plan of a building is generated by projecting its surfaces onto the horizontal plane and merging them together. If the generated ground plan is not a single polygon, then save the model as multiple buildings, in which each polygon represents one building. Then simplify each ground plan polygon with the algorithm proposed by Sester and Brenner (2004) and modified by Fan et al. (2009). Next create the Delaunay triangulation with each centroid of ground plans, cut the triangulation by the road and cluster the connected buildings into groups. For each building group, create a CityTree for dynamic visualisation by iteratively merging the two buildings with the shortest distance in the group as shown in Figure 4.2.
The main contribution of this paper is the proposed framework of visualisation based on a multiple representation data structure (CityTree) and generalisation methods to generate the multiple LOD representations. Delaunay triangulation and road network are used to cluster the concerted city objects, e.g. buildings, in this paper. An improved ground plan aggregation algorithm is proposed that generates reasonable results. Based on clustering and aggregation, CityTree, a multiple representation data structure of the city objects, is created and used to visualise the multi-scale 3D city models dynamically.

Compared with existing Reactive-tree (Oosterom 1991), GAP-tree (Oosterom 1993) or P-tree (Coors 2003) that are generated by dividing the area top-down, CityTree is easily implemented because it is created by merging the nearest two nodes iteratively bottom-up. Also, CityTree is based on a more reasonable building neighbourhood relationship indicated by the building Delaunay triangulation cutting by road network. The aggregated nodes are generated from the merged ground plan polygons and weighted heights, which has better visual effects than just the minimum bounding box in P-tree or quadtree (Wen et al. 2009).

In Paper II, we improve the visualisation efficiency using an aggregation method, based on which a multiple representation structure is created. The algorithm proposed in Paper II can be used to generalise the dense areas in the 3D city models, while the method in Paper III can be applied to sparse city areas.
4.3 Paper III: Typification of linear building group

Paper III is mainly about typification of 3D city models. In the cities, there are many “regular” areas suitable for typification. By “regular” we mean that the buildings in these areas have some common property, e.g. evenly distributed along a road. The suitable generalisation operator for these regular areas is typification that replaces a large number of objects by a smaller number of objects while trying to ensure that the typical spatial structure of the objects is preserved (Shea and McMaster 1989). In this paper, the linear structure of the city object is adopted for typification.

The typification methodology proposed in this thesis is described in Figure 4.4. The linear building groups suitable for typification are selected using MST. For each group, the typification method is removing the buildings with minimum influence to the overall visual effect. The removed buildings selection is based on the visual similarity evaluation method. Then adjust the positions of the remaining buildings along the linear building group to keep their initial distance ratio. The height and terrain are also considered in the method.

![Figure 4.4: Work flow chart of dynamic typification.](image)

For each selected building list, typification on each level is performed and the results are saved for dynamic visualisation. In the visualisation step, the ratio between the original length of the linear building group and the projected length is used to decide how many buildings will be removed in a group, which takes into consideration the user viewpoint and view angle. The remaining buildings in the group are adjusted in location and size. Finally, the city model is visualised using X3D.

In this paper, the removed buildings are selected based on their influence to the overall visual effect that is evaluated by ARG and NEMD method. The ARG is
employed to represent the visual features of the 3D city models (building chains) and the NEMD algorithm is used to calculate the difference between ARGs. Several typification strategies are tested with the proposed evaluation methods and the assessment results agree with the human perception. This indicated that both the evaluation method and the typification method are suitable for the typification application.

The contribution of this study is mainly the typification algorithm. The proposed method is original and it has the advantage to be very simple and efficient. It often generates appropriate results within a specific scale range. Meanwhile, the linear structure widely exists in the urban area, so the proposed method can be extensively applied for 3D city model generalisation. The proposed evaluation method is proven to be effective in the removal of buildings.

Compared with other related researches, Paper III takes the terrain features into the 3D city model typification for the first time. In dynamic visualisation, the relationship between the projected and original models is studied to determine the generalisation degree. Meanwhile, the visual similarity calculated by ARG/NEMD algorithm is used to improve the typification process. Paper II and III mainly focus on the generalisation on city area level. Next, the 3D city model generalisation on building level will be studied in Paper IV and V.

4.4 Paper IV: Shell model generation

The aim of this study is to create a simplified representation for the detailed LOD3 building models. The LOD3 buildings in CityGML denote architectural models with components such as walls, roofs, openings, etc. These objects are usually represented as cuboids (multi-surface) instead of planar surface. Therefore, at least six polygons are needed to model a simple wall, and the storage space required by these models is quite large.

If the building models in LOD3 are replaced by their exterior shells, the data volume is reduced remarkably. The visual difference between the shell model and the original one can only be detected when the building is observed very closely. And at a certain distance, no difference is visible. For example, when the 3D building models are viewed on small display (like PDA etc.) or a group of buildings have to be shown in the same view field on the monitor. Therefore, it is reasonable to represent the LOD3 buildings with their shell models.

Four steps are required to generate the shell model:

1. Extract exterior shell of wall elements;
(2) Project the selected polygons of windows and doors on the corresponding walls;
(3) Select the polygons of windows and doors whose plane are parallel to the walls where the windows and doors are sealed;
(4) Extract exterior shell of roof structure.

The proposed approach is implemented in Java and tested on a number of 3D buildings for the official site of CityGML. The experiments show that shell model can be extracted with very high efficiency. The average computing time is about 0.04 seconds for a single building, and the storage reduction of the tested buildings is over 88%.

To testify the visual similarity of the building shell model with the original one, a user survey has been conducted. Totally, 289 students at the Technische Universität München and the Nanjing Normal University participated in the user survey. The results of the user survey show that more than 90% of the participants found that the shell model can give them very similar visual impression in contrast to the original LOD3 model.

The main contribution of this paper is to prove the rationality of replacing the LOD3 model with its shell model and to verify that the visual difference is small by a user survey. The user survey containing 289 students demonstrates its visual similarity to the original LOD3 models. Besides, an efficient shell model generalisation method is proposed and implemented.

Compared with the existing LOD definition in CityGML, Paper IV introduces additional LOD (SLOD3, exterior shell of LOD3) to fulfil the gap between LOD3 and LOD2. Therefore, the proposed SLOD3 models can be used to generate the LOD1, LOD2 models used in Paper II and III. Meanwhile, the street view selection in Paper V and building facade generalisation in Paper VI can be based on the SLOD3 in Paper IV.

### 4.5 Paper V: Building selection in street view

The objective of this paper is to generalise the 3D city models for the street level visualisation. Buildings in the street view level are often in higher LODs and contain texture information. Therefore, the data volume of the 3D models required to support the applications with the street view function is huge, which is a challenge for effective visualisation. To deal with the problem, the selection operation in the model generalisation is chosen to simplify the 3D city models, considering the number of visible objects is limited in the street view.
To detect the visibility of each building, all the polygons of the 3D models are projected into 2D according to the current user viewpoint. Then visibility of each polygon is analysed based on the distance to current viewpoint. Only buildings with visible projections are selected as the generalisation results for the current viewpoint. Based on the projected 2D view, the visual salience of each building is based on the height, size, colour and relationship with other buildings. Three types of visual salience (minimum projection area, local difference and global difference) are computed in this paper.

- In the minimum projection area method, the visual salience of a building is determined by its visible area in the projected 2D plane, since the building with the relative large visible area usually can draw more attention than others.

- In the local difference method, the colour difference between the removed building and its background are used to determine the visual salience.

- In the global difference method, the visual features of all visible buildings are represented by ARG, in which the attributes are their visual salience and the relationships are the 2D distance in the projection plane. The visual importance of a building $A$ is defined as the NEMD distance between the original models and the models without $A$.

A user survey is conducted to test the efficiency of the three methods. It shows that in the street level visualisation, the local difference method is most consistent with the user perception, followed by the global method and then the minimum area method.

Because the proposed visibility analysis and visual importance calculation are quite time consuming tasks, it is not efficient to perform these processes whenever the user viewpoint changed in the dynamic visualisation. Based on the characters of the street view, the limited viewpoint (on the street) and limited view angle (along the street), an index structure is pre-generated to store the visibility and visual importance information at certain viewpoint along the street. These indexes cover the road network at certain interval. In dynamic visualisation, the corresponding index is selected for the current user viewpoint, and the generalisation results will be created based on the information of the selected index. There may be some difference in the visible models between the actual viewpoint and its index, but it does not affect the overall visual impression in most cases according to the visual similarity comparison.
Then in the dynamic visualisation stage, the $n$ visually most important buildings will be rendered in the 3D scene, and the number $n$ is decided by the applications or the network condition in the online situation.

The paper has three main contributions. First, based on the projected 2D view, it introduces the minimum area projection, the local difference and the global visual methods to compute salience values to analyse the buildings in the street view. Second, a user test is performed that indicates the local difference method is the preferable method when the salience values are used for removing buildings in the street view. The third contribution is that it has been shown that using two nearby indexes is better than only increasing the density of the indexes.

Compared with the related studies, Paper V calculates the visual salience of 3D building models based on their projected views, which conforms to human visual perception better than directly based on 3D models and reduces the computation complexity greatly. This paper mainly deals with the buildings in street level view in which the texture/colour is essential according to the salience evaluation results. It extends the studies in geometry (Paper II, III, IV) to the texture facade (Paper VI).

4.6 Paper VI: Texture generalisation

Texture is an important part of 3D city models, since it creates the realistic impression for the users. But texture files (images) are usually quite large and comprise the main storage proportion of the whole 3D model, especially for the automatically generated ones. Therefore, texture generalisation methods are used to reduce the data volume. It also increases the visual efficiency of the textured facades. In this paper, the building texture is studied.

Based on the features of the building facade (full of horizontal and vertical structures), two steps are proposed to generalise the texture on different levels: image compression and texture colouring. Image compression has been studied intensively for a long time, and this paper is not focusing on general image compression but on building texture images. These images are first rectified to make their $X$ or $Y$ axis along with the horizontal or vertical direction according to the facade geometry structure. Then, these rectified texture images are compressed in $X$ and $Y$ directions using wavelet transform and generate the multiple compressed images. In the visualisation step, the compressed image files are selected based on the facade distance from the viewpoint and the view angle. The experimental results indicate that the proposed wavelet based compression algorithm is suitable for building facades and can reduce the size of the loaded images greatly while preserving the visual impression.
For further generalisation, textured polygons in the facade are represented by coloured sub-polygons, so image files are not required anymore and the data volume is greatly reduced. In the building facade, the rectangle is the most common structure which is widely used in windows, doors, etc. Therefore, the texture image of a facade polygon is first processed by mean shift algorithm for clustering and Canny operator for edge detection. Based on the detected edges, the image is iteratively divided by the horizontal or vertical edge into homochromatic segmentations. Based on the colour difference, TextureTree, a multiple representation structure is created to store the segmentations on the different generalisation levels.

The main contribution of this paper is in two fields: dynamic texture image compression and the multiple representation structure of the textures based on segmentation. Compared with previous research, this paper utilizes the wavelet technique to compress the texture image in two directions, horizontal and vertical, and to dynamically load the suitable textures for 3D buildings not only based on their distance to the view point but also the angle of the view line, which is especially effective for the street level visualisation (see Paper V). The proposed multiple representation structure, TextureTree, can easily be combined with the block level models (Paper II, III) and building/street level models (Paper IV, V).
5. Conclusions and future research

5.1 Conclusions

This thesis focuses on the visualisation and generalisation of 3D city models in multi-scale. An online visualisation framework is proposed based on CityGML and X3D to represent and present the 3D city models respectively. The experimental results indicate that the proposed framework can realise the mainstream web browser based online 3D city model visualisation without plugins, but it is difficult to support the detailed 3D city models in large data volumes. Therefore, generalisation methods and multiple representation structures in different scales (block, building, and facade) are investigated to improve dynamic visualisation efficiency.

On the city block level, aggregation and typification methods are studied. A multiple representation data structure, CityTree, is proposed to support dynamic aggregation and visualisation. MST is used to identify the city areas with linear structure buildings and these areas will be typified with removing buildings and adjusting the remaining ones. The experimental results show that in the dynamic visualisation, the data volume and load time is reduced while the visual similarity is preserved.

On the building level, a detailed shell model representation of building is suggested as a substitute of LOD3 objects in CityGML. The storage reduction is approximately 90% for the tested models. A user survey involving 289 students shows that more than 90% of the survey participants found that the shell model gives them a visual impression similar to the original LOD3 model. Based on the shell representation, the 3D city models in street view are generalised by selection. Three different methods, minimum projection area, local difference and global difference, are employed to measure the visual salience of a building. Buildings with the smallest salience values are removed in the selection process. A user survey shows that the local difference is more suitable than the other two methods in street view building salience evaluation. Meanwhile, an index structure is proposed to store the selection results for the dynamic visualisation. This structure reduces the number of loaded 3D object dramatically, but still preserves the most salient ones so that the user can barely identify the difference from the original models.

On the facade level, the texture of buildings is first generalised by a wavelet based image compression method. Then the texture images are segmented and converted into coloured polygons for dynamic visualisation from different distances. By merging the connected polygons into a tree structure, the dynamic visualisation of the facade texture is implemented.
This thesis research makes the following contributions to address the scientific questions proposed in section 1.2.

1) *How to represent and visualise 3D city model efficiently?*

The proposed framework can implement the mainstream browser based online visualisation of 3D city models without plugins. The framework takes CityGML files as input, analyses the 3D city models in a Java platform and presents 3D scene using X3D technologies. On the client side, X3DOM is employed to deliver the 3D scenes through the browser without any plugin. This framework can represent and integrate data from different sources in CityGML, make use of existing Java analysis source codes and visualise the 3D scene by software or browsers that support X3D.

(2) *How to automatically generate 3D city models in different LODs and implement dynamic visualisation of 3D city models in multi-scale?*

Generalisation methods of 3D city models in three scales (block, building and facade) are studied. On block level, aggregation and typification methods are developed and applied, which are shown to be useful. On building level, a shell model has proven to be able to give the user the same visual impression as the LOD3 models when observed from a certain distance. On facade level, mainly texture is studied and the experimental results show that texture can be effectively generalised by image compression and segmentation into coloured polygons.

Multiple representation structures are proposed to improve the dynamic visualisation of 3D city models in multi-scale. Tree structures, CityTree and TextureTree are used to organise the building aggregation and texture division respectively. Street view index is used to store the visible buildings in a certain place. The structures are tested to be effective in dynamic visualisation and can reduce the loading data volume and increase the interaction speed quite much.

(3) *How to assess the visual quality of generalised 3D city models?*

Visual salience of building in the 3D city models is studied to calculate the similarity between the original models and the generalised ones. In our implementation, the visual features of 3D city models can be represented by ARG and the visual difference can be calculated by NEMD methods. The ARG/NEMD based evaluation method is applied in the typification and the results achieved are satisfying. In street level visualisation, the 3D city models are first projected into a 2D plane, and the visual similarity is then calculated based on the 2D view, which leads to more reasonable results since the 3D
similarity is viewpoint dependent. Besides, a user survey was carried out to verify not only the generalisation results but also the proposed evaluation methods to see if they conform to the user perceptions.

Through the studies conducted in this thesis, the specific objectives proposed in section 1.2 are achieved. First, an HTML5 based visualisation framework is presented to create online 3D scenes with X3DOM. This framework represents 3D city models in CityGML and visualise them with X3D on the web browser without specific plugins. Second, new algorithms (Dynamic aggregation, MST based typification, 3D shell model, street view index, texture compression and segmentation) are proposed to support 3D city model generalisation in multi-scale. Third, visual salience is introduced in the generalisation evaluation. Visual salience evaluation algorithms based on the 3D models and its 2D projections are proposed for different scales. User surveys are carried out to verify both the generalisation results and the automatic evaluation methods.

5.2 Future research

Future research is planned in the following directions:

First, the proposed generalisation algorithms and multiple representation structures in the thesis will be applied to the online visualisation framework introduced in Paper I. Especially, it is necessary to implement the textured building generalisation in the browser, since the ability of textured model visualisation is rather weak in X3DOM according the experiments results in Paper I.

Second, the proposed generalisation methods will be further tested using more data from different 3D city models of different urban environment. Currently, the visualisation and generalisation algorithms are implemented on the dataset from citygml.org and ViSuCity project. It is necessary to apply the proposed methods on 3D city models in larger areas and multi-scale to evaluate their performance in real applications.

Third, the evaluation algorithm will be improved to be more conformed to the human visual perception. More user surveys will be conducted to improve the accuracy of the proposed evaluation method. It is necessary to include researches from visibility engineering psychology and cognitive science to analyse the human visual perception of 3D model generalisation. For example, the eye-track technique is a potential method to apply.

Fourth, the proposed visualisation and generalisation methods will be extended to paralell computing environment. Due to the increasing volumes of the 3D city
models worldwide, it is necessary to introduce parallel computing technologies to store, process and distribute 3D city models. For example, the MapReduce program can be used since 3D city models can be processed separately. Therefore, the proposed generalisation algorithms can be re-implemented using parallel computing technologies to improve efficiency.

In addition, new technologies in web, mobile application and location based services, will be studied to create more efficient 3D online or mobile visualisation.
References


Douglass H., and Peuckert K., (1973): Algorithms for the reduction of points required to represent a digitized line or its caricature, Canadian Cartographer, 10, 1 12- 122.


Cartographic Association Commission on GeoVisualization, November 2010, pp. 496-507.


Zhang X. P., Yan T, Cui X. and Zhu D., (2009): Quantitatively Optimized Selection of Proper Scale in Land Use Cartographic Generalization, IFIP
