Automatic Generation of 3D Networks in CityGML via MUSCLE Model

Ismail Rakip Karas[1], Alias Abdul-Rahman[2], Umit Atila[3]

[1]Department of Computer Engineering, Karabuk University, Karabuk, Turkey
E-Mail: ismail.karas@karabuk.edu.tr

[2]Department of Geoinformatics, Universiti Teknologi Malaysia, Johor, Malaysia
E-Mail: alias@utm.my

[3]Directorate of Computer Center, Gazi University, Ankara, Turkey
E-Mail: umitatila@gmail.com

ABSTRACT

This paper describes the usage of MUSCLE (Multidirectional Scanning for Line Extraction) Model for automatic generation of 3D networks in CityGML format (from raster floor plans). The algorithm of the model implements line thinning and simple neighborhood methods for vectorization. The model allows user to define specific criteria which are crucial for acquiring the vectorization process. The network models are topologically structured in CityGML format. A Java based software was developed for visualizing the network model generated and converted in CityGML. After the generation process, it is possible to perform 3D network analysis based on these models for evacuation purposes. We will highlight some future works.

Keywords: 3D GIS, Network Analyses, Geo-database, Topology

1 . INTRODUCTION

When we consider 3D navigation systems we may need to solve complex topologies, 3D modelling, topological network analysis and so on. For realizing all these processes we need 3D spatial data.

Data collection used to be the major task which consumed over 60% of the available resources since geographic data were very scarce in the early days of GIS technology. In most recent GIS projects, data collection is still very time consuming and expensive task; however, it currently consumes about 15- 50% of the available resources (Longley *et al.* 2001).

Data generation is also still a problem for the researchers who work on GIS based 3D navigation systems which consumes their time more than achieving their applications or doing their researches. Pu and Zlatanova (Pu and Zlatanova 2005) have pointed out that automatically extracting geometry and logic models of a building is difficult and the nodes and links have to be created manually or half-manually.

Most of geographical information systems work with raster images data such as scanned maps and engineering drawings in CAD format. In order to manipulate, for example transform or select the lines and the other features from such raster images, these features must be extracted through a vectorization process (Nieuwenhuizen *et al.* 1994).

Line is one of the most fundamental elements in graphical information systems. Line detection is a common and essential task in many applications such as automatic navigation, military surveillance, and electronic circuits industry (Shpilman and Brailovsky 1999; Climer and Bhatia 2003).

In previous studies, there are a large number of algorithms developed for detecting lines from raster images (Miao *et al*. 2002; Lagunovsky and Ablameyko 1999; Madhvanath *et al*. 1999; Hori and Tanigawa 1993) that use traditional vectorization processes like line following-chain coding and vector reduction stages.

In this paper, the automated 3D network generation of buildings from raster plans by using MUSCLE Model (Multidirectional Scanning for Line Extraction) is described. Unlike traditional vectorization process that use line following-chain coding and vector reduction stages, the algorithm of the model generates straight lines based on line thinning and simple neighborhood techniques.

Once the network generation process completed, the data is converted into CityGML format in LOD0 (Level of Detail) by using some convertion methods. The results indicate that the MUSCLE model can be eligibly used to generate 3D topological network models of buildings.

2 . THE MUSCLE MODEL

MUSCLE Model is a conversion method which was developed to vectorize the straight lines through the raster images including township plans, maps for GIS, architectural drawings, and machine plans. Unlike traditional vectorization process, this model generates straight lines based on a line thinning algorithm, without performing line following-chain coding and vector reduction stages. By using this model, it is also possible to generate 3D Building models based on the floor plan of the building (Karas *et al*. 2008).

The model can be described in 4 main stages:
1. Threshold processing
2. Horizontal and vertical scanning of the binary image
3. Detecting wrongly vectorized lines
4. Correcting wrongly vectorized lines by using diagonal scanning

Process of MUSCLE Model is described in Figure 1.

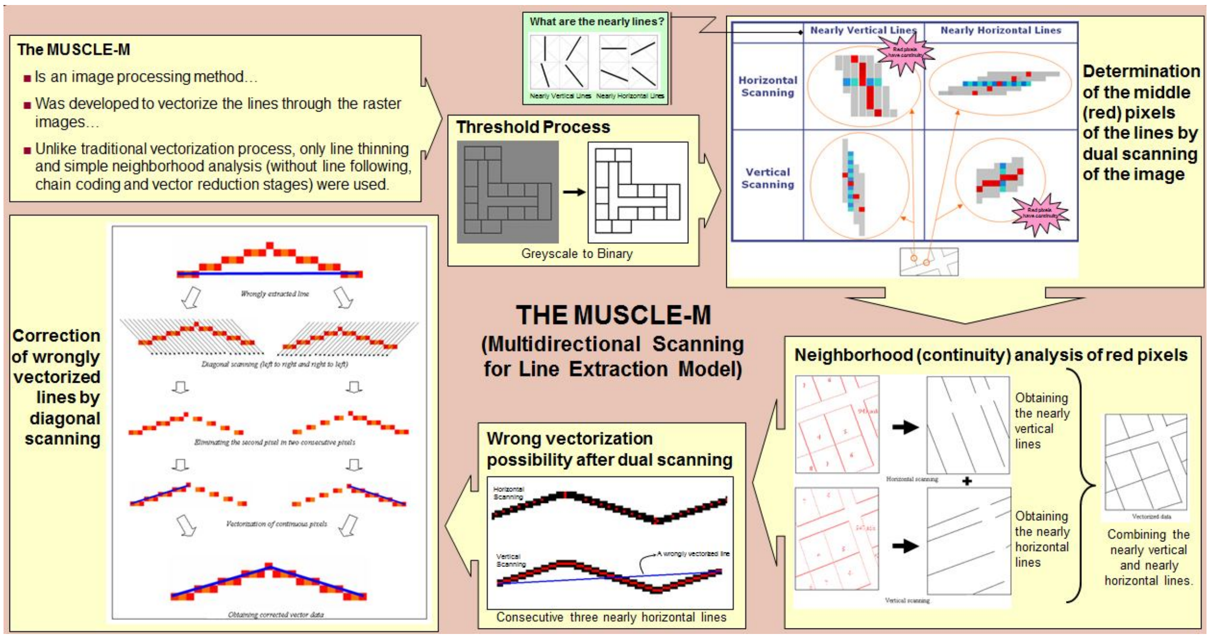
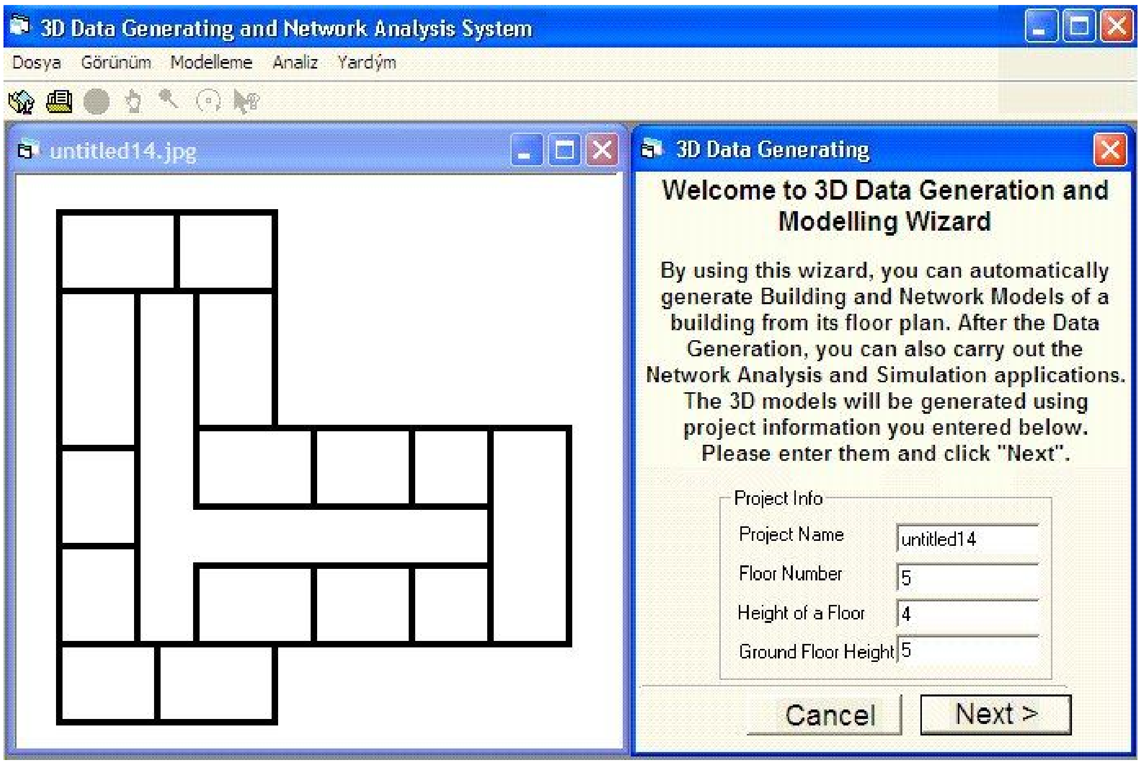

The MUSCLE-M
Is an image processing method…
Was developed to vectorize the lines through the raster images…
Unlike traditional vectorization process, only line thinning and simple neighborhood analysis (without line following, chain coding and vector reduction stages) were used.
What are the nearly lines?
Nearly Vertical Lines
Nearly Horizontal Lines
Threshold Process
Greyscale to Binary
Horizontal Scanning
Vertical Scanning
Red pixels have continuity
Determination of the middle (red) pixels of the lines by dual scanning of the image
THE MUSCLE-M
(Multidirectional Scanning for Line Extraction Model)
Correction of wrongly vectorized lines by diagonal scanning
Wrongly extracted line
Diagonal scanning (left to right and right to left)
Eliminating the second pixel in two consecutive pixels
Vectorization of continuous pixels
Obtaining corrected vector data
Wrong vectorization possibility after dual scanning
A wrongly vectorized line
Consecutive three nearly horizontal lines
Neighborhood (continuity) analysis of red pixels
Horizontal scanning
Vertical scanning
Obtaining the nearly vertical lines
Obtaining the nearly horizontal lines
Vectorized data
Combining the nearly vertical and nearly horizontal lines.

Figure 1: MUSCLE model process

3. GENERATING 3D NETWORK MODEL

By using the MUSCLE Model as briefly described in Figure 1, the 3D Building and Topological Network model of a building can be generated automatically from raster floor plans. The user interface of 3D Model Generation Software is shown in Figure 2.



Welcome to 3D Data Generation and Modelling Wizard

By using this wizard, you can automatically generate Building and Network Models of a building from its floor plan. After the Data Generation, you can also carry out the Network Analysis and Simulation applications. The 3D models will be generated using project information you entered below. Please enter them and click "Next".

Project Info

Project Name	untitled14
Floor Number	5
Height of a Floor	4
Ground Floor Height	5

Figure 2: 3D Model Generation User Interface

3.1 Generating Corridors

In NM, corridor is the main backbone in the floor plan since it connects the rooms with all the other entities in the building. Therefore, determining and modeling the corridor is very important. Once corridor was provided by the user, algorithm leaves only the corridor in the image, and then, determines the middle lines based on the method described in the previous section. After number of processes on selected middle lines, topological model and coordinates of the corridor are found as seen in Figure 3.

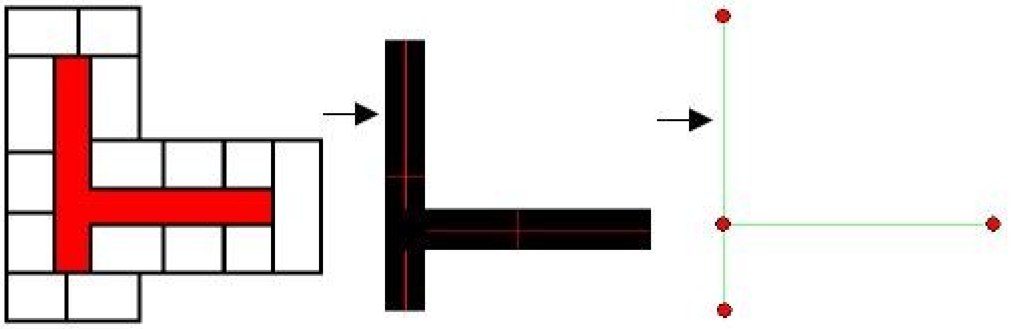


Figure 3: Generating Corridors

3.2 Generating the Rooms

In determining the rooms, corridor is excluded from the image and only the rooms are left. Then, by applying the method, middle point of the rooms are determined and defined as the nodes which represent the rooms Figure 4.

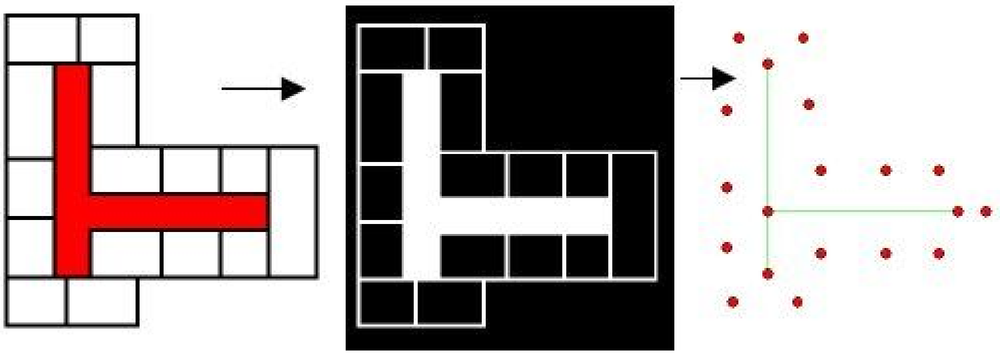


Figure 4: Generating Rooms

3.3 Integrating Corridors with Rooms

After locating the nodes that indicates corridor and rooms, user interactively points out which room nodes connect with which corridor nodes, and geometric network for 2D floor plan is generated (Figure 5a).

After stairs (or elevator) nodes are indicated by a user, the network is automatically designed by assigning different elevation values for each floor based on various data such as floor number and floor height, and then, 3D NM is generated as seen in Figure 5b.

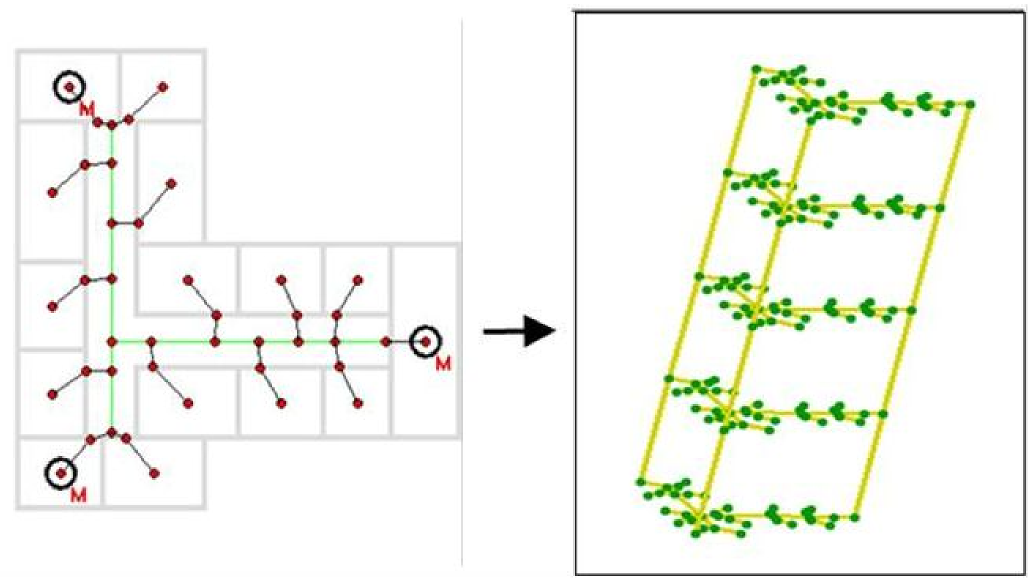


Figure 5: Generating Network Model

4 . CONVERTING NETWORK MODEL into CityGML FORMAT

4.1. CityGML

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible interna-tional standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211 (Gröger *et al*. 2008).

The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable mainte-nance of 3D city models, allowing the reuse of the same data in different application fields. CityGML not only represents the graphical appearance of city models but specifically addresses the representa-tion of the semantic and thematic properties, taxonomies and aggregations (Gröger *et al*. 2008).

CityGML includes a geometry model and a thematic model. The geometry model allows for the consistent and homogeneous definition of geometrical and topological properties of spatial objects within 3D city models. The base class of all objects is CityObject which is a subclass of the GML class Feature. All objects inherit the properties from CityObject (Gröger *et al*. 2008).

CityGML supports different Levels of Detail (LOD). LODs are required to reflect independent data collection processes with differing application requirements (Figure 6). The coarsest level LOD0 is essentially a two and a half dimensional Digital Terrain Model, over which an aerial image or a map may be draped. LOD1 is the well-known blocks model comprising prismatic buildings with flat roofs. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated surfaces. Vegetation objects may also be represented. LOD3 denotes architectural models with detailed wall and roof structures, balconies, bays and projections. High-resolution textures can be mapped onto these structures. In addition, detailed vegetation and transportation objects are components of a LOD3 model. LOD4 completes a LOD3 model by adding interior structures for 3D objects. For example, buildings are composed of rooms, interior doors, stairs, and furniture(Gröger *et al*. 2008).

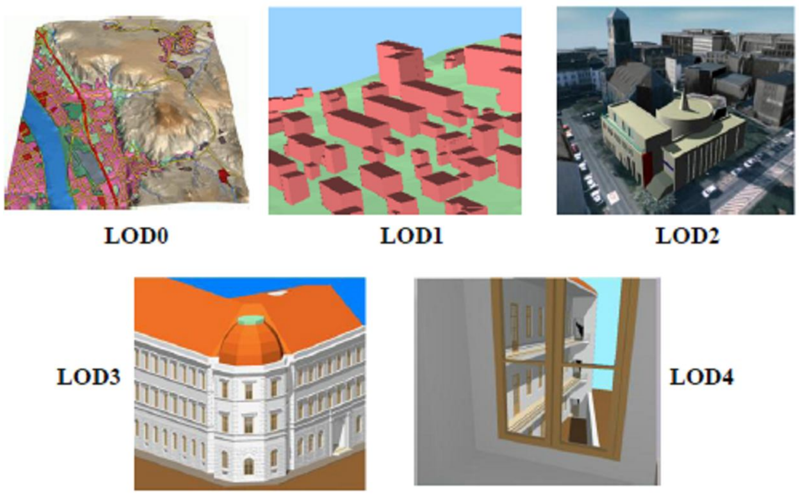


Figure 6: The five levels of detail (LOD) defined by CityGML

3D Network Model is represented using Transportation Module of CityGML. The transportation model of CityGML is a multi-functional, multi-scale model focusing on thematic and func-tional as well as on geometrical/topological aspects. Transportation features are represented as a linear network in LOD0. Starting from LOD1, all transportation features are geometrically described by 3D surfaces. The main class is transportationComplex, which represents, for example, a road, a track, a railway, or a square. Representation of a TransportationComplex for LOD0 is illustrated in Figure 7 (Gröger *et al*. 2008).

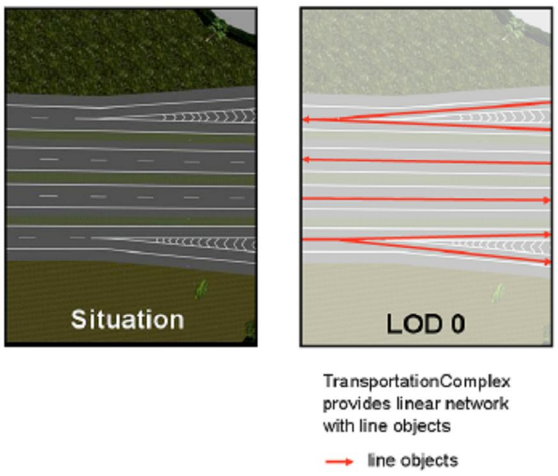

Situation
LOD 0
TransportationComplex provides linear network with line objects
line objects

Figure 7: TransportationComplex in LOD0 (Example shows part of a Motorway)

4.2. Visualization of Network Model from CityGML

Once the network generation process completed, the data is converted into CityGML format in LOD0 (Level of Detail) by using some convertion methods. The whole data generation process is described in a flow chart showed in Figure 8.

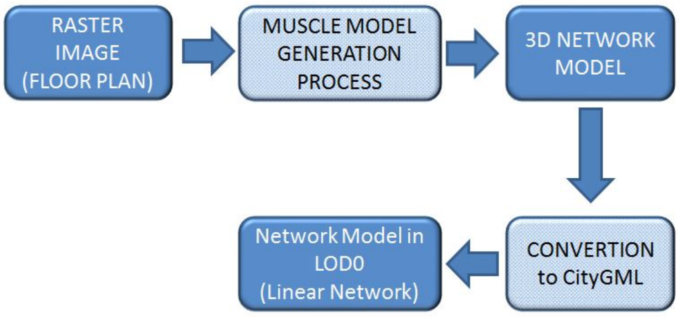
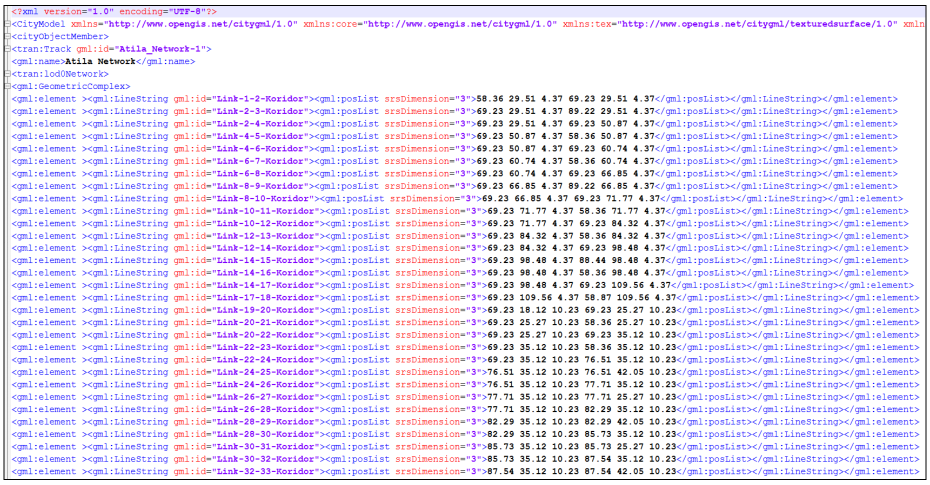

RASTER IMAGE (FLOOR PLAN)
MUSCLE MODEL GENERATION PROCESS
3D NETWORK MODEL
CONVERTION to CityGML
Network Model in LOD0 (Linear Network)

Figure 8: Data Generation Process

3D Network Model is represented as a linear network using Transportation Module of CityGML. Network model in CityGML format is shown in Figure 9.



```
<?xml version="1.0" encoding="UTF-8"?>
<CityModel xmlns="http://www.opengis.net/citygml/1.0" xmlns:core="http://www.opengis.net/citygml/1.0" xmlns:tex="http://www.opengis.net/citygml/texturedsurface/1.0" xmln
<cityObjectMember>
<tran:Track gml:id="Atila_Network-1">
<gml:name>Atila Network</gml:name>
<tran:lod0Network>
<gml:GeometricComplex>
<gml:element ><gml:LineString gml:id="Link-1-2-Koridor"><gml:posList srsDimension="3">58.36 29.51 4.37 69.23 29.51 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-2-3-Koridor"><gml:posList srsDimension="3">69.23 29.51 4.37 89.22 29.51 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-2-4-Koridor"><gml:posList srsDimension="3">69.23 29.51 4.37 69.23 50.87 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-4-5-Koridor"><gml:posList srsDimension="3">69.23 50.87 4.37 58.36 50.87 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-4-6-Koridor"><gml:posList srsDimension="3">69.23 50.87 4.37 69.23 60.74 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-6-7-Koridor"><gml:posList srsDimension="3">69.23 60.74 4.37 58.36 60.74 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-6-8-Koridor"><gml:posList srsDimension="3">69.23 60.74 4.37 69.23 66.85 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-8-9-Koridor"><gml:posList srsDimension="3">69.23 66.85 4.37 89.22 66.85 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-8-10-Koridor"><gml:posList srsDimension="3">69.23 66.85 4.37 69.23 71.77 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-10-11-Koridor"><gml:posList srsDimension="3">69.23 71.77 4.37 58.36 71.77 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-10-12-Koridor"><gml:posList srsDimension="3">69.23 71.77 4.37 69.23 84.32 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-12-13-Koridor"><gml:posList srsDimension="3">69.23 84.32 4.37 58.36 84.32 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-12-14-Koridor"><gml:posList srsDimension="3">69.23 84.32 4.37 69.23 98.48 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-14-15-Koridor"><gml:posList srsDimension="3">69.23 98.48 4.37 88.44 98.48 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-14-16-Koridor"><gml:posList srsDimension="3">69.23 98.48 4.37 58.36 98.48 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-14-17-Koridor"><gml:posList srsDimension="3">69.23 98.48 4.37 69.23 109.56 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-17-18-Koridor"><gml:posList srsDimension="3">69.23 109.56 4.37 58.87 109.56 4.37</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-19-20-Koridor"><gml:posList srsDimension="3">69.23 18.12 10.23 69.23 25.27 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-20-21-Koridor"><gml:posList srsDimension="3">69.23 25.27 10.23 58.36 25.27 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-20-22-Koridor"><gml:posList srsDimension="3">69.23 25.27 10.23 69.23 35.12 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-22-23-Koridor"><gml:posList srsDimension="3">69.23 35.12 10.23 58.36 35.12 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-22-24-Koridor"><gml:posList srsDimension="3">69.23 35.12 10.23 76.51 35.12 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-24-25-Koridor"><gml:posList srsDimension="3">76.51 35.12 10.23 76.51 42.05 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-24-26-Koridor"><gml:posList srsDimension="3">76.51 35.12 10.23 77.71 35.12 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-26-27-Koridor"><gml:posList srsDimension="3">77.71 35.12 10.23 77.71 25.27 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-26-28-Koridor"><gml:posList srsDimension="3">77.71 35.12 10.23 82.29 35.12 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-28-29-Koridor"><gml:posList srsDimension="3">82.29 35.12 10.23 82.29 42.05 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-28-30-Koridor"><gml:posList srsDimension="3">82.29 35.12 10.23 85.73 35.12 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-30-31-Koridor"><gml:posList srsDimension="3">85.73 35.12 10.23 85.73 25.27 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-30-32-Koridor"><gml:posList srsDimension="3">85.73 35.12 10.23 87.54 35.12 10.23</gml:posList></gml:LineString></gml:element>
<gml:element ><gml:LineString gml:id="Link-32-33-Koridor"><gml:posList srsDimension="3">87.54 35.12 10.23 87.54 42.05 10.23</gml:posList></gml:LineString></gml:element>
```

Figure 9: Network model represented in LOD-0 linear network in CityGML

For visualizing the network model in CityGML, a java based application was developed. The application uses *citygml4j* Java class library and API for facilitating work with the CityGML. Application uses JOGL Java bindings for OPENGL to carry out visualization (Figure 10).

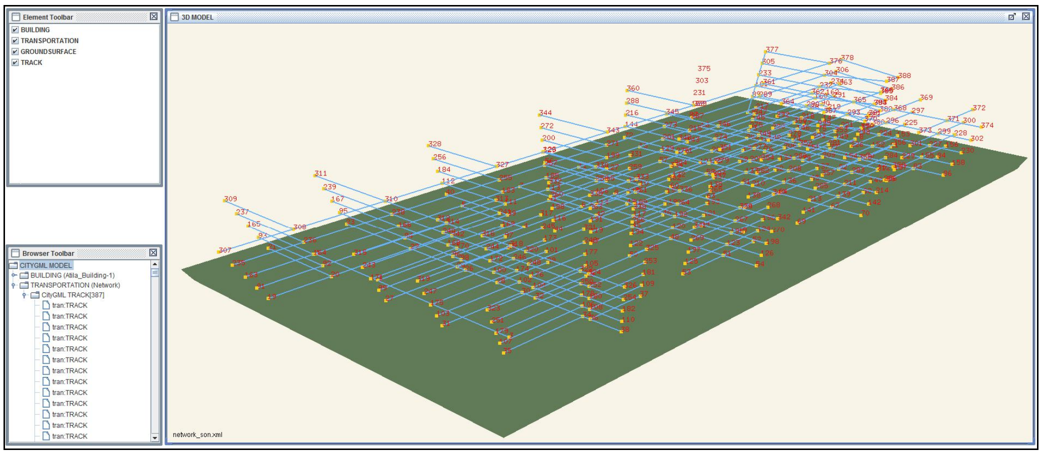


Figure 10: Visualization of Network Model from CityGML

CONCLUSIONS

In this study, usage of MUSCLE Model (Multidirectional Scanning for Line Extraction) was described for automatic generation of 3D topological networks in CityGML format from raster image format of floor plans. The algorithm of the model generates the line thinning and the simple neighborhood techniques for vectorization processes. Unlike traditional vectorization process, this model generates straight lines based on a line thinning algorithm, without performing line following-chain coding and vector reduction stages. The results indicate that the model may successfully be used to generate the network models of the buildings and convert them into CityGML. We plan to highlight some future work on 3D network analysis of generated models.

REFERENCES

1. Climer, S., Bhatia, S. K., 2003. Local Lines: A linear time line detector. *Pattern Recognition Letters*, 24, 2291–2300.
2. Gröger, G., Kolbe, T.H., Czerwinski, A., Nagel, C., 2008. OpenGIS City Geography Markup Language (CityGML) Encoding Standard, Version 1.0.0, International OGC Standard. *Open Geospatial Consortium*.
3. Hori, O., Tanigawa, S., 1993. Raster-to-vector Conversion by Line Fitting Based on Contours and Skeletons. In: *Proceedings of Int. Conf. Document Analysis and Recognition*, Tsukuba (Japan): 353-358.
4. Karas, I. R., Bayram, B, Batuk, F., Akay, A., Baz, I, 2008. Multidirectional Scanning Model, MUSCLE, to Vectorize Raster Images with Straight Lines. *Sensors*, ISSN: 14248220, Volume: 8, Issue: 4, 2673-2694.
5. Lagunovsky, D., Ablameyko, S., 1999. Straight-line-based primitive extraction in grey-scale object Recognition. *Pattern Recognition Letters*, 20(10), 1005-1014.
6. Longley, P.A., Goodchild, M.F., Maguire, D.J. and Rhind, D.W., 2001. GIS Data Collection, Geographic Information Systems and Science, Hoboken, NJ: John Wiley & Sons: 203-224.
7. Madhvanath, S., Kim, G., Govindaraju, V., 1999. Chaincode Contour Processing for Handwritten Word Recognition. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 21(9), 928-932.
8. Miao, L., Liu, X., Peng, Q., Bao, H., 2002. BRDC: binary representation of displacement code for line. *Computers & Graphics*, 26(3), 401–408.
9. Nieuwenhuizen, P.R., Kiewiet, O., Bronsvoort, W.F., 1994. An Integrated Line Tracking and Vectorization Algorithm. *Eurographics'94*, 3(3), 349-359.
10. Pu S. and Zlatanova S., 2005, Evacuation route calculation of inner buildings, In: PJM van Oosterom, S Zlatanova & EM Fendel (Eds.), Geo-information for disaster management, Springer Verlag, Heidelberg: 1143-1161.
11. Shpilman, R., Brailovsky, V., 1999. Fast and robust techniques for detecting straight line segments using local models. *Pattern Recognition Letters*, 20(9), 865-877.