EARTH OBSERVATION AND GEOMATICS ENGINEERING
website: https://eoge.ut.ac.ir

Photorealistic versus Procedural Texturing of the 3D Buildings in Virtual Smart Cities

Behnam Alizadehashrafi [1 *], Volker Coors [2], Alias Bin Abdul Rahman [3]

[1] Tabriz Islamic Art University, Faculty of Multimedia, Taleghani Square, Tabriz, Iran
[2] HFT Stuttgart, Scientific Director of the IAF, Germany
[3] University Technology Malaysia, Johor Bahru, Skudai, Malaysia

Article history:
Received: 2022-02-12, Received in revised form: 2022-03-12, Accepted: 2022-03-19

ABSTRACT

This paper discusses different methods of texturing techniques such as photorealistic and procedural texturing e.g., Dynamic Pulse Function (DPF). Based on a predefined XML schema, an unlimited number of layers can be created for windows, doors, and backgrounds. The system can use many textures for a single layer to increase the Level of Realism (LoR) in 3D virtual models. The advantage of this technique is the geospatial database of each layer in XML-schema. It can include the name of the layer, width, height, geometry, and starting point of the object in the layer with respect to the upper-left corner of the façade. The concept behind DPF, is to use logical operations to project the texture on the background image which is dynamically proportional to real geometry. The process of projection is based on two vertical and horizontal dynamic pulses starting from the upper-left corner of the background in the down and right directions respectively. The logical one/zero on the intersections of two vertical and horizontal dynamic pulses projects/does not project the texture on the background image. Orthogonal and rectified perpendicular symmetric photos of the 3D objects that are proportional to the real façade geometry were utilized for the generation of the output frame for DPF. This produces a very high quality and small data size of output image compared with the photorealistic texturing method. Complex geometries such as coconut or palm trees can be designed as a single implicit geometry and utilized in the CityGML environment via URL to deal with rendering and lagging problems of visualization. In the current work the texture of the façade was created based on preprocessed procedural DPF technique and the output image was utilized in 3D modelling as a texture.

KEYWORDS

Dynamic Pulse Function,
Semantic Modelling,
Photorealistic Texturing,
Procedural Texturing

1. Introduction

Unlike computer graphics, visualization is not the only target of GIS. Semantic information and database in XML schema for the objects on the façade are vital in GIS for analyzing the data and 3D object characteristics within a virtual urban area. Web-based applications such as Google Earth and Microsoft Virtual Earth or bing map are suitable for visualization purposes but not for geospatial data infrastructure and geodatabase. The concept is to address 3D modeling of any objects within the city along with semantic databases such as terrain, road network, building wireframes or solid models, and street furniture. Pictometry is a new highly automated 3D modeling technique developed in 2008 and can be used for lots of different applications but there are some serious texturing problems in this technique (Wang et al., 2008). The huge size of DB in 3D semantic models for the textures of geometries is another problem that should be managed properly (Aerts et al., 2006). Cropping and rectifying a small portion of aerial photos, fog, dust, deformation, and distortion of the rectified photos and pollution can reduce the quality of the textures significantly. In addition to the aforementioned problems in Pictometry,

* Corresponding Author
E-mail addresses: B.alizadehashrafi@tabriziau.ac.ir (B. Alizadehashrafi)

disturbing objects such as trees, towers, skyscrapers and shadows parked cars, pedestrians and even flying birds are also can be caused many problems while texturing the 3D models (Alizadehashrafi et al., 2010). In fact, procedural texturing is a computer-generated texture that is very similar to the realistic photos and there are many methods created by researches (Bogdahn and Coors, 2011)

2. Dynamic Pulse Function

(Coors and Zipf, 2007) defined an algorithm for creating the texture for the façade on mobile devices based on a limited number of layers using the pulse function on J2ME platform. The priority of the door layer is higher than the window layer and it can be placed on the left, middle or right side of the façade based on the parameter of its real position of the facade in a square shape and the window cannot be placed on the door. (Alizadehashrafi et al., 2009a, 2009b) implemented the same concept of the aforementioned pulse function (PF) method to generate the façade image based on an unlimited number of layers but on a fixed output frame which was not proportional to real geometry. This method caused deformation and distortion while stretching or shrinking the generated façade image on the geometry. This system could create the output image file in very high quality and small data size along with an XML schema file for different layers such as walls, doors, and windows but the quality is reduced while mapping on the geometry because of distortion. The output image file was even smaller than the textures which were used to create the image of the façade. The 3D textured model in VRML was the final product in LOD2. VRML does not support procedural texturing and pulse function so the final preprocessed procedural textures based on PF were mapped on the 3D buildings wireframe in VRML manually via coding in notepad.

Texture deformation while mapping the square output image on rectangle geometry, is the main problem of the pulse function method along with a meaningless database of parameters within the XML schema file which are not proportional to reality. To deal with this problem Dynamic Pulse Function comes in handy and can be used for semantic and geometric databases such as name of layer, its position, size and vertical and horizontal clusters and so on. As the output frame is proportional to real geometry in DPF, based on height and width of the façade it is possible to query the XML database and convert them to units such as meters automatically. In this technique, the perpendicular terrestrial photo from the façade is rectified by employing projective transformation based on the frame which is in constraint to real geometry. The rectified photos which are not suitable for texturing, can be resized in constraint proportion to real geometry before measuring process. For instance, height and width of the resized and rectified photo which is proportional to real geometry can be employed for generating the output frame for procedural texturing based on real rectified perpendicular photos from windows, doors and walls. The height and width of windows, doors, horizontal and vertical distance or cluster between windows from the upper left corner of the photo can be measured via number of pixels. The system can use these parameters and texture file names and file paths to create the façade semi-automatically. The final image file resolution is proportional to the real geometry of the façade which can prevent the image from distortion while mapping on the related geometry. To avoid leaning geometry the textures of windows, doors etc., should be cropped and rectified from perpendicular photos, so that they can be used in the program to create the whole façade. Texture enhancement should be done in advance such as removing disturbing objects, exposure setting, left-right up-down transformation, and so on. Figure 1 illustrates the process of texture generation based on dynamic pulse function and Figure 2: (d) shows the output procedural façade image nearly 17KB in a very high quality including semantic database on XML schema. The database for the layers in XML schema is a part of semantic and geometric modelling. In fact, the quality, small data size and scale along with semantic databases for each façade are the advantage of the DPF method for texturing 3D models and querying the parameters and positions of the object on the façade.

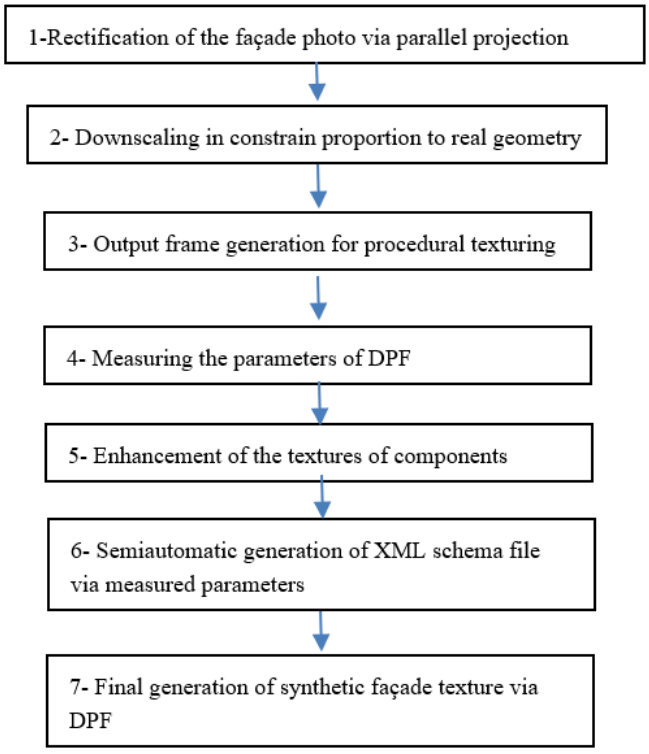

Figure 1. The flowchart for Dynamic Pulse Function

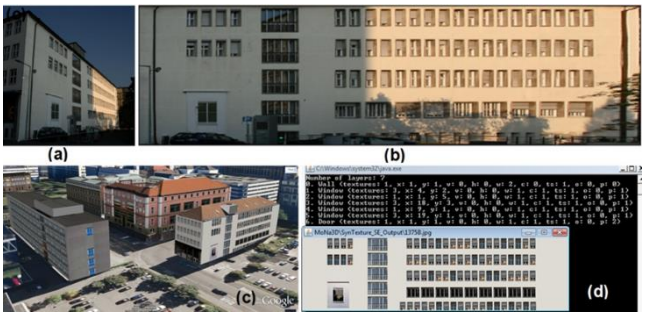

Figure 2. (a) is the original side view photo (4MB) and (b) is rectified photo (7MB). (d) is a generated photo (17KB in very high quality) and (c) is the 3D model on Google Earth. (The building of Stuttgart University of Applied Science)

Figure 3. One pixel is used for the wall texture and three random textures are used for three window layers to increase the Level of realism.

3. The Algorithm of Dynamic Pulse Function

The texture files are extracted and cropped from vertical photos of windows and doors. Reasonable differences can be seen in the data size and quality of the rectified image and the output image file. The size of the picture taken by a 10-megapixel wide-angle digital camera is (3888*2592) 2.85MB and the rectified image size is (8348 * 2854) 6.93MB. The rectified picture contains lots of leaning geometries and lack of quality in further sections from exposure point or camera position and focus point (see Figure 4). In addition, disturbing objects like pedestrians and parked cars are on the façade. In case of reducing the dimension of the rectified photo in the same size as output image frame dimensions with respect to constraint proportion, the pictures will be pixelated and still few hundred KB. On the other hand, the size of output image file is just (747 * 256) 35.4KB with a very high quality and without disturbing objects (see Figure 5).

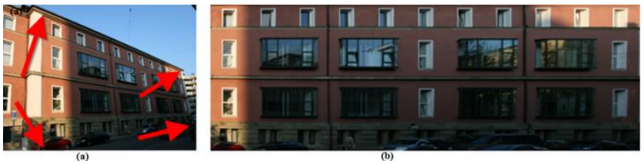

Figure 4. Four points selected on the photo (a) and four points on the corner of the output frame which is proportional to the real geometry and rectified based on projective transformation.

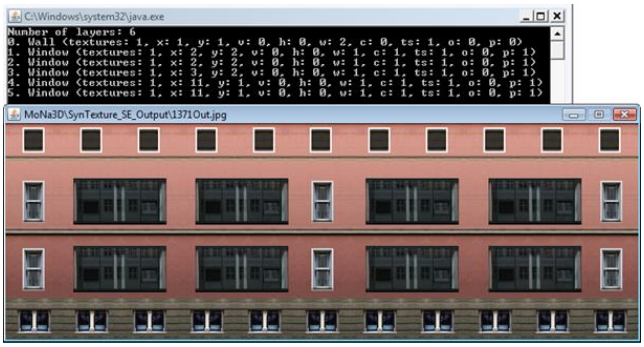

Figure 5. Generated façade image by DPF. The size of the generated output image file is less than the size of some of its textures (747 * 256) 35.4KB. (The building of Stuttgart University of Applied Science)

Based on the algorithms in the JavaScript program the group of windows which are exactly in the same rows, columns, size, shape, and vertical or horizontal distances can be assumed as one layer. The operator can make decisions on the number of layers for the windows by observing each facade and its rectified image. For instance, in Figure 6 five layers might be needed for just the window layers. In this example layers 3 and 4 can be merged manually into one layer as they are using the same texture file but based on the algorithm which is using two nested loops with the algorithm complexity of $O(n^2)$ for each layer, it is not possible to merge those layers automatically while creating the XML file. Because they do not have the same horizontal distances even if they have the same shape and size and vertical distances. First of all, the output frame of the dynamic pulse function should be generated proportional to the dimensions of the rectified façade or real geometry. After generating the frame of the output image file, the measurement process should be done based on the upper left corner of the rectified photo as a starting point. Including the wall layer, six layers should be adequate for this façade. The union of five XML schema files can be an XML schema file with six layers because of the participation and intersection of the wall layer in all of those files. The algorithm of the JavaScript program can be summarized in the following flowchart (see Figure 6).

Parameters of windows and doors with respect to the upper left corner of the rectified façade and the name and path of texture files are needed to run the program (see Figure 6).

Figure 6. Measuring all the required parameters for DPF via programming in Matlab in pixel unit.

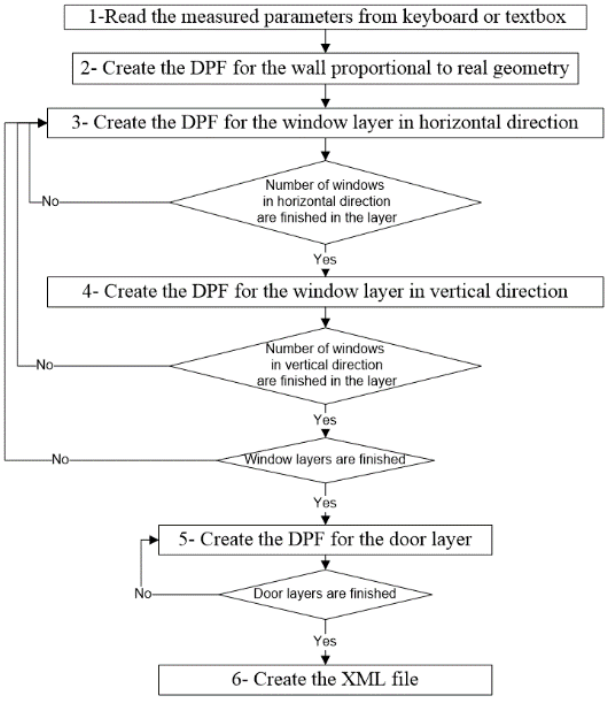

Figure 7. Flowchart for generating XML schema file based on DPF for the façade

Figure 8. Textures that are used for procedural generation of figure 5 via DPF.

The textures from left to right in figure 8 are wall texture with a resolution of 30*256 pixels and a data size of 25.9KB, and then window texture for layer one which is illustrated in figure 9 with a resolution of 251*273 pixels and a data size of 35.3KB. The third texture in figure 8 is for layer 2 in figure 9 with a resolution of 282*556 pixels and a data size of 49.4KB. The fourth texture from left to right is the window texture for layers 3 and 4 with a resolution of 1053*586 and a data size of 83.6KB and the last but not least is the fifth texture for layer 5 with a resolution of 481*330 pixels and data size of 45.4KB.

Figure 9. 5 window layers and 1 wall layer for the process of DPF

Due to lots of raised or deep geometries in some buildings, it is not possible to assume the whole façade of the building as a unique façade. To deal with this problem, the operator should be creative enough to divide the façade heuristically and generate the texture for each part separately. Dynamic Pulse Function can be used for each subsection to tackle problems such as lack of quality and leaning geometries and also using efficient storage. After cropping, rectifying and down-sampling the left part of the façade with constraint proportion, (See Figure 10: d) the dimensions of the photo have been entered into a java program and compiled in order to generate classes based on the output frame (See Figure 10: a). The parameters are measured from figure 10: d and imported to a JavaScript program using IE (see Figure 10: b) in order to create a geometric database for the façade elements in XML schema. The XML file has been used as an input for a Java program to generate the façade in a very high quality and small data size of about 8KB with a dimension of 150*512 (See Figure 10:c). The generated texture can be used for the right side of the façade by flipping the texture in the left-right direction, as they are symmetrical textures (see Figure 10: e). This can reduce the data size of 3D models significantly. The size of the whole 3D model for each building in the Kolej Perdana dormitory located in University Technology Malaysia is around 133KB.

Figure 10. The process of generating the façade via DPF and visualization in Google Earth was accepted by the Google Earth 3D community. (c) is the procedurally generated texture (15KB). (Kolej Perdana U4B, UTM, Johore Bahru, Skudai, Malaysia)

Table 1. Comparing PF and DPF.

	Dynamic Output Frame	XML DB for the Geometries
Pulse Function	No	Yes (n.a. Units)
Dynamic Pulse Function	Yes	Yes

4. Increasing the Quality of the Pictometry 3D Models by DPF

Pictometry oblique photos with roughly 45-degree angle are not good enough for 3D modeling. Low quality and distortion of the textures on the geometries can be seen while navigating closer to the 3D models that are automatically generated via pictometry algorithm with 12 to 20

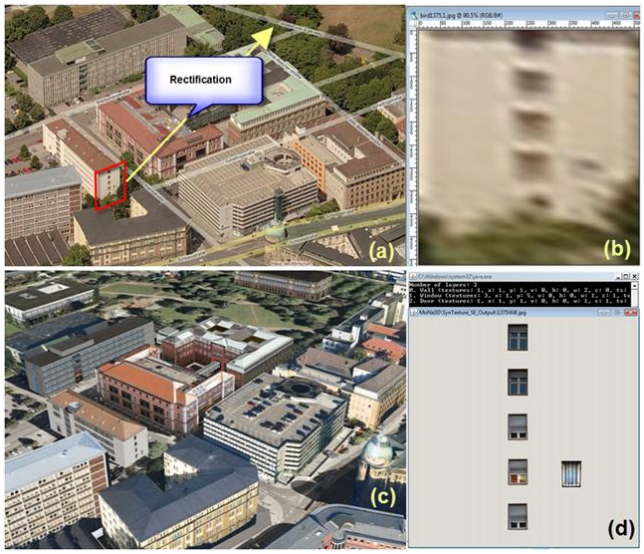

Figure 11. (b) the façade texture extracted from oblique pictometry photo from Microsoft virtual earth of bing maps(a) and regenerated based on existing database and dynamic pulse function (d) and projected on the 3D model and accepted via Google Earth 3D community(c). (The building of Stuttgart University of Applied Science)

overlapping georeferenced aerial photos from each point on the ground with the use of MicroStation and Bentley applications. In this technique, the height of the flight plane is almost 1000 meters using the wide-angle lens. In the case of having a database of textures for windows, doors, and walls from a newly constructed urban area, these textures can be used based on Dynamic Pulse Function in order to regenerate the pictometry 3D models in very high quality along with small data size. The pictometry applications can be employed for measuring the width and height of the façade, windows, doors and other necessary parameters. These parameters along with terrestrial and perpendicular photos from the small textures can be used in Dynamic Pulse Function in order to regenerate the textures for the pictometry 3D models in a very high quality and small data size. Figure 11 illustrates the process of generating the texture from oblique pictometry photos by employing DPF and an existing database of window and wall textures.

5. Features of JavaScript Program for Generating XML File

In the JavaScript program, importing and selecting window and door layers are not compulsory but wall texture should be selected along with its texture file. If there is nothing in the textbox for the window and door layers, the program can recognize them automatically and no layer will be created in the XML file for door and window. In addition, an unlimited number of random textures can be used for each window layer to increase the Level of Realism (LoR) (see Figure 12). The background layer is used and tessellating the texture is based on the dimension of the texture and frame size of DPF.

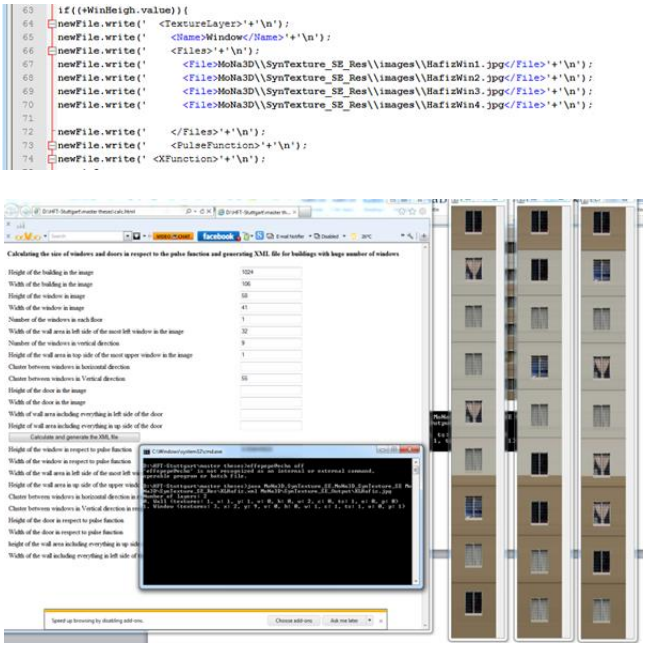

Figure 12. JavaScript program uses random textures for the window layer to increase the Level of Realism

If the height of the background texture and frame are equal, the texture tessellates horizontally to generate the

background (see Figure 13). In the case of equal width of background texture and frame, it repeats vertically to cover the background. For instance, by tiling one pixel in both horizontal and vertical directions, the wall can be created.

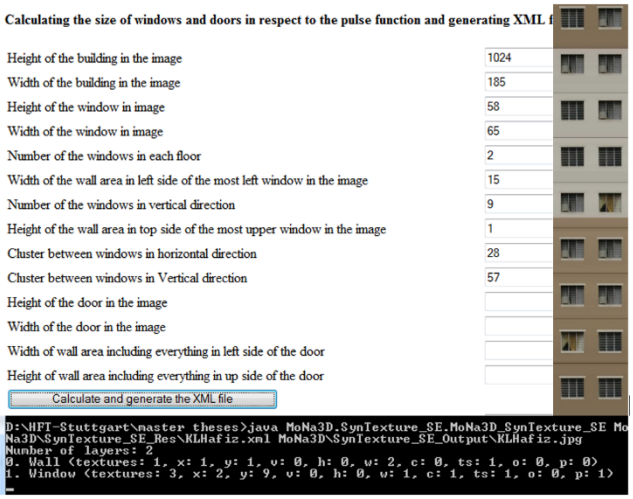

Figure 13. JavaScript program creates the XML file for the façade and Java graphics generates the façade using two layers for wall and window in this example.

6. Problems and Limitations for Dynamic Pulse Function

Some of the building façade cannot be generated by dynamic pulse function because of the strange and weird form of background or shape and geometry. The concept behind dynamic pulse function is to repeat a small sample in a very high quality to create a whole façade along with geometric information via synthetic texturing method. This can be employed for uniform facades or symmetrically uniform facades. For instance, it is very difficult to use this technique for the left and right constructions in Figure 14 because of abnormal geometries and middle image in Figure 14 (Genting Highland in Malaysia) because of unrepeatable background texture.

Figure 14. Samples that cannot be textured by DPF.

In addition to the aforementioned façade textures, DPF cannot be used for some complex geometry such as heritage, historical and religious landmarks, and their roof shape which is mostly in the domical form. The shape of the window is not an important issue in this method as a PNG file can be employed as a window texture that supports transparency and it is easier to edit than GIF and also smaller in terms of data size. In fact, the shape of the image is in rectangle form in computer graphics. The picture elements are usually indexed from the upper left corner to the lower right corner similar to the 3D matrix, but it is possible to make some parts of the image transparent as needed to create the texture for the window (see Figure 15). The final textured 3D model is illustrated in Figure 16 in a very high quality and without disturbing object and leaning geometries along with semantic definitions for window layer and wall layer in XML file.

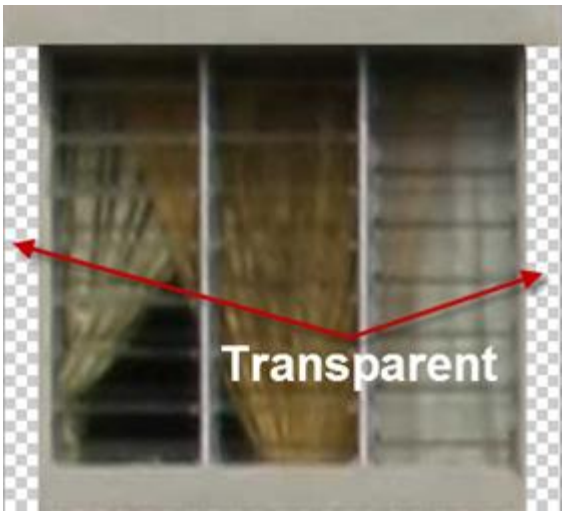

Figure 15. The image is a rectangle but the window texture is not rectangle and has transparent parts in PNG format as it supports alpha channel and transparency.

Figure 16. Textured via DPF.

7. Comparison between Photorealistic Texturing and DPF

The photo of the façade, taken by wide angle lens, should be rectified based on projective transformation. It is better not to reduce the dimension of the rectified photo as it can cause deformation and lack of quality. Disturbing objects such as air-conditioners, hanging clothes, pedestrians, parked cars, and so on, can be edited and removed manually by photo editing software such as Photoshop but it is time-consuming. In this method, it is not possible to solve the problem of leaning windows, especially for upper windows on the rectified photo taken from ground level (See Figure 17).

Figure 17. 3D model textured via photorealistic method.

8. Tiling and Tessellating

In the Putrajaya area, data collection has been done by a 10.2-megapixel Nikon D60 camera (AF-S NIKKOR 18-55mm). The image files are prohibitively large. Although it is possible to overcome this problem by reducing the resolution of the image, such down sampling can cause a lack of quality and information loss. The textures which are tessellating vertically should be up-down transformed symmetrically and those which are tessellating horizontally should be transformed in a left-right direction symmetrically. This can create a horizontal or vertical symmetric texture. In the case of tiling the texture both vertically and horizontally, the combination of the aforementioned methods can be used. The algorithm and concept are very simple and can be coded in any programming language or even photo editing applications to produce very high-quality textures along with a very small data size in KB. This technique was applied to the floor of Putra Mosque (See Figure 18). Normally while tiling the texture, the overlapping parts should match, this is the main reason that the symmetrical texture can be matched in all sides for better visualization and higher LoR. Figure 18 illustrates textures, which are created based on symmetrical left-right transformation to increase LoR.

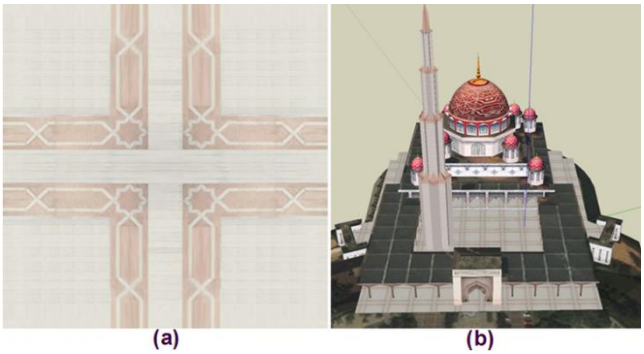

Figure 18. (a) is the created texture for the floor of Putrajaya Mosque based on rectification and symmetric transformation. Better visualization and higher LoR can be seen in the model after tessellating (b).

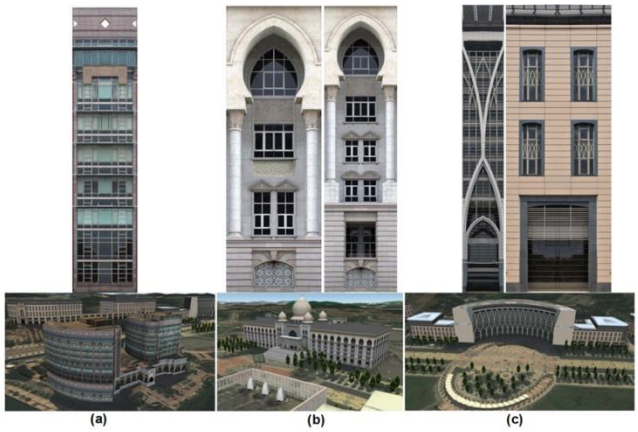

Figure 19.Producing textures via symmetrical left-right transformation (a) is Putrajaya Holding department, (b) is the Court of Justice and (c) is the Ministry of Finance.

9 Texturing Sophisticated Geometries

9.1 Texturing based on symmetric rectified photos

In some of the complex geometries such as the dome of the Putra Mosque, Sultan Mizan Mosque, Palace of Justice, and Putrajaya Prime Minister Department, perpendicular photos were taken from the dome at the same level and after rectification and projective transformation symmetrically; the image is imported to the SketchUp as an image or standalone object vertically. Half of the dome geometry is sketched on a planar surface of the image. A void or hollow circle is sketched as a footprint of the dome based on its diameter. Available tools such as “follow me” are used to generate the geometry of the domical structures. Finally, the texture projected from the image on the dome geometry and the dome is defined as a component in order to be moved to its proper position (See Figures 20 and 21).

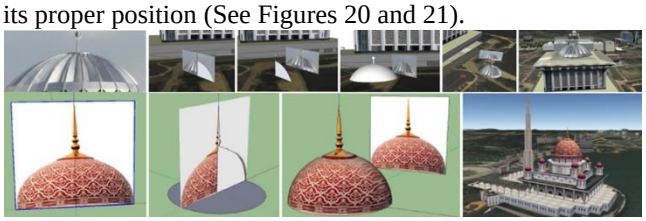

Figure 20. Texturing domical shapes of the Putra and Sultan Mizan mosques based on rectified and symmetric photos taken from the same level in Putrajaya.

9.2 Texturing based on Different Exposure Setting

From the architectural point of view, the geometries which are closer to the camera should be brighter and those which are farther should be darker. It is possible to create a very light-weighted geometry in LOD 2 for web-based applications such as Google Earth, Microsoft Virtual Earth, and mobile navigators based on this concept to reduce the data size of the geometries and textures accordingly. To increase LoR, the geometry with lots of raised columns is represented as flat rectangle geometry and the brightness of the columns is increased or for the deeper parts, the brightness decreased. This is a trick, which can be used for reducing the geometry and number of texture files and increasing LoR as well.

Figure 21. Using higher exposure to represent raised Geometry for Putrajaya Prime Minister Department.

Figure 21 shows the concept clearly. In the texture of the façade, the RGB pixel values are increased for the column in the left side of the texture to represent raised geometry and then the texture left-right transformed symmetrically and finally projected on the rectangle geometry based on the concept of the DPF as a wall or background texture.

9.3 Texturing based on Alpha Channel

PROTO is a user-defined prototype for a particular object within a virtual environment (VE). VRML or X3D as a web-based 3D interactive VE delivery system includes more than 60 standard primitives called nodes. It also has the capability of extension via user-defined PROTO nodes (Neiderer, 2005). It is possible to use the PROTO library in VRML along with the recursion concept in order to create 3D trees similar to reality which was adopted from (Alizadehashrafi et al., 2007). Unfortunately, the final result can not be converted to other formats since this kind of PROTO is not accepted by the majority of the applications.

Conversion to 3DS MAX, SKP, and CityGML are not possible as PROTO has not been defined in some of these 3D formats. Of course, this type of tree can be used as implicit geometry in a form of an external VRML file inside the CityGML viewer via URL. This type of procedural tree is not recommended as an implicit geometry within the 3D virtual smart cities.

CityGML supports implicit geometries, prototypic objects, and scene graph concepts such as vegetation models, city furniture and generic objects. Implicit geometry is a kind of prototypical geometry that can be generated once and reused an unlimited number of times based on 7 well-known parameters. The implicit geometry can be an external file in different formats such as VRML, X3D, DXF or 3DS MAX on a local or remote machine and loaded to the CityGML scene via URL. Alternatively, it can be defined by a GML3 geometry object (Gröger et al., 2008).

Portable Networks Graphics (PNG) was introduced in 1994-1998 by (Boutell et al., 1999). PNG is a suitable image format for transmitting via network and uses a lossless data compression algorithm. This format supports RGBA (Red, Green, Blue, Alpha) similar to BMP and PSD formats, and also it is patent-free. The alpha channel (8-bit) has 256 grey levels (0-255). White (255) acts as the visible area black (0) acts as an invisible or transparent area. It varies from 0% (255 grey levels) up to 100% (0 grey levels) which is completely transparent. Generating real 3D geometries for lamp posts and trees along with their textures can cause rendering overhead because of using system resources tremendously. Around 100 lamp posts were designed with their real geometries and related textures within the terrain and tested on a high-end PC. These are the main reasons that the concept of PNG as a free patented format was employed to create lamp posts, trees, fences in front of the Putrajaya Corporation Department, traffic lights, and other street furniture.

Some statues in Stuttgart in front of New Castle, in front of the church, within the garden in front of the Stuttgart University of Applied Science, annexes between the faculties along with cement fences in University Technology Malaysia (UTM) and trees based on the concept of the alpha channel were modeled (see Figure 22). The models in Stuttgart were already uploaded to Google Earth and approved via the google earth 3D community and in UTM uploaded both in Google Earth and CityServer3D for semantic databases. 3D models of statues, annexes between the departments, and fences were created based on Alpha Channel and projected on the rectangular polyhedron geometry. Some lamp posts and trees were created on two perpendicular rectangular geometries.

Figure 22. 3D models of statues, annexes between the departments, fences and lampposts based on Alpha channel.

9.4 Modelling 3D trees with uniform leaves via Alpha channel

The designed leaf can be defined as a component and repeated in different angles and sizes on the generated trunk, to complete the whole tree. The whole model should be selected and defined as a group before exporting to VRML and CityGML.

Figure 23. Illustration of designing a palm tree and converting it from SKP to VRML (BitManagment) and CityGML (LandXplorer) including 892 faces, 653edges, and 2 texture files.

SketchUp Pro has the capability of exporting 3D models in VRML, X3D, OBJ, FBX, DXF, DWG, 3DS, KMZ, and DAE file formats without any plugin. In order to export CityGML, the CityGML plugin "CityGML_Editor_17.zip (9.553,3 kB)" should be downloaded from http://www.citygml.de and extracted to the plugins folder of SketchUp 2013 and older versions.

9.5 Solutions for "Heavy" Data Size of CityGML

In order to highlight the problem of the heavy data size of CityGML in comparison to other formats such as SKP and VRML, 61 coconut trees were generated similar to the previous example of the palm tree. The data size of the whole 3D model for 61 coconut trees in SKP and KMZ is 540KB and 128KB respectively including orthophoto from Google Earth. Repeating the same component such as a leaf within a tree or repeating tree, as a component on different positions on the terrain in SketchUp, does not increase the size of the SKP and KMZ files significantly and the data size of SKP and KMZ files remain almost at the same size.

KMZ file is the zip form of KML and in case of renaming the KMZ extension to a zip extension, the content can be extracted as a "doc.kml" file along with one folder called "models" including the model in Collada world format (.dae) and its textures. Both KML and DAE files within KMZ are XML-based files but they support components and groups similar to SketchUp. The process of converting from SKP to VRML along with visualisation in BitManagement software as illustrated in Figure 24 and 25.

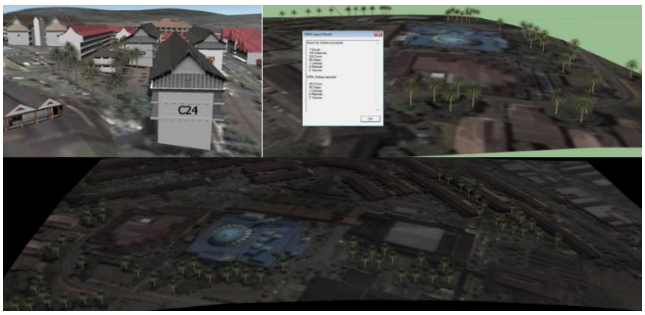

Figure 24. (UTM, Johor) The conversion process of SKP to VRML text file for 61 coconut trees along with visualization in BitManagement software.

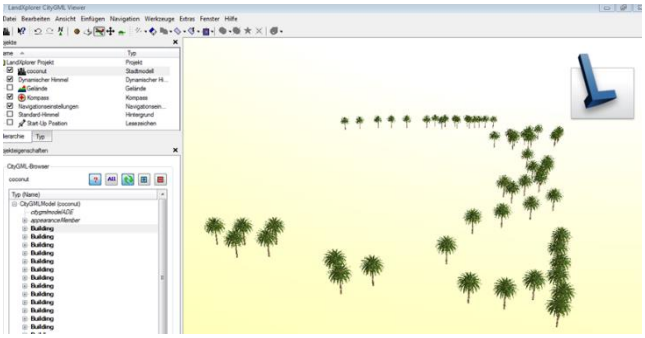

Figure 25. Loading and rendering the CityGML file with the size of 100 MB (105,519,518 bytes) in LandXplorer.

As CityGML is standardized by OGC, three different methods can be proposed for data compression and reducing the heavy data size of CityGML.

- Transformation based on seven parameters can be defined within CityGML and the components or instances can be repeated just by these parameters using classes to avoid redundancy based on simple and local coordinate systems. This is similar to an external implicit geometry prototype such as an external VRML file within the CityGML visualizer via URL on a local or remote machine (Gröger et al., 2008).

- An arbitrary coordinate system can be used for all the subclasses with the origin of (0, 0, 0) and finally the superclass as a group can be transformed to its real position including all of its subclasses. This technique can reduce the data size of XML files significantly. This is also similar to the implicit prototypical geometry within CityGML and IFC modeling building via CityObjectGroups (Gröger et al., 2008).

-Encoding and decoding systems can be used for coordinate systems on the visualizer or by the user to store the data to reduce the data size. For instance, integer type is just 2 bytes (ranges from -32768 to 32767) but in string format, it can be up to 5 bytes.

9.6 Polygonal-based Texture Mapping for 3D Building

(Tsai and Lin, 2006) used a polygon-based texturing method for curved surfaces and irregular geometries such as semi-elliptical shapes. The geometric distortion of this kind of façade tremendously depends on the exposure angle. It is challenging to project such a texture either on a rectangle image frame or on the generated virtual 3D geometry. In polygonal-based texture mapping a two-step adjustment was processed before registration. The approximation of top and bottom curved lines of the façade, is processed in the first step by employing polynomial functions in order to make the curved top and bottom lines horizontal. In the second step, the image stretched horizontally until the vertical lines on the left and right sides of the image were aligned correctly (See Figure 26).

$$y_j = y_t + \frac{y_t - y_b}{MAX(y_t - y_b)} j$$

$$x_i = x_l + \frac{x_r - x_l}{} i$$

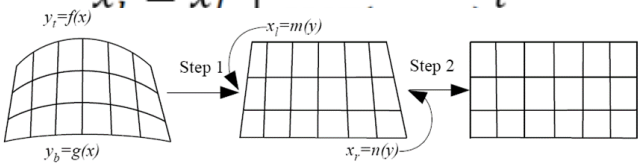

Figure 26. (Tsai and Lin, 2006) Two-step adjustment of irregular-shaped façade image.

Where are polynomial functions, representing the bottom, top, left, and right sides of the frame, and i, j are horizontal and vertical pixel indices respectively. MAX in first and second equations measure the maximum distance of top-bottom and right-left polynomials respectively. Finally, the transformed texture should be in rectangular form and projected on the rectangle and curved geometry. In fact, the texture must be projected on the curved geometry from the rectangle flat geometry and finally the rectangle geometry can be omitted from the model (See Figure 27. c).

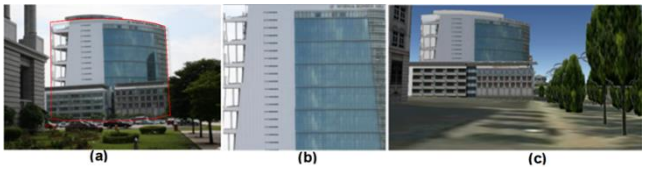

Figure 27. Texturing semi-elliptical building of MaCGDI in Putrajaya. Top and bottom curved lines are illustrated in red in the original photo (a). The generated texture (b) projected on the 3D model (c).

10 CityGML and CityServer3D

CityGML can support any kind of sophisticated geometry in varying LoDs strictly based on polygons within CityGML format via MySQL and JVM (Java Virtual Machine). Polyhedron geometry which is supported by CityGML is sufficient to model any kind of object such as street furniture and buildings in Putrajaya. CityGML format is beneficial for managing 3D city models as a multipurpose data source. The

external code list contains a set of attributes defined by OGC for the semantic modeling of objects by their type of class, function, usage, roof type, installation, material, and so on. External code lists are indexed databases and can be used for querying and analyzing for multipurpose issues. The end users may not need to know these codes but they can recognize the semantic definition of the external code list by query in the interface of CityServer3D. The operator can define different attributes for the different parts of a building based on layers or components.

Polygon and polyhedron can be converted from any 3D format to any other 3D file format. For instance, the result of converting a cylinder from 3DS MAX to VRML creates a smaller file size than converting the same cylinder from SKP to VRML. The cylinder is defined both in VRML and 3DSMAX but not in SKP so it must be represented by lots of polygons which makes a huge VRML file size. The same problem may happen while converting SKP to CityGML where a polygon is the only thing that is already defined. This is the main reason that CityGML files are of a huge size of XML files, especially for complex and sophisticated geometries.

10.1 Healing of Invisible Faces in CityGML

Mostly the models which are designed in SketchUp without any texture problems may have missing faces or solid non-textured faces in CityGML after conversion in any CityGML viewer such as KMZ viewer, LandXplorer, and CityServer3D because of non-oriented faces. An automated 3D healing model has been proposed and started by (Bogdahn and Coors, 2010) in Stuttgart University of Applied Science in order to solve this problem. The system receives the original model and based on a validation and healing plan it will generate the correct model automatically or create a correction recommendation for the operator to heal the 3D model based on the operator's supervision. Finally, the system creates the quality and healing report. The complexity of this algorithm is O (n!) and can only be computed for a small number of points in a 3D model because of heavy overhead.

The list of nodes for each polygon, facing the camera position in CityGML, must be counter clockwise so that it can be visible in any CityGML visualizer. The solution for this problem proposed by the author is to make all the faces counter clockwise before converting to CityGML in SketcuUp via Orient or Inverse options. By checking the 3D model in Monochrome mode, those faces which are in grey are not oriented. These faces can be reversed one by one or the whole model can be healed via the Orient option at once before converting to CityGML or an automatic algorithm can be utilized which is provided by (Bogdahn and Coors, 2010). Two faces on the roof of C26 in UTM are grey in a monochrome mode in Figure 28.a (not oriented) and even if they are visible in the SketchUp environment. After converting to CityGML these faces are invisible Figure 28.b). This problem was solved in Figure 28.c.

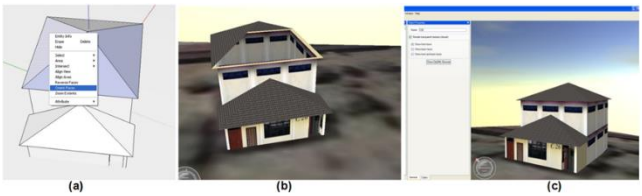

Figure 28. Solution for invisible non-oriented faces in CityGML.

10.2 Billboards and moving objects in CityGML

VRML also supports moving objects like the light emitting isometric sign of the Mercedes on the top of the tower of the main Station in Stuttgart, which is rotating in the 3D model the same as reality. The code in VRML via TimeSensor and ClockRotation nodes can be used for this purpose. Figure 30 illustrates the animation and rotation of the textured circle geometry with its transparent parts on the top of the tower of the main station in the Stuttgart virtual 3D model.

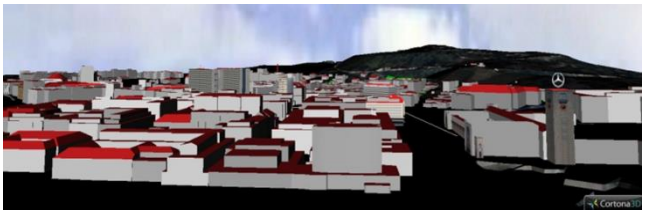

Figure 29. Rotating sign of Mercedes Benz on the tower of the main station in the 3D model of Stuttgart downtown in VRML (Alizadehashrafi, B. 2008).

10.3 External code lists for test infrastructure

In order to represent the object attributes of the city, having an enumerative range of values is necessary and the concept of dictionaries as provided by GML should be used. The values are defined in a file called CityGML_ExternalCodeLists.xml, which comes with the CityGML schema document, but is not a normative part of this schema, since it may be modified, augmented, or replaced, by users or other communities. The actual values in the file CityGML_ExternalCodeLists.xml are a suggestion of the SIG 3D (Special Interest Group 3D). In the process of designing semantic and thematic information for 3D city objects, these predefined attributes are necessary to assign to these objects which can be standardized for each area and country such as Malaysia. Attributes are employed to classify the objects and recognize them via queries or clicking and selecting the objects within the 3D virtual environment of CityServer3D. It is similar to the indexing system and a unique value relates to each attribute within a single file. These values are defined according to the attributes' name. The files are placed in a directory, which is called code list. The code lists for the predefined objects in Germany can be extracted from the CityGML schema and the data can be changed, modified, enlarged and replaced according to the requirement of the data and the user's needs. These code lists are initially equipped by SIG 3D and defined by Open Geospatial Consortium (OGC) in CityGML.

The external code list file defines attribute values and assigns a unique identifier to each value. In a CityGML instance document, an attribute value is denoted by an identifier of a value, not by the value itself. These identifiers are known terms for the operators and users. Thus, printing or typing errors are avoided and it is ensured that the same concept is denoted the same way, by the same identifier, and not by two different terms with identical meanings. This is why the use of code lists facilitates semantic and syntactic interoperability, since they define common terms within an information community. Furthermore, the dictionary concept enables more than one term to be assigned to the same dictionary entry, thus the same concept may be explained in different languages. To differentiate between the languages, code spaces are used (Gröger et al., 2008).

The 'DefinitionCollection' element is used in the dictionary concept for representing the values of an attribute, where each value is given by a Definition entry. In CityGML a definition entry is identified by the name element, which is qualified by the SIG 3D code space. The unqualified name element represents the value of the attribute and an optional description defines the value. CityGML does not use GML identifiers (gml:id) to link to attribute values, since IDs are restricted syntactically, and must be globally unique, which is not feasible for code lists (Groger et al., 2009).

All the required code lists are not available in CityGML external code lists though it covers most of the city objects. Some code lists are proposed according to the suitability for Malaysia especially in vegetation types such as species type because the vegetation type of each country can be different according to climate. In addition to the proposed code lists for the trees and plant species in Malaysia some code lists were proposed for roof types of the buildings (Munankarmi, 2011).

10.4 Semantic Database via CityGML

The most important part of CityGML is the semantic database in XML schema via predefined or user-defined tags in GML3 without requiring CityServer3D. The database can be queried based on user requirements. It is possible to extend existing code-lists to define semantic databases for geometries such as the dome of the mosque. In the OGC, the CityGML Standard Working Group decided how to deal with external code lists in future. The code lists will not be a part of the normative standard, but informative. This is an advantage as new code lists will not have to pass a formal decision of the OGC but are national responsibility. OGC is only offering a registry to find existing code lists that can be adopted by others.

10.5 Semantic database via CityServer3D

CityServer3D is a spatial information management system along with 2D and 3D view services which can provide a complete GIS application, designed by Fraunhofer IGD in Germany. The system consists of AdminTool and server components and the 3D models along with semantic databases can be queried and visualized from clients as well as servers.

The server stores data in the database, accessible via a variety of interfaces. In addition to querying the stored data in its database, it is also possible to query additional data sources. The CityServer3D uses a meta-model to handle this concept. After processing the query, the result can be transferred in various ways to another server's functional unit or clients via different interfaces in 2D or 3D formats. The interoperability of the server must be guaranteed for data exchange securely and confidently. E-commerce can be used on the system based on the server components for authentication and transaction monitoring and recording. The system consists of different layers such as interface (IL), converter (CL), functional (FL), data schema (DSL) and database management system (DBMSL). The DBMSL is based on the relational concept and oracle spatial and DSL composed of GeoDB1, and GeoBase21 that can be employed not only for geometry but also for storing thematic and spatial classification of the data sets. The controller unit (CU) controls the interaction between layers. Two tools are integrated for the clients of CityServer3D to manage the contents and access the 3D city model. Firstly, a web-based visualization system and secondly, management of the database for importing and exporting data are necessary for the client's system (Haist and Coors, 2005).

The data management component in Google Earth is file based such as KML files that are not feasible for large models and are done by Google with their 3D geo database. The size of the 3D models in Google Earth is seriously restricted and normally they are in LOD2 along with textures for each object but in CityServer3D buildings in LOD4 such as MaCGDI can be imported along with semantic data.

Direct access to the MySQL database behind the scenes of CityServer3D is possible using DDL and DML to define and manipulate the data directly. Access to the database should always be done via CityServer3D along with Java Framework. Web3D-Service interface can be used to access the database which is rather difficult and challenging. However, this will be the best interface to access the CityServer3D database from other programs and clients. Fraunhofer is currently preparing an experimental OGC interoperability exercise to test the Web3DS specification (see OGC website). To change or add external code-lists for some religious landmarks such as the dome of the mosque or the tower of the mosque for mounting the speakers on that, some knowledge of the implementation is needed which could be done only in a joint project with Fraunhofer IGD.

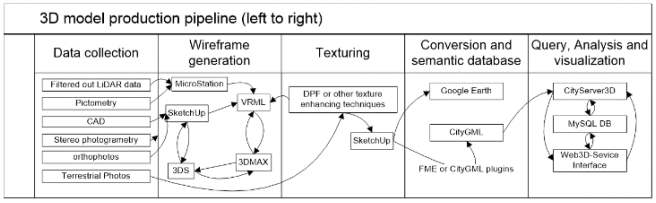

Figure 30. The 3D model production pipeline for visualization, analysis, and semantic modeling for CityServer3D.

In Putrajaya as a newly constructed and under construction area, the CAD data set was available and this is the key to having precise virtual 3D models along with high-quality textures (Alizadehashrafi and Rahman, 2011). These 3D models are much better than pictometry 3D models in European countries. Pictometry is a highly automated 3D modeling system that is fast enough to model thousands of buildings within a few hours via MicroStation but in a lower quality. CAD-based 3D modeling along with dynamic pulse function can create very high-quality 3D models in comparison to pictometry. Many recharters created 3D models form 2D CAD data within that last couple of decades (Lewis and Séquin, 1998).

The terrain and Ortho-photo of an area from Putrajaya Prime Minister Department (PPMD) to Putrajaya International Convention Centre (PICC) about 5 kilometers by one kilometer was created using Google terrain with a grid size of 76 metres. 25 buildings, 4 bridges, and all the street furniture such as spot lights, lamp posts, trees, and traffic lights were modeled and textured by aforementioned methods in this area along with semantic data in CityServer3D (see Figures 33 and 34) without using implicit geometry from an external file (Gröger et al., 2008).

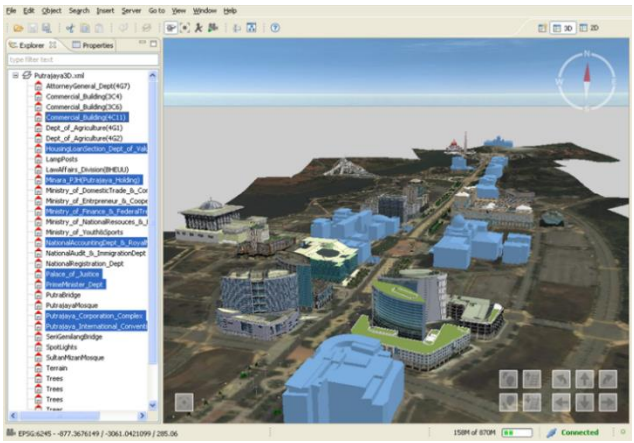

Figure 31. Visualization of Putrajaya3D including semantic DB via client system's AdminTool from CityServer3D.

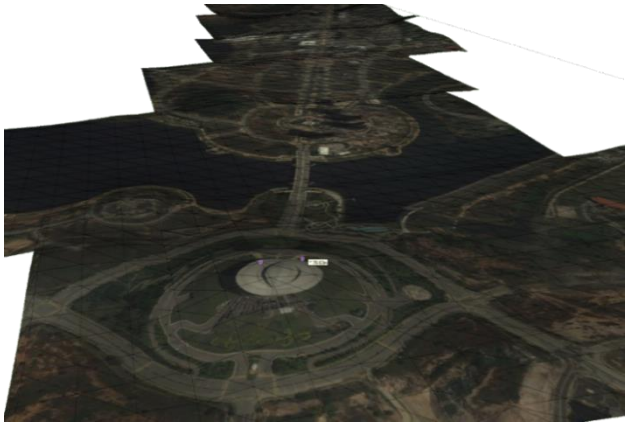

Figure 32. DTM was produced using Google Earth terrain with a grid size of 76m.

11 Summary

Texture enhancement methods such as texture left-right or up-down symmetrical transformation and higher radiometric adjustment for raised surfaces and invented Dynamic Pulse Function by the author which was awarded a gold medal in INATEX2009 and also a gold medal in MTE2010 along with the best award among 504 international exhibitors, were used for texturing. DPF has a very high quality along with a small data size in comparison to photorealistic texturing methods. These texturing methods can be done based on the situation of the façade and observation of the tessellating of the textures. DPF can be used for repetitive textures on the facade of historical landmarks and similar geometries. In the near future the DPF can be embedded in CityGML along with a detailed database for the components of the facade such as door geometry, and window geometry along with their layers.

References

Aerts, K., Maesen, K. and Rompaey, A. V. (2006). A Practical Example of Semantic Interoperability of Large-Scale Topographic Databases Using Semantic Web Technologies. 9th AGILE Conference on Geographic Information Science, Visegrád, Hungary., 8.

Alizadehashrafi, B. (2008). Synthetic texturing for 3D urban models in pedestrian navigation. Master of Science, Stuttgart University of Applied Sciences

Alizadehashrafi, B., Abid, M. and Diabate, K. (2007). CIV Project given by Prof. Volker Coors during master degree (Virtual 3D trees of Stuttgart downtown in VRML). Stuttgart

Alizadehashrafi, B., Coors, V. and Rahman, A. A. (Year). Texturing of building façades by dynamic pulse function ISG2009 Conference, 9th -10th Aug 2009 2009a.

Alizadehashrafi, B. and Rahman, A. A. (2011). CAD-Based 3D Semantic Modeling of Putrajaya. Proceedings of the Joint ISPRS Workshop on 3D City Modelling & Applications and the 6th 3D GeoInfo Conference.

Alizadehashrafi, B., Rahman, A. A. and Coors, V. (2010). Developing 3D City Models based on Dynamic Pulse Function and CityGML. MRSS 6th International Remote Sensing and GIS Conference and Exhibition.

Alizadehashrafi, B., Rahman, A. A., Coors, V. and Schulz, T. (2009b). 3D Navigation Systems based on Synthetic Texturing. Wscg, Poster Proceedings, 21-26, 56.

Bogdahn, J. and Coors, V. (2010). TOWARDS AN AUTOMATED HEALING OF 3D URBAN MODELS 5th International 3D GeoInfo Conference, Nov 3-4 , Berlin, Germany, XXXVIII-4/W15, 13-17.

Bogdahn, J. and Coors, V. (2011). MOBILE PEDESTRIAN NAVIGATION USING 3D CITY MODELS AND PROCEDURAL FAÇADE TEXTURES. 15.

Boutell, T., Lane, T. and Roelofs, G. (1999). GD.pm -- A perl5 interface to Thomas Boutell's gd library. ABSTRACT.

Coors, V. and Zipf, A. (2007). MONA 3D -- MOBILE NAVIGATION USING 3D CITY MODELS

Haist, J. and Coors, v. (2005). The w3ds-interface of cityserver3d. European Spatial Data Research (EuroSDR). Next Generation 3D City Models, 63-67.

Lewis, R. and Séquin, C. (1998). Generation of 3D building models from 2D architectural plans. Computer-Aided Design, 30, 765-779.

Munankarmi, M. (2011). 3D Semantic City Modelling of Putrajaya Area, Malaysia Master of Science, Stuttgart University of Applied Sciences.

Neiderer, A. M. (2005). Simulating Collision Avoidance by a Reactive Agent Using VRML. ARMY RESEARCH LAB ABERDEEN PROVING GROUND MD COMPUTATIONAL AND INFORMATION SCIENCES.

Tsai, F. and Lin, H. C. (2006). Polygon-based Texture Mapping for Cyber City 3D Building Models International Jornal of Geographical Information Science

Wang, Y., Schultz, S. and Giuffrida, F. (2008). Pictometry's proprietary airborne digital imaging system and its application in 3d city modelling, the international. Archives of the photogrammetry. Remote Sensing and Spatial Information Sciences, 37, 1065-1066.