USING PARAMETERS OF DYNAMIC PULSE FUNCTION FOR 3D MODELING IN LOD3 BASED ON RANDOM TEXTURES

B.Alizadehashrafi

Faculty of computer arts
b.alizadehashrafi@tabriziau.ac.ir

Commission VI, WG VI/4

KEY WORDS: Dynamic pulse function, Random textures, Automatic 3D modeling, LoD3, building façade, CityGML,

ABSTRACT:

The pulse function (PF) is a technique based on procedural preprocessing system to generate a computerized virtual photo of the façade with in a fixed size square(Alizadehashrafi et al., 2009, Musliman et al., 2010). Dynamic Pulse Function (DPF) is an enhanced version of PF which can create the final photo, proportional to real geometry. This can avoid distortion while projecting the computerized photo on the generated 3D model(Alizadehashrafi and Rahman, 2013). The challenging issue that might be handled for having 3D model in LoD3 rather than LOD2, is the final aim that have been achieved in this paper. In the technique based DPF the geometries of the windows and doors are saved in an XML file schema which does not have any connections with the 3D model in LoD2 and CityGML format. In this research the parameters of Dynamic Pulse Functions are utilized via Ruby programming language in SketchUp Trimble to generate (exact position and deepness) the windows and doors automatically in LoD3 based on the same concept of DPF. The advantage of this technique is automatic generation of huge number of similar geometries e.g. windows by utilizing parameters of DPF along with defining entities and window layers. In case of converting the SKP file to CityGML via FME software or CityGML plugins the 3D model contains the semantic database about the entities and window layers which can connect the CityGML to MySQL(Alizadehashrafi and Baig, 2014).
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 upper-left corner of the background wall in down and right directions respectively based on image coordinate system. The logical one/zero on the intersections of two vertical and horizontal dynamic pulses projects/does not project the texture on the background image. It is possible to define priority for each layer. For instance the priority of the door layer can be higher than window layer which means that window texture cannot be projected on the door layer. Orthogonal and rectified perpendicular symmetric photos of the 3D objects that are proportional to the real façade geometry must be utilized for the generation of the output frame for DPF. The DPF produces very high quality and small data size of output image files in quite smaller dimension compare with the photorealistic texturing method. The disadvantage of DPF is its preprocessing method to generate output image file rather than online processing to generate the texture within the 3D environment such as CityGML. Furthermore the result of DPF can be utilized for 3D model in LOD2 rather than LOD3. In the current work the random textures of the window layers are created based on parameters of DPF within Ruby console of SketchUp Trimble to generate the deeper geometries of the windows and their exact position on the façade automatically along with random textures to increase Level of Realism (LoR)(Scarpino, 2010).

As the output frame in DPF is proportional to real geometry (height and width of the façade) it is possible to query the XML database and convert them to units such as meter automatically. In this technique, the perpendicular terrestrial photo from the façade is rectified by employing projective transformation based on the frame which is in constrain proportion to real geometry. The rectified photos which are not suitable for texturing but necessary for measuring, can be resized in constrain proportion to real geometry before measuring process. Height and width of windows, doors, horizontal and vertical distance between windows from upper left corner of the photo dimensions of doors and windows are parameters that should be measured to run the program as a plugins in SketchUp Trimble. The system can use these parameters and texture file names and file paths to create the façade semi-automatically. To avoid leaning geometry the textures of windows, doors and etc, should be cropped and rectified from perpendicular photos, so that they can be used in the program to create the whole façade along with its geometries. Texture enhancement should be done in advance such as removing disturbing objects, exposure setting, left-right up-down transformation, and so on. In fact, the quality, small data size, scale and semantic database for each façade are the prominent advantages of this method.

1. INTRODUCTION

In fact there are many different methods of 3D modelling for different purposes. For instance in order to have high accurate 3D models from heritage and historical landmarks the laser scanning data acquisition technique and related devices has been performed with a Trimble (Mensi) GS200. This device is a Time-Of-Flight (TOF) laser scanner with a Field of View of 360° in horizontal and 60° in vertical directions. It has a low beam divergence and a typical accuracy of 1.4 mm at 50 m (Brutto and Spera, 2011). The huge data size of point clouds and lots of holes due to curvatures and missing geometries that cannot be seen from the device position are some the problems which should be solved after data collection. The holes can be filled by utilizing application such as Image Master which comes along with imaging total stations such as Top Con or Trimble. The photogrammetric data acquisition also can be used for 3D models by means of the ZScan system by Menci Software. ZScan system is a 3D scanning instrument for point cloud acquisition through a digital camera placed on a calibrated bar and a software for image matching. The images acquisition is performed translating the camera along a calibrated bar in three different positions with a known baseline (Brutto and Spera, 2011). Since the image matching systems may cause heavy overhead on the CPU (Central Processing Unit), the optimized methods such epipipolar stereo geometry lines which can do the matching process in 1D rather than 2D, can be used. Agisoft PhotoScan software also is an application which can create point cloud and low poly 3D models out of non-calibrated camera images. For instance we used fifty five images from different angles with a simple SUMSUNG SM-G530H with a focal length of 3.3 and resolution of 3264x1836 and ISO 100 and Shutter 1/30 to create the 3D model of the ICE age sid famous dull (10893 faces and 5544 vertices)see figure 1. It is possible to export different 3D formats such as COLLODA, VRML, 3DS, OBJ, KMZ and etc for different purposes such as Unity Game Engine or Google Earth. The 3DS format can be converted to CithGML via 3Ds Max and SketchUp to be used in LandXplorer or CityServer3D.

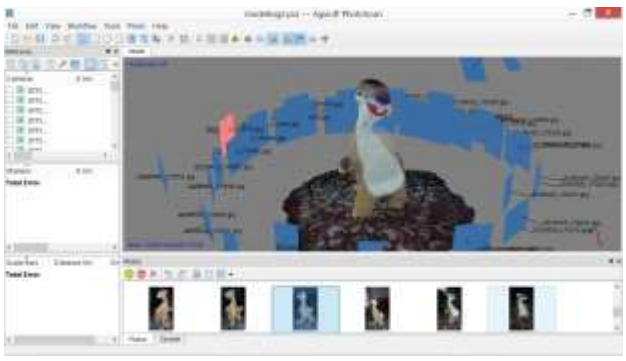

Figure 1. The 3D model of famous ICE age Sid dull were created by AgiSoft software.

In addition to the mentioned methods it is also possible to create the 3D models from precise AutoCAD data by using SketchUp Trimble (Alizadehashrafi and Rahman, 2011). Procedural modelling or procedural texturing are computerized programs which can be used to create the 3D model automatically or texture the 3D model automatically. These techniques can be used for the 3D models with repetitive structures such as Petronas Twine Tower in KL.

SketchUp was developed by startup company @Last Software of Boulder, Colorado, co-founded in 1999 by Brad Schell and Joe Esch and debuted in August 2000 as a general-purpose 3D content creation tool, and finally won a Community Choice Award at its first tradeshow in 2000. On June 8 2005, Google purchased the software and announced a free downloadable version of Google SketchUp including integrated tools for uploading content to Google Earth and to the Google 3D Warehouse. Trimble Navigation acquired SketchUp from Google on June 1, 2012 and in 2013 SketchUp 2013 was released. SketchUp has a Ruby console, an environment for codding in Ruby which supports software extensions for its import and export (From Wikipedia).

The computer cannot understand the meaning of any element of 3D model without semantic information, database and external code list. The external code lists are very useful for indexing all the elements of a 3D model for inserting data types and their values (Kolbe et al., 2005). Retrieving data from the MySQL database which is connected to CityServer3D also can be done based on queries on CityServer3D or phpMyAdmin interface. There are two different kinds of thick (CityServer3D AdminTool) and thin (Google Earth) servers. The thick servers are dealing with thinner client side (Cityserver3D Viewer) and the thin servers are dealing with thicker client side (Google Earth application). It is possible to manage our own 3D models database along with all of their elements and components within CityServer3D on the server side but not on Google Earth server which belongs to Google.

2. Ruby console in SketchUp Trimble

To The same concept of DPF can be used for the window layer along with extra parameter of deepness of the windows for each layer. In this case all the windows and doors and many other geometries on the façade can be generated automatically by codding the plugin and adding to the menu bar. The sematic database can also be included within CityGML file. By converting the 3D model to CityGML and importing to CityServer3D the main building ID with all of its subgroups along with semantic information will be available for adding entities and relevant database and external code lists. It is possible to have the same building in different LODs in the same CityServer3D for different applications. The following plugin Ruby code can represent the concept of DPF by detail. The parameters can be measured from a rectified photo of the façade by means of some markers on the façade. Parameters such as height and width of the building, number of similar windows in each floor and layer, deepness of the windows, number of floors or levels, the height and width of the window in each layer, horizontal and vertical distances between two windows in each layer, the texture file name and path (which are assumed 1.jpg , 2.jpg and 3.jpg in this code to created random textures for higher LoR) are necessary for creating the façade along with all the detailed geometries in LoD3 automatically within a second. The complexity of the program is O (n2) as it's using two nested loops to create one layer of window automatically.

```
require 'sketchup.rb'
UI.menu("PlugIns").add_item("Create Windows") {
createWindows
}
def createWindows
  hightBuilding = 64.m
  widthBuilding = 110.m
  numberOfWindows = 25
  deep = -0.2.m
```

```
levels = 8
windowWidth = 1.8.m
windowHight = 3.18.m
horizontalCluster = 2.35.m
verticalCluster =3.30.m
horizontalDistanceUpperLeftCorner = 1.25.m
verticalDistanceUpperLeftCorner=2.10.m
model = Sketchup.active_model
entities = model.entities
wall= entities.add_face [widthBuilding,0, 0], [widthBuilding, 0,hightBuilding ], [0,0,hightBuilding],[0,0,0]
z = hightBuilding - verticalDistanceUpperLeftCorner - windowHight
for step in 1..levels
        x = horizontalDistanceUpperLeftCorner
  for step in 1..numberOfWindows
    x += windowWidth
                y = 0
    pt1 = [x, y, z]
    pt2 = [x, y, z + windowHight]
                pt3 = [x - windowWidth, y, z + windowHight]
                pt4 = [x - windowWidth, y, z ]
    x += horizontalCluster
                new_WINDOW = entities.add_face pt1, pt2, pt3, pt4
                imdadr = Sketchup.find_support_file "/windows/" + (1+rand(3)).to_s + ".jpg", "Plugins"
                mats = Sketchup.active_model.materials
    nwmat = mats.add "nmat1"
    nwmat.texture = imdadr
    new_WINDOW.material = nwmat
     pts = []
     pts[0] = Geom::Point3d.new(pt1)
     pts[1] = Geom::Point3d.new(1,0,0)
                pts[2] = Geom::Point3d.new(pt2)
     pts[3] = Geom::Point3d.new(1,1,0)
     pts[4] = Geom::Point3d.new(pt3)
     pts[5] = Geom::Point3d.new(0,1,0)
                pts[6] = Geom::Point3d.new(pt4)
     pts[7] = Geom::Point3d.new(0,0,0)
                new_WINDOW = new_WINDOW.position_material(nwmat, pts, true)
                new_WINDOW.pushpull deep
        end
        z -= windowHight
        z -= verticalCluster
end
groupWINDOWS = ent.add_group new_WINDOW.all_connected
end
```

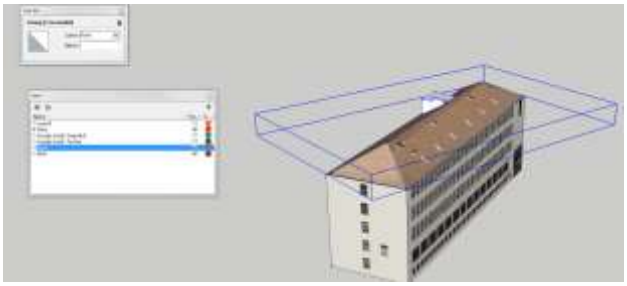

Figure 2. Represents the HFT4 building in Stuttgart in LoD2 with different layers for roof, wall, floor and windows.

In figure 2 the previous version of PDF in Java graphics and JavaScript were utilized for pre-processing texture generation in high quality and small data size without the use of Ruby Plugin. There are many different methods to convert the 3D model in SKP format to CityGML such as FME software or CityGML plugin for SketchUp. To create your CityGML file form SketchUp 3D models, it is possible to download the CityGML plugin from http://www.citygml.de/index.php/sketchup-citygml-plugin.html and extract the zip file to the plugins folder of SketchUp 2013. Unfortunately the newer versions of SketchUp such as 2014 and 2015 do not support the CityGML plugin. Figure 3 illustrates the process of installing CityGML plugin and exporting CityGML file from SKP file. Figure 4, illustrates the Stuttgart HFT building in LOD2 within CityServer3D with the data size of 15,891 bytes in CityGML format without textures with the hipped roof type external code list as 1040 which is defined for Germany ISO.

Figure 3. Installing CityGML plugins and exporting CitGML file (Putrajaya in Malaysia).

Figure 4. Illustrates the Stuttgart HFT building in LOD2 in CityServer3D (15,891 bytes in CityGML format) with the hipped roof type external code list as 1040 which is defined for Germany.

By utilizing the mentioned Ruby plugins for each side of the façade and each layer separately, it is possible to create precise 3D model of HFT4 in LoD3 automatically within less than 10 minutes. Figure 5, represents the HFT4 building in Stuttgart which is generated automatically via Ruby programming in SketchUp Trimble within different layers such as celling, column, windows, walls, roof, floor and stairs. Figure 6, is the same building in CityServer3D with the hipped roof type external code list in LoD2 and LoD3 (the size of CityGML file in LoD3 is 1,482,276 bytes). It means that the size of the CityGML file in LoD3 has a direct relationship with the number of geometries in the building. In this example 1.5 Mbyte of LoD3 is 100 times bigger than the same building in LoD2 which was 15 KB.

Figure 5. Illustrates HFT4 in Stuttgart which is generated automatically via Ruby programming in SketchUp Trimble.

Figure 6. Illustrates HFT4 in Stuttgart in CityServer3D in LoD3 and LoD2 (1,482,276 bytes in LoD3). Some layers became invisible in CityServer3D.

Figure 7. Illustrates the Tabriz Islamic Art University in different layers and entities.

Figure 8. Illustrates the process of defining some attributes and variables for parking lots of staff members as an example of semantic information on CityServer3D Admin tool.

3. Conclusions and remarks

In this research parameters of DPF along with some additional parameters for the deepness of windows and doors are utilized to generate the 3D model in LOD3 automatically. The algorithm is similar to DPF system with extra parameters such as deepness of the windows and etc. In some buildings with huge number of windows it is boring job to do the modeling in LOD3 manually which can be very time consuming and targeting to non-precise 3D model. Window layers can be defined based on needs as groups of windows for further investigation and data analysis. The building might composed of many subgroups along with semantic database in CityServer3D which can be queried based on user requests. In the near future it is possible to make use of this algorithm to generate LOD3 models from a whole urban area to investigate illegal changes that might be done by many people in residential and commercial areas. It is also possible to analyses the shadows and the sun direction during the year for installation of solar panels and targeting green university in Iran.

4. References

ALIZADEHASHRAFI, B. & BAIG, S. U. 2014. Framework for Malaysian 3D SDI in CityGML. FIG2014.

ALIZADEHASHRAFI, B. & RAHMAN, A. CAD-based 3D semantic modeling of Putrajaya. Proceedings of the Joint ISPRS Workshop on 3D city modelling & applications and the 6th 3D GeoInfo conference, 2011.

ALIZADEHASHRAFI, B. & RAHMAN, A. A. 2013. Towards Enhancing Geometry Textures of 3D City Elements. Developments in Multidimensional Spatial Data Models. Springer.

ALIZADEHASHRAFI, B., RAHMAN, A. A., COORS, V. & SCHULZ, T. 2009. 3D navigation systems based on synthetic texturing. WSCG2009, 6.

BRUTTO, M. & SPERA, M. 2011. Image-based and range-based 3D modeling of archaeological cultural heritage: the Telamon of the temple of Olympian Zeus in Agrigento (Italy). International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 38.

FROM WIKIPEDIA, T. F. E.

KOLBE, T. H., GRÖGER, G. & PLÜMER, L. 2005. CityGML: Interoperable access to 3D city models. Geo-information for disaster management. Springer.

MUSLIMAN, I. A., ALIZADEHASHRAFI, B., CHEN, T.-K. & ABDUL-RAHMAN, A. 2010. Modeling visibility through visual landmarks in 3D navigation using Geo-DBMS. Developments in 3D Geo-Information Sciences. Springer.

SCARPINO, M. 2010. Automatic SketchUp: Creating 3-D Models in Ruby.