A look at Building Model Simplification

Ziyang Zhao, Jingwen Wang

August 16, 2014

Abstract

This report presents an approach for the simplification of 3D building models which keeps the original basic structure of the buildings. The proposed solution simplifies a building model by scanning the building model in the vertical direction, analyzing the profiles of the building model, and constructing the simplified model. An experiment is then performed by implementing the proposed method and comparing to the Qslim algorithm, and the result show that our solution has better performance.

1 Introduction

3D building visualization has been seen widely used in city planning, architectural designing and certain map applications. For complex building models and large urban models, the simplification and reduction that decreases the size of models and minimizes storage requirements is an important topic. The simplified model can be further applied to mobile applications in order to reduce the network consumption. More specifically, simplifying the model details and eliminating redundant geometry can reduce the complexity for analyzing and processing the models.

Currently a lot of methods have been proposed for 3D object simplification [**?**], most of which consider a polygon surface model. These methods simplify the models by deleting/merging vertices, edges or triangle faces according to certain rules. They can achieve good performance on most polygonal meshes, especially triangle mesh models. However, some of the principles proposed for polygonal meshes do not work well on building models which have such unique features as.

In this project, we proposed an approach which deals with the simplification of building models. It takes consideration of the features of building models, and can provide a view-independent, refinement based, discrete LOD. This is achieved by finding the footprint of the building model, updating the profile of the building at each discrete height according to the footprint.

The remainder of this paper is structured as follows. Related work is surveyed in Section 2. Section 3 gives an overview of this approach and we discuss the detail of this approach in Section 4.An performance evaluation that compares the proposed approach with the Qslim algorithm [3] will be performed in Section 5. Section conclude both the advantages and limitations of this approach and discuss the future work.

2 Related Work

2.1 Simplification of Polygonal surface

Schroeder, Zarge, and Lorenson [1] developed a vertex decimation algorithm which iterates over all the vertices in the model and deletes vertices until no more vertices can be removed. During each iteration,the algorithm deletes the vertex and the surrounding polygons are retriangualted if the distance between it and the approximating plane of the surrounding vertices is below a user-defined threshold and the removal should not violate the neighborhood's local topology.

A vertex clustering algorithm proposed by Rossignac and Borrel [2] subdivides the objects' bounding volume into regular grids of boxes. All vertices are assigned different importance: vertices attached to large faces and vertices of high curvature are considered more important than vertices attached to small faces and vertices of low curvature. Next, the algorithm merges all vertices within each cell of the grid to the single most important vertex within the cell.

The Qslim [3] algorithm proceeds by iteratively merging pairs of vertices. It introduces a novel way to represent the error of vertex merge operations that is called the quadric error metric. The vertex pair with the lowest merge error will be merged and the model will be updated.

2.2 Simplification of Building Model

Since the existing techniques for polygonal surface do not work well for building models, some approaches specialized for buildings models have been further proposed.

Anders [4] proposed an algorithm for simplifying 3D urban model by projecting building models onto three orthogonal planes and obtained simplified models based on the projection. However the algorithm is only suitable for simple symmetric models.

Loya [6] presented an approach which generates an abstract representation for a building based on photographs or images of the building by identifying the key features, namely, the color and structure of the building, and then translates the features into wavesforms in both vertical and horizontal directions, and finally obtains a parametric building model.

Nan and Sharf [7] developed an algorithm based on the Gestalt principles that describe how human recognizes a group of elements as a larger aggregate entity. This algorithm clusters 2D architectural drawing into groups of elements according to Gestalts rules via graph cut. For each group which is referred to as a gestalt, the best summarization of the elements is obtained, and then a simplified shape can be used to replace the gestalt.

Xie [5] proposed an approach for simplifying both 3D single building models and building groups. This approach first adjusts the orientations of the building footprint according to the building wall, and removes structures whose volume is less than a given threshold. Next, a complex roof will be divided into several roof combinations and replaced by different simplified roof structures. For the building facades which are nearly vertical to the footprint, The algorithm will also make the building facades that are nearly vertical to the footprint to be exactly vertical. Finally, some parallel facades can be grouped and merged into one simplified facade.

3 An Overview of Building Model

3.1 Features of Building Model

Building models have some unique features which distinguish them from other 3D models. First, a lot of building models can be shaped as boxes or groups or boxes, and the details are rectangles. Second, even for some building which is asymmetrical and the footprint of the building is not rectangles, such as circle, polygons, or other more complex shapes, the walls of architectures are vertical to the ground, which makes the projection of the building in the vertical direction will fall into the footprint of the building model.

3.2 Constraints of Qslim on building model

1. Considering the features of the building model, the importance of each edge no longer depends on the quadric error. On the contrary, the importance of elements is based on the structure of the building. For example, the main walls and the roof are more important than a window.

2. Qslim is mainly focused on polygon meshes, especially triangle meshes. For building model that have curves and/or circles, it will not work well. And for polygons, edge collapse may lead to the new vertex is no longer on the plane of the original face which is a planar, which will change the basic shape of the model.

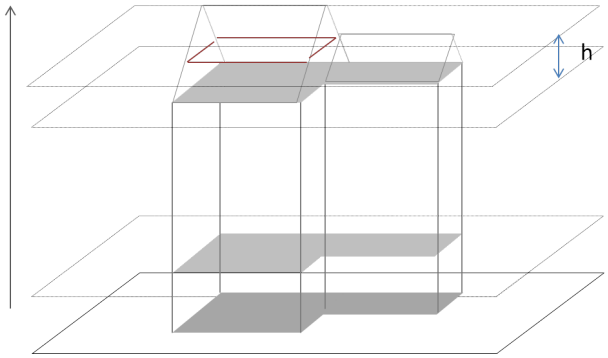
3. Similar to a lot of polygon mesh simplification methods, Qslim simplifies the model edge by edge. However, for building models, simplifying the model element by element is more suitable. For example, deleting a whole window or removing some details at once.

3.3 The framework of Our Algorithm

Our algorithm starts from finding the footprint of the building model, and then moves a plane parallel to the footprint up along the vertical direction and scans the whole building model. After moving the plane for

a certain distance, which we call the distance height h, our algorithm simplifies the part of building model between the position of the plane before and after the moving. We call this part of the model a segment, as shown in Figure 1. Our algorithm then replaces the segment with a simplified shape. For each segment, we can get two cross profile cut by the plane. The difference of the two cross profiles will be calculated, which is referred to as the error between the two profiles. If the error is below a user-defined threshold *thre*, we assume that the two cross profiles are identical and the building model does not change much in the horizontal plane. So we can replace the second cross profile with the first one and connect the outlines of two cross profiles. Otherwise, the error between these two profiles is large, we will keep the two profiles in their original shapes and connect the outline of two profiles. In this way all the small details and special structures on the facades of the segment will be removed from the building model. The pseudo code is showed in Algorithm 1.

Figure 1: proposed approach



Algorithm 1 Pseudo-code of the New Approach

Z_{min} lowest height in the mesh
Z_{max} highest height in the mesh
thre threshold of error
h distance height

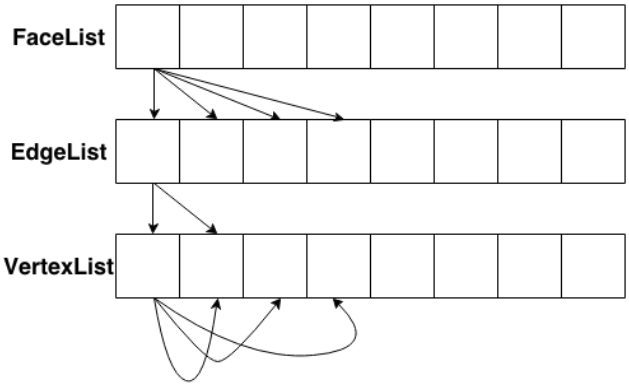
TranslateMesh()
$compVertex$, $compEdge \leftarrow$ GetProfile(Z_{min})
SimplifyProfile($compVertex$, $compEdge$)
for *height* from Z_{min} to Z_{max} by h **do**
  $currVertex$, $currEdge \leftarrow$ GetProfile(*height*)
  SimplifyProfile($currVertex$, $currEdge$)
  $error \leftarrow$ FindRelationAndError($currVertex$, $compVertex$)
  if $error > thre$ **then**
    UpdateProfile($currVertex$)
    ConnectOutline($compVertex$, $currVertex$)
    $compVertex \leftarrow currVertex$
    $compEdge \leftarrow currEdge$
  end if
end for

4 A Close Look at the Algorithm

4.1 Data Structure

Most building models are composed by vertexs, edges and faces. In our project, this information is stored in three lists, the relationship among them is showed in Figure 2. To simplify computation, all the three lists are sorted. Vertices are listed in the ascending order of the value of z coordinate, x coordinate, y coordinate. The edges are sorted by its start vertex and end vertex. All the faces are sorted by it's lowest point, i.e, the point with smallest z value, and the highest point.

Figure 2: data structure



4.2 Translate the Original Building Mesh

After translation, the footprint of the building mesh should be parallel to X-Y plane, and the altitude is positive along the Z axis. Since most available building models on the Internet follow this established custom, we did not implement this part in current version of our project.

4.3 Get the Profile at a Height

Let h denotes the distance height between two adjacent profiles which is specified by user, and Z_{min} be the height of the footprint of this building. Then after n iterations, we need to get the profile of height $Z_{curr} = Z_{min} + n * h$, which is the intersection surface of the building mesh and plane $z = Z_{curr}$.

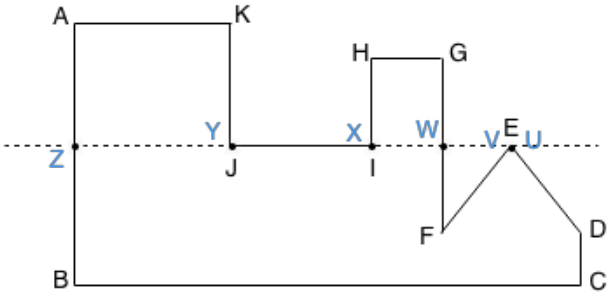
To achieve this, we first find all the faces with a lowest point not higher than Z_{curr} and a highest point not lower than Z_{curr}, the faces conform to this constrait would have intersecting lines with plane $z = Z_{curr}$. Then for each face we get, find the intersection lines with plane $z = Z_{curr}$. Suppose one of the face we get is what showed in Figure 3. We get all the edges whose start vertex is not higher than Z_{curr} and end vertex is not lower than Z_{curr}. In this case, they are AB, ED, EF, GF, HI, IJ and KJ. Among these edges, IJ is a special one cause its start vertex and end vertex has the same z coordinate, this means IJ itself is a part of the intersection surface, so add this edge into the $currEdges$, which is the set of edges of the intersection profile of the current height, add the start vertex and end vertex into $currVertexs$, which is the set of vertex of the current intersection profile. For other edges, we compute the intersection point–U, V, W, X, Y, Z–add it into $currVertexs$. Next step is sort them and connect them two by two, and get the intersection line WX and YZ(Since U and V has the same coordinates, we just ignore UV), add them to $currEdges$.

In this way, all the information about the intersection surface is stored in $currVertexs$ and $currEdges$ after we analysised all the faces.

4.4 Simplify a Section

This is not implemented in current version of project. While some existing polygon simplification algorithm can be used.

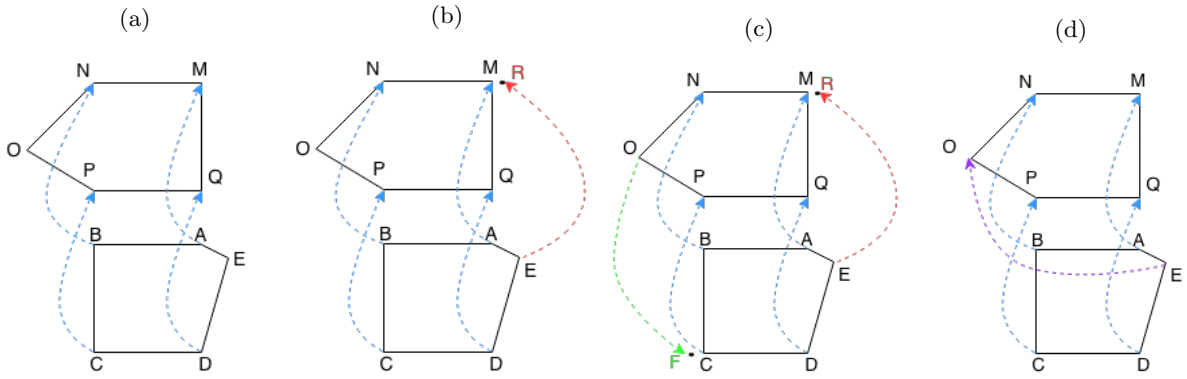
Figure 3: Get the Intersection Lines and from a Face



4.5 Find the Relationship Between Two Sections

When we get the intersection surface of a height, let it be $currProfile$, we need to compare this surface with another surface of a lower height, denoted by $compProfile$, find the relationship and compute the difference between these two profiles.

Figure 4: Get the Relationship Between Two Profiles



Suppose the $compareProfile$ we get is $ABCDE$, and the $currProfile$ is $MNOPQ$, as showed in Figure 4. Our solution is that, for each vertex in $compProfile$, find the closest point in $currProfile$(this distance is a X-Y plane distance), set them as a pair. In this case, $A-M$, $B-N$, $C-P$ and $D-Q$ are associated vertexs. The closest vertex of E is M, while M is already associated with A, so we create a vertex R with the same coordiate as M, and set E and M as a pair, as showed in Figure 4(b).

Then for every vertex $currProfile$ which is not in a pair, O in this case, find its closest point in $compProfile$, which is C, create a new vertex F at the same position as C and associate O and F.

In this way, we associate the vertexs in $currProfile$ and $compProfile$ together, and get six pairs–AM, BN, FO, CP, DQ and ER, showed in Figure 4(c). The error of each pair is the distance on the X-Y plane.

There are two other approach to match the points in $currProfile$ and $compProfile$, one is to use distance as the preference and find the best match between the two set of vertexs; the other one is minimize the total distance of all the vertex pairs. However, both these two methods may give unexpected result when dealing with special case. Take this case as an example, both the best match algorithm and the minimize total distance approach may result in Figure 4(d), where O and E is associated, and this is obviously a wrong match.

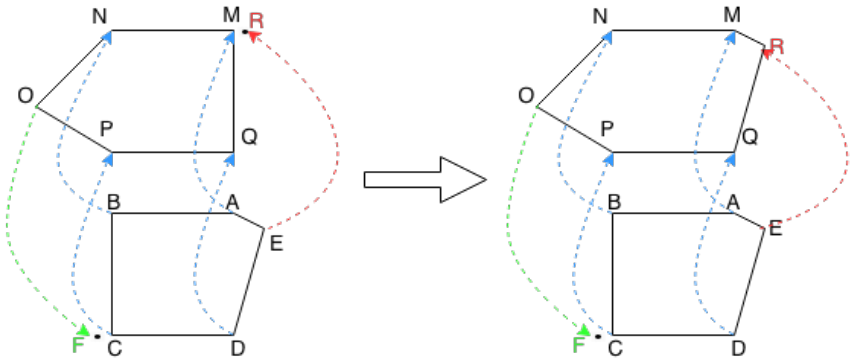
4.6 Update the Vertexs

If the total error we get from the previous step is bigger than the error threshold $thre$, a little change should be make to the $currProfile$. If a vertex has an error smaller than another error threshold $thre'$, for example,

R in Figure 5, we the adjust the R' x and y coordinates to the same as its associated point E. Otherwise, such as $F - O$, there error is bigger than $thre'$, then O keeps unchanged.

The reason why this approach adjust the upper profile according to the lower profile instead of vice versa is that most architectures have a larger foundation, whose profile polygons are more regular.

Figure 5: Update Vertexs



4.7 Connect the Outlines

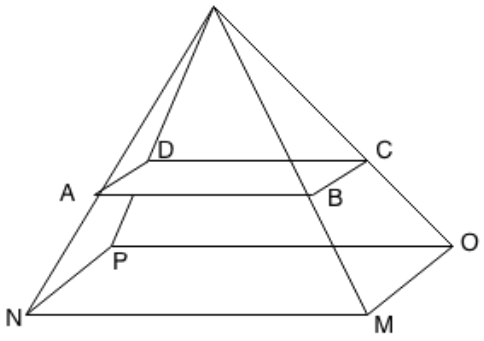
If the total error between $compProfile$ and $currProfile$ is big enough, we need to consider how one profile changes to another to build the outline surface between them. Basically, there are two kinds of transition between two shapes. One is smoothly transition, as showed in Figure 6(a). Another transition is sudden change, the shape of the lower profile changes to the shape of the upper profile all of a sudden at some height between the two profiles, as showed in Figure. 6(b). The problem is how to decide which transition to it is, and if is the second one, how to find the height where the change happens.

An ideal solution is to get the profile at the middle height of the upper profile and lower profile. If the corresponding point is closer the to the point at the upper profile in x-y plane, then we could roughly say this is a sudden change and happens at the height of the lower profile, or vice versa. If the middle point is close to neither of the points in upper and lower profile, we consider this as a smooth transition.

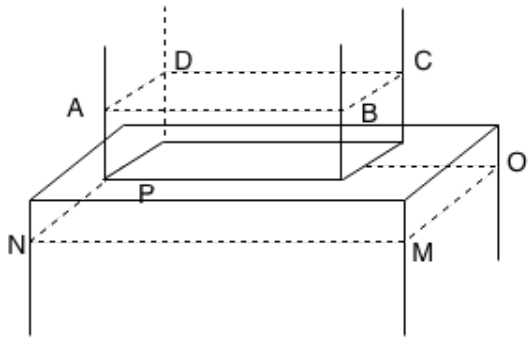
However, in the current version of our project, we only consider of the transitions as sudden change happen at the upper profile height.

Figure 6: Transition Between Two Profiles

(a) Smooth Transition



(b) Sudden Transition



5 Result and Analysis

We use the mesh of Washington National Cathedral as the original mesh, showed in Figure 7, this mesh has 714 faces, 1714 edges and 1025 vertexs. We compare the simplification result of our algorithm with Qslim, the result of Qsilm is showed in Figure 8, and the new approach in Figure 9. We can see that after reducing about four handred edges, comparing Figure 8(a) to Figure 9(a), the result of Qslim looks a little mess, while the result of our algorithm, though not perfect, looks ok. However as the increasing of the number of the edges reduced, the advantage of our algorithm becomes more and more obvious compared to Qslim. When one thousand edges have been reduced, compare Figure 8(c) to Figure 9(c), the result of Qslim is already a disaster, while our algorithm still looks like Washington National Cathedral and keeps its symmetry.

Figure 7: Washington National Cathedral Original Mesh(F=714, E=1714, V=1025)

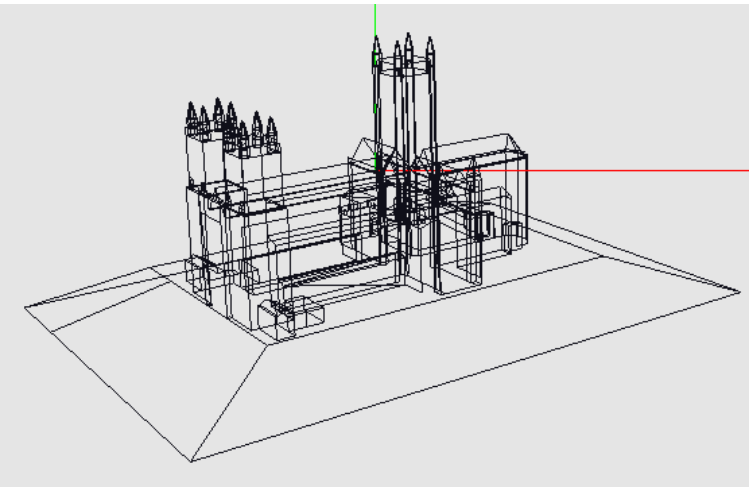
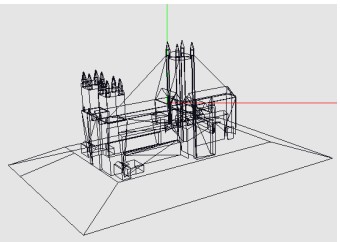
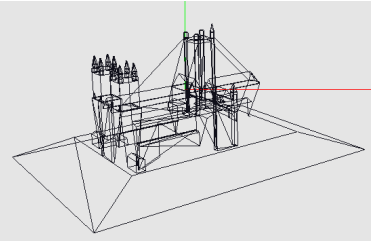


Figure 8: Simplification Results of Qslim

(a) F=573; E=1335; V=722



(b) F=439; E=1052; V=522



(c) F=272; E=690; V=322

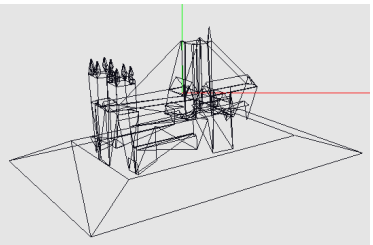
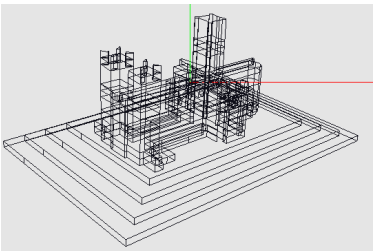
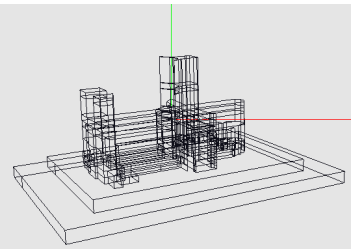


Figure 9: Simplification Results of the New Approach

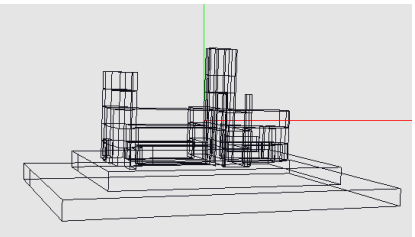
(a) F=578; E=1336; V=772



(b) F=480; E=1098; V=578



(c) F=278; E=665; V=364



6 Conclusion and Future Work

6.1 Conclusion

As we can see from the previous comparsion, our algorithm performs well when simplifing building models. The result building mesh still looks like the original building and keeps its symmetry. As the increasing of the number of the edges reduced, the advantage of our algorithm becomes more and more obvious compared to Qslim.

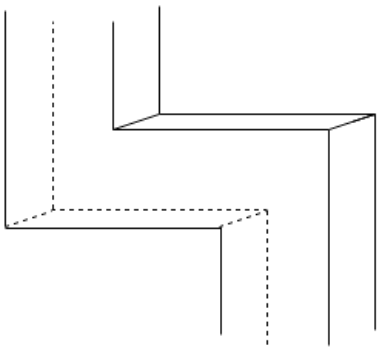
However, this algorithm has its shortcomings. First of all, to simplify the building mesh, we need user' input—height, error threshold, the user need to know the scale of the model in advance in order to give the input. While even if the user has access to these information, the result still can not be predicted.

Besides, when we choose a component to reduce, we only take the x-y position into consideration, which would cause problems. For example, if there is a small window and a big door in the building mesh, intuitively, we should simplify the window first. While if the window is depth into the wall, this algorithm may reduce the door first.

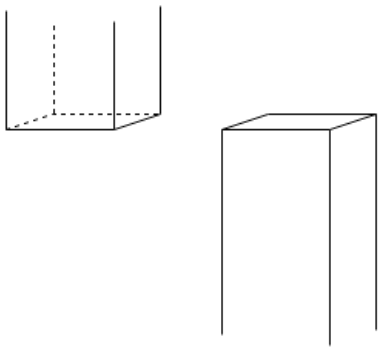
There is also some special cases that this approach would give a wrong result. For example, if the original mesh is like Figure 10(a), then after simplification, it may looks like Figure 10(b).

Figure 10: Special Case

(a) Before Simplification



(b) After Simplification



6.2 Future Work

This algorithm can be improved in the following aspects. The model translation and section simplification part can be added in the follow-up work. Adaptive height and error threshold selection can be employed so that the algorithm will no longer depends on user calculating and choosing the appropriate parameters. Then we can take texture into consideration when computing the error between two profiles. Furthermore, this algorithm is view-independent, we can derive a view-dependent algorithm in future. Besides, a hierarchical simplification can also be implemented, which means we can first divide the building model into several parts and apply this algorithm to each parts.

References

[1] W.SchroeDer, J.Zarge, and W.Lorenson, "Decimation of Triangle Meshes," *ComputerGraphics*(Proc. Siggraph 92), vol.26, ACM Press, 1992, pp.65-70.

[2] WJ.Rossignac and P.Borrel, "Multi-Resolution 3D Approximations for Rendering Complex Scences," *Geometric Modeling in Computer Graphics*, Springer-Verlag, 1993, pp.455-465.

[3] M.Garland and P.Heckbert, "Simplification Using Quadric Error Metrics," *Computer Graphics*(Proc.Siggraph 97), vol.31, ACM Press, 1997, pp.209-216.

[4] Anders, K. Heinrich. "Level of detail generation of 3D building groups by aggregation and typification." In*International Cartographic Conference*, vol. 2. 2005.

[5] J. Xie, L. Zhang, J. Li, H.Wang, and L. Yang. "Automatic simplification and visualization of 3D urban building models." *International Journal of Applied Earth Observation and Geoinformation* ,2012, pp.222-231.

[6] A.Loya, N.Adabala, A. Das, and P. Mishra. "A practical approach to image-guided building facade abstraction." *Computer Graphics International 2008*, 2008.

[7] L.Nan, A. Sharf, K.Xie,T.Wong and O.Deussen, "Conjoining gestalt rules for abstraction of architectual drawings", *ACM*,vol.30,2011