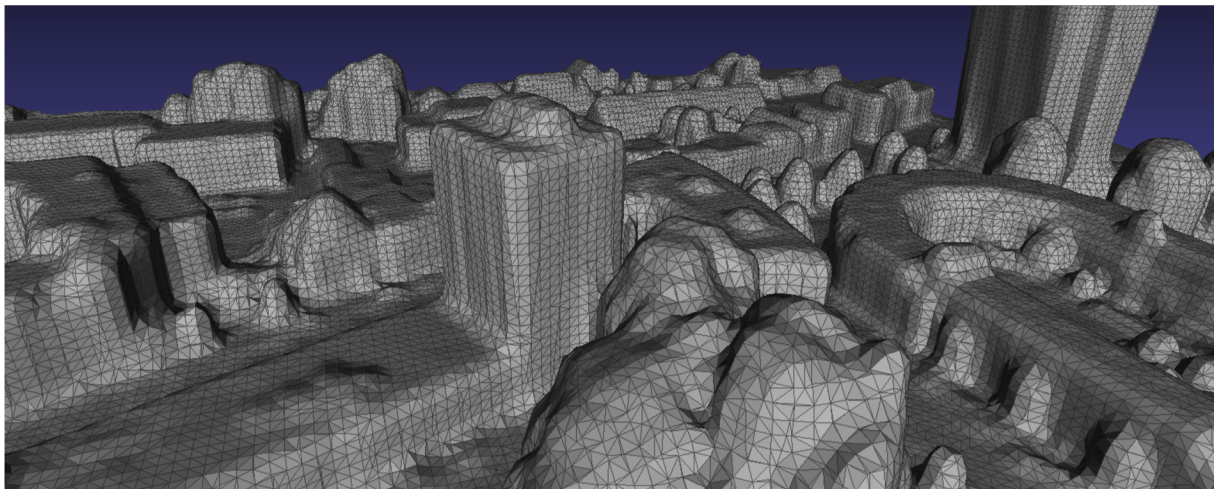
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Simplification of large-scale triangle meshes using geometric and radiometric information



Interdisciplinary Project 2018HS, January 2019
Institute of Geodesy and Photogrammetry, ETH Zürich

Author: Jingtong Li
Supervisors: Dr. Mathias Rothermel
Prof. Dr. Konrad Schindler

Abstract

We present a mesh simplification method that could be applied to large-scale triangle meshes. The main achievement of this method is preserving local features of interest during decimating irrelevant triangles in other regions. In this method, both geometric and radiometric information will be applied. As geometric information we exploit intensity images that capture the extent of a mesh. Thus local features can be extracted on these images by using image processing methods. These features can be then reprojected to the mesh given known camera calibration. As radiometric information the semantic segmentation of these 2D images are available in form of pixel-wise likelihoods for all possible classes. Finally both local geometric features and semantic segmentation will be embedded into an existing mesh simplification framework as constraints. We evaluate our methods on a real worlds dataset and the results show a promising applicability to the problem of adapted mesh simplification.

Contents

1 Introduction

This report is written within the master program Geomatics at ETH Zürich as Interdisciplinary Project Work during the fall semester 2018. This project involves both the chair of Photogrammetry and Remote Sensing at D-BAUG and the chair of Computer Vision and Geometry Group at D-INFK. In this section, background information about this project will be given regarding motivation, main tasks and structure of this report.

1.1 Motivation

3D scene reconstruction from multiple images is one of the main problems in both photogrammetry and computer vision [1]. One standard output of photogrammetric reconstruction is a digital surface model in form of a triangle mesh. Since most of reconstructed models are built upon dense matching method, the resulting meshes often have a very high level of detail. This often causes the problem that a large number of triangles are unnecessarily created that carry very little information. Thus mesh simplification has become an essential step in mesh processing for the purpose of complexity reduction.

Uniform mesh simplification can, however, be insufficient or unsatisfactory in practice. In many cases, some crucial parts of a mesh should not be simplified as heavily as other parts which are comparably irrelevant. These crucial parts could indicate geometric properties, e.g. curvature or edges between surfaces, or they could denote desirable semantics, e.g. windows on a facade. Therefore adaptive mesh simplification is imperative when mesh triangles should be variously decimated.

Unlike many existing methods, which focus on mesh simplification of single object like a statue or an industrial part, our method aims for simplification of large-scale scenes like city models. On the one hand this may lead to a problem since city scenes could have a great diversity and complexity in term of both shape and semantics. But on the other hand large-scale scenes enable the potential of exploiting 2D aerial images as an extra tool. Aerial images are not only required for 3D reconstruction, but can also contribute to mesh simplification e.g. by providing local features of interests. This enriches the possibility that a mesh can be simplified based on constrains from external sources.

1.2 Tasks

The main objective of this project is developing an adaptive simplification method on large-scale mesh with aerial images and their semantic segmentation. The first task should be initialization of a general mesh decimation framework. In this step a mesh model can be simplified uniformly without additional modifications. The second task aims to gather local features from aerial images and their semantic segmentation. The final task focuses on adding local features to the initial mesh decimation framework as constraints to realize an adaptive mesh simplification. After constructing this method, it will be applied to a real dataset for evaluations.

1.3 Structure of this report

After this introduction section notable previous works on the research area of mesh simplification will be briefly described in section 2. In section 3 the methodology of this mesh simplification method is thoroughly explained in steps. And results and discussion of application of our method on a real dataset are presented in section 4. A conclusive summary of our method is stated in the final section 5.

2 Related Work

A wide range of methods has been presented to simplify a triangle mesh. An overview of these methods can be found in [3].

2.1 Vertex Clustering

Some of the earliest algorithms fall under the classification of vertex clustering. Its basic idea is to divide the bounding box of a mesh into a grid, e.g. in the simplest case a rectilinear grid. All vertices inside a given cell are replaced with one single representative vertex. And faces that become degenerate in this process will be removed to ensure a consistent topography. A typical work that follows this idea is presented by [7] for approximating arbitrary polyhedral. [8] presents an adaptive vertex clustering approach using a dynamic octree.

2.2 Incremental Decimation

An alternative class of mesh simplification is incremental decimation. These algorithms take an iterative approach, in which a primitive decimation operator is continuously applied to a selected region in mesh. The operator is usually chosen to minimize the incremental error incurred by the operation and will stop being conducted till no further reduction is possible. A typical decimation operator is vertex removal where two adjacent vertices are either merged to a new vertex or into one of these both end points.

2.3 Multi-View Mesh Refinement

Many approaches have found potential benefits of exploiting multi-view images for mesh refinement. They often formalize the disagreement between the given mesh and images in a cost function and then attempt to minimize this cost function to learn proper adjustments on the mesh, like in [4] and [6]. [1] presents a method to jointly refine the geometry and semantic segmentation of 3D surface meshes. However, most of these approaches work on optimization of geometric consistency for 3D reconstruction. Using multi-view images for mesh simplification is the starting point for this project.

3 Algorithm

In this section the main algorithm of our mesh simplification method is explained in details. As a starting point we assume that an initial large-scale surface mesh is provided, e.g. from 3D reconstruction. Furthermore it is assumed that a set of calibrated images are available. Both intensity images and semantic segmentations in form of per pixel likelihoods for possible classes are considered as given. The main processing procedure is divided into five parts. In section 3.1 an initial simplification framework should be established from an existing general approach. Section 3.2 presents the technique of feature extraction from given images. The process of projecting extracted features from images to the mesh is described in section 3.3. And section 3.4 explains semantic label assignment of each vertex in the mesh. At last information from images and semantic segmentation will be formalized into constraints in section 3.5 to be embedded to the simplification framework.

3.1 Initial simplification framework

As an initial simplification framework we choose an iterative decimation approach with quadric error as criterion. The quadric error is originally promoted in [5] and has become one of the most commonly applied techniques in mesh simplification.

$$p = (x, y, z, 1)^T, q = (a, b, c, d)^T \tag{1}$$

$$\text{dist}(q, p)^2 = \left(q^T p\right)^2 = p^T \left(qq^T\right) p =: p^T Q_q p \tag{2}$$

$$Q_q = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & b^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \tag{3}$$

Given a vertex p in mesh and an adjacent plain q, the point-plane distance from the vertex p to the plain q is defined as in equation (2). This Q_q matrix known as the error quadric can be used to find the squared distance of any point in space to the plane q. We want to find a point p^* that minimizes the quadric error of sum distances to all adjacent plains in equation (4). It is evident that this point p^* should follow the condition in equation (5).

$$\sum_i \text{dist}\left(q_i, p\right)^2 = \sum_i p^T Q_{q_i} p = p^T \left(\sum_i Q_{q_i}\right) p =: p^T Q_p p \tag{4}$$

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} p^* = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{5}$$

This quadric error is needed to select vertices, on which a decimation operator should be performed. Instead of vertex pair contraction where two adjacent vertices are merged to a vertex in a new position, we applied the principle of halfedge collapse (Figure 1). In halfedge data structure each edge is decomposed into two halfedges with opposite orientations. Thus a halfedge collapse corresponds to moving a vertex to one of its adjacent vertices such that they overlap with each other.

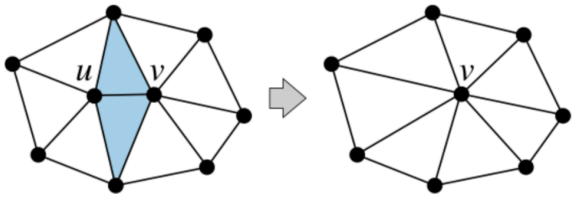


Figure 1: Halfedge collapse

As previously mentioned, this initial simplification framework will be conducted in iterations. In each iteration, the quadric error of each vertex should be computed. Based on this error value a priority queue will be created such that the lower the error value, the more a potential collapse moves to the front of the queue. The one with the lowest error will always be the candidate for the next collapse. We set a threshold for the quadric error so that only vertices with errors smaller than the threshold will be decimated.

3.2 Feature extraction from images

In this step we intend to extract local geometric features from given images. There are various types of feature that have been applied in the area of computer vision. In this project we use the canny edge detector. Because edges can reflect the general boundary of a object or a scene and describes changes of pixel intensity along a certain direction. In terms of mesh simplification, we want to increase the number of triangles in flat areas and preserve a high level of detail on fine structures which can be specified by edges.

(a) Original image (b) Canny edge image

Figure 2: An example of the Canny edge detector

The Canny Edge detector was developed by John F. Canny in 1986. Canny algorithm aims to satisfy three main criteria: low error rate, good localization and Minimal response

[2]. Like other approaches the Canny Edge detector mainly computes the intensity gradients of the image. And there are mainly three parameters in the Canny algorithm: kernel size, lower threshold and upper threshold. The kernel size affects the computation of intensity gradients. The upper and lower threshold jointly determine whether a pixel should be accepted as an edge or be rejected.

In many cases the Canny algorithm starts with a Gaussian filter to smooth the image in order to remove noise. But in this project this processing should not be carried out without further consideration. Since detected edges will be later projected to the mesh as lines, smoothing the image could affect the shape of resulting lines in the mesh. This will be further discussed in section 3.3.

3.3 Projection from images to mesh

One of the challenges in this project is the projection of edges from image to the given mesh. One prerequisite of a correct projection is the camera calibration information. This means that all images should be provided with camera projection matrices and capture a certain part of the mesh. Since no 3D reconstruction or multi-view triangulation is performed in this project, it is impossible to directly project a pixel from an image to a vertex in the mesh. Thus it should be accomplished reversely. A vertex in mesh should be reprojected to a pixel from an image. If this pixel is detected as an edge, then the corresponding vertex is also considered as an edge. In the process of this projection, the following three conditions must be fulfilled to declare a vertex as an edge.

Direction of vertex normal Any vertices must be captured from images before they are decided to be an edge. Thus we want to exclude occluded vertices in the first place for the possibility of being an edge. For a given mesh, the normal vector of each faces can be easily calculated. Thus the normal vector of a vertex can be computed by averaging the normal vector of its belonging faces. Since camera projection matrix is available for each image, the projection vector of an image can be acquired in object coordinates. After that the angle between each vertex normal vector and the camera projection camera can be computed. In an ideal case, a vertex is considered as pointing towards the image if the angle is near 180° (two vectors are nearly parallel). But it is evident that an image can capture a wide range of scenes and the angle does not necessarily be 180°. Thus we set a threshold for the angle value that only vertices with a angle value large than the threshold can be a candidate of an edge. This condition of checking direction of vertex

normal contributes to neglect occluded vertices in mesh, for example the vertices on the back of a building.

Image coordinates of vertices In practice it is almost impossible for each image to record the whole scene of a mesh since we are dealing with large-scale mesh like city scenes. With the help of camera projection matrix, each vertex can be projected to the image plain in image coordinates. Vertices outside the range of the image will also be projected into image coordinates that are either negative or larger than the maximal values on boundary. Those vertices should be neglected for further determination.

Edges in image Vertices that survive from the first two conditions are correctly reprojected to images. Their image coordinates must be integer. If this is not the case for some vertices, they should be rounded to the nearest integer. If the pixel with the same image coordinates are detected as an edge, then the corresponding vertex will also be labeled as an edge.

Although the above conditions contributes to transfer local edges to mesh, it does not guarantee that an edge in image will be projected to mesh also as a perfect edge. A key character is the relation between the image resolution and the density of the mesh. Let's consider the ideal case, where an edge in image is perfectly transfered to the mesh. That means each pixel on an edge uniquely corresponds to one vertex in a bijection form. And these vertices form a perfect edge in the mesh. In practice if the mesh has a higher density, more vertices can be projected to edges in images. This results in a region in the mesh instead of a single edge. If the mesh has a lower density, there will exist pixels with no corresponding vertex in mesh. So the resulting feature may have blank gaps or in worst case does not form an edge at all. This effect can be partly mitigated by filtering the images. If the mesh has a higher density, the Canny edge images can be blurred resulting in edges with large width. If vice versa, the original images should be blurred resulting in less number of edges.

3.4 Assignment of semantic labels to mesh

In this project, we assume that semantic segmentation of each provided image is available in the form of pixel-wise likelihoods for all possible classes. In order to make use of semantic information for mesh simplification, it should be transfered to the given mesh in such a way that each vertex has a semantic label. To achieve that, we reproject vertices

to images again. If a vertex can be reprojected to n images, then we extract the semantic likelihood vectors of n corresponding pixels, one on each of these images. These likelihoods vectors will be summed up as the semantic vector for these vertex. The final semantic label of this vertex is defined as the class with maximum likelihoods.

3.5 Embedding Constraints to initial simplification framework

In this part the information of local geometric features and semantic segmentation of the mesh should be embedded to the initial simplification framework as constraints. As described in section 3.1, a vertex should be removed if its quadric error is smaller than the threshold. Since we want to preserve local features (edges) from being collapsed, we want to formalize a constraint that can possibly determine whether a vertex belongs to an edge. In this project we introduce a parameter for each vertex as the times of being projected to an edge in the entire set of images. So a vertex will be considered as local feature if being projected more than a threshold for this parameter. In the meanwhile different thresholds could be applied according to the semantic label of a pixel. Here prior assumptions about the surface shape of different semantic classes could be experimented to find out a proper set of thresholds.

4 Experiments

In this section the presented adaptive mesh simplification method will be experimented on a real world dataset. This dataset is briefly introduced in section 4.1. Results of simplified mesh including quantitative and qualitative evaluation are presented in section 4.2.

4.1 Data

The processed dataset mainly covers the city Enschede (Netherlands). It consists of an original 3D surface mesh, intensity images and semantic segmentation maps. The original mesh has 108490 vertices. For a better understanding of out method, it has been densified to 1729159 vertices as the input mesh. Figure 3 gives an overview of the input mesh. A total number of 15 grayscale aerial images are available, 3 in portrait and 12 in landscape. All images have the same resolution of 1404×936 and their camera projection matrices are provided alongside. Examples for these aerial images are illustrated in Figure 4. For semantic segmentation four class labels are defined: roof, facade, vegetation and ground.

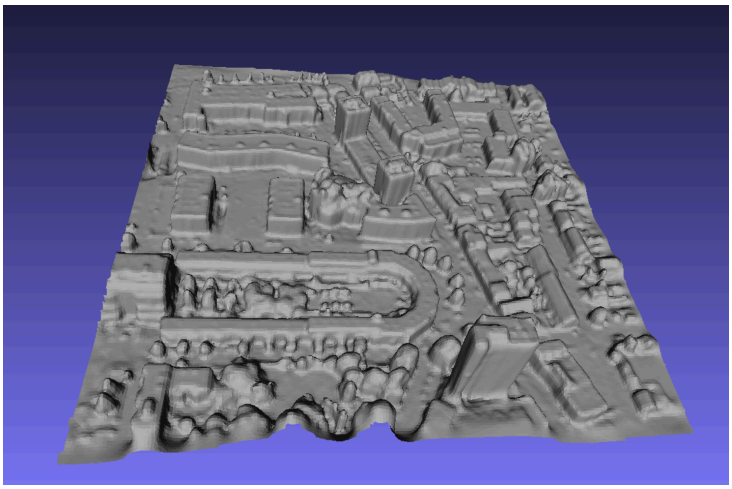


Figure 3: An overview of the input mesh

(a) Portrait orientation

(b) Landscape orientation

Figure 4: Examples of input grayscale aerial images

An example of semantic segmentation maps is given in Figure 5. Note this example is the result of *hard* classification by assigning the semantic class of maximum likelihood for each pixel in an single image. The reason for this processing is a better visualization, while actual semantic labels of vertices in mesh will be computed by summing up semantic vectors of corresponding pixels. Figure 6 gives an overall configuration of the input mesh and the aerial images. It also shows that many small triangles in the mesh are redundant and thus should be suitably removed.

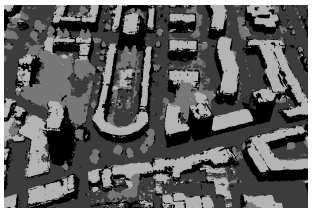


Figure 5: Example of semantic segmentation map

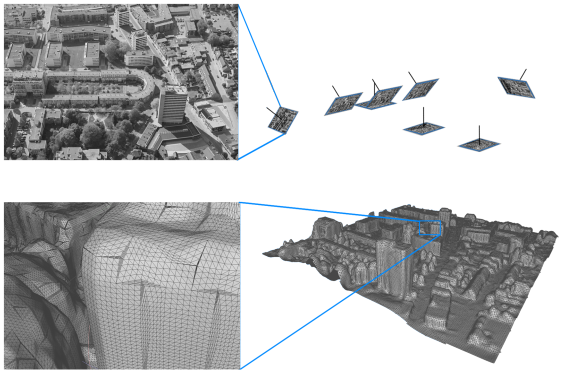


Figure 6: Conceptual configuration of mesh and images

4.2 Results and Discussion

In this project, most results are qualitatively evaluated since a perfect mathematical representation of the simplified mesh is merely possible. Intermediate results are also presented according to the methodology in section 3.

4.2.1 Edge detection

There are mainly three parameters that can be tuned in the Canny edge detector. We choose the kernel size to be 3, which is the size of the Sobel kernel. As for upper and lower thresholds, we applied the ratio between them to be 3 (following Canny's recommendation). The choice of the lower threshold depends on the application of the project and can be empirically tuned with plausible values. Here we set the lower threshold to 150 as shown in Figure 7, because edges should be rather sparse in image, especially when dealing with large-scale scenes. A smaller value results in more edges which can be a negative effect by accepting noise as edges.

(a) Original image

(b) Edge image with a lower threshold of 50

(c) Edge image with a lower threshold of 100

(d) Edge image with a lower threshold of 150

Figure 7: Edge detection with different threshold values

4.2.2 Semantic segmentation of the mesh

Using the method in section 3.4, per-vertex semantic labels of the mesh can be generated by projecting the per-image class scores onto the surface with aggregation. The resulting semantic segmentation of the mesh is illustrated in Figure 8. The four classes roof, facade, vegetation and ground are colored in red, yellow, green, and dark gray respectively. Generally most parts of the mesh are correctly labeled according to human perception. But it is also evident that the constructed semantics have a inevitable noise effect and the semantic consistency in some regions should be further improved.

4.2.3 Projection from images to mesh

As mentioned, projection local geometric features from images to mesh is a key point in this projection. Thus this must be confirmed before being embedded to the simplification framework. The straightforward solution is projecting edges from one single image and checking whether they are correctly localized in 3D mesh. From Figure 9 we can see that most of the edges are projected into correct positions in 3D space. And these edges can well represent the boundary of objects for e.g. building constructions. Also windows on building facades are properly symbolized with edges.

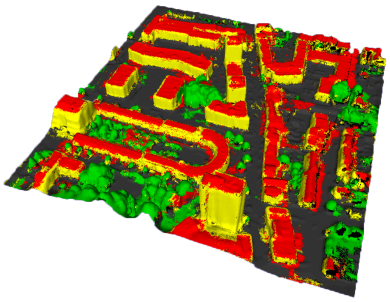


Figure 8: Semantic segmentation of the mesh

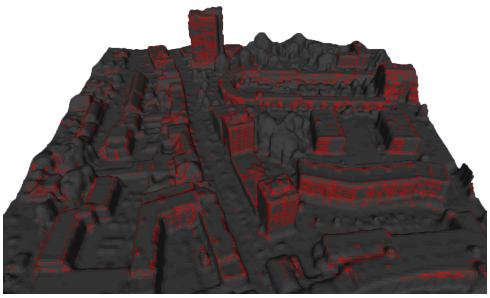


Figure 9: Projection of one edge image to the mesh

4.2.4 Simplified mesh

As mentioned in section 3.5, local features and semantics will be embedded into the simplification framework as constraints. The parameter we introduced is the counting of in how many images a vertex is detected as an edge. This parameter can be thresholded to different values according to the semantic label of vertices. So at this point prior about the surface shape of different semantic classes could be assumed. In general, we think ground and facade should be rather smooth. Roof and vegetation could have more complex details that may need more triangles be to modeled. After experiments, we consider the thresholds in Table 1 as plausible to be applied on our data.

Semantic label	roof	facade	vegetation	ground
Threshold	1	3	1	4

Table 1: Thresholds for different semantics

We compare our suggested adaptive mesh simplification method with the initial quadric

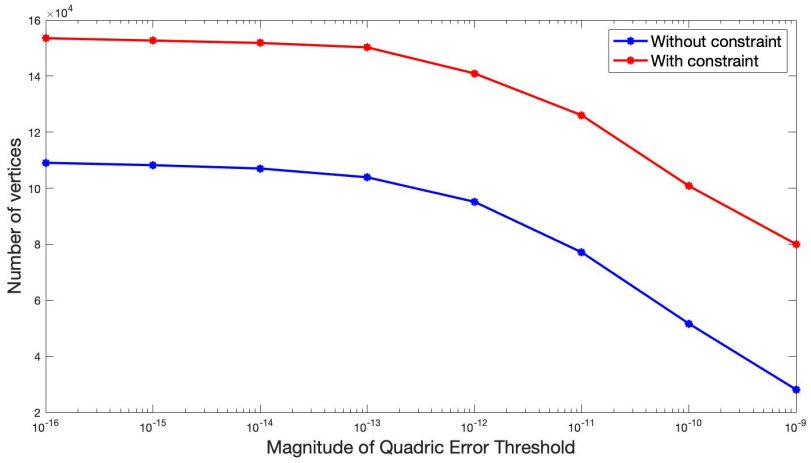


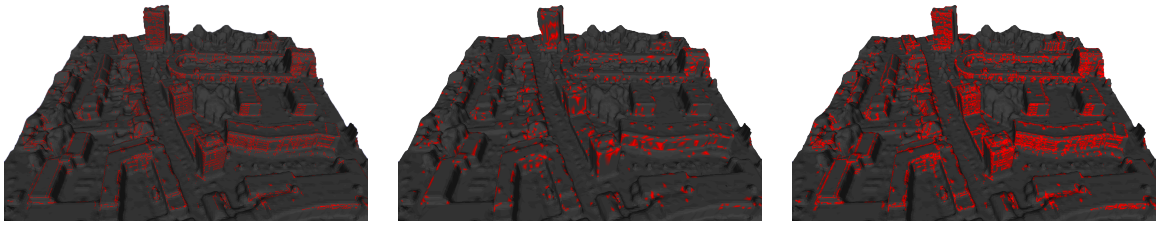
Figure 10: Comparison of the number of vertices after simplification with and without constraint

error framework with no constraints. The number of vertices will be considered as an quantitative evaluation criterion. Figure 10 shows the result of comparisons with the quadric error in different magnitudes. It is logical that the resulting mesh of constrained simplification has more vertices. But we can see that the number of vertices remains in the same magnitude and the difference in between does not vary a great deal.

As qualitative evaluations we first want to make use of the projection of a single image to the mesh. Since vertices that are detected as edges are marked in color, we can track these vertices after simplification to see whether the structure of edge still remains. Figure 11 shows that after a general simplification almost all previous edges are ruined into pieces. On the other hand, using our method could preserve those edges. They look a bit blurred because they are now modeled with less triangles.

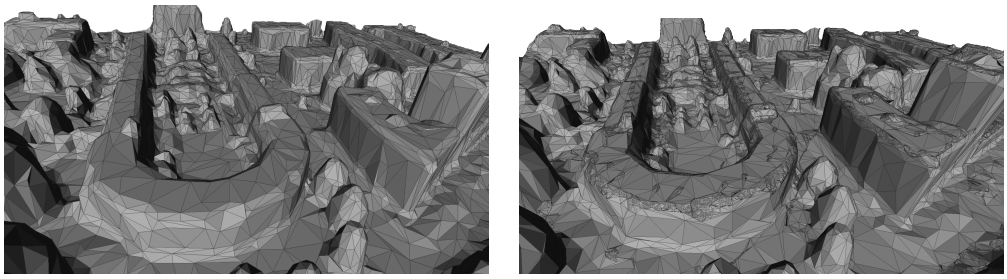
A closer look of the simplified mesh is shown in Figure 12. It is evident that details on the roof of buildings are preserved while the other parts are heavily simplified. Yet it is also possible for our method to be applied on different prior assumptions of the surface by changing the thresholds. Figure 13 shows the result of preserving features on facade while features on surfaces of other classes are ignored. In this circumstance windows of different floors can be roughly recognized.

During experiments, we found out that the semantic information could also contributes



(a) Before simplification (b) After general simplification without constraint (c) After adaptive simplification with constraint

Figure 11: Behavior of local features after simplification



(a) General simplification without constraint (b) Adaptive simplification with constraint

Figure 12: A near view of a simplified region of the mesh

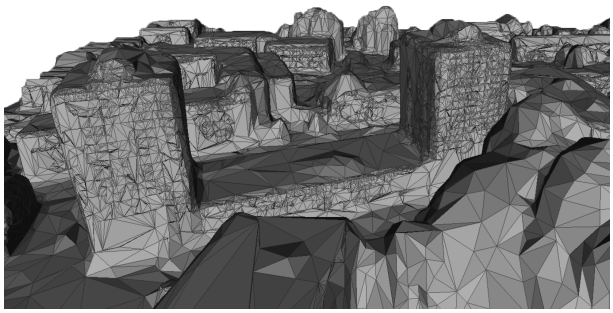
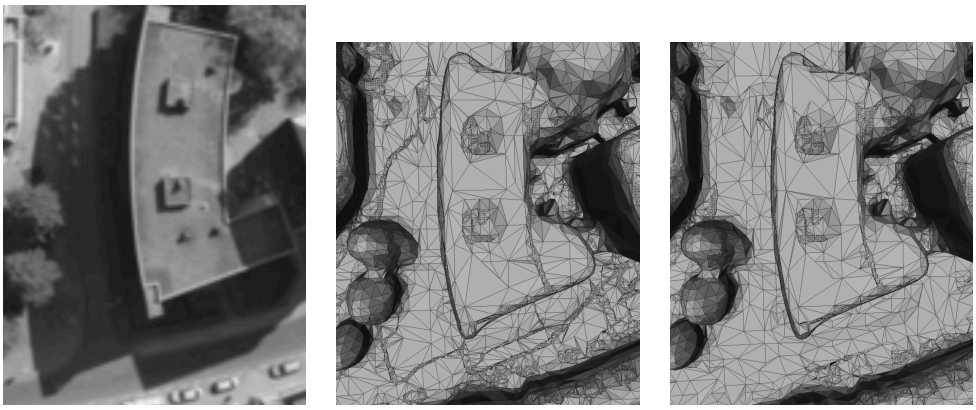


Figure 13: Preservation of features on facades



(a) A small region in an aerial image (b) Uniform thresholding (c) Hard thresholding for the class ground

Figure 14: Shadows on the ground after mesh simplification

to solve or mitigate the problem of shadow. An example is given in Figure 14. Shadow is a problem in aerial image processing since shadows of building on the ground could be detected as edges with a high probability. But these edges carry no information and the flat ground should be simplified in a few triangles. By a hard thresholding of the parameter for the semantic class ground, the detected edges could be neglected during the processing of simplification.

5 Conclusion

Within this project the potential use of calibrated aerial images and semantic segmentation to constrain mesh simplification is investigated. The presented method focuses on simplification on large-scale meshes e.g. city models, because many of them are reconstructed from image dense matching in the first place. Our method exploits intensity images to extract local geometric features. These features should be then projected to the given mesh and preserved from being collapsed during decimation. We introduce a parameter to quantify the probability of features and interpret this parameter as constraint by means of thresholding. Semantic information helps to build prior assumptions about surface of different classes, which could be realized by thresholding non-uniform parameters according to semantic labels. We test our method on a read dataset and acquire promising results.

Further improvement is possible in different aspects. For example, instead of counting the

times of feature detection, a probability mechanism could be introduced as parameter. In this way a vertex considered as local features only in few images could also be preserved if the detection probability is high enough. Another alternative of improvement is differential level of simplification for feature and non-feature surfaces. Instead of simple preservation, feature areas on surface could also be weakly simplified or modified in other operations depending on application.

References

[1] M. Bláha, M. Rothermel, M. R. Oswald, T. Sattler, A. Richard, J. D. Wegner, M. Pollefeys, and K. Schindler. Semantically informed multiview surface refinement. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3839–3847. IEEE, 2017.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[3] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37–54, 1998.

[4] A. Delaunoy, E. Prados, P. G. I. Piracés, J.-P. Pons, and P. Sturm. Minimizing the multi-view stereo reprojection error for triangular surface meshes. In *BMVC 2008-British Machine Vision Conference*, pages 1–10. BMVA, 2008.

[5] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997.

[6] Z. Li, K. Wang, W. Zuo, D. Meng, and L. Zhang. Detail-preserving and content-aware variational multi-view stereo reconstruction. *IEEE Transactions on Image Processing*, 25(2):864–877, 2016.

[7] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Modeling in computer graphics*, pages 455–465. Springer, 1993.

[8] S. Schaefer and J. Warren. Adaptive vertex clustering using octrees. *SIAM geometric design and computing*, 2(6), 2003.