Reconstructing Buildings as Textured Low Poly Meshes from Point Clouds and Images

Irene Reisner-Kollmann[1,2] Christian Luksch[1] Michael Schwärzler[1]

[1]VRVis Research Center Vienna
[2]Vienna University of Technology

Abstract

Current urban building reconstruction techniques rely mainly on data gathered from either laser scans or image-based approaches, and do usually require a large amount of manual post-processing and modeling. Difficulties arise due to erroneous and noisy data, and due to the huge amount of information to process.
We propose a system that helps to overcome these time-consuming steps by automatically generating low-poly 3D building models. This is achieved by taking both information from point clouds and image information into account, exploiting the particular strengths and avoiding the relative weaknesses of these data sources: While the segmented point cloud is used to identify the dominant planar surfaces in 3D space, the images are used to extract accurate edges, fill holes and generate textured polygonal meshes of urban buildings.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages

1. Introduction

Fast and easy reconstruction of three-dimensional buildings as polygonal models is highly demanded in various applications like GIS systems, cultural heritage, CAD systems or games. Both laser scanning and photometric methods are widely utilized in this field, providing different types of input data and bringing along various kinds of artifacts.

3D point sets often have poor quality due to noise, outliers, or missing data which can be handled e.g. by exploiting repeating structures [ZSW*10]. Our system overcomes these difficulties by reconstructing piecewise planar surfaces from both a point set and multiple images. While point sets provide extensive 3D information, images contain more accurate edges and boundaries. By combining these advantages, our approach is suitable for quickly generating textured low-poly 3D models of buildings in a typical urban scene.

A remarkable aspect of our reconstruction system is that it can be used fully automatically for a wide range of architectural scenes. At the same time, it is very easy to manually intervene during the process for improved results. This is traced back to the fact that individual steps are calculated fast, intermediate results can be individually re-calculated, and customizable parameters are easy to understand.

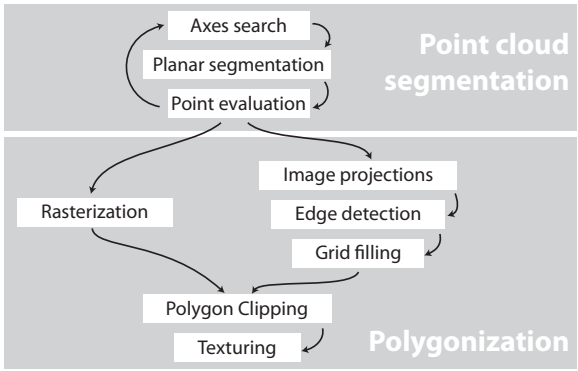


Figure 1: *Pipeline of our proposed method.*

As depicted in Figure 1, our system starts with segmenting the input points into planar clusters. A polygonal surface of each cluster is subsequently reconstructed by a grid-based evaluation exploiting multi-view image information, or if no

such data is available, by a point rasterization step. Finally, all polygons are combined to a joint, textured 3D model.

1.1. Input Data

Our reconstruction system uses a 3D point cloud and photos with known camera parameters (image shots with position and orientation) as input data. The points have to be oriented, i.e. each point has a surface normal assigned. It is sufficient to estimate the normals with a simple least squares method, as exact object edges are extracted from the images.

The input data can be acquired by image-based *structure from motion* techniques [SSS08, FP09], or by laser scanners with additional, registered photos. While both capturing methods provide a good impression of the scene to reconstruct, common artifacts like noise, mismatches and holes cannot be avoided due to reflective surfaces, occluders or varying lighting conditions.

2. Point Cloud Segmentation

The goal of our point cloud segmentation is to divide the oriented points into planar pieces and to remove points which are not located on a planar surface. The point cloud is first divided into clusters with similar surface normals and subsequently according to spatial distances. Finally, all unclustered points are evaluated and potentially added to an existing cluster. The segmentation is done iteratively on the remaining points until no additional clusters can be found.

2.1. Axes Search

The segmentation process starts with the search for dominant axes in the scene by evaluating all surface normals. For typical urban scenes the segmentation is improved by fitting predefined axis frames (90°, 45° or 60°) to the surface normals: Inaccuracies from the normal estimation are removed, and well aligned point clusters which only contain a relatively small number of points are detected. For the automatic system we search for 45 degrees in the first iteration, then for angles with 60 degrees and finally for the remaining dominant axes without angular relations.

The axis frame fitting uses a RANSAC approach. Two surface normals are taken randomly from the point set and define the orientation of the selected axis frame. The quality of a specific axis frame orientation is defined by the number of inliers, i.e. the surface normals that are within a maximum angle α to one of the predefined axes. Axes which have only a small number of inliers are removed from the axis frame. Figure 2 shows an example for a fitted orthogonal axis frame and the corresponding inlier normals.

The orientation of the best axis frame is improved by an iterative mean-shift [Che95] optimization step as denoted in Equation 1, where a identifies an axis, n a normal and

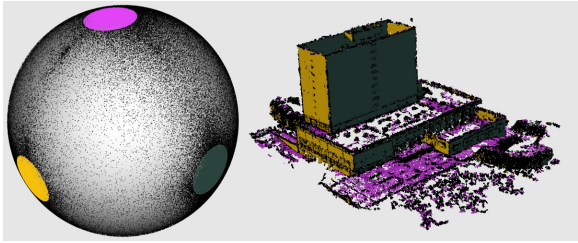


Figure 2: *Three orthogonal main axes have been identified in this scene. The colored areas on the normal sphere denote all points belonging to one axis cluster. The clustered points are shown with the same colors in the right figure.*

$Rot(u,v)$ returns the rotation matrix between two directions. For each axis, the mean unit vector is calculated from all normals within angle α weighted with function w. For each axis, the rotation to its local maximum on the unit sphere is computed. The final transformation for one iteration step is the weighted average of all rotation matrices, followed by an SVD-based orthogonalization.

$$R = \sum_a Rot\left(a, \frac{\sum_n w(a,n)\cdot n}{\|\sum_n w(a,n)\cdot n\|}\right)\cdot\sum_n w(a,n)$$
$$w(a,n) = \left(\frac{n\cdot a - \cos(\alpha)}{1-\cos(\alpha)}\right)^2 \quad (1)$$

The search for axes without angular relations starts with equally spaced seed axes on the unit sphere. The number of seeds depends on the angle α, which defines the maximum distance for potential inliers. All initial axes are independently optimized by mean-shift rotations, and will converge to centroids of local clusters. Only axes with sufficiently large clusters at their final position are accepted.

2.2. Planar Segmentation

Points with orientation close to a detected axis direction are grouped to a cluster, wherein multiple 3D planes are fitted. All cluster points are projected to a 1D space along the axis direction and clustered in order to find dominant planes. Detected planes are fit to their cluster points with weighted least squares. A small threshold is needed for clustering along the axis orientation in order to prevent two planes from merging.

Subsequent to the plane search, a 2D distance based clustering is performed to separate different surfaces which are located in the same plane. The points are clustered with single-linkage, i.e. there is a minimum distance between any two points of different clusters. The minimum distance threshold can be chosen less strictly, since potential holes in a surface will be removed during polygonization.

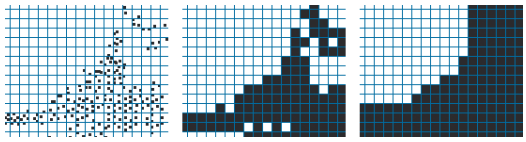


Figure 3: *Cluster points are rasterized into a binary image. Morphological closing is applied to remove small holes.*

Figure 4: *Nearly collinear line segments are merged and subsequently connected to form a grid.*

2.3. Point Evaluation

The previous clustering steps might miss some points due to noisy 3D positions or erroneous normals. Therefore, the final step of one iteration evaluates each unassigned point and adds it to the best-suited point cluster within a maximum distance. Similar as in Section 2.2, one threshold is used for the orthogonal distance to the plane, and one for the distance to the nearest point in the cluster. The remaining unclustered points are processed in the next iteration.

3. Polygonization

The final reconstruction step generates triangle meshes for the planar point clusters. The points are usually densely distributed, but the segment borders are not exactly defined, and holes may occur due to missing data. We provide two different algorithms for creating polygons: the first one only depends on the point cloud itself (Section 3.1), while the second one uses image information (Section 3.2-3.4).

3.1. Point Rasterization

Each point cluster is transformed into 2D space by projecting it onto its fitting plane. The 2D points are rasterized in a binary image, and small holes are removed by morphological operations (cf. Figure 3). In order to avoid rasterization artifacts, the points are aligned to globally dominant axes before rasterization. These dominant axes can be taken from the calculated main axes for normal segmentation (Section 2.1) or computed by principal component analysis.

This approach is fast and has proven to be robust in our test scenes, but artifacts in the point cloud propagate to the final polygon mesh. It is not possible to discriminate between real surface holes and artifacts due to missing data, and surface boundaries are often jagged. Hence, we incorporate image information as described in the next sections. Nevertheless, the rasterization approach is a good alternative for areas with missing or inferior image information.

3.2. Image Projection

The image-based surface reconstruction starts with selecting the best suited images for each planar cluster. Depending on the overall number of input images, two to four photos are selected. The selection favors photos with a steep view angle and a large unoccluded projection area of the planar cluster. In practice, this is done with the help of hardware occlusion queries and point splats.

The selected images are projected onto the slightly increased bounding region generated by the cluster points in the fitted plane. The projected images overlap correctly in areas where the plane coincides with the real surface.

3.3. Edge Detection

The final polygons should be bounded by strong image edges which correspond to edges located in the cluster plane. Gradient images are generated from all projected images, but high gradient values are only accepted if they occur in a majority of the images. This prevents taking edges into account which are not located on the current plane surface. Straight lines are extracted from the cumulated gradient image with a Canny edge filter followed by an edge linking step.

Experiences with many test scenes indicated that edges evoked by changes in texture are often much stronger than edges evoked by depth changes. Another observation was that in the case of buildings, depth changes are often aligned to orthogonal main directions. For this reason, two perpendicular main directions in the current plane image are detected, and only edges along these directions are accepted.

3.4. Grid Filling

The set of line segments is simplified by merging nearly collinear segments. Based on the resulting line segments, a rectilinear grid is created by extending all lines to their next intersection with another line (cf. Figure 4). The outer grid boundary is defined by the union of all line segments and all projected cluster points.

The point density for each cell is defined as the ratio between the number of its inlying points and its area, normalized by the median density of all cells. Cells belonging to the object surface are identified by thresholding their density. A polygon per cluster is then created by simply transforming the boundaries of accepted cells from 2D plane coordinates to 3D world coordinates.

3.5. Combination and Texturing of Polygons

In the previous sections the polygonization was done for each point cluster separately, but in most cases the individ-

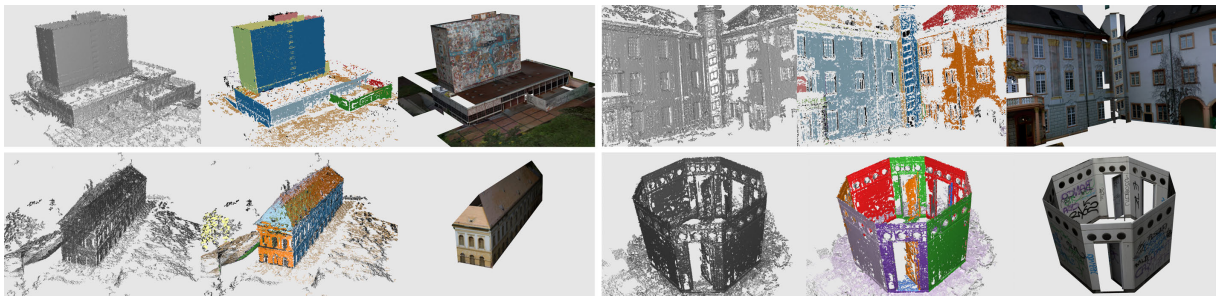
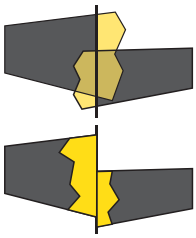


Figure 5: *Results: Library and Castle-P19 (top row), Manor and Octagon(bottom row).*

ual polygons have direct neighbors from other point clusters. The overall result can be highly improved by closing gaps and correcting intersections between neighboring polygons.



Clipping splits the geometry of a cluster by the intersection line of the two neighboring polygon planes. A part is deleted if its relative size is very small, or if the number of inlying cluster points is very low. *Merging* extends the existing geometry from the polygon border to the plane intersection line if the relative size of the new part is small. Both operations have very low complexity since all polygons are planar and have a low number of triangles.

The combined polygonal 3D model is textured by stitching together the photos selected in the image projection step (Section 3.2) using Poisson image blending.

4. Results

We have tested our approach on several input data sets. All point clouds have been created with PMVS [FP09] from oriented images. Figure 5 shows the input point cloud, the segmentations and the final 3D model for each test data set. The second data set *Castle-P19* is a publicly available multi-view test data set [SvHG*08]. Table 1 lists the number of input points, input images, planar segments, and the computation time for generating the triangle meshes. The reconstructions for the data sets *Library* and *Octagon* have been done fully automatically, whereas for the other data sets some parameters in the processing steps have been adjusted manually.

5. Conclusion

We have presented a fully automatic pipeline for generating low-poly 3d models of buildings using a point cloud and additional information from oriented images. The method is divided into a planar segmentation of the point cloud, and a subsequent polygonization of the point segments. Optional manual interventions are very simple and do not need sophisticated knowledge of parameters.

	Points	Images	Segments	Time (s)
Library	253126	21	19	91
Castle-P19	330165	19	23	105
Manor	179278	21	10	68
Octagon	560092	27	19	82

Table 1: *Detailed numbers about the data sets in Figure 5.*

Due to the coarse approximation of buildings as planar polygons, small features are not reconstructed in detail. This is largely compensated by the generated textures, and can be additionally enhanced by advanced texturing techniques such as normal or displacement mapping. Moreover, the low geometric complexity might be desired in various applications, or a starting point for further manual modeling.

Acknowledgments

This work has been accomplished as part of the lecture *AKGEO Topics in Geometry Processing* at TU Vienna in Spring 2010. We thank Niloy Mitra for his valuable input and support.

References

[Che95] CHENG Y.: Mean shift, mode seeking, and clustering. *PAMI 17*, 8 (1995), 790–799. 2

[FP09] FURUKAWA Y., PONCE J.: Accurate, dense, and robust multi-view stereopsis. *PAMI* (2009). 2, 4

[SSS08] SNAVELY N., SEITZ S. M., SZELISKI R.: Modeling the world from Internet photo collections. *IJCV 80*, 2 (2008), 189–210. 2

[SvHG*08] STRECHA C., VON HANSEN W., GOOL L. J. V., FUA P., THOENNESSEN U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR* (2008). 4

[ZSW*10] ZHENG Q., SHARF A., WAN G., LI Y., MITRA N. J., COHEN-OR D., CHEN B.: Non-local scan consolidation for 3d urban scenes. In *SIGGRAPH* (2010). 1