Semantically Aware Urban 3D Reconstruction with Plane-Based Regularization

Thomas Holzmann, Michael Maurer, Friedrich Fraundorfer, Horst Bischof

Institute of Computer Graphics and Vision, Graz University of Technology
{lastname}@icg.tugraz.at

Abstract. We propose a method for urban 3D reconstruction, which incorporates semantic information and plane priors within the reconstruction process in order to generate visually appealing 3D models. We introduce a plane detection algorithm using 3D lines, which detects a more complete and less spurious plane set compared to point-based methods in urban environments. Further, the proposed normalized visibility-based energy formulation eases the combination of several energy terms within a tetrahedra occupancy labeling algorithm and, hence, is well suited for combining it with class specific smoothness terms. As a result, we produce visually appealing and detailed building models (i.e., straight edges and planar surfaces) and a smooth reconstruction of the surroundings.

1 Introduction

Nowadays 3D reconstruction software is available, with which it is possible to easily create accurate 3D models from image data (commercial products as well as research community-based approaches). However, for certain applications like visualization of urban environments for construction industry, real estate companies and mapping services like Google Maps, a compact and visually appealing reconstruction consisting of planar surfaces and straight outlines is desired. Such a reconstruction should contain little noise and not necessarily all details from every scene part. We define the following criteria for a visually appealing urban 3D model, which are usually not tackled by current 3D reconstruction approaches:

- *Planar shape prior:* When planar and nearly planar surfaces exist in the scene (e.g., facades, roofs), they should also be reconstructed as planar surfaces.
- *Straight building outlines:* Edges of buildings should be straight and represented by a straight line (i.e., no noisy edges).
- *Detailed buildings and smoothed surroundings:* As for several applications (e.g., real estate companies) detailed building reconstructions are required, details should be kept while regularizing with a plane prior. Simultaneously, the surrounding of buildings does not necessarily be reconstructed with high details but with a smoothed, visually appealing surface.

These criteria do not only define a visually appealing reconstruction, but make it also easier to reduce the amount of data in a post-processing step (i.e., points lying on a planar surface can be diminished without changing the surface).

Fig. 1. *Results of the proposed approach textured with [33].* One can observe that planar parts in the scene (facades, windows, roof) are represented by planar surfaces and building edges where two facades intersect each other are represented by straight lines. Note that holes in the model are due to missing visibility information during texturing.

To apply these criteria, we use semantic priors in the reconstruction process to treat building and surrounding scene parts differently. For building parts, we incorporate plane priors in order to achieve planar surfaces and straight outlines while still keeping important details. For non-building parts, we impose a smooth surface by reestimating a smoother 3D representation of the scene, setting class specific sparsification parameters and smoothness terms. We partition the scene into volumetric cells using a Delaunay triangulation and perform Graph Cut-based inside/outside labeling. As a result, we compute a watertight polygonal mesh resulting from the interface of inside and outside labeled cells of the triangulation. In order to reduce the amount of data needed, mesh simplification can be applied in a post-processing step without loosing accuracy.

Our main contributions are threefold: First, a plane detection algorithm using 3D lines for detecting planes. Compared to point-based RANSAC methods like [29] we are able to detect more planes especially in urban environments which often contain poorly textured and, hence, sparsely reconstructed scene parts (e.g., white building facades). Second, we introduce an improved visibility-based energy term. In order to intuitively combine visibility-based terms with additional energy terms, a normalized energy is necessary. Currently used formulations (e.g., proposed by Labatut et al. [16]) lack in the possibility of normalizing the resulting energy. Third, we incorporate semantic priors into the reconstruction process. Depending on the semantic classes, we process the 3D data differently and set class-specific shape priors in order to get planar surfaces and straight edges for buildings and a smooth reconstruction for the surrounding. Fig. 1 visualizes textured results of our approach.

2 Related Work

In the past years, several works were presented focusing on creating visually appealing results for urban 3D reconstruction. Generally, most of them follow the idea that urban scenes can be largely approximated by geometric primitives and, hence, detected primitives are used to approximate the scene or to denoise and smooth the 3D reconstruction. Others use semantic information in order to simultaneously optimize the reconstruction and the semantic labeling.

Primitive Fitting. Several approaches try to fit primitives and then create a 3D scene reconstruction by directly using the fitted primitives (e.g, [35, 24, 21, 20]) or incorporating them into an optimization framework (e.g., [11, 25, 17, 18]) in order to create compact and visually appealing 3D models. Often, a RANSAC-based primitive detection approach (e.g., as described by Schnabel et al. [29]) is used, but there exist also other methods, especially for plane detection (e.g.,[4]). However, all these methods use point clouds for fitting primitives and, hence, the extracted primitive set is likely to be incomplete if primitives are represented with too few points (frequently happens at poorly textured facades reconstructed with image-based methods). Hence, we are using a different scene information (3D lines) which is more likely to be present at poorly textured urban scenes.

Reconstruction Using a Scene Hypothesis. Different works focus on reconstructing scenes with a very specific scene prior and, hence, work well for these specific scenes but do not generalize to others. Li et al. [20] use detected planes to create a set of axis-aligned boxes that approximate the geometry of a building. Following the idea of slicing of the scene (as proposed for indoor scenes in [34, 26]), Holzmann et al. [10] presented an approach to create visually appealing building models. Even though these approaches produce well regularized models, they are restricted to a specific scene arrangement (Manhattan world assumption or scene dividable into slices). Assuming a mainly planar scene, Monszpart et al. [24] aim to extract a regular arrangement of planes. Nan and Wonka [25] proposed an approach where they fit planes and intersect all the detected planes with each other to generate a possible set of faces for the final reconstruction. The final surface is generated by solving an optimization problem using all these face candidates. Even though these methods are not constrained to a Manhattan world assumption, they are designed for scenes containing mainly planar surfaces. In contrast, our approach has a special regularization prior for planar surfaces but can handle arbitrary scene structure.

Shape Priors Incorporated in Global Optimization. In the works of Labatut et al. [17] and Lafarge et al. [18, 19] primitives are incorporated within a tetrahedral representation of the scene and can be selected within the Graph Cut optimization. Following their idea, [11] added an improved plane augmentation which does not require a dense oversampling of the scene and added additional regularization terms. All these methods can reconstruct arbitrary scenes and regularize scene parts depending on detected shapes. However, wrongly regularized scene parts may result in artifacts, which we tackle by using semantic information and regularizing differently depending on semantic label.

Semantic Reconstruction Recently, several methods incorporated semantic information in the 3D reconstruction process [7, 1, 32]: They incorporate semantic information into an optimization framework to solve a multi-label 3D reconstruction problem using a voxel grid or tetrahedral representation. By using class-specific shape priors, they simultaneously optimize the 3D reconstruction and the semantic labeling. Compared to them, we do not simultaneously optimize semantics and the 3D reconstruction but use the semantic information in order to create visually appealing 3D models following our defined criteria.

3 Urban 3D Reconstruction with Semantic Priors

In this section, we give an overview of our processing pipeline and subsequently describe each part in detail. As a result, we deliver beautiful, visually appealing 3D models from urban scenes where buildings have planar surfaces and straight edges while still containing relevant details embedded in a smoothed surrounding.

Taking images from a scene as input, we first compute the camera poses using Structure-from-Motion, a dense point cloud using a Multi-View Stereo algorithm and a line-based 3D reconstruction. As further preprocessing steps, we compute dense depth maps for every camera and semantically label every image. In order to generate a very smooth reconstruction of the surrounding and a detailed but prior-based smoothing of the buildings, we semantically label all 3D information and incorporate the building and non-building classes differently into a Delaunay triangulation-based reconstruction framework. For buildings, we use all the available 3D information (i.e., 3D lines and points). We detect planes in the scene by using the reconstructed 3D lines from buildings and enforce the triangulation to include all planes. For non-building parts, we compute a smooth Poisson surface reconstruction and use a sampled representation of this surface. Our final 3D reconstruction results from a 3D Delaunay triangulation, where every cell is labeled inside or outside by solving an energy minimization problem using Graph Cuts including energy terms depending on the semantic label.

3.1 Semantic Segmentation

The goal of the semantic segmentation is to get the 3D reconstruction semantically enhanced to be able to perform automated decisions throughout our processing pipeline. To achieve this goal we follow the work of [23] to perform pixel-wise semantical segmentation of the input images. To transfer the labels from 2D to 3D, each 3D point is back projected according to its visibility to 2D and a final majority voting determines the label of the 3D point.

For the semantic segmentation of the input images we use a Fully Convolutional Neural Network (FCN) [22] to get pixel-wise segmentations. The network presented in [22] is adjusted to represent the number of output classes required for our task. We define five output classes, namely: street/pavement, building, vegetation, sky and clutter. As our intermediate aim is to semantically enhance the 3D reconstruction we need a pixel accurate segmentation of the input images where the segmentation boundaries are aligned with the objects present. Thus, the receptive field of the FCN of 32 px and the final up sampling by a factor of eight are too coarse to achieve this goal. We extend the 2D segmentation network by adding a Conditional Random Field represented as Recurrent Neural Network (CRFasRNN) as presented in [36]. The Conditional Random Field exploits the probabilities of the FCN and refines them by taking binary constraints into account. This enforces label changes being aligned with edges.

Having the pixel-wise semantic segmentation of the input images, we propagate this information to 3D: Assuming 3D points with visibility information, we back project every point into every image in which it is visible in and compute

a point label by majority voting. For getting labels of the 3D lines, we sample every line with points and compute a label for every point as explained above. The most frequent label within the sampled points defines the line label.

3.2 Plane Detection Using Lines

A very common approach to detect planes in 3D is to use a RANSAC-based algorithm with a point cloud as input (e.g., [29]). However, especially in urban environments where scene parts like facades might be poorly textured, these approaches fail due to missing reconstructed 3D points. In comparison, 3D lines are more likely to be detected at building facades, as some high-gradient elements like windows or building outlines usually exist. Hence, we are using this 3D line information to improve plane detection in urban environments. As our goal is to reconstruct a well smoothed surrounding of the building, we just use lines labeled as building and ignore all the others.

Assuming to have a 3D reconstruction consisting of line segments, we first detect line triples which already describe a plane hypothesis. Then we cluster the triples which are coplanar and in vicinity. Finally, we detect all inlier lines from the plane hypothesis.

Line Triple Detection. As we explicitly want to model man-made scenes frequently having rectangular outlines, we search for perpendicular coplanar line pairs which can be used to describe a plane. Additionally, we only accept line pairs which have a small distance between each other. For the coplanarity and perpendicularity tests we accept errors up to $\alpha_{error} = 5\ deg$, and the normal distance from start/end point of the line segment to the computed plane hypothesis must not be bigger than $d_{inlier} = 0.15\ m$. The distance between start/end point of the two line segments must not be bigger than $1.5\ m$ and we ignore line segments shorter than $0.8\ m$. In order to perform an early removal of spurious planes, we search for a third supporting line which has to be coplanar with the line pair and with small distance to the pair. We just accept the line pair if a third line exists. Note that line segments can also be part of several line triples, which is beneficial for lines which are exactly, e.g., at corners of a house.

Line Triple Clustering. After having estimated plane hypotheses by detecting line triples, several hypotheses can be nearly identical. Therefore, we cluster line triples which represent the same planar surface. We cluster line triples by first checking if the triples are nearly coplanar (i.e., normals of plane hypotheses with enclosing angle smaller α_{error}, normal distance from line triple (i.e., start/end point of its segments) to the current plane hypothesis lower than d_{inlier}). From these coplanar line triples, we greedily add all line triples to a cluster which have a maximum distance of the line projections on the plane of $12\ m$ to the previous line triple. After having clustered the triples, the lines are sampled and the sampled points are used to reestimate the plane using SVD.

Inlier Detection and Outline Estimation. Finally, we detect all inlier (i.e., lines segments being nearly coplanar in terms of angle α_{error} and distance d_{inlier}) which have a distance of the line projections on the plane smaller than $1.2\ m$.

We estimate an outline of the plane by computing a bounding box around all inlier segments and reestimate the plane parameters with all inliers using SVD.

Plane Filtering and Plane-Based Denoising. Having estimated the planes, we filter out plane segments which are included in another plane hypothesis (i.e, having the same plane parameters and the outline is included). Additionally, we filter out planes which don't have sufficient supporting 3D data (i.e., points and sampled line segments, see Sec. 3.3) within d_{inlier} normal distance. These planes are usually erroneous detections due to a specific line segment arrangement. Finally, the input data is denoised: We move all points and line segments which are in d_{inlier} normal distance to a plane onto the plane by normal projection.

3.3 Input Data Subdivision and Processing

Depending on the semantic label, we subdivide the scene into two parts which will be processed differently: For the building part we keep all available 3D information. For the non-building part we sparsify the input data, compute a Poisson surface and use the sampled Poisson surface which results in a smoother, visually appealing reconstruction (e.g., less spurious peaks at vegetation). In the final optimization the subdivided parts are combined again in order to create a reconstruction of the whole scene.

As we want to have a point cloud representation of the building part which covers all important details, we add all input points and additionally add points from sampled building line segments to the scene (line sampling distance $0.05\,m$). Adding the sampled line points especially helps at poorly textured scene parts where few reconstructed points are available.

In contrast to the building part, we want a very smooth representation of the surroundings of the building. Hence, we first sparsify these classes: For the clutter class we just keep every fifth point, for street/pavement and vegetation we keep every third point. Then, we compute a Poisson surface [14] using these selected points. For the subsequent reconstruction steps, we don't use the original street/pavement, vegetation and clutter points but a sampled point representation of the computed Poisson surface. This results in a much smoother surface of this part of the scene in the final reconstruction.

3.4 3D Reconstruction using Tetrahedral Occupancy Labeling

In this section, we describe the final reconstruction process. We explain the tetrahedra subdivision using the detected planes and the visibility prediction using depth maps, we propose a normalized visibility-based energy term and define class-dependent energy terms. As our approach is using a tetrahedral representation of the whole scene, we compute a Delaunay Triangulation of all the scene points (i.e., all available points of building parts and the sampled Poisson surface for the surroundings). Then, we subdivide the tetrahedra using the detected planes and solve an energy minimization problem by minimizing the following energy using Graph Cuts [2]:

$$\underset{\ell}{\text{minimize}} \quad E_{\text{Vis}}(\ell) + E_{\text{Class}}(\ell) \ , \tag{1}$$

where $E_{\text{Vis}}(\ell)$ is the visibility-based energy and $E_{\text{Class}}(\ell)$ are class-specific energy terms, which will be explained in more detail in this section. Finally, we get an inside/outside labeling for every cell from which a surface mesh can be extracted.

Tetrahedra subdivision. Even though many points lying on the detected planes are included in the triangulation, it is not guaranteed that the whole planar surfaces are included as faces. Hence, following the method described in [11], we compute intersections of planes and tetrahedra and add the resulting facets and vertices to the triangulation. After this subdivision step, the triangulation is not necessarily Delaunay any more but contains all detected planar surfaces which consequently can be selected by the final optimization.

Visibility Prediction Using Depth Maps. To compute visibility-based energy terms, the knowledge of the visibility information of every 3D point (i.e., camera-point correspondences) is necessary. Hence, most of the methods following visibility-based cost computations similar to Labatut et al. [16] assume that the visibility information is known. However, when tetrahedra are subdivided, new points without visibility information are created. Additionally, input 3D information without visibility information cannot be used. Hence, we propose to compute this information for all 3D points using depth maps.

Assuming to have dense depth maps for every camera, we project every 3D point into all cameras. If the point is within the image boundaries and in front of the camera, we compare the actual distance of the point to the camera with the depth value in the depth map (using nearest neighbor). If the depth difference is small enough (i.e., smaller than 0.03 m), we assume that the current point is actually visible in this camera and store this camera-point correspondence.

Improved Visibility-Based Energy. Visibility-based energy terms as proposed by Labatut et al. [16] tend to be very hard to normalize, as the magnitude of the energy depends on the density of the point cloud, the number of cameras the current tetrahedron is visible in and the visibility information of the surrounding. Hence, combining it with other energy terms tends to be hard.

Therefore, we propose an improved energy formulation, which is slightly better in terms of accuracy and, more importantly, has a normalized magnitude with which a more intuitive combination with additional energy terms is possible.

Inspired by [16] and [12], the energy terms are based on ray casting from every vertex to every camera the vertex is visible in. Unary costs are assigned to cells intersected by a ray adjacent to the visible vertex (i.e., before and behind the vertex) and pairwise costs are assigned to facets intersected by rays (see Fig. 2). Opposing to [16], we do not only assign pairwise terms in one direction but in both, and we additionally add unary costs in front of a visible vertex. This has shown to significantly improve the result when using normalization afterwards. In order to generate normalized cost terms, our general idea is the following: The visibility-based energies should change significantly when the terms are still low and additional information is added, but the influence of additional visibility information when already sufficient information is available should be decreased. Hence, it should have a significant effect if a point is visible by 1 or 5 cameras, but the effect should be reduced if the visibility is changed from 21 to 25 cameras.

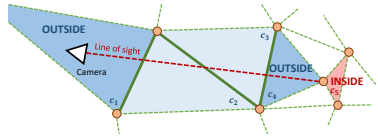


Fig. 2. *Visibility-based energy computation.* We use ray casting to compute the visibility terms: The cell where the camera is located in (c_1) is labeled as outside by adding infinite weights. Then, every facet (green) which is intersected by the line of sight (red) gets pairwise costs assigned in both directions. Finally, the cell in front of the visible vertex (c_4) is labeled as outside by adding finite weights and the cell behind the vertex (c_5) is labeled as inside by adding finite weights.

The normalized unary terms for cells directly in front of and behind visible vertices are defined as follows:

$$E_{unary}(t) = (1 - e^{-\frac{\#rays}{limit_u}})limit_u, \tag{2}$$

where t defines the current tetrahedron, $\#rays$ define the number of rays intersecting the tetrahedron and $limit_u$ is the energy limit approached asymptotically.

Additionally, cells including a camera and infinite cells are labeled as outside.

The normalized pairwise terms for every facet are defined as follows:

$$E_{pairwise}(f) = (1 - e^{-\frac{\#rays}{limit_p}})limit_p, \tag{3}$$

where f defines the current facet, $\#rays$ define the facet's number of ray intersections and $limit_p$ defines the energy limit. The limits of the unary and pairwise energies need to be set so that additional energy information is not ignored too early. Additionally, the pairwise terms need to be allowed to become stronger, as otherwise at large facets at holes or below roofs the unary term might spuriously dominate and, hence, artifacts might arise. We found out empirically that a setting for $limit_u = 8$ and $limit_p = 24$ is a good choice for most scenes.

These improved visibility-based energy terms are non-negative and submodular. The output is a normalized energy having maximum unary and pairwise terms, which is crucial for combining it with additional energy terms and, hence, makes it possible to easier find scene-independent parameter settings.

Class-Dependent Energy Terms. Depending on the semantic class facets and cells in the triangulation are assigned to, additional energy terms are added. First, we compute the class dependence for every facet and cell by computing a majority vote using all their corresponding vertices. Then, scene parts get assigned different energy terms depending on their semantic labels.

The energy terms assigned to building parts favor Manhattan-like structures but simultaneously aim to keep important details and are defined in [11]: They consist of a Manhattan regularity term E_{Man}, which favors label transitions with Manhattan-like surface structures (i.e., neighboring faces with enclosing angles similar to 0 or multiples of 90 degrees), and a level of detail term E_{LoD}, which punishes volumetric errors with respect to the unregularized model. Hence, E_{LoD} is the counterpart to E_{Man} and brings back smoothed out details which are not

supported by planes. Using these energy terms, we strongly favor planar and Manhattan-like structure while still keeping sufficiently big details.

For non-building parts, our goal is to get a reconstruction which is as smooth as possible. Therefore, we just add an area smoothness term E_{area} as defined in [16]. This term should remove spurious artifacts.

Hence, the class-specific energies are defined as follows:

$$E_{\text{Class}}(\ell) = \begin{cases} \alpha_{\text{Man}} E_{\text{Man}}(\ell) + \alpha_{\text{LoD}} E_{\text{LoD}}(\ell) & \text{if building} \\ \alpha_{\text{area}} E_{\text{area}}(\ell) & \text{else} \end{cases}, \tag{4}$$

where α_{Man}, α_{LoD} and α_{area} define the amount of smoothing.

4 Experiments

In this section, we first describe implementation details and the input data. Then, we evaluate the plane detection algorithm, the improved visibility-based energy and the effect of semantic priors. Finally, we compare results from our approach with others and show the effect of mesh simplification as post-processing step.

Implementation Details. Our pipeline is mainly implemented in C++ using CGAL [3] for the Delaunay triangulation and the Poisson meshing (used for the surroundings of the building). The semantic segmentation network is realized in the Caffe framework [13]. For initialization we exploited the weights of the PASCAL-Context network [27] and performed a transfer learning of the network based on 27 labeled training images (16 manually labeled, 11 taken from eTRIMS dataset [15]) that were augmented in scale (0.8, 1.0, 1.2), rotation [deg] (0, 90, 180, 270) and mirroring. Additionally the augmented images were cropped to patches of 256×256 *px* to easily fit to GPU memory. In total we resulted in a training database of 32,016 image patches. The training itself has been performed in stages to consecutively train the FCN32s, FCN16s, FCN8s and FCN8s with CRFasRNN. Each stage was trained for 400,000 iterations using Stochastic Gradient Descent with a momentum of 0.99, weight-decay of 0.0005 and a learning rate of $1e^{-9}, 1e^{-10}, 1e^{-12}$ and $1e^{-12}$ for each stage respectively.

The parameters for the final reconstruction were set as follows: For datasets *House* and *Residential Area* the parameter set was $\alpha_{Man} = 1000$, $\alpha_{Lod} = 500$. For the dataset *Block Building*, the parameters were set differently to impose a stronger plane prior as this dataset contains more noise near to planar surfaces ($\alpha_{Man} = 2500$, $\alpha_{Lod} = 1250$). α_{area} was set to 0.5 for all datasets.

Input Data. We evaluated on three datasets, from which example images are depicted in Fig. 3. Each of them consists of images acquired with a Micro Aerial Vehicle (MAV) and captured with a Sony Alpha 6000 camera with 24.3 MPixel. The first dataset, to which we refer to as *House*, contains 233 images and shows a scene with a family house and surroundings consisting of mostly grass and trees. The second dataset, *Residential Area*, contains 446 images and consists of two family houses (the others are not covered sufficiently by the images). For these two datasets, also ground truth captured with a total station is available and ground control points were measured to align the ground truth with the image based reconstruction. The third dataset is named *Block Building*, contains

House Residential Area Block Building

Fig. 3. *Example input images from the evaluation datasets.* The *House* dataset consists of a family house surrounded mainly by vegetation. The *Residential Area* dataset consists of several family houses and the *Block Building* dataset consists of a office building with mainly Manhattan-like structure.

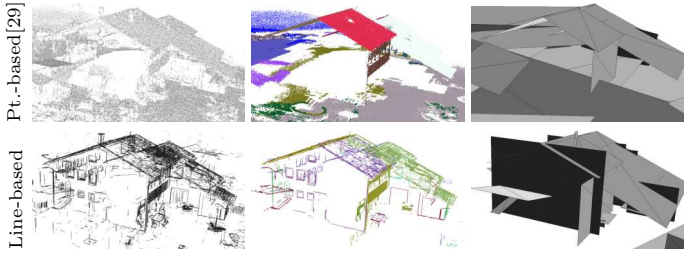


Fig. 4. *Comparison to point-based plane detection.* In the left column, one can see 3D input data (top: 3D points, bottom: 3D line segments): Especially at the front facade the point cloud has very few points while the line representation still contains some lines (e.g., at windows and at building edges). In the middle column, points and lines corresponding to extracted planes are illustrated (randomly colored). One can observe that the point-based approach detects planes at densely sampled surfaces well while missing sparsely reconstructed surfaces. The proposed line-based approach also detects planes which are just represented with few line segments. In the right column, one can see plane segments created by fitting bounding boxes around the inlier data on the plane surfaces. In the result of the point-based approach, spurious plane segments become visible while the line-based approach mainly contains the planes of the building.

232 images and includes a Manhattan-like building consisting mainly of poorly textured facades and windows. As this dataset has no metric scale, we manually scaled it to be approximately metric.

To compute the camera poses, we used our own Structure-from-Motion implementation. The dense point cloud for *House* was computed using Sure [28] and consists of approx. 900K points. The dense point clouds for *Residential Area* and *Block Building* were computed with PMVS2 [5] and contain 3.6M points and 1.4M points, respectively. The 3D line reconstructions were computed with Line3D++ [9] and the input depth maps with PlaneSweepLib [8].

Plane Detection Using Lines. In this experiment, we compare our proposed plane detection algorithm using lines with a state-of-the-art RANSAC-based plane detection algorithm proposed by Schnabel et al. [29]. Fig. 4 shows an overview of the detection process for both approaches. For comparison, we used the implementation of [29] in CGAL [3]. We changed the default parameters to make the results comparable to our approach: We set the inlier distance to $0.15m$

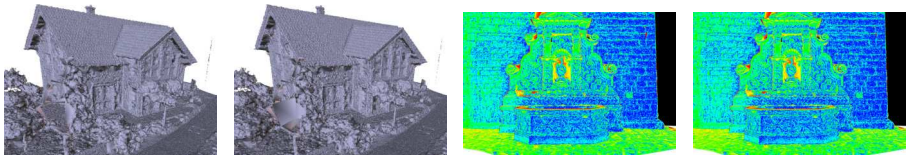


Fig. 5. *Errors of proposed visibility-based energy terms compared to the visibility-based energy in [16] (in m). Left:* Evaluation on the *Residential Area* dataset. Visually, the result of the proposed energy is very similar while having a slightly lower error (error definition see Sec. 4) and being normalized, which is crucial for combining it with other energies. *Right:* Error visualization of the Fountain-P11 [30] dataset. Blue means low error, red high error. Also on this dataset, the errors of the proposed formulation are slightly lower (error definition see [30]).

as in the line-based approach and reduced the minimum supporting points per plane to 0.5% to generate more plane hypotheses. As can be seen in the results in Fig. 4, [29] misses out important facade planes due to missing 3D points and detects several spurious planes whereas our approach detects a more complete plane set due to the availability of line structure on the facades. For additional comparisons we refer to the supplementary material.

Normalized Visibility-Based Energy Term. In this experiment, we show that the proposed normalized visibility-based energy formulation slightly improves the reconstruction accuracy while simultaneously being easier to handle in combination with other energy terms. We evaluated the proposed energy term on the *Residential Area* dataset and, additionally, on the *Fountain-P11* [30] dataset. In Fig. 5, one can see the results and error metrics with just using visibility-based energies (i.e., without additional energy terms). One can observe that the visual results are similar. For both datasets, the error metrics are very similar but slightly better for the proposed, normalized visibility-based energy formulation. For a more detailed evaluation of the individual steps changed in the energy formulation we refer the reader to the supplementary material.

When looking at the result of *Residential Area* using just visibility-based energy, it can be seen that some surfaces are very noisy. Hence, regularizing some parts with plane priors and using a smooth surface approximation for others like vegetation is very beneficial for creating visually appealing models.

Semantic Priors. In Fig. 6 it can be observed that when using no semantics (same as treating everything as building), the surroundings of buildings are noisy and less visually appealing. Also, more data needs to be used to describe the noisy mesh, while with semantics a sparser Poisson reconstruction describes the surroundings. Further, artifacts may arise as the Manhattan regularity term is applied everywhere even though it might not be suited to smooth e.g. vegetation.

Results. Below we compare reconstruction results from the proposed algorithm with generic 3D reconstruction algorithms and with specialized urban recon-

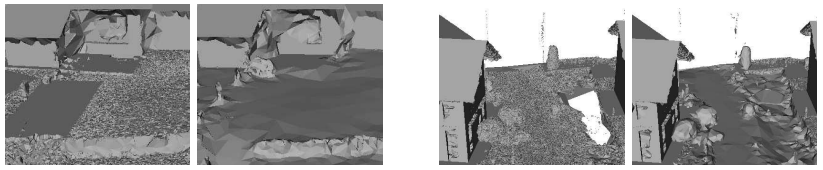


Fig. 6. *Comparison of results computed with/without semantic information.* As can be seen in the detail examples of two datasets, the reconstructed surface of the surroundings of buildings is more noisy without semantics. Due to the Poisson reconstruction and the different smoothness terms, the surroundings get reconstructed much smoother and with less points (i.e. less data) when using semantics. Also, detected planes are removed in the surroundings, as just building lines are used for plane detection. These changes due to semantics contribute to a more visually appealing final reconstruction.

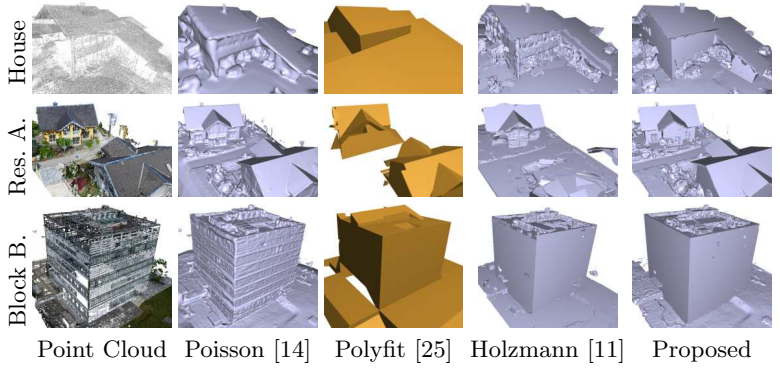


Fig. 7. *Results and comparison with state-of-the-art methods.* From left to right: Input point clouds, Poisson meshes [14] computed from the input point clouds, results of Polyfit [25], Holzmann et al. [11] and the proposed approach. The Poisson mesh looks visually appealing, but often produces rounded edges and spurious surfaces like bubbles where data is missing. Polyfit is not able to reconstruct all buildings sufficiently well. As it depends on planes detected from point clouds, undetected surfaces were just not included in the possible solution set and spurious planes lead to erroneous reconstruction results. Holzmann et al. regularizes some parts of the scene well with its included plane prior. However, at some parts of the scene not all planes were detected and, hence, planar surfaces remain noisy. Further, due to an unpredictable visibility-based energy term it is difficult to set correct weights for smoothness terms. Hence, some parts of the scene can be smoothed out very quickly. The proposed approach creates 3D models with planar surfaces at facades/roofs while still keeping the building details like chimneys and reconstructs the surroundings with a smooth surface.

struction approaches. For a comparison with commercial reconstruction pipelines we refer the reader to the supplementary material.

In Fig. 7 and Fig.1, results from the proposed approach and state-of-the-art reconstruction algorithms are depicted, where some of them also try to follow the same idea of a visually appealing 3D reconstruction. The Poisson surface

reconstruction produces smooth surfaces, which results in rounded edges. Additionally, it cannot handle missing data very well and creates spurious artifacts. Polyfit [25] heavily depends on the (point-based) plane detection result and reconstructs non-planar parts not very well, as it just relies on detected planes and uses only the plane surfaces to create an optimized surface model. The method of Holzmann et al. [11] incorporates plane priors into the reconstruction and, hence, aims to follow a similar idea of a visually appealing 3D model. However, some planar surfaces are not detected correctly and due to the unpredictability of the visibility-based energy it is hard to set the smoothness energy weights correctly. This might lead to artifacts like smoothed out wall parts or whole buildings. As this approach has no semantic class specific smoothing, also the surroundings of buildings are smoothed heavily according to a plane prior and, hence, some parts (like, e.g., vegetation) are smoothed too aggressively and artifacts looking like slices might arise. In comparison, the proposed approach uses a more complete plane set as shape prior and imposes planar surfaces just on buildings. Due to the improved visibility-based energy formulation it is easier to set correct smoothness term weights and, hence, to avoid over-smoothing. The representation of the surroundings is smooth while still not over-smoothed.

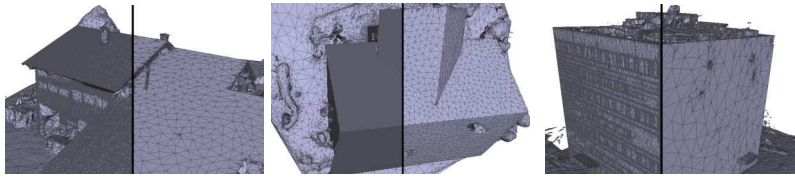
In Tab. 1 error metrics for two datasets with respect to the ground truth are depicted. It can be observed that the error metrics for the proposed approach are the best for both the *House* and *Residential Area* data set. Apparently, the plane prior incorporated within the reconstruction process handles noise and clutter better than, for example, a smooth Poisson surface. For more information about the ground truth, we refer to the supplementary material.

For this comparisons, we used the following parameter settings: For Poisson surface reconstruction we set octree depth to 9. For Polyfit we used default parameters. We also tried to vary the parameters, but the results did not improve significantly. For Holzmann et al. [11] we used the parameter settings as described in the paper for *House* and *Block B.*. For *Residential Area* we set α_{LoD} to 250K.

Due to the planarity of big parts of the resulting mesh of the proposed method, the amount of faces in the mesh can be significantly reduced without

Table 1. *Error statistics compared to the ground truth.* We computed the minimal distances of the ground truth points to the surface reconstructions with a maximum distance of 1 m. On both datasets, the proposed approach has the lowest error. Polyfit has significantly higher errors on both datasets, as it does not reconstruct the whole scene but just (parts of) buildings well. Holzmann et al. has the highest error on the *Residential Area* dataset due to a wrongly smoothed out building. Poisson produces over-smoothed (i.e., no sharp edges) results, but has comparable errors.

	House		Residential Area	
	μ [m]	σ[m]	μ [m]	σ[m]
Poisson [14]	0.165	0.237	0.101	0.157
Polyfit [25]	0.515	0.352	0.304	0.375
Holzmann et al.[11]	0.137	0.237	0.415	0.385
Proposed	**0.126**	**0.233**	**0.055**	**0.086**



House, $r = 0.154$ $\#faces = 698K/107K$ | Res. Area, $r = 0.081$ $\#faces = 4569K/371K$ | Block Building, $r = 0.054$ $\#faces = 2190K/118K$

Fig. 8. *Simplification as post-processing.* Every subfigure consists of the resulting mesh of the proposed approach (*left*) and the mesh simplified by Quadric Edge Collapse [6] in a lossless way (*right*) (i.e., restricting the edge collapse to a quadric error of 10^{-13}). Below, reduction factors and $\#faces$ before/after simplification are depicted. Due to the perfect planarity of the building parts, faces can be merged without changing the surface of the mesh. Though, non-building parts stay untouched as they are not perfectly planar. Very low reduction factors $r = \frac{\#faces_{simpl}}{\#faces_{orig}}$ (i.e., high compression) can be reached for all models. In comparison, when applying Quadric Edge Collapse on Poisson meshes in Fig. 7, the best reduction factor was 0.929 ($\#faces = 393k/365K$) at House. Quadric Edge Collapse was performed with VCGlib [31].

changing the surface. In Fig. 8, results of applying Quadric Edge Collapse [6] as post-processing step are depicted. As the maximum error of Quadric Edge Collapse was set to 10^{-13} (i.e., nearly zero), only edges on planar surfaces were removed. It can be observed that even though the mesh surface did not change, the amount of data was extremely reduced.

Assuming to have a precomputed dense point cloud, 3D line model, depth maps and semantically labeled images, our approach needs 13 min for the *House* dataset, 60 min for the *Block Building* and 153 min for the *Residential Area* (on an Intel Xeon E5-2680 running at 2.8GHz with 40 cores and 264 GB RAM). Most of the time is needed for cell cutting and visibility term computations.

5 Conclusion

We presented a 3D reconstruction approach for urban scenes, with which it is possible to get planar surfaces and straight outlines for buildings, while the surroundings of buildings are represented by a smooth surface. Our introduced line-based plane detection algorithm detects a more complete plane set compared to point-based approaches and by using semantic information we can regularize individual scene parts differently. We have shown that we can produce visually appealing and compact 3D reconstructions while still reaching slightly better accuracies compared to state-of-the-art methods.

Acknowledgements. This research was funded by the Austrian Science Fund (FWF) in the project V-MAV (I-1537). We thank Prof. Werner Lienhart and Slaven Kalenjuk from IGMS, TU Graz, Jesus Pestana and Christian Mostegel for providing datasets and Martin R. Oswald for discussion.

References

1. Blaha, M., Vogel, C., Richard, A., Wegner, J., Schindler, K., Pock, T.: Large-scale semantic 3d reconstruction: an adaptive multi-resolution model for multi-class volumetric labeling. In: Proceedings IEEE Conference Computer Vision and Pattern Recognition (2016). https://doi.org/10.1109/CVPR.2016.346
2. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(12), 1222–1239 (2001)
3. CGAL. Computational Geometry Algorithms Library: `http://www.cgal.org`
4. Dzitsiuk, M., Sturm, J., Maier, R., Ma, L., Cremers, D.: De-noising, stabilizing and completing 3D reconstructions on-the-go using plane priors. In: International Conference on Robotics and Automation (May 2017)
5. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. IEEE Transactions on Pattern Analysis and Machine Intelligence (2010)
6. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: ACM Trans. on Graphics (SIGGRAPH). pp. 209–216. ACM Press/Addison-Wesley Publishing Co., New York (1997)
7. Häne, C., Zach, C., Cohen, A., Angst, R., Pollefeys, M.: Joint 3d scene reconstruction and class segmentation. In: Proceedings IEEE Conference Computer Vision and Pattern Recognition (2013)
8. Häne, C., Heng, L., Lee, G.H., Sizov, A., Pollefeys, M.: Real-time direct dense matching on fisheye images using plane-sweeping stereo. In: International Conference on 3D Vison (3DV) (2014)
9. Hofer, M., Maurer, M., Bischof, H.: Efficient 3d scene abstraction using line segments. Computer Vision and Image Understanding (2016), `http://dx.doi.org/10.1016/j.cviu.2016.03.017`
10. Holzmann, T., Fraundorfer, F., Bischof, H.: Regularized 3d modeling from noisy building reconstructions. In: Fourth International Conference on 3D Vision, 3DV 2016, Stanford, CA, USA, October 25-28, 2016. pp. 528–536 (2016)
11. Holzmann, T., Oswald, M.R., Pollefeys, M., Fraundorfer, F., Bischof, H.: Plane-based surface regularization for urban 3d reconstruction. In: 28th British Machine Vision Conference. vol. 28 (9 2017)
12. Hoppe, C., Klopschitz, M., Donoser, M., Bischof, H.: Incremental surface extraction from sparse structure-from-motion point clouds. In: Proceedings British Machine Vision Conference (2013)
13. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
14. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Eurographics Symposium on Geometry Processing (2006)
15. Korč, F., Förstner, W.: eTRIMS Image Database for interpreting images of man-made scenes. Tech. Rep. TR-IGG-P-2009-01 (April 2009), `http://www.ipb.uni-bonn.de/projects/etrims_db/`
16. Labatut, P., Pons, J.P., Keriven, R.: Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In: Proceedings International Conference on Computer Vision (2007)
17. Labatut, P., Pons, J.P., Keriven, R.: Hierarchical shape-based surface reconstruction for dense multi-view stereo. In: International Workshop on 3-D Digital Imaging and Modeling (3DIM), ICCV Workshops. pp. 1598–1605. Kyoto, Japan (Oct 2009)

18. Lafarge, F., Alliez, P.: Surface reconstruction through point set structuring. Comput. Graph. Forum **32**(2), 225–234 (2013)
19. Lafarge, F., Keriven, R., Brédif, M., Vu, H.: A hybrid multiview stereo algorithm for modeling urban scenes. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 5–17 (2013)
20. Li, M., Wonka, P., Nan, L.: Manhattan-world urban reconstruction from point clouds. In: Proceedings European Conference on Computer Vision (2016)
21. Li, Y., Wu, X., Chrysanthou, Y., Sharf, A., Cohen-Or, D., Mitra, N.J.: Globfit: consistently fitting primitives by discovering global relations. ACM Trans. Graph. **30**(4), 52:1–52:12 (2011)
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)
23. Maurer, M., Hofer, M., Fraundorfer, F., Bischof, H.: Automated inspection of power line corridors to measure vegetation undercut using uav-based images. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (2017)
24. Monszpart, A., Mellado, N., Brostow, G., Mitra, N.: RAPter: Rebuilding man-made scenes with regular arrangements of planes. ACM SIGGRAPH 2015 (2015)
25. Nan, L., Wonka, P.: Polyfit: Polygonal surface reconstruction from point clouds. In: Proceedings International Conference on Computer Vision (2017)
26. Oesau, S., Lafarge, F., Alliez, P.: Planar shape detection and regularization in tandem. Comput. Graph. Forum **35**(1), 203–215 (2016)
27. PASCAL-Context network: `http://dl.caffe.berkeleyvision.org/pascalcontext-fcn32s-heavy.caffemodel`
28. Rothermel, M., Wenzel, K., Fritsch, D., Haala, N.: Sure: Photogrammetric surface reconstruction from imagery. In: Proceedings LC3D Workshop, Berlin (2012)
29. Schnabel, R., Wahl, R., Klein, R.: Efficient ransac for point-cloud shape detection. Computer Graphics Forum **26**(2), 214–226 (Jun 2007)
30. Strecha, C., von Hansen, W., Gool, L.V., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: Proceedings IEEE Conference Computer Vision and Pattern Recognition (2008)
31. The VCG Library: `http://vcg.isti.cnr.it/vcglib/`
32. Vogel, C., Richard, A., Pock, T., Schindler, K.: Semantic 3d reconstruction with finite element bases. In: 28th British Machine Vision Conference. vol. 28 (9 2017)
33. Waechter, M., Moehrle, N., Goessele, M.: Let there be color! - large-scale texturing of 3d reconstructions. In: Proceedings European Conference on Computer Vision (2014)
34. Xiao, J., Furukawa, Y.: Reconstructing the world's museums. International Journal of Computer Vision **110**(3), 243–258 (2014)
35. Zebedin, L., Bauer, J., Karner, K.F., Bischof, H.: Fusion of feature- and area-based information for urban buildings modeling from aerial imagery. In: Proceedings European Conference on Computer Vision (2008)
36. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.S.: Conditional Random Fields as Recurrent Neural Networks. In: IEEE Computer Vision and Pattern Recognition (CVPR). pp. 1529–1537 (2015). https://doi.org/10.1109/ICCV.2015.179, arXiv: 1502.03240