Finding Good Configurations of Planar Primitives in Unorganized Point Clouds

Mulin Yu, Florent Lafarge

Finding Good Configurations of Planar Primitives in Unorganized Point Clouds

Mulin Yu Florent Lafarge
Inria - Université Côte d'Azur
firstname.lastname@inria.fr

Abstract

We present an algorithm for detecting planar primitives from unorganized 3D point clouds. Departing from an initial configuration, the algorithm refines both the continuous plane parameters and the discrete assignment of input points to them by seeking high fidelity, high simplicity and high completeness. Our key contribution relies upon the design of an exploration mechanism guided by a multi-objective energy function. The transitions within the large solution space are handled by five geometric operators that create, remove and modify primitives. We demonstrate the potential of our method on a variety of scenes, from organic shapes to man-made objects, and sensors, from multiview stereo to laser. We show its efficacy with respect to existing primitive fitting approaches and illustrate its applicative interest in compact mesh reconstruction, when combined with a plane assembly method.

1. Introduction

Geometric primitives are popular representations for processing massive 3D data. Such parametric shapes offer a more compact yet meaningful description than raw point clouds or dense meshes [16]. Among the most common types of primitives, planes are particularly important due to their relevance to man-made environments, e.g. urban and indoor scenes. They have been successfully used in many data processing tasks such as data registration [22, 55], object recognition [40], simultaneous localization and mapping [15, 17], structure from motion [39, 54], urban modeling [12, 56] and surface reconstruction [1, 7].

Fitting planar primitives to 3D point clouds is a long-standing problem in computer vision. This is commonly formulated as a clustering operation: input points are grouped into planar components under a user-specified fitting tolerance. Solving this problem is, however, not trivial as (i) the number of primitives is unknown, and (ii) some input points, called *outliers*, are associated with no primitive as they do not fall in the fitting tolerance zones of the primitives. Finding a *good configuration* of planar primitives is somehow arbitrary and turns out to be a tradeoff between three objectives:

1. Fidelity: the distance between a primitive and its associated input points, called *inliers*, must be small,
2. Simplicity: the number of primitives must be small,
3. Completeness: the ratio of inliers must be high.

Unfortunately, existing methods are not designed to explicitly control these three objectives simultaneously. They usually perform with one, sometimes two, objectives through either incremental mechanisms [30, 37, 42], multi-labeling energy minimization models [14, 28, 33] or neural networks [21, 26, 43] that cannot fully explore the large configuration space of this problem.

We propose an algorithm for fitting planar primitives to unorganized point clouds by seeking high fidelity, high simplicity and high completeness altogether. Our key contribution relies on the design and efficient implementation of an exploration mechanism that refines an initial configuration by minimizing a multi-objective energy function. The transitions within the large solution space are handled by five geometric operators that create, remove and modify primitives and re-assign inliers and outliers. The exploration works in tandem by alternating variational optimization at the global scale and a priority queue that sorts the local operations likely to decrease the energy. Optionally, our method offers the guarantee for not degrading the fidelity, simplicity and completeness of the initial configuration.

We show the potential of our method on different types of scenes, from organic to man-made, as well as on multiple sensors such as laser scanning and multiview stereo (MVS). We also demonstrate its efficacy with respect to existing approaches and its benefits for reconstructing compact meshes, when combined with a plane assembly method.

2. Related work

We distinguish four families of methods for fitting planar primitives to unorganized 3D point clouds. Note that most of these methods can also detect quadric surface primitives.

Incremental mechanisms. RANSAC and Region Growing are two widely used mechanisms that detect primitives

in an iterative manner. The former [38,42,48] samples many plane hypotheses and verifies them against the input data. The plane hypothesis with the largest number of inliers is then kept as a primitive. The latter [27, 37, 49] operates by growing a local plane hypothesis in a spatial neighborhood of a seed point. Voting schemes in discretized spaces of the primitive parameters [3,6,10,45] are also a popular strategy when the input points are accurately oriented. While Gaussian sphere mapping is the traditional choice for planes and cylinders [36], more complex parameter spaces can also be used for fitting any type of quadrics [2]. These fast and scalable mechanisms constitute the de facto solutions from real-world data with efficient implementations in popular geometry processing libraries such as CGAL [31]. However, they do not offer a good control on the output quality, leading often to overly complex plane configurations.

Energy-based models. Variational shape approximation [8, 50] minimizes an approximation error between the input data and a set of primitives using Lloyd's clustering algorithm [25]. This approach can be combined with sequences of splitting and merging of primitives so that the number of primitives has not to be constant and fixed a priori by the user [44]. Input data are however supposed to be free of outliers, leading to good results on synthetic data only. Another popular strategy consists in detecting a set of plane hypotheses before assigning one of them to each input point using a multi-labeling energy minimization [14, 32, 33]. While a priori knowledge on label smoothness, output complexity or geometric regularity can be encoded easily, such discrete formulations require good plane hypotheses. This can rarely be guaranteed in practice with existing incremental mechanisms, even when plane hypotheses are enriched with geometric and spatial priors [28]. In contrast, we formulate an energy in a mixed discrete-and-continuous configuration space where point assignment and estimation of plane parameters are operated jointly.

Neural network architectures. Deep learning methods have recently emerged as a promising solution to the tedious parameter tuning problem of traditional algorithms. The end-to-end networks SPFN [21] and ParSeNet [43] predict per-point properties using off-the-shelf architectures, typically Pointnet++ [34], before estimating the primitive types and parameters. CPFN [26] proposes an adaptive patch sampling network to assemble primitive detection results at coarse and fine scales. HPNet [51] extracts primitives using a mean-shift clustering operating on a combination of three learned features with adaptive weights. These networks require high computing resources and can only handle point clouds of a hundred thousand points at best. PrimitiveNet [13] offers a solution to this scalability issue with a region growing mechanism operated from the output of two networks, one producing per-point high dimensional features and the other predicting whether two neighboring points belong to the same primitive. This solution, however, requires defect-free, dense meshes as input. In practice, these learning methods do not generalize well on real-world data as training sets are composed of synthetic point clouds sampled from datasets of Computer-Aided Design (CAD) models [19, 41, 53]. Note that some neural networks also detect planes from single image [24, 35, 52] or depth maps [57].

Methods with regularity enforcement. Some methods are designed to enforce geometric regularities between planes such as parallelism, orthogonality or some types of symmetry. This can be done by alternating fitting and regularization of primitives within the traditional incremental mechanisms, i.e. Region Growing [30] and RANSAC [23], or by inserting pairwise priors in energy-based models [28]. Mutually orthogonal planes that respect the Manhattan-World assumption [9] can also be fitted efficiently by Gaussian sphere mapping [46]. Some methods [11, 20] go further by analyzing the structure of objects at different key abstraction levels so that no user-specified fitting tolerance is required. In practice, the assumptions for such geometric and structural regularities are relevant for specific application domains only. Our algorithm is generic and does not rely upon such assumptions.

3. Algorithm

Our algorithm takes as input an unorganized 3D point cloud which is typically generated from MVS, laser scanning, depth cameras, or directly sampled from CAD models. The precision of the output planar primitives is controlled by the two key parameters of traditional primitive fitting algorithms: (i) a fitting tolerance, denoted by ϵ, that specifies the maximal distance of an inlier to its planar primitive, and (ii) a minimal primitive size, denoted by σ, that allows primitives with a too low number of inliers to be discarded. Note that specifying these two parameters is relatively intuitive when data are defined in a meaningful system of units. It also requires less efforts than creating training sets for learning methods.

The output primitives are clusters of inlier points each associated with a supporting plane, i.e. the best least square fitting plane to its inlier points. We denote such a configuration of planar primitives by $\mathbf{x} = (\mathbf{p}, \mathbf{l})$ where $\mathbf{p}$ is a set of supporting planes parametrized in the continuous domain, and $\mathbf{l}$ is a configuration of discrete labels that indicates whether input points are outliers or inliers to one of the supporting planes of $\mathbf{p}$.

3.1. Energy formulation

We measure the quality of a primitive configuration $\mathbf{x}$ with an energy U of the form

$$U(\mathbf{x}) = \omega_f U_f(\mathbf{x}) + \omega_s U_s(\mathbf{x}) + \omega_c U_c(\mathbf{x}) \qquad (1)$$

where terms U_f, U_s and U_c defined in the interval $[0, 1]$ account for our objectives of fidelity, simplicity and completeness respectively, and ω_f, ω_s and ω_c are positive weights balancing the three terms such that $\omega_f + \omega_s + \omega_c = 1$.

Fidelity term. U_f measures the mean distance error between planar primitives and their associated inliers

$$U_f(\mathbf{x}) = \frac{1}{n_\mathbf{x}} \sum_{p \in \mathbf{p}} \sum_{i \in p} d_\epsilon(i, p) \tag{2}$$

where $n_\mathbf{x}$ is the total number of inliers in the configuration $\mathbf{x}$, and $d_\epsilon(i, p)$ represents the Euclidean distance between the inlier point i and the supporting plane p normalized by the fitting tolerance ϵ. Note that other metrics can be considered such as the normal deviation between the normal of the point and the orthogonal vector of the plane. This option is discussed in Supplementary Material.

Simplicity term. U_s encourages configurations with a low number of primitives

$$U_s(\mathbf{x}) = \frac{|\mathbf{p}|}{n_\sigma} \tag{3}$$

where $|\mathbf{p}|$ denotes the number of primitives of the configuration $\mathbf{x}$, and n_σ is the maximal number of detectable primitives computed as the ratio of the number of input points n to the minimal number of inliers per primitive σ.

Completeness term. U_c favors configurations with a high ratio of inliers

$$U_c(\mathbf{x}) = 1 - \frac{n_\mathbf{x}}{n} \tag{4}$$

Energy U is formulated in a simple and natural way regarding our three initial objectives. Figure 1 illustrates the impact of the weights. Note that none of the three configurations can be considered as better than the others as they each perform best on one of the objectives. In our experiments, we give the same importance to each objective.

3.2. Exploration mechanism

Both continuous variables for parametrizing the supporting planes and discrete labels for grouping input points are involved in the minimization of the non-convex energy U. In order to explore efficiently such a large solution space, we propose an iterative mechanism inspired by mesh decimation techniques [4]. Starting from an initial configuration $\mathbf{x}_0$, the idea consists in computing the energy variations induced by *local geometric operators* that create, remove or modify the primitives and re-assign the inliers and outliers. By sorting the energy variations of all possible operations in ascending order in a *priority queue*, we rank the operations that improve best the quality of the configuration. The algorithm then performs the geometric operation on top of the priority queue, updates the queue, and iterates until the energy does not decrease anymore.

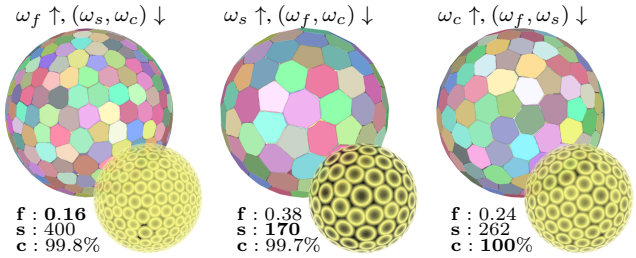


Figure 1. Tradeoff between energy terms. Putting weight on ω_f favors a low geometric error between planar primitives (see colored polygons respresenting their α-shapes) and input data, here 100K points uniformly sampled on a sphere (left). Increasing ω_s reduces the number of primitives (middle). A high value of ω_c produces configurations with few outliers (right). **f**, **s**, and **c** correspond to the mean Euclidean distance of inliers to their associated supporting plane normalized by ϵ, the number of primitives, and the percentage of inliers respectively. The colored point clouds show the geometric error distribution (yellow=0, black$\geq \epsilon$)

Local geometric operators. Figure 2 illustrates the five types of operators used for visiting the configuration space. The operators are local and only affect one or two primitives. This condition is important to guarantee a fast computation and sorting of the energy variations. To do this, we define a notion of spatial proximity between input points and between primitives. Two points are considered as adjacent if they are connected in the k-nearest neighbor graph of the input point cloud. Two primitives are adjacent if at least a pair of their respective inlier points are adjacent.

The **transfer operator** exchanges inliers between two adjacent primitives and refines their supporting planes. This operation is performed using the popular K-means algorithm (with K=2) from the two inlier sets. In particular, we use the same metrics d_ϵ from point to plane as in Equation (2), and update the cluster centroids with the best least square fitting plane of the cluster of points. In order to keep primitives compact, we only transfer inliers located where the two primitives meet, i.e. the ones adjacent to inliers of the other primitive.

The **exclusion operator** reassigns the inliers far away from their supporting plane to outliers. This operation is performed by (i) sorting in descending order the distance of all the inliers to their supporting plane, and (ii) changing the k first inliers of the sorting list to outliers, k being fixed to ten in our experiments.

The **insertion operator** reassigns outlier points to the inliers of a primitive. We first list the outlier candidates whose distance to the supporting plane is smaller than the distance to the supporting plane of any other primitive while being smaller than the fitting tolerance ϵ. We then sort in ascending order the distance of these candidates to the supporting plane, and insert the k first outliers to the set of inliers, k being fixed to ten in our experiments.

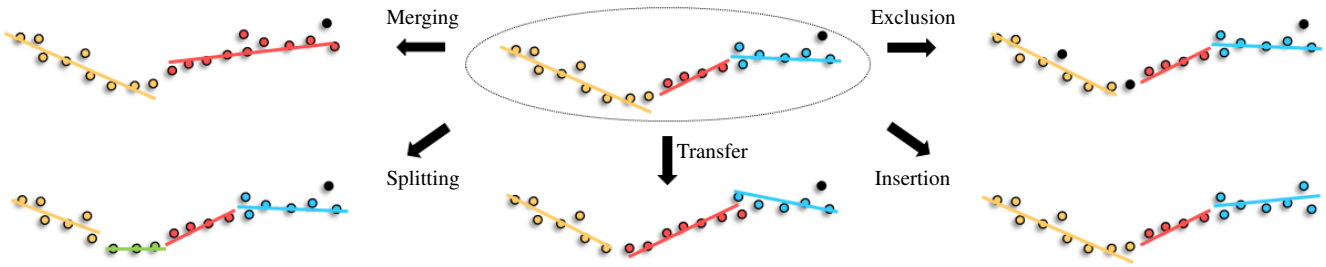
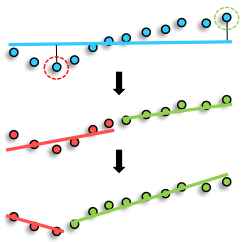


Figure 2. Local geometric operators. Five types of modifications can be operated from a configuration of planar primitives (top middle). The merge of two primitives (top left) and the split of a primitive (bottom left) alter the number of primitives in the configuration. The transfer of inlier points from a primitive to another (bottom middle) allows the refinement of supporting planes. Finally, the insertion of outliers as inliers of a primitive (bottom right) and the exclusion of inlier points of a primitive to outliers (top right) modify the completeness of the configuration. The inliers and supporting plane of each primitive are represented by colored points and a line-segment while outliers are displayed by black points.

The **merging operator** merges two adjacent primitives by reassigning their inliers to a new primitive or as outliers if located at a distance higher than ϵ. The supporting plane of the latter is then computed as the best least square fitting plane of its inliers.

The **splitting operator** divides a primitive into two new ones. This operator first identifies the farthest inlier on each side of the supporting plane, as illustrated by the dashed circles on the top part of the inset. The other inliers are then associated with one of these two farthest points by spatial proximity, leading to the creation of two new primitives (see red and green pairs of points and line-segment in the middle). Finally, the transfer operator is performed on the two primitives in order to refine the assignment of inliers and the supporting planes, as shown on the bottom part of the inset.



These five operators have complementary roles in the exploration. The transfer operator seeks a higher fidelity without altering simplicity and completeness. The merging and splitting operators aim at exploring configurations with different complexity whereas the exclusion and insertion operators have direct impact on completeness.

Priority queue. After each modification of the current configuration, the priority queue is updated. The concerned operation and all the operations with primitives impacted by the modification are first removed from the queue. The energy variations of all possible operations affecting the modified primitives are then computed and inserted in the queue.

Initialization. The exploration mechanism requires a good initial configuration as it finds a local minimum. As illustrated in Figure 3, starting from initial configurations with an already good fidelity, simplicity and completeness helps the exploration to reach better configurations. In our experiments, we use Region Growing [37] on defect-free data and RANSAC [42] on defect-laden data.

Stopping condition. The exploration mechanism stops when no more energy variation sorted in the priority queue is negative, i.e., when no operation makes the energy decrease. This condition guarantees the exploration mechanism to converge quickly without bumping effects.

Algorithm 1 Pseudo-code of the exploration mechanism

```
Initialize the primitive configuration x
repeat
    Initialize the priority queue Q
    while top operation i of Q decreases energy U do
        Update x by operation i
        Update Q
    end while
    Update x by the global transfer operator
until no update modifies x any more
```

Details for accelerating the exploration. The transfer operator improves one objective, i.e. fidelity, without altering the other two: this particular feature makes it being called a high number of times in a priority queue. However, these local refinements between any pairs of adjacent primitives are likely undone later by the other four operators, leading to slow converge in practice. To speed up the exploration, we prefer using the transfer operator outside the priority queue in a global manner, i.e. by transferring inliers between all the pairs of adjacent primitives simultaneously. The exploration then alternates between series of priority queue updates where only splitting, merging, exclusion and insertion operations are considered, and global transfer of inliers by K-means with K fixed to be the number of primitives. Because the global transfer operator cannot degrade fidelity and modify the simplicity and completeness, the exploration cannot loop infinitely between priority queue and

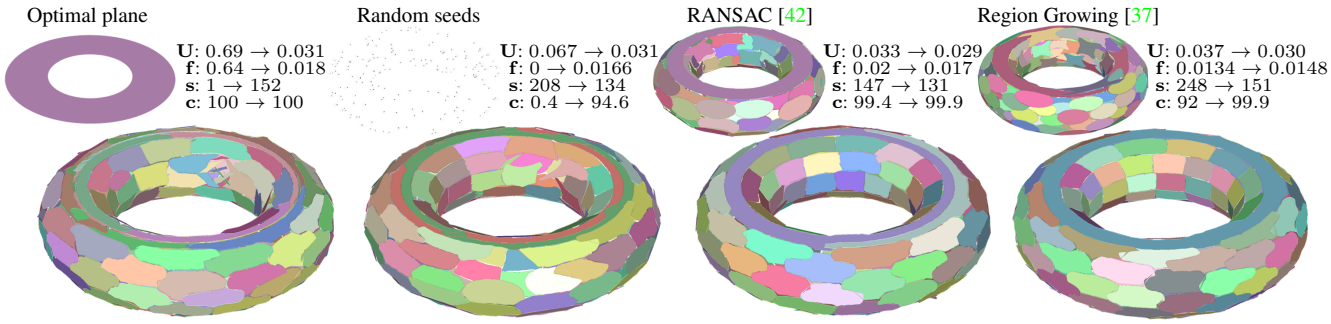


Figure 3. Initialization. Top row shows different initial configurations from 100K input points uniformly sampled on a torus, while bottom row shows results obtained after exploration. Starting the exploration from a good initial configuration given by incremental mechanisms such as RANSAC or Region Growing allows us to reach better configurations than from the optimal plane fitted to the input points or from many small primitives randomly distributed from input data. **U** refers to the energy of configurations.

global transfer. Algorithm 1 describes the pseudo-code of our exploration scheme.

As illustrated in Figure 4, this exploration in tandem reaches similar energies as the original scheme while being one order of magnitude faster. It is also an efficient solution against a non-local simulated annealing that is three orders of magnitude slower for final configurations of identical quality.

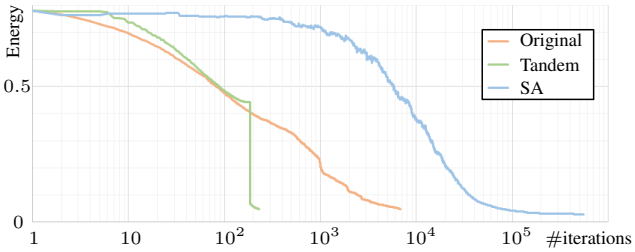


Figure 4. Evolution of energy U during exploration mechanisms. Our exploration in tandem (green curve) converges quickly whereas the original scheme without global transfer interruptions (orange curve) and a simulated annealing optimization (SA, blue curve) require respectively one and three orders of magnitude more iterations for reaching similar energies.

Optional constraints. We can optionally impose that the final configuration does not degrade the fidelity, simplicity and completeness of the initial configuration. This can be done by discarding from the priority queue all the operations that lead to lower fidelity, simplicity and completeness than those of the initial configuration. In this case, the energy weights can be fixed proportionally to the initial configuration, i.e. by taking $\omega_f = K^{-1}U_f(\mathbf{x_0})^{-1}$, $\omega_s = K^{-1}U_s(\mathbf{x_0})^{-1}$ and $\omega_c = K^{-1}U_c(\mathbf{x_0})^{-1}$ where $K = U_f(\mathbf{x_0})^{-1} + U_s(\mathbf{x_0})^{-1} + U_c(\mathbf{x_0})^{-1}$. This option offers the user the guarantee not to score lower than the initial configuration on each of the three objectives, but it typically reduces the overall quality of the reached solution.

4. Experiments

Our algorithm has been implemented in C++ using the Computational Geometry Algorithms Library (CGAL). In our experiments, we typically set the fitting tolerance ϵ to 0.5% of the bounding box diagonal and the minimal shape size σ from 0.001% to 1% of the number of input points, depending on the complexity of scenes. The values of ϵ and σ used to produce the presented results are provided in Supplementary Material.

Flexibility and robustness. As shown in Figures 5 and 7, we tested the algorithm on various scenes and objects ranging from indoor and urban environments to statues through furniture elements. This good flexibility originates from the absence of domain-specific geometric priors in our method. We also performed tests on input data generated from different types of acquisition systems including Laser and MVS. Our algorithm offers good robustness on these data, even on noisy MVS point clouds where our insertion and exclusion operators allow an efficient selection of inliers and outliers.

Comparisons. We compared our algorithm with the traditional methods Region Growing [37], its seeding variant [31] and RANSAC [42], as well as with the deep learning methods SPFN [21], ParSeNet [43] and HPNet [51]. We measure fidelity as the mean Euclidean distance from inliers to primitives normalized by the longest side of the bounding box, as proposed by [21]. Simplicity and completeness are measured as the number of primitives and the ratio of inliers respectively.

We first compared our algorithm with the traditional incremental methods by using plane as the only primitive type. We tested on a sample of 5,000 models randomly chosen from the ABC dataset [19] (each CAD model was sampled with 100K points) and on the 42 real-world models of the KSR dataset [1] partly based on Tanks and Temples [18]. Table 1 shows that our algorithm outperforms

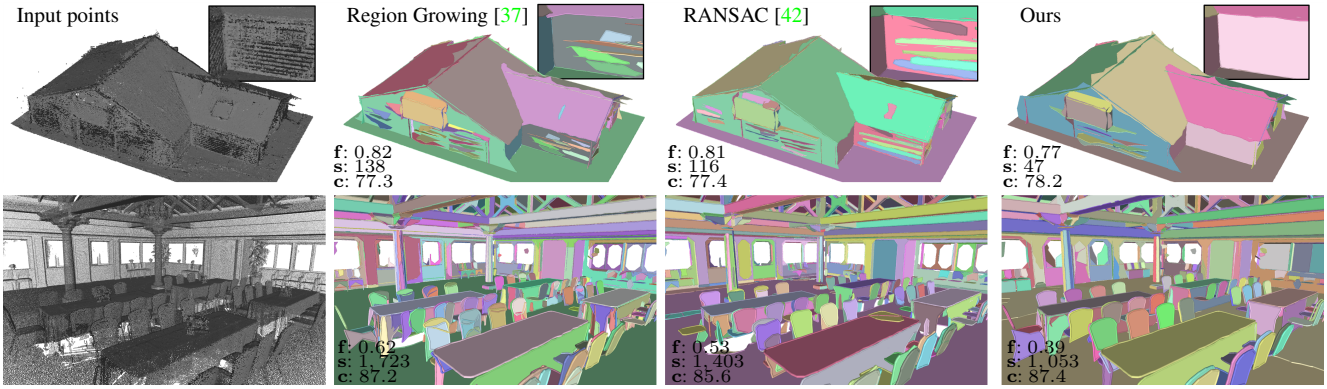


Figure 5. Visual comparisons with incremental mechanisms. Region Growing and RANSAC often produce inaccurate planar primitives from real-world acquisition data with, for instance, over-detected primitives on the noisy facade (top, aerial MVS) and on thin furniture elements such as chairs (bottom, indoor Laser). Our algorithm performs better by detecting fewer yet more meningful primitives without sacrificing fidelity and completeness. **f**, **s**, and **c** refer to the fidelity, simplicity and completeness scores respectively. Models from [18].

these methods on the three objectives by a large margin on the KSR dataset, and to a lesser extent, on the ABC dataset. The gain is higher from real-world data where traditional methods often produce inaccurate and overly complex configurations, as illustrated in Figure 5. In contrast, our algorithm can handle efficiently data defects, in particular by merging and splitting primitives in case of over- and under-detection, and by selecting inliers and outliers efficiently.

		Fidelity ($\times 10^2$)	Compl.	Simpl.
KSR	RG [37]	0.39	83.6	654.2
	SRG [31]	0.43	83.9	612.7
	RANSAC [42]	0.42	83.3	684.7
	Ours	**0.33**	**84.1**	**572.4**
ABC	RG [37]	0.28	97.6	69
	SRG [31]	0.30	97.1	65
	RANSAC [42]	0.21	97.7	55.5
	Ours	**0.19**	**97.9**	**39.4**

Table 1. Comparison with traditional methods. Our algorithm achieves better scores of fidelity, simplicity and completeness on both the real-world KSR and CAD-sampled ABC datasets.

		Fidelity ($\times 10^2$)	Compl.	Simpl.
ABC*	SPFN [21]	2.835	90.0	12.2
	ParSeNet [43]	0.410	99.1	8.8
	HPNet [51]	0.224	96.8	**8.3**
	Ours	**0.130**	**99.8**	**8.3**
ANSI*	SPFN [21]	0.760	**95.5**	12.1
	ParSeNet [43]	1.064	91.0	**9.0**
	HPNet [51]	0.087	83.0	13.3
	Ours	**0.085**	**95.5**	9.7

Table 2. Comparison with deep learning methods. The ANSI* and ABC* datasets are composed of piecewise planar models only.

We also compared with the learning methods for smaller point clouds, i.e. 10K points or less. Because these methods also detect quadrics, we tested on a collection of 653 and 132 purely piecewise planar models from the ABC [19] and ANSI [21] datasets respectively. SPFN was trained on the ANSI dataset as detailed in [21], whereas ParSeNet and HPNet were trained on the ABCParts dataset [43] generated from the ABC dataset. We used the end-to-end model of SPFN to predict both point assignment and plane parameters. For ParSeNet and HPNet, we combined their point assignment predictions with least square fittings for estimating plane parameters. As shown in Table 2, our algorithm competes well on the three metrics. The gain is particularly high on fidelity as the inlier-to-plane error is not directly controlled by a fitting tolerance parameter in learning methods. Deep learning methods also suffer from a low generalization from one dataset to the other, in particular in terms of completeness and simplicity. Only ParSeNet provides a better simplicity score than our method on the ANSI dataset,

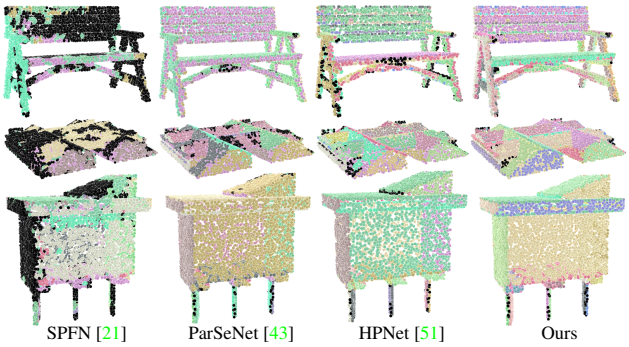


Figure 6. Visual comparison with learning methods. Input points are colored per primitive cluster. Trained from another dataset, SPFN does not generalize well and exhibits many outliers (black points). ParSeNet and HPNET perform better but remain affected by frequent local mislabeling, in contrast to our method. Models from the ABC dataset.

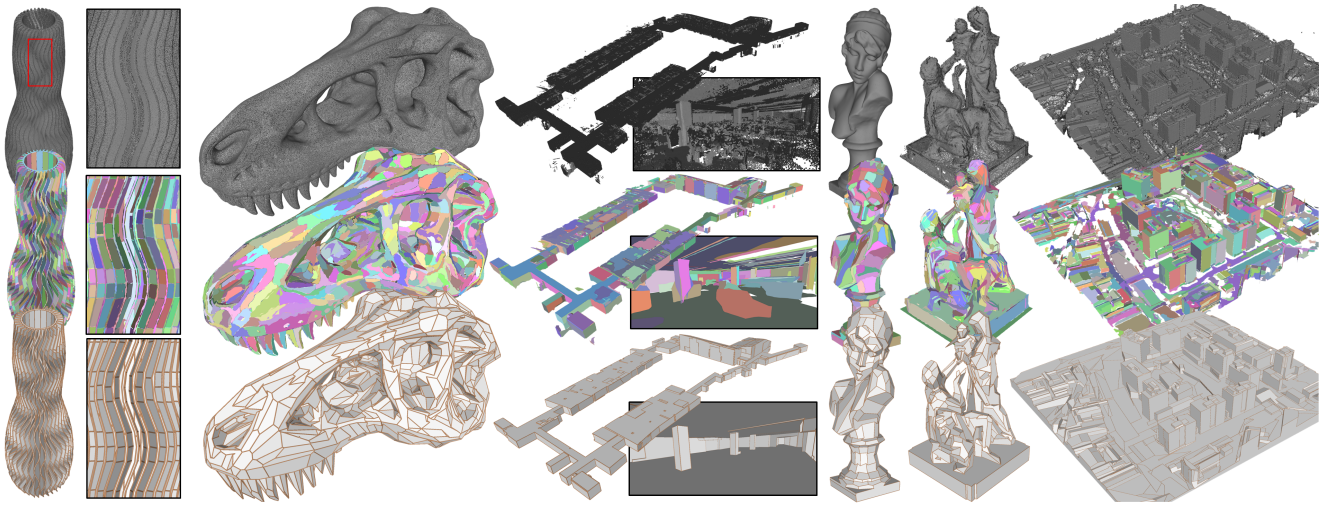
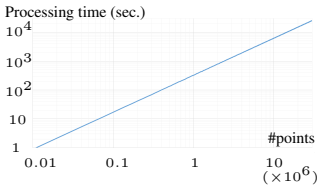


Figure 7. Compact mesh reconstruction and approximation on various scenes. Planar primitives detected by our algorithm (middle row) from input point clouds (top row) are assembled into compact, watertight, intersection-free polygonal surface meshes (bottom row). Such a generic and scalable pipeline produces accurate results on a variety of scenes and sensors (from left to right: CAD-sampled vase, laser scanned Trex skull, laser scanned indoor scene, laser scanned statue, MVS statue and MVS urban scene).

but it scores low on fidelity and completeness. Figure 6 shows a visual comparison on piecewise planar objects.

Performance. We tested our algorithm on data ranging from 8K points to 30M points. It offers a good scalability thanks to a low memory consumption. The latter results from the design of the geometric operators which are purely local. As shown in the inset, our algorithm typically requires a few minutes for processing several millions of input points on a standard computer with a processor clocked at 2.9Ghz. Processing time of our sequential implementation of the algorithm is reasonable but remains higher than incremental mechanisms.



Application to compact mesh reconstruction. We applied our algorithm to the compact mesh reconstruction problem. Starting from an oriented point cloud, we used a plane assembly method [1] to produce a watertight, intersection-free polygonal surface mesh from the planar primitives fitted by our algorithm. Such a pipeline allows both the reconstruction of piecewise planar structures and the approximation of freeform objects.

Figure 8 illustrates the benefits of using our algorithm instead of Region Growing in the plane assembly method of [1]. Output meshes are both more accurate and more compact thanks to fewer yet more meaningful planar primitives. In particular, the shape of output polygonal facets adapts well to the local surface geometry, in coherence with the predictions of the Dupin indicatrix [47].

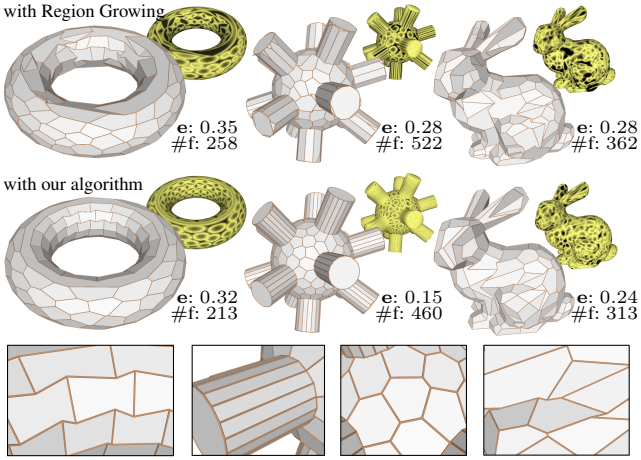


Figure 8. Reconstruction of freeform objects. Using our algorithm instead of Region Growing favors the assembling of planes into more accurate yet more compact polygon meshes. In particular, the shape of polygonal facets adapts adequately to the local surface geometry with concave hexagons when hyperbolic, rectangles when parabolic and convex hexagons when spherical and elliptic (closeups, from left to right). e and #f refer to the mean Hausdorff distance of input points to output mesh normalized by ϵ and the number of polygonal facets respectively. The colored point clouds show the Hausdorff distance distribution (yellow=0, black$\geq \epsilon$)

We tested the pipeline on a variety of scenes and sensors, as illustrated in Figure 7. It offers a good robustness to noise, outliers and, to a lesser extent, occlusions when planar components are not entirely missing in the data. The pipeline also offers a good scalability by digesting millions of points and thousands of planar primitives, as shown with the large-scale yet highly detailed results in Figure 7.

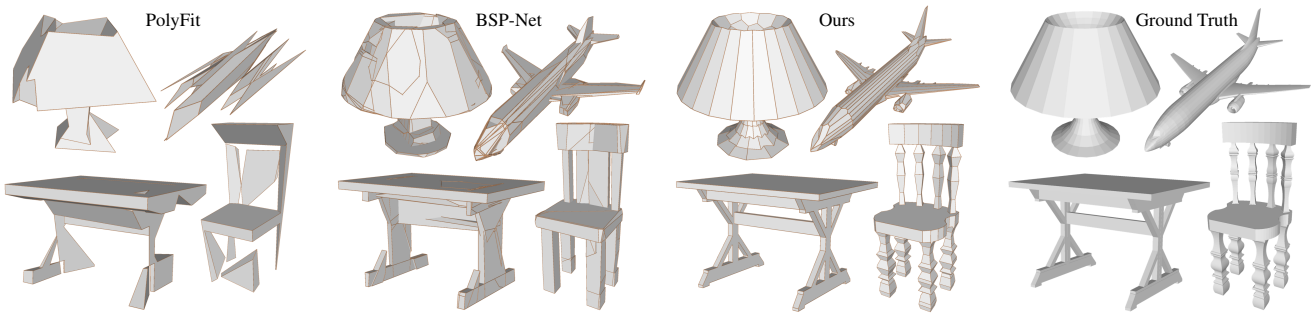


Figure 9. Visual comparison of compact mesh reconstruction methods. The low scalability of Polyfit [29] and BSP-Net [7] does not allow objects to be described with many planar components. In contrast, our pipeline can capture fine details such as the ornaments on the chair feet with highly compact meshes. Models from ShapeNet [5].

We compared our pipeline with the specialized compact mesh reconstruction methods PolyFit [29] and BSP-Net [7]. We measured the accuracy on different object categories from ShapeNet [5], the dataset used to train BSP-Net. As shown in Table 3, the accuracy scores of our pipeline are significantly higher on each object category. This gap comes from the low scalability of Polyfit and BSP-Net, the former being limited to the assembly of 50 planes at best and the latter imposing a volume discretization of objects by 64^3. As shown in Figure 9, the better scalability of our pipeline allows the capture of fine details with highly compact meshes. Note that Polyfit and our pipeline offer the guarantee to produce watertight, intersection-free meshes, in contrast to BSP-Net that outputs a soup of convex polytopes likely to intersect.

	Airplane	Car	Chair	Lamp	Table
Polyfit [29]	6.79	1.54	1.84	5.21	2.09
BSP-Net [7]	1.49	1.74	2.05	3.25	1.69
Ours	**0.34**	**0.38**	**0.40**	**0.34**	**0.33**

Table 3. Accuracy evaluation of compact mesh reconstruction methods. Accuracy is measured as the mean symmetric Hausdorff distance between input points and output meshes (values are normalized by the bounding box diagonal and multiplied by a factor of 10^2) on five object categories from the ShapeNet Dataset.

Limitations. Because the exploration mechanism is local and energy U is not convex, the quality of the initial configuration influences results. Starting from Region Growing or RANSAC is a fast and scalable solution, but it might not be an optimal choice on data highly corrupted by occlusions. That said, our algorithm will also benefit from future advances in the field to produce even better results. Processing time on massive point clouds also remains high in comparison with the traditional incremental mechanisms. This can be penalizing when planar primitives must be quickly detected as a preprocessing step for a 3D vision problem. Finally, our algorithm is not designed to preserve geometric regularities. For example, the top and bottom sides of the torus in Figure 8 do not have exactly symmetric layouts of planar primitives. Taking into account such knowledge in our framework could be done by reformulating the simplicity term. However, this would require a preliminary detection of regularities, which is not a trivial task in practice.

5. Conclusion

We proposed an algorithm for fitting planar primitives to unorganized point clouds. The key contribution of our work relies upon the design and efficient implementation of an exploration mechanism that seeks configurations with high fidelity, high simplicity and high completeness simultaneously. Inspired by geometry processing techniques, this mechanism delivers high quality results that outclass those obtained with traditional methods and recent deep learning models. We also demonstrated the efficiency and robustness of our algorithm on a variety of objects and scenes in terms of size, complexity and acquisition characteristics. We finally showed its applicability on the difficult problem of compact mesh reconstruction.

As future work, we would like to improve performances on large scenes in terms of processing time without sacrificing the memory efficiency of our exploration mechanism. One solution could be to perform multiple local geometric operations simultaneously at different locations. The design of such a parallelization scheme is however a challenging problem with our non-Markovian energy. We also plan to investigate the design of new geometric operators to both accelerate the exploration and reduce the dependence on initialization.

Acknowledgments. This work was partially supported by CSTB. We thank Sven Oesau and Bruno Hilaire (CSTB) for technical discussions, and Yanfeng Zhang (Huawei Technologies) for providing the indoor and urban scene datasets of Figure 7.

References

[1] Jean-Philippe Bauchet and Florent Lafarge. Kinetic shape reconstruction. *Trans. on Graphics*, 39(5), 2020. 1, 5, 7

[2] Tolga Birdal, Benjamin Busam, Nassir Navab, Slobodan Ilic, and Peter Sturm. Generic primitive detection in point clouds using novel minimal quadric fits. *TPAMI*, 42(6), 2020. 2

[3] Dorit Borrmann, Jan Elseberg, Kai Lingemann, and Andreas Nüchter. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):3, 2011. 2

[4] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon Mesh Processing*. AK Peters, 2010. 3

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012. 8

[6] Jie Chen and Baoquan Chen. Architectural modeling from sparsely scanned range data. *IJCV*, 78(2-3), 2008. 2

[7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *CVPR*, 2020. 1, 8

[8] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *ACM SIGGRAPH*, 2004. 2

[9] James M. Coughlan and Alan L. Yuille. The manhattan world assumption: Regularities in scene statistics which enable bayesian inference. In *NIPS*, 2000. 2

[10] Bertram Drost and Slobodan Ilic. Local hough transform for 3d primitive detection. In *3DV*, 2015. 2

[11] Hao Fang, Florent Lafarge, and Mathieu Desbrun. Planar Shape Detection at Structural Scales. In *CVPR*, 2018. 2

[12] Thomas Holzmann, Michael Maurer, Friedrich Fraundorfer, and Horst Bischof. Semantically aware urban 3d reconstruction with plane-based regularization. In *ECCV*, 2018. 1

[13] Jingwei Huang, Yanfeng Zhang, and Mingwei Sun. Primitivenet: Primitive instance segmentation with local primitive embedding under adversarial metric. In *ICCV*. 2

[14] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *IJCV*, 97(2), 2012. 1, 2

[15] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *ICRA*, 2015. 1

[16] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. *Computer Graphics Forum*, 37, 2018. 1

[17] Pyojin Kim, Brian Coltin, and H Jin Kim. Linear RGB-D SLAM for planar environments. In *ECCV*, 2018. 1

[18] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *Trans. on Graphics*, 36(4), 2017. 5, 6

[19] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *CVPR*, 2019. 2, 5, 6

[20] Thibault Lejemble, Claudio Mura, Loïc Barthe, and Nicolas Mellado. Persistence analysis of multi-scale planar structure graph in point clouds. *Computer Graphics Forum*, 39(2), 2020. 2

[21] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *CVPR*, 2019. 1, 2, 5, 6

[22] Muxingzi Li and Florent Lafarge. Planar Shape Based Registration for Multi-modal Geometry . In *BMVC*, 2021. 1

[23] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. Globfit: Consistently fitting primitives by discovering global relations. *Trans. on Graphics*, 2011. 2

[24] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *CVPR*, 2019. 2

[25] Stuart Lloyd. Least squares quantization in pcm. *Trans. on Information Theory*, 28(2), 1982. 2

[26] Eric-Tuan Lê, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and Niloy J. Mitra. Cpfn: Cascaded primitive fitting networks for high-resolution point clouds. In *ICCV*. 1, 2

[27] David Marshall, Gabor Lukacs, and Ralph Martin. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *TPAMI*, 23(3), 2001. 2

[28] Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. Rapter: rebuilding man-made scenes with regular arrangements of planes. *Trans. on Graphics*, 34(4), 2015. 1, 2

[29] Liangliang Nan and Peter Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *ICCV*, 2017. 8

[30] Sven Oesau, Florent Lafarge, and Pierre Alliez. Planar shape detection and regularization in tandem. *Computer Graphics Forum*, 35(1), 2016. 1, 2

[31] Sven Oesau, Yannick Verdie, Clément Jamin, Pierre Alliez, Florent Lafarge, Simon Giraudot, Thien Hoang, and Dmitry Anisimov. Point set shape detection. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.3 edition, 2021. 2, 5, 6

[32] Trung-Thanh Pham, Tat-Jun Chin, Jin Yu, and David Suter. The random cluster model for robust geometric fitting. In *CVPR*, 2012. 2

[33] Trung T. Pham, Markus Eich, Ian Reid, and Gordon Wyeth. Geometrically consistent plane extraction for dense indoor 3d maps segmentation. In *IROS*, 2016. 1, 2

[34] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 2

[35] Yiming Qian and Yasutaka Furukawa. Learning pairwise inter-plane relations for piecewise planar reconstruction. In *ECCV*, 2020. 2

[36] Rongqi Qiu, Qian-Yi Zhou, and Ulrich Neumann. Pipe-run extraction and reconstruction from point clouds. In *ECCV*, 2014. 2

[37] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. *International archives of photogrammetry, remote*

sensing and spatial information sciences, 36(5), 2006. 1, 2, 4, 5, 6

[38] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm. Usac: A universal framework for random sample consensus. *TPAMI*, 35(8), 2013. 2

[39] Carolina Raposo, Michel Antunes, and Joao P. Barreto. Piecewise-planar stereoscan: Sequential structure and motion using plane primitives. *TPAMI*, 40(8), 2018. 1

[40] Zhongzheng Ren, Ishan Misra, Alexander G. Schwing, and Rohit Girdhar. 3d spatial recognition without spatially labeled 3d. In *CVPR*, 2021. 1

[41] Chiara Romanengo, Andrea Raffo, Yifan Qie, Nabil Anwer, and Bianca Falcidieno. Fit4cad: A point cloud benchmark for fitting simple geometric primitives in cad objects. In *3D Object Retrieval workshop*, 2021. 2

[42] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer graphics forum*, 26(2), 2007. 1, 2, 4, 5, 6

[43] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Chaudhuri, and Radomyr Mech. Parsenet: A parametric surface fitting network for 3d point clouds. In *ECCV*, 2020. 1, 2, 5, 6

[44] Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. Variational shape approximation of point set surfaces. *Computer Aided Geometric Design*, 80, 2020. 2

[45] Christiane Sommer, Yumin Sun, Erik Bylow, and Daniel Cremers. PrimiTect: Fast Continuous Hough Voting for Primitive Detection. In *ICRA*, 2020. 2

[46] Julian Straub, Guy Rosman, Oren Freifeld, John Leonard, and John Fisher. A mixture of manhattan frames: Beyond the manhattan world. In *CVPR*, 2014. 2

[47] Dirk Jan Struik. *Lectures on Classical Differential Geometry*. Courier Corporation, 1961. 7

[48] Bo Sun and Philippos Mordohai. Oriented point sampling for plane detection in unorganized point clouds. In *ICRA*, 2019. 2

[49] Anh-Vu Vo, Linh Truong-Hong, Debra F Laefer, and Michela Bertolotto. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104, 2015. 2

[50] Jianhua Wu and Leif Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24(3), 2005. 2

[51] Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qixing Huang. Hpnet: Deep primitive segmentation using hybrid representations. In *ICCV*. 2, 5, 6

[52] Fengting Yang and Zihan Zhou. Recovering 3d planes from a single image via convolutional neural networks. In *ECCV*, 2018. 2

[53] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10, 000 3d-printing models. *arXiv:1605.04797*, 2016. 2

[54] Zihan Zhou, Hailin Jin, and Yi Ma. Robust plane-based structure from motion. In *CVPR*, 2012. 1

[55] Chen Zhu, Zihan Zhou, Zirang Xing, Yanbing Dong, Yi Ma, and Jingyi Yu. Robust plane-based calibration of multiple non-overlapping cameras. In *3DV*, 2016. 1

[56] Lingjie Zhu, Shuhan Shen, Xiang Gao, and Zhanyi Hu. Large scale urban scene modeling from mvs meshes. In *ECCV*, 2018. 1

[57] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *ICCV*, 2017. 2