Primitive Fitting Using Deep Geometric Segmentation

Duanshun Li[a]* and Chen Feng[b]*

[a]Department of Civil and Environmental Engineering, University of Alberta, Canada
[b]Tandon School of Engineering, New York University, United States
E-mail: duanshun@ualberta.ca, cfeng@nyu.edu

Abstract -

To identify and fit geometric primitives (e.g., planes, spheres, cylinders, cones) in a noisy point cloud is a challenging yet beneficial task for fields such as reverse engineering and as-built BIM. As a multi-model multi-instance fitting problem, it has been tackled with different approaches including RANSAC, which however often fit inferior models in practice with noisy inputs of cluttered scenes. Inspired by the corresponding human recognition process, and benefiting from the recent advancements in image semantic segmentation using deep neural networks, we propose BAGSFit as a new framework addressing this problem. Firstly, through a fully convolutional neural network, the input point cloud is point-wisely segmented into multiple classes divided by jointly detected instance boundaries without any geometric fitting. Thus, segments can serve as primitive hypotheses with a probability estimation of associating primitive classes. Finally, all hypotheses are sent through a geometric verification to correct any misclassification by fitting primitives respectively. We performed training using simulated range images and tested it with both simulated and real-world point clouds. Quantitative and qualitative experiments demonstrated the superiority of BAGSFit.

Keywords -

Geometric Segmentation; Primitive Fitting; As-Built BIM

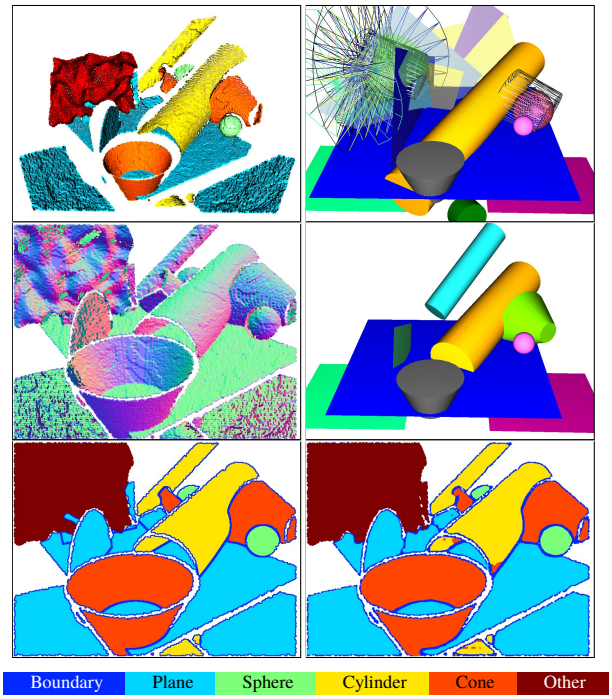


Figure 1: Primitive fitting on a simulated test range image (top left) with BAGSFit (middle right) vs. RANSAC (top right) [4]. Estimated normals (middle left) and ground truth labels (bottom left) are used to train a fully convolutional segmentation network in BAGSFit. During testing, the boundary-aware and thus instance-aware segmentation (bottom right) is predicted, and sent through a geometric verification to fit final primitives (randomly colored). Comparing with BAGSFit, the RANSAC-based method produces more misses and false detections of primitives (shown as transparent or wire-frame), and thus a less appealing visual result.

1 Introduction

The idea of decomposing a scene or a complex object into a set of simple geometric primitives for visual object recognition dates back as early as 1980s when Biederman proposed the object Recognition-By-Components theory [1], in which primitives were termed "geons". Although some real scenes can be more complicated than simple combinations of "geons", there are many useful ones that can be efficiently modeled for the purpose of as-built building information modeling [2, 3], where man-made structures are primarily composed of basic primitives. Also, it is desirable to model individual components in a building separately because they have different functions and were built with different types of material and construction methods. Separated components can then be used in building information modeling systems to carry its relevant semantic information and perform quantitative analysis.

This primitive fitting problem is a classic chicken-and-egg problem: with given primitive parameters, point-to-

*The authors contributed equally. And Chen Feng is the corresponding author.

primitive (P2P) membership can be determined by nearest P2P distance; and vise versa by robust estimation. The challenge comes when multiple factors present together: a noisy point cloud (thus noisy normal estimation), a cluttered scene due to multiple instances of a same or multiple primitive models, and also background points not explained by the primitive library. See Figure 1 for an example. The seminal RANSAC-based method [4] often tends to fit inferior primitives that do not well represent the real scene.

Different from existing work for this multi-model multi-instance fitting problem, we are inspired by human visual perception of 3D primitives. As found by many cognitive science researchers, human "observers' judgments about 3D shape are often systematically distorted" [5]. For example, when looking at a used fitness ball, many people would think of it as a sphere, although it could be largely distorted if carefully measured. This suggests that human brain might not be performing exact geometric fitting during primitive recognition, but rather rely on "qualitative aspects of 3D structure" [5] from visual perception.

Due to recent advancements in image semantic segmentation using convolutional neural networks (CNN) [6,7], it is natural to ask: whether CNN can be applied to this problem with geometric nature and find the P2P membership as segmentation on the range image without geometric fitting at all? Our answer is yes, which leads to the BAGSFit framework that reflects this thought process.

Contributions This paper contains the following key contributions:

- We present a methodology to easily obtain point-wise ground truth labels from simulated dataset for supervised geometric segmentation, demonstrate its ability to generalize to real-world dataset, and released the first simulated dataset [1] for development and benchmarking.
- We present a novel framework for multi-model 3D primitive fitting, which performs both qualitatively and quantitatively superior than RANSAC-based methods on noisy range images of cluttered scenes.
- We introduce this geometric segmentation task for CNN with several design analyses and comparisons.

Related Work Constructive Solid Geometry (CSG) and Boundary Representation (BRep) are the most common 3D representations in building information modeling (BIM) because of their simplicity and flexibility [8]. Both methods model a complex object with a collection of basic primitives. However, creating these models manually can be costly and time-consuming [9], especially for as-built modeling because the dimension of the objects are implicitly constrained by the scanned data. In [8], as-built modeling was separated into several auxiliary tasks, including geometric primitive fitting, point cloud clustering, shape fitting, and point cloud classification. The study of primitive fitting-based as-built modeling in construction mainly focus on the modeling of indoor scene with parametric planar surface [10] or industrial plant with cylinders [9,11]. Such methods lack the capability to deal with complicated buildings with planar walls, cylinder-shaped columns, cone-shaped roofs, and so on. The remaining of this section will investigate existing methods for multi-model primitive fitting.

By assuming the existence of potentially multiple classes of primitives in a scene, it is more realistic than the previous group, and thus more challenging when a cluttered scene is observed with noisy 3D sensors. Previous work with this assumption can be roughly grouped further into the following categories: *Segmentation*: Segmentation methods [12–14] segment a point cloud into individual clusters and performs classification and fitting either during the segmentation or afterwards.

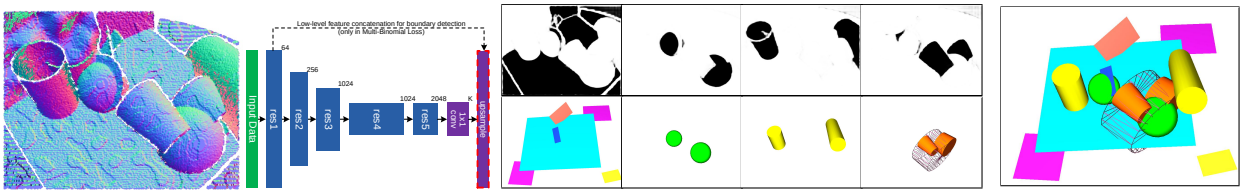
RANSAC: Since the seminal work by Schnabel et al. [4], the idea of selecting sampled primitive hypotheses to maximize some scoring functions becomes a default solution to this problem. In this research, we chose this method as our baseline. In practice, the 3D sensor noise is often more structured (e.g., depth dependent noises for range images) than uniform or Gaussian in 3D as experimented in many of these papers. What really makes the problem difficult is that those noisy points belonging to other partially occluded primitive instances become outliers of the primitive to be fit at hand, causing false detections of "ghost" primitives not existed in the real scene but still with very small fitting errors and large consensus scores, e.g. the ghost cones fitted with cylinder and background points in the top right of Figure 1. More recently, prior probabilities or quality measure of the data [15,16] were used to improve the probability of sampling an all-inlier subset.

Energy Minimization: Unlike the sequential and greedy nature of RANSAC based methods, it is appealing in theory to define a global energy function in terms of P2P membership that once minimized results in desired solution [17–20]. However most of them are only shown on relatively small number of points of simple scenes without much clutters or occlusions, and it is unclear how they will scale to larger datasets due to the intrinsic difficulty and slowness of minimizing the energy function.

2 Framework Overview

Figure 2 gives an overview of the multi-model primitive fitting process by our BAGSFit framework. As introduced

[1]The dataset is available at https://github.com/ai4ce/BAGSFit.



(a) CNN-based Segmentation (b) Geometric Verification (c) Fitted Primitives

Figure 2: BAGSFit overview. In 2a, a proper form of a range image, e.g., its normal map, is input to a fully convolutional neural network for segmentation. We use the same visualization style for the CNN as in [7], where each block means layers sharing a same spatial resolution, decreasing block height means decimating spatial resolution by a half, and red dashed lines means loss computation. The resulting segmentation probability maps $\mathbf{Y}_k$ (top row of 2b, darker for higher probability) for each primitive class k are sent through a geometric verification to correct any misclassification by fitting the corresponding class of primitives (bottom row of 2a). Finally, fitted primitives are shown in 2c. Without loss of generality, this paper only focuses on four common primitives: plane, sphere, cylinder, and cone.

above, the front-end of this framework (Figure 2a) mimics the human visual perception process in that it does not explicitly use any geometric fitting error or loss in the CNN. Instead, it takes advantage of a set of stable features learned by CNN that can robustly discriminate points belonging to different primitive classes. The meaning of a pixel of the output probability map (top row of Figure 2b) can be interpreted as how much that point and its neighborhood look like a specific primitive class, where the neighborhood size is the CNN receptive field size.

Such a segmentation map could already be useful for more complex tasks [21], yet for the sake of a robust primitive fitting pipeline, one cannot fully trust this segmentation map as it inevitably contains misclassification, just like all other image semantic segmentations. Fortunately, by separating pixels belonging to individual primitive classes, our original multi-model problem is converted to an easier multi-instance problem. Following this segmentation, a geometric verification step based on efficient RANSAC [4] incorporates our strong prior knowledge, i.e., the mathematical definitions of those primitive classes, to find the parametric models of the objects for each type of primitives. Note that RANSAC variants using prior inlier probability to improve sampling efficiency are not adopted in this research, because 1) they are orthogonal to the proposed pipeline; and 2) the robustness of primitive fitting is highly dependent on the spatial distribution of samples. Different from spatial consistency based methods [22, 23] mainly dealing with homography detection, in our 3D primitive fitting task, samples with points very close to each other usually lead to bad primitive fitting results [4]. Thus the potential of using the CNN predicted class probabilities to guide the sampling process, while being interesting, will be deferred for future investigations.

The advantage for this geometric segmentation task is that exact spatial constraints can be applied to detect correct primitives even with noisy segmentation results. One could use the inliers after geometric verification to correct the CNN segmentation results, similar to the CRF post-processing step in image semantic segmentation that usually improves segmentation performance.

3 Ground Truth from Simulation

Before going to the details of our segmentation CNN, we need to first address the challenge of preparing training data, because as most state-of-the-art image semantic segmentation methods, our CNN needs supervised training. To our best knowledge, we are the first to introduce such a geometric primitive segmentation task for CNN, and there is no existing publicly available datasets for this task. For image semantic segmentation, there have been many efforts to use simulation for ground truth generation. Yet it is hard to make CNNs trained over simulated data generalize to real world images, due to intrinsic difficulties of tuning a large number of variables affecting the similarities between simulated images and real world ones.

However, since we are only dealing with geometric data, and that 3D observation is less sensitive to environmental variations, plus observation noise models of most 3D sensors are well studied, we hypothesize that simulated 3D scans highly resemble real world ones such that CNNs trained on simulated scans can generalize to real world data. If this is true, for this geometric task, we can get infinite number of point-wise ground truth almost for free.

Although saved from tedious manual labeling, we still need a systematic way of generating both random scene layouts of primitives and scan poses so that simulated scans are meaningful and covers true data variation as much as possible. Due to the popular Kinect-like scanners, which mostly applied in indoor environment, we choose to focus on simulating indoor scenes. And note that this dose not limit our BAGSFit framework to only indoor situations. Given a specific type of scenes and scanners, one

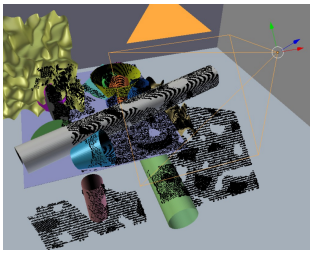


Figure 3: A simulated Kinect scan of a random scene. Black dots represents the scanned points.

should be able to adjust the random scene generation protocols similarly. Moreover, we hypothesize that the CNN is less sensitive to the overall scene layout. What's more important is to show the CNN enough cases of different primitives occluding and intersecting with each other.

Thus, we choose to randomly generate a room-like scene with 10 meters extent at each horizontal direction. An elevated horizontal plane representing a table top is generated at a random position near the center of the room. Other primitives are placed near the table top to increase the complexity. Furthermore, empirically, the orientation of cylinder/cone axis or plane normal is dominated by horizontal or vertical directions in real world. Thus several primitive instances at such orientations are generated deliberately in addition to fully random ones. For planes, two additional disk shaped planes are added to make the dataset more general. To make the training set more realistic, two NURBS surfaces (class name "Other" in Figure 1) are added, representing objects not explained by our primitive library in reality.

An existing scanner simulator, Blensor [24], was used to simulate VGA-sized Kinect-like scans, where class and instance IDs can be easily obtained during the virtual scanning process by ray-tracing. For each scene, we obtain a total number of 192 scans with varying view directions surrounding the scene. Totally 20 scenes were generated following this protocol. 18 scenes, i.e. 3456 scans, were split for training, and the other 2 scenes, i.e. 384 scans, were used for validation. Figure 3 shows the screenshot of such a scan. The test set is generated through a similar protocol, containing 20 scenes (each with 36 scans). Note that invalid points were converted to the zero-depth point avoiding computation issues.

4 Boundary Aware Geometric Segmentation

Our segmentation network (Figure 2a) follows the same basic network as described in [7], which is based on the 101-layer ResNet [25] with minor modifications to improve segmentation performance. While the semantic segmentation CNN architecture is actively being developed, there are several design choices to be considered to achieve the best performance on a given base network for our new task.

Position vs. Normal Input. The first design choice is about the input representation. Since we are dealing with 3D geometric data, what form of input should be supplied to the CNN? A naive choice is to directly use point positions as a 3-channel tensor input. After all, this is the raw data we get in reality, and if the CNN is powerful enough, it should be able to learn everything from this input form. However, it is unclear how or whether necessary to normalized it.

A second choice is to use estimated per-point unit normals as the input. This is also reasonable, because we can almost perceive the correct segmentation by just looking as the normal maps as shown in Figure 2a. Plus it is already normalized, which usually enables better CNN training. However, since normals are estimated from noisy neighboring points, one might have concerns about loss of information compared with the previous choice. And a third choice is to combine the first two, resulting in a 6-channel input, through which one might hope the CNN to benefit from merits of both.

Separate vs. Joint Boundary Detection. When multiple instances of a same primitive class occlude or intersect with each other, even an ideal primitive class segmentation can not divide them into individual segments, leaving a multi-instance fitting problem still undesirable for the geometric verification step to solve, which discounts the original purpose of this geometric segmentation. Moreover, boundaries usually contains higher noises in terms of estimated normals, which could negatively affect primitive fittings that use normals (e.g., 2-point based cylinder fitting). One way to alleviate the issue is to cut such clusters into primitive instances by instance-aware boundaries. To realize this, we also have two choices, 1) training a separate network only for instance boundary detection, or 2) treating boundary as an additional class to be segmented jointly with primitive classes. One can expect the former to have better boundary detection results as the network focuses to learn boundary features only, although as a less elegant solution with more parameters and longer running time. Thus it is reasonable to trade the performance a bit for the latter one. Note that with such a step, we could already move from category- to boundary- and thus instance-aware segmentation by region-grow after removing all instance-aware boundaries.

Handling of Background Class. When generating random scenes, we added NURBS modeling background points not explained by the four primitive classes, for a more realistic and challenging dataset. Thus we need to properly handle them in the CNN. Should we ignore back-

ground class when computing the loss, or add it as an additional class?

For all of the above design questions, we will rely on experiments to empirically select the best performing ones.

5 Geometric Verification and Evaluation

5.1 Verification by Fitting

Given the predicted probability maps $\{\mathbf{Y}_k\}$, we need to generate and verify primitive hypotheses and fit primitive parameters of the correct ones to complete our mission.

One direct way of hypothesis generation is to simply binarize the BAGS output $\{\mathbf{Y}_k\}$ by thresholding to produce a set of connected components, and fit only one k-th class primitive for a component coming from $\mathbf{Y}_k$. However, when the CNN incorrectly classify certain critical regions due to non-optimal thresholds, two instances can be connected, thus leading to suboptimal fittings or miss detection of some instances. Moreover, a perfect BAGS output may bring another issue that an instance gets cut into several smaller pieces due to occlusions (e.g., the top left cylinder in Figure 2a). And fitting in smaller regions of noisy scans usually result in false instance rejection or lower estimation accuracy. since the core contribution of this paper is to propose and study the feasibility of BAGS-Fit as a new strategy towards this problem, we leave it as our future work to develop more systematic ways to better utilize $\{\mathbf{Y}_k\}$ for primitive fitting.

In this work, we simply follow a classic "arg max" prediction on $\{\mathbf{Y}_k\}$ over each point, and get K groups of hypothesis points associated to each of the K primitive classes. Then we solve K times of multi-instance primitive fitting using the RANSAC-based method [4]. Note this does not completely defeat the purpose of BAGS. The original RANSAC-based method feed the whole point cloud into the pipeline and detect primitives sequentially in a greedy manner. Because it tends to detect larger objects first, smaller primitives close to large ones could often be missed, as their member points might be incorrectly counted as inlier of larger objects, especially if the inlier threshold is improperly set. BAGS can alleviate such effects and especially removing boundary points from RANSAC sampling is expected to improve its performance.

5.2 Primitive Fitting Evaluation

It is non-trivial to design a proper set of evaluation criteria for primitive detection and fitting accuracy, and we are not aware of any existing work or dataset that does so. It is difficult to comprehensively evaluate and thus compare different primitive fitting methods partly because 1) as mentioned previously, due to occlusion, a single instance are commonly fitted into multiple primitives, both of which may be close enough to the ground truth instance; and 2) such over detection might also be caused by improper inlier thresholds on a noisy data.

Pixel-wise average precision (AP) and AP of instances matched at various levels (50~90%) of point-wise intersection-over-union (IoU) are used for evaluating image based instance segmentation problems [26]. However, this typical IoU range is inappropriate for our problem. More than 50% IoU means at most one fitted primitive can be matched for each true instance. Since we don't need more than 50% of true points to fit a reasonable primitive representing the true one, this range is over-strict and might falsely reject many good fits: either more than 50% true points are taken by other incorrect fits, or during observation the true instance is occluded and split into pieces each containing less than 50% true points (see Figure 5 for more examples). After all, a large IoU is not necessary for good primitive fitting.

Thus, the IoU is replaced by intersection-over-true (IoT) in this problem. It indicates the number of true inliers of a predicted primitive over the total number of points in the true instance. Thus, a predicted primitive and a true instance is matched iff 1) IoT>30% and 2) the predicted primitive having the same class as the true instance. This indicates that one instance can have at most 3 matched predictions.

Based on the above matching criteria, a matched instance (if exists) can be identified for each predicted primitive. On the contrary, each true instance may have several best matching prediction candidates. To eliminate the ambiguity, the candidate that has the smallest fit error is selected as the best match. To be fair and consistent, fitting error is defined as the mean distance to a primitive by projecting all of the points in the true instance onto the predicted primitive. After the matches are found, *primitive average precision* (PAP) and *primitive average recall* (PAR) are used to quantify the primitive detection quality.

$$PAP = N_{p2t}/N_p, PAR = N_{t2p}/N_t, \quad (1)$$

where N_{p2t} is the number of predictions having a matched true instance, N_p the total number of predicted primitives, N_{t2p} the number of true instance with a best prediction, and N_t the total number of true instances, all counted over the whole test set.

6 Experiments and Discussion

6.1 Geometric Segmentation Experiments

Network Short Names. To explore answers to the questions raised in section 4, we designed several CNNs and their details with short names are listed as follows:

Table 1: Geometric segmentation evaluation. Red highlights the best along a column, and magenta for the top 3 best.

	Precision						Recall						IoU						F1						Accuracy
	BND	PLN	SPH	CYL	CON	AVE	BND	PLN	SPH	CYL	CON	AVE	BND	PLN	SPH	CYL	CON	AVE	BND	PLN	SPH	CYL	CON	AVE	
N+BO	0.944						0.820						0.781						0.877						0.964
P		0.915	0.811	0.867	0.642	0.809		0.971	0.620	0.715	0.664	0.743		0.891	0.599	0.655	0.488	0.658		0.939	0.664	0.762	0.611	0.744	0.871
N		0.979	0.915	0.934	0.727	**0.889**		**0.988**	**0.884**	0.788	**0.829**	**0.872**		**0.968**	**0.860**	0.752	0.633	**0.803**		**0.983**	**0.894**	0.826	0.734	**0.859**	**0.924**
PN		0.978	0.913	0.919	0.710	0.880		0.984	0.868	0.806	0.797	0.864		0.962	0.847	0.758	0.601	0.792		0.980	0.882	0.838	0.711	0.853	0.920
N+BAGS	0.868	0.963	0.908	0.926	**0.756**	**0.888**	0.849	0.976	0.874	**0.833**	0.821	**0.871**	0.752	0.941	0.848	**0.790**	**0.654**	**0.797**	0.858	0.969	0.884	**0.859**	**0.755**	**0.865**	0.918
N5		**0.980**	**0.917**	**0.940**	0.744	**0.895**		0.979	0.877	0.809	0.808	**0.868**		0.960	0.854	0.776	0.642	**0.808**		0.979	0.889	0.844	0.741	**0.863**	**0.940**
N5+BAGS	0.847	0.966	0.906	0.932	0.728	0.883	0.804	0.970	0.873	0.808	0.812	0.853	0.702	0.939	0.845	0.769	0.630	0.777	0.825	0.968	0.883	0.842	0.732	0.850	**0.921**

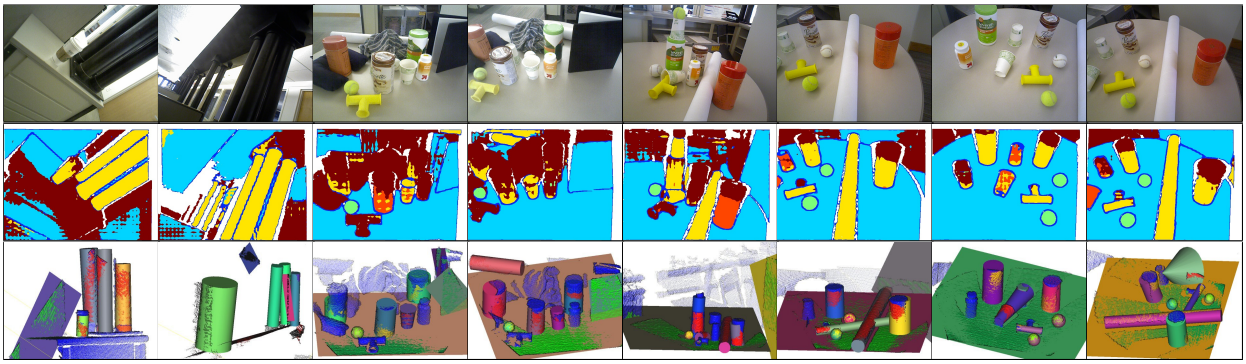


Figure 4: BAGSFit (**N5+BAGS**) on real Kinect scans. Top: RGB image of the scanned scene. Middle: segmentation results. Bottom: fitted primitives (randomly colored) rendered together with real scans.

- **P/N/PN**. Basic networks, using position (**P**), normal (**N**), or both (**PN**) as input, trained with a multinomial loss function, outputting a 4-channel *mutual-exclusive* class probability maps (i.e., each pixel's probabilities sum up to one, $K = 4$). Background class points, the NURBS, are ignored for loss computation.
- **N+BAGS**. Network trained with normal input and BAGS labels (i.e., instance-aware boundary as an additional class jointly trained, $K = 5$).
- **N5**. Same as basic network **N** except treating the background class as an additional class involved in loss computation ($K = 5$).
- **N5+BAGS**. Same as **N+BAGS** except trained using a multi-binomial manner (i.e., boundary and NURBS are two additional classes jointly trained, $K = 6$).
- **N+BO**. Same as **N** except only trained to detect boundary (i.e., a binary classifier, $K = 2$).

Implementation Details. We implemented the geometric segmentation CNNs using *Caffe* [27] and *DeepLabv2* [6]. Normals were estimated by PCA using a 5×5 window. We use meters as the unit for networks requiring position input. Instance-aware boundaries were calculated if not all pixels belong to a same instance (or contain invalid points) in a 5×5 window. Input data size was randomly cropped into 440×440 during training time, while full VGA resolution was used during test time. All of our networks were trained with the following hyper-parameters tuned on the validation set: 50 training epochs (i.e. 17280 iterations), batch size 10, learning rate 0.1 linearly decreasing to zero until the end of training, momentum 0.9, weight decay 5e-4. The networks were trained and evaluated on several NVIDIA TITAN X GPUs each with 12 GB memory, with a 2.5Hz testing frame rate.

Discussions. Evaluation results of all 12 networks on the test set of 720 simulated scans are in table 1.

1. Comparing the **P/N/PN** rows, we found that normal input turned out to be the best, and interestingly outperforming combination of both normal and position. This may be caused by the difficulty in normalizing position data for network input.
2. Comparing the **N** with **N+BAGS**, we found that adding additional boundary detection to the segmentation only have very small negative influences to the segmentation performance. This is appealing since we used a single network to perform both segmentation and boundary detection. Further comparing the **N+BAGS** with **N+BO**, we found that BAGS in fact increases the boundary recall comparing to **N+BO** that only detects boundaries.
3. Comparing the **N5** with **N**, we found that the effect of ignoring background class is inconclusive in terms of significant performance changes, which however suggests the benefit of jointly training the background class, as this enables the following steps to focus only on regions seemingly explainable by the predefined primitive library.

Just for reference, we tried SVM using neighboring 7×7 or 37×37 normals or principal curvatures for this task, and the highest pixel-wise accuracy we obtained after many

Table 2: Primitive fitting evaluation. Red highlights the best along a column, while magenta highlights the top 3 best.

	No. Primitives Fitted (N_p)					No. Matched Instance(N_{t2p})					$\frac{N_{t2p}^{all}}{N_p^{all}}$	Primitive Average Precision (PAP)					Primitive Average Recall (PAR)					Fitting Error (cm)				
	PLN	SPH	CYL	CON	ALL	PLN	SPH	CYL	CON	ALL		PLN	SPH	CYL	CON	ALL	PLN	SPH	CYL	CON	ALL	PLN	SPH	CYL	CON	ALL
ERANSAC	4596	1001	2358	3123	11078	2017	542	942	879	4380	0.395	0.453	0.541	0.402	0.286	0.403	0.500	0.432	0.403	0.443	0.456	0.915	0.324	**0.766**	0.954	0.810
P	5360	621	2242	2037	10260	2448	591	1219	944	5202	0.507	0.470	0.952	0.549	0.468	0.516	0.607	0.471	0.521	0.476	0.541	0.936	**0.248**	0.931	**0.519**	**0.759**
N	4617	961	2789	2492	10859	**2565**	**870**	1456	1254	**6145**	0.566	0.571	0.905	0.532	0.507	0.576	**0.636**	**0.693**	0.623	**0.633**	**0.640**	0.903	0.403	1.229	0.657	0.866
PN	4537	888	3172	2133	10730	2522	859	**1498**	1197	**6076**	0.566	0.572	0.967	0.480	0.570	0.577	0.625	0.684	**0.641**	0.604	**0.632**	0.903	0.397	1.196	0.628	0.852
N+BAGS	3893	845	2299	2108	9145	2279	796	1453	1149	5677	**0.621**	0.594	0.942	0.637	0.548	**0.626**	0.565	0.634	0.621	0.580	0.591	0.765	0.363	1.144	0.587	**0.768**
N5	3701	863	1874	1876	8314	2490	859	1458	**1226**	**6033**	**0.726**	**0.693**	0.995	0.793	**0.663**	**0.740**	0.617	0.684	0.624	0.619	**0.628**	0.841	0.395	1.163	0.617	0.815
N5+BAGS	3500	804	1765	1730	7799	2254	804	1397	1129	5584	**0.716**	0.654	**1.000**	**0.796**	0.658	**0.723**	0.559	0.640	0.598	0.570	0.581	**0.742**	0.367	1.096	0.555	**0.740**

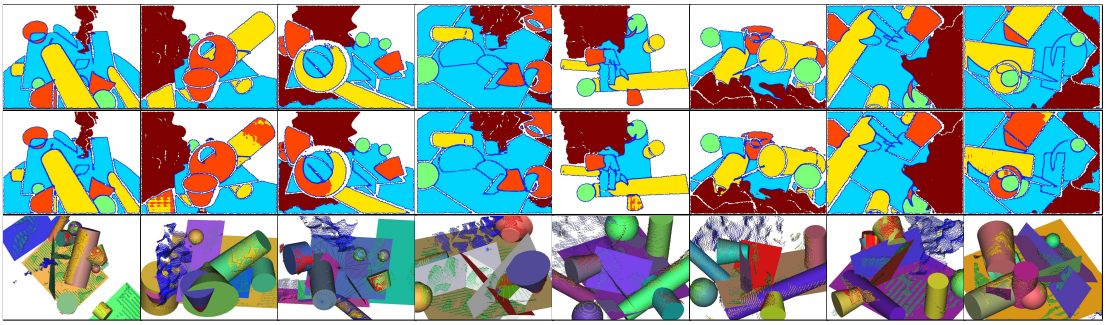


Figure 5: BAGSFit (**N5+BAGS**) on simulated test scans. Top: Ground truth labels. Middle: segmentation results. Bottom: fitted primitives (randomly colored) rendered together with real scans.

parameter tuning is only 66%.

Generalizing to Real Data. Even though we did not tune the simulated scanner's noise model to match our real Kinect scanner, Figure 4 shows that the network trained with simulated scans generalizes well to real world data.

6.2 Primitive Fitting Experiments

For fitting primitives, we used the original efficient RANSAC implementation [4] both as our baseline method (short name **ERANSAC**) and for our geometric verification.

Experiment Details. We used the following parameters required in [4] for all primitive fitting experiments, tuned on the validation set in effort of maximizing **ERANSAC** performance: min number of supporting points per primitive 1000, max inlier distance 0.03m, max inlier angle deviation 30 degrees (for counting consensus scores) and 45 degrees (for final inlier set expansion), overlooking probability 1e-4. The simulated test set contains 4033 planes, 1256 spheres, 2338 cylinders, 1982 cones, and in total 9609 primitive instances.

Discussions. Using respective network's segmentation as input to the geometric verification, the primitive fitting results were evaluated on the simulated test set and summarized in table 2 together with the **ERANSAC** baseline.

1. **ERANSAC** performance is significantly lower than most variants of BAGSFit, in accordance with our qualitative evaluation.
2. **N5** related experiments receives highest PAP scores, which is reasonable due to the recognition and removal of background classes that greatly reduce the complexity of scenes.
3. In terms of average fitting error, **N+BAGS** < **N**, **N5+BAGS** < **N5** which strongly supports the benefit of BAGS as mentioned in section 5.1.
4. **N5+BAGS** gets the lowest fitting error, benefiting from both background and boundary removal.

More results. Figure 5 shows more testing results.

7 Future Work

Our next step is to compare BAGSFit with MaskR-CNN [28], quantitatively evaluate it on real scans, and apply it in as-built BIM generation. We also plan to extend the network so it can directly predict primitive parameters.

Acknowledgment

This work was supported by Mitsubishi Electric Research Labs (MERL) and New York University. We thank Yuichi Taguchi, Srikumar Ramalingam, Zhiding Yu, Teng-Yok Lee, Esra Cansizoglu, and Alan Sullivan for their helpful comments.

References

[1] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[2] Pingbo Tang, Daniel Huber, Burcu Akinci, Robert Lipman, and Alan Lytle. Automatic reconstruction of as-built building information models from laser-scanned point clouds: A

review of related techniques. *Automation in construction*, 19(7):829–843, 2010.

[3] Jianxiong Xiao and Yasutaka Furukawa. Reconstructing the world's museums. *Int'l J. Computer Vision*, 110(3):243–258, 2014.

[4] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Wiley Online Library, 2007.

[5] James T Todd. The visual perception of 3d shape. *Trends in cognitive sciences*, 8(3):115–121, 2004.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[7] Z. Yu, C. Feng, M. Y. Liu, and S. Ramalingam. CASENet: Deep category-aware semantic edge detection. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017.

[8] Viorica Pătrăucean, Iro Armeni, Mohammad Nahangi, Jamie Yeung, Ioannis Brilakis, and Carl Haas. State of research in automatic as-built modelling. *Advanced Engineering Informatics*, 29(2):162–171, 2015.

[9] Soon-Wook Kwon, Frederic Bosche, Changwan Kim, Carl T Haas, and Katherine A Liapi. Fitting range data to primitives for rapid local 3d modeling using sparse range point clouds. *Automation in construction*, 13(1):67–81, 2004.

[10] Jaehoon Jung, Sungchul Hong, Seongsu Jeong, Sangmin Kim, Hyoungsig Cho, Seunghwan Hong, and Joon Heo. Productive modeling for development of as-built bim of existing indoor structures. *Automation in Construction*, 42:68–77, 2014.

[11] Joohyuk Lee, Hyojoo Son, Changmin Kim, and Changwan Kim. Skeleton-based 3d reconstruction of as-built pipelines from laser-scan data. *Automation in construction*, 35:199–207, 2013.

[12] Aleš Leonardis, Alok Gupta, and Ruzena Bajcsy. Segmentation of range images as the search for the best description of the scene in terms of geometric primitives. 1990.

[13] Zahra Toony, Denis Laurendeau, and Christian Gagné. Describing 3d geometric primitives using the gaussian sphere and the gaussian accumulator. *3D Research*, 6(4):42, 2015.

[14] Kristiyan Georgiev, Motaz Al-Hami, and Rolf Lakaemper. Real-time 3d scene description using spheres, cones and cylinders. *arXiv preprint arXiv:1603.03856*, 2016.

[15] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 220–226, 2005.

[16] B. J. Tordoff and D. W. Murray. Guided-mlesac: faster image transform estimation by using matching priors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1523–1535, 2005.

[17] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *Int'l J. Computer Vision*, 97(2):123–147, 2012.

[18] Oliver J Woodford, Minh-Tri Pham, Atsuto Maki, Riccardo Gherardi, Frank Perbet, and Björn Stenger. Contraction moves for geometric model fitting. In *Proc. European Conf. Computer Vision (ECCV)*, pages 181–194. Springer, 2012.

[19] Daniel Barath and Jiri Matas. Multi-class model fitting by energy minimization and mode-seeking. *arXiv preprint arXiv:1706.00827*, 2017.

[20] Paul Amayo, Pedro Pinies, Lina M Paz, and Paul Newman. Geometric multi-model fitting with a convex relaxation algorithm. *arXiv preprint arXiv:1706.01553*, 2017.

[21] Wolfram Martens, Yannick Poffet, Pablo Ramón Soria, Robert Fitch, and Salah Sukkarieh. Geometric priors for gaussian process implicit surfaces. *IEEE Robotics and Automation Letters*, 2(2):373–380, 2017.

[22] T. Sattler, B. Leibe, and L. Kobbelt. Scramsac: Improving ransac's efficiency with a spatial consistency filter. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, pages 2090–2097, 2009.

[23] Kai Ni, Hailin Jin, and F. Dellaert. Groupsac: Efficient consensus in the presence of groupings. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, pages 2193–2200, 2009.

[24] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: blender sensor simulation toolbox. *Advances in visual computing*, pages 199–208, 2011.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016.

[26] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. *arXiv preprint arXiv:1611.08303*, 2016.

[27] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, 2014.

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, pages 2980–2988, 2017.