Robust 3D Reconstruction of Complicated Scenes with Industrial Structures

Anonymous ECCV submission

Paper ID 1437

Abstract. This paper presents an automatic approach to reconstruct complicated 3D scenes with industrial structures from large-scale noisy point clouds. We observe three key elements in this modeling problem, namely, primitives, similarities, and joints. While *primitives* capture the cylindric and planar shapes, *similarities* reveal the inter-primitive relations that intrinsically exist in industrial structures because of human design and construction. By applying statistical analysis over point normals and point positions, we directly discover a set of primitive similarities from raw data, which are adopted as constraints in primitive fitting algorithm to make it insensitive to data problems. Finally, *joints* are automatically generated to seal cracks and propagate connectivity information. The resulting model is more than a collection of 3D triangles, as it contains semantic labels for pipes, cylinders, and planar surfaces as well as their connectivity information.

1 Introduction

The fast development of acquisition techniques has turned scanning devices into commercial off-the-shelf tools. Current scanners are capable of scanning large-scale industrial sites and producing data sets with billions of points. Since the increasing density and complexity of point scans has made manual 3D reconstruction more tedious and expensive, there is a need to develop automatic, robust and noise-insensitive approaches to reconstruct complete, and accurate 3D models for industrial structures.

A popular strategy of 3D reconstruction from point scans is to simplify a triangular mesh that minimizes the geometric fitting error with respect to the input points (*e.g.*, [1–3]). In general, these methods assume that the point cloud is the best description of the reality and tend to *fit* a mesh to the raw data. However, we argue that in complicated industrial structures, the complexity of multiple scans, occlusions, and reflections give rise to severe noise and incompleteness to the raw data (*e.g.*, Figure 1(a)). A pure data-driven method suffers from these issues and generates 3D meshes with rough surfaces and cracks although they are faithful to the point scans (*e.g.*, Figure 1(b)). In addition, the resulting triangles contain no structural semantic or connectivity information. To alleviate this problem, prior knowledge of model shapes are introduced, known as *primitives*. They are detected and fitted to the raw data, forming a simple and clean representation of the scene (*e.g.*, [4–6]). To further refine the locally fitted primitives,

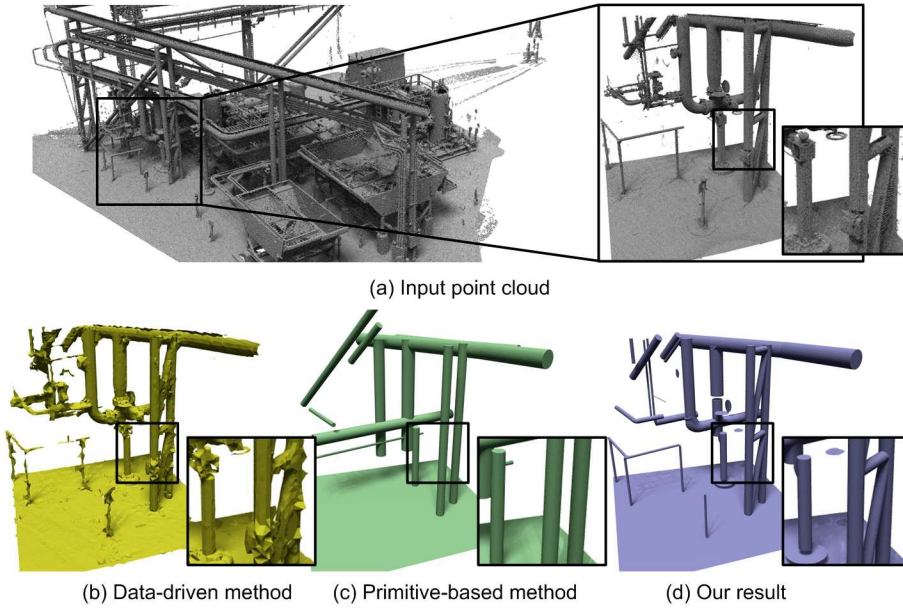

Fig. 1. Given the point scan of a complicated scene with industrial structures (a), we show 3D reconstruction results from three different approaches: (b) a data-driven method [3], (c) GlobFit [7], and (d) our method.

higher level prior knowledge (*e.g.*, global relation across primitives [7, 8]) is introduced in a post-processing step. Although these methods work well on small objects and simple scenes, the primitive fitting quality drops greatly when the complexity of the scene grows. Primitive detection failure and significant fitting error makes the post-refinement an intractable task (*e.g.*, Figure 1(c)).

We present an automatic, robust method to reconstruct complicated 3D scenes with industrial structures. In order to deal with noise and data incompleteness, we adopt prior knowledge of both primitive shapes and global similarities. Instead of following the fit-primitive-then-refine strategy in traditional primitive-based methods, we choose to introduce the global information in an early stage. In particular, we directly detect and estimate the global similarities of primitive orientations and primitive placements individually, based on statistical analysis of point normals and point positions. These global data are then used as constraints in creating primitives and fitting them to the point cloud. Since the global information is discovered before primitive fitting, it avoids the problems of early primitive fitting such as detection and fitting errors. In addition, to further capture the primitive junctions and propagate the connectivity between primitives that broadly exist in industrial structures, we create *joints* between adjacent parts. Our method can robustly reconstruct 3D scenes with numerous types and variations of industrial structures as shown in Figure 1(d).

2 Related Work

We scan previous work in the following two categories, *i.e.*, data-driven reconstruction and primitive-based modeling.

2.1 Data-Driven Reconstruction

A general strategy of 3D reconstruction from point cloud is to simplify a triangular mesh model that minimizes the distance between the input points and the mesh surface. Research efforts following this strategy are usually known as *data-driven* reconstruction because they take the input data as *truth* and make a trade-off between the size of the mesh and the geometry fitting error [1, 2]. Different heuristics are introduced to produce modeling preferences such as smoothness [9, 10] and sharp feature [11].

2.2 Primitive-Based Modeling

Data-Driven reconstruction has the advantage of generality and adaptation to a variety of input point clouds. However, in many cases, introducing prior knowledge of object shapes can significantly reduce the solution space and thus greatly simplifies the reconstruction problem. For instance, Schnabel *et al.*[5] proposed a efficient RANSAC approach to detect primitive shapes from point clouds. Hofer *et al.*[12] adopt line geometry for the recognition and reconstruction of 3D surfaces. Many efforts of fitting primitives are made towards reverse engineering such as [13]. In addition to the prior knowledge of primitive shapes, higher level of knowledge representing the similarities and relations between primitive elements are also introduced and explored by Li *et al.*[7]. This work deals with small scale objects and produces primitives exhibiting global relations. However, as stated in Section 1, it relies heavily on fitting quality of primitives, thus loses accuracy when dealing with complicated industrial structures.

3 Overview

3.1 Typical Features

We observe from manual model-creation (*e.g.*, Figure 2) that the following are the main features a human operator uses to approximate the scene:

1. **Primitives**: Most industrial structures are composed of primitives (*e.g.*, planes, cylinders, as shown in Figure 2(a)). It is intuitive to look such facilities and scenes as compositions of primitives.
2. **Similarities**: Similarities among primitives are common. For example, in Figure 2(b), parallelism of cylinders is a type of similarity.
3. **Joints**: Joints often exist for maintaining connectivity of primitives, *e.g.*, Figure 2(c).

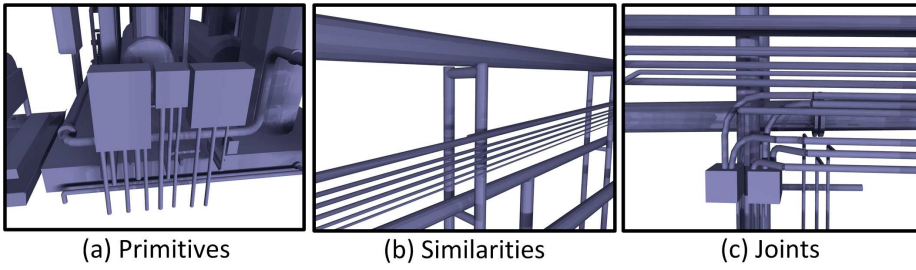

Fig. 2. Close-ups of manual created models illustrating three typical features considered in our approach. (a) Planes and cylinders make up the primitive set adopted in our approach; (b) Three oriented sets of parallel cylinders (two horizontal and one vertical) illustrate similarities of primitives; (c) Joints between cylinders smoothly connect different primitives.

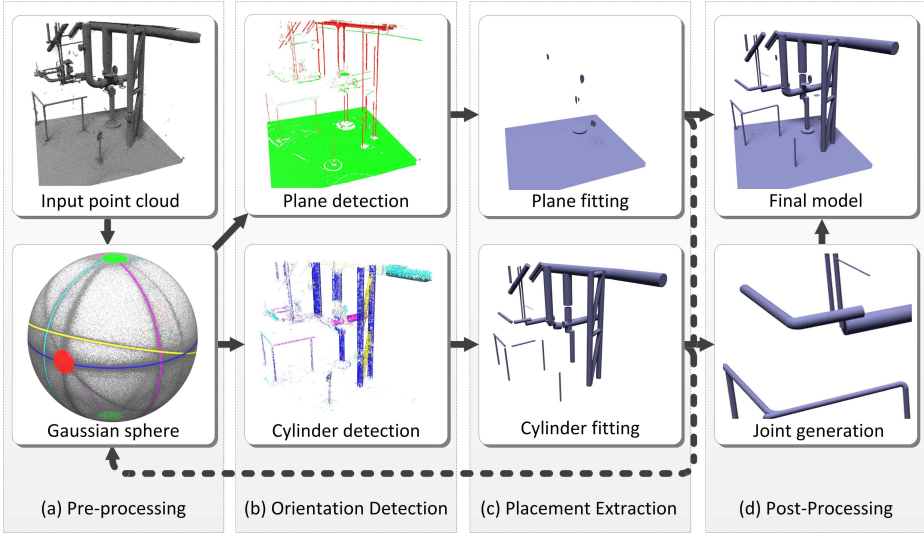

Fig. 3. Our robust modeling pipeline. Normals of input points are projected onto a Gaussian sphere, where two patterns (i.e., clusters and great circles) are detected to determine orientations and segment points into groups. Points within groups are further separated to decided placement of different primitives. Orientation detection and placement extraction are iteratively performed until enough primitives are detected. Joints between cylinders are generated, and merged into cylinder models to produce final models.

3.2 Pipeline

Our processing pipeline accepts large-scale point cloud data as input, and aims at modeling industrial structures in the scene. It is composed of the following steps (as shown in Figure 3).

1. **Pre-Processing:** Normals of input points are mapped to a Gaussian sphere, called the Normal sphere.
2. **Primitive Orientation Detection:** Several patterns are detected to indicate orientations of primitives. Point clouds are classified based on the normals (or directions) of primitives they belong to, while irrelevant points are removed.
3. **Primitive Placement Extraction:** Points within groups are further segmented and processed separately. Positions and boundaries of planes and cylinders are determined and modeled by triangle meshes.
4. **Post-Processing:** Joints of cylinders are generated to connect cylinders with C^1 continuity.

4 Primitive Orientation Detection

We make a key observation that in this noisy and complicated environment, global information (*e.g.*, regularities among primitives) is extremely useful and can significantly improve the quality of fitting. For example, in many scenes, there are cylinders (*i.e.*, pillars) that are built upright due to architectural reasons (Figure 2(b)). Knowing that these upright cylinders belong to the same group helps us create parallel cylinder primitives, which (1) fits better to human intuition (and thus can create more realistic models), (2) decreases the degree-of-freedom of the fitting problem, (3) reduces sensitivity to the noise of data, and thus greatly improves the robustness of the algorithm.

4.1 Global Information Acquisition

In our problem, we focus on two major kinds of global information: normals of planes, and axis-orientation of cylinders. We adopt a statistical model based on normal analysis, which detects this global information automatically. Normals of points are estimated using covariance analysis [14], and then projected onto the Normal sphere, from which we observe the following patterns.

1. **Clusters:** Points on the same plane are supposed to have the same normal (Figure 4(a)). In noisy environment, the normals do not exactly coincide, but rather form a cluster on the Gaussian sphere (Figure 4(b)). The center of the cluster is a good estimation of the accurate normal. Note that plane patterns in a local area usually fall into a few dominant of directions (*e.g.*, vertical normal for horizontal plane primitives). A normal cluster may contain normals from a number of plane primitives in the area.
2. **Great circles:** Normals of points on a cylinder primitive are orthogonal to the axis of this cylinder (Figure 5(a)). They form a great circle on the Normal sphere (Figure 5(b)). Variations of the great circle occur due to noise, but the pattern can be observed clearly on the sphere. The normal of this great circle is an indication of the cylinder axis. Once again, multiple cylinders oriented in same direction can contribute to a circle.

4.2 Plane Detection

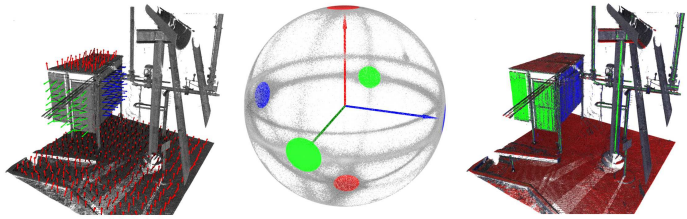

Fig. 4. Illustration for plane detection: (a) Points on the parallel planes have similar normals; (b) Normals are projected onto a Gaussian sphere and clusters are detected; (c) Points are segmented based on clustering result.

We denote the Gaussian sphere as $\mathbf{S_G} = \{\mathbf{P_{G_1}}, \mathbf{P_{G_2}}, \ldots, \mathbf{P_{G_t}}\}$, where $\mathbf{P_{G_i}}$ denotes a point on the Gaussian sphere. Mean-shift algorithm [15] is employed on $\mathbf{S_G}$ to detect cluster centers $\mathbf{C_\Pi} = \{\mathbf{C_{\Pi_1}}, \mathbf{C_{\Pi_2}}, \ldots, \mathbf{C_{\Pi_n}}\}$. Only clusters with density above a threshold are preserved. Each cluster center $\mathbf{C_{\Pi_i}}$ is considered to be a candidate plane normal. Since normals vary due to noise, a small neighborhood of each cluster center is calculated to tolerate small variations.

$$\mathbf{N_{\Pi_i}} = \{\mathbf{p} \in \mathbf{S_G} || \mathbf{p} - \mathbf{C_{\Pi_i}}| < \delta_\Pi\} \tag{1}$$

Each point from the original point cloud is then checked and grouped based on which cluster $\mathbf{N_{\Pi_i}}$ its normal belongs to (Figure 4(c)).

4.3 Cylinder Detection

Plane normal candidates are removed from Normal sphere $\mathbf{S_G}$ by the following formula, because they may interfere in cylinder detection.

$$\mathbf{S'_G} = \mathbf{S_G} - \bigcup_{i=1}^{n} \mathbf{N_{\Pi_i}} \tag{2}$$

The new Gaussian sphere is denoted as $\mathbf{S'_G} = \{\mathbf{P'_{G_1}}, \mathbf{P'_{G_2}}, \ldots, \mathbf{P'_{G_{t'}}}\}$. Great circles are detected on Normal sphere $\mathbf{S'_G}$ using RANSAC method [16]. Specifically, we randomly pick N normal pairs from $\mathbf{S'_G}$, namely, $(\mathbf{P'_{G_{i_1}}}, \mathbf{P'_{G_{j_1}}})$, $(\mathbf{P'_{G_{i_2}}}, \mathbf{P'_{G_{j_2}}}), \ldots, (\mathbf{P'_{G_{i_N}}}, \mathbf{P'_{G_{j_N}}})$. Each pair $(\mathbf{P'_{G_{i_k}}}, \mathbf{P'_{G_{j_k}}})$ decides a unique great circle on the Normal sphere, whose normal is decided by the following formula:

$$\mathbf{P_{R_k}} = \mathbf{P'_{G_{i_k}}} \times \mathbf{P'_{G_{j_k}}} \tag{3}$$

These vectors $\mathbf{S_R} = \{\mathbf{P_{R_k}}\}$ form another sphere called RANSAC sphere. Mean-shift algorithm is employed again to detect cluster centers on $\mathbf{S_R}$, namely,

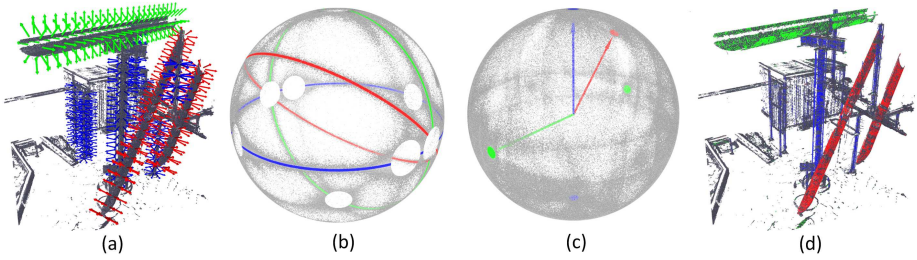

Fig. 5. Illustration for cylinder detection: (a) Points on the parallel cylinders have normals orthogonal to their axis; (b) Normals are projected onto Normal sphere and great circles are detected; (c) Detection of great circles in Normal sphere is converted into detection of clusters in RANSAC sphere; (d) Points are segmented based on clustering result.

$\mathbf{C_O} = \{\mathbf{C_{O_1}}, \mathbf{C_{O_2}}, \ldots, \mathbf{C_{O_m}}\}$. Similarly, a narrow strip $\mathbf{N_{O_i}}$ on Normal sphere is defined corresponding to each cluster center $\mathbf{C_{O_i}}$.

$$\mathbf{N_{O_i}} = \{\mathbf{p} \in \mathbf{S'_G} | \mathbf{p} \cdot \mathbf{C_{O_i}} < \delta_O\} \tag{4}$$

Each point from the original point cloud is then checked and grouped based on which strip $\mathbf{N_{O_i}}$ its normal belongs to (Figure 5(d)).

5 Primitive Placement Extraction

In Section 4, primitives are grouped based on their directions, while unrelated points are ruled out. To generate models of these primitives, exact positions and boundaries need to be determined.

5.1 Cylinder Fitting

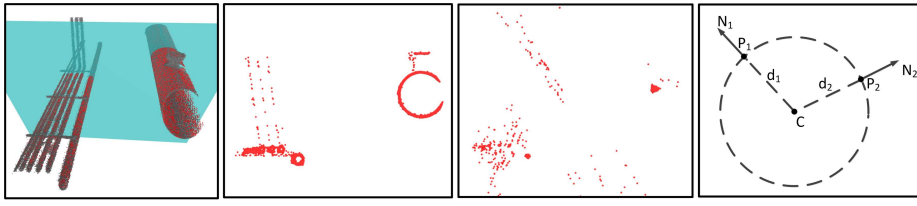

Fig. 6. Cylinder position calculation: (a) Points in the same group are projected to a plane that is orthogonal to their normals; (b) Image of projected points in the same group; (c) Image of possible centers of circles, where RANSAC is performed to; (d) The case that two points with normals decide a unique circle.

Cylinder Position Calculation The candidate points for cylinders are grouped based on the directions of detected cylinders. Next we want to further divide the points within groups into several parts, each of which belongs to a separate cylinder, and rule out unrelated points. For example, in Figure 6(a), the red points have been detected belonging to cylinders with direction $\mathbf{n}$. We project these points (with their normals) into a plane $\mathbf{\Pi}$ (*i.e.*, the green plane) that is orthogonal to $\mathbf{n}$ (Figure 6(b)). Circles on this image correspond to cylinders of input point clouds.

RANSAC algorithm [16] is employed again to accomplish this task. N pairs of points with normals are randomly picked up, and each of which votes for a circle that goes through this pair of points and matches their normals, if it exists. For example, Figure 6(d) shows two points $\mathbf{P_1}$, $\mathbf{P_2}$ with their normals $\mathbf{N_1}$, $\mathbf{N_2}$, respectively. The intersection of $\mathbf{N_1}$ and $\mathbf{N_2}$ is calculated and denoted as $\mathbf{C}$. A required circle exists if the following condition is satisfied,

$$|\mathbf{CP_1}| = |\mathbf{CP_2}| \tag{5}$$

The required circle is centered at $\mathbf{C}$ with radius $r = |\mathbf{CP_1}| = |\mathbf{CP_2}|$. The centers of all these circles are drawn on a image (Figure 6(c)), and Mean-Shift algorithm [15] is adopted to find cluster centers, which are centers of extracted cylinders.

Cylinder Boundary Extraction The remaining parameters to be determined are the boundaries. A naïve algorithm would be scanning the points that lie on a specific infinite cylinder, and calculate the interval on the axis of this cylinder. However, this algorithm is not robust in the noisy environment, because accidental noise would place dramatic effects on the length of cylinders.

To solve this problem, we cut each cylinder into multiple small sections, and reject immediately those sections with too few inliers on them. Thus we get a string S consisting of "0"s and "1"s, where "1"s indicate survived sections and "0"s indicate rejected sections. We want to find the longest substring S', starting and ending with "1", which does not contain too many "0"s (e.g., the red box of Figure 7). Specifically, we quantize this requirement that the substring contains at least p percent of ones. The boundaries of extracted cylinder correspond to boundaries of substring S'.

5.2 Plane Fitting

Before plane fitting is performed, inlier points on detected cylinders are ruled out, since they may interfere with plane fitting. Candidate points for planes have been separated into groups in Section 4.2, based on normals of the planes.

A region-growing algorithm is adopted to segmented the points within each group into connected areas. For each connected area, a convex hull is calculated and output as the plane model.

Due to noise, small pieces of planes are often modeled, containing very few inlier points on them. To deal with this, planes with sub-threshold small areas are discarded. Our observation si that such noise is often sparse and scattered.

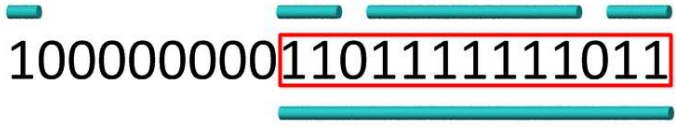

Fig. 7. The process of determining the boundaries of output cylinder. The red box indicates the longest substring that starts with "1", ends with "1", and contains more than p percent of "1"s.

6 Joint Generation

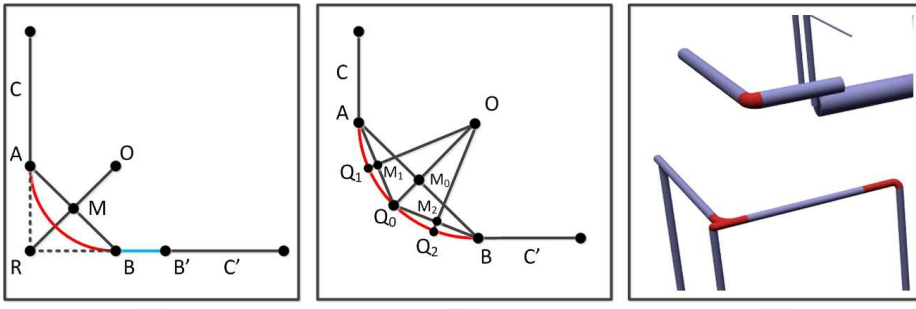

Fig. 8. Illustration for joint generation of non-parallel cylinders. (a) Calculation of parameters (*i.e.*, center and radius) of the joint torus piece; (b) Illustration of "binary interpolation" process; (c) Example of a model after joint generation (red part shows the generated joints).

It is quite common that joints exist between cylinders maintaining the continuity between adjacent cylinders. In this stage, joints composed of torus pieces are generated automatically for non-parallel cylinders, while both cylinders are kept unchanged. In the following discussion, we use a nine-tuple $\mathbf{C} = (\mathbf{P_0}, \mathbf{N}, r, d_{min}, d_{max})$ to denote a cylinder, where $\mathbf{P}$ denotes a point on the axis, $\mathbf{N}$ denotes direction of the axis, r denotes radius, d_{min} and d_{max} denote the interval on the axis (*i.e.*, the two boundary points of the axis are $\mathbf{B_1} = \mathbf{P_0} + d_{min} \cdot \mathbf{N}$ and $\mathbf{B_2} = \mathbf{P_0} + d_{max} \cdot \mathbf{N}$).

Suppose the two cylinders to be joined are $\mathbf{C} = (\mathbf{P_0}, \mathbf{N}, r, d_{min}, d_{max})$ and $\mathbf{C}' = (\mathbf{P_0}', \mathbf{N}', r', d'_{min}, d'_{max})$, and parametric representation of their axes are:

$$L : \mathbf{P} = \mathbf{P_0} + u\mathbf{N} \tag{6}$$

$$L' : \mathbf{P}' = \mathbf{P_0}' + v\mathbf{N}' \tag{7}$$

The criteria of cylinders C and C' are the following.

- Radius Difference: $|r - r'| < \Delta_r$
- Skew Distance: $d_{skew}(L, L') < \Delta_s$
- Angel Restriction: $\langle \mathbf{N}, \mathbf{N}' \rangle > \Delta_\theta$
- Gap Distance: $\min_{1 \leqslant i \leqslant 2, 1 \leqslant j \leqslant 2}(\mathbf{B_i}, \mathbf{B'_j}) < \Delta_g$

The intersection of two axes (*i.e.*, L and L') is determined by the following equation.

$$\begin{bmatrix} N_x & -N'_x \\ N_y & -N'_y \\ N_z & -N'_z \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} P_{0x} - P'_{0x} \\ P_{0y} - P'_{0y} \\ P_{0z} - P'_{0z} \end{bmatrix} \tag{8}$$

Due to existence of skew distance of two axes, a unique solution (*i.e.*, intersecting point) does not always exist. In such cases, the Least Mean Squares [17] solution will determine two points $\mathbf{P_1}$ and $\mathbf{P_1}'$, satisfying:

$$|\mathbf{P_1}\mathbf{P_1}'| = \min_{\mathbf{Q} \in L, \mathbf{Q}' \in L'} |\mathbf{Q}\mathbf{Q}'| \tag{9}$$

The average point $\mathbf{R} = \frac{\mathbf{P_1} + \mathbf{P_1}'}{2}$ is regarded as the approximate intersecting point. One of the cylinders is extended to ensure the distances of two end-points to the intersecting point are the same (*e.g.*, in Figure 8(a), cylinder C' is extended from point $\mathbf{B}'$ to $\mathbf{B}$, so that $|\mathbf{AR}| = |\mathbf{BR}|$). We denote the near-end boundary points of L and L' as $\mathbf{A}$ and $\mathbf{B}$, mid-point of $\mathbf{A}$ and $\mathbf{B}$ as $\mathbf{M}$.

We put a restriction that the joint should have C^1 continuity, that is, shares the same tangent vector with the axes of cylinders. Figure 8(a) shows the axes of cylinders and joints. The center point $\mathbf{O}$ and radius r_0 of the torus (as shown in Figure 8(b)) are calculated by the following equations.

$$\mathbf{O} = \mathbf{M} + \overrightarrow{\mathbf{RM}} \cdot \tan^2\left(\frac{\langle \mathbf{N}, \mathbf{N}' \rangle}{2}\right) \tag{10}$$

$$r_0 = \frac{|\mathbf{OA}| + |\mathbf{OB}|}{2} \tag{11}$$

Next, enough number of cross sections of the torus is generated by "binary interpolation" (Figure 8(b)). To be specific, we calculate midpoint $\mathbf{M_0}$ of $\mathbf{A}$ and $\mathbf{B}$, and extend $\overrightarrow{\mathbf{OM_0}}$ to length r_0 and get $\mathbf{Q_0}$. Then we calculate midpoint $\mathbf{M_1}$ of point $\mathbf{A}$ and $\mathbf{Q_0}$, and extend $\overrightarrow{\mathbf{OM_1}}$ to get point $\mathbf{Q_1}$. Similarly we can get $\mathbf{Q_2}$. This iterative process is adopted iteratively until enough precision is achieved. Notice that we interpolate radius simultaneously, so that it changes gradually from r to r'. These cross sections are connected to form the joint (Figure 8(c)).

7 Hierarchical Process

To optimize the robustness and accuracy of primitive detection, a hierarchical fitting process is adopted. More specifically, we remove the points belonging to

the primitives that have been already detected, and run the same processing-pipeline iteratively with parameters optimized for smaller primitives, until all primitives are detected and modeled.

Since each level of the hierarchy is independent, multiple instances may be created for the same primitive in different levels. To compensate for this, a method of removing redundant primitives is adopted. In particular, we perform an enumeration on the primitives and check whether one is an subset of another, and remove the smaller one(s) if necessary.

8 Experimental Results

Figure 9 shows a site reconstructed from 381M LiDAR points. As a pre-process, we employ a voxel-grid method [18] to downsample the input data. Also, to overcome any scene-size limitations to our method, we clip an arbitrary-size data set into cubic areas of 5m×5m×5m to be modeled separately. The subvolume results are merged together in a later process. Our method successfully generates 1017 cylinders and 514 planes in 12 hours on a consumer-level desktop (Intel Core i7-2600 3.40GHz CPU with 8GB memory).

To verify the capability of handling various scenes, we did experiments on a set of input point clouds using previous methods, our method and manual creation (Figure 10). In addition, we compared the triangle number, average squared distance from input point clouds to triangle meshes, and the ratio of outlier points with distance greater than 4cm, as shown in Table 1. The data-driven method [3] is the most accurate representation of the input point cloud, but is unaware of any constraints or semantic information, and therefore produces rough models that include all of the noisy input points. The Globfit method [7] takes primitives and their global relationships into consideration, but is still based on and therefore bounded by the effects of RANSAC. Our method, on the other hand, adopts a complete top-down hierarchical algorithm, detects orientation and placement separately, achieves the best robustness and produces clean and smooth models comparable to manually created models[1].

9 Conclusion

We present a robust processing pipeline to automatically build industrial models from large-scale 3D point clouds. This approach introduces the global information in an early stage of primitive fitting, and treats orientation detection and placement extraction as two separate sub-problems, thus avoid getting lost in local details. Joints are also created to smoothly connect primitives, making our results human-intuitive.

[1] The manual models used in our experiments are not complete because they were built within limited budget for time and effort.

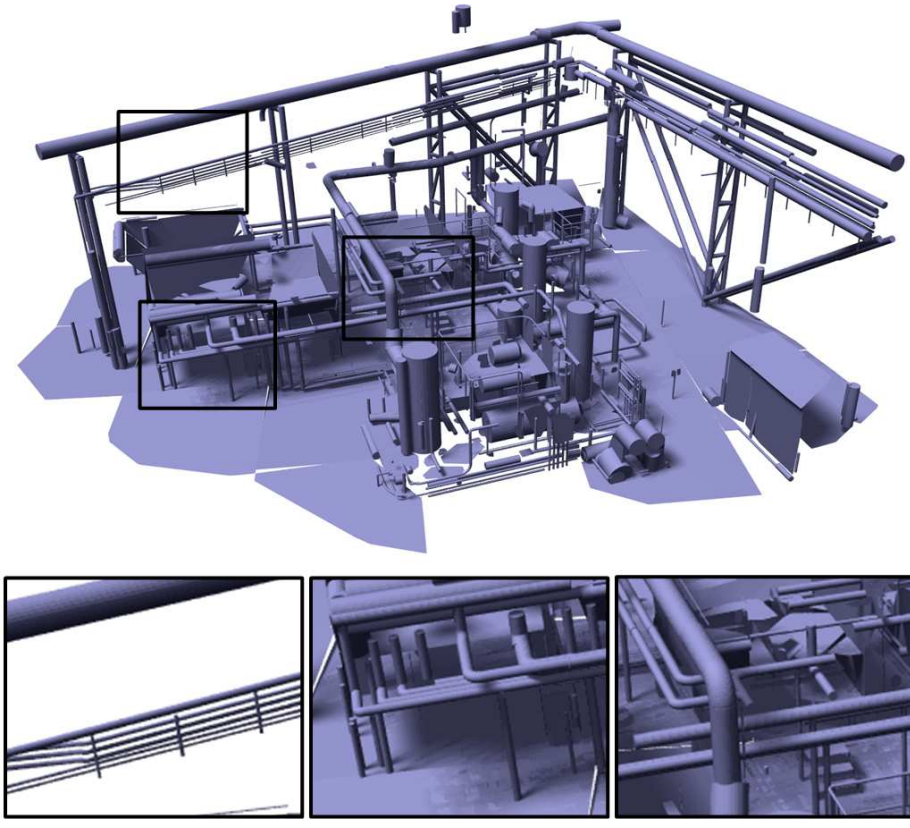

Fig. 9. Modeling result of our method for a 25m×25m×10m scene.

Models in Figure 10		Data-driven method [3]	GlobFit method [7]	Our method	Manual creation
First row (960638 points)	Triangle number	242574	2587	13646	224
	Average distance^2	1.79	10189.09	79.68	22736.04
	Outlier ratio	0.42%	83.64%	15.81%	83.59%
Second row (597783 points)	Triangle number	157998	5120	11776	1030
	Average distance^2	0.63	71.07	1690.53	1077.66
	Outlier ratio	0.09%	18.93%	16.91%	36.77%
Third row (173250 points)	Triangle number	93941	1792	8201	1196
	Average distance^2	0.11	9.86	1.37	59.18
	Outlier ratio	$< 0.01\%$	5.57%	1.74%	66.07%

Table 1. Quantitative evaluation of results shown in Figure 10.

References

1. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: ACM SIGGRAPH. (1992)

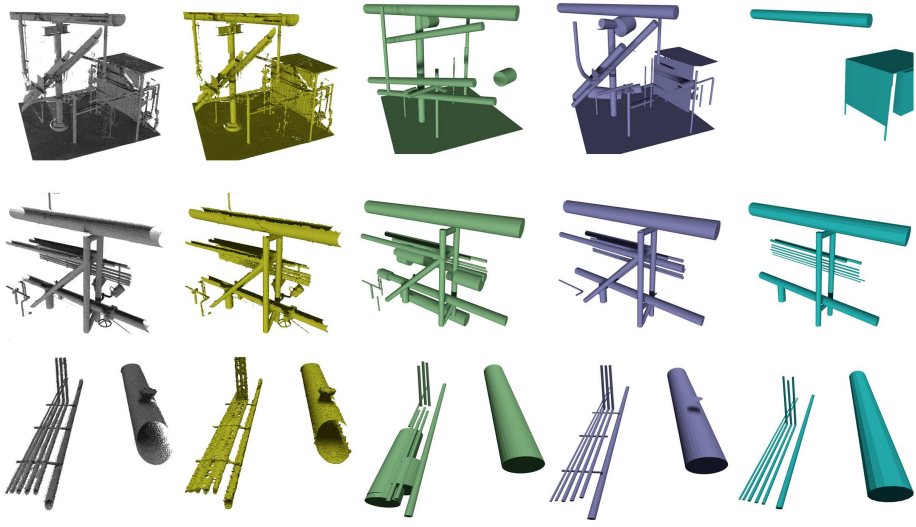

Fig. 10. Models created from various input point clouds (leftmost) using different methods (from left to right): data driven method [2], GlobFit method [7], our method, manual creation.

2. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. Visualization and Computer Graphics, IEEE Transactions on **5** (1999) 349–359
3. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: Meshlab: an open-source mesh processing tool. In: Eurographics Italian Chapter Conference. (2008)
4. Gal, R., Shamir, A., Hassner, T., Pauly, M., Cohen-Or, D.: Surface reconstruction using local shape priors. In: Symposium on Geometry Processing. (2007)
5. Schnabel, R., Wahl, R., Klein, R.: Efficient ransac for point-cloud shape detection. In: Computer Graphics Forum. Volume 26., Wiley Online Library (2007) 214–226
6. Schnabel, R., Degener, P., Klein, R.: Completion and reconstruction with primitive shapes. Computer Graphics Forum (2009)
7. Li, Y., Wu, X., Chrysathou, Y., Sharf, A., Cohen-Or, D., Mitra, N.: Globfit: consistently fitting primitives by discovering global relations. In: ACM Transactions on Graphics (TOG). Volume 30., ACM (2011) 52
8. Zhou, Q., Neumann, U.: 2.5d building modeling by discovering global regularities. In: Computer Vision and Pattern Recognition, IEEE (2012)
9. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3d objects with radial basis functions. In: ACM SIGGRAPH. (2001)
10. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Symposium on Geometry Processing. (2006)
11. Fleishman, S., Cohen-Or, D., Silva, C.: Robust moving least-squares fitting with sharp features. In: ACM Transactions on Graphics (TOG). Volume 24., ACM (2005) 544–552

12. Hofer, M., Odehnal, B., Pottmann, H., Steiner, T., Wallner, J.: 3d shape recognition and reconstruction based on line element geometry. In: ICCV. (2005)
13. Benkö, P., Martin, R.R., Várady, T.: Algorithms for reverse engineering boundary representation models. CAD (2001)
14. Pauly, M.: Point primitives for interactive modeling and processing of 3D geometry. Hartung-Gorre (2003)
15. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on **24** (2002) 603–619
16. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24** (1981) 381–395
17. Lawson, C., Hanson, R.: Solving least squares problems. Volume 15. Society for Industrial Mathematics (1995)
18. Rusu, R., Cousins, S.: 3d is here: Point cloud library (pcl). In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE (2011) 1–4