DeepPipes: Learning 3D PipeLines Reconstruction from Point Clouds

Lili Cheng[a] Zhuo Wei[a] Mingchao Sun[a] Shiqing Xin[a] Andrei Sharf[b]
Yangyan Li[c] Baoquan Chen[d] Changhe Tu[a]

[a]Shandong University [b]Ben-Gurion University [c]Alibaba Group [d]Peking University

Abstract

Pipes are the basic building block in many industrial sites like electricity and chemical plants. Although pipes are merely cylindrical primitives which can be defined by axis and radius, they often consist of additional components like flanges, valves, elbows, tees, etc. 3D pipes are typically dense, consisting of a wide range of topologies and geometries, with large self-occlusions. Thus, reconstruction of a coherent 3D pipe models from large-scale point clouds is a challenging problem. In this work we take a prior-based reconstruction approach which reduces the complexity of the general pipe reconstruction problem into a combination of part detection and model fitting problems. We utilize convolutional network to learn point cloud features and classify points into various classes, then apply robust clustering and graph-based aggregation techniques to compute a coherent pipe model. Our method shows promising results on pipe models with varying complexity and density both in synthetic and real cases.

Keywords: Point cloud, Pipes reconstruction, Convolution network, Skeleton extraction

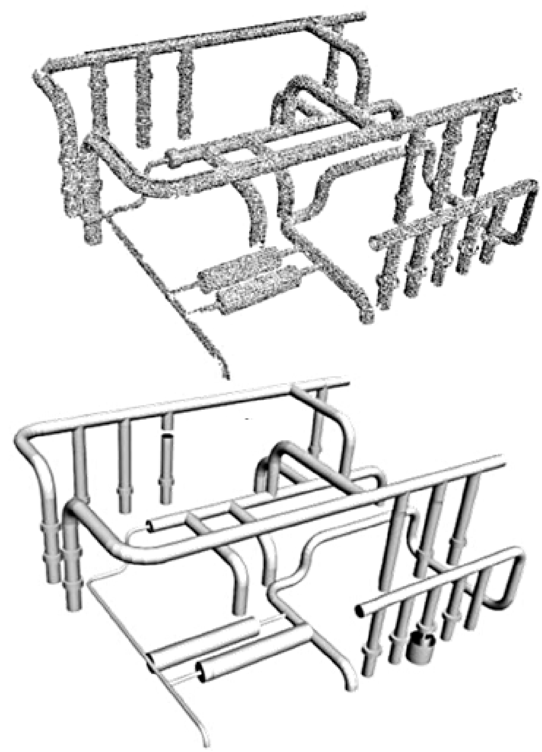

Figure 1. DeepPipes enables 3D reconstruction of a full pipeline with complex parts and relations.

1. Introduction

High quality 3D models of power-plants, petrochemical plants and other industrial sites are crucial in many applications, including disaster simulations, monitoring and executive training. Industrial sites are built according to specific plans often accompanied by 3D CAD models of their structures. Nevertheless, modeling a fully detailed and accurate 3D replica model is a laborious task. Furthermore, such models may not exist for older facilities or may not reflect the current appearance of the site. Nowadays, modern laser scanners allow capturing 3D surfaces and geometries with high accuracy, generating dense point cloud samplings. Nevertheless, in the case of 3D pipes, capturing and sampling the surface geometry is especially challenging.

Pipelines are dominant structures in many industrial sites due to their functional importance and prevalence. They consist of thin structures defined by long cylinders organized in dense and complex configurations. Although pipes are merely cylindrical primitives which can be easily defined by their axis and radius, they often consist of additional components such as flanges, valves, inlets, elbows, tees, etc. Thus, 3D scanning and reconstruction of pipelines is error-prone due to small pipe surfaces and their intricate structure causing large self-occlusions, missing parts and insufficient sampling.

A common approach in 3D reconstruction from scanned data is fitting shape priors to the raw data in a bottom-up manner [6, 32, 31]. Such strategies are well-suited to

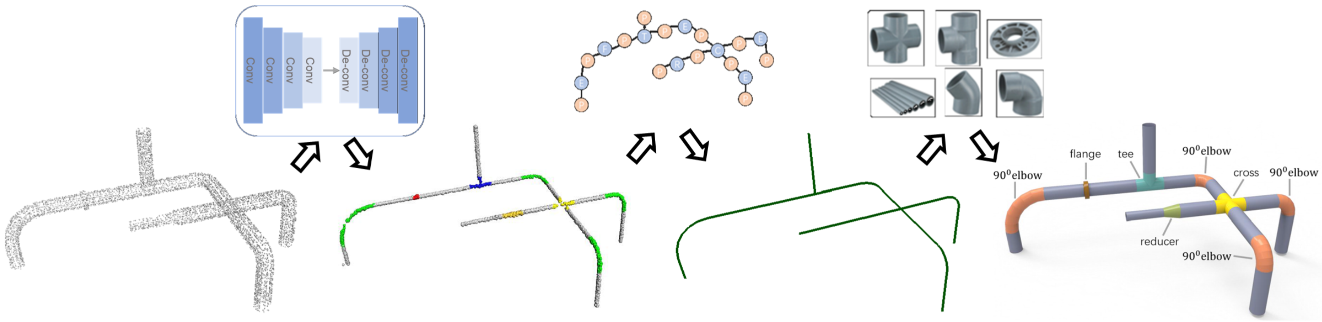

Figure 2. Overview. Left-to-right, starting from a raw pipeline scan, we apply neural network to detect parts. We use graph processing to compute valid relations leading to a coherent full pipeline reconstruction with multiple part types at varying scales and orientations.

industrial sites and mechanical designs since most models are composed of primitive shapes [12]. Nevertheless, such bottom-up methods suffer from locality and can rarely reconstruct models such as full powerplants with accurate connectivity. Bottom-up primitive fitting techniques are also sensitive to noise and outliers due to their lack of global and content aware considerations.

We present an automatic and robust method to pipe reconstruction from noisy 3D scans. Previous techniques [22, 27] focus on recovering the cylindrical pipes and joints structures in industrial plants. Although cylindrical shapes are often the dominant geometry in such sites, real data consists of a large variety of other structures such as flanges, valves, inlets, elbows, tees, etc. (see Figure 1).

We take a prior-based learning approach where we train a deep learning network to detect any part as candidate features in a 3D point cloud. Since such prior detection is often noisy, we incorporate robust clustering [5] with connectivity pruning techniques to filter detection results and generate a consistent graph-like global pipe model. Similar to [18], we embed the initial unreliable local prior detection in a processing framework which accounts for global properties and semantic structures.

Thus, our technique reconstructs local structures that adhere to connectivity rules and semantic relations in the pipes. Our results demonstrate that our method robustly reconstructs complete pipe networks from point clouds of industrial structures.

2. Related Work

In the following we discuss previous works related to reconstruction of 3D pipes, thin structure reconstruction and primitive fitting.

2.1. 3D Pipes Reconstruction

A commonly used approach to 3D pipe reconstruction from point clouds is based on geometry processing and fitting.

Liu *et al*. [22] propose a method that reduces the problem of 3D plant reconstruction into detection of projected pipes as 2D circles in the plane. However, this method is limited to tube-shaped pipes that are orthogonal or parallel to the ground.

Researchers have also investigated fully automated techniques for entire pipeline reconstruction [13]. They perform skeleton extraction followed by segmentation into individual components, and a set of parameters for them are calculated. However, this method has high time complexity and results are easily influenced by noise.

Qiu *et al*. [29] combine primitive similarity detection and fitting to increase reconstruction robustness. They use distribution of points normal to detect similar cylindrical pipes which are then fitted by cylinders. Joints are then heuristically positioned to connect pipes into a fully connected model. Our work bears similarity to Qiu *et al.* in enhancing primitive fitting with detection. Nevertheless, their work searches specifically for self-similarities in the cylinder set while ours is generic and learns a variety of features, learning to detect pipes, joints, flanges and other relevant part configurations in the scene.

Commercial software [8] is also available to interactively reconstruct pipe-runs. However, these products usually require substantial manual work. Our method, on the other hand, is fully automatic without any user intervention.

Hough transform [30] is modified for automatic detection of cylinder parameters in point clouds [27]. After detection, the relationship between cylinders is reconstructed to form a continuous network. Data is post-processed using Smart Plant 3D (SP3D) to model the entire pipeline. However, the range of radius is small.

A technique using normal-based region growing and RANSAC [32] for point cloud processing is proposed for inspection of piping systems of industrial plants [26]. Specifically, the method compares between the CAD design and real scan of the plant models. The inspection result depends strongly on quality of the input point cloud. Similarly, automatic extraction of pipe and flange pairs in point clouds using geometric primitives was demonstrated recently [24]. In their work, they superimpose a clean CAD model with the scanned data to guide the 3D extraction of noisy pipes and flanges. It focuses on extracting pipe and flange pairs, not reconstructing the whole scene.

2.2. Thin Structures Reconstruction

In automated reverse engineering of industrial environment, many researchers have explored the problem of reconstructing arbitrary thin structures such as fences, truss bridges, steel frame buildings, etc. [35, 33]. Similar to pipes reconstruction, they detect main structure, and joints are added to create the connected graph to reconstruct the whole frame.

Besides, many works discuss reconstructions of thin tube, which focuses on the restoration of the skeleton topology. A deformable curve model was introduced [14] that simultaneously captures the topology and geometry of 1D curve-like objects. Reconstruction of thin tubular structures, such as cables or ropes has been explored in [25]. The authors introduce physics simulation to faithfully reconstruct jumbled and tangled cables in 3D. Their method estimates the topology of the tubular object in the form of a single 1D path and also computes a topology-aware reconstruction of its geometry. Similarly, a method that reconstructs continuous 3D bending wires (common in furniture design, metal sculpting, wire jewelry) was presented [20]. The method exploits both simplicity and smoothness priors to overcome severe self-occlusions and missing data.

There is also work using RGBD camera to help rebuild thin structures. Thin 1D curve structures were reconstructed at interactive rates using a handheld RGBD camera [21]. The technique basically aligns and iteratively merges small skeleton curve segments together to form the final complete curve skeleton. Similarly, [16] utilize curves to leverage thin structure reconstruction from sparse multi-view stereo data. Their method integrates between 3D curves and points to compute a 3D manifold reconstruction by considering both.

In a different context, an automatic approach that robustly reconstructs skeletal structures of trees from scanned points was introduced [23]. The method performs a series of global optimizations that fit skeletal structures to the often sparse, incomplete, and noisy point data. Inspired by the optimization of graph structure in this work, we use graph to assist in obtaining skeleton of pipes.

Pipe reconstruction also needs to capture the skeleton and topology. In contrast to other thin structures, pipelines have a specific cylindrical nature while lacking regular patterns such as fences. Furthermore, they are typically rigid bodies in contrast to e.g., flexible wires and their industrial significance demands for a highly accurate result.

2.3. Primitive Fitting

CAD and mechanical models are predominantly made of repetitive basic structures to facilitate easy and economic fabrication. Surface reconstruction involving local fitting of primitive structures has long been the standard in reverse engineering [32]. Starting from an input scan, Gal *et al*. [6] use multi-scale partial matching to fit a small set of basic shapes to local neighborhoods as local priors. Schnabel *et al*. [2009] [31] present an interesting hole-filling algorithm that is guided by primitive detection.

To account for both local fitting accuracy along with global relations an algorithm was developed [18]. The local fit of the primitive model is determined by how well the inferred model agrees to the observed data, while the global relations are iteratively learned and enforced through a constrained optimization.

Robust cylinder detection and extraction in raw point clouds were introduced in [34]. They utilize point normal and curvature for cylinder fitting followed by mean shift [4] clustering. Due to the high noise levels in industrial plants scans, hand-crafted features as the above may prove heuristically. Instead we take a deep learning approach to pipe features in scanned points.

A primitive-based segmentation method for mechanical CAD models was introduced [12]. The method assumes a limited number of dominant orientations that primitives are either parallel or orthogonal to, narrowing down their search space. Thus, they simply search for 2D primitives such as circles and lines in dominant directions 2D projections. Finally, they generate an over-complete set of primitives and formulate the segmentation as a set cover optimization problem.

Recently, a new approach to robustly extract cylindrical primitives from a 3D point cloud was introduced [28]. The method computes an optimal subset of fitting cylinders from multiple candidates through the optimization of a metric. However, it is not aimed at reconstructing entire pipeline.

3. Overview

Our method takes as input a raw scan of a pipeline and outputs its part-based reconstruction. Thus, our method assumes that industrial plants are generally an assembly of mechanical parts. Here we focus on parts such as pipes, elbows, flanges, tees and crosses.

Besides parts types, their specific attributes govern their appearance in the general pipe reconstruction. In our experiments we consider parts length, radius and orientation. Note that in real scenes, other parts may be present such as rails, stairs, floors, etc. Our technique can incorporate additional parts in the same manner.

Given a point cloud, semantic segmentation is usually used to understand scene. Traditional methods [11, 36] use

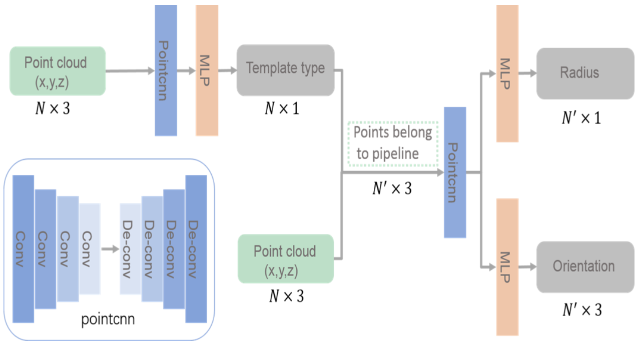

Figure 3. Our DeepPipe neural network architecture. Points are initially fed into a network which produces the part type of each point. Points are then filtered and fed to a second network, composed of two branches, that learns the radius and orientation of part types per point.

patch feature, such as normal vector and flatness of point neighborhood, to achieve segmentation. To couple semantic category and instance label into a single task, [9] introduced patch clusters as an intermediate representation between patches and semantic labels. The semantic segmentation is achieved along with labeling. [2] proposed a novel convolutional neural network architecture to get semantic label. It applies 2D convolutional neural network (CNN) on the extracted patch feature and depth maps of point cloud to get semantic label. In addition to the coordinates of points, it also needs the color as input. Deep neural network has achieved good results in signal reconstruction and inversion problems [15, 19]. Recently, it has also been designed to learn global and multi-scale point set features [17]. To process the point cloud directly using convolution, PointCNN [17] extends convolution from 2D to 3D by solving the problem of irregular and disordered point cloud and achieves better performance in classification. It is a general convolutional framework for learning feature of point clouds, which learns the order of convolution input mainly by the proposed x-transform. We use it to extract points feature.

Since our scanned scene is composed of specific parts, our technique first converts reconstruction into a recognition problem using neural networks. We use deep learning and design a CNN to learn a 3D point classification and regression. Specifically, each scan point is classified by part type and part radius label (our part radii are discrete classes). The part orientation is regressed using a direction 3D vector per point.

Given a classification of our point set into primitive parts, we compute point clusters by their labeling which define candidate parts in the scene. We then use graphs to process part relations in the scene. We first connect candidate parts arbitrarily and use a minimum spanning tree (MST) algorithm to obtain the correct primitive relations in the scene. This yields a skeleton graph with no loops that spans the scene.

We use the graph skeleton relations as well as part attributes to compute the final 3D model which reconstructs a subset of predefined parts in the scene. See Figure 2 for an overview of our method.

4. Technical Details

4.1. Deep Learning Pipes

We initially train a convolution network to predict for each point p in the scanned data S three labels: the part type it belongs to, the part radius and orientation. We use the per-point orientation vector to compute the part position in 3D space. While part types and radius are discrete terms, point orientation is continuous and thus is regressed using our network (Figure 4).

In pipeline design, pipe scenes are composed of pipe components and pipe support elements. In this work we choose to focus on pipe components and ignore supports such as floors, fences, etc. due to the problem magnitude. Nevertheless, it is easy to use our framework to add and remove components. To demonstrate our technique, we choose five types of pipe components as our primitives: pipe, flange, elbow, tee and cross. We also maintain a no-part label for points in the 3D scene belonging to parts outside the above five types.

We also use a discrete set of predefined radii for each component type as this is the common case in the industry. Thus, we have 5 times the number of radii number of classes. Utilization of discrete classes instead of continuous regression has also better accuracy and performance. To compute the part orientation we regress a normalized orientation vector per-point. Thus, we can compute the position, size and orientation of each part and fit it to the points.

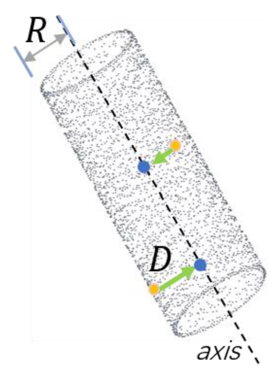

Figure 4. DeepPipes learn the radius R and the orientation vector D per scan point. D is also orthogonal to the displacement of a scanning point to the central axis of a pipe part.

Our network is illustrated in figure 3. It obtains as input a point cloud $p \in S$ where a point is defined by its position $p(x, y, z)$. To classify per-point primitive type, the top-left network branch uses PointCNN and learns a 6-channel feature map using multilayer perceptron (MLP) followed by a soft-max activation. We then use this classifier to filter our outliers, noise and points not belonging to our part labels (dashed box).

Next, we predict the per-point part radius and orientation in the top-right and bottom-right networks respectively. In Multi-task network [3], multiple learning tasks are solved at the same time, while exploiting commonalities and differences across tasks. As claimed in [3], we can enable our model to generalize better on original task by sharing representations between related tasks. Both radius and orientation are related to the displacement vector from the scan points to the part axis. Thus, we define a multi-task network that handles both classification and regression. Part radii range from 0.2 to 4.6 meters, with 23 discrete values specified by the pipe design standard. The classification branch outputs a 23-channel feature map followed by a softmax activation function (top-right). The regression branch (bottom-right) outputs 3-channel feature maps corresponding to the 3D orientation vector.

We perform multi-task training to train the full network simultaneously. We use cross-entropy loss on the classification outputs:

$$L_{CE}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_j^{(i)} \ln \hat{y}_j^{(i)},$$

where y is ground truth, $\hat{y}$ is predicted label, N is the number of samples and C is the number of categories, and L_2 loss on the regression output. In Multi-task network, we adopt a weight sharing framework between the two tasks, where tasks share the first few CNN layers, leading to better accuracy and convergence rates.

4.2. Relational Skeleton Graph

The network output is typically inconsistent in terms of per-point part types, radii and also noisy regression output. Specifically, adjacent points may be assigned different labels, especially in noisy parts and at boundaries between different part types. In this section we process our network output to obtain a coherent part assignment and fitting.

Given per point type and radii labels as well as 3D orientations, we compute primitive part candidates in the scene. As one of the most common clustering algorithms, density-based spatial clustering of applications with noise (DBSCAN) is a density-based clustering non-parametric algorithm, which groups together points that are closely packed together (points with many nearby neighbors). Using the parts center and axis (i.e., position and orientation) we proceed by clustering together parts based on their type, position and orientation attributes using DBSCAN. This yields clusters of candidate parts, reducing the number of candidates by the clusters. We then filter out points with no clusters and too small clusters as outliers and noise.

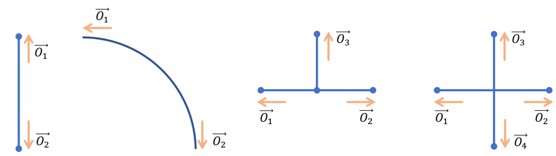

Figure 5. Illustration of our different part relations rules. Left-to-right are pipes, elbows, tees and crosses.

Given the pipe parts candidates denoted P, we build a corresponding graph $G(P, E)$ where each node $p_i \in P$ corresponds to a pipe part in the scene. For each part $p_i \in P$, we select its k-nearest neighbors $\{p_{i1}, p_{i2}, p_{ij}, ..., p_{ik}\}$ based on their centers Euclidean distance and define their connecting edges in the graph. We filter out edges with Euclidean distance higher than a threshold ε, as this defines too far parts.

We define the edge weight between two nodes in the graph as their Euclidean distance. We use edge weights to compute a minimum spanning forest $T = \{t_1, ..., t_i, ...\}$ which yields the pipeline skeleton graph of the scene. Minimum spanning forest is a union of the MST for connected components of a graph. Specifically, for each MST in the forest t_i, we compute its diameter (i.e., max distance path), remove it from t_i and add it to our skeleton graph. We then update the minimum spanning forest by recomputing trees after the diameter removal. This process repeats iteratively and computes long pipe paths as trees diameters until all parts are added to the skeleton graph. Our skeleton graph computation algorithm is summarized in Algorithm 1.

Algorithm 1: Compute skeleton graph

```
input  : candidate parts set P
output: pipe skeleton graph D

initialize G(P, E)
foreach part p ∈ P do
    compute k-nearest neighbors to p with distance
      ≤ τ1
calculate minimum spanning forest T of G
while P ≠ ∅ do
    foreach t ∈ T do
        calculate diameter path d of t
        add d to D
        remove all nodes p ∈ d from P
        update T
```

Finally, we refine the pipeline graph to conform to the following relations between parts (Figure 5):

- **pipe and flange** parts have two neighbors in the graph

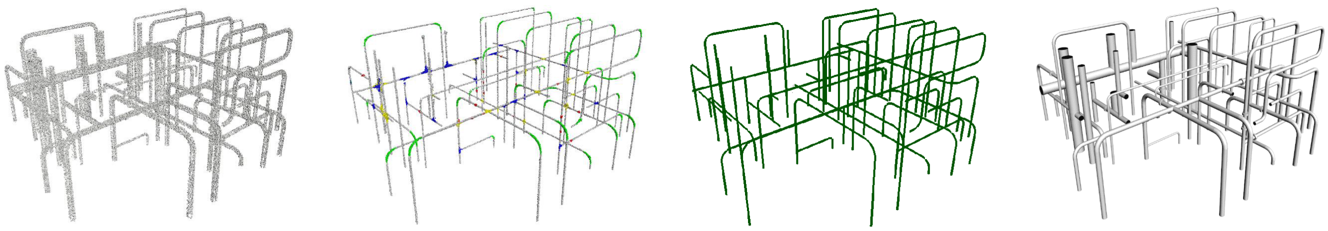

Figure 6. 3D reconstruction of a complex pipeline. Left-to-right are the input point cloud, DeepPipes segmentation and labeling (colors denote different part types), skeleton graph and 3D model reconstruction.

(at most);
neighbors endpoints form an angle close to straight within a certain threshold;

- **elbow** parts have two neighbors in the graph (at most); neighbors endpoints form a perpendicular angle;
- **tee** parts have three neighbors (exact); neighbor endpoints form angles either perpendicular or straight (forming a T-shape);
- **cross** parts have four neighbors (exact); neighbor endpoints form angles either perpendicular or straight (forming a cross shape);

In the last step, we replace graph nodes by the actual 3D part models and reconstruct the scene. For fine tuning, we readjust the parts fitting using iterative closest point (ICP) [1], which is an algorithm employed to minimize the difference between two clouds of points, and transform them to better fit the point data.

5. Results

To evaluate our method, we have used PointCNN neural networks for classification and regression tasks. Our networks consist of four convolutional layers, four deconvolutional layers and MLP. Each MLP has three layers. For MLP in primitive type classifier, the width of the three layers is 128, 128, 6. Similarly, the width for radius and orientation networks are 128, 128, 23 and 128, 128, 3, respectively. The radius and orientation networks are trained in parallel and consist of weight sharing connections since they perform similar tasks.

We trained our model on synthetic pipeline models. For this purpose, we implemented a 3D pipeline generator that resemble real-world models. We initially generate random skeleton graphs. Next, for each graph node, we randomly assign part type and radius labels as well as orientation. We generate the pipeline 3D model by fitting and assembling the correct parts together following our random graph and node labels.

We use a virtual scanner library to sample the 3D pipeline surface with points, resembling scanned data. For each scanned point, we acquire its part type, radius and orientation from its projection on the surface, yielding our ground truth scanned training data.

Our generator is implemented in Python, taking approx. 1 minute to generate an entire pipeline scene using a desktop PC with Intel(R) Core(TM) I7-7700K CPU, 4.20 GHz with 16-GB RAM. By this way, we create our DeepPipes training set consisting of 1750 different pipeline models ranging from 70K to 200K of scanned points per model.

We have implemented our DeepPipes on a desktop PC with Intel(R) Core(TM) I7-7700K CPU, 4.20 GHz with 16-GB RAM. We train our part segmentation network separately and then in parallel train the radius and orientation networks. Each of the two training steps take approx. 60 hours to converge. During testing time, for 180K points, it takes 8 seconds to run networks and get per-points labels, 10 seconds to compute MST paths and 25 seconds to fit parts and obtain reconstructed model. Table 1 summarizes our pipeline models in terms of number of scan points and number of different pipe parts.

We evaluate our technique both qualitatively and quantitatively using synthetic and real pipeline raw scans. In figure 1 we show the 3D reconstruction result of a scanned mid-scale pipeline plant. It consists of besides pipes also flanges, elbows and different connectors. Besides missing a tee connector and a flange (top and bottom parts of the image), our technique was able to accurately recover the entire model and parts.

Figure 6 demonstrates the full 3D reconstruction process of a complex pipeline scan with intermediate steps. Given a raw scan, we predict its segmentation into parts using DeepPipes. Colors represents different part types where gray-pipe, green-elbow, blue-tee, yellow-cross and red flange. Following is the relations graph and the final 3D reconstructed model.

In Figure 7 we compare our technique with different methods for processing 3D scanned pipelines. We evaluate our method based on the relevant indicators of the radius and the number of pipeline extractions. Considering the limitation of the radius range and radius accuracy, Liu *et al*. [22] method is the most suitable comparison method compared to other methods. We compare our method with

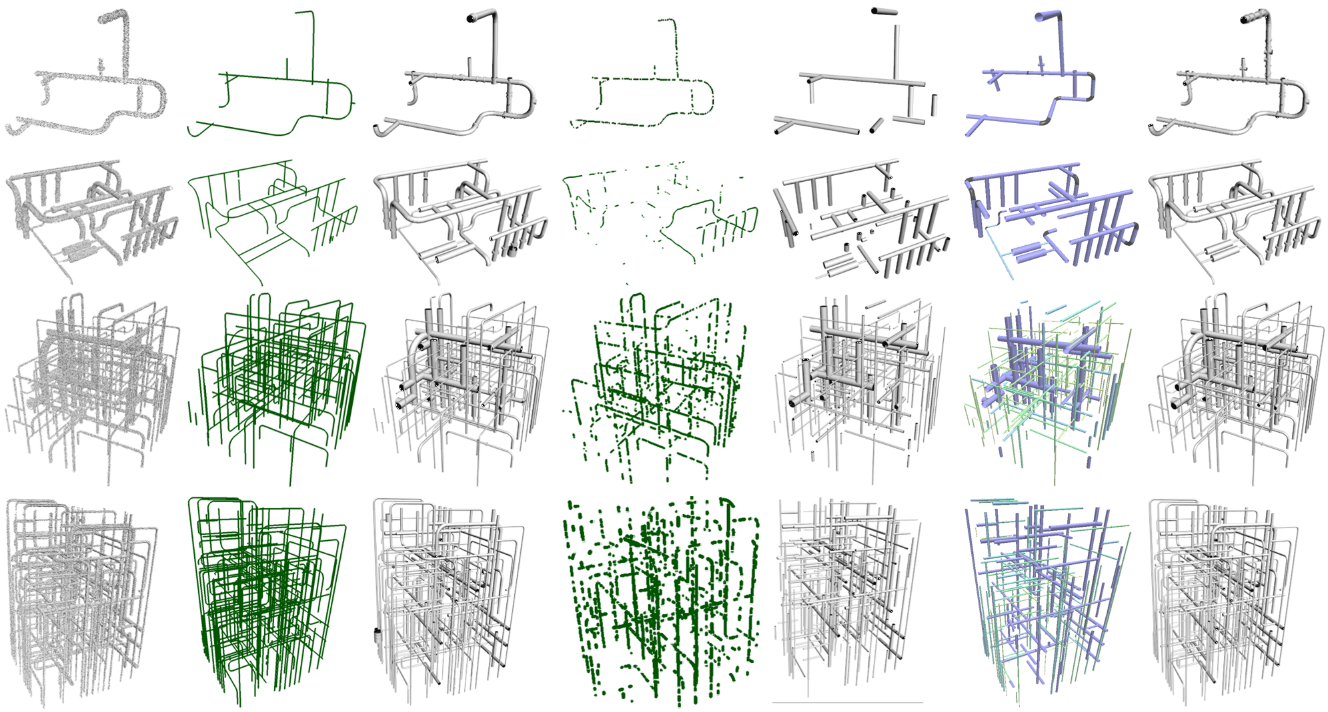

Figure 7. Comparisons on four pipelines of different complexity ranging from low to high (top-to-bottom rows resp.). Left-to-right are the scanned data, our skeleton, our reconstruction, Huang *et al*. [10] skeleton, Liu *et al*. [22] reconstruction, EdgeWise Plant [8] reconstruction and ground truth.

the method of Liu *et al*. [22] which detects and reconstructs pipes using their 2D projections on dominant planes. Similarly, we compare our technique to Huang *et al*. [10] which extracts skeletal structures from points using a robust L_1 approach. Their method was especially designed to handle points consisting of noise and large missing parts as is the case with pipeline scans. Therefore, we compare our skeleton computation with theirs. Finally, we also compare our results with a commercial software EdgeWise Plant [8].

Our comparison consists of four different scanned pipelines of different scale and complexity, ranging from simple small scale to complex large scale (top-to-bottom rows respectively).

The methods of Liu *et al*. [22] and Huang *et al*. [10] are both missing pipe parts when complexity and pipe density increases. EdgeWise Plant [8] involves manual interaction and therefore can obtain more accurate reconstructions. Note that both our skeleton extraction and reconstruction outperforms other methods. This is mainly due to our utilization of neural networks which yield accurate detection and segmentation of primitive parts in noisy scans.

To evaluate the robustness of our algorithm we introduce noise and sparsity in the scanned synthetic 3D pipelines (Figure 8). This simulates problems encountered in real world pipeline scanning such as occlusions, poor illumination and reflections resulting in high noise levels and missing parts.

Starting from a dense clean scan, we gradually increase per-point noise and sparsity by controlling the virtual scanner parameters. Specifically, sparsity level is controlled by the number of virtual cameras and the number of views per camera. We then add per-point Gaussian noise levels by adjusting the Gaussian parameters. In Figure 8 rows show increasing levels of scan sparsity and noise (top-to-bottom, resp.). Density levels are 100%, 80%, 65%,50%.

Table 2 summarizes the quantitative evaluation of our method compared to others on different noise levels. Results are showing that our method was able to generate good results suffering from a moderate decrease in quality that corresponds to the increase in noise and sparsity. In this comparison, our method still outperforms other techniques.

We have also applied our synthetically trained technique to real-world scanned pipelines. Figure 9 demonstrates reconstruction results on three real-world pipelines datasets. Top row, we reconstruct a 3D pipeline model of 600K points sampling a $150 \times 100 \times 150$ meters scene. Since there are many non-pipe points, our algorithm filters them out after the per-point part labeling step using our first PointCNN network. Middle row shows a pipeline plant of 190K points sampling a $160 \times 140 \times 170$ meters scene. And bottom row shows a similar pipeline plant of 220K points sampling a $160 \times 110 \times 110$ meters scene. Here a significant num-

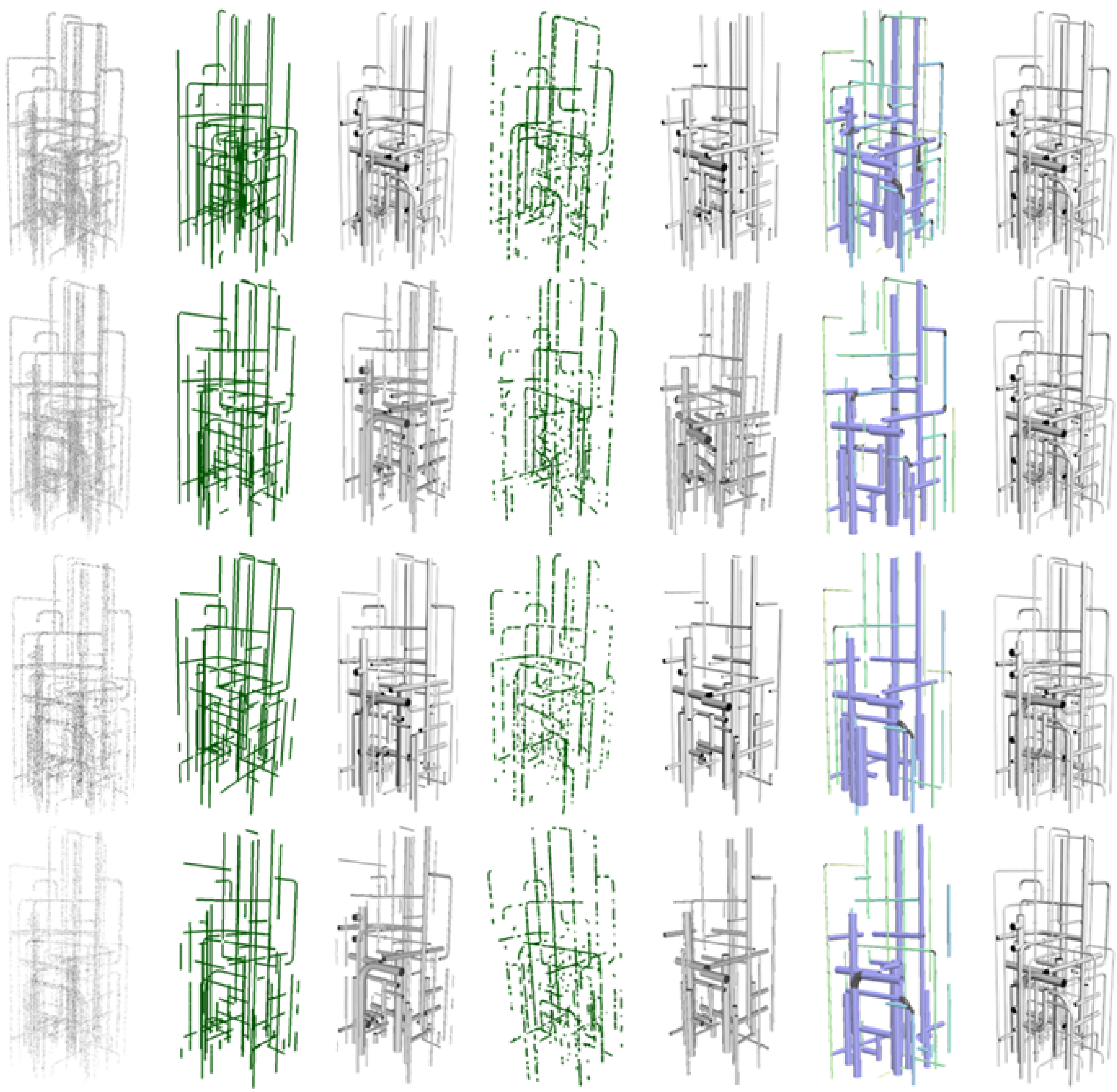

Figure 8. 3D pipeline reconstruction evaluation with different scan noise levels ranging from low to high (top-bottom rows resp.). Left-to-right are the scanned data, our skeleton, our reconstruction, Huang *et al*. [10] skeleton, Liu *et al*. [22] reconstruction, EdgeWise Plant [8] reconstruction and ground truth.

ber of points belong to floors and walls and stairs which are extracted and manually fitted for visual purposes.

To provide a quantitative evaluation of our method, we compare our reconstruction results with ground truth in the synthetic cases. We define an error metric that takes in account the absolute error between radius and orientation prediction and ground truth. Since absolute error is affected by radius size, we also normalize radius distance defining a relative error. Given a point p with r and $r^{'}$ being ground truth and predicted radii respectively, we define a relative distance, normalized by radius scale as: $E_{relative} = \frac{||r-r^{'}||}{r}$.

We also compute the recall ratio R_{ratio} and precision P_{ratio} for each scene. Let N_T be the number of total parts instances in the scene. N_{right} the number of true detected parts and N_{detect} is the total number of detected

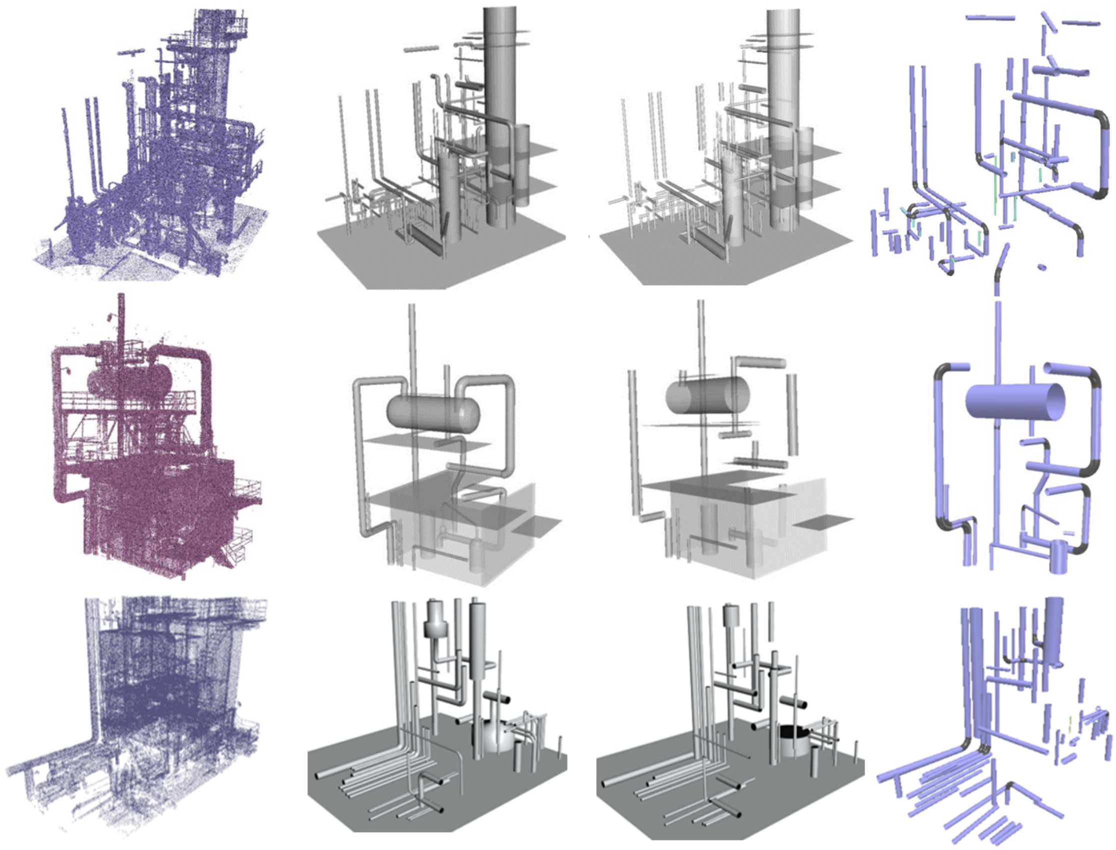

Figure 9. 3D reconstruction of two real-world pipelines (top-bottom rows). Left-to-right are the input scan, 3D reconstruction by our method, Liu *et al*. [22] and rightmost col. EdgeWise Plant [8].

parts. Then,

$$R_{ratio} = \frac{N_{right}}{N_T}$$

and

$$P_{ratio} = \frac{N_{right}}{N_{detect}}$$

We summarize the quantitative evaluation of our method in Table 3. Each row of the table provides precision and recall values of our technique compared to other methods. Note that Huang *et al*. method could not provide meaningful parts and we compute the error among skeletons. Similarly, since EdgeWise software does not provide parts with meaningful radii, we could not compute a meaningful error distance.

Similarly, we summarize our results on the real-world data in Table 4. Note that for real-world pipelines, we ask an expert to manually reconstruct the 3D pipeline from the scan and provide a ground truth.

6. Conclusions

In this work we take a prior-based approach for reconstruction of entire 3D pipelines from raw scans. In our approach, we learn recognition of parts in the scene, thus reducing the complexity of the general pipe reconstruction problem into a combination of part detection and model fitting problems. We utilize a convolutional network to learn 3D point cloud features and the classification into various classes. The pipe classification is noisy and we apply robust clustering and graph-based aggregation techniques to compute a coherent pipe model. Our method shows promising results on pipe models with varying complexity and density both in synthetic and real cases.

In terms of limitations, while our neural networks yield good classification and segmentation results, they do not consider part relations, connectivity and pipeline topology. Therefore, results are still incoherent and require further processing using clustering, graphs and MST computation.

Our method can be extended to some scenes assembled by basic components. There are many scenarios similar to the pipeline organization structure, such as the steel bars in the reinforced concrete at construction sites and the steel frame buildings mentioned in [33]. These structures are basically constructed according to fixed rules and consist of repeating parts. The proposed method can be extended to

Table 1. Data summary

model	#point	Pipe	Flang.	Elbow	Tee	Cross
Fig 7 Row1	47K	24	31	7	5	14
Fig 7 Row2	114K	93	23	22	23	0
Fig 7 Row3	331K	366	23	120	30	43
Fig 7 Row4	523K	619	78	189	68	61
Fig 8 Row1	231K	232	26	66	31	21
Fig 8 Row2	187K	232	26	66	31	21
Fig 8 Row3	150K	232	26	66	31	21
Fig 8 Row4	115K	232	26	66	31	21
Fig 9 Row1	599K	198	0	84	29	2
Fig 9 Row2	187K	33	0	16	0	0
Fig 9 Row3	213K	76	0	32	7	1

Table 2. Quantitative evaluation of synthetic pipelines under various noise levels. (Figure 8)

Models	Row1			Row2			Row3			Row4		
	Error	Precision	Recall	Error	Precision	Recall	Error	Precision	Recall	Error	Precision	Recall
Liu	0.0400	0.7958	0.5889	0.0581	0.6982	0.5089	0.0689	0.6064	0.3800	0.0815	0.5185	0.2489
Huang	0.6087	nan	nan	0.7361	nan	nan	0.8015	nan	nan	0.9169	nan	nan
EdgeWise	nan	0.8966	0.5778	nan	0.7147	0.4956	nan	0.6266	0.3356	nan	0.6698	0.3200
Ours	**0.0098**	**0.8981**	**0.7244**	**0.0168**	**0.8833**	**0.6733**	**0.0281**	**0.7273**	**0.5333**	**0.0302**	**0.6759**	**0.4355**

reconstruct these scenes by detecting the types of basic parts and obtaining the relationships between them.

In future work, we plan to investigate a full neural network for 3D pipeline reconstruction. Using recurrent neural network (RNN) and Long short-term memory (LSTM) [7] architectures, we may incorporate neighborhood relations and topology in the scanned pipeline processing framework.

Acknowledgments

This work was supported by National Key Research and Development Project (2017YFB1002603) end the NSFC Project (61772318, 61772016).

References

[1] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992. 6

[2] J.-X. Cai, T.-J. Mu, Y.-K. Lai, and S.-M. Hu. Deep point-based scene labeling with depth mapping and geometric patch feature encoding. *Graphical Models*, 104:101033, 2019. 4

[3] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 5

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002. 3

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231, 1996. 2

[6] R. Gal, A. Shamir, T. Hassner, M. Pauly, and D. Cohen-Or. Surface reconstruction using local shape priors. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, pages 253–262, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. 1, 3

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 10

[8] http://www.clearedge3d.com/. Clearedge3d: Edgewise plant, 2012. 2, 7, 8, 9

[9] S.-M. Hu, J.-X. Cai, and Y.-K. Lai. Semantic labeling and instance segmentation of 3d point clouds using patch context analysis and multiscale processing. *IEEE transactions on visualization and computer graphics*, 2018. 4

[10] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen. L1-medial skeleton of point cloud. *ACM Trans. Graph.*, 32(4):65–1, 2013. 7, 8

[11] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *Advances in neural information processing systems*, pages 244–252, 2011. 3

[12] T. Le and Y. Duan. A primitive-based 3d segmentation algorithm for mechanical cad models. *Computer Aided Geometric Design*, 52:231–246, 2017. 2, 3

[13] J. Lee, H. Son, C. Kim, and C. Kim. Skeleton-based 3d reconstruction of as-built pipelines from laser-scan data. *Automation in Construction*, 35:199 – 207, 2013. 2

[14] G. Li, L. Liu, H. Zheng, and N. J. Mitra. Analysis, reconstruction and manipulation using arterial snakes. *ACM Trans. Graph.*, 29(6):152, 2010. 3

Table 3. Quantitative evaluation of synthetic pipelines (Figure 7)

Models	Row1			Row2			Row3			Row4		
	Error	Precision	Recall	Error	Precision	Recall	Error	Precision	Recall	Error	Precision	Recall
Liu	0.0508	0.6429	0.5556	0.0401	0.6273	0.4286	0.0426	0.7429	0.5342	0.0630	0.5938	0.5054
Huang	0.6261	nan	nan	0.7812	nan	nan	0.6857	nan	nan	0.7991	nan	nan
EdgeWise	nan	0.6711	0.6296	nan	0.6720	0.5217	nan	**0.8635**	0.5308	nan	0.6167	0.3734
Ours	**0.0081**	**0.8169**	**0.7160**	**0.0231**	**0.9177**	**0.9006**	**0.0177**	0.8565	**0.7106**	**0.0277**	**0.7616**	**0.8276**

Table 4. Quantitative evaluation of real pipelines(Figure 9)

Models	Row1		Row2		Row3	
	Precision	Recall	Precision	Recall	Precision	Recall
Liu	0.5618	0.5402	0.7500	0.3061	0.6779	0.4090
EdgeWise	**0.7500**	0.2797	0.7692	0.6122	**0.7936**	0.4545
Ours	0.7290	**0.7009**	**0.7750**	**0.6327**	0.6938	**0.6181**

[15] S. Li, B. Liu, Y. Ren, Y. Chen, S. Yang, Y. Wang, and P. Jiang. Deep learning inversion of seismic data. *arXiv preprint arXiv:1901.07733*, 2019. 4

[16] S. Li, Y. Yao, T. Fang, and L. Quan. Reconstructing thin structures of manifold surfaces by integrating spatial curves. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2887–2896, 2018. 3

[17] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. 4

[18] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4):52:1–52:12, July 2011. 2, 3

[19] B. Liu, Q. Guo, S. Li, B. Liu, Y. Ren, Y. Pang, X. Guo, L. Liu, and P. Jiang. Deep learning inversion of electrical resistivity data. *IEEE Transactions on Geoscience and Remote Sensing*, 2020. 4

[20] L. Liu, D. Ceylan, C. Lin, W. Wang, and N. J. Mitra. Image-based reconstruction of wire art. *ACM Trans. Graph.*, 36(4):63:1–63:11, July 2017. 3

[21] L. Liu, N. Chen, D. Ceylan, C. Theobalt, W. Wang, and N. J. Mitra. Curvefusion: Reconstructing thin structures from rgbd sequences. *ACM Trans. Graph.*, 37(6):218:1–218:12, Dec. 2018. 3

[22] Y.-J. Liu, J.-B. Zhang, J.-C. Hou, J.-C. Ren, and W.-Q. Tang. Cylinder detection in large-scale point cloud of pipeline plant. *IEEE transactions on visualization and computer graphics*, 19(10):1700–1707, 2013. 2, 6, 7, 8, 9

[23] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana. Automatic reconstruction of tree skeletal structures from point clouds. *ACM Transactions on Graphics (TOG)*, 29(6):151, 2010. 3

[24] R. Maalek, D. D. Lichti, R. Walker, A. Bhavnani, and J. Y. Ruwanpura. Extraction of pipes and flanges from point clouds for automated verification of pre-fabricated modules in oil and gas refinery projects. *Automation in Construction*, 103:150–167, 2019. 3

[25] T. Martin, J. Montes, J.-C. Bazin, and T. Popa. Topology-aware reconstruction of thin tubular structures. In *SIGGRAPH Asia 2014 Technical Briefs*, SA '14, pages 12:1–12:4, 2014. 3

[26] C. H. P. Nguyen and Y. Choi. Comparison of point cloud data and 3d cad data for on-site dimensional inspection of industrial plant piping systems. *Automation in Construction*, 91:44 – 52, 2018. 2

[27] A. K. Patil, P. Holi, S. K. Lee, and Y. H. Chai. An adaptive approach for the reconstruction and modeling of as-built 3d pipelines from point clouds. *Automation in construction*, 75:65–78, 2017. 2

[28] M. Pistellato, F. Bergamasco, A. Albarelli, and A. Torsello. Robust cylinder estimation in point clouds from pairwise axes similarities. In *8th International Conference on Pattern Recognition Applications and Methods*, pages 640–647, 01 2019. 3

[29] R. Qiu, Q.-Y. Zhou, and U. Neumann. Pipe-run extraction and reconstruction from point clouds. In *European Conference on Computer Vision*, pages 17–30. Springer, 2014. 2

[30] T. Rabbani and F. Van Den Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. *Isprs Wg Iii/3, Iii/4*, 3:60–65, 2005. 2

[31] R. Schnabel, P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2):503–512, 2009. 1, 3

[32] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007. 1, 2, 3

[33] M. Song and D. Huber. Automatic recovery of networks of thin structures. *2015 International Conference on 3D Vision*, pages 37–45, 2015. 3, 9

[34] T.-T. Tran, V.-T. Cao, and D. Laurendeau. Extraction of cylinders and estimation of their parameters from point clouds. *Computers & Graphics*, 46:345–357, 2015. 3

[35] B. Ummenhofer and T. Brox. Point-based 3d reconstruction of thin objects. In *2013 IEEE International Conference on Computer Vision*, pages 969–976, 2013. 3

[36] G. Vosselman. Point cloud segmentation for urban scene classification. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, 1:257–262, 2013. 3