Slicing-Tracking-Detection: Simultaneous Multi-Cylinder Detection from Large-scale and Complex Point Clouds

Zhuheng Lu, Weiwei Mao, Yuewei Dai, Weiqing Li, and Zhiyong Su

Abstract—Multiple cylinders detection from large-scale and complex point clouds is a historical but challenging problem, considering the efficiency and accuracy. We propose a novel framework, named slicing-tracking-detection (STD), that detects multiple cylinders accurately and simultaneously from point clouds of large-scale and complex process plants. In this framework, the 3D cylinder detection problem is reformulated as a cylinder ingredients tracking task based on multi-object tracking (MOT). Firstly, we generate slices from the input point cloud, and render them to slice sequence. Then, the cycle of a cylinder is modeled with a Markov Decision Process (MDP), where the ingredient is tracked with a template and the miss tracking is associated with ingredient proposals through reinforcement learning. Finally, by applying MDP for each cylinder, multiple cylinders can be detected simultaneously and accurately. Extensive experiments show that the proposed STD framework can significantly outperform the state-of-the-art approaches in efficiency, accuracy, and robustness. The source code is available at http://zhiyongsu.github.io.

Index Terms—Cylinder detection, markov decision process, primitive fitting, 3D reconstruction

1 Introduction

Reconstruction of process plants is a crucial task for many applications in the petrochemical industry, such as facilities maintenance, plant rehabilitation, training and disaster simulation using virtual reality techniques. Due to the rapid development in light and ranging (LIDAR) technologies, very large sets of dense and accurate 3D points can be collected easily and quickly for large-scale and complex process plants. Therefore, generating as-built models of process plants from large-scale 3D point clouds produced by laser scanners is becoming standard practice in the petrochemical industry. Most process plant objects are composed of basic primitives such as planes, spheres, cylinders, tori, and cones. Cylinders account for the greatest proportion in these objects such as pipes, sleeves, connectors [1], [2], [3], as shown in Fig.1. Therefore, cylinder detection is the core task in the reconstruction of process plants.

Point clouds of process plants characterize their large-scale and complex internal structure, which leads to serious problems in the process of cylinder detection. First, the plant consists of hundreds of components, and its number of points can reach the magnitude of tens of millions easily, which leads to high computational complexity. Second, hundreds and thousands of cylinders with different sizes and orientations are contained in realistic process plants. For example, a medium-sized process plant usually contains hundreds of pipelines and each pipeline is composed of multiple pipes (cylinders). Third, the laser-scanned data inevitably suffers from the missing region due to limited viewing angles, occlusions, or unstable measurement in the texture-less region or specular materials. Therefore, in the commercial sphere, the level of automation in the cylinder detection is still limited, which reveals that accurately and efficiently detecting cylinders in large-scale point clouds remains a challenging problem.

(a) Original plant

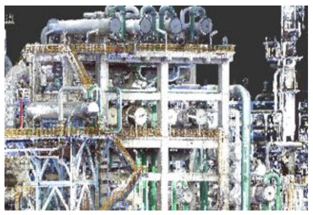

(b) Laser-scanned point clouds

Fig. 1. Three-dimensional process plant model.

Various studies have been conducted on cylinder detection from point clouds. Some algorithms detect cylinders in 3D space. The three fundamental techniques are random sample consensus (RANSAC), Hough transform, and deep learning-based methods. RANSAC methods [1], [4], [5] detect one cylinder from the data set at a time. These methods are applicable to different geometric primitives but sensitive to noise. Besides, users need to set several thresholds that vary from model to model. Approaches based on Hough transform [6], [7] are known as voting methods with high computational requirements. The time and space complexities caused by accumulator discretization actually degrade performance, especially for the large-scale and complex

Z. Lu, W. Mao, and Z. Su are with the Visual Computing Group, School of Automation, Nanjing University of Science and Technology, Nanjing, Jiangsu Province 210094, P.R. China (e-mail: lzharsenal@163.com, banyimao@163.com, su@njust.edu.cn)
Y. Dai is with the School of Electronic and Information Engineering, Nanjing University of Information Science and Technology, Nanjing, Jiangsu Province 210044, P.R. China (e-mail: dywjust@163.com).
W. Li is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu Province 210094, P.R. China (e-mail: li_weiqing@njust.edu.cn).
Manuscript received 00 00, 0000; revised 00 00, 0000. (Corresponding author: Zhiyong Su.)

point clouds. Deep neural networks have been proposed to solve the primitive fitting problem in both supervised [8], [9] and unsupervised [10], [11] settings. These works either put emphasis on fitting to specific point sets, or restrict the number of input points. They cannot directly scale up to large scenarios due to their high computational and memory costs. Another thread of cylinder detection methods can be simplified to a circle extraction problem in 2D space. Projection methods [2], [12] project the point clouds onto a plane or unit hemisphere and detect circles in 2D space. However, in practice, these methods not only cost large calculation but also are sensitive to noise and outliers. Furthermore, the adverse influence is even more under the large-scale point clouds with complex internal structure, which poses challenges to the cylinder detection from point clouds of large-scale and complex process plants.

In this paper, we propose a novel simultaneous multiple cylinders detection framework, termed slicing-tracking-detection (STD), for large-scale and complex point clouds of process plants. The proposed framework consists of three parts: slicing, tracking, and detection. In the slicing stage, we generate slice sequences from the point cloud of process plant in an arbitrary direction and render these slices to 2D images. The slicing strategy is beneficial for complex occlusion between primitives in large-scale and complex scenarios. Second, in the tracking stage, we formulate the multi-cylinder detection problem as a multiple object tracking (MOT) task taking advantage of continuity between slices. We suppress noise and interference terms except cylinder ingredients to robustly associate ingredient detection on a new slice with previously tracked ingredients. In the detection stage, we create a Markov Decision Process (MDP) for each cylinder, so we can detect cylinders with different sizes and orientations simultaneously through the slice sequence. The main contributions of this paper are highlighted as follows:

- We propose a novel simultaneous multi-cylinder detection framework, in which the 3D cylinders detection problem is formulated as a cylinder ingredients tracking task through the multi-object tracking method, for large-scale and complex point clouds.
- We employ the MDP to model the cycle of each cylinder, thus allowing for tracking every cylinder in its entirety through slice sequence generated from input point cloud.
- To track cylinder ingredients with deformable appearance, various features of cylinder ingredients are designed and integrated into the data association via reinforcement learning.
- The proposed STD surpasses the state-of-the-art cylinder detection methods on multiple large-scale point clouds with regard to efficiency, accuracy, as well as robustness.

The rest of the paper is organized as follows. We discuss related work that has been conducted on cylinder detection from point clouds and multiple object tracking in Section 2. Section 3 provides an overview of the proposed method for cylinder detection. In Section 4, Section 5, and Section 6, we describe our STD algorithm in detail. After that, we present experimental results in Section 7. Finally, conclusions and recommendations for future research are given in Section 8.

2 Related Work

In this section, we review a number of previous works on cylinder detection from 3D point clouds and MOT.

2.1 Cylinder Detection

To efficiently detect cylinders from point clouds, various kinds of cylinder detection methods have been proposed over the past decades [8], [13]. In 3D space, existing cylinder detection techniques can be classified into three broad categories: Hough transform, RANSAC, and deep learning-based methods. Besides, some methods convert cylinder detection in 3D space to a circle extraction problem or an optimization problem with a priori.

Hough transform [14] casts a vote for each point in the input point clouds to detect which points could possibly contain any geometrical object like cylinders [6], [7] and sphere [15], [16]. Hough transform detects cylinder by calculating five parameters: a 2D accumulator for orientation and a 3D accumulator for the position and radius in the parametric space. Some authors have improved Hough transform to speed up the procedure by splitting and pruning the parametric space. Patil et al. proposed area-based adaptive Hough transform to estimate multiple cylinder orientation [17]. However, the time and space complexities caused by accumulator discretization both restrict these methods to deal with point clouds of large-scale and complex process plants.

RANSAC fits a model by using random sampling with the minimum number of data points [4], [5]. Chaperon et al. combined the Gaussian sphere with the RANSAC to detect cylinders and estimate their parameters [18]. This method divides the extraction of a cylinder into two steps: first, the direction of a cylinder is estimated by a plane on its Gaussian image, and in the next step, the cylinder size and position are extracted using random sampling. Tran et al. added a validation step into RANSAC to extract multiple cylinders at the same time [1]. However, RANSAC-based methods suffer from the problem of low efficiency, since they detect cylinders sequentially, especially for the large-scale point cloud. Besides, the detection results of RANSAC depend on the initial selection of points, especially when there are noisy data and outliers.

Deep learning-based researches have been proposed to solve the primitive detection problem both in supervised [8], [9] or unsupervised [10], [11] way. Li et al. introduced an end-to-end neural network that can detect a varying number of primitives at different scales [8]. Zou et al. proposed an LSTM-based architecture that predicts boxes given input depth images [9]. Tulsiani et al. presented a volumetric network that predicts a fixed number of cuboids to describe an input 3D shape [10]. Sharma et al. proposed a recurrent neural network that parses input shapes into their constituent modeling primitives [11]. These approaches achieve impressive results for primitive detection. However, almost all of them are limited to tiny 3D point clouds (e.g., 4k points), and cannot be directly extended to large-scale point clouds (e.g., millions of points) due to their

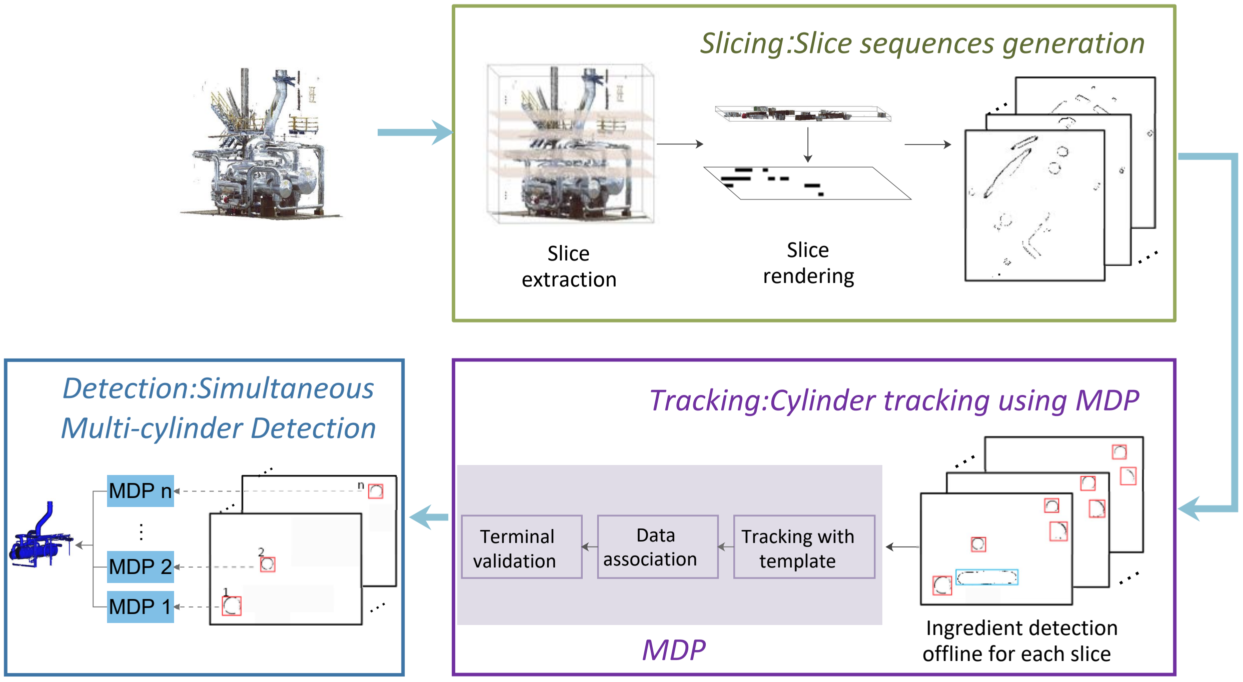

Slicing:Slice sequences generation
Slice extraction
Slice rendering
Detection:Simultaneous Multi-cylinder Detection
MDP n
MDP 2
MDP 1
Tracking:Cylinder tracking using MDP
Terminal validation
Data association
Tracking with template
MDP
Ingredient detection offline for each slice

Fig. 2. Overview of our slicing-tracking-detection pipeline. Given the process plant point cloud, slice sequence is firstly generated. Secondly, the cylinder is tracked with MDP through the slice sequence. Ingredients between adjacent slices are tracked with templates and the miss tracking is associated with ingredient proposals using reinforcement learning. Finally, all cylinders are detected simultaneously by employing multiple MDPs.

high computation and memory costs. For a large-scale point cloud, which usually consists of hundreds of objects, the existing local feature learners are incapable of capturing complex structures due to their limited size of receptive fields. Besides, most existing local feature learners usually rely on computationally graph construction and expensive kernelisation, thereby being unable to process the massive number of points. Furthermore, training of such networks typically requires a large number of training models. Thus, lacking training data and high labeling costs also limit their application to large-scale point clouds.

Recently, several methods take dimensionality reduction into consideration, so the cylinder extraction in 3D space can be simplified to a circle extraction problem in 2D space. Liu et al. projected points of the model onto the plane and detected circles on a 2D plane [2]. Arajo et al. projected the point clouds onto a set of directions over the unit hemisphere and detected circular projections [12]. Given the cylinder’s axis, Maalek et al. reduced pipe extraction to the points extraction of a circle on the plane [19]. Ahmed et al. re-sampled the point cloud by slicing and detect circles using Hough transform [7]. However, the limitation of existing projection methods is that direct projection can not handle the occlusions between complex primitives. Besides, they assumed that the cylinders were run in the three main orthogonal axes. Bey et al. also proposed a method that fits an optimized cylinder model to the scanned points using a priori CAD model of the plants [20]. However, except the priori CAD model, manual interactions are also necessary to handle the complex scene.

Although the mentioned methods show promise in detecting cylinders from point clouds, computational complexity, accuracy and robustness still require further investigation to provide a comprehensive and generalizable solution to the multi-cylinder detection from large-scale and complex point clouds.

2.2 Multiple Object Tracking

The MOT focuses on locating multiple objects, maintaining their identities, and yielding their individual trajectories given an input video. MOT is widely used in computer vision research applications, including video surveillance, traffic detection, and robotic assistance, etc. Generally, MOT methods can be classified into Detection-Based Tracking (DBT) and Detection-Free Tracking (DFT). Recent research in MOT has focused on DBT, where the main task is the data association problem in linking object detections to form correct trajectories. DFT requires manual initialization of a fixed number of objects in the first frame, then localizes these objects in subsequent frames.

DBT is more popular because new objects are discovered and disappearing objects are terminated automatically. On the one hand, Berclaz et al. [21], Shitrit et al. [22], Li et al. [23], Park et al. [24], Valsangkar et al. [25], and Zhang et al. [26] solved the MOT problem from a global optimization perspective with a flow graph. On the other hand, online methods have also been proposed such as Oh et al. [27] and Khan et al. [28], in which they utilized Markov Chain Monte Carlo (MCMC) to estimate the association. Bae et al. [29] , Kalal et al. [30] and Kim et al. [31] introduced learning to track into MOT. Milan et al. [32] used Recurrent Neural Networks (RNN) to encode the state-space and solve the association problem. In our work, we extend and improve the MDP framework for MOT proposed in [33], which is an online method that uses reinforcement learning to solve the data association problem.

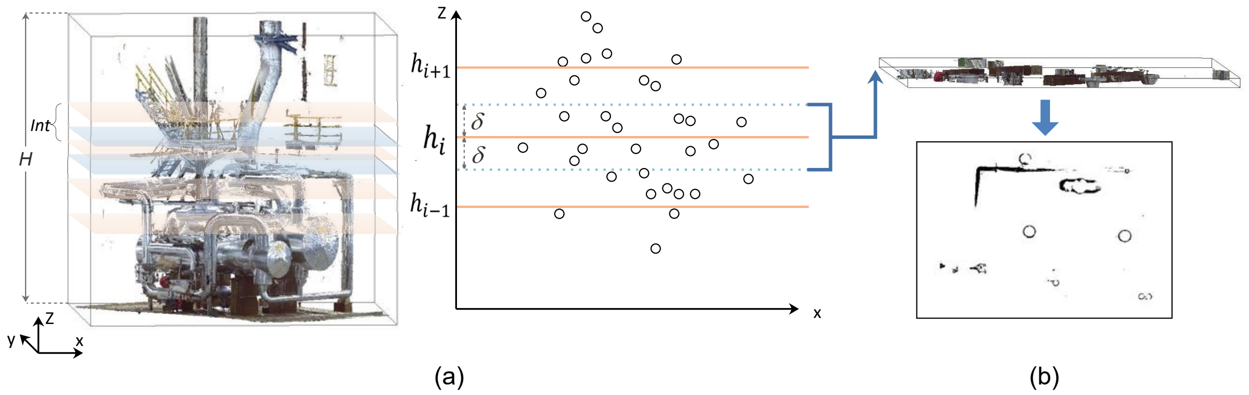


Fig. 3. An example of slice sequence extraction. (a) Slice extraction from the point cloud of a process plant. (b) Example of slice rendering.

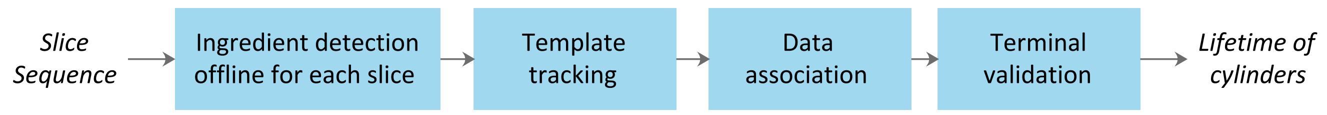


Fig. 4. Diagram for cylinder cycle modeling using MDP. Cylinder ingredients are firstly detected offline for each input slice. Then, *Online* ingredients are tracked with templates and *Offline* ingredients are associated with proposals. Finally, the cylinder terminal is validated by backtracking the sequence.

3 Algorithm Overview

The proposed STD framework covers three major steps: slicing, tracking, and detection, as illustrated in Fig.2.

The first phase is slice sequence generation. For each input point cloud, we firstly create continuous slices with the cross section parallel to the ground. Then, we render the extracted points of each slice into images. This mapping from 3D space to 2D space contributes significantly to reducing the computational complexity.

The second phase is cylinder tracking from the slice sequence. We convert cylinder detection in 3D space into cylinder ingredients tracking from the slice sequence in 2D space. Inspired by MOT in videos, we propose a novel algorithm for extracting multiple cylinders in the same sequence. Due to the relevance between slices, the methodology of MDP [34] is introduced to describe the correlation through the sequence. We formulate an episodic task for each cylinder, where the cycle of a cylinder is the length of the episode and modeled with an MDP. Once one cylinder is traced completely, initial and terminal states are tagged to determine the episode of the cylinder.

The third phase is simultaneous multi-cylinder detection with MDPs. We create an MDP for each cylinder based on the method in the second phase, so we can detect multiple cylinders simultaneously through the slice sequence. The appearance and disappearance of the cylinders can be treated as state transitions in the MDP.

4 Slice Sequence Generation

The slice sequence generation algorithm contains two major steps, as shown in Fig.3. First, a set of slices is extracted from the input point cloud. Then, these slices are used as rendering primitives to generate slice sequence $\mathcal{X} = \{x_i\}_{i=1}^{L}$, where L is the number of slices.

4.1 Slice Extraction

The input point cloud is cut into slices based on specified parameters: the interval Int and the section neighborhood δ. It should be pointed out that, in theory, the slice extraction direction can be arbitrary. In this paper, we choose the plane parallel to the ground as the cross section to extract the slices, since the majority of pipelines and equipment in process plants are perpendicular or parallel to the ground.

Given a process plant point cloud, we first calculate its axis-aligned bounding box. Then, the set of points is divided into $L = floor(H/Int)$ blocks perpendicular to the z-axis through $L-1$ cross sections, where H is the height of the bounding box. Finally, the neighboring points of the i-th cross section with its height h_i, whose z-value is within $[h_i - \delta, h_i + \delta]$, are selected as the intersection points between the i-th cross section and point cloud. This strategy is able to reliably balance the tracking efficiency and detection precision. In that case, sampling at intervals effectively determines the number of slices. And, neighboring points selection obtains the reliable projection.

4.2 Slice Rendering

For each cross section, we render its intersection points to a $w \times h$ slice image x_i by projecting these points into the cross section plane using the following projection function.

$$\begin{aligned} r &= (x - x_{min}) * w/(x_{max} - x_{min}) \\ c &= (y - y_{min}) * h/(y_{max} - y_{min}) \end{aligned} \tag{1}$$

where (x, y) denotes x-coordinate and y-coordinate of a 3D point, and (r, c) denotes the 2D pixel position of its projection. (x_{min}, x_{max}) and (y_{min}, y_{max}) denote the coordinates of the extreme point in the x-axis and y-axis, respectively. Each projected point is rendered as a black pixel. Then we can get the final image sequence of the input point cloud and refer to it as the slice sequence $\mathcal{X} = \{x_i\}_{i=1}^{L}$. We record the mapping relation between slices and 2D images which will be used to recover cylinders from the slice sequence in the following section. Fig.3 illustrates the slice generation process in the z-axis.

5 Cylinder Tracking Using MDP

In this paper, cylinder detection in 3D space is formulated as the cylinder ingredients tracking and association among 2D slice sequence, as shown in Fig.4. The cycle of a cylinder is modeled with an MDP. For every slice x_i in a sequence $\mathcal{X}$, the cylinder ingredient detector first detects all cylinder ingredients offline. Then, cylinder ingredients are tracked based on the ingredient templates through the sequence. Generally, cylinder ingredients and detected proposals need to be associated when the cylinder ingredients are not constantly tracked in the tracking process. Finally, the cylinder terminal is validated by backtracking the sequence.

5.1 Cylinder Cycle Modeling with MDP

MDP is appropriate for dynamic environments where an agent needs to perform certain tasks by making decisions and executing actions sequentially. In our framework, we consider a single cylinder tracker as an agent in MDP, whose task is to track the entire cylinder. Then, we learn an optimal policy for the MDP with reinforcement learning, and employ multiple MDPs to track multiple cylinders. The cycle of a cylinder is described as an MDP which consists of the tuple $(\mathcal{S}, \mathcal{A}, T(\cdot,\cdot), R(\cdot,\cdot), \pi(\cdot,\cdot))$, where:

- States $s \in \mathcal{S}$ encode the status of the target.
- Actions $a \in \mathcal{A}$ dictate the actions that can be taken.
- The state transition function $T : \mathcal{S} \times \mathcal{A} \longmapsto \mathcal{S}$ defines the state transition based on an action.
- The real-valued reward function $R : \mathcal{S} \times \mathcal{A} \longmapsto \mathbb{R}$ describes the immediate reward received after executing action a in state s.
- The policy $\pi : \mathcal{S} \longmapsto \mathcal{A}$ is a mapping from the state space $\mathcal{S}$ to the action space $\mathcal{A}$ in MDP.

The state space $\mathcal{S}$ is composed of four subspaces: *Positive*, *Negative*, *Online*, and *Offline*, which represent four statuses of the cylinder episode, as illustrated in Fig.5. The *Positive* state and *Negative* state represent the initial and terminal state, respectively. Besides, the cylinder ingredient is in an *Online* state when it is tracked in the tracking phase. On the contrary, the cylinder ingredient is in an *Offline* state when it is not tracked. Thus, cylinder ingredient tracking between consecutive slices can be considered as a decision-making process in these four states.

The policy π is a mapping from the state space $\mathcal{S}$ to the action space $\mathcal{A}$ in MDP. Given the current state of the ingredient, a policy is the probability distribution of actions to be taken. Equivalently, the decision-making in MDP is performed by following a policy. The goal of policy learning is to find a policy that maximizes the total rewards obtained.

5.2 Cylinder Ingredient Detection

Cylinder ingredients in our issue are the intersection patches between cylinders and cross sections in 2D slice images. For our ingredient detection framework, we generate the

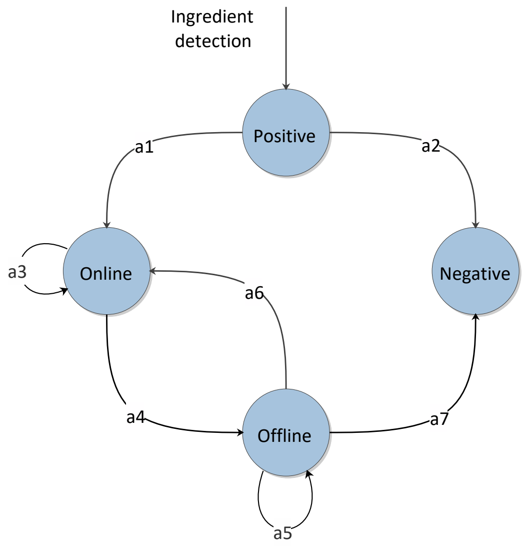


Fig. 5. The MDP in our framework. Cylinder ingredient tracking between adjacent slices can be considered as a decision-making process in four states: *Positive*, *Negative*, *Online*, and *Offline*. a_i dictates the action for state transition.

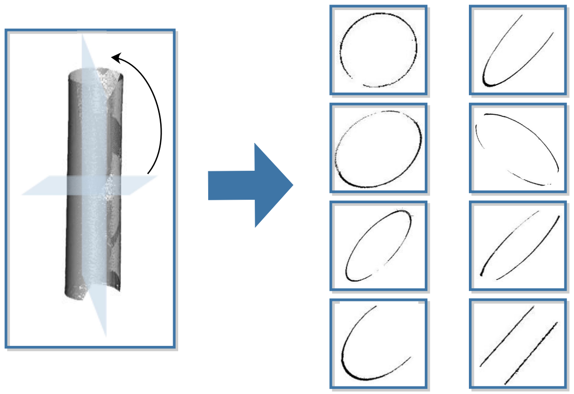


Fig. 6. An example of different cylinder ingredients in slices with different cutting directions.

training data in advance. Given the input sequence $\mathcal{X}$, the cylinder ingredient detector first selects cylinder ingredient proposals $\mathcal{D}_i = \{d_{ik}\}_{k=1}^{N_i}$ in each slice x_i offline, where N_i is the number of proposals in x_i. These proposals will be used in data association in the following section. Then, the MDP determines whether the proposal in a *Positive* state belongs to a cylinder or not.

5.2.1 *Training Data Generation*

To train the cylinder ingredient detector and sort the proposals, a variety of cylinders with different radiuses and lengths are selected from point clouds of different process plants to generate the training data. The selected cylinders are both intact and perforated to ensure the integrity of samples. For each input cylinder, we sample the neighboring points with the cross section similar to the slice generation method in Section 4.1. To obtain the ingredients with different shapes, the cross section crosses the cylinder from different angles and positions. Specifically, with the fixed angle interval, the cross section is rotated on a rotation axis from the initial

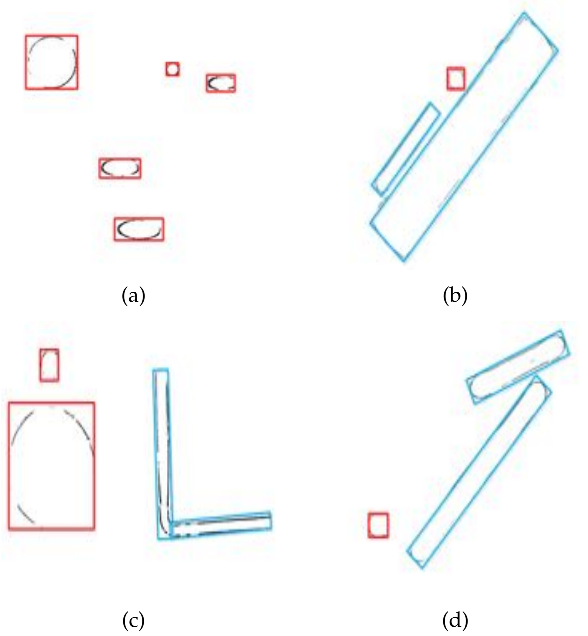


Fig. 7. Four examples of cylinder ingredients in slices. Ingredients with red boxes belong to invariant class L_{in}, and ingredients with blue boxes belong to the variant class L_{va}.

position, as shown in Fig.6. Then, the initial position of the cross section is translated from the bottom to the center of the cylinder with a plane step, while the rotation axis is translated with an axis step in the cross section. Then, the sampled points are rendered to an image which is normalized simultaneously.

In addition to the real datasets described above, synthetic cylinder point clouds are also generated in our approach. The synthetic cylinder dataset is automatically generated with different radiuses, lengths, and densities. Likewise, we generate training data with the synthetic dataset.

5.2.2 *Proposals Detection*

A cylinder ingredient detector is presented to detect all deformable proposals $\mathcal{D}_i = \{d_{ik}\}_{k=1}^{N_i}$ in each slice x_i as cylinder ingredient candidates. Due to the diverse orientation, the appearance of the cylinder ingredient in slices may be a variety of regular and irregular shapes. Fig.7 shows examples of different shapes of cylinder ingredients. The detector scans the input image by a scanning-window to obtain sub-images as input patches. Then, a cascaded classifier is designed to determine the probability of a proposal existence for each patch. The cascaded classifier, adapted to the different slice sequences generated in Section 4, consists of part variance and ensemble classifier that both reject the problematic patches.

The part variance rejects all patches whose location scores of proposals are lower than δ_d through the Deformable Part Models (DPM) method [35]. The goal of this component is to improve the performance of the cascaded classifier when facing the deformable appearance of the proposal. Accordingly, a cylinder ingredient is considered a multi-part symmetric model. The model is defined by a coarse root filter F_r and n parts filters $\{F_1, \dots, F_n\}$. The score of filters specifies a feature map for a finite number of scales in a fixed range. Let v_f denote the vector obtained by concatenating the feature vectors in the sub-window of a feature pyramid with the top-left corner in row-major order. Let F' be the vector obtained by concatenating the weight vectors in the filter F in row-major order. Therefore, the location score s_d of the model at a particular position and scale within a slice is defined as

$$s_d = r_{0,l_0}(x_0, y_0) + \sum_{m=1}^{n} d_{m,l_0-\lambda}(2(x_0, y_0) + q_m) + b_d \quad (2)$$

where $r_{0,l_0}(x_0, y_0) = F_r' \cdot v_f(x_0, y_0, l_0)$ is the response score of the root filter at the given location (x_0, y_0) and scale l_0, λ is the number of levels we need to go down in the feature pyramid, q_m is the deflection coefficient between m-th parts filter and root filter while b_d is the deflection coefficient between parts filters. Here, $d_{m,l_0-\lambda}$ is the maximum contribution of the m-th parts filter, which is computed as

$$\begin{aligned} d_{m,l}(x, y) = \\ \max_{dx,dy}(F_m' \cdot v_f(x + dx, y + dy, l)) - d_m \cdot v_d(dx, dy) \end{aligned} \quad (3)$$

where (dx, dy) is the displacement of the part relative to its anchor position, d_m is the coefficient of deformation cost, and $v_d(dx, dy) = (dx, dy, dx^2, dy^2)$ is deformation feature. Further details can be found in [35].

The task of the ensemble classifier is to filter the patches from the part variance, and reject the patches that the average posterior of each base classifier is lower than δ_e. The ensemble classifier consists of E base classifiers. Each base classifier $\mathcal{B}_j \in \{B_1, \dots, B_E\}$ performs Q pixel comparisons [36] on the patch. Then, each comparison returns 0 or 1, and these measurements are concatenated into a binary code $\alpha \in \mathbb{R}^Q$ which indexes to an array of posteriors $P_j(\beta|\alpha)$, where $\beta \in \{0, 1\}$. The posterior probability is estimated as $P_j(\beta|\alpha) = \dfrac{N_P}{N_P + N_N}$, where N_P and N_N are the numbers of positive and negative patches, respectively. The score of average posterior is defined as

$$s_e = \frac{\sum_j^E P_j(\beta|\alpha)}{E} \quad (4)$$

5.2.3 *Ingredient Determination*

After detecting all proposals, the MDP learns the policy π for those proposals in the *Positive* state. A binary Support Vector Machine (SVM) is trained to sort proposals into *Online* or *Negative* state with the training data sets, since the proposals can still be misclassified from the detector. The SVM classifier is able to handle the proposals from different sequences. The MDP transfers the proposal that is determined as a cylinder ingredient into an *Online* state. Then, the *Online* cylinder ingredient t is added into the ingredient set $\mathcal{T}$. The normalized 6-dimensional feature vector v_{P} used in the classifier contains coordinates, width, height, the score of the proposal, and the orientation label. The reward function R_{P} in a *Positive* state is defined as

$$R_{\mathrm{P}}(s, a) = y(a)(w_{\mathrm{P}}^T v_{\mathrm{P}}(s) + b_{\mathrm{P}}) \quad (5)$$

where $(w_{\mathrm{P}}, b_{\mathrm{P}})$ defines the hyperplane in SVM, $y(a) = +1$ if action $a = a_1$, while $y(a) = -1$ if $a = a_2$, as illustrated in Fig.5.

The cylinder ingredients are classified into two groups to improve the tracking accuracy in the following tracking phase. It is worth pointing out that appearance differences exist between cylinder ingredients in the slice sequence because of the diverse orientation and internal occlusion of the cylinders. For example, the appearance of a cylinder ingredient is a circle if the direction of the cross section is perpendicular to the cylinder. In addition, the horizontal section generates the cylinder ingredients with the appearance of the long strip while the slant cutting surface generates the cylinder ingredients with the elliptic appearance. Therefore, two subclasses of the cylinder ingredients are defined as the invariant class L_{in} and the variant class L_{va} based on the orientation label. For the beveled ingredient, we classify them as the invariant class because of the invariable appearance between the adjacent slices. This means that each ingredient with the variant label will typically have a bounding box whose width varies regularly through the sequence, while the ingredient with the invariant label will maintain a square with a constant aspect ratio over time.

5.3 Tracking with Templates

For an *Online* cylinder ingredient, the MDP needs to learn the policy and determine whether to keep tracking the ingredient with the ingredient template or to transfer it into an *Offline* state. Tracking template updates with the state transition of the cylinder ingredient.

5.3.1 Template Tracking

The task of template tracking is to track the *Online* cylinder ingredient in the following slices. Since our key observation is that the displacement of cylinder ingredient between adjacent slices is small and regular, the optical flow-based tracking algorithm is employed for the tracking component of STD. The optical flow tracker represents the ingredient by a bounding box, and estimates its motion between consecutive slices. To be specific, the appearance of the cylinder ingredient is simply represented by a template which is an image patch of the ingredient in a slice, as illustrated in Fig.8. Whenever a cylinder ingredient is transferred to the *Online* state, we initialize the tracking template with the bounding box of the proposal. Then, given a pixel $u = (u_x, u_y)$ on the template, we can extract the corresponding location $u_p = u+\delta_u = (u_x+\delta_x, u_y+\delta_y)$ of the ingredient in the new slice using the iterative Lucas-Kanade method with pyramids [37], [38], where $\delta_u = (\delta_x, \delta_y)$ is the optical flow at u. Further details can be found in [37], [38].

To improve the matching accuracy, the Forward-Backward (FB) error [30] is used to filter matching mistakes. The FB error of a pixel is defined as the Euclidean distance between the original pixel and the forward-backward prediction. Given a pixel u and the corresponding matching pixel u_p, we can compute the backward flow of pixel u_p to the cylinder ingredient template and obtain a new prediction $u^{'}$. Therefore, the Forward-Backward error of a pixel is denoted by $e(u) = \parallel u - u^{'} \parallel^2$. The median of the FB errors of all sampled pixels is denoted by $e_{\text{FB}} = median(\{e(u_i)\}_{i=1}^{M_p})$, where M_p denotes the total number of the pixels. The matching is filtered if the corresponding e_{FB} exceeds the threshold e_0.

In a slice sequence $\mathcal{X}$, each cylinder should be assigned a unique identity that stays constant. However, the cylinder ingredient may be inseparable from the other throughout the sequence. Besides, the ingredients of some other primitives in slices may be circles, such as the sphere and cone. Therefore, these interference terms have to be filtered along with the cylinder episode.

To avoid blending identity of close cylinders and filter out the interference, we examine the past K cylinder ingredients, and compute each area size O_k of the bounding box. The observation is that the cylinder ingredient that belongs to the same cylinder is almost invariant if it is in the invariant class L_{in} mentioned above. Accordingly, we compute the area difference of bounding boxes $E(k) =\mid O_{k+1} - O_k \mid$ between the cylinder ingredients in the adjacent slices. $e_{\text{mean}} = mean(\{E(k)\}_{k=1}^{K-1})$ is the mean area difference for ingredients in the past K slices. On the contrary, for the cylinder ingredient in the variant class L_{va}, the area difference $E(k)$ between adjacent slices is larger than the threshold value e_2. Finally, we define the reward function in an *Online* state s with feature representation $v_{\text{On}}(s) = (e_{\text{FB}}, e_{\text{mean}})$

$$R_{\text{On}}(s,a) = \begin{cases} y(a), & \text{if } L_{in} = 1, e_{\text{FB}}<e_0 \text{ and } e_{\text{mean}}<e_1; \\ & \text{if } L_{va} = 1, e_{\text{FB}}<e_0 \text{ and } e_{\text{mean}}>e_2; \\ -y(a), & \text{otherwise} \end{cases} \tag{6}$$

where e_0 , e_1 and e_2 are fixed thresholds, $y(a) = +1$ if action $a = a_3$, while $y(a) = -1$ if $a = a_4$, as illustrated in Fig. 5.

5.3.2 Template Updating

The appearance model of the template needs to be updated during the tracking process to prevent accumulating tracking errors, since the appearance of the cylinder ingredient may be variable through the slice sequence. In our case, we take advantage of the updating rule with the shift of the tracking state. As stated above, the ingredient template is initialized with the bounding box of the detected ingredient. The tracking template stays the same if it is able to track the cylinder ingredient. Once the current template fails to track the cylinder ingredient due to the appearance change, the MDP transfers the cylinder ingredient into an *Offline* state and relies on the data association to handle the appearance change and continue tracking. The tracking template is reinitialized with the associated proposal when the cylinder ingredient is transferred from *Offline* to *Online*. Meanwhile, MDP stores past K templates of the ingredients in the *Online* state when computing their area size O_k, which will also be used for data association in the *Offline* state.

5.4 Data Association via Reinforcement Learning

This section focuses on the data association problem between the *Offline* cylinder ingredients and the proposals from the cylinder ingredient detector. For an *Offline* cylinder ingredient, the MDP needs to learn the policy and determine whether to keep the ingredient *Offline*, transfer it to an *Online* state, or mark it as *Negative*. The ground truth data is first prepared based on three annotation rules. Next, reinforcement learning is applied to learn the parameters

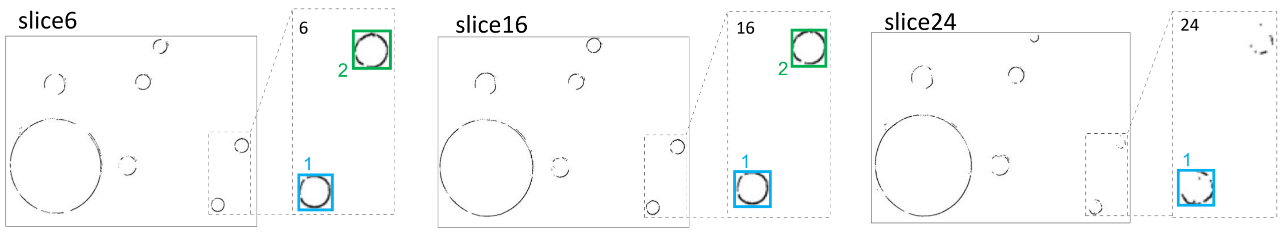


Fig. 8. The appearance of the cylinder ingredient is represented by a template in a slice. Cylinder 1 in the blue bounding box is tracked stably in slice 6,16,24, respectively. Cylinder 2 in the green bounding box is tracked in slice 6 and slice 16, but is *Offline* in slice 24 due to the missing points.

of a binary classification model, which is desired to solve the association problem.

5.4.1 Data Preparation

The strategies proposed in $MOTChallenge$ 2015 [39] are adopted to prepare slice sequences for training and testing. To generate ground truth data for data association, we use the bounding box to align with the extent of the ingredient. Specifically, the annotation rules are represented as follows. First, the bounding box of the ingredient should contain all pixels belonging to that ingredient and approximate the extent of that object accurately at the same time. Second, the label of orientation is assigned to the ingredient through the entire interactive session, which is classified into invariant and variant classes. Third, the ingredient annotation starts as early as possible and ends as late as possible to accurately represent the cycle of the cylinder.

5.4.2 Data Association

Let $t_{ij} \in \mathcal{T}$ denote an *Offline* cylinder ingredient in slice x_i, and d_{ik} be an ingredient proposal in $\mathcal{D}_i$. The goal of data association is to predict the label $y \in \{+1, -1\}$ of the pair (t_{ij}, d_{ik}) indicating that the ingredient is linked $(y = +1)$ or not linked $(y = -1)$ to the proposal. To solve the association problem, a binary classification model is designed to determine whether a cylinder ingredient is linked to the proposal or not. Specifically, we perform the binary classification using a real-valued linear function

$$f(t_{ij}, d_{ik}) = (w_{\text{Off}})^T v(t_{ij}, d_{ik}) + b_{\text{Off}} \qquad (7)$$

where $(w_{\text{Off}}, b_{\text{Off}})$ are the parameters that control the function, and $v(t_{ij}, d_{ik})$ is the feature vector that captures the similarity between the *Offline* cylinder ingredient t_{ij} and the proposal d_{ik}. The rule is given by $y = 1$ if $f(t_{ij}, d_{ik}) \geq 0$, else $y = -1$. Therefore, the reward function for data association in an *Offline* state s with feature representation $v_{\text{Off}}(s) = \{v(t_{ij}, d_{ik})\}_{k=1}^{P}$ is defined as

$$R_{\text{Off}}(s, a) = y(a)(\max_{k=1}^{P}((w_{\text{Off}})^T v(t_{ij}, d_{ik}) + b_{\text{Off}})) \qquad (8)$$

where $y(a) = +1$ if action $a = a_6$, $y(a) = -1$ if $a = a_5$, as illustrated in Fig.5, k indexes P potential proposals for association. Then, the task of policy learning in the *Offline* state turns into learning the parameters $(w_{\text{Off}}, b_{\text{Off}})$ in the decision function.

TABLE 1
Feature Representations Used for Data Association

Type	Notation	Feature Description
FB error	$v_1 \in \mathbb{R}^4$	Mean of the median forward-backward errors from the entire, left half, right half, upper half, and lower half of the templates in optical flow, respectively
NCC	$v_2 \in \mathbb{R}$	Mean of the median Normalized Correlation Coefficients (NCC) between image patches around the matched pixels obtained from optical flow
NCC	$v_3 \in \mathbb{R}$	Mean of the NCC between image patches of the proposal and the predicted bounding boxes from optical flow
Height ratio	$v_4 \in \mathbb{R}$	Mean of the ratios in bounding box height between the proposal and the predicted bounding boxes obtained from optical flow
Height ratio	$v_5 \in \mathbb{R}$	Ratio in bounding box height between the cylinder ingredient and the proposal
Aspect ratio	$v_6 \in \mathbb{R}$	Mean of ratios between height and width of predicted bounding boxes obtained from optical flow
Overlap	$v_7 \in \mathbb{R}$	Mean of the bounding box overlaps between the proposal and the predicted bounding boxes from optical flow
Score	$v_8 \in \mathbb{R}$	Normalized proposal score
Distance	$v_9 \in \mathbb{R}$	Euclidean distance between the centers of the cylinder ingredient and the proposal after motion prediction of the ingredient with a linear velocity model
Orientation label	$v_{10} \in \mathbb{R}$	Label difference between the cylinder ingredient and the proposal

5.4.3 Feature Representation

The 13-dimensional feature vector $v(t_{ij}, d_{ik})$ is used to calculate the similarity between an *Offline* cylinder ingredient t_{ij} and a proposal d_{ik}. The details of our feature representation are given in Table 1. First of all, the history of the cylinder ingredient is represented by K templates in the past K slices when the cylinder ingredient is being tracked before it transfers to the *Offline* state. Second, given the ingredient proposal d_{ik}, we compute optical flow from each template to the proposal. Then, we measure the quality of the optical flow in different aspects and use the metrics as features. In addition, the label of orientation and the aspect ratio of the bounding box are also added to the feature representation to facilitate tracking cylinders in different directions.

5.4.4 *Parameters Learning*

The binary classifier for data association is trained with reinforcement learning using slice sequences in our MDP to learn the value of the parameter pair $(w_{\text{Off}}, b_{\text{Off}})$, like the strategies proposed in [33]. The initial weights (w_0, b_0) and the empty training collection $\mathcal{P}_0 = \emptyset$ are the initialization value for training. Given the training sequences and corresponding ground truth trajectories, our goal is to train the MDP to successfully track all ingredients. Therefore, the training algorithm follows the current policy of the MDP to track the ingredients by looping over all the sequences and all the ingredients.

The binary classifier is updated only when the MDP makes a mistake in data association. In this case, the MDP takes a different action as indicated by the ground truth trajectory. Suppose the MDP is tracking the j-th cylinder c_j in sequence $\mathcal{X}$. Two different mistakes may occur due to the complex disturbing term if the MDP is in an *Offline* state on the i-th slice x_i. The first one is that the MDP associates the cylinder ingredient t_{ij} with the wrong proposal d_{ik}. Then, $v(t_{ij}, d_{ik})$ is added in the training collection $\mathcal{P}$ as a negative example. The other is that the MDP decides not to associate the cylinder ingredient to any proposal, but the cylinder ingredient is visible and correctly detected by a proposal d_{ik} according to the ground truth. Therefore, $v(t_{ij}, d_{ik})$ is added in the training collection $\mathcal{P}$ as a positive example.

The binary classifier is trained to process all testing sequences when the current collection is augmented with the pair of feature v and corresponding label $y \in \{+1, -1\}$. Specifically, given the current training collection $\mathcal{P} = \{(v_m, y_m)\}_{m=1}^{M}$, we solve the following soft-margin optimization problem by adding a slack variable ε_m to obtain a max-margin classifier for data association:

$$\begin{aligned} &\min_{w_{\text{Off}}, b_{\text{Off}}, \varepsilon_m} \frac{1}{2} \parallel w_{\text{Off}} \parallel^2 + \Phi \sum_{m=1}^{M} \varepsilon_m \\ &s.t.\ \ y_m(w_{\text{Off}}^T v_m + b_{\text{Off}}) \geq 1 - \varepsilon_m, \varepsilon_m \geq 0 \end{aligned} \tag{9}$$

where ε_m is a slack variable, Φ is a regularization parameter. Once the classifier has been updated, we obtain a new policy that is used in the next iteration of the training process. We keep iterating and updating the policy until all the ingredients are successfully tracked.

5.5 Terminal Validation

The backtracking method is applied to validate the cylinder terminal from the ingredients in the *Offline* state. We transfer the cylinder ingredient into a *Negative* state and put an *End* tag on it if the cylinder ingredient is stuck in the *Offline* state more than T slices. Accordingly, the end of the cylinder appears. To get an accurate terminal, we backtrack the slices that contain cylinder ingredients in *Offline* state and mark the first one as the end of the cylinder. Then, we store the parameters of the cylinder episode, including terminal coordinate, orientation as well as location, and recover the points of cylinder $c_j \in \mathcal{C}$ using the mapping recorded in the slice rendering base on these parameters.

Algorithm 1: Simultaneous multi-cylinder detection with MDPs

```
Input: A slice sequence $\mathcal{X} = \{x_i\}_{i=1}^{L}$, cylinder
       ingredient proposals $\mathcal{D}_i = \{d_{ik}\}_{k=1}^{N_i}$ of each
       slice $x_i$, weights $(w_{\text{Off}}, b_{\text{Off}})$ of binary classifier
       for data association
Output: Cylinder ingredient set $\mathcal{T} = \{t_{ij}\}$, detected
        cylinder set $\mathcal{C} = \{c_j\}_{j=1}^{J}$
1  Initialization: $\mathcal{T} = \emptyset$; $\mathcal{C} = \emptyset$;
2  foreach slice $x_i$ in $\mathcal{X}$ do
3      //process ingredients in Online states
4      foreach Online cylinder ingredient $t_{ij}$ in $\mathcal{T}$ do
5          Track the ingredient with template according
           to section 5.3, move the MDP of $t_{ij}$ to the next
           state;
6      end
7      //process ingredients in Offline states
8      foreach Offline cylinder ingredient $t_{ij}$ in $\mathcal{T}$ do
9          foreach proposal $d_{ik}$ not covered by any Online
           cylinder ingredient do
10             Compute
               $f(t_{ij}; d_{ik}) = (w_{\text{Off}})^T v(t_{ij}, d_{ik}) + b_{\text{Off}}$;
11         end
12     end
13     Data association with Hungarian algorithm for the
       Offline cylinder ingredients according to Section
       5.4;
14     foreach Offline cylinder ingredient $t_{ij}$ in $\mathcal{T}$ stays
       Offline in more than $T$ slices do
15         Transfer the cylinder ingredient $t_{ij}$ into
           Negative according to Section 5.5;
16         Recover cylinder $c_j$;
17         $\mathcal{C} = \mathcal{C} \cup c_j$;
18     end
19     // initialize new cylinders
20     foreach proposal $d_{ik}$ not covered by any tracked
       cylinder ingredient in $\mathcal{T}$ do
21         Initialize a MDP for a new cylinder ingredient
           $t_{ij}$ with proposal $d_{ik}$;
22         $\mathcal{T} = \mathcal{T} \cup t_{ij}$;
23     end
24 end
```

6 Simultaneous Multi-cylinder Detection

After learning the policy and reward of the MDP, the MDP is applied to detect multiple cylinders simultaneously. We create an MDP for each cylinder episode, so that the MDP determines the length of the cylinder episode guided by the policy.

More specifically, given a new input slice, the MDP follows the learned policy to process *Online* and *Offline* ingredients successively. First, the ingredients in the *Online*

TABLE 2
Model Statistics

Model	Model 1	Model 2	Model 3	Model 4
Points	35,290,033	25,765,409	110,169,866	231,981,431
N_T	109	52	116	353

state are processed to determine if they should stay *Online* or transfer to the *Offline* state. An *Online* ingredient keeps the *Online* state if it can be tracked in the following slices as described in section 5.3. Otherwise, it is transferred to the *Offline* state. Next, the *Offline* cylinder ingredients are associated with the proposals. To achieve this, the pairwise similarity is computed between the *Offline* ingredient and proposals that are not covered by the *Online* ingredients, where proposals covered by *Online* ingredients are suppressed to reduce ambiguities in data association. The similarity score is computed by the binary classifier for data association based on Section 5.4. After that, the Hungarian algorithm [40] is used to obtain the assignment between proposals and *Offline* ingredients with the similarity scores. According to the assignment, *Offline* ingredients which are linked to some ingredient proposals are transferred back to *Online* states. Otherwise, they stay *Offline*. Finally, an MDP is initialized for each ingredient proposal which is not covered by any *Online* cylinder ingredient. The ingredients in the *Online* state enjoy the priority in the tracking and association progress. Algorithm 1 describes our multiple cylinders detection algorithm using MDPs in detail.

7 Experimental Results And Performance Analysis

In this section, we first demonstrate the accuracy of STD on four large-scale and complex point clouds of process plants. Next, we show the robustness of STD against noise and occlusion. Finally, we systematically evaluate the overall efficiency of STD on real-world large-scale point clouds.

7.1 Experimental Configuration

Experimental Dataset. Four real-world scans of large process plants are tested in our experiment. These large-scale and complex point clouds contain a variety of pipes, joints and other structures, whose statistics are presented in Table 2, where N_T is the number of total true cylinders. Thereinto, Model 4 represents the largest scenario which contains 231,981,431 points and 353 cylinders in the 33 meters $\times$ 25 meters $\times$ 26 meters scene.

Parameter settings. In this paper, the interval Int and the section neighborhood δ_t for the slice extraction in Section 4.1 are respectively set to be 0.1 and 0.035 meters based on scanning precision. Slice sequences in Section 5.2 are generated with rotation parameters as follows: angle step s_{an}= 5 degrees, axis step s_{ax} = 1 cm, plane step s_p = 1 cm. The initial bounding box in proposal detection is generated with the following parameters: scale step s_s= 1.2, horizontal step s_h = 5 percent of width, vertical step s_v = 5 percent of height, minimal bounding box size s_b = 40 pixels. Besides, two thresholds δ_d and δ_e in proposals detection are set to be 65 and 50, respectively.

Implementation details. The algorithms designed in this study and all comparisons are performed on a desktop PC with Intel(R) Core(TM) I7 CPU and 8-GB RAM. Since the existing deep learning-based methods cannot directly scale up to large-scale point clouds due to their high computational and memory costs, we only compare our method with the cylinder detection methods proposed in [1], [2], [4], [17].

7.2 Accuracy Evaluation

In the experiment, we qualitatively and quantitatively evaluate the accuracy of the proposed method for detecting cylinders. Three different evaluation metrics [2], [17] are designed as follows:

- Recall ratio $r_R = N_R/N_T$. N_R is the number of true pipes recognized by the algorithm, and N_T is the total number of cylinders in the plant. An algorithm that has a good ability to recognize more true cylinders should have a higher value of recall ratio.
- Error ratio $r_E = N_F/N_T$. N_F is the number of false cylinders that output from the algorithm. An algorithm may output a huge number of cylinders that include all true ones. However, a good algorithm should have a low error ratio at the same time.
- Precision ratio $r_P = N_R/(N_R + N_F)$. $N_R + N_F$ is the number of cylinders that are detected by the algorithm. We use r_P as an auxiliary measure to evaluate the algorithmic performance.

We say that a detected cylinder is a true detection if the cylinder and its corresponding ground truth share at least 50 percent of their samples. On the contrary, the detection is considered false. Furthermore, if one identical cylinder is detected more than once, only one detection is counted as a recognized cylinder, and the others are considered false detections.

Fig. 9 demonstrates a collection of qualitative detection results on four scanned large-scale models applying different methods. The blue points represent detected cylinders. Accordingly, the different results of STD and state-of-the-art cylinder detection methods [1], [2], [4], [17] can be seen in Fig. 9, respectively. These results show that our method is successful in detecting most of the complete and partial cylinders from large-scale and complex point clouds.

To illustrate the accuracy of STD, we compare our method with the cylinder detection methods in [1], [2], [4], [17]. From Table 3, our cylinder detection algorithm outperforms the other detection approaches for all three metrics r_R, r_E, and r_P. Table 3 quantitatively illustrates that, compared to the second best, STD improves the recall ratio by 40, 34, 61, and 65 percent on four models, respectively. Besides, our evaluation procedure confers a strong advantage on error detections in terms of error ratio r_E. False detections are produced when we fit cylinders in 3D space considering the noise and complex structure. On the contrary, our method produces one false alarm in the *Online* and *Offline* state respectively according to the adjacent slices. As illustrated in Table 3, compared to the second best [17], STD decreases the error ratio by 82, 89, 82, and 84 percent on four models, respectively. Table 3 also shows that STD

TABLE 3
Performance Comparison of Cylinders Detection with Three Evaluation Metrics

method	Recall ratio r_R					Error ratio r_E					Precision ratio r_P				
	STD	[17]	[2]	[4]	[1]	STD	[17]	[2]	[4]	[1]	STD	[17]	[2]	[4]	[1]
Model 1	**0.899**	0.642	0.523	0.486	0.587	**0.045**	0.275	0.367	0.752	0.624	**0.951**	0.700	0.588	0.392	0.485
Model 2	**0.904**	0.615	0.442	0.557	0.673	**0.038**	0.346	0.461	1.192	0.615	**0.959**	0.640	0.511	0.318	0.522
Model 3	**0.914**	0.552	0.405	0.526	0.567	**0.052**	0.302	0.491	1.138	0.517	**0.946**	0.646	0.452	0.316	0.524
Model 4	**0.878**	0.532	0.430	0.337	0.532	**0.056**	0.373	0.453	1.255	0.501	**0.939**	0.588	0.487	0.212	0.515

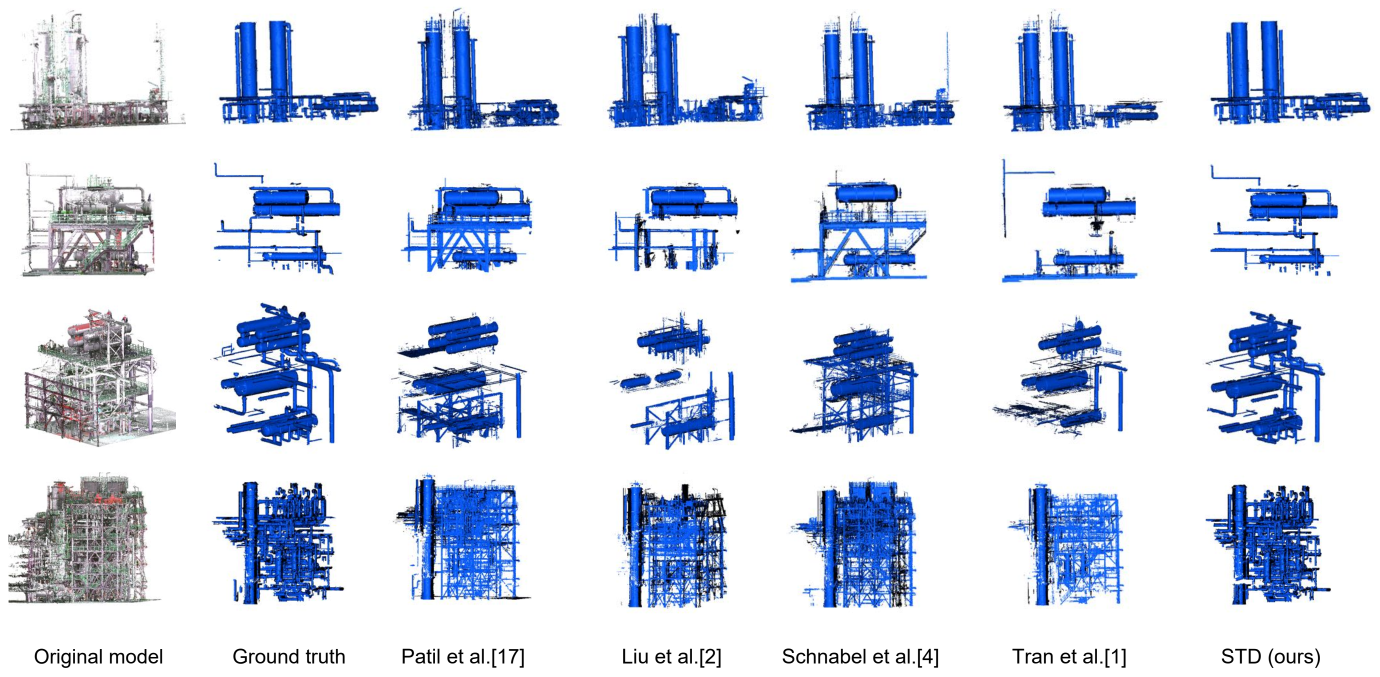


Fig. 9. A collection of qualitative results we generated on four scanned large-scale and complex point clouds of process plants by applying our approach and methods in [1], [2], [4], [17]. Ground truth is shown in the second column.

outperforms by 35, 49, 46, and 59 percent the second best [17] considerably on precision ratio r_P, respectively.

7.3 Robustness Evaluation

In a large-scale and complex process plant, the performance of cylinder detection from point clouds is limited by noise, which mainly includes clutter around cylinders, similar primitives, and object occlusion. To illustrate the robustness of our method against noise and occlusion, STD was tested on complex process plants.

Noise. The robustness of STD against different levels of noise is demonstrated in Fig.10. Gaussian noise with zero mean and standard deviations of 0-10% is first added to the point cloud. STD still achieves good results when processing the point clouds corrupted by Gaussian noise directly. The reason is that the local density of point clouds can keep the ingredient outline distinct when the slices are generated. Therefore, dimensionality reduction can decrease the impact of Gaussian noise. Fig.10 also shows a quantitative comparison of our STD with the methods in [1], [2], [4], [17].

Outliers. Fig.11 illustrates the robustness against outliers, including the clutter around cylinders and similar primitives. To make the contrast even more remarkable, we mark the samples of detected cylinders with blue points. With template matching and interference filtering in the tracking process, our method can deal with the clutter around the cylinders. For the interference primitives, the ingredient detector can successfully discriminate the ingredients at the detection stage. In addition, interference terms also can be filtered at the tracking stage based on the two different classes of feature representation. Fig.11 shows that our methods outperform the Area-based method [17] when facing the outliers.

Occlusion. As can be seen from Fig.12(a), complex occlusions exist in realistic process plant scenarios. In such a scenario, pipes may traverse the operation platform. The direct projection in [2], [12] is not applicable in this scenario. In contrast, our method is more effective in handling these occlusions due to the dimension reduction, which not only reduces the computational complexity, but also moderates the connection complexity between structures. Fig. 12(b) and Fig. 12(c) present a close-up view of the experiment results using our approach and method in [2], respectively. It can be clearly seen that our approach is successful in detecting multiple cylinders when facing the occlusions.

TABLE 4
Computing Time and Ratio between the Computing Time and Points Number

Model	Time(sec)					Ratio				
	STD	[17]	[2]	[4]	[1]	STD	[17]	[2]	[4]	[1]
Model 1	**210**	756	537	622	6174	**0.595**	2.142	1.522	1.762	17.495
Model 2	**175**	588	401	514	4563	**0.672**	2.282	1.556	1.995	17.710
Model 3	**256**	2156	1744	1880	19742	**0.232**	1.957	1.583	1.706	17.919
Model 4	**607**	4895	4121	4457	43252	**0.261**	2.110	1.776	1.921	18.644

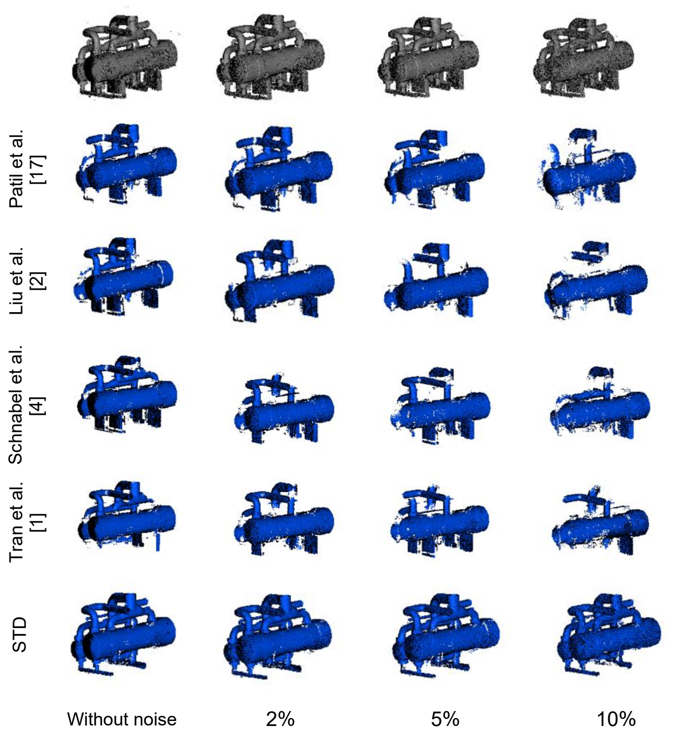


Fig. 10. Cylinders detected by the compared techniques with different levels of noise. The top row shows the original models with different levels of noise.

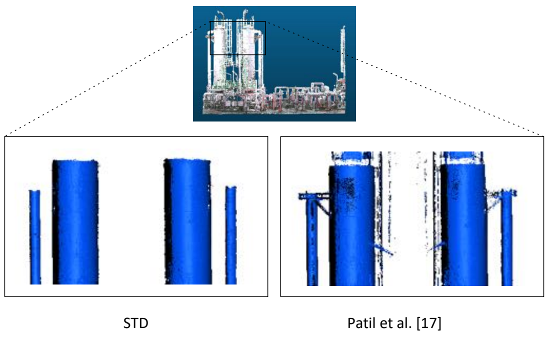


Fig. 11. A close-up view of STD (left) against outliers compared with the area-based method in [17] (right). Blue points represent samples of detected cylinders. STD is robust against outliers while area-based method is sensitive to outliers.

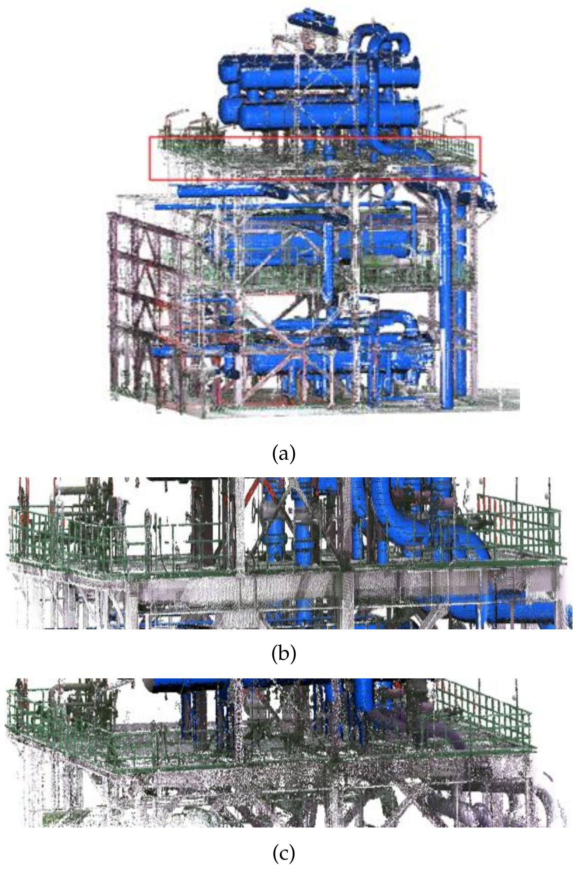


Fig. 12. Part of the petrochemical plant dataset with occlusion. (a) The detection result in occlusion scenarios with STD. (b) A close-up view of a local platform. (c) A close-up view of a local platform using the projection method in [2]. The blue points are the extracted cylinders using our method.

7.4 Efficiency Evaluation

To demonstrate the efficiency of STD, we compare our method with methods [1], [2], [4], [17] on the same four models, as shown in Fig.13. Table 4 shows the computing time of our method and other state-of-the-art methods [1], [2], [4], [17], which quantitatively demonstrates the efficiency and competitiveness of our method. As stated earlier, the cylinder detection method in 3D space has high complexity with respect to time. This is made even more severe when the number of points reaches the magnitude of tens of millions. In contrast, our method reduces the computational complexity using dimensionality reduction, since the proposed method maps the point clouds to the pixels in images. The computing time depends on the number of slices and the number of cylinders. Therefore, our method is an order of magnitude faster, which is a major contribution of this paper.

It is worth pointing out that the algorithmic complexity

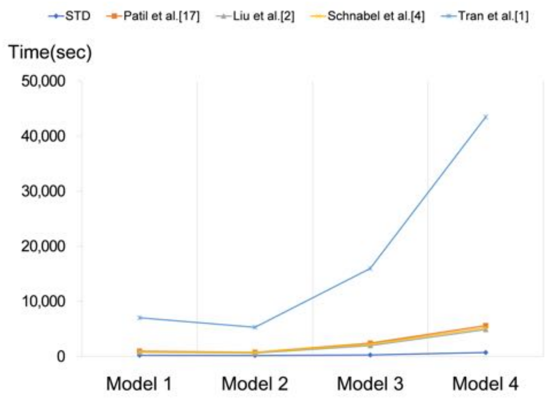


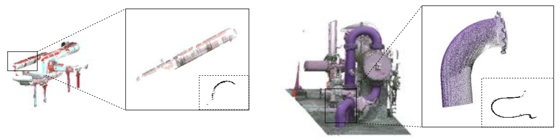
Fig. 13. Efficiency comparison with the methods in [1], [2], [4], [17].

TABLE 5
Time Comparison between Different Sampling Rate

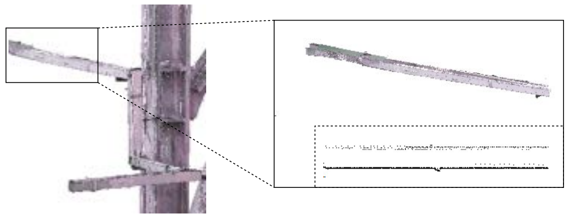
Time(sec)	μ=1	μ=2	μ= 3	μ= 5
STD	149	152	153	158
Patil et al. [17]	227	475	701	1347
Schnabel et al. [4]	332	687	1131	1804
Liu et al. [2]	202	392	694	1201
Tran et al. [1]	1632	3511	5042	9153

of STD is independent of the local sampling density of point clouds. In Table 5, we compare the time consumption of our method with the state-of-the-art methods at the different sampling rates. The original point cloud model in Fig.2 is represented as a uniform probability distribution in according with the local sampling density. From the experiments, the computing time of other methods increases linearly as the number of points increases. Therefore, in practice, down sampling of point clouds should always be conducted before adopting the methods in 3D space, such as the RANSAC-based method [1] and area-based approach in [17]. However, density sampling will bring significant influence to the detection results. For STD, the local sampling density and number of points have little effect on computing time, even when the rate μ of the local point density reaches 5. Thus, our algorithm is feasible for the models with high sampling density.

Besides, our approach is not subject to the scale of the point numbers but to the number of slices. We define $T_{ave} = 10^5 \times T_c/N_p$ to describe the relationship between the computing time and the number of points, where T_c is the computing time and N_p is the number of points. As can be seen in Table 4, the methods in [2] and [17] have an empirically linear time complexity with respect to the number of points, which does not conform to our method. The superiority of our approach is demonstrated in models with more points. Particularly, the number of points in Model 4 in Fig.9 is almost six times larger than Model 1, but the time consumption of Model 4 is just three times the Model 1.



(a) Miss detection



(b) False alarm

Fig. 14. Two typical failure cases. (a) Samples of miss detection: pipes with large gaping holes and elbow joints. (b) Samples of false alarm: objects with strip structure. Ingredient examples are in dotted boxes.

7.5 Limitation and Failure Cases

One limitation of STD is that we need to generate and label some data sets for training. It is a time-consuming phase for data preparation. It would, thus, be beneficial to extend these sets to other research, such as other geometric primitives detection and semantic segmentation of large-scale point clouds.

The miss detection cases contain two major classes as shown in Fig.14(a). The first class is that the pipes contain large gaping holes due to the internal occlusion. Accordingly, the generated cylinder ingredients contain a severe defect. The second class includes the cylinders existing in the elbow joints. Both two groups of cylinders make it hard for the cylinder ingredient detector to detect proposals due to the irregular appearance.

The false alarm cases contain the strip structures as shown in Fig.14(b). They could be detected in the proposals detection phase when the slices are extracted horizontally due to the similar appearance to the cylinder ingredients. This will be more alleviative when the disturbance terms are filtered by taking advantage of the appearance changes between slices.

8 Conclusion

In this paper, we present a novel multiple cylinder detection framework for large-scale and complex process plants. By using MOT, the 3D cylinders detection problem is modeled as a cylinder ingredients tracking task. Extensive experimental results show that our method outperforms state-of-the-art approaches on accuracy and efficiency. Experiments clearly demonstrate that our proposed algorithm is also robust against the noise outliers and occlusions. It would be interesting to apply our idea to detect other geometric primitives. Besides, some other applications can also be investigated following the framework, including skeletons extraction, semantic segmentation, and surface reconstruction.

ACKNOWLEDGMENTS

The authors sincerely acknowledge the anonymous reviewers for their insights and comments to further improve the quality of the manuscript. They also would like to thank the participants in the study for their valuable time, and to the members of the Visual Computing Group at NJUST for their feedback. This work was supported by National Key R&D Program of China (2018YFB1004904), Fundamental Research Funds for the Central Universities (30918012203) and the Pre-research Project of The 13th Five Year Plan (61409230104 and 315100104).

REFERENCES

[1] T. T. Tran, V. T. Cao, and D. Laurendeau, "Extraction of cylinders and estimation of their parameters from point clouds," *Computers and Graphics*, vol. 46, pp. 345–357, 2015.

[2] 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 Trans. Vis. Comput. Graph.*, vol. 19, no. 10, pp. 1700–1707, 2013.

[3] 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*, vol. 91, pp. 44–52, 2018.

[4] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.

[5] R. C. Bolles and M. A. Fischler, "A ransac-based approach to model fitting and its application to finding cylinders in range data," in *International Joint Conference on Artificial Intelligence*, 1981.

[6] T. Rabbani and F. V. D. Heuvel, "Efficient hough transform for automatic detection of cylinders in point clouds," *Proc. ISPRS Workshop Laser Scanning*, pp. 60–65, 2005.

[7] M. F. Ahmed, C. T. Haas, and R. Haas, "Automatic detection of cylindrical objects in built facilities," *Journal of Computing in Civil Engineering*, vol. 28, no. 3, pp. 1–11, 2014.

[8] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. Guibas, "Supervised fitting of geometric primitives to 3d point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[9] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem, "3d-prnn: Generating shape primitives with recurrent neural networks," in *IEEE International Conference on Computer Vision*, 2017.

[10] S. Tulsiani, H. Su, L. J. Guibas, A. A. Efros, and J. Malik, "Learning shape abstractions by assembling volumetric primitives," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[11] G. Sharma, R. Goyal, D. Liu, E. Kalogerakis, and S. Maji, "Csgnet: Neural shape parser for constructive solid geometry," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[12] A. M. Arajo and M. M. Oliveira, "Connectivity-based cylinder detection in unorganized point clouds," *Pattern Recognition*, vol. 100, pp. 107–161, 2020.

[13] T. Birdal, B. Busam, N. Navab, S. Ilic, and P. Sturm, "Generic primitive detection in point clouds using novel minimal quadric fits," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 6, pp. 1333–1347, 2020.

[14] R. Dahyot, "Statistical hough transform," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 8, pp. 1502–1509, 2009.

[15] A. Abuzaina, M. S. Nixon, and J. N. Carter, "Sphere detection in kinect point clouds via the 3d hough transform," in *15th International Conference on Computer Analysis of Images and Patterns*, 2013.

[16] M. Camurri, R. Vezzani, and R. Cucchiara, "3d hough transform for sphere recognition on point clouds," *Machine Vision and Applications*, vol. 25, no. 7, pp. 1877–1891, 2014.

[17] A. K. Patil, P. Holi, K. L. Sang, and Y. H. Chai, "An adaptive approach for the reconstruction and modeling of as-built 3d pipelines from point clouds," *Automation in Construction*, vol. 75, pp. 65–78, 2017.

[18] T. Chaperon and F. Goulette, "Extracting cylinders in full 3d data using a random sampling method and the gaussian image," in *Vision Modeling and Visualization Conference*, 2001.

[19] 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*, vol. 103, pp. 150 – 167, 2019.

[20] A. Bey, R. Chaine, R. Marc, G. Thibault, and S. Akkouche, "Reconstruction of consistent3d cadmodels frompointcloud data usingapriori cadmodels," *ISPRS - International Archives of the Photogrammetry*, vol. XXXVIII-5/W12, pp. 289–294, 2012.

[21] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1806–1819, 2011.

[22] H. B. Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Multi-commodity network flow for tracking multiple people," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1614–1627, 2014.

[23] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybrid-boosted multi-target tracker for crowded scene," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[24] Y. Park, V. Lepetit, and W. Woo, "Extended keyframe detection with stable tracking for multiple 3d object tracking," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 11, pp. 1728–1735, 2011.

[25] A. A. Valsangkar, J. M. Monteiro, V. Narayanan, I. Hotz, and V. Natarajan, "An exploratory framework for cyclone identification and tracking," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 3, pp. 1460–1473, 2019.

[26] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[27] S. Oh, S. Russell, and S. Sastry, "Markov chain monte carlo data association for multi-target tracking," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 481–497, 2009.

[28] Z. Khan, T. Balch, and F. Dellaert, "Mcmc-based particle filtering for tracking a variable number of interacting targets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1805–1819, 2005.

[29] S. H. Bae and K. J. Yoon, "Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[30] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, 2012.

[31] S. Kim, S. Kwak, J. Feyereisl, and B. Han, "Online multi-target tracking by large margin structured learning," in *Asian Conference on Computer Vision*, 2012.

[32] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *AAAI Conference on Artificial Intelligence*, 2017.

[33] Y. Xiang, A. Alahi, and S. Savarese, "Learning to track: Online multi-object tracking by decision making," in *IEEE International Conference on Computer Vision*, 2015.

[34] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, pp. 679–684, 1957.

[35] Felzenszwalb, F. Pedro, Girshick, B. Ross, McAllester, David, Ramanan, and Deva, "Object detection with discriminatively trained part-based models." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010.

[36] V. Lepetit and P. Fua, "Keypoint recognition using randomized trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1465–1479, 2006.

[37] J. Y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm," Intel Corporation, pp. 1–9, 2000.

[38] T. Lee and T. Hollerer, "Multithreaded hybrid feature tracking for markerless augmented reality," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 3, pp. 355–368, 2009.

[39] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking," *arXiv:1504.01942 [cs]*, 2015.

[40] Munkres and James, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

Zhuheng Lu is currently working toward the Ph.D. degree at the School of Automation, Nanjing University of Science and Technology, China. He received the M.S. degree from the School of Automation, Nanjing University of Science and Technology, in 2014. His research interests include computer graphics, 3D model recognition and machine learning.

Weiwei Mao is currently working toward the Ph.D. degree at the School of Automation, Nanjing University of Science and Technology, China. He received the M.S. degree from the School of Automation, Nanjing University of Science and Technology, in 2011. His research interests include computer vision, image retrieval and graph learning.

Yuewei Dai is currently a Professor at the School of Electronic and Information Engineering, Nanjing University of Information Science and Technology, China. He received the M.S. and Ph.D. degrees from the Nanjing University of Science and Technology, in 1987 and 2002, respectively, all in automation. His research interests include information security, signal, and image processing.

Weiqing Li is currently an associate professor at the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. He received the B.S. and Ph.D. degrees from the School of Computer Sciences and Engineering, Nanjing University of Science and Technology in 1997 and 2007, respectively. His current interests include computer graphics and virtual reality.

Zhiyong Su is currently an associate professor at the School of Automation, Nanjing University of Science and Technology, China. He received the B.S. and M.S. degrees from the School of Computer Science and Technology, Nanjing University of Science and Technology in 2004 and 2006, respectively, and received the Ph.D. from the Institute of Computing Technology, Chinese Academy of Sciences in 2009. His current interests include computer graphics, computer vision, augmented reality, and machine learning.