Connectivity-based Cylinder Detection in Unorganized Point Clouds

Abner M. C. Araújo, Manuel M. Oliveira*

Instituto de Informática - UFRGS

Abstract

Cylinder detection is an important step in reverse engineering of industrial sites, as such environments often contain a large number of cylindrical pipes and tanks. However, existing techniques for cylinder detection require the specification of several parameters which are difficult to adjust because their values depend on the noise level of the input point cloud. Also, these solutions often expect the cylinders to be either parallel or perpendicular to the ground. We present a cylinder-detection technique that is robust to noise, contains parameters which require little to no fine-tuning, and can handle cylinders with arbitrary orientations. Our approach is based on a robust linear-time circle-detection algorithm that naturally discards outliers, allowing our technique to handle datasets with various density and noise levels while using a set of default parameter values. It works by projecting the point cloud onto a set of directions over the unit hemisphere and detecting circular projections formed by samples defining connected components in 3D. The extracted cylindrical surfaces are obtained by fitting a cylinder to each connected component. We compared our technique against the state-of-the-art methods on both synthetic and real datasets containing various densities and noise levels, and show that it outperforms existing techniques in terms of accuracy and robustness to noise, while still maintaining a competitive running time.

*Corresponding author. Tel.: +55 51 3308 6821; Fax: +55 51 3308 7308.
Email addresses: `amcaraujo@inf.ufrgs.br` (Abner M. C. Araújo), `oliveira@inf.ufrgs.br` (Manuel M. Oliveira)
URL: `http://inf.ufrgs.br/ amcaraujo` (Abner M. C. Araújo), `http://inf.ufrgs.br/ oliveira` (Manuel M. Oliveira)

Key words: Cylinder detection, Unorganized point clouds, Reverse engineering, Industrial sites

1. Introduction

CAD models of industrial sites are extremely important assets, as they provide documentation and simplify inspection, planning, modification, as well as a variety of physical and logistics simulations of the corresponding installations. Despite these clear advantages, many industrial sites do not have CAD models, or have trouble keeping them up-to-date. This is often due to the amount of effort required to create and maintain CAD models updated. Hopefully, the recent popularization of 3D scanning devices is promoting the development of reverse engineering, allowing the creation of 3D representations of real environments from point clouds. This is a key step towards obtaining CAD models for existing installations.

In industrial sites, cylinders are used as pipes, ducts, and tanks, which are key elements of these environments. Thus, the ability to detect cylinders is essential to reverse engineering of these sites [1, 2, 3]. Besides Reverse Engineering, cylinder detection is also a key step in many other applications, such as segmentation [4], hand and human pose estimation [5, 6, 7], robotic manipulation [8] and urban scene reconstruction [9]. However, detecting cylinders in unorganized point clouds is a challenging task. Cylinders may appear with various radii, lengths, textures, and materials. Moreover, unorganized point clouds introduce additional challenges such as noise, non-uniform sampling density, incomplete models due to occlusions, and lack of semantic relationship among samples.

Previous techniques for cylinder detection are mostly based on Hough transform or RANSAC. They often contain several parameters which depend on the noise level of the point cloud, thus being cumbersome to adjust, and requiring in-depth knowledge of both the environment and the device used to sample it. This problem is accentuated if plane detection is required as an intermediate step for the actual cylinder detection, as this increases the number of parameters involved. To simplify the detection process, many techniques make assumptions about expected cylinder radii and orientations [10, 11, 12], which significantly restricts their applicability.

We present a fast cylinder-detection technique that is robust to noise, uses parameters which require little to no fine-tuning, and can handle cylinders with arbitrary orientations. It is based on a robust linear-time ($O(N)$) circle-detection algorithm that naturally discards outliers, allowing our technique to handle datasets with various density and noise levels while using a set of default parameter values. It works by projecting the point cloud onto a set of uniformly-distributed directions defined over the unit hemisphere. Only samples whose normals are approximately perpendicular to a given direction are projected along such direction. It then refines these directions and detects circular projections formed by samples defining connected components in 3D. The extracted cylindrical surfaces are obtained by fitting a cylinder to each connected component that passes a validity test.

We demonstrate the effectiveness of our approach by comparing its performance against the state-of-the-art techniques for cylinder detection on five datasets, three of which were acquired from real industrial sites. In these experiments, our method achieved the best overall accuracy using the *same set of (default) parameter values for all evaluated datasets.* This is in contrast to the other techniques, for which their parameter values were individually adjusted for each combination of technique and dataset to achieve their best results in each case. This demonstrates the robustness of our approach, which does not require fine tuning to perform well on arbitrary point clouds. Figure 1 illustrates the use of our technique applied to a point cloud of a petrochemical plant containing cylinders with various orientations, lengths, diameters, and positions.

The **contributions** of this paper include:

- A fast technique for cylinder detection in unorganized point clouds that is robust to noise and can handle cylinders with arbitrary orientations

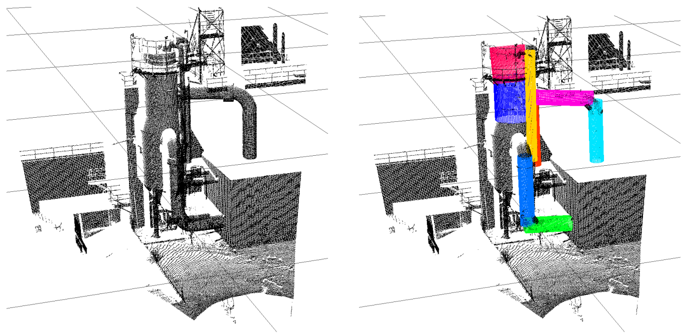

Figure 1: Example of automatic cylinder detection using our technique. (left) Point cloud of an actual petrochemical plant. (right) Detected cylinders shown as highlighted polygonal meshes.

(Section 3). Its parameter values require little to no fine-tuning to work well with general point clouds;

- A deterministic circle-recognition technique capable of filtering noisy samples (Section 3.4 and 3.5). It is faster than traditional alternatives such as Hough transform and RANSAC.

2. Related Work

Existing cylinder detection techniques can be classified in three broad categories: (i) Hough transform, (ii) RANSAC, and (iii) Region growing. The *Hough transform* (HT) [13] is a popular technique to detect patterns in images and point clouds. It consists of mapping the input data to some feature space accumulator, through a voting process. Typically, each input element votes for all possible patterns that may contain it. Detection then corresponds to identifying peaks of votes in the accumulator, from whose parameter values the detected patterns are recovered.

Cylinders can be minimally characterized by five parameters: axis (θ,ϕ), radius (r), and center (C_x, C_y), this last one corresponding to the projection of the axis on a plane perpendicular to it. Such parameters spawn a 5D feature space which is both memory- and computationally-prohibitive. As a result, direct application of the Hough transform for cylinder detection is not practical.

Rabbani et al. proposed a two-step Hough transform to mitigate this restriction [14]. First, they map the sample normals to a Gauss map. Since the normals of a cylinder form a great circle on the Gauss map, the authors proceed by detecting the planes containing these circles, followed by the projection of the associated samples onto the corresponding detected planes. For each plane, its normal is used as an estimate of a cylinder axis, while the projected circle is used to estimate the cylinder's radius and center. The detections of both planes and circles use Hough transforms. A downside of this technique is that errors during the plane detection step are propagated to the circle detection. Moreover, one needs to adjust parameters for two distinct Hough transforms to achieve good results, which is cumbersome.

Ahmed et al. re-sample the point cloud by slicing it using pre-determined intervals along the X, Y and Z directions, and on each slice they use a Hough transform to detect circles using the projections of the samples falling inside each interval [12]. Circles with similar centers along the same slicing direction are merged to obtain a cylinder. The advantage of this technique is that it removes the need of a plane detection technique on the Gauss map. However, it is restricted to cylinders aligned with the X, Y, or Z directions.

Patil et. al [15] proposed an improvement over the technique described in [14]. Instead of creating a spherical accumulator with a fixed number of cells in the Gauss map used to detect planes, the number of cells is adjusted according to the density of the samples in the map. Thus, regions with higher densities in the Gauss map cover more cells in the spherical accumulator.

Figueiredo et. al [16] proposed a framework to detect cylinders placed on top of flat surfaces. Given an RGBD image representing a flat surface with some objects on top, the planar surface depth values are used to segment the objects. An image containing the RGB pixel values of each segmented object is then presented to a CNN classifier trained on a set of images to determine if it is cylinder, in which case a Hough transform is used to obtain the cylinder parameters (axis, radius, and center). This technique is not applicable to unorganized point clouds.

Another category of cylinder detection techniques is based on the *Random Sample Consensus* (RANSAC). RANSAC is an iterative stochastic technique that works in two steps: first, it collects a minimal sample set necessary to estimate a model (*e.g.*, a cylinder), and then evaluates the fitness of this model, measuring the number of samples that satisfy an inlier criterion (*e.g.*, minimum distance to the cylindrical surface). It then chooses the model with best fit after a predefined number iterations.

Bolles et al. used RANSAC to detect cylinders with known radius and orientation in range data [10]. For each scanline, the technique tries to detect an ellipse and its center. A second RANSAC is used to detect lines in 3D passing through the centers of the detected ellipses, thus obtaining a cylindrical volume.

Chaperon and Goulette [17] used a two-step RANSAC to detect cylinders in unorganized point clouds. First, they detect planes on a Gauss map and then detect circles on the projection of the point cloud onto the planes found in the first step. This is conceptually similar to and suffers from the same limitations as Rabbani et al.'s [14], but was proposed earlier.

Schnabel et al. proposed a RANSAC technique to detect cylinders, among other geometric primitives [18]. To estimate a cylinder, given two samples, the vector resulting from the cross product of these samples' normals is used as the cylinder axis estimate. They then project these samples and their normals onto the plane perpendicular to the cylinder axis. The projected samples are used to estimate the cylinder's radius, whose center is obtained by extending the projected normals positioned at the projected samples. Albeit presenting good results, the technique suffers from limitations inherent to RANSAC. Thus, it may require many iterations to converge, is non-deterministic, and it may be difficult to detect underrepresented shapes. This later problem is accentuated in point clouds with non-uniform distributions, which is the case of most point clouds obtained from real environments.

Liu et al. [11] proposed a RANSAC technique to detect cylinders in pipeline plants which is similar to the work of Chaperon and Goulette [17]. It, however, assumes that every cylinder is either orthogonal or parallel to the ground.

Jin et. al [19] proposed a RANSAC-based technique to detect cylinders. The authors fit spheres to different regions of the point cloud, and a RANSAC technique is applied to determine straight lines passing through the centers of the spheres, with the assumption that this will result in the axes of existing cylinders. This technique does not work well in the presence of occlusions.

Qiu et. al [1] proposed a technique focused on the detection of pipes. It is based on the observation that pipes often present some locally similar properties (orientations, radii, etc.). First, the point cloud is partitioned into cells using an octree, and for each cell a random set of pairs of samples is chosen to estimate a space of possible cylinder orientation candidates. This is done by calculating the cross product between the normals of each sample pair. The most frequent orientations are determined using a clustering technique, and for each such cluster another random set of sample pairs is chosen to span the space of circle center candidates. Although faster than RANSAC-based techniques, the assumption of local similarity of the cylinders may not always hold, especially in regions cluttered with other objects.

Region growing techniques try to grow surfaces from some seed samples. Tran et al. [20] use curvature information to select potential seed sample candidates. For each candidate, the technique selects a neighborhood around it and performs an iterative fitting step. This consists of using principal component analysis (PCA) to detect the cylinder axis, and a circle fitting procedure to detect the cylinder's center and radius on the projection of the samples. Samples with good fitting to the estimated parameters are used as seed samples in the next iteration. This process stops after a pre-determined number of iterations. This technique relies on good curvature estimation, which is not trivial to obtain. Moreover, since the iterative fitting procedure is performed for each sample found in high-curvature regions, it is computationally expensive.

Nurunnabi et. al [21] presented a technique based on Robust PCA (RPCA) and robust regression that is applicable to scenes containing a single cylinder.

RPCA is used to determine the cylinder axis and center from a set of samples. The samples are then projected onto the plane determined by the cylinder axis and a technique based on robust regression is used to detect the resulting circle, which is used to estimate the cylinder radius. It is not clear how this technique could be extended to detect multiple cylinders in a scene.

Unlike previous methods, our technique does not rely on detecting planes before the actual cylinder detection, does not make any assumptions about cylinder radius or orientation, and can handle multiple cylinders in the same scene.

3. Connectivity-based Cylinder Detection

In order to detect cylinders with arbitrary orientations, our technique projects the point cloud onto a set of uniformly-distributed directions on a unit hemisphere (Section 3.1). Each direction defines a tangent plane onto which we orthographically project the samples whose normals are approximately perpendicular to the plane normal (Figure 2). We then refine the orientations of these projection planes and re-project the samples onto them (Section 3.3). For this, for each group g_i of projected samples that form a connected component in 3D, we compute a new plane orientation applying PCA to the normals of these samples in 3D. Such samples are then re-projected onto the new plane. Then, a novel circle-recognition technique is applied to elements of g_i to detect circular projections (Section 3.4). Delimited cylindrical surfaces are finally obtained by merging related connected components in 3D and fitting cylinders to the merged components (Sections 3.6 to 3.8). Samples belonging to detected cylinders are removed from the point cloud, and the remaining connected components are evaluated. This process is illustrated in Figure 3 and on Algorithm 1. Its details are presented in Sections 3.1 to 3.8.

3.1. Projection directions

Given a circular cylinder, the projection of its samples onto a plane perpendicular to the cylinder's axis defines a circle. Thus, finding circular patterns

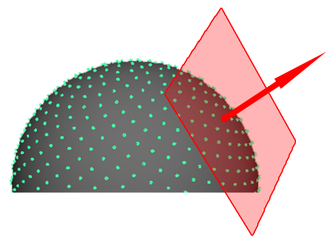

Figure 2: Uniform sampling of a hemisphere defining the initial projection orientations.

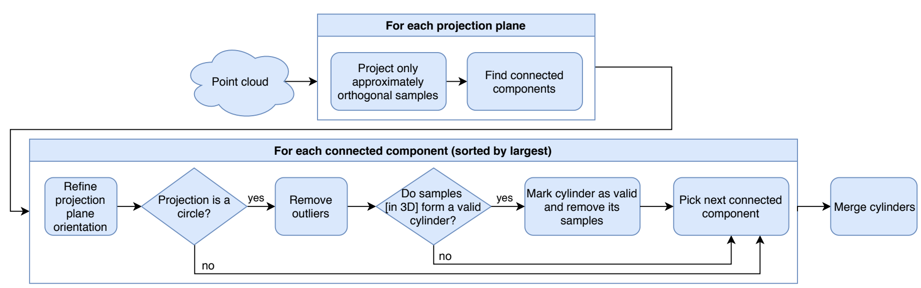

Figure 3: Our cylinder detection pipeline. Given an input point cloud, it is projected along a set of directions on the unit hemisphere. These directions are further refined. Projected circles are detected and outliers removed. Cylindrical surfaces are then obtained by fitting cylinders to connected components in 3D corresponding to detected circles. The samples of detected cylinders are removed from the point cloud, and related components are merged into single cylinders.

Algorithm 1 Cylinder Detection

```
Require: PC {point cloud with estimated normals and neighborhood information} N_s {number of sampling directions}
procedure DetectCylinders(PC, N_s)
    cylinders ← ∅
    components ← ∅
    directions ← Fibonacci(N_s)    ▷ sampling directions: Fibonacci mapping
    for all D ∈ directions do
        P' ← Project(PC, D)
        c ← FindConnectedComponents(P')
        components.insert(c)
    for all c ∈ components do
        P' ← Reproject(c)
        if ContainsCircle(P') then
            Cylinder ← FitCylinder(c)
            if IsValid(Cylinder) then
                cylinders.insert(Cylinder)
                RemoveSamples(c)
                components ← FindNewConnectedComponents(components)
    return cylinders
```

resulting from projected samples can be used to greatly simplify the detection of cylinders in point clouds. Unfortunately, plane detection techniques applied on the Gauss map and used for cylinder detection [11, 17, 14] are error prone and computationally expensive. Thus, in order to select the projection directions, we perform an initial uniform sampling of the unit hemisphere, which is further refined to adjust them to the point cloud content. Any uniform sampling technique can be used, but we opt for the spherical Fibonacci mapping [22] (due to its simplicity), with 100 sampling directions (Figure 2). Algorithm 2 implements the Fibonacci mapping.

3.2. Detecting connected components

Given the projection directions defined by the Fibonacci mapping, we project the point cloud along each direction $d_{(\theta,\phi)}$, considering only the samples whose normals are approximately perpendicular to $d_{(\theta,\phi)}$ (*i.e.*, $d_{(\theta,\phi)} \pm \tau$). For all results shown in the paper, we used an angular tolerance $\tau = 10°$.

Naively detecting circles on each projection can be, not only computationally

Algorithm 2 Hemispherical Fibonacci Mapping

Require: N_s {number of sampling directions}

procedure FIBONACCI(N_s)
    $directions \leftarrow \emptyset$
    $offset \leftarrow 2/N_s$
    $increment \leftarrow \pi(3 - \sqrt{5})$
    for $i \leftarrow 1$ to $2N_s$ **do**
        $y \leftarrow i \times offset + offset/2 - 1$
        $r \leftarrow \sqrt{1 - y^2}$
        $\alpha \leftarrow Modulo(i + 1, N_s) \times increment$
        $x \leftarrow rcos(\alpha)$
        $z \leftarrow rsin(\alpha)$
        if $z > 0$ **then**
            $directions.insert([x, y, z])$
    return $directions$

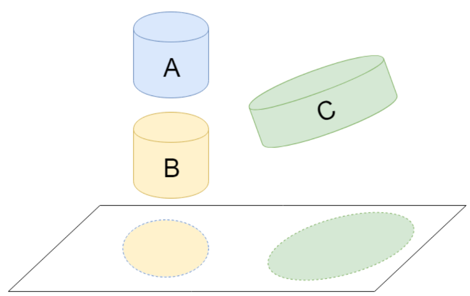

Figure 4: The projections of cylinders A and B overlap. The projection of cylinder C produces an ellipse.

expensive, but also error prone, for two reasons: (i) real scenes may contain cylinders whose projections may exactly overlap, causing two (or more) cylinders to be detected as a single one; and (ii) none of the projection planes may be perpendicular to a given cylinder axis, resulting in projected ellipses, which may not be detectable by standard approaches. Both situations are illustrated in Figure 4.

To avoid having two (or more) cylinders detected as a single one, we split each projection into groups of samples (g_i's), where each group forms a connected component in 3D. To obtain the connected components, we compute a neighborhood graph G for the point cloud performing a k-nearest neighbors search in 3D from each sample (we use $k = 50$). Then, for each projected sample, we perform a breadth-first search (BFS) on G considering *only* the set of projected samples. This search will naturally return a connected component. We perform searches until all projected samples have been visited.

3.3. Refine projection plane orientations and samples

In order to avoid elliptical projections, the projection directions need to be refined for some cylinders. These will correspond to the new cylinder axes. Thus, let c_i be a connected component in 3D belonging to a cylinder $\mathcal{C}_j$ in the point cloud, and corresponding to the projected samples in g_i. Since c_i's samples should have normals perpendicular to $\mathcal{C}_j$'s axis, we estimate the cylinder axis by applying principal component analysis (PCA) to the set of normals of c_i's samples. The direction with least variance corresponds to $\mathcal{C}_j$'s axis. We can then project c_i's samples onto the plane perpendicular to newly estimated $\mathcal{C}_j$'s axis using the procedure described in Section 3.2. Both cylinder axis refinement and sample reprojection are shown in Algorithm 3.

Algorithm 3 Refine cylinder axis and samples

Require: c {connected component} G {neighborhood graph}

procedure READJUSTCOMPONENT(c, G)
    $pca \leftarrow PCA(c.normals)$
    $axis \leftarrow pca[1]$ ▷ eigenvector with smallest eigenvalue: $|\lambda_1| \leq |\lambda_2| \leq |\lambda_3|$
    $q \leftarrow \emptyset$ ▷ initialize sample queue
    for all $s \in c.samples$ **do**
        $q.enqueue(s)$
    while $q \neq \emptyset$ **do**
        $front \leftarrow q.dequeue()$
        for all $n \in G.neighbors(front)$ **do**
            if $n \notin c \wedge acos(dot(n.normal, axis)) > \alpha$ **then**
                $c.insert(n)$
    return $Project(c, axis)$

3.4. Circle recognition

Once each connected component has been refined (Sections 3.2 and 3.3), our technique checks if its projection fits a circle (see pipeline in Figure 3). For this, we introduce a fast and robust circle recognition technique by exploring

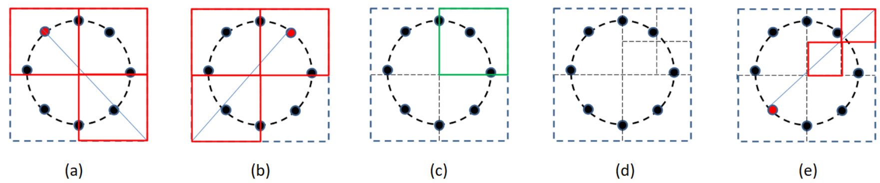

Figure 5: The subdivision strategy used to find a circle. At first, rays are traced from each sample position (small circles) along its reversed normal direction (a) (b), then the quadtree cell most intersected by these rays is chosen (c), and the samples which intersected it are evaluated using a histogram (Figure 6). If the histogram is uniform, then cell is subdvided (d). This process is performed recursively (e), until the quadtree reaches a pre-defined depth.

the fact that extended normals from each sample of a circle should intersect at the circle's center. One can then classify a set of samples as being on a circle or not depending on the existence of such *point of intersection*, which can be quickly checked using a subdivision strategy. It works as following: consider a quadtree, initialized with the bounding box of the set of projected samples, and containing four child nodes (Figure 5 (a)). For each projected sample in g_i, we check if the (reverse) ray formed by its position and (reverse) projected normal intersects the bounding box of each quadtree's child, as shown in Figures 5 (a) and (b), in which case the sample is added to the child's list of intersections. Algorithm 5 summarizes this procedure.

Algorithm 4 Check 2D Ray-AABB Intersection

Require: R {Ray} BB {Bounding Box}

procedure INTERSECT(R, BB)
- $tx_1 \leftarrow (BB.x_{left} - R.O_x)/R.N_x$ ▷ $R.O = Origin$
- $tx_2 \leftarrow (BB.x_{right} - R.O_x)/R.N_x$ ▷ $R.N = Direction(normalized)$
- $t_{min} = min(tx_1, tx_2)$
- $t_{max} = max(tx_1, tx_2)$
- $ty_1 \leftarrow (BB.y_{up} - R.O_y)/R.N_y$
- $ty_2 \leftarrow (BB.y_{down} - R.O_y)/R.N_y$
- $t_{min} = max(t_{min}, min(ty_1, ty_2))$
- $t_{max} = min(t_{max}, max(ty_1, ty_2))$
- **return** $t_{max} \geq t_{min}$

After checking intersections for all samples in g_i, the quadtree child node with most intersections (Figure 5 (c)) is chosen and checked to determine if

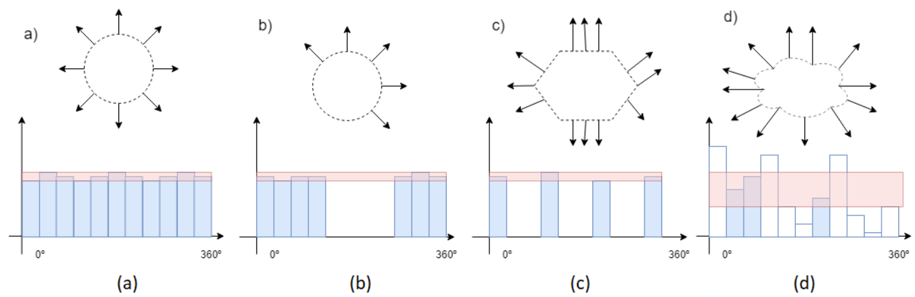

Figure 6: Histogram for different shapes after mapping the range of the *arctan2* function from $[-\pi, \pi)$ to $[0°, 360°)$. Circular shapes have their bins more uniformly distributed (a) and (b), while the bins of other shapes are either too sparse (c), or non-uniformly distributed (d). Each light red rectangle is the range of bin heights that would cause the histogram to be considered uniform. Such a range is computed using only the nonempty bin heights: it is centered on the mean, covering two standard deviations above and below the mean.

the projected samples whose (reverse) rays intersect it form a circular shape. This is done by first computing a histogram of the angles measured between the projected normal directions and the horizontal axis, according to Equation 1. If a shape is circular, this histogram must be uniformly distributed (Figure 6 (a)). However, this would only happen in case the shape is a full circle. Since point clouds are susceptible to occlusion, it is desirable to also consider arcs of varying lengths. Thus, we check if a percentage of the angular bins are uniformly distributed (*i.e.*, if the number of elements in each nonempty bin is approximately equal to the ratio between the number of projected samples and the number of nonempty bins). This is illustrated in Figure 6 (b). Figures 6 (c) and (d) show examples of histograms associated with non-circular projections.

$$\theta_N = arctan2(N_y, N_x) \tag{1}$$

A bin element is considered part of a uniform bin distribution if its value falls in the interval built around the mean (*i.e.*, number of projected samples in g_i divided by the number of bins), μ_{bins}, using three standard deviations of the bin values, σ_{bins}: $[\mu_{bins} - 2\sigma_{bins}, \mu_{bins} + 2\sigma_{bins}]$, since for a normal distribution

2 σ represents 95% of the area under the curve. For the examples shown in the paper, we subdivided the histogram into 72 bins, and considered the minimum acceptable number of uniformly-distributed bins to be 18 (one quarter of the total number of bins, or equivalent to 90 degrees).

If the quadtree child node with most intersections does not meet the above condition, the connected component is immediately rejected as a circle candidate. Otherwise, we subdivide the child node and call the same procedure recursively up to five times, considering only the set of samples in g_i whose associated reverse rays intersect the child node. This process is illustrated in Figure 5 (c) and (d).

Algorithm 5 Find a circle in a set of samples

Require: Q {Quadtree} h Minimum quadtree height

procedure FINDCIRCLE(Q, h)
  if $Q.height > h$ **then**
    $FilterNoise(Q_{samples})$
    return $Q_{samples}$
  if $IsNotCircular(Q)$ **then**
    return $NULL$
  for all $p \in Q_{samples}$ **do**
    for all $q \in Q_{children}$ **do**
      if $Intersect(p, q)$ **then**
        $q_{samples}.insert(p)$
  ▷ Q's child node with maximum number of intersections
  $q_{withMaxInters} \leftarrow MaxIntersection(Q_{children})$
  return $FindCircle(q_{withMaxInters}, h)$

The main advantage of our circle-detection algorithm over classical approaches, such as Hough transform and RANSAC, is its lower computational cost. Being significantly faster than previous techniques, our solution enables testing the projection of the point cloud along a larger number of directions in the same amount of time, ultimately improving the accuracy of cylinder detection.

Given N samples, the computational cost of our circle-detection technique is $O(D \times N)$, where D is the maximum depth of the quadtree. Since D is small ($D = 5$ for all examples shown in the paper), the cost is $O(N)$. The Hough transform for circle detection has cost $O(bins_x \times bins_y \times bins_{radius} \times N)$, where

$bins_x$, $bins_y$, and $bins_{radius}$ are the number of bins in each dimension of the accumulator. RANSAC, in turn, has cost $O(I \times N)$, where I is the number of iterations required to detect a circle, which can be arbitrarily high, depending on the noise level and the specified thresholds.

Another advantage of our technique is its independence of noise-level thresholds. For RANSAC, in particular, it is difficult to set a good distance threshold which works well for all projected samples from a point cloud. This happens because such threshold depends on the noise level, which is not uniform in most unorganized point clouds (the further a sample is from the sensor, the stronger the noise level).

3.5. Robust outlier removal

Once a circle has been detected by the technique described in Section 3.4, we perform an outlier-removal procedure before fitting a cylinder to the set of samples whose projections were used to detect the circle. For each g_i, its samples are used to obtain robust estimates for a circle center and radius.

In Robust Statistics, each estimator has a breakdown-point, a percentage of supported outliers beyond which the estimate is no longer reliable. The mean estimator has 0% breakdown-point, since a single outlier impacts its result. The median, on the other hand, is a much more robust estimator, with a breakdown-point of 50%. Like the median, there is a robust alternative to the standard deviation called *median absolute deviation* (MAD):

$$MAD(X) = k \times median(|x_i - median(X)|), \tag{2}$$

where x_i represents all individual samples in the set X, while k is required to make MAD consistent with the standard deviation estimator. For a normal distribution, $k = 1.4826$ [23].

To estimate the circle center, we take n groups of three projected samples (we set n equal to the number of samples in g_i) and from the j-th triple we estimate a circle and its center at (C_x^j, C_y^j), creating two new sets of observations:

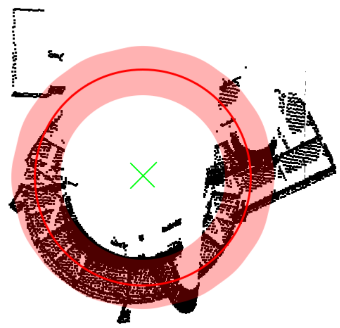

Figure 7: Outlier removal. Red circle obtained using robust estimates for its center (green X, MC_i) and radius MR_i. The pink disk represents the interval defined by Equation 3. Samples (black dots) outside this interval are discarded as outliers.

$C_{ix} = [C_x^1, C_x^2, ..., C_x^n]$ and $C_{iy} = [C_y^1, C_y^2, ..., C_y^n]$. The circle center is estimated as the median of each set, *i.e.*, $MC_i = (median(C_{ix}), median(C_{iy}))$.

The radius MR_i of the circle is obtained as the median of the set $\Delta_i = \{\delta_{i1}, \delta_{i2}, ..., \delta_{in}\}$ of distances from the projection of each sample $s_j^{g_i} \in g_i$ to MC_i. In order to robustly detect outliers with 99.7% of confidence, we define an interval I_i centered at MR_i with 3 $MADs$ of extent to each side (Equation 3). Any sample whose distance to $MC_i = median(\Delta_i)$ falls outside this interval is discarded as an outlier. This process is illustrated in Figure 7.

$$I_i = [median(\Delta_i) - 3 \times MAD(\Delta_i); median(\Delta_i) + 3 \times MAD(\Delta_i)]. \quad (3)$$

3.6. Detecting false positives

The detection of a projected circle does not guarantee that the corresponding connected component forms a cylinder. Projections of other 3D shapes, such as spheres and cones, also produces circles (Figure 8). Thus, a mechanism for detecting false positives is required.

After removing outliers from g_i, our technique uses a least-squares procedure to fit a cylindrical surface to the set of corresponding samples in the associated connected component c_i in 3D. It is based on Equation 4, which computes the

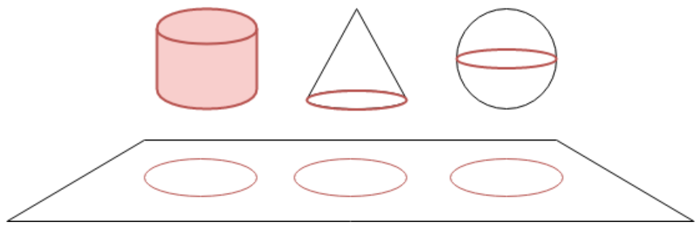

Figure 8: Samples from objects with different shapes may produce circular projections.

distance from a given sample $s_j \in c_i$ to the cylindrical surface:

$$d_j = \left(\frac{||(s_j - \mathcal{C}_c) \times (s_j - (\mathcal{C}_c + A))||}{||(\mathcal{C}_c + A) - \mathcal{C}_c||} - r \right)^2 = (||(s_j - \mathcal{C}_c) \times (s_j - \mathcal{C}_c - A)|| - r)^2 \,, \tag{4}$$

where s_j is a sample position in 3D, $\mathcal{C}_c$ is the cylinder center, A is the cylinder axis and r is the cylinder radius. And since the cross product is a linear transformation,

$$\begin{aligned} d_j = (||(s_j - \mathcal{C}_c) \times (s_j - \mathcal{C}_c) - (s_j - \mathcal{C}_c) \times A|| - r)^2 \\ = (||(\mathcal{C}_c - s_j) \times A|| - r)^2. \end{aligned} \tag{5}$$

After the fitting, a cylinder is considered valid if it meets three conditions: (i) at least 50% of the normals from the samples used to fit the cylinder are perpendicular to the fitted cylinder axis (with tolerance $\tau = 10°$); (ii) at least one quarter of the bins of the angular histogram (*i.e.*, at least $90°$) are uniformly distributed; (iii) the ratio between the fitted cylinder radius and its height must be below a threshold γ (conservatively set to 5). Conditions (i) and (ii) evaluate the quality of the fitting. Condition (iii) prevents cones, spheres, and related geometric shapes from being detected as cylinders (Figure 8). For a non-cylindrical geometric shape whose projection produces a circle, only a small section of it will be actually projected. Such a section can be bigger or smaller depending on the angular threshold τ allowed between the sample and plane normals. Regardless, it is expected that the height/radius ratio for false positive cylinders be small and much smaller than $\gamma = 5$.

3.7. Recomputing connected components

The samples associated with detected cylinders are removed from the point cloud. This may affect connected components that may not have been analyzed yet, as a sample may belong to more than one connected component (*e.g.*, consider a sample at the intersection of two cylinders). Thus, after removing samples, the connectivity of all components that have not been evaluated yet need to be re-checked. If a connected component has been split, the original component is removed and its subcomponents are added to the list of connected components.

In order to speed up this process, the largest components are analyzed first, as we want to remove the largest number of samples as soon as possible, preventing them from being projected and analyzed multiple times unnecessarily. After recalculating the connected components, they are sorted based on their number of samples.

3.8. Merging connected components belonging to the same cylinders

A cylinder may be fragmented into multiple connected components. Since our technique treats each connected component separately, a cylinder may be detected multiple times, once per fragment, and these components need to be merged. This situation is illustrated in Figure 9.

To identify the cylinders whose components should be merged, we iterate over each pair of detected cylinders $\mathcal{C}_i$ and $\mathcal{C}_j$, which should satisfy three similarity tests: (i) they must have similar orientations (*i.e.*, similar axis directions); (ii) they must have similar radii values; and (iii) they must have similar center positions. The connected components that satisfy such criteria are merged using union-find operations. After all merge operations have been performed, a resulting cylinder is obtained from each union through least-squares fitting of its samples (Figure 9 (right)).

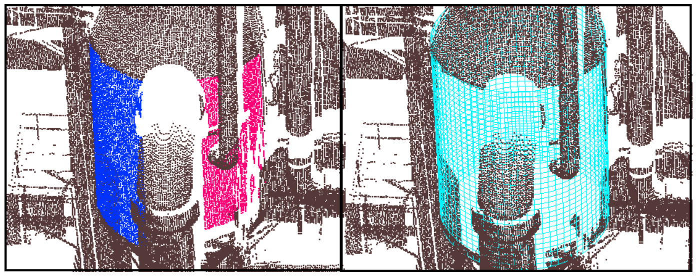

Figure 9: Merging multiple connected components belonging to a cylinder. (left) Due to partial occlusion, two connected components from the same cylinder in the Petrochemical plant dataset, marked in blue and red, are detected as possibly belonging to separate cylinders. (right) The resulting detected cylinder after the merging process.

Test (i) checks if the angle between the axes of $\mathcal{C}_i$ and $\mathcal{C}_j$ is below a certain threshold α (experimentally set to 10°). Test (ii) checks if the ratio $max(r_i, r_j)/min(r_i, r_j)$ between the radii r_i and r_j of the two cylinders is below a certain threshold β (experimentally set to 2). For test (iii), let l_i and l_j be the line segments corresponding to the limits of the projections of the samples in the connected components c_i and c_j on the axes of $\mathcal{C}_i$ and $\mathcal{C}_j$, respectively (Figure 11). l_i and l_j approximate the medial axes of the surfaces defined by c_i and c_j, respectively. For the connected components of $\mathcal{C}_i$ and $\mathcal{C}_j$ to be merged, the distance between l_i and l_j should be below a certain threshold (experimentally set as $max(r_i, r_j)/10$).

We opted for a large threshold β for the cylinders' radii ratio because the radius value estimated by least squares is not as reliable as the estimated axis, especially when just a small section of a cylinder is available (*i.e.*, a 90° arch). This situation is illustrated in Figure 10.

3.9. Complexity analysis

The cost of our cylinder detection technique is $O(PN^3)$, where P is the number of projection directions on the uniform sampling of the hemisphere (Figure 2), and N is the number of samples in the point cloud. Next, we analyze the cost of the individual steps of the algorithm.

For each projection direction, we analyze each sample and project it if its normal is approximately perpendicular to the given direction. This has cost

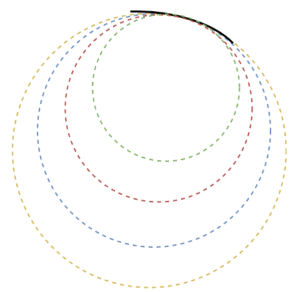

Figure 10: Radius estimation uncertainty. Four cylinders (top view) with different radii estimated from the samples shown in black (solid arch). Although the differences in radii are big, the actual fitting errors are small in all four cases.

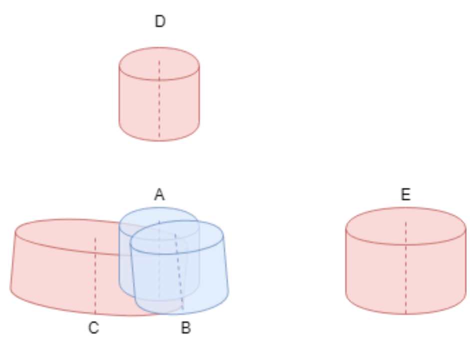

Figure 11: Evaluating multiple conditions to merge the connected components from two cylinders. Cylinders A and B will be merged because they have similar orientations, radii, and distance from their medial axes (dotted line segments) is sufficiently small. Cylinder C will not be merged with A or B because its radius is much bigger than the other two. Cylinder D, despite having same orientation and radius, will not be merged with cylinder A because the distance between their medial axes is too big. The same applies to cylinder E with respect to cylinders A and B.

$O(PN)$. For each projection, we compute the connected components using a neighborhood graph G. We compute G using k-nearest neighbors, which for an individual sample has cost $O(klog(N)) = O(log(N))$ (using a kd-tree as auxiliary data-structure, where k is the number of neighbors, and assuming $k << N$). For all N samples, the cost of computing G is $O(Nlog(N))$. Since we perform a breadth-first search on G to compute the connected components, this step has cost $O(E + N)$, where E is the number of edges (neighbors) and N is the number of vertices (samples) in G. Since, for G, $E << N$, this has cost $O(N)$.

The projection directions are refined, with more samples being included in the connected component, if necessary. Such refinement is performed only once using PCA [24], whose cost is $O(N^3)$. Thus, this whole refinement process has cost $O(PN^3)$.

For each connected component, we perform a circle recognition using a quadtree to check the intersection of the sample normals with the quadtree node cells (Algorithm 4). For each level, only one cell is subdivided, and in the worst case this cell will be intersected by rays from all samples. This has cost $O(DN)$, where D is the maximum number of quadtree subdivisions for a circle to be recognized. Therefore, this step has cost $O(PDN)$. However, since $D \leq 5$, this step has cost $O(PN)$.

For the robust outlier removal, we calculate mean and MAD of a set of samples to obtain the inlier interval. Calculating the median of a set of distances can be performed in $O(N)$, and thus this step has cost $O(N)$.

In the false-positive test, the most expensive operation is the least-squares fitting of a set of samples, whose cost is $O(N^3)$. Since one sample can be projected at most P times, this test has cost $O(PN^3)$.

For merging connected components from multiple cylinders, each cylinder is compared to all the others. Since the minimal theoretical number of samples required to represent a cylinder is five [17], in the worst case we have $N/5$ cylinders. Thus, this step has cost $O(N^2)$. Therefore, the total cost of our technique is $O(PN^3)$.

3.10. Complexity analysis of compared techniques

For completeness, this section provides the time complexity of the methods compared against our technique in Section 4. The cost of the standard version of RANSAC is $O(IN)$, where I is the number of iterations used to detect one instance of the queried model. The technique proposed by Liu et. al [11] is composed by two main steps: (i) detection of the planes in the scene, and (ii) detection of circles in the projections on each plane. For the detection of the planes, all samples are mapped to a Gauss map, where it is partitioned into cells and one of them will be chosen according to a maximal connected component criterion. This step can be performed in linear time. The second step uses a standard RANSAC. Hence, the technique has a total cost of $O(INL)$, where L is the number of planes in the scene. The technique proposed by Tran et. al [20] is a region growing algorithm consisting of the selection of seed samples followed by an iterative fitting step for each seed. Each such step consists of applying PCA to a neighborhood around the seed sample. The cost of PCA is $O(N^3)$ and the number of iterations is constant. Since each sample in the point cloud can potentially be a seed sample, the total cost of this technique is $O(N^4)$. The technique proposed by Ahmed et al. [12] consists of partitioning the point cloud into regular intervals along the three main axes (X, Y, Z) and then applying a Hough transform to the projections of the samples within each such interval to detect circles. The cost of the standard Hough transform to detect circles is $O(N^3)$ (since it requires a three-dimensional feature space), and therefore this technique has cost $O(TN^3)$, where T is the number of intervals used to sample the X, Y, and Z directions. The technique proposed by Schnabel et. al [18] presents many improvements to the standard RANSAC, but its asymptotic cost is still the same, *i.e.*, $O(IN)$. From this analysis, one can sort the compared techniques according to computational efficiency (from fastest to slowest) as: Schnabel et al. [18], Liu et al. [11], ours, Ahmed et al. [12] and finally Tran et al. [20]. This is consistent with the results in Table 2.

4. Results

In order to evaluate our technique, we used five datasets consisting of two synthetic (including a complex one that models an oil refinery) and three obtained from real scenes (Figure 12). Such models were chosen to stress the ability of cylinder-detection techniques to handle different features and configurations found in actual installations. The datasets are: (i) *Synthetic scene*: a scene containing one cube and two cylinders on top of a plane. The positions of these samples were corrupted using a uniform distribution of noise values ranging from 0 to 1% of the side of the point cloud's cubic bounding box. (ii) *Synthetic oil refinery*: an oil refinery scene modeled using 3D software and converted to point cloud by uniformly sampling the polygonal mesh; (iii) *Pump room*: a sewer treatment plant pump room; (iv) *Petrochemical plant*: a frontal scan from a petrochemical site; and (v) *Boiler room*: a set of integrated scans from the complex environment of a boiler room consisting of almost six million samples. The real datasets were obtained from Leica's public sample repository [25]. The ground truth for each dataset was obtained by manually selecting the samples of each cylinder and using them to least-squares fit a cylindrical surface. Table 1 shows the number of cylinders, the percentage of the scene area covered by cylindrical surfaces, the density level, and the capture method of each dataset.

Table 1: Dataset Information: Number of Cylinders, Percentage of Cylindrical Scene Coverage, Density and Source (sensor).

	#samples	#cylinders	% cylinder regions	density	source (sensor)
Synthetic scene	138,000	2	34	high	synthetic
Synthetic oil refinery	499,976	3	22	high	synthetic
Pump room	166,976	13	13	low	LiDAR (single view)
Petrochemical plant	358,116	8	13	high	LiDAR (single view)
Boiler room	5,990,481	21	7	non-uniform	LiDAR (multiple views)

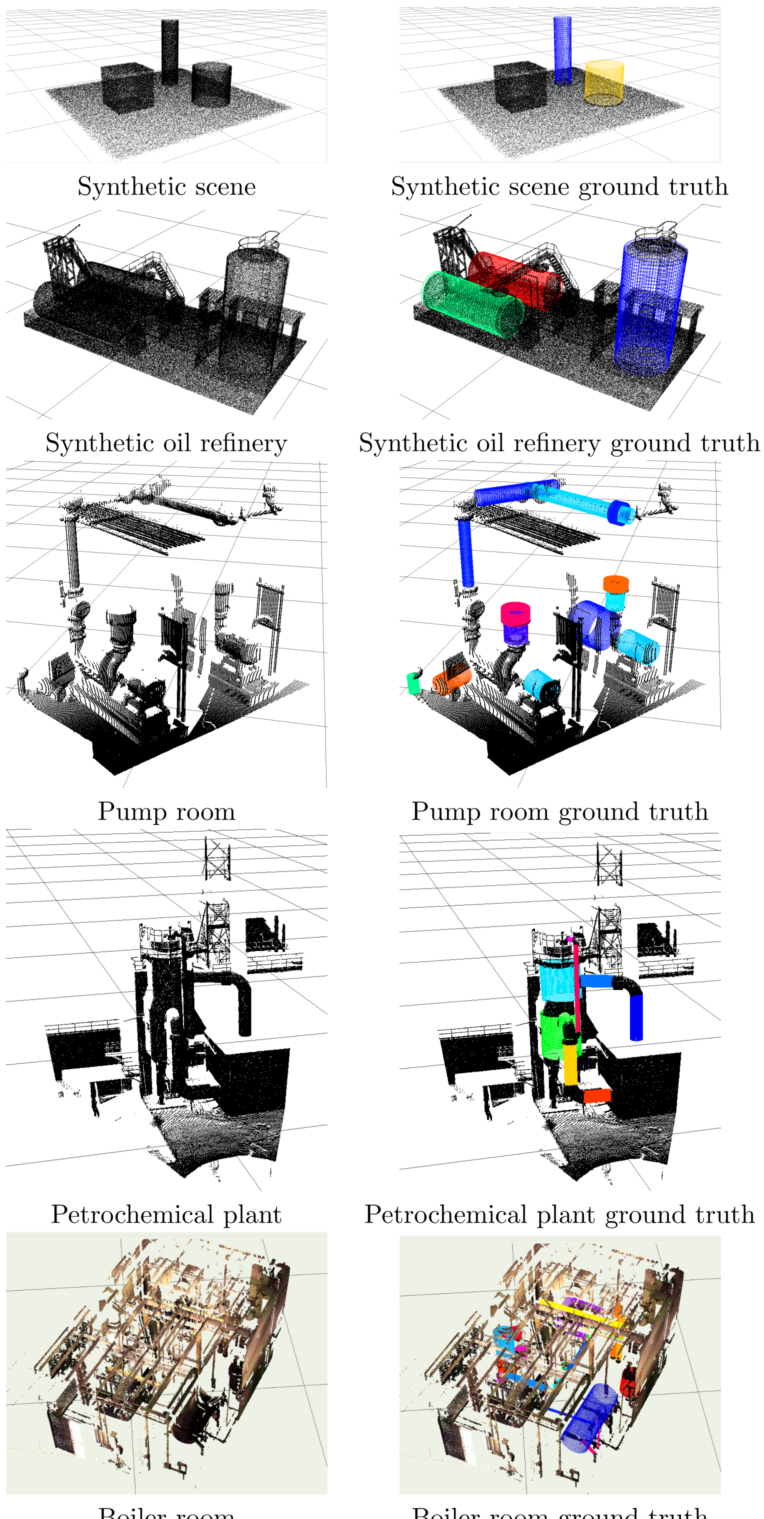

Figure 12: Ground-truth for each dataset.

4.1. Evaluation metrics

In order to objectively compare our technique to previous ones, we adapted three well-known metrics from Information Retrieval to our context: precision, recall, and F1-score. *Precision* is the percentage of correctly retrieved instances (*i.e.*, true positive) among all retrieved ones (*i.e.*, true positive + false positive). *Recall* is the percentage of correctly retrieved instances among all correct instances (*i.e.*, true positive + false negative). Finally, *F-1 Score* is the harmonic mean of precision and recall. These measures are summarized by Equations 6 to 8.

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{6}$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{7}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{8}$$

We say that a detected cylinder corresponds to a true positive if its orientation differs by no more than 20° from the ground truth's orientation, and they share at least 50% of their samples. The first condition ensures that the cylinders have similar orientations, while the second ensures that they are centered at approximately the same position in space.

4.2. Experiments

We implemented our technique in C++ using Eigen[26] as our linear algebra library. We compared it against the most popular as well as the most recent approaches for cylinder detection using the five datasets shown in Figure 12. These methods are based on RANSAC, Hough transform, and region growing. All experiments were executed on a Intel® Core$^{\text{TM}}$ i7-7700K 4.20GHz CPU with 32GB RAM. Next, we describe the compared techniques and present the reasons for their choices.

In the Hough transform category, the work of Ahmed et al. [12] was chosen because it corresponds to a state-of-art technique for cylinder detection. In terms of RANSAC, the work of Schnabel et al. [18] was chosen since it is a popular and traditional RANSAC approach. Liu et al. [11] was chosen because it is a recent approach to detect pipes, which is also the main intent of our work. Finally the work of Tran et al. [20] was chosen since, to our knowledge, it is the most recent work on cylinder detection. It is based on region growing.

With the exception of Schnabel et al. [18], we could not find implementations of the other techniques. Thus, we implemented them ourselves, as faithfully as possible, also in C++. In order to prevent bias, for techniques that require normals, we used the same normal estimation approach used for our technique. The normals were estimated using FAST-MCD [27], in a neighborhood of size 50. Table 2 summarizes the results of our experiments, including values for precision, recall, F1-score, and execution time in seconds. The average and standard deviation of the elapsed time were calculated upon 10 executions. Note that our technique achieved the best F-1 score for four out of the five evaluated datasets, while still maintaining a competitive running time. The ground truths and the cylinders detected by each technique are shown in Figure 13. High-resolution versions of these results are available in the supplementary materials, which we encourage the readers to inspect.

For the *Synthetic scene* dataset, our technique achieved the best results. RANSAC-based techniques such as Schnabel et al.'s tend to detect planar surfaces as spurious incomplete cylinders with very large radius. Thus, for this dataset, Schnabel et al.'s detected the base plane as a spurious cylinder. Liu et al.'s was unable to detect one of the cylinders. Ahmed et al.'s detected the cylinders, but with some missing slices. Tran et al.'s detected spurious cylinders in regions with high curvature, such as the edges of the cube.

For the *Synthetic oil refinery* dataset, our technique was able to detect two out of the three cylinders. For one of the cylinders, it overextended its height, as some close-by samples from the ground plane, whose normals are perpendicular to the axes of these cylinder, were mistaken as belonging to these cylinders. Schnabel et al.'s again detected planes as spurious cylinders. Liu et al.'s was able to obtain a clean detection of the two horizontal cylinders. However, for

Table 2: Performance of the evaluated techniques on each dataset. Number of detected cylinders over total number of cylinders (#), Precision (P), Recall (R), F1-Score (F1), Elapsed time Average (T_μ) and Standard deviation (T_σ) both in seconds. Best results in bold.

	#	P	R	F1	T_μ	T_σ
Synthetic scene						
Schnabel et al. [18]	2/2	0.41	**0.94**	0.58	0.29	0.04
Liu et al. [11]	1/2	**0.99**	0.41	0.58	0.44	0.002
Tran et al. [20]	2/2	0.91	0.33	0.49	4.33	4.37
Ahmed et al. [12]	2/2	1.0	0.29	0.45	7.78	1.78
Our technique	2/2	0.98	0.88	**0.93**	1.61	0.17
Synthetic oil refinery						
Schnabel et al. [18]	3/3	0.43	**0.93**	0.59	0.29	0.07
Liu et al. [11]	3/3	**0.99**	0.33	0.50	2.06	0.02
Tran et al. [20]	3/3	0.86	0.57	**0.68**	65.33	49.86
Ahmed et al. [12]	3/3	**0.99**	0.35	0.52	27.71	8.11
Our technique	2/3	0.68	0.64	0.66	7.18	0.13
Pump room						
Schnabel et al. [18]	2/13	0.08	0.33	0.14	0.20	0.05
Liu et al. [11]	1/13	0.17	0.08	0.11	0.74	0.01
Tran et al. [20]	3/13	0.65	0.31	0.42	11.87	6.45
Ahmed et al. [12]	1/13	0.39	0.12	0.19	20.59	1.81
Our technique	3/13	**0.77**	**0.38**	**0.51**	0.95	0.04
Petrochemical plant						
Schnabel et al. [18]	6/8	0.14	**0.70**	0.24	0.46	0.04
Liu et al. [11]	4/8	0.27	0.31	0.29	1.31	0.01
Tran et al. [20]	3/8	0.85	0.30	0.45	12.21	9.10
Ahmed et al. [12]	5/8	**0.86**	0.42	0.56	5.23	0.58
Our technique	5/8	0.53	0.62	**0.57**	2.27	0.11
Boiler room						
Schnabel et al. [18]	1/21	0.04	0.23	0.07	7.83	1.51
Liu et al. [11]	1/21	0.02	0.08	0.03	29.45	2.17
Tran et al. [20]	2/21	0.50	0.13	0.20	1,150.73	30.27
Ahmed et al. [12]	0/21	0	0	-	197.42	38.14
Our technique	12/21	**0.73**	**0.56**	**0.63**	47.23	1.01

the vertical cylinder, the circle detection did not obtain a good fit, reducing its recall. Ahmed et al.'s detected all cylinders but for the horizontal ones it only detected small sections of them. Tran et al.'s detected spurious cylinders on the edges.

The *Pump room* dataset proved to be quite hard because it only contains partial views of all cylinders. Despite only detecting 3 of the 13 cylinders in the scene, our technique still obtained the best results. Since this scene is not perfectly aligned with the X-Z axis (it is slightly rotated around the Y axis), Ahmed et al.'s performed badly on this dataset, only being able to detect one cylinder along the Y axis. Both RANSAC based techniques - Schnabel et al.'s and Liu et al.'s - detected spurious cylinders.

For the *Petrochemical plant* dataset, our technique was able to detect 5 out of the 8 cylinders. An occlusion split one cylinder into two sufficiently far apart from each other, and each part had an arc lesser than $90°$ during the circle detection procedure. Nevertheless, our technique still obtained the highest F1-score. Once again, RANSAC-based techniques detected planar surfaces as spurious cylinders. Ahmed et al's technique achieved the highest precision, but also detected a large spurious cylinder. Tran et al's detected 3 out of the 8 cylinders, and had the lowest recall.

The *Boiler room* is the most complex dataset and proved to be the hardest among all five. For such dataset, our technique achieved the best precision, recall, and F1-score. It was able to detect 12 out of 21 cylinders. For comparison, Ahmed's, Schnabel, Liu's, and Tran's techniques detected 0, 1, 1, and 2 cylinders, respectively.

For four out of five datasets, our technique obtained the best F-1 score, demonstrating its superior accuracy. We should emphasize that for the experiments reported in Table 2, we have fine-tuned the parameters of each competing technique for the individual datasets in order for them to obtain their best results in each case. For our technique, on the other hand, we used the same set of default parameter values for all datasets, demonstrating its robustness and independence of parameter tuning. The accuracy of our technique results

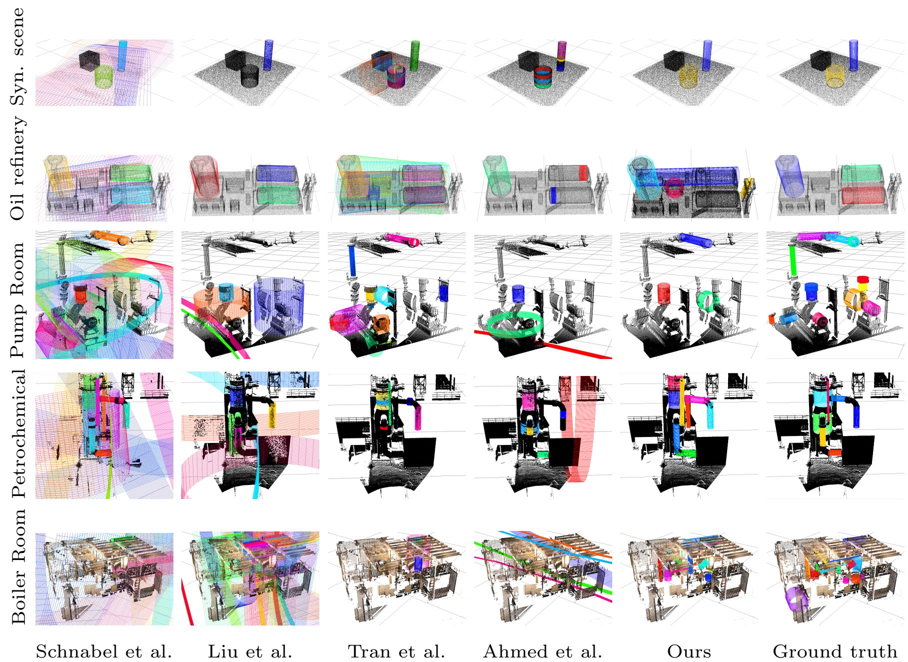

Figure 13: Cylinders detected by the compared techniques for all datasets. Ground truth is shown in the rightmost column. For each pair of technique and dataset, the detected cylinders have been highlighted using different colors. Black dots represent samples treated as outliers by each technique. Left-click on the images to zoom in and inspect the details. Larger versions of these images are also available in the supplemental material.

from the ability of our circle-detection algorithm to automatically filter outliers (both in terms of positions and normal directions), making it more robust to noise and, consequently, more independent of parameter tuning. Although our technique is not the fastest (Schnabel et al.'s being first), it is much faster than Tran et al.'s and Ahmed et al.'s. Another point to be stressed is that our technique is deterministic, unliked RANSAC-based solutions such as Schnabel et al.'s and Liu et al.'s, whose results may very among multiple executions on the same dataset.

4.3. Noise-Handling Evaluation

In order to evaluate the techniques' robustness to noise, we performed an experiment that consisted of processing versions of the original datasets containing increasing amounts of noise. For each dataset, we perturbed the po-

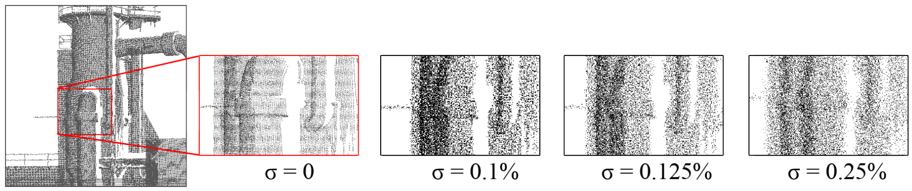

Figure 14: Part of the petrochemical plant dataset shown with increasing levels of noise.

sition of each sample using Gaussian distributions with standard deviations of $\sigma = 0.1\%$, $\sigma = 0.125\%$, and $\sigma = 0.25\%$, respectively, relative to the size of the (cubic) bounding box of the dataset. All sample normals were re-estimated, resulting in datasets containing not only noisy positions but also noisier normals. Figure 14 illustrates a portion of one of the datasets after the perturbations.

We evaluated all techniques on these noisier datasets. For each combination of technique and dataset, we used the same parameter values used for producing Table 2, *i.e.*, for each technique, with the exception of our own, we fine-tuned the parameter values to each original dataset and used them for that dataset with all noise levels. For our technique, on the other hand, we used the same default parameter values regardless of dataset or noise level. For these noisier datasets, we calculated the ratio of detection, *i.e.*, the number of correctly detected cylinders over the total number of cylinders in the scene. The results of our technique for each dataset can be found in Table 3. A weighted average ratio of detection was calculated for all techniques, as:

$$W_\sigma = \sum_{i \in dataset_\sigma} \frac{C^i_{detected}}{C^i_{total}} \times \frac{C^i_{total}}{\sum\limits_{j \in dataset_\sigma} C^j_{total}}, \tag{9}$$

where W_σ is the weighted average ratio of detection for the noise level σ, $C^k_{detected}$ and C^k_{total} are, repectively, the number of detected cylinders and the total number of cylinders in dataset k. These weighted averages are shown in Figure 15.

Although all techniques experienced a performance drop, ours maintained the best performance at all noise levels being able to detect, on average, at least 32% more cylinders than the second-best ranked technique, despite of using the

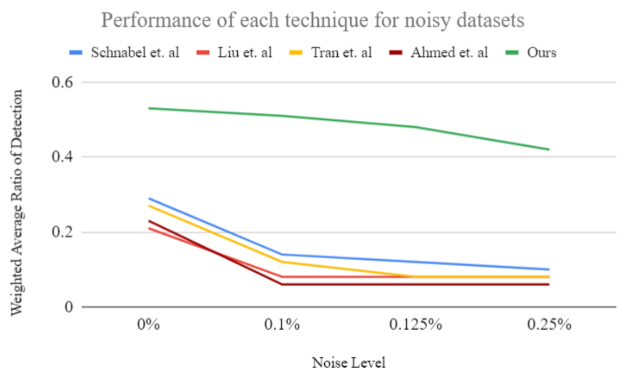

Figure 15: Weighted average performance of each technique considering all datasets with increasing amount of added Gaussian noise. Standard deviation corresponding to 0.1%, 0.125%, and 0.25% of the size of each dataset's bounding box.

same set of default parameters for all evaluated datasets.

Table 3: Ratio of detected cylinders of our technique in each one of the evaluated datasets containing increasing amount of noise.

	$\sigma = 0\%$	$\sigma = 0.1\%$	$\sigma = 0.125\%$	$\sigma = 0.25\%$
Synthetic scene	2/2	2/2	2/2	2/2
Synthetic oil refinery	2/3	2/3	1/3	2/3
Pump room	3/13	3/13	3/13	2/13
Petrochemical plant	5/8	4/8	4/8	3/8
Boiler room	12/21	13/21	13/21	11/21

4.4. Limitations

Currently, during the analysis of the connected components, we only check if the normal of each sample is perpendicular to the normal of the projection plane. This can add undesired samples to the connected component, as shown in the Oil Refinery dataset, where the cylinders were overextended using some near-by samples from the plane (whose normals are also perpendicular to the axes of the two cylinders). A further verification would need to be done in order to disassociate such samples from the cylinder.

One disadvantage of our technique with respect to RANSAC- and Hough-transform-based solutions is that, if a cylinder is fragmented into disconnected sections whose projections form arcs smaller than 90° each, the cylinder will not be detected, as each arc does pass the circle recognition test, demonstrated in the Petrochemical plant dataset. Addressing this issue requires obtaining new samples covering the missing regions. Also, the number k used to build the connectivity graph G may impact the performance of our technique. If k is chosen too small, regions which are visually connected might be disconnected in the graph, making some cylinders to go undetected, due to the reasons just mentioned. If, on the other hand, k is chosen too big, G may connect regions which are visually disconnected, causing their projections to fail the circle recognition test. According to our experience, $k = 50$ works well for all tested point cloud configurations.

5. Conclusion

We presented a fast and robust technique for automatic detection of cylinders with arbitrary orientations in unorganized point clouds. It consists of orthographically projecting the point cloud along multiple directions and refining them, detecting circular projections, removing outliers, fitting cylinders to connected components in 3D, and merging them when appropriate. We also presented a circle-detection technique that is faster than RANSAC- and Hough-transform-based solutions, and does not require the specification of noise-level thresholds.

We demonstrated the effectiveness of our approach by performing a detailed comparison with the most popular as well as with the most recent approaches for cylinder detection. Such techniques were evaluated on five datasets chosen to stress different aspects and configurations found in real environments. Our technique achieved the best F1-score on all datasets. For these experiments, the parameters used by the competing techniques were individually tuned for each dataset in order to produce their best results in each case. For our technique, on the other hand, we used the same set of default parameter values for all datasets, showing its robustness and independence to parameter tunning, and ability to handle point clouds in general.

Acknowledgments

This work was sponsored by CNPq-Brazil (fellowships and grants 312975/2018-0, 130895/2017-2, and 423673/2016-5) and by ONR Global Award # N62909-18-1-2131. The LiDAR datasets (*Pump room*, *Petrochemical plant*, and *Boiler room*) are from Leica (http://hds.leica-geosystems.com/en/Support-Downloads-Example-Databases_29453.htm).

References

[1] R. Qiu, Q.-Y. Zhou, U. Neumann, Pipe-run extraction and reconstruction from point clouds, in: European Conference on Computer Vision, Springer, 2014, pp. 17–30.

[2] G. Pang, R. Qiu, J. Huang, S. You, U. Neumann, Automatic 3D industrial point cloud modeling and recognition, in: 2015 14th IAPR International Conference on Machine Vision Applications (MVA), IEEE, 2015, pp. 22–25.

[3] V. Raja, K. J. Fernandes, Reverse Engineering: an industrial perspective, Springer Science & Business Media, 2007.

[4] G. Wang, Z. Houkes, G. Ji, B. Zheng, X. Li, An Estimation-based approach for range image segmentation: on the reliability of primitive extraction, Pattern Recognition 36 (1) (2003) 157–169.

[5] Y. Zhou, G. Jiang, Y. Lin, A Novel finger and hand pose estimation technique for real-time hand gesture recognition, Pattern Recognition 49 (2016) 102–114.

[6] J. Shen, W. Yang, Q. Liao, Part Template: 3D representation for multiview human pose estimation, Pattern Recognition 46 (7) (2013) 1920–1932.

[7] M. Sigalas, M. Pateraki, P. Trahanias, Full-body pose tracking—the top view reprojection approach, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (8) (2015) 1569–1582.

[8] R. B. Rusu, A. Holzbach, M. Beetz, G. Bradski, Detecting and segmenting objects for mobile manipulation, in: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, IEEE, 2009, pp. 47–54.

[9] F. Lafarge, R. Keriven, M. Brédif, H.-H. Vu, A hybrid multiview stereo algorithm for modeling urban scenes, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (1) (2012) 5–17.

[10] R. C. Bolles, M. A. Fischler, A RANSAC-based approach to model fitting and its application to finding cylinders in range data., in: IJCAI, Vol. 1981, 1981, pp. 637–643.

[11] Y.-J. Liu, J.-B. Zhang, J.-C. Hou, J.-C. Ren, W.-Q. Tang, Cylinder detection in large-scale point cloud of pipeline plant, IEEE Transactions on Visualization and Computer Graphics 19 (10) (2013) 1700–1707.

[12] M. F. Ahmed, C. T. Haas, R. Haas, Automatic detection of cylindrical objects in built facilities, Journal of Computing in Civil Engineering 28 (3) (2014) 04014009.

[13] P. V. C. Hough, Method and means for recognizing complex patterns, US Patent 3,069,654 (Dec. 18 1962).

[14] T. Rabbani, F. Van Den Heuvel, Efficient Hough transform for automatic detection of cylinders in point clouds, ISPRS WG III/3, III/4 3 (2005) 60–65.

[15] A. K. Patil, P. Holi, S. K. Lee, Y. H. Chai, An Adaptive approach for the reconstruction and modeling of as-built 3D pipelines from point clouds, Automation in Construction 75 (2017) 65–78.

[16] R. Figueiredo, A. Dehban, P. Moreno, A. Bernardino, J. Santos-Victor, H. Araújo, A Robust and efficient framework for fast cylinder detection, Robotics and Autonomous Systems 117 (2019) 17–28.

[17] T. Chaperon, F. Goulette, Extracting cylinders in full 3d data using a random sampling method and the Gaussian image, in: Vision Modeling and Visualization Conference 2001 (VMV-01), 2001.

[18] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, in: Computer Graphics Forum, Vol. 26, Wiley Online Library, 2007, pp. 214–226.

[19] Y.-H. Jin, W.-H. Lee, Fast cylinder shape matching using random sample consensus in large scale point cloud, Applied Sciences 9 (5) (2019) 974.

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

[21] A. Nurunnabi, Y. Sadahiro, R. Lindenbergh, Robust cylinder fitting in three-dimensional point cloud data, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 42 (1/W1) (2017).

[22] B. Keinert, M. Innmann, M. Sänger, M. Stamminger, Spherical fibonacci mapping, ACM Transactions on Graphics (TOG) 34 (6) (2015) 193.

[23] P. J. Rousseeuw, C. Croux, Alternatives to the median absolute deviation, Journal of the American Statistical Association 88 (424) (1993) 1273–1283.

[24] H. Zou, T. Hastie, R. Tibshirani, Sparse principal component analysis, Journal of Computational and Graphical Statistics 15 (2) (2006) 265–286.

[25] Leica cyclone/cloudworx example databases, `https://hds.leica-geosystems.com/en/29453.htm`, accessed: 2018-07-01.

[26] Eigen C++ library for linear algebra, `http://eigen.tuxfamily.org/index.php?title=Main_Page`, accessed: 2018-07-01.

[27] P. J. Rousseeuw, K. V. Driessen, A Fast algorithm for the minimum covariance determinant estimator, Technometrics 41 (3) (1999) 212–223.