Detecting Objects in Scene Point Cloud: A Combinational Approach

Jing Huang and Suya You
University of Southern California
Los Angeles, CA, USA
{huang10,suya.you}@usc.edu

Abstract

Object detection is a fundamental task in computer vision. As the 3D scanning techniques become popular, directly detecting objects through 3D point cloud of a scene becomes an immediate need. We propose an object detection framework combining learning-based classification, local descriptor, a new variance of RANSAC imposing rigid-body constraint and an iterative process for multi-object detection in continuous point clouds. The framework not only takes global and local information into account, but also benefits from both learning and empirical methods. The experiments performed on the challenging ground Lidar dataset show the effectiveness of our method.

1. Introduction

Object detection is one of the most basic tasks in computer vision. There are numerous works focusing on detection of human, faces, cars, ships, daily life objects, etc. Until recently, however, most methods work on the 2D image data. As the sensor technology develops fast these years, 3D point clouds of the real scenes have been increasingly popular and precise, while there are much fewer methods trying to directly detect objects from the 3D data. In this paper, we focus on the task to detect some user-specified target objects, e.g. industrial parts (Fig. 1), from a 3D scene point cloud (Fig. 2).

3D point cloud data give certain advantages compared to 2D image data, e.g. the spatial geometric structure are clear per se. However, there are quite a few challenges for 3D point cloud data detection. Firstly, the texture information is not as clear as 2D images; secondly, 3D data can still be affected by noises and occlusions, in a different way from 2D data; thirdly, the objects have more degrees of freedom in the 3D space, increasing the difficulty in alignment; finally, the point cloud data are typically very large with millions of points even in a single scene, thus the efficiency of the algorithm is vital for an algorithm.

We propose a combinational approach to deal with the challenges above. Our main contribution includes: (1) Combine various aspects of point cloud processing to form a workable hierarchical scene understanding system upon complicated point clouds; (2) Combine SVM and FPFH descriptor [22] in per-point classification; (3) Propose an efficient variant of RANSAC with rigid body constraint.

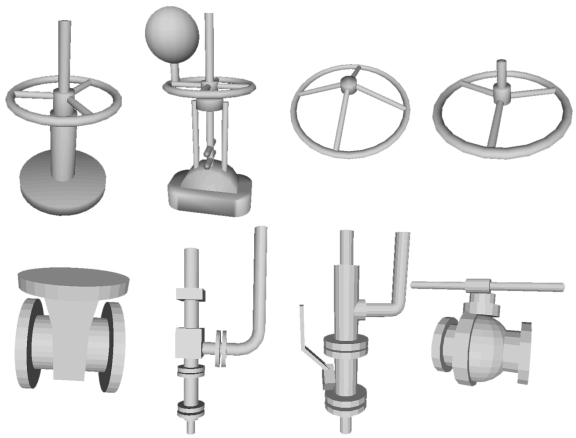


Figure 1. The CAD models of the part targets, which will be converted into point clouds by a virtual scanner in our system.

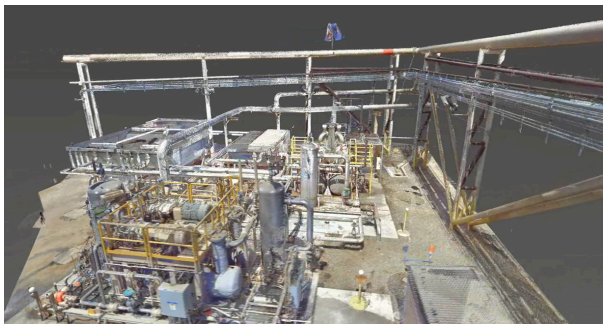


Figure 2. The colored visualization of the highly complicated scene point cloud we are dealing with.

2. Related Work

Previous works in point cloud processing could be divided into several categories based on their focus: segmentation, classification, matching, modeling, registration and

detection. Object detection in the scene point cloud is a systematic work that typically requires multiple techniques in different aspects.

There are numerous works processing on different types of point cloud data. One big category focuses on urban scenes. For example, [1], [2] and [4] try to detect vehicles; [5] and [6] concentrate on detecting poles; while [3] tries to detect both vehicles and posts as well as other small urban objects. Another category deals with indoor scenes. [7] used a model-based method to detect chairs and tables in the office. [8] aims at classifying objects such as office chair. [9] uses a graphical model to capture various features for indoor scenes. [18] tries to obtain 3D object maps from the scanned point cloud of indoor household objects. However, very few of them work on industrial scene point cloud: [10] reviewed some techniques used for recognition in industry as well as urban scenes. [17] presents a RANSAC algorithm that could detect basic shapes, including planes, spheres, cylinders, cones and tori, in the point clouds. However, it does not deal directly with the complex shapes that might represent a part.

Extensive studies have been made on the 3D local descriptors. For example, Spin Image [25] is one of the most widely used 3D descriptor, and has variations such as [29]. Other 3D descriptors are extended from 2D descriptors, including 3D Shape Context [30], 3D SURF [26] and 3D SSIM [19]. Heat Kernel Signature [27] and its variation [28] apply to non-rigid shapes. We will use two local descriptors in different stages of our system i.e. FPFH [22] in point classification and 3D-SSIM for matching.

Learning-based method is widely used in detection and/or classification tasks. One of the most successful learning-based framework is the boosted cascade framework [11], which is based on AdaBoost [12] that selects the dominating simple features to form rapid classifiers. Other common techniques include SVM (e.g. [13]), Conditional Random Field (e.g. [14]) and Markov networks (e.g. [15] [16]), among which SVM is famous for its simple construction yet robust performance. MRF-based methods focus on the segmentation of 3D scan data, which could be an alternative for segmentation and clustering in our framework.

3. System Overview

We depict the brief pipeline in Fig. 3.

The target library contains both mesh models and point clouds. If a mesh is given, a virtual-scanning process is used to create a corresponding point cloud in the library. The virtual scanner simulates the way a real scanner works, using a Z-buffer scan conversion and back-projection to eliminate points on hidden or internal surfaces. The point clouds in the target library are pre-processed so that features and descriptors are prepared for matching.

Another part of offline processing is the training of the

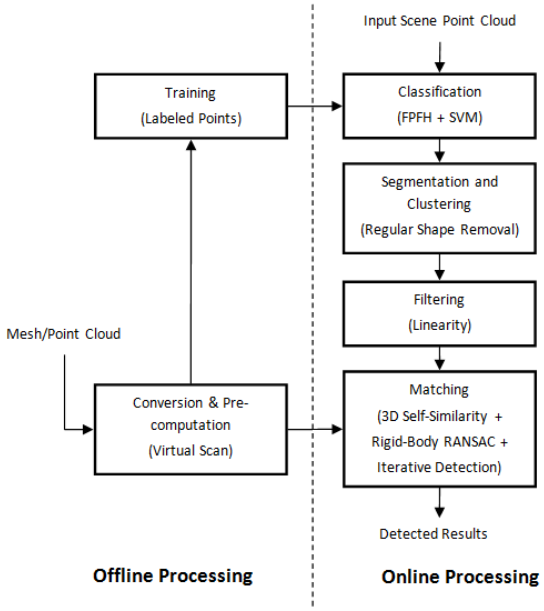


Figure 3. Flowchart of our system.

SVM for the so-called *regular* points, including plane, pipe and edge points. We calculate the Fast Point Feature Histogram [22] for each point, then feed several trunk of a positive/negative clusters in the SVM trainer.

During online processing, we first calculate the FPFH at each point and apply SVM-test on them. Points are thus classified as one of the 4 regular categories (e.g. plane) or others. Large connected components of the same category are then extracted and fitted, while the remaining points are clustered based on Euclidean distance.

Next, each cluster is passed through a cluster filter. The cluster filter is designated to consist of one or several filters, based on application, that can rule out or set aside clusters with or without certain significant characteristic. We currently support one filter i.e. linearity filter.

The clusters that pass the test of the filter will be matched with the targets in the library. The descriptors for the candidate clusters generated online are compared against the descriptors for the targets generated offline and the transformation is estimated if possible.

The key matching step is the feature comparison, the process of comparing the feature representations with point descriptors between the candidate clusters and part library targets. Initially all nearest-neighbor correspondences, or pairs of features, with the Nearest Neighbor Distance Ratio (NNDR) value, are computed and then, a greedy filtering strategy is used to look for the top four correspondences that fit the distance constraint. A transformation is estimated based on all correspondences in accord with the hypothesis, and refined through Gram-Schmidt Orthogonalization. The percentage of aligned points will be used as the matching score. If the matching score between a cluster and a target

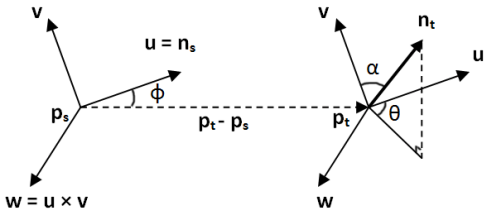


Figure 4. Illustration of Point Feature Histogram calculation.

is higher than some threshold, the cluster is considered to be a detected instance of the target.

In case that there are multiple targets in a single cluster, we iteratively remove the aligned part and check the remaining part of the cluster until it's small enough.

4. SVM-based Point Classification

We initially observed that, in our data set of large outdoor industrial scene, a large portion of the points belong to basic geometrical shapes, mainly planes (e.g. ground, ladders and boxes) and pipe-shapes (cylindrical pipes, bent connection, posts). Therefore, removing large clusters of such points will largely ease and accelerate our processing and help us focus on interested objects that we would like to detect.

4.1. Fast Point Feature Histogram

We select the 33-dimensional Fast Point Feature Histogram (FPFH) as our descriptor. The reason for selecting FPFH could be found in Section 8.1.

FPFH is an approximate and accelerated version of Point Feature Histogram (PFH) [21]. PFH uses a histogram to encode the geometrical properties of a point's neighborhood by generalizing the mean curvature around the point. The histogram representation is quantized based on all relationships between the points and their estimated surface normals within the neighborhood (Fig. 4).

The local frame for computing the relative difference between two points p_s and p_t is defined in Equation 1.

$$\begin{cases} \vec{u} = \vec{n}_s \\ \vec{v} = \vec{u} \times \dfrac{(p_t - p_s)}{\|p_t - p_s\|_2} \\ \vec{w} = \vec{u} \times \vec{v} \end{cases} \tag{1}$$

With this frame, the difference between the point-normal pair can be represented by the following angles (Equ. 2):

$$\begin{cases} \alpha = \vec{v} \cdot \vec{n}_t \\ \phi = \vec{u} \cdot \dfrac{(p_t - p_s)}{\|p_t - p_s\|_2} \\ \theta = \arctan(\vec{w} \cdot \vec{n}_t, \vec{u} \cdot \vec{n}_t) \end{cases} \tag{2}$$

These angles are then quantized to form the histogram.

FPFH [22] reduces the computational complexity of PFH from $O(nk^2)$ to $O(nk)$, where k is the number of neighbors for each point p in point cloud P, without losing much of the discriminative power in PFH:

$$FPFH(p_q) = PFH(p_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_k} \cdot PFH(p_k). \tag{3}$$

In our experiment, we use the version implemented in the open-source Point Cloud Library (PCL) [23].

4.2. Classification by SVM

In the offline training stage, We manually select and label about 75 representative small trunks of point cloud, adding up to around 200k labeled points.

For support vector machine, we use LIBSVM package [31] with radial basis function (RBF) as kernel, in which parameters $C = 8$ and $\gamma = 0.5$. Details of SVM could be found in [31] and here we focus on the selection of training data that determines the property of the classifier.

4.3. More Regular Classes: Edge and Thin Pipe

During experiments, we found that near places where two or more planes intersect, some points would not be classified as plane point due to the interference of another plane in their neighborhood. On the other hand, these points obviously do not belong to parts when they group together as a large cluster. Therefore, we assign them to another category, namely the Edge.

Besides edges, we also found some thin pipes missing in pipe detection. Experiments show that simply adding them in the training dataset might have negative effects on pipes with larger sizes, which suggests that they may need to be regarded as a separate category from pipes (partially due to the neighborhood size of the FPFH descriptor).

To judge the generalization ability, or distinctiveness of pipes in different sizes, we perform a series of cross validation, summarized in Table 1. We can see that the 10/15/20-cm pipe classifiers could classify the 10/15/20-cm pipes interchangeably, while 5-cm pipe classifier will distinguish the 5-cm pipe from the others. This evidence also offers support to separate the category Thin-Pipe from the category Pipe. If we need to distinguish between 10/15/20-cm pipes, however, we may add the other sizes as negative examples to get more precise boundaries between them.

We perform SVM 4 times, once for each category. Points are thus labeled as plane, pipe, edge, thin-pipe or others. Figure 5 shows the classified pipe points and plane points. Note that, some pipe-shaped objects e.g. tanks are so huge that locally they are like planes. In our experiments, we found it better for segmentation if we label the large tanks with small curvature as planes rather than cylinders.

Training/Testing	5 cm	10 cm	15 cm	20 cm
5 cm	Y	N	N	N
10 cm	N	Y	Y	Y
15 cm	N	Y	Y	Y
20 cm	N	Y	Y	Y

Table 1. Cross validation result for pipes of different sizes. The left column means the training data, and the top row means the testing data. Y means at least 80% of the testing points are classified as pipe, while N means the opposite.

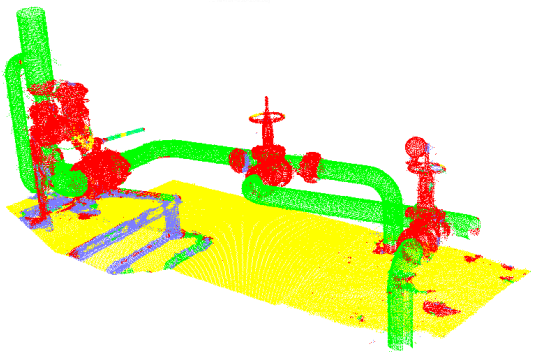


Figure 5. Classification result of pipe points (in green), plane points (in yellow), edge points (in blue), thin-pipe points (in dark green) and the others (in red).

5. Segmentation and Clustering

There are many segmentation methods, e.g. the min-cut [34] approach and other approaches derived from 2D cases. However, since we have made quite confident classification of non-part (background) points, the segmentation task could be done in a fast and light-weighted manner.

We iteratively select a random unvisited seed point and expand it to unvisited neighboring points within a given Euclidean distance using the classical Flood-Fill algorithm. The neighbor threshold is determined by the granularity of the input cloud. Apparently after finite steps the residual cloud will be divided into a number of disjoint clusters.

We apply the clustering routine for five times. First we do clustering on points labeled as one of the four categories and get a list of plane/pipe/edge/thin-pipe clusters, respectively. Then we subtract the *big* clusters from the original point cloud. This is important since we don't want to remove small area of regular shapes that might lie on a big part. Finally, clustering is performed on the remaining points that we believe to be part points.

Using $S(C)$ to denote the set of points in category C, the algorithm could be written as:

$$\begin{aligned} S(Candidate) :=& S(All) - S(Plane) - S(Pipe) \\ & - S(Edge) - S(ThinPipe). \end{aligned} \tag{4}$$

Finally, we can formally make an extendable definition of the candidates by Equ. 5:

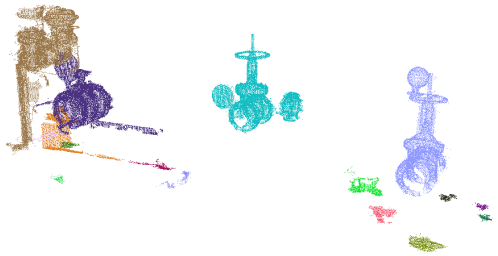


Figure 6. Segmentation result of the remaining candidate points.

$$\begin{aligned} S(Candidate) &:= S(Irregular) \\ &= S(All) - \bigcup_i S(Regular_i), \end{aligned} \tag{5}$$

where $Regular_i$ can be any connected component with repetitive patterns and large size. This definition also offers us the possibility of discovering candidates of new targets that are actually not in the database.

Figure 6 shows the segmentation and clustering result from the previous steps.

6. Cluster Filter

We observed that not all clusters are worth doing the detailed matching. In fact, most clusters in a scene will not be what we are interested in even at first glance. Therefore, we first pass the clusters through filters that can quickly rule out or set aside clusters with or without certain characteristic. The filters should be extremely fast while able to filter out quite a number of impossible candidates. Currently we have implemented a linearity filter.

The linearity of a cluster is evaluated by the absolute value of the correlation coefficient in the Least Squares Fitting on the 2D points of the three projections on the $x-y$, $y-z$ and $z-x$ planes. For example,

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 * \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{6}$$

and the linearity score is measured by Equation 7:

$$r = \max(|r_{xy}|, |r_{yz}|, |r_{zx}|). \tag{7}$$

The cluster would be considered to be linear if $r > 0.9$, meaning that at least one of its projections could be fitted by a line with high confidence. Note that planes and thin-pipes may fall in this linear category, but since both of them are supposed to be removed in the classification step, any remaining linear clusters are considered as lines, missed pipes, missed planes or noise. Experiments show that the linearity scores of all representative targets are below 0.8, with a substantial margin from the threshold of 0.9, meaning that our target objects won't be filtered out in this step.

In one of our experiments, only 1491 candidates are remaining, out of 1905 initial clusters from previous steps, after ruling out the most linear clusters.

7. Matching based on 3D SSIM Descriptor

7.1. Feature and Descriptor Generation

We follow the method in [19] to match between the candidate and the target point clouds. However, we use a simplified version i.e. the normal similarity since there's typically no intensity information in the targets.

The feature extraction process is performed such that 3D peaks of local maxima of principle curvature are detected in spatial-space. Given an interest point and its local region, there are two major steps to construct the descriptor.

Firstly, the 3D extension of the 2D self-similarity surface described in [20], is generated using the normal similarity across the local region. The normal similarity between two points x and y is defined by the angle between the normals, as Equ. 8 suggests (Assume $||\vec{n}(\cdot)|| = 1$).

$$\begin{aligned} s(x, y, f_n) &= [\pi - \cos^{-1}(f_n(x) \cdot f_n(y))]/\pi \\ &= [\pi - \cos^{-1}(\vec{n}(x) \cdot \vec{n}(y))]/\pi. \end{aligned} \tag{8}$$

Note that when the angle is 0, the function returns 1; whereas the angle is π, i.e. the normals are opposite to each other, the function returns 0.

Then, the self-similarity surface is quantized along log-spherical coordinates to form the 3D self-similarity descriptor in a rotation-invariant manner. This is achieved by using local reference system at each key point: the z-axis is the direction of the normal; the x-axis is the direction of the principal curvature; and the y-axis is the cross product of z and x directions. In this application we set 5 divisions in radius, longitude and latitude, respectively, and replace the values in each cell with the average similarity value of all points in the cell, resulting in a descriptor of $5 \times 5 \times 5 = 125$ dimensions. The dimension is greatly reduced without deduction of performance, which is another important difference from [19]. Finally, the descriptor is normalized by scaling the dimensions with the maximum value to be 1.

7.2. Matching

Our detection is based on matching, during which the descriptors for the candidate clusters generated online are compared against the descriptors for the targets generated offline and the transformation is estimated.

To establish the transformation, it's natural to first calculate the nearest-neighbor correspondences with any Nearest Neighbor Distance Ratio (NNDR) (introduced in [24]) between the candidate cluster and the target from the library. Then, unlike the normal RANSAC procedure, we propose a greedy algorithm based on the observation that (1) the top-ranked correspondences are more likely to be correct; (2) the objects we are detecting are rigid industrial parts with fixed standard size and shapes. In general, the transformation can be represented as Equ. 9:

$$p' = sRp + T. \tag{9}$$

where s is the scaling factor, R is the rotation matrix and T is the translation vector. For rigid-body transformation, $s = 1$, so we need to solve for the 12 unknowns in 3×3 matrix R and 3×1 vector T, which requires at most 4 correspondences of (p, p') here. (Note, however, that there are only 7 independent variables.)

We propose a greedy 4-round strategy to find the 4 correspondences based on the following insight: rigid-body transformation preserves the distance between the points.

Initially we have all nearest-neighbor correspondences with any NNDR value in the candidate set; In the beginning of round i, the correspondence with the minimum NNDR value, $c_i = (a_i, b_i)$, is added to the final correspondence set and removed from the candidate set. Then, for each of the correspondence $c' = (a', b')$ in the candidate set, calculate the Euclidean distance $dist(a', a_i)$ and $dist(b', b_i)$, and if the ratio of the (squared) distance is larger than some threshold (1.1), c' will be removed from the candidate set.

If the candidate set becomes empty within 4 rounds, we will discard the match as a failed rigid-body match; otherwise the transformation could be estimated over at least 4 distance-compatible correspondences. Specifically, a 3×3 affine transformation matrix and a 3D translation vector is solved from the equations formed by the correspondences.

To prevent matching from being sensitive to the first correspondence, multiple initial seeds are tried and only the transformation with the highest alignment score is selected. Finally, the rigid-body constraint is used again to refine the result, through the Gram-Schmidt Orthogonalization of the base vectors ($R = (\vec{u}_1, \vec{u}_2, \vec{u}_3)$):

$$\begin{cases} \vec{u}'_1 = \vec{u}_1 \\ \vec{u}'_2 = \vec{u}_2 - proj_{\vec{u}'_1}(\vec{u}_2) \\ \vec{u}'_3 = \vec{u}_3 - proj_{\vec{u}'_1}(\vec{u}_3) - proj_{\vec{u}'_2}(\vec{u}_3) \end{cases} \tag{10}$$

which are then normalized using:

$$\vec{e}'_i = \frac{\vec{u}'_i}{\|\vec{u}'_i\|} \qquad (i = 1, 2, 3). \tag{11}$$

A point p in cluster A is said to be aligned with cluster B if the nearest neighbor in cluster B to p under the transformation is within some threshold (5e-4 for industrial scene). The alignment score is thus defined as the percentage of aligned points. If the alignment score between a cluster and a target is larger than 0.6, the cluster is considered to be a detected instance of the target.

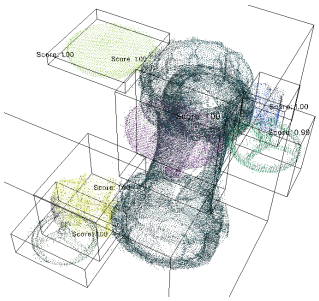


Figure 7. Parts detection in an assembly.

7.3. Iterative Detection

In case that there are multiple targets in a single cluster, we iteratively remove the aligned part through the cloud subtraction routine, and examine the remaining part of the cluster until it's too small to be matched.

Here is a demonstration of our algorithm against multi-target detection in a single cluster. We cut an assembly into several parts and use them as query. They were detected one-by-one through our alignment and iterative detection process (Fig. 7). Note that no segmentation is involved, and the descriptors are not identical at the same location of the part and the assembly.

8. Discussion

8.1. Choice of Descriptor for Classification

The principle for the selection of descriptor for classification can be depicted as: everything should be as simple as possible, but not simpler. This is because we aim at classifying basic geometric shapes. By simple we mean that the calculation is not expensive, and the dimension is small. One of the simplest features is Shape Index [33], however, it only considers the principal curvatures, which is not distinctive enough for multiple categories in our case. On the other hand, brief experiments on simple shape classification show that other descriptors such as Spin Image [25] and 3D Shape Context [30] are outperformed by FPFH [22] in both accuracy and efficiency. Note that, however, the performance of the descriptors could be quite different when they are used in, for example, complex shape matching.

8.2. Choice of Learning Method

As mentioned in Section 2, there are many learning methods available. We choose SVM as our classifier based on the following reasons: Firstly, we only aim at classifying simple geometric shapes, thus we need neither too complicated descriptors nor too complicated classifiers; secondly, we need the generalization ability from the classifier since we only have a limit number of training examples (especially for the parts) while there might be many more types of shapes; finally, although there can be as many as 5 categories, they are not equivalent because Part is the only category that will be used in detection, thus we propose a subtraction process over the 2-class classification results (Equ. 4, 5). We do want to consider the scalability when we try to apply SVM to more complicated multi-class objects in the future, however, since we don't care much now about the boundaries between categories other than the parts (e.g. big pipes can be classified as plane, pipes can be classified as thin pipes as long as the result is consistent), the algorithm is efficient enough to solve the seemingly multi-class, but intrinsically 2-class problem.

Classifier	#TrC	#TrP	#SV
Plane	14/23	94602	2069/2063
Pipe	14/9	91613	1496/1503
Edge	9/24	94838	1079/1121
Thin Pipe	8/22	83742	1020/1035

Table 2. Classifiers. TrC = Training Cluster, TrP = Training Point, SV = Support Vector. The ratio means Positive/Negative.

	Original	Plane	Pipe	Edge	Thinpipe
#Pts	25,135k	14,137k	8,767k	6,015k	5,534k
(%)	100.0%	56.2%	34.9%	23.9%	22.0%

Table 3. Remaining points after removal of each point category.

9. Experiment Results

9.1. SVM Classifier Training

Generally speaking we have five categories of the training clusters: Plane, Pipe, Edge, Thin-Pipe and Part. Since we are using two-class classification, when we are training one kind of classifier, all clusters labeled as this class will be regarded as the positive example, while the negative samples will be selected from the remaining categories. We summarize the statistics of training data in Table 2. The number of support vectors shows the complexity of the category, in order to distinguish it from the others. Table 3 shows the number of points in the residual point cloud after removing each of the four categories. Nearly half of the points are classified as plane, while after removing pipes there are one third of points, and finally only one fifth of the original points need to be considered in detection, which shows the effectiveness of the point classification step.

9.2. Ground Lidar

In this section, we show the testing results of our method on the industrial scene point cloud.

A result for the sub-scene is shown in Figure 8, where detected parts are highlighted with colors and bounding boxes. Figure 9 shows another result with respect to ground truth: the red color means false negative i.e. the object is miss detected or the point on the candidate cluster is misaligned; the blue color means false positive i.e. there is no target at

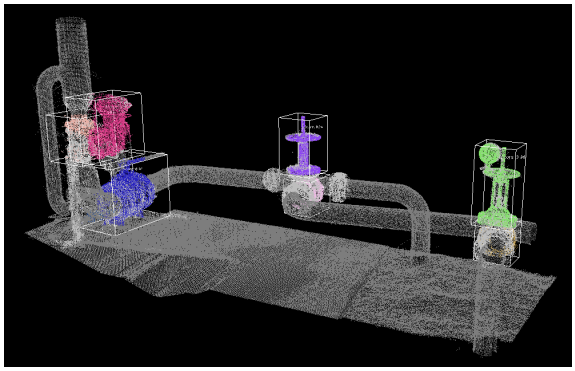


Figure 8. Detected parts, including handles, valves, junctions and flanges.

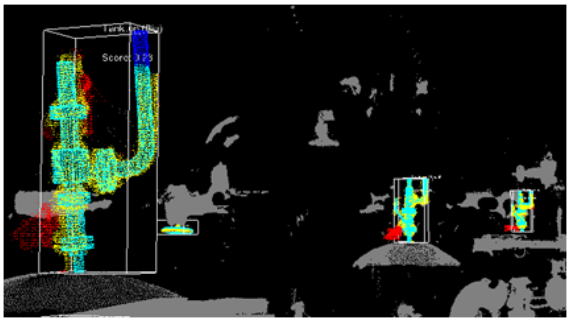


Figure 9. Detected parts on top of the tanks.

Category	#Sub-cat.	#Truth	#TP	#FP	#FN
Ball	1	22	11	4	11
Connection	2	3	0	0	3
Flange	4	32	20	3	12
Handle	6	10	3	1	7
Spotlight	1	6	1	0	5
Tanktop	2	4	3	3	1
T-Junction	5	25	7	0	18
Valve	12	25	17	24	8
All	33	127	62	35	65

Table 4. Statistics of detection. There are 8 big categories, 33 sub-categories and 127 instances (Ground-truth) of targets in the scene. Among them 62 are correctly identified (TP = True Positive), while 35 detections are wrong (FP = False Positive), and 65 instances are missed (FN = False Negative).

the position but the algorithm detected one, or the point on the target is misaligned. Yellow/cyan both mean true positive i.e. the aligned points are close enough.

Table 4 and Fig. 10 show the statistical results of the industrial scene. Balls, flanges, tanktops and valves are more successfully detected than the other objects.

9.3. Publicly Available Data

The experiment results show that our method works with the virtual point clouds almost as well as with the real point clouds. Since the virtual point clouds could be automatically generated from mesh models with the virtual scanner, we can expect the fully-automatic matching between the point clouds and the mesh models.

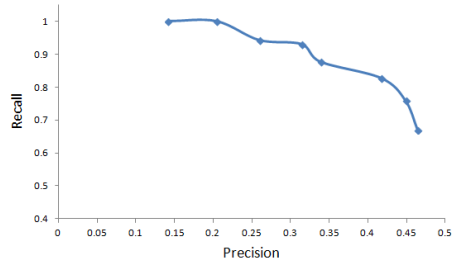
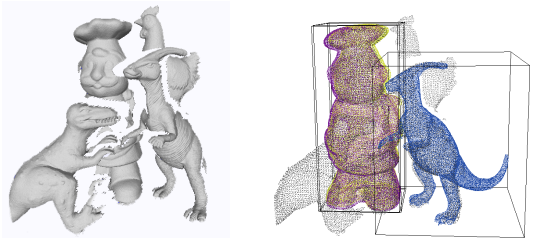


Figure 10. Precision-recall curve of the industrial part detection.



(a) (b)

Figure 11. Detection and alignment result from the cluttered scene. The chef is detected twice from two different sub-parts.

To compare our method with the other methods, we have also tested our algorithm on some of the following publicly available data. Figure 11 shows one detection result of the cluttered scene [32]. Note that only one face is present in the scene point cloud, and occlusions lead to discontinuity of some parts, which make the data quite challenging. Moreover, our point classification phase does not contribute to the result in this case.

10. Conclusion

In this paper, we present an object detection framework for 3D scene point cloud, using a combinational approach containing SVM-based point classification, segmentation, clustering, filtering, 3D self-similarity descriptor and rigid-body RANSAC. The SVM+FPFH method in pipe/plane/edge point classification gives a nice illustration of how descriptor could be combined with training methods. Applying two local descriptors (FPFH and 3D-SSIM) in different phases of processing shows that different descriptors could be superior under different circumstances. The proposed variant of RANSAC considering the rigid body constraint also shows how prior knowledge could be incorporated in the system. The experiment results show the effectiveness of our method, especially for large cluttered industrial scenes.

Acknowledgement

This research was supported by CiSoft project sponsored by Chevron. We appreciate the management of Chevron for the permission to present this work.

References

[1] A. Patterson, P. Mordohai and K. Daniilidis. Object Detection from Large-Scale 3D Datasets using Bottom-up and Top-down Descriptors. ECCV 2008. 2

[2] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing Objects in Range Data Using Regional Point Descriptors. ECCV 2004. 2

[3] A. Golovinskiy, V. G. Kim, T. Funkhouser. Shape-based Recognition of 3D Point Clouds in Urban Environments. ICCV 2009. 2

[4] Rapid Object Indexing Using Locality Sensitive Hashing and Joint 3D-Signature Space Estimation. PAMI 2006. 2

[5] H. Yokoyama, H. Date, S. Kanai and H. Takeda. Detection and Classification of Pole-like Objects from Mobile Laser Scanning Data of Urban Environments. ACDDE 2012. 2

[6] M. Lehtomäki, A. Jaakkola, J. Hyyppä, A. Kukko, H. Kaartinen. Detection of Vertical Pole-Like Objects in a Road Environment Using Vehicle-Based Laser Scanning Data. Remote Sensing 2010. 2

[7] B. Steder, G. Grisetti, M. V. Loock and W. Burgard. Robust On-line Model-based Object Detection from Range Images. IROS 2009. 2

[8] A. Nüchter, H. Surmann, J. Hertzberg. Automatic Classification of Objects in 3D Laser Range Scans. IAS 2004. 2

[9] H. Koppula, A. Anand, T. Joachims and A. Saxena. Semantic Labeling of 3D Point Clouds for Indoor Scenes. NIPS 2011. 2

[10] G. Vosselman, B. Gorte, G. Sithole, T. Rabbani. Recognising Structure in Laser Scanner Point Clouds. IAPRS 2004. 2

[11] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proc. CVPR 2001. 2

[12] Y. Freund and R. Schapire. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. Journal of Computer and System Sciences, 55(1):119-139, August 1997. 2

[13] M. Himmelsbach, A. Müller, T. Lüttel and H.-J. Wünsche LIDAR-based 3D Object Perception IWCTS 2008. 2

[14] X. Xiong, D. Huber. Using Context to Create Semantic 3D Models of Indoor Environments BMVC 2010. 2

[15] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, A. Ng. Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2005, IEEE Computer Society, pp 169-176. 2

[16] R. Shapovalov, A. Velizhev. Cutting-Plane Training of Non-associative Markov Network for 3D Point Cloud Segmentation. 3DIMPVT 2011. 2

[17] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection Computer Graphics Forum, vol. 26, no. 2, pp. 214-226, June 2007. 2

[18] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3D Point Cloud Based Object Maps for Household environments. Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge), 2008. 2

[19] J. Huang, and S. You. Point Cloud Matching based on 3D Self-Similarity International Workshop on Point Cloud Processing (Affiliated with CVPR 2012), Providence, Rhode Island, June 16, 2012. 2, 5

[20] E. Shechtman and M. Irani. Matching Local Self-Similarities across Images and Videos. In Proc. CVPR, 2007. 5

[21] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz. Persistent Point Feature Histograms for 3D Point Clouds. In Proceedings of the 10th International Conference on Intelligent Autonomous Systems, 2008. 3

[22] R. B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D Registration in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, May 12-17 2009. 1, 2, 3, 6

[23] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11), Shanghai, China, May 2011. 3

[24] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pp. 1615-1630, Oct. 2005. 5

[25] A. Johnson and M. Hebert. Object recognition by Matching Oriented Points. In Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA, pages 684.689, 1997. ICCV 1997. 2, 6

[26] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool. Hough Transform and 3D SURF for Robust Three Dimensional Classification. In: ECCV. 2010. 2

[27] J. Sun, M. Ovsjanikov, and L. Guibas. A Concise and Provably Informative Multi-scale Signature based on Heat Diffusion. In: SGP. 2009 2

[28] M. M. Bronstein and I. Kokkinos. Scale-Invariant Heat Kernel Signatures for Non-rigid Shape Recognition. In Proc. CVPR 2010. 2

[29] S. Ruiz-Correa, L. G. Shapiro, and M. Meliă. A New Signature-based Method for Efficient 3-D Object Recognition. In Proc. CVPR, 2001. 2

[30] M. Körtgen, G.-J. Park, M. Novotni, and R. Klein. 3D shape matching with 3D Shape Contexts. In The 7th Central European Seminar on Computer Graphics, April 2003. 2, 6

[31] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. (2011) [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm 3

[32] Ajmal Mian, M. Bennamoun and R. Owens. 3D Model-based Object Recognition and Segmentation in Cluttered Scenes. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), vol. 28(10), pp. 1584–1601, 2006. 7

[33] J. J. Koenderink. Solid Shape. MIT Press, 1990. 6

[34] A. Golovinskiy and T. Funkhouser. Min-cut based Segmentation of Point Clouds. in IEEE Workshop on Search in 3D and Video (S3DV) at ICCV, Sept. 2009. 4