Change Detection in Laser-Scanned Data of Industrial Sites

Jing Huang
University of Southern California
huang10@usc.edu

Suya You
University of Southern California
suya@usc.edu

Abstract

As laser scanners become widely used in 3D data acquisition of industrial sites, one challenging problem emerges: given two data of the same site scanned/modeled at different times, how can we tell the difference between the two? In this paper, we formulate this problem as the 3D change detection problem, and propose a novel method for detecting object-level changes. In general, we notice that the changes can be viewed as the inconsistency between the global alignment and the local alignment. Therefore, we propose a change detection framework that comprises global alignment, local object detection and a novel change detection method. Specifically, we propose a series of change evaluation functions for pairwise change inference, based on which we formulate the many-to-many object change correlation problem as the weighted bipartite matching problem which could be solved efficiently. Finally, we demonstrate the feasibility of our approach through experiments on both synthetic and real industrial datasets.

1. Introduction

Change detection plays vital roles in a variety of applications such as surveillance, tracking, data updating and smart environments. Most previous change detection works have been done on 2D data, i.e., images taken at different times or different frames in a video. However, since 2D data are projected data that do not explicitly reflect the 3D structures, recovering the 3D pose from 2D data is often needed, which faces many challenges such as illumination and occlusion problems. On the other hand, 3D data have the advantage that they are intrinsically correct in terms of the 3D pose, thus can avoid many problems caused by 2D distortions. Although 3D data become easier to achieve nowadays, most changes detection in 3D data were manually done by professionals, which is rather time consuming, as [5] pointed out. Therefore, it's necessary to develop a change detection approach specialized in 3D data. Intuitively, the change detection needs to be based on the alignment of the two data. Once the data are aligned, there could be multiple ways to detect the changes. A naive way is using octree to compare the spatial difference [5]. For specific applications, different methods could be applied, e.g. 3D change detection for buildings [9]. In this work, however, we focus on the 3D change detection of objects in industrial sites. We investigate how change detection could be enhanced beyond the traditional align-and-compare approaches.

Specifically, we are interested in changes of certain types of objects, since in many applications only objects of interest are considered important. Also, in many built facilities, changes on certain types of objects are observed more often than the other changes. For example, in an industrial scene, the pose of a handwheel is changed frequently and the difference is vital since it could determine whether a valve is open or closed. Therefore, we focus on the object-level changes, whereas the changes of other large-scale substance such as pipes and planes are not highlighted in our system. Figure 1 illustrates the typical causes of the changes, including appearance/disappearance, pose change (translation and/or rotation) and replacement. In this paper, these causes are naturally incorporated in the design of our change detection algorithm.

Note that 3D changes not only occur between point clouds scanned at different times, but exist between point clouds and mesh models as well. The mesh model could either be the original design of the site, or manually modeled data from real scans. Since we can apply a virtual scanner to convert the mesh model to the point cloud format, our point cloud based approach could be applied in both scenarios.

We summarize our major contribution as follows:

(1) We propose an object-level change detection algorithm based on the estimation of the degree of change in the pairwise scenario and the non-trivial change selection based on weighted bipartite matching.

(2) We propose and integrate the whole change detection system consisting of three essential modules: alignment, object detection and change detection.

(3) To the best of our knowledge, this work is the first one to address the change detection problem on 3D data for complex industrial scenes at object level.

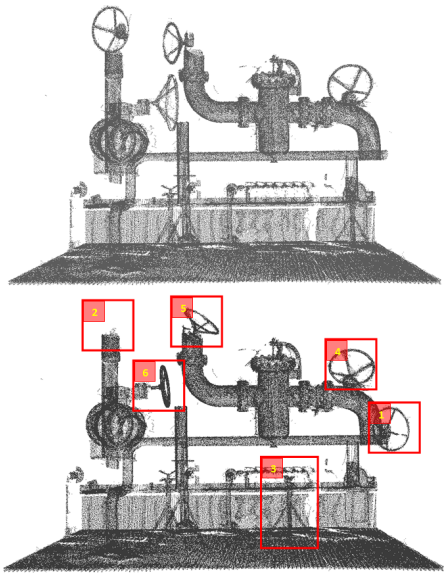


Figure 1. An illustration of possible causes of changes between old data (top) and new data (bottom). (1) New - The new object appears; (2) Missing - The old object is missing; (3)-(5) Pose - The pose of the object is changed due to translation and rotation; (6) Replacement - The old object is replaced by the new object.

2. Related Work

Change detection has been one of the basic topics in computer vision. A systematic survey of 2D image change detection algorithms can be found in [12], which focuses on changes between general raw images instead of application-specific change detection.

Mas [10] tests six change detection procedures on land cover images and concludes that classification-based methods are less sensitive and more robust. We make an analogy to this idea in our method: given the categories of objects representing a small region of point clouds, we can obtain results more robustly and with more semantic meanings. Another object-based approach [14] classifies groups of pixels corresponding to existing objects in the GIS database. The term object here is used to denote geo-objects instead of the industrial objects in this paper.

There are some common issues across both 2D and 3D change detection, such as alignment, classification, detection and occlusion handling. However, the differences are also obvious. For example, one of the key challenges in 2D change detection is the illumination factor, while 3D change detection does not have such issue. There have been benchmarks such as [6] for detecting changes among frames in videos. We can extend some ideas from the 2D scenario. For example, background subtraction is the key idea of many image/video change detection approaches (e.g. [1]). In this paper, we treat the objects with relatively small scales in the scene as foreground, and the backbone objects such as planes and pipes as the background. Through this analogy, we apply the object detection method [8] that contains a backbone object subtraction step.

Case	Explanation
Existence	The object appears or disappears.
Pose	The object is translated and/or rotated.
Replacement	The object is replaced by another object.

Table 1. Causes of object-level changes.

On the other hand, there have not been many existing works dealing with the change detection in 3D data. One of the earliest work in 3D data was proposed by Girardeau-Montaut *et al*. [5]. They compare several octree-based comparing strategies including average distance, best fitting plane orientation and Hausdorff distance. The overlapping score used in this paper has the similar effects to Hausdorff distance. The deformation measurement system proposed by Monserrat and Crosetto [11] also combines the global registration and local matching, however, in their approach manual selection of the objects is required, and the problem of multiple correspondences among the objects is not tackled. Recently, Xiao *et al*. [15] presents a change detection system for urban point clouds. Their method is based on the consistency of the occupancies of scan rays. Our method, however, does not need knowledge of the scan rays and can thus be applied to any point clouds.

3. Causes of Change

Given that most industrial parts are rigid, we make an assumption that each object is a rigid body. If part of some object could be transformed (e.g. the handwheel on a valve), it would be considered as a separate part.

Table 1 summarizes the possible causes of change of rigid objects. For any single object, there can be two types of changes: existence change and pose change. There are two possible existence changes, *appearing*, i.e., the object does not present in the first data but appears in the second data, and *disappearing*, i.e., the object presents in the first data but disappears in the second data. The pose change can be caused by translation, rotation or a combination of translation and rotation. For example, a handwheel could be rotated to control the flow in a pipe. Finally, for the inter-object scenario, i.e., the two objects are not the same object, there could be a replacement change. Our goal is thus to detect changes caused by these reasons.

4. Overview

The whole framework consists of three stages: global data alignment, object detection and change detection. Figure 2 shows the pipeline with the step-by-step results for the proposed system. The input of the system contains reference data and target data, both of which can be either 3D point clouds or mesh models. If a mesh model is given,

it could be automatically converted to a point cloud by a virtual scanner. In the first stage, the Sparse Iterative Closest Point (SICP) method proposed in [3] is used to compute a global alignment of the two point clouds. The target data can thus be transformed to the same pose as the reference data. Then, in the object detection stage, we apply the method proposed in [8] to detect the objects of interest and as a result, the detected objects are locally aligned with the library objects. In other words, our change detection system is based on the inconsistency between the global alignment and the local alignment. Finally, pairwise changes are estimated based on several change estimation functions, and we explore two approaches to solve the change selection problem, including the Greedy Nearest Neighbor (GNN) and the Weighted Bipartite Matching (WBM).

5. Point Cloud Alignment

Since we are dealing with industrial data, where the global structures have inherent rigid properties, it's suitable to apply the rigid registration techniques. The major difficulty for traditional ICP [2] is that it's very sensitive to outliers. However, the cases for missing data are often observed in 3D scans due to various reasons.

Sparse Iterative Closest Point (SICP) [3] is a recently proposed extension of the traditional ICP. Specifically, they formulate the general sparse ℓ_p optimization problem and empirically select $p = 0.4$ as the balanced result between performance and robustness. There are several advantages of SICP. First, it's heuristic-free, compared to previous variations of ICP. Second, it has only one free parameter, which is easy to tune for specific applications. Therefore, we apply SICP for global alignment of data in this work.

6. 3D Object Detection

Given the aligned point clouds, we follow the idea of the 3D object detection method proposed by [8] to detect objects from each of them, respectively.

Detecting 3D objects from point cloud itself is a challenging fundamental problem. For industrial datasets, a majority of points belong to planes and pipes. Therefore, it's necessary to first filter such points that have little probability of being in an object.

For each point we compute a normalized Fast Point Feature Histograms (FPFH) descriptor [13]. Five categories are defined, i.e. four backbone shapes pipe, plane, edge and thin-pipe, as well as the remaining points denoted as the part category since they are most likely to be part of an industrial part. We train four SVM-classifiers using LIBSVM [4] over these categories, including pipe vs non-pipe, plane vs non-plane, edge vs non-edge and thin-pipe vs non-thin-pipe. Finally, each point is assigned with four labels corresponding to the four classification results.

The points in each categories are clustered using the region growing algorithm based on Euclidean distance, respectively. Large connected components of the primitive shapes are then removed from the original point cloud while the smaller clusters are kept. After that, all the remaining points are iteratively clustered using adaptive segmentation [8] to obtain a list of candidate clusters.

For each of the point clouds from the templates in the database and the candidate clusters, we perform feature extraction using Maxima of Principal Curvature (MoPC) method proposed in [7] to obtain a subset of points as the key-points. For each key-point, we compute the 3D Self-Similarity descriptor [7] with curvature similarity.

To determine which candidate clusters belong to which object category in the database, we apply the greedy matching scheme based on rigid-body constraints and overlapping scores described in [8].

7. Change Detection

From the previous stages we can obtain two groups of detected objects along with the transformation with regard to the library objects. For a naive change detection system, all the detected object locations are reported as auxiliary information and the user is required to manually decide the actual changes. However, we propose a method that could automatically infer the changes among the objects.

In order to figure out the changes between two groups of objects, we first consider the simplified case where each group contains exactly one object.

7.1. Pairwise Change Estimation

Given any two detected objects X and Y in different times, we could make several change evaluations between them. The most fundamental evaluation is the category to which each of them belongs. It's natural to define that, if they belongs to the same category, the category change is 0, otherwise the change is 1 (Eq. 1):

$$C_c(X,Y) = \begin{cases} 0, Cat(X) = Cat(Y), \\ 1, Cat(X) \neq Cat(Y) \end{cases} \tag{1}$$

Another significant change between two objects one would observe immediately is the difference in locations. We can thus define the location change as the distance between the centers of the objects (Eq. 2):

$$C_l(X,Y) = ||\overline{X} - \overline{Y}||. \tag{2}$$

Even if the locations of the two objects are close, there could be rotational change between them, e.g., a handwheel could be rotated even if it stays in the same location. Let T be the template of X and Y in the database, from the object

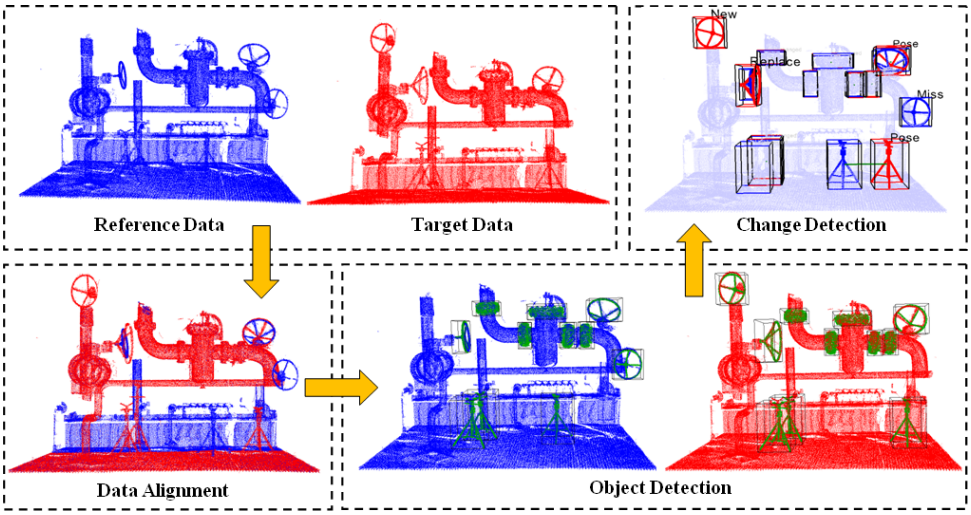


Figure 2. System pipeline for object change detection. The input contains the reference data (left) and the target data (right). Results from major stages including global alignment, object detection and change detection are shown. In the illustration of object detection, detected objects are highlighted with green color and bounding boxes. The visualization of change detection results is explained in Section 8.

detection module we know:

$$\begin{cases} X = R_1 T + b_1 \\ Y = R_2 T + b_2 \end{cases} \tag{3}$$

where R_1 and R_2 are rotational matrices and b_1 and b_2 are translation vectors. By eliminating T we have:

$$Y = R_2 T + b_2 = R_2[R_1^{-1}(X - b_1)] = R_2 R_1^{-1} X + b_3. \tag{4}$$

Therefore, we can directly compute the rotational matrix between X and Y as

$$R := R_2 R_1^{-1}. \tag{5}$$

The remaining problem is how we can measure the degree of rotation. We use the Euler angles (yaw, pitch, roll) derived from the rotational matrix (Eq. 6):

$$\begin{cases} \alpha = atan2(R_{21}, R_{11}) \\ \beta = atan2(-R_{31}, (R_{32}^2 + R_{33}^2)^{\frac{1}{2}}) \\ \gamma = atan2(R_{32}, R_{33}) \end{cases} \tag{6}$$

Then, the rotation change can be represented by the norm of (α, β, γ) (Eq. 7):

$$C_r(X, Y) = \sqrt{\alpha^2 + \beta^2 + \gamma^2}. \tag{7}$$

Finally, if the objects are close enough, we can measure their degree of overlapping. Specifically, we compute the overlapping score between the two point clouds as is proposed by [8]. The overlapping score with threshold $\theta = 0.005$ is defined as the proportion of the points in point cloud A where there exists a point in point cloud B in its θ-neighborhood (Eq. 8):

$$\Omega(A, B) = \frac{|\{x \in A; \exists y \in B(||x - y|| < \theta\}|}{|A|} \tag{8}$$

The overlap change is then defined by Eq. 9:

$$C_\Omega(X, Y) = \sqrt{(1 - \Omega(X, Y))^2 + (1 - \Omega(Y, X))^2}. \tag{9}$$

Finally, if we presume that the two objects have a relationship, we can decide the reasons of changes between them using the similarity scores proposed above. Figure 3 illustrates the decision flow of pairwise change estimation. If the categories of the two objects are different, then there's a replacement change; otherwise if the overlap change is small than θ_Ω (In our experiments $\theta_\Omega \approx 0.7$ corresponding to the case where overlapping rate is 50%, i.e., $\Omega(X, Y) = \Omega(Y, X) = 0.5$), we claim that there's no change between the objects. Otherwise the location change and the rotation change are checked by comparing them to thresholds $\theta_l = 0.1$ and $\theta_r = \sqrt{0.2^2 + 0.2^2 + 0.2^2} \approx 0.346$.

7.2. Change Selection

Given the pairwise change estimation result, we now consider the problem of selecting the most convincing changes. A degenerated case is the one-to-many problem: given an object X and several candidate objects Y_j, we need to figure out which object among Y_j is the most probable one corresponding to X. A straightforward method is using the greedy algorithm to select the best candidate for each object based on the distance. If the distance of the nearest neighbor is too large (e.g. $> 1m$), then the candidate is considered to have no matched object. We refer to this method as the Greedy Nearest Neighbor (GNN) method.

However, the minima might not be achieved simultaneously, which requires that we make a balance among the measurements. Also, instead of only the location proximity, qualitatively we expect that the object Y_j with the same category, the minimum location change, the minimum ro-

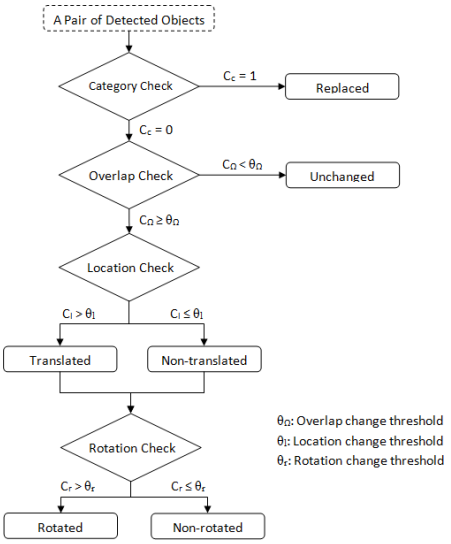


Figure 3. Decision flow of pairwise change estimation.

tation change as well as the minimum overlap change with respect to X be the most probable candidate, i.e., the one with the minimum changes.

In order to answer this question, we need a comparable measurement of similarity between X and each of Y_j. Also, the best candidate might not exist, as there could be missing/disappearing or new/appearing objects. For example, if the location change is too large, then the objects are more likely to be different ones, even if their category and pose are similar. In the GNN method a threshold is put on the distance, however, we propose a smoother solution by using the Gaussian function (Eq. 10):

$$g(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}. \tag{10}$$

Formally, we define the change estimation between the two objects X and Y as Eq. 11, where C_c, C_l, C_r and C_Ω are the estimation of category change, location change, rotation change and overlap change, respectively:

$$\begin{aligned} C =& (g(C_l;0,\sigma_{l0}) \cdot (1 + g(C_r;0,\sigma_r)) + g(C_\Omega;0,\sigma_\Omega)) \\ & \cdot (1 - C_c) + g(C_l;0,\sigma_{l1}) \cdot C_c. \end{aligned} \tag{11}$$

Here's the explanation of Eq. 11. The standard deviations σ_{l0}, σ_{l1}, σ_r and σ_Ω can be viewed as the *soft* threshold for deciding correlation and non-correlation. The category change estimation C_c acts as the indicative function. If the objects are not in the same category, then the rotation change and the overlap change are not well defined, thus only the location change takes effect. If the objects are in the same category, the rotation change would take effect only when the location change is small, while the overlap change would act independently since it inherently requires a close distance. Moreover, the distance threshold for the same category σ_{l0} is larger than that for different categories σ_{l1}, which gives a larger range for search of objects of the same category. Generally speaking, objects of the same category are preferred to be correlated.

To deal with the case where there's no suitable candidate, we set a cutoff threshold C_0 for the change estimation. The value is set to be the change estimation value with respect to two *dummy* objects Y_{D1}, Y_{D2} at a location that is far enough. We empirically define Y_{D1} to be an object with the different category at a distance of 0.5, and Y_{D2} to be an object with the same category at a distance of 1. C_0 is thus defined as Eq. 12:

$$C_0 = max(g(0.5;0,\sigma_{l1}), g(1;0,\sigma_{l0})). \tag{12}$$

Instead of the single value for the pairwise change evaluation, we now have a matrix of the change evaluation values representing all the pairwise relationship between the two sets of the objects. Next, we need to figure out which relations should be kept. We formulate the problem as the classical Weighted Bipartite Matching problem.

7.2.1 Weighted Bipartite Matching (WBM)

Given a graph $G = (V, E)$, G is bipartite if there exists partition $V = V_x \cup V_y$ s.t. $V_x \cap V_y = \emptyset$ and $E \subseteq V_x \times V_y$. $M \subseteq E$ is a matching on G if $\forall v \in V$ there's at most one edge in M that connects v. We can define a weight function $w : E \to \mathbb{R}$ on the edges. The weight of M is given by

$$w(M) = \sum_{e \in M} w(e). \tag{13}$$

The goal of maximum weight bipartite matching is thus to find $\arg\max_M(w(M))$, which can be solved efficiently in $O(V^2 \log V + VE)$ with Hungarian algorithm using Fibonacci heap.

7.2.2 Conversion to WBM Problem

We can now convert our problem to the maximum weight bipartite matching problem. Each detected object $X_i (1 \leq i \leq n)$ in the first data and $Y_j (1 \leq j \leq m)$ in the second data can be represented by a vertex, and each weighted edge represents the change evaluation value between the objects:

$$w(i,j) = C(X_i, Y_j), 1 \leq i \leq n \wedge 1 \leq j \leq m. \tag{14}$$

Note that when the change is more significant, by the Gaussian-based definition of function C (Eq. 11) the weight is smaller, so to maximize the weights is equivalent to minimize the changes.

Since n and m might not be equal, we need to add dummy vertices as well as dummy edges to balance the graph. The traditional method is to assign zero weight to

the dummy edges, however, in our case not all objects need to be matched to another object, meaning that certain objects could be preferably matched to a dummy object even if there are unmatched real objects. In the extreme case, all the objects are matched to dummy objects. Therefore, we add exactly m dummy vertices to the X side and n dummy vertices to the Y side, and assign the weights of the dummy edges to be the cutoff threshold:

$$w(i,j) = C_0, n < i \leq n+m \vee m < j \leq n+m. \quad (15)$$

Finally, for each matched pair (X, Y), if X is a dummy vertex while Y is not, then Y is considered as the *new* object; if Y is a dummy vertex while X is not, then X is considered as the *missing* object. If both X and Y are not dummy, we can apply the algorithm described in Fig. 3 to decide the change between them.

8. Experimental Results

We demonstrate the feasibility of our approach on both synthetic dataset and real dataset.

8.1. Synthetic Data

We create a data generator that can automatically generate pairs of changed scenes. Each scene is a combination of randomly placed objects without backbone objects such as pipes and planes.

We automatically generate two synthetic datasets, Sparse and Dense, each containing 10 pairs of randomly generated scenes. For each pair of data, $5-15$ objects are unchanged, $10-20$ objects have pose changes (including translation and/or rotation), $0-5$ objects are replaced, $0-5$ objects are missing and $0-5$ objects are new. For those with pose changes, we limit the position difference to be no more than $0.6m$. All objects are restricted in the $l \times l \times l$ area to enable contacts among the objects. For the Sparse dataset $l = 4m$ and for the Dense dataset $l = 3m$. The objects in the Dense dataset have more chance of contacts, so Dense is more challenging than the Sparse dataset. The process of data generation ensures that the global alignment and object detection are nearly perfect, thus the actual performance of the change detection algorithm is evaluated. Each change is represented as a tuple (16):

$$\tau = (T, C_r, C_t, P_r, P_t), \quad (16)$$

where T is the type of change, C_r and C_t are the categories of the correlated objects in the reference data and the target data, and P_r and P_t represent the locations of the objects. The judgment of whether a detected change is true or not is depending on the proximity of the tuples (17):

$$\begin{aligned}\tau_1 = \tau_2 \Leftrightarrow & (T_1 = T_2) \wedge (C_{r1} = C_{r2}) \wedge (C_{t1} = C_{t2}) \\ & \wedge (||P_{r1} - P_{r2}|| < \epsilon) \wedge (||P_{t1} - P_{t2}|| < \epsilon),\end{aligned} \quad (17)$$

Method	Precision	Recall
Perfect Object Detection + GNN	68.4	70.3
Perfect Object Detection + WBM	100.0	100.0
Automatic Object Detection + GNN	42.6	54.1
Automatic Object Detection + WBM	61.4	73.0

Table 3. Statistics of the change detection results on real dataset.

where $\epsilon = 0.1$ is the tolerance of localization error.

Table 2 shows the statistics of the two change detection methods on the synthetic datasets. WBM outperforms GNN on both datasets. Specifically, while WBM and GNN have similar recall, WBM has significantly higher precision.

8.2. Real Data

We test the methods on both cloud-to-cloud and mesh-to-cloud data of real industrial sites, containing totally 37 changes in ground truth. The real dataset is more challenging than the synthetic dataset in that the global alignment and object detection are not perfect. For example, if an object failed to be detected in the object detection stage, the change on it would not be detected in the change detection stage as well. Therefore, we compare 4 combinations of methods, including the two change detection algorithms WBM and GNN combined with either automatic/manual object detection. The statistics are summarized in Table 3, where we list the precision and recall on the real dataset for either perfect (manual) object detection or automatic object detection combined with WBM or GNN.

Obviously perfect object detection would given better results than the automatic object detection which could miss some objects. Still, it's surprising that given the perfect object detection results WBM would achieve a perfect precision and recall in our test dataset. On the other hand, even with the perfect object detection results, GNN could only achieve around 70% in both precision and recall.

Figure 4 illustrates the example of cloud-to-cloud change detection and Figure 5 shows the example of mesh-to-cloud change detection. In final results, the reference data is shown as the baseline in light blue, the objects in the reference data that are found to have a change are highlighted with deep blue, while the objects in the new data that are found to have a change are highlighted with red. Each detected object has a bounding box, and the ones that are not highlighted with deep blue or red colors are the ones found to be unchanged. Four labels are assigned to highlight the changes, including New, Miss, Pose and Replace. For pose change and replacement change, there's a green line to represent the correspondence between the objects.

From the figures and the table we can see that, The false alarms are mainly due to the imperfectness of the first two steps. For example, the false alarms of the flanges in the first data case (Fig. 4) are due to false alarms and mis-detection

		WBM					GNN				
Data	Truth	TP	FP	FN	Precision	Recall	TP	FP	FN	Precision	Recall
Sparse	212	201	4	11	98.0	94.8	199	18	13	91.7	93.9
Dense	212	183	23	29	88.8	86.3	182	46	30	79.8	85.8

Table 2. Statistics of the change detection results aggregated from 10 pairs of randomly generated scenes in each synthetic dataset.

in the object detection module. The false alarms of the pose change in the second data case (Fig. 5) are due to the small displacement of the global alignment (Fig. 5 (c)). It's also important to note that the unchanged objects are not counted in the statistics as the ground truth, while our method successfully identifies many of the such cases as in Fig. 4.

It's worth pointing out that the change detection module is extremely fast. For an input of two point clouds each containing $150k$ points, the running time of the whole system containing global alignment and object detection modules is about one hour, while the running time of the change detection module using either method is less than 10 seconds.

9. Conclusion and Future Work

In this paper, we define and address the problem of object-level change detection in point clouds. We present a new approach, in which the global alignment, local object detection and object change inference are combined to achieve robust detection of changes. Specially, we propose the change evaluation functions for pairwise change estimation, and use the weighted bipartite matching to solve the many-to-many object correlation problem. As far as we know, this is the first work addressing the 3D change detection problem in complex industrial scenes at object level. In the future, the scheme could be extend to detect changes of large-scale non-part objects such as pipes and planes.

Here are some insights of possible improvements:

(1) Enhancement of global alignment. The accuracy of global alignment could largely affect the accuracy of the whole system, therefore, it's important to ensure a good global alignment. Effectively the object detection module does not rely on the global alignment, moreover, it can provide necessary information that can help the initial setup of the point clouds, to which the global alignment is sensitive. The first idea is to align the backbone objects separately. Note that ground planes, which contain many points, typically provide only one orientation cue, while the pipes, acting as the skeleton of an industrial site, might be easier to be aligned. This is more significant considering the case where ground is not present in exactly one of the data. The second idea is to use the detected as hyper-key-points, while the categories could be viewed as a one-dimensional descriptor. The global alignment now turns into a hyper-matching problem among the objects.

(2) Enhancement of object detection. On the other way around, object detection could benefit from the global alignment as well, especially when we are not able to perform a perfect detection on any data, due to noises and occlusions. For each detected object in each of the data, we can try to search more thoroughly in the nearby area in the other data, after the two data have been globally aligned.

We plan to explore these possible improvements in our future work.

Acknowledgement

This work is supported by Chevron U.S.A. Inc. under the joint project Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California.

References

[1] O. Barnich and M. Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *Image Processing, IEEE Trans. on*, 20(6):1709–1724, 2011. 2

[2] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992. 3

[3] S. Bouaziz, A. Tagliasacchi, and M. Pauly. Sparse iterative closest point. In *Computer Graphics Forum*, volume 32, pages 113–123. Wiley Online Library, 2013. 3

[4] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. 3

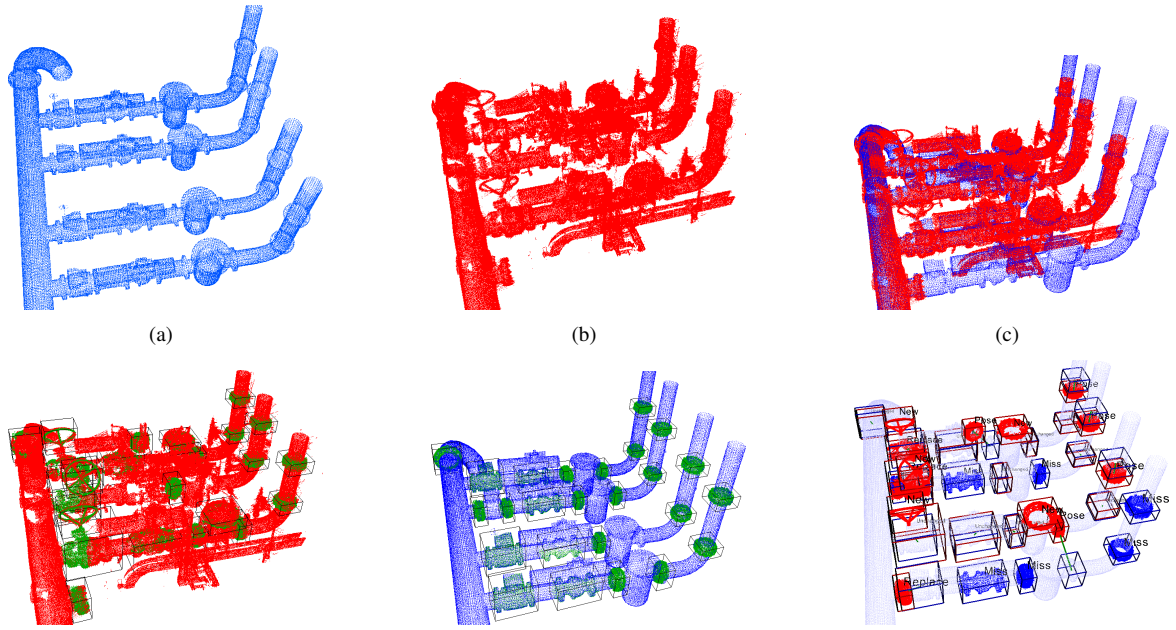
[5] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault. Change detection on points cloud data acquired with a ground laser scanner. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(part 3):W19, 2005. 1, 2

[6] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection. net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 1–8. IEEE, 2012. 2

[7] J. Huang and S. You. Point cloud matching based on 3d self-similarity. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 41–48. IEEE, 2012. 3

[8] J. Huang and S. You. Segmentation and matching: Towards a robust object detection system. In *Winter Conference on Applications of Computer Vision (WACV)*, 2014. 2, 3, 4

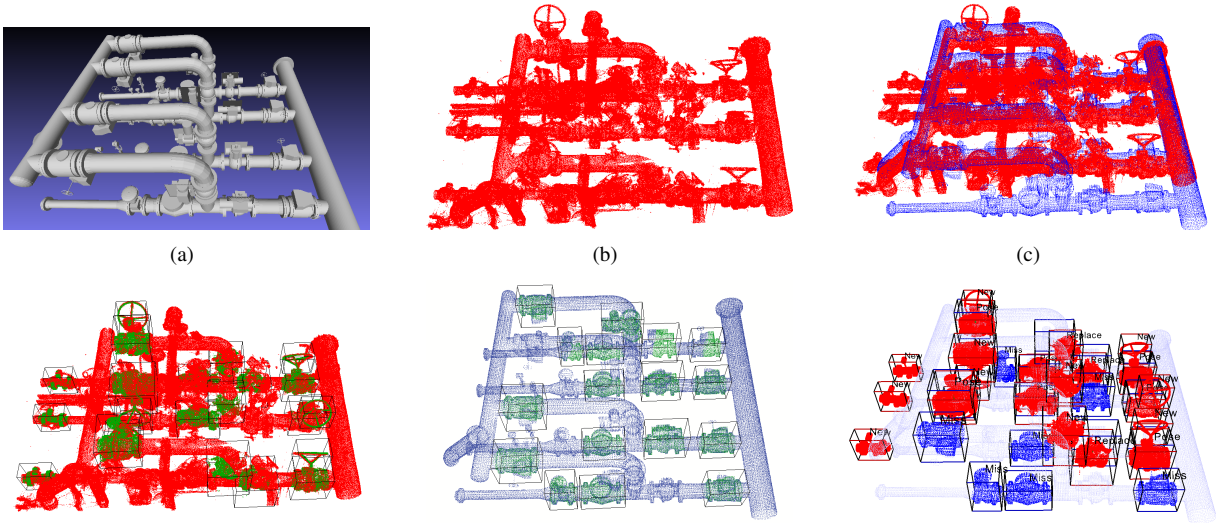
[9] Z. Kang and Z. Lu. The change detection of building models using epochs of terrestrial point clouds. In *Multi-Platform/Multi-Sensor Remote Sensing and Mapping (M2RSM), Intl. Workshop on*, pages 1–6. IEEE, 2011. 1



(a) (b) (c)

(d) (e) (f)

Figure 4. (a) (b) Original point clouds. (c) Global alignment result. (d) (e) Detection results. (f) Change detection results. The visualization of the change detection results is explained in Section 8.



(a) (b) (c)

(d) (e) (f)

Figure 5. (a) Original mesh model. (b) Original point cloud. (c) Global alignment result. (d) (e) Detection results. (f) Change detection results. The visualization of the change detection results is explained in Section 8.

[10] J.-F. Mas. Monitoring land-cover changes: a comparison of change detection techniques. *International journal of remote sensing*, 20(1):139–152, 1999. 2

[11] O. Monserrat and M. Crosetto. Deformation measurement using terrestrial laser scanning data and least squares 3d surface matching. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):142–154, 2008. 2

[12] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *Image Processing, IEEE Transactions on*, 14(3):294–307, 2005. 2

[13] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. 3

[14] V. Walter. Object-based classification of remote sensing data for change detection. *ISPRS Journal of photogrammetry and remote sensing*, 58(3):225–238, 2004. 2

[15] W. Xiao, B. Vallet, and N. Paparoditis. Change detection in 3d point clouds acquired by a mobile mapping system. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1(2):331–336, 2013. 2