Volume 0 (1981), Number 0 pp. 1–13 COMPUTER GRAPHICS *forum*

Detection of geometric temporal changes in point clouds

Gianpaolo Palma[1], Paolo Cignoni[1], Tamy Boubekeur[2], Roberto Scopigno[1]

[1]Visual Computing Lab - ISTI - CNR, Italy
[2]Telecom ParisTech - CNRS LTCI - Institut Mines Telecom, France

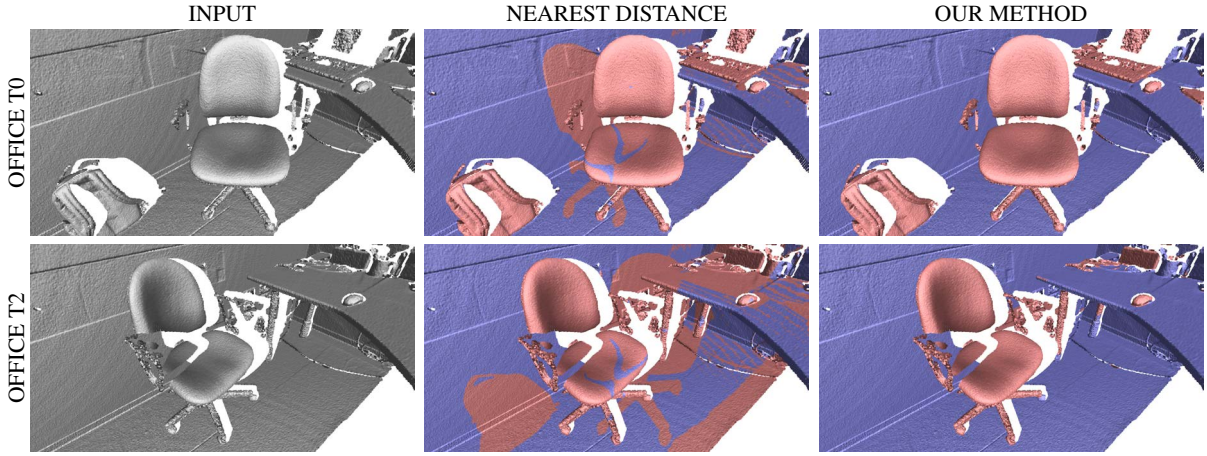


Figure 1: *Comparison between the proposed method and a solution based on thresholding (4mm) the distance from the nearest point in the other capture (NEAREST DISTANCE). The figure shows the binary segmentation (red = change, blue = no-change). Our method produces a better output in the temporal interception regions (on the chair) and less noise due to the different sampling directions of the scanner (over the desk and on the floor under the desk).*

Abstract

Detecting geometric changes between two 3D captures of the same location performed at different moments is a critical operation for all systems requiring a precise segmentation between change and no-change regions. Such application scenarios include 3D surface reconstruction, environment monitoring, natural events management and forensic science. Unfortunately, typical 3D scanning setups cannot provide any one-to-one mapping between measured samples in static regions: in particular, both extrinsic and intrinsic sensor parameters may vary over time while sensor noise and outliers additionally corrupt the data. In this paper, we adopt a multi-scale approach to robustly tackle these issues. Starting from two point clouds, we first remove outliers using a probabilistic operator. Then, we detect the actual change using the implicit surface defined by the point clouds under a Growing Least Square reconstruction that, compared to the classical proximity measure, offers a more robust change/no-change characterization near the temporal intersection of the scans and in the areas exhibiting different sampling density and direction. The resulting classification is enhanced with a spatial reasoning step to solve critical geometric configurations that are common in man-made environments. We validate our approach on a synthetic test case and on a collection of real datasets acquired using commodity hardware. Finally, we show how 3D reconstruction benefits from the resulting precise change/no-change segmentation.

Categories and Subject Descriptors (according to ACM CCS): [Computer Graphics]: Shape modeling—Point-based models

1. Introduction

Capturing time varying 3D point clouds has become more and more common in the last years thanks to the introduction of fast 3D acquisition devices. The analysis of the output data produced by these devices is a challenging task, especially in the field of the automatic and robust detection of geometric changes. In this context, the ultimate goal is often to obtain an accurate segmentation between the change and the no-change parts of an evolving scene acquired at different moments. This segmentation can be used in several applications, such as 3D reconstruction [YSL*14], urban growth analysis [TBP13], environment monitoring [MRMT05] or natural events management [LFM*13], which are characterized by different and opposite requirements. On the one hand, the data consolidation of the input 3D geometry to produce a clean mesh, extracting only the static and stable part of the scene removing the transient portions. On the other hand, monitoring applications need to detect and analyze the dynamic part to model the type of change. Over the last few years, several solutions based on probabilistic frameworks have been proposed to solve the change detection problem in image datasets [PM07] or in mixed acquisitions (3D model and photos at different times) [TBP11]. Other methods are based on the acquisition of a dense and constant stream of 3D scans [YSL*14], using RGB-Depth sensors and making assumptions on the continuity of the changes in the scene. Less attention has been devoted to the temporal comparison of simple 3D scans without temporal coherence where no assumption can be done on the delay between the captures. In this case, to obtain an as accurate as possible segmentation, the required features of a change detection algorithm are: (i) to manage inputs without assumptions on connectivity and temporal coherence; (ii) to be robust against noise, temporal intersections and changes in sampling density and direction; (iii) to account for challenging geometrical configurations, such as occlusions or the movement of an object with coherent overlapping in time.

This paper proposes a new method to automatically detect geometric temporal changes between point clouds. Our contribution is twofold: first we introduce a new point classification based on multi-scale analysis; second we propose a spatial reasoning method for detecting and fixing critical geometric configurations.

The Growing Least Square framework (GLS) [MGB*12] presents several good features for our problem. It allows a continuous scale-space analysis of a point cloud without any parameterization, connectivity or uniform sampling condition. This framework defines a robust geometric descriptor based on an Euclidean neighborhood that does not only provide curvature measurements, like other scale-space approach based on Difference of Gaussians [ZBVH09], making its interpretation more intuitive and its computation faster. In particular it permits the comparison of pairs of scale-space locations using a dimension-less and scale-invariant dissimilarity function that simplifies the design of an automatic method without tuning any scale-dependent or dataset-dependent parameters. As we build our temporal multi-scale analysis on top of GLS, our approach robustly interprets the geometric elements in the context of their surrounding, characterizing accurately each point. This aspect is crucial for the proper classification of points laying near the temporal intersection of two surfaces belonging to different scans. It also makes the algorithm robust against difference of densities and sampling directions between the point clouds (see Figure 1).

Our second contribution is a spatial reasoning procedure that, using the information from the surrounding regions, allows to detect and correct wrong change/no-change classifications in critical geometric arrangements that are very common in man-made environments. In particular, based on the analysis of the consistency and proximity of the geometric data over time, we address two critical situations: the classification of an occluded point; the wrong segmentation of a surface with zero Gaussian curvature moved along the direction with zero curvature, creating overlay in the time. Finally, we present a method based on a controlled subsampling of the input point clouds to speed-up the algorithm.

2. Background

Our work builds upon Algebraic Point Set Surfaces [GG07] and their extension for scale-space manifold analysis, the Growing Least Square method (GLS) [MGB*12]. Given a set of points $P = \{\mathbf{p}_i \in \mathbb{R}^d\}$, the APSS framework defines a smooth surface S_P approximating P by applying a moving least square spherical fit to the data, based on an algebraic form of the 0-isosurface of the following scalar field:

$$s_u(\mathbf{x}) = [1 \;\; \mathbf{x}^T \;\; \mathbf{x}^T\mathbf{x}] \cdot \mathbf{u} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{u} \in \mathbb{R}^{d+2} = [u_c \;\; \mathbf{u}_l \;\; u_q]^T$ is the vector of scalar coefficients of the sphere with $u_c, u_q \in \mathbb{R}$ $\mathbf{u}_l \in \mathbb{R}^d$. The goal of the GLS framework is to characterize any point $\mathbf{p} \in \mathbb{R}^d$ of a manifold at any scale t by an algebraic sphere that best approximates its neighborhood $\mathcal{P}_t = \{\mathbf{x}_i | \|\mathbf{x}_i - \mathbf{p}\| \leq t\}$, using the fast fitting technique proposed by Guennebaud et al. [GGG08]. It assumes that each point $\mathbf{x}_i$ is equipped with a normal $\vec{n}_i \in \mathbb{R}^d$. To give a geometric interpretation of the scalar field s_u, the framework applies the Prat normalization [Pra87] and proposes an alternative parametrization $s_\mathbf{p} = \langle t, \tau, \kappa, \phi, \eta \rangle$ based on five parameters: the scale value t; the algebraic offset distance τ between the evaluation point $\mathbf{p}$ and the 0-isosurface; the signed curvature κ of the hypersphere; the alignment measure ϕ between the scalar field and the input normals $\vec{n}_i$; the unit normal η of the scalar field at $\mathbf{p}$. With this parametrization, we can compare a pair of arbitrary scale-space locations $s_\mathbf{a} = \langle t_\mathbf{a}, \tau_\mathbf{a}, \kappa_\mathbf{a}, \phi_\mathbf{a}, \eta_\mathbf{a} \rangle$ and $s_\mathbf{b} = \langle t_\mathbf{b}, \tau_\mathbf{b}, \kappa_\mathbf{b}, \phi_\mathbf{b}, \eta_\mathbf{b} \rangle$ using the following dimension-less and scale-invariant dissimilarity function (see [MGB*12] for

more details):

$$\delta(s_{\mathbf{a}}, s_{\mathbf{b}}) = (\frac{\tau_{\mathbf{a}}}{t_{\mathbf{a}}} - \frac{\tau_{\mathbf{b}}}{t_{\mathbf{b}}})^2 + (t_{\mathbf{a}}\kappa_{\mathbf{a}} - t_{\mathbf{b}}\kappa_{\mathbf{b}})^2 + (\phi_{\mathbf{a}} - \phi_{\mathbf{b}})^2 \quad (2)$$

The parameter η is ignored in this pairwise comparison.

2.1. Related Work

Change detection has been extensively studied in computer vision with the goal of finding regions of change in images of the same scene taken at different times (see [RAAKR05] for an overview). Extending the problem to 3D data translates the problems to the 3D-to-3D and 2D-to-3D comparisons.

The most simple change detection technique consists in computing the distance of the point cloud from the reference 3D model. Assuming that the input clouds are already aligned, the idea is to compute the point-to-point, the point-to-mesh or the mesh-to-mesh Hausdorff distance, using for instance the Metro tool [CRS98], and to map this distance to a quality value range using a threshold that specifies the minimum displacement to be considered as a change. Montaut et al. [MRMT05] propose an octree-based strategy, comparing three different distance measures between the octree cells using the points in each cells: the average distance from the nearest neighbor; the angle between the best fitting planes; the Hausdorff distance. Butkienicz et al. [BCWR08] propose a solution for urban LIDAR change detection based on the projection of the point on a 2D Delaunay triangulation. Zeibak et al. [ZF07] convert a 3D laser scan in a range panorama whose axes are the latitudinal and longitudinal scanning angles and the ranges are the intensity values. After the alignment of the panoramic range maps, they do a direct comparison of the depth values for corresponding pixels. A final processing is done to correct the region with no-data. All these approaches exhibit a too local characterization of the points, with a flawed management of the intersections of two objects in the time. In this case all the methods return a distance very close to zero for the points near the intersection, even if they belong to different surfaces. Another challenging problem is the selection of the distance threshold for the change/no-change segmentation that makes the methods less robust against noise and density variation. Our solution overcomes these problems by computing a more robust characterization that depends on the orientation of the surrounding surface. The direct point clouds comparison problem is analyzed in a formal way in [MS04]. The main idea is to use the Gromov-Hausdorff distance in a probabilistic setting combined with a pairwise geodesic distance in order to obtain a computational implementation of the framework. The point clouds have to be densely and uniformly sampled from the metric space, they must represent unique objects (manifolds) and the comparison result is global.

Another class of solutions solve the problem from a sequence of images using 3D reasoning. Pollard et al. [PM07] propose an approach based on a 3D voxel model, where probability distributions for surface occupancy and image appearance are stored in each voxel. This probability distributions are used to compare a new input image with the existing model and to extract a change binary mask for the image. This approach was extended in [UM14] to generate 3D change volumes rather than pixel-level image change probabilities, without reconstructing a 3D model for the entire scene. Schindle et al. [SD10] present a general framework to recover the spatial and temporal information about the scene structure and the cameras (a date for each camera and a time interval for each 3D point). The framework decomposes the problem into two steps: first the traditional Structure from Motion problem and then a temporal inference problem by reasoning probabilistically about visibility and presence of the objects in the scene. Sakurada et al. [SOD13] propose a solution to compute a probabilistic density of depths and to estimate whether the scene changes or not by integrating the obtained depth density. A statistical approach is used also in [XVP13] to compute the consistency of the space occupancy from different 3D datasets. Taneja et al. [TBP11] exploits an existing 3D geometry to detect inconsistencies across the input images. After the image-to-geometry registration, a probabilistic framework verifies the image consistency with the geometry, incorporating semantic knowledge to ignore changes occurring on non relevant parts of the scene (vegetation, cars and pedestrians). A similar approach is used in [TBP13] to detect changes in the geometry of a city (cadastral 3D model) using panoramic images captured by a car driving around the city.

Li et al. [LFM*13] propose a specialized solution to analyze 4D point clouds of a plant and characterize its growing. They propose an interleaved spatial and temporal analysis in 4D to accurately locate budding and bifurcation events. They present a forward-backward analysis to detect future events and pull them back in time for an accurate location in their infancy. Denning et al. [DP13] present MeshGit, an algorithm for diffing and merging polygonal meshes from 3D modeling. It is based on a mesh edit distance, defined as the cost of matching vertices and faces between meshes, and on an iterative greedy algorithm to efficiently approximate this distance. Yan et al. [YSL*14] propose a new scanning method in which the user actively modifies the scene while scanning it, in order to reveal occluded regions and to reconstruct together the static parts into a complete unified 3D model. They take advantage from the temporal coherence of a continuous streams of 3D scans. Schmidt et al [SPA*14] present a new comparative visual analysis technique for 3D meshes which enables the simultaneous comparison and allows the interactive exploration of their differences.

3. Overview

Given two point clouds A and B, sampling the same environment and acquired in different times, the goal of the proposed algorithm is to automatically detect the geometric

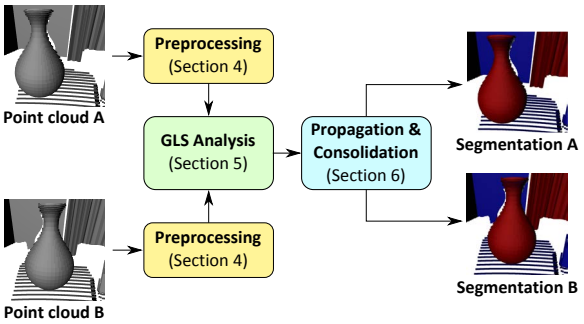


Figure 2: *Algorithm overview.*

changes and to segment the no-change ($\mathcal{NC}(A)$ and $\mathcal{NC}(B)$) and the change regions ($\mathcal{C}(A)$ and $\mathcal{C}(B)$) of each cloud. We assume that the input clouds have been already aligned. Our algorithm is organized in three steps (Figure 2):

1. preprocessing of each point cloud to remove the outliers and compute the required additional info, such as point radius and normal (Section 4);
2. detection of the changes, using the GLS framework to compute a multi-scale difference value for each point, followed by a consistency check to detect inconsistencies near the boundary with a change region (Section 5);
3. correction of the obtained results in special spatial configurations, such as occlusions and the motion with coherent overlapping in the time, using the temporal proximity and the propagation of the information from the surrounding regions (Section 6).

For the notation used in the following sections see the Appendix in the additional material.

4. Preprocessing

The main tasks to prepare the input data are the detection and removal of the outliers, the computation of the point radius and eventually the estimation of the point normals. The detection of the outliers is performed using the Local Outliers Probability (LoOP) measure [KKSZ09], a local density based method that provides an outlier score in the range of $[0,1]$. This value represents the probability of a point for being an outlier. We delete from the clouds all the points with a LoOP measure greater than 0.5 (Figure 3). For the computation of the point radius we use a local Gabriel Graph. For each point $\mathbf{p}$, we get the k-nearest neighbors $N_k(\mathbf{p})$ (16 in our test) and we compute the subset $S_k \subseteq N_k$ of points $\mathbf{x} \in N_k(\mathbf{p})$ such that $\forall \mathbf{s} \in N_k(\mathbf{p}) - \mathbf{x}$, $\|\mathbf{p}-\mathbf{s}\|^2 + \|\mathbf{x}-\mathbf{s}\|^2 \geq \|\mathbf{p}-\mathbf{x}\|^2$. This means that the sphere centered at the midpoint between $\mathbf{p}$ and $\mathbf{x}$ and with diameter equal to the edge from $\mathbf{p}$ to $\mathbf{x}$ does not contain any other point in $N_k(\mathbf{p}) - \mathbf{x}$. We assign as radius of the point $radius(\mathbf{p}) = \max\{\|\mathbf{p}-\mathbf{x}\| \mid \mathbf{x} \in S_k\}$. Finally, if the input clouds are not equipped with the point normals we compute them with a best fitting plane method [HDD*92] using 16 neighbors.

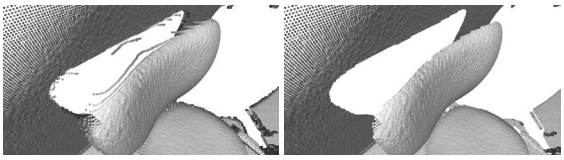


Figure 3: *Outlier removal: (Left) input point cloud; (Right) point cloud after outlier removal using a Local Outliers Probability threshold equal to* 0.5.

5. GLS Analysis

The general idea is to compute a multi-scale GLS descriptor for a uniform sub-sampling of the volume occupied by the point clouds and to map the differences of these descriptors in the time over the original clouds (see Algorithm 1). As already described in Section 1, the GLS framework guarantees robustness against the temporal interception of different surfaces and the different sampling characteristics of the clouds.

Algorithm 1 Change detection analysis by GLS

1: $cellSide \leftarrow 5 \cdot \text{COMPUTEAVGRADIUS}(A,B)$
2: $maxScale \leftarrow 2 \cdot cellSide$
3: $Q \leftarrow \text{EXTRACTNONEMTPYCELL}(A,B,cellSide)$
4: $GLS^A \leftarrow \text{COMPUTEGLS}(A,Q)$
5: $GLS^B \leftarrow \text{COMPUTEGLS}(B,Q)$
6: $DIF \leftarrow \text{COMPUTEDIFFERENCE}(GLS^A, GLS^B)$
7: $\text{MAPDIFFERENCE}(A, DIF, maxScale)$
8: $\text{MAPDIFFERENCE}(B, DIF, maxScale)$
9: $\text{CONSISTENCYCHECK}(A,B)$

The algorithm starts by embedding the point clouds in a uniform grid aligned with the union of the two bounding boxes. The size of the cell is chosen as a multiple of the mean of the average point radius of each cloud (five times the average radius). The middle points Q of the non-empty cells are used as query points for the computation of the GLS descriptors (Figure 4a). For each query point $\mathbf{c} \in Q$, the algorithm computes two GLS descriptors: $GLS^A_{\mathbf{c}}$ and $GLS^B_{\mathbf{c}}$ using only points of the cloud A and B, respectively, around $\mathbf{c}$. Each GLS descriptor $GLS^A_{\mathbf{c}} = \{s^A_{\mathbf{c}}[i] = \langle t_i, \tau_i, \kappa_i, \phi_i \rangle\}$ is made of 20 algebraic spheres computed by increasing the search radius (the scale value t) such that $\forall i \in [1 \ldots 20]$ $t_i = maxScale \cdot i/20$. The maximum radius/scale is twice the cell side. Doing so, we guarantee a minimum overlap between adjacent descriptors that makes the final computation of the change value more robust. For each query point $\mathbf{c}$, the GLS descriptors $GLS^A_{\mathbf{c}}$ and $GLS^B_{\mathbf{c}}$ are compared by computing the dissimilarity set $DIF_{\mathbf{c}} = \{\langle \delta_i, t_i \rangle_{\mathbf{c}} \mid \forall i \in [1 \ldots 20]\ \delta_i = \delta(s^A_{\mathbf{c}}[i],\ s^B_{\mathbf{c}}[i])\}$ using the function δ defined in Equation 2. The resulting dissimilarity sets $DIF_{\mathbf{c}}$ are used to compute the dissimilarity values for the original point cloud samples using a weighted scheme (Figure 4b). For each point $\mathbf{p}$ of the cloud, we collect the dissimilarity values $E_{\mathbf{p}}$ in the $DIF_{\mathbf{c}}$ at the smallest scales of each query point that contains the

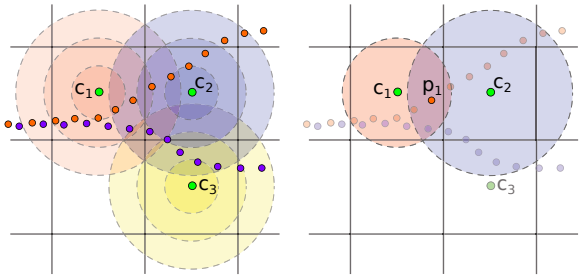
point $\mathbf{p}$:

$$E_{\mathbf{p}} = \{\langle \delta_i, t_i \rangle_{\mathbf{c}} \in DIF_{\mathbf{c}} \mid t_{i-1} < \|\mathbf{p}-\mathbf{c}\| \leq t_i\} \quad (3)$$

For example in the Figure 4b, for the point $\mathbf{p}$, we select the second scale of the query point $\mathbf{c}_1$ and the third scale of the query point $\mathbf{c}_2$, ignoring the scales of the query point $\mathbf{c}_3$ that do not contain $\mathbf{p}$. The change value $ch_{\mathbf{p}}$ is computed as weighted mean of the collected dissimilarity values in $E_{\mathbf{p}}$, with weight proportional to the distance from the query points of the descriptors:

$$ch_{\mathbf{p}} = \frac{\sum_{\langle \delta_i, t_i \rangle_{\mathbf{c}} \in E_{\mathbf{p}}} \delta_i \cdot (maxScale - \|\mathbf{p}-\mathbf{c}\|)}{\sum_{\langle \delta_i, t_i \rangle_{\mathbf{c}} \in E_{\mathbf{p}}} (maxScale - \|\mathbf{p}-\mathbf{c}\|)} \quad (4)$$

The dimensionless value of $ch_{\mathbf{p}}$ is in the range $[0,1]$ and it can be interpreted as a probability of the point to be a change. Analyzing the experiment results we found the value 0.05 as a good threshold for the change/no-change segmentation.



(a) (b)

Figure 4: *(a) Query points for the computation of the GLS descriptors (orange = point cloud A, violet = point cloud B, green = query points for the GLS computation). A dissimilarity set with 3 scales is associated to each query point. (b) Scales selected for the computation of the change value of the point* $\mathbf{p}_1$.

The results can exhibit some classification inconsistencies near the boundary between the change and the no change areas of the geometry, attributing a change status to a no-change point (see the point on the table near the phone in the second column of Figure 7a). This happens because, when the query point of a GLS descriptor is very close to a big change, the high dissimilarity values computed for the smallest scales propagate their influence to the larger ones. To solve this problem we add a consistency check based on the intuition that a false-positive change point $\mathbf{p}$ meets three conditions (Figure 5):

1. the dissimilarity value $min_{\mathbf{p}} = \arg\min_{\langle \delta_i, \ldots \rangle_{\mathbf{c}} \in E_{\mathbf{p}}} \|\mathbf{p}-\mathbf{c}\|$ of the smallest scale of the nearest query point is less than 0.05;
2. the nearest point $\mathbf{q}$ in the other cloud is close in Euclidean distance ($\|\mathbf{p}-\mathbf{q}\| < \varepsilon_1$ with $\varepsilon_1 = 2\,radius(\mathbf{p})$);
3. the nearest point $\mathbf{q}$ in the other cloud has similar normal ($\vec{n}_{\mathbf{p}} \cdot \vec{n}_{\mathbf{q}} > \theta_1$ with $\theta_1 = 0.99$).

The output of this consistency check is shown in the third columns of Figures 7a and 7b. The numerical values for this thresholding were found experimentally and worked for all the dataset we tested.

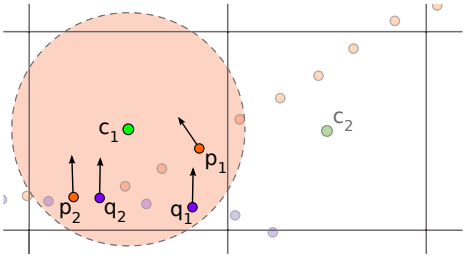


Figure 5: *Data involved in the consistency check for the points* $\mathbf{p}_1$ *and* $\mathbf{p}_2$. *The algorithm checks the change value of the smallest scale of the nearest query point* $\mathbf{c}_1$ *and the distance in space and normal with the nearest point in the other cloud, respectively* $\mathbf{q}_1$ *and* $\mathbf{q}_2$. *In the example, the check succeeds for* $\mathbf{p}_2$ *while fails for* $\mathbf{p}_1$.

The use of a uniform grid allows a faster processing by reducing the number of points actually used for the computation of the GLS descriptors. An alternative solution could be to uniformly sample the datasets using a temporal-aware Poisson-Disk pruning procedure [CCS12] but this is computational expensive (10 times slower). The discretization introduced by the grid in the computation of the change value $ch_{\mathbf{p}}$ is negligible thanks to the effect of the maximum scale value (twice the cell side size), which guarantees the overlap of adjacent descriptors and then a more robust change characterization. Finally the choice of the cell side size impacts, not only on the computation time (by reducing or increasing the number of GLS descriptors), but also on the minimum change features that we are able to detect: when the cell side size grows we lose the detection of small changes, while the areas with inconsistencies near a real change/no-change boundary, where we apply the consistency check, are extended. The values used for the cell side size, the maximum radius and the number of scales for GLS descriptor were selected experimentally trading computation time for classification accuracy.

6. Change Propagation and Consolidation

The results obtained with the procedure of Section 5 show an unsatisfying behavior in two critical cases having a specific spatial configurations. The first one is a small movement of an object over a plane so that the object overlaps itself in time (Figure 6a). This case involves objects with a zero Gaussian curvature that are moved along the zero principal curvature direction, aligning perfectly some regions of the geometry. For instance, when a book slightly moves over a desk, the algorithm detects a change only at the points around the sides

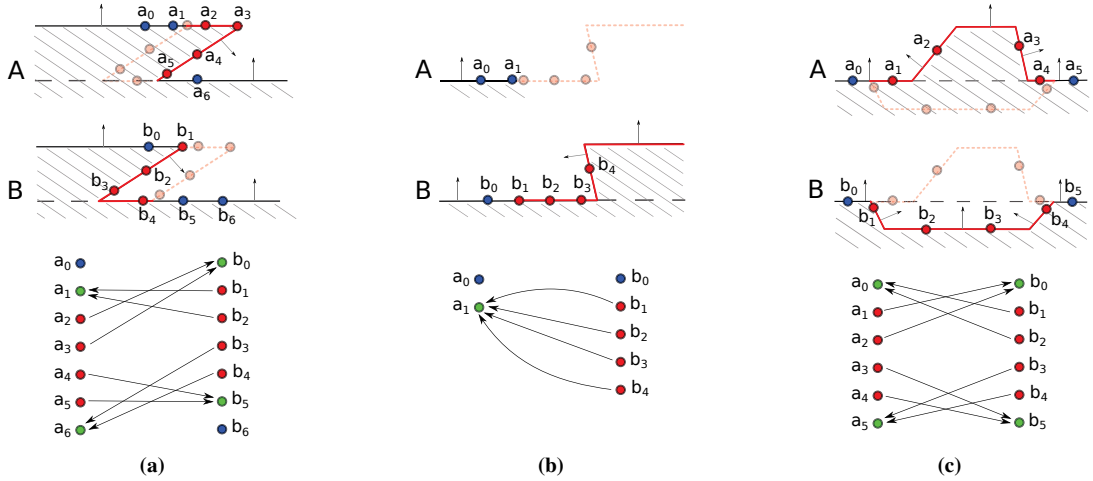


Figure 6: *2D example of critical configuration: (a) the small movement of a simple object over the plane; (b) occlusion; (c) temporal hole filling. The red segments are the regions detected as change with the algorithm in Section 5. On the bottom the temporal proximity graph (red point = change, blue point = no change, green point = no change point classified as accumulation point by the proximity graph). The transparent elements show the change points of the other time.*

of the book, ignoring the ones on the top since they have the exact same geometry in the two times. The second case is the acquisition of new geometry in regions occluded in the other scan (Figure 6b). In this case all the new acquired points are classified as change. We propose a solution to correct the classification using only geometric information and without any semantic data. More precisely, the method recognizes as change all points of the moved object and as no-change the points in the occluded area.

Algorithm 2 Propagation algorithm

```
1: BUILDPROXIMITYGRAPH(A, B)
2: repeat
3:     changeA ← PROPAGATECHANGE(A, B)
4:     changeB ← PROPAGATECHANGE(B, A)
5: until (changeA + changeB > 0)
6: PROPAGATENOCHANGE(A)
7: PROPAGATENOCHANGE(B)
```

To do so, we need to identify the regions characterized by these critical cases and to propagate the information. These objectives are reached in three steps as described in Algorithm 2: (i) the construction of a graph using the proximity information in time to identify the two critical cases (Section 6.1); (ii) the propagation of the change information on the points classified as no-change until there are no more updates on both the clouds (Section 6.2); (iii) the propagation of the no-change info to the points classified as change that have not updated their state in the previous step (Section 6.2). Figures 7a and 7b show the evolution of the change classification during the different steps of the algorithm in two examples with a critical spatial configuration.

6.1. Proximity Graph Construction and Analysis

To identify the critical cases our algorithm builds a temporal proximity graph between the point clouds A and B. This graph is a bipartite directed graph where there is a directed edge from a change point $\mathbf{x} \in \mathcal{C}(A)$ to a no-change point $\mathbf{y} \in \mathcal{NC}(B)$ if the point $\mathbf{y}$ is the closest one to $\mathbf{x}$ (see the graphs in the Figure 6). The closest point in $\mathcal{NC}(B)$ is chosen according to the geodesic distance on the implicit surface defined by the cloud A and vice-versa for the points in B. For the construction of the graph, we use a parallel flooding algorithm that spreads the minimum distance from a set of source points (in our case the no-change points near a border with a change regions) with an iterative procedure (see Algorithm 1 in the additional material). The obtained graph is characterized by vertices with in-degree greater than one (green points in the graphs in Figure 6) that we call *accumulation points*. These points can be classified in two categories, depending on the proximity of a similar accumulation point in the other point cloud (see Algorithm 3 in the additional material). For each accumulation point $\mathbf{x} \in A$ the method looks for the set $D = \{\mathbf{y} \in B \mid \|\mathbf{x}-\mathbf{y}\| \leq 2\, radius(\mathbf{x})\}$ of nearby points in the other cloud. If D does not contain any accumulation point then $\mathbf{x}$ is a single-time accumulation vertex ($\mathbf{a}_1$ in Figure 6b). This happens typically around the regions where the change classification is due to an occlusion. Otherwise $\mathbf{x}$ is a double-time accumulation vertex (points $\mathbf{a}_1, \mathbf{a}_6, \mathbf{b}_0, \mathbf{b}_5$ in Figure 6a) and it characterizes all the other kinds of changes.

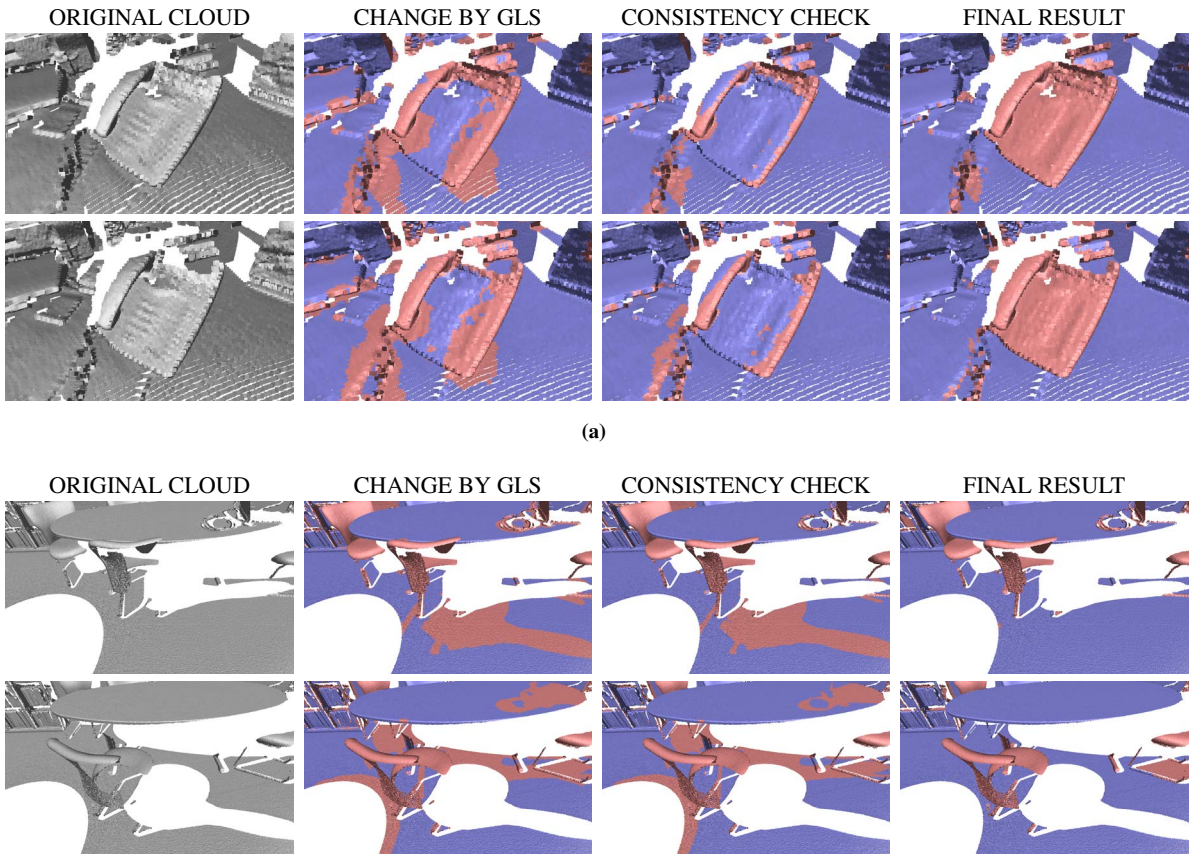


Figure 7: *Evolution of the change/no-change segmentation (red = change, blue = no-change) during the main steps of the algorithm in the case of critical configurations: (a) a small movement of an object over a plane (a phone slightly rotated over the desk); (b) occlusion (region of the floor occluded by a chair that has been moved before the second scan). For each sub-figure: (Top) OFFICE T0. (Bottom) OFFICE T1.*

6.2. Propagation procedure

The propagation of the change and no-change information in the cloud uses an iterative approach that proceeds in two stages: (i) the selection of reliable points that can propagate their information on the near points with a different state; (ii) the propagation of the information using a local planar test. The algorithm starts with the propagation of the change data (see Algorithm 5 in the additional material).

During the first iteration the algorithm selects the change points that have a double-time accumulation vertex as nearest in the proximity graph and are outside the volume defined by the other point cloud. For example, in Figure 6a, the points $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4$, which are outside the volume of the cloud B, are selected as good candidates for the propagation while $\mathbf{b}_2$, $\mathbf{b}_3$, $\mathbf{b}_4$, which are inside the volume of the cloud A, are ignored. To determine if a point is inside the volume, we locally fit an algebraic sphere. Given a change point $\mathbf{x} \in \mathcal{C}(A)$ and its nearest point $\mathbf{y} \in B$ according the Euclidean distance, we retrieve all the points $Q = \{\mathbf{p} \in B \mid \|\mathbf{p}-\mathbf{y}\| \leq \|\mathbf{x}-\mathbf{y}\|\}$. Then we compute the algebraic sphere s_u using Q and we check the product $u_q \cdot s_u(\mathbf{x})$ between the quadratic term u_q of the sphere and the value of the scalar field defined by s_u in the point $\mathbf{x}$. If the value is greater than zero then the point $\mathbf{x}$ is outside the volume of B otherwise it is inside. In the following iteration the algorithm simply selects all the points that have just modified their state in the previous one.

For the propagation stage the algorithm tries to spread the change info of the selected points on the close no-change points that are on the same plane. Formally, given a point $\mathbf{x}$, for each close no-change point $\mathbf{y} \in \mathcal{NC}(A)$ such that $\|\mathbf{x}-\mathbf{y}\| \leq radius(\mathbf{x})$, the method modifies the state of $\mathbf{y}$ from no-change to change if the following condition is true:

$$\vec{n}_\mathbf{x} \cdot \vec{n}_\mathbf{y} > \theta_2 \quad \wedge \quad |(\mathbf{y}-\mathbf{x}) \cdot \vec{n}_\mathbf{x}| < \varepsilon_2 \tag{5}$$

with $\vec{n}_{\mathbf{x}}$ and $\vec{n}_{\mathbf{y}}$ the normals of the points $\mathbf{x}$ and $\mathbf{y}$. In all our experiments we used the same values for the thresholds $\theta_2 = 0.99$ and $\varepsilon_2 = radius(x)/10$. During this propagation stage, when a point $\mathbf{x} \in A$ modifies its status, it sets as change also the nearest point $\mathbf{p} \in B$ iff $\mathbf{x}$ and $\mathbf{p}$ satisfy the condition of Equation 5. This check guarantees the propagation of the change info in the other point cloud that could not have change points outside the volume. For example in Figure 6a the point cloud B does not have any change point outside the volume of A that can propagate the change info on the top plane. To solve this situation, during the first iteration the point $\mathbf{a}_0$ propagates its state to the point $\mathbf{a}_1$ and to the nearest point $\mathbf{b}_1$ in the cloud B. In this way, in the following iteration, $\mathbf{b}_1$ can continue to propagate its change info, inducing a correct characterization of the other points on the top plane. The algorithm stops when no point modifies its state.

The final step is the propagation of the no-change information using a similar procedure to the propagation of the change status (see Algorithm 4 in additional material). In the first iteration the method modifies the state of the change points that have a single-time accumulation vertex as nearest in the proximity graph. For example, in Figure 6b, after the first iteration, $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$, $\mathbf{b}_4$ modify their state in no-change. Then it applies the same iterative flooding algorithm to spread the no-change state to the nearest change points that satisfy the conditions in Equation 5. In this process, the algorithm ignores the points that have just switched state in the previous propagation of the change info. For example, in Figure 6a, this step classifies as no change the point $\mathbf{b}_4$ that in the other time was occluded by the motion of an object.

In these propagation steps there is a special case that needs more attention: the temporal hole filling, that is a hole closed in the subsequent scan (Figure 6c). In this case we want to avoid the propagation of the change info of the points that cover the hole over the near no-change points and, vice-versa, the propagation of the no-change info over the points of the hole. The algorithm identifies the change patches that are outside the volume, using a clustering method based on the local density (all the change points within the point radius belong to the same patch). If for each change point on the border of the patch there exist a close no-change point that satisfies Equation 5 then all the points inside the patch are hidden to the propagation procedures (they cannot propagate their change state and cannot update their state in no-change). For example in Figure 6c, the points $\mathbf{a}_1$ and $\mathbf{a}_4$ cannot propagate their change info to $\mathbf{a}_0$ and $\mathbf{a}_5$ and vice versa $\mathbf{a}_0$ and $\mathbf{a}_5$ cannot propagate its no-change info to $\mathbf{a}_1$ and $\mathbf{a}_4$, therefore they retain the current state.

7. Results

We tested our algorithm on synthetic and real data. The synthetic dataset is composed by two point clouds (LIVING T0, LIVING T1) of a living room generated by ray casting a modeled triangle mesh using a panoramic camera model. The point clouds were acquired from different viewpoint and were generated by moving some objects in the room. This dataset has been used as ground truth to test the robustness of the proposed method and to compute the error metrics. Figure 8 shows the comparison of the segmentation obtained by the proposed algorithm with the ground truth. The figure shows also the results of two state-of-the-art algorithms. The first one (NEAREST DISTANCE) computes a segmentation based on a threshold on the distance of the nearest neighbor in the other cloud. It produces inaccurate results near the intersection of two surfaces and where the scanner follows different sampling directions. The second ones (DEPTH COMPARISON) computes a segmentation based on the depth difference along the scanning direction. We create a panoramic depth map for each point cloud, we align the depth maps and we compare the depth values of corresponding pixels. For this method we need to know the position in space and the sampling steps of the scanner (number of points along the latitude and the longitude axes). The results of our method are very accurate with a low number of points with a wrong change classification (see Table 1). Generally these wrong points are located around very thin objects, where the propagation step of the change/no-change information has a higher probability to fail. Table 1 shows the statistics (number and percentage of points with a correct and wrong classification) of the results obtained by our method and the two state-of-the-art algorithms in the LIVING ROOM dataset.

	# True change	# True no change	# False change	# False no change
	GROUND TRUTH			
LIVING T0	165946 (2.924%)	5508694 (97.076%)	0 (0%)	0 (0%)
LIVING T1	166414 (2.935%)	5503521 (97.065%)	0 (0%)	0 (0%)
	OUR METHOD			
LIVING T0	165619 (2.9186%)	5508512 (97.0726%)	182 (0.0032%)	317 (0.0056%)
LIVING T1	166209 (2.932%)	5503495 (97.064%)	26 (0.0004%)	205 (0.0036%)
	NEAREST DISTANCE			
LIVING T0	143667 (2.532%)	5342394 (94.145%)	166300 (2.931%)	22269 (0.392%)
LIVING T1	144311 (2.545%)	5372273 (94.75%)	131248 (2.315%)	22103 (0.39%)
	DEPTH COMPARISON			
LIVING T0	149608 (2.637%)	5384700 (94.89%)	123994 (2.185%)	16328 (0.288%)
LIVING T1	150486 (2.654%)	5411156 (95.436%)	92365 (1.629%)	15928 (0.281%)

Table 1: *Statistics (number of points and percentage against the size of each cloud) on the segmentation results showed in Figure 8 against the ground truth of the synthetic dataset (LIVING T0 = 5674630 points, LIVING T1 = 5669935 points).*

We also made experiments with real datasets, by captur-

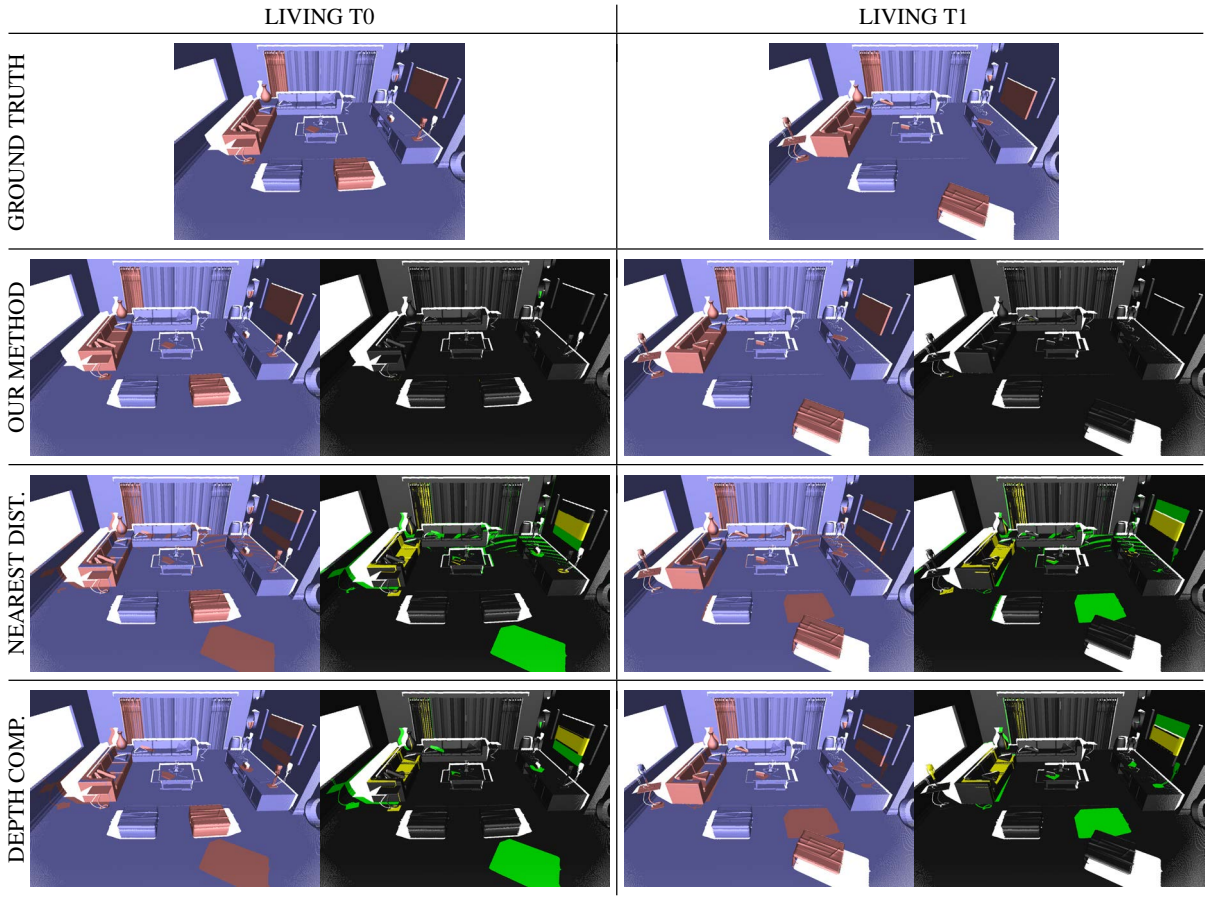


Figure 8: *Comparison of the segmentation of the proposed algorithm with the ground truth (synthetic dataset). The Figure shows also the results of two other state-of-the-art algorithms: (NEAREST DIST.) threshold on the distance of the nearest neighbor in the other time (threshold 3mm); (DEPTH COMP.) threshold on the depth difference along the scanning direction between the two scans (threshold 2mm). On the right of each segmentation result (red = change, blue = no-change) there is the map of the differences from the ground truth (black = point with a correct classification, yellow = false no-change points, green = false change points).*

	Cloud1	Cloud2	Outliers1	Outliers2	Changes1	Changes2	Time
OFFICE T0 - OFFICE T1	6354k	6363k	293k (4.61%)	292k (4.58%)	517k (8.13%)	470k (7.38%)	251 sec
OFFICE T0 - OFFICE T2	6354k	6358k	293k (4.61%)	299k (4.70%)	732k (11.58%)	640k (10.06%)	339 sec
OFFICE T1- OFFICE T2	6363k	6358k	292k (4.58%)	299k (4.70%)	607k (9.53%)	566k (8.9%)	297 sec
LAB T0 - LAB T1	6226k	6194k	328k (5.26%)	327k (5.27%)	855k (13.73%)	695k (11.22%)	368 sec
OFFICE S1 - OFFICE S0.125	6354k	794k	293k (4.61%)	30085 (3.70%)	10525 (0.15%)	897 (0.11%)	88 sec
OFFICE S1 - OFFICE S0.125 DOWN	6354k	794k	293k (4.61%)	30085 (3.70%)	36989 (0.56%)	5217 (0.62%)	55 sec
OFFICE P1 - OFFICE P2	6280k	6283k	300k (4.77%)	303k (4.82%)	1084k (17.26%)	984k (15.66%)	504 sec
OFFICE P1 - OFFICE P2 DOWN	6280k	6283k	300k (4.77%)	303k (4.82%)	1100k (17.51%)	820k (13.05%)	112 sec

Table 2: *Test case and performance data*

ing two rooms in different times with a ToF laser scanner. The first dataset shows an office in three times (OFFICE T0, OFFICE T1, OFFICE T2)(for the comparisons of each pair of scans see Figures 9, 10 and 11). The second dataset shows a lab in two times (LAB T0, LAB T1) (Figure 12). All the figures show a binary segmentation (red = change, blue = no-change).

The different steps of the algorithm use three types of thresholds: i) the change/no-change classification threshold used in Equation 4; ii) the distance between normals, defined as cosine of the angle of the two vectors, used in the consis-

tency check (θ_1) and in the Equation 5 (θ_2); iii) the distances ε_1 and ε_2 between points used in the consistency check and in Equation 5. The thresholds for the change/no-change classification and the distance between normals are numerical dimensionless values (respectively 0.05 and 0.99) that were found experimentally and worked for all the dataset we tested. The thresholds for the points distance are defined with respect to the local features of the point clouds, like the local radius, using the dimensional unit of the acquisition device of the scans.

Table 2 contains the info about the test cases and the processing time. For each test we have the number of points of the input clouds, the number and the percentage of removed outliers, the number and the percentage of points detected as changes and the time taken by the processing. We perform our test on a PC with an Intel Core i7-3770 with 16GB of RAM. Typically the processing of 6M point clouds takes from 4 to 6 minutes, depending on the number of real changes. To speed-up the search of the nearest points needed in several steps of the algorithm, we compute a kd-tree of each point cloud. In general the results show a precise change/no-change segmentation, with a good interpretation of the critical spatial configurations described in Section 6 (Figures 7a and 7b). Still, some classification errors remain on thin objects that present very few points.

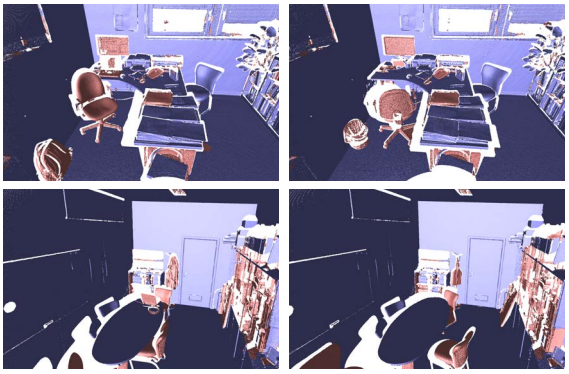


Figure 9: *Change detection results between OFFICE T0 (Left) and OFFICE T1 (Right) in two different views. More info in the 1st row of the Table 2.*

7.1. Robustness against density

To study the robustness against the point density we compared a scan (OFFICE S1) with three different sub-sampling versions of the same scan with 50% (OFFICE S0.5), 25% (OFFICE S0.25) and 12.5% (OFFICE S0.125) of the original points. These versions are obtained with a random selection of the original points with uniform distribution. As shown in the left column in Figure 13 for OFFICE S0.125 (see Figure 5 in additional material for the other cases), the obtained results remain coherent, even if the algorithm detects few false change points on thin structures. More data about for OFFICE S0.125 is reported in Table 2 (5th row). To speed-up the algorithm in the case of very different local density we tested a sub-sampling preprocessing step in or-

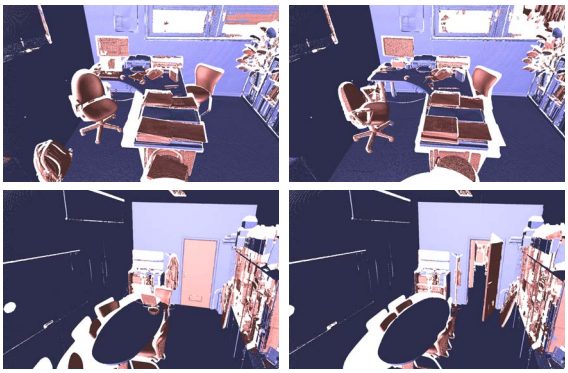


Figure 10: *Change detection results between OFFICE T0 (Left) and OFFICE T2 (Right) in two different views. More info in the 2nd row of the Table 2.*

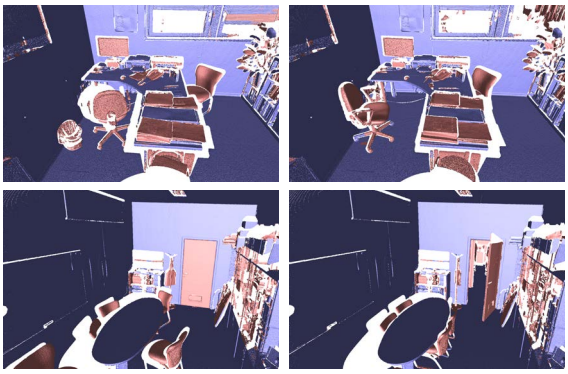


Figure 11: *Change detection results between OFFICE T1 (Left) and OFFICE T2 (Right) in two different views. More info in the 3rd row of the Table 2.*

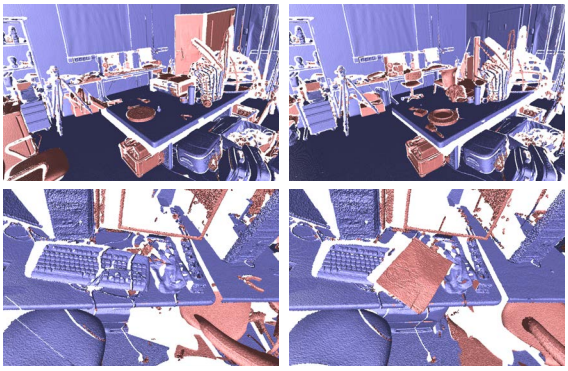


Figure 12: *Change detection results between LAB T0 (Left) and LAB T1 (Right) in two different views. More info in the 2nd row of the Table 2.*

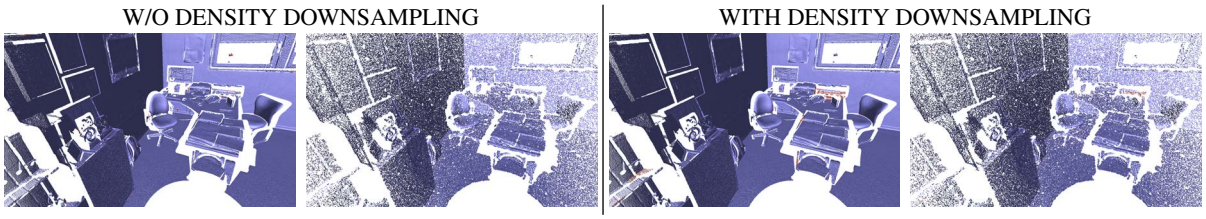


Figure 13: *Change detection between a scan and its sub-sampled versions to verify the robustness against the point density (OFFICE S1 vs OFFICE S0.125). The left column shows the result obtained using directly the two scans while the right column shows the result with the sub-sampling preprocessing to reduce the scans at the same local density.*

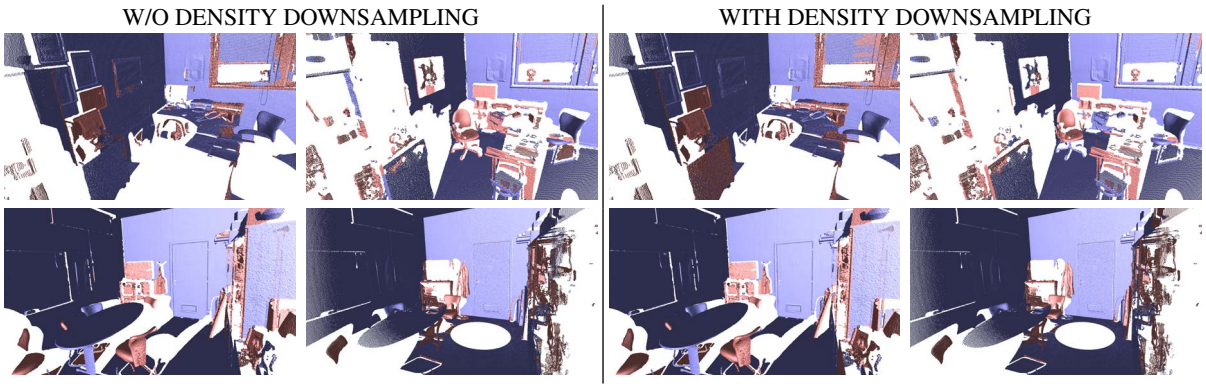


Figure 14: *Change detection between two scans of the office acquired from the opposite corner of the room (OFFICE P1 and OFFICE P2) and displayed from different viewpoints. The left column shows the result obtained using directly the two scans. The right column shows the result with the sub-sampling preprocessing to reduce the scans at the same local density.*

der to reduce the point clouds at the same local density. This sub-sampling procedure is based on a variable radii Poisson disk pruning where, for each point, the disk radius is equal to the maximum between its radius and the radius of the nearest point in the other cloud. In order to obtain a smooth variation of the scalar field that defines the disk radius, we apply a local mean filtering of the field. The right column of Figure 13 shows the obtained results after the sampling processing. The algorithm extracts the sub-sampled versions of the clouds, it computes the change quality values and updates the change values on the original clouds by up-sampling (weighted average using the distance from the neighbors in the sub-sampled cloud, where the neighbors are identified by the local construction of a Gabriel Graph). Comparing the results (right column Figure 13) with those computed directly on the original clouds (left column Figure 13), the differences are negligible with the advantage of a faster processing time (6th row in Table 2). These two characteristics are very useful when we want to compare two 3D scans of the same environment acquired from different viewpoints, like the example in Figure 14 (acquisition of the office from two opposite corners: OFFICE P1 and OFFICE P2). In this case, the detection results are quite similar but the processing time is four times faster (see last two rows in Table 2).

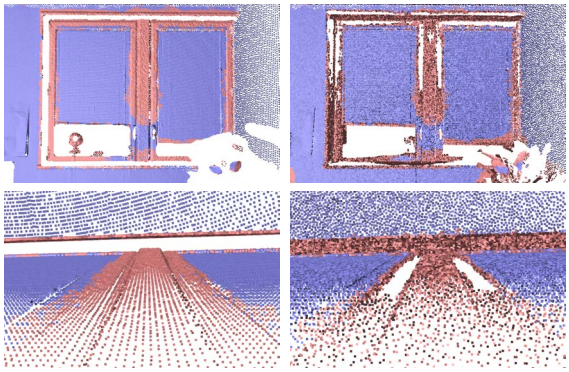


Figure 15: *Change detection results in noisy point clouds: (Left) OFFICE P1; (Right) OFFICE P2. In this case the detection of change points on the aluminum window frame is due to the high level of noise in the scan OFFICE P2.*

7.2. Limitations

Figure 15 shows the change/no-change segmentation in noisy data (a window with an aluminum frame). In this case the algorithm detects as change almost all the points on the frame due to the high level of noise that characterizes the second acquisition. In the specific the noise does not allow a

robust comparison of the GLS descriptors and prevents the propagation of the change info due to its influence on the estimation of the normal (it is more probable that Equation 5 is unsatisfied). A more robust and complex comparison should be able to detect the first scale of the GLS descriptors where the contribution of the noise in the estimation of the algebraic sphere is negligible (a larger scale than the one currently used). This implies to lose all the changes detected up the scale that we need to ignore for the noise. However the selection of such a scale is not trivial and should be spatially adaptive to the noise level. We aim at exploring this research direction in the future. For these reasons, our algorithm is not suitable for the comparison of highly noisy point clouds, such as the output of Structure-From-Motion algorithms. For this class of input, the state-of-art methods that work directly on the images are more appropriate.

7.3. Enhanced 3D Reconstruction

An advantage of using a precise change/no-change segmentation is the possibility to enhance the output of a 3D reconstruction algorithm starting from inputs similar to our datasets. For example, Figure 16 (and Figure 6 in the additional material) shows the triangulation results obtained using different inputs: the cloud OFFICE T0; the union of OFFICE T0 with the no change regions of the cloud OFFICE T1; the simply union of the two clouds OFFICE T0 and T1. The triangle meshes are obtained with the volumetric reconstruction algorithm available in MeshLab [CCC*08], based on a discrete distance field and a standard Marching Cubes algorithm. In this case, the use of our segmentation allows the reconstruction of a more complete model (more data on the desk and under the occlusion of the chair and of the bag) without inconsistencies (better reconstruction of phone and chairs).

8. Conclusion

We have presented a new approach for the automatic detection of temporal geometric changes in point clouds. Our method computes a multi-scale shape descriptor for few points in the volume and then map the differences of these descriptors in the time over the original clouds. This approach allows to characterize every point by using the orientation of the surrounding implicit surface computed using a Moving Least Squares approach. The results after this first step show a good change/no-change segmentation with some inconsistencies due to special geometric configurations, which are handled by modifying the change classification using only geometric information around the point without any semantic data. The final segmentation is accurate and consistent in the detection of the real changes, as proved by the statistics on the synthetic dataset. The algorithm shows robustness against high density variations between the two clouds and a reasonable tolerance to noise, detecting as a change only the noisier surfaces. Finally we have presented a simple preprocessing sub-sampling procedure to reduce at the same density point clouds acquired from different positions. It introduces a limited amount of errors in the change classification but permits significantly faster processing.

As future work, we plan to account for color data in the change detection. The main challenge is the extension of the GLS framework in order to take account of the color differences. We envisage two possible solutions: (a) integrating the color data in the weighting function used for the fitting of the algebraic sphere; (b) fitting the data in a 6D space (position + color). A further extension is the development of an interactive change detection method by moving the computation on GPU. Last, our segmentation results can be also instrumental to perform higher level analysis such as understanding what moved, what is new and what disappeared.

Acknowledgment

The research leading to these results was funded by EU FP7 project ICT FET Harvest4D (http://www.harvest4d.org/, G.A. no. 323567).

References

[BCWR08] BUTKIEWICZ T., CHANG R., WARTELL Z., RIBARSKY W.: Visual analysis and semantic exploration of urban lidar change detection. In *Proceedings of the 10th Joint Eurographics / IEEE - VGTC Conference on Visualization* (Eindhoven, The Netherlands, 2008), Eurographics Association, pp. 903–910. 3

[CCC*08] CIGNONI P., CALLIERI M., CORSINI M., DELLEPIANE M., GANOVELLI F., RANZUGLIA G.: Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference* (Salerno, 2008), Scarano V., Chiara R. D., Erra U., (Eds.), Eurographics, pp. 129–136. 12

[CCS12] CORSINI M., CIGNONI P., SCOPIGNO R.: Efficient and flexible sampling with blue noise properties of triangular meshes. *Visualization and Computer Graphics, IEEE Transactions on 18*, 6 (June 2012), 914–924. 5

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174. 3

[DP13] DENNING J. D., PELLACINI F.: Meshgit: Diffing and merging meshes for polygonal modeling. *ACM Transaction on Graphics 32*, 4 (July 2013), 35:1–35:10. 3

[GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. *ACM Transaction on Graphics 26*, 3 (July 2007). 2

[GGG08] GUENNEBAUD G., GERMANN M., GROSS M. H.: Dynamic sampling and rendering of algebraic point set surfaces. *Compututer Graphics Forum 27*, 2 (2008), 653–662. 2

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph. 26*, 2 (July 1992), 71–78. 4

[KKSZ09] KRIEGEL H.-P., KRÖGER P., SCHUBERT E., ZIMEK A.: Loop: Local outlier probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (Hong Kong, China, 2009), ACM, pp. 1649–1652. 4

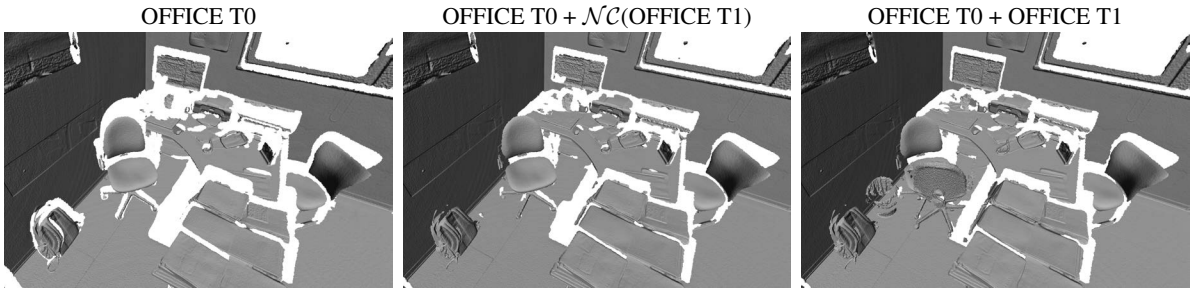


Figure 16: *Triangulation results from three inputs: (Left) OFFICE T0; (Center) OFFICE T0 plus the no change regions of OFFICE T1; (Right) union of OFFICE T0 and OFFICE T1. The pictures in the center show a more complete reconstruction without inconsistencies due to the intersection of different objects.*

[LFM*13] LI Y., FAN X., MITRA N. J., CHAMOVITZ D., COHEN-OR D., CHEN B.: Analyzing growing plants from 4d point cloud data. *ACM Transaction on Graphics 32*, 6 (Nov. 2013), 157:1–157:10. 2, 3

[MGB*12] MELLADO N., GUENNEBAUD G., BARLA P., REUTER P., SCHLICK C.: Growing least squares for the analysis of manifolds in scale-space. *Computer Graphics Forum 31*, 5 (Aug. 2012), 1691–1701. 2

[MRMT05] MONTAUT D. G., ROUX M., MARC R., THIBAULT G.: Change detection on point cloud data acquired with a ground laser scanner. In *Proceedings of the ISPRS Workshop Laser scanning* (Enschede, Netherlands, 2005), ISPRS, pp. 30–35. 2, 3

[MS04] MÉMOLI F., SAPIRO G.: Comparing point clouds. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Nice, France, 2004), ACM, pp. 32–40. 3

[PM07] POLLARD T., MUNDY J. L.: Change detection in a 3-d world. In *IEEE Conference on Computer Vision and Pattern Recognition* (Minneapolis, 2007), IEEE, pp. 1–6. 2, 3

[Pra87] PRATT V.: Direct least-squares fitting of algebraic surfaces. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 145–152. 2

[RAAKR05] RADKE R. J., ANDRA S., AL-KOFAHI O., ROYSAM B.: Image change detection algorithms: A systematic survey. *IEEE Transaction on Image Processing 14*, 3 (Mar. 2005), 294–307. 3

[SD10] SCHINDLER G., DELLAERT F.: Probabilistic temporal inference on reconstructed 3d scenes. In *IEEE Conference on Computer Vision and Pattern Recognition* (San Francisco, 2010), IEEE, pp. 1410–1417. 3

[SOD13] SAKURADA K., OKATANI T., DEGUCHI K.: Detecting changes in 3d structure of a scene from multi-view images captured by a vehicle-mounted camera. In *IEEE Conference on Computer Vision and Pattern Recognition* (Columbus, 2013), IEEE, pp. 137–144. 3

[SPA*14] SCHMIDT J., PREINER R., AUZINGER T., WIMMER M., GROLLER M., BRUCKNER S.: Ymca - your mesh comparison application. In *IEEE Conference on Visual Analytics Science and Technology,* (Paris, Oct 2014), IEEE, pp. 153–162. 3

[TBP11] TANEJA A., BALLAN L., POLLEFEYS M.: Image based detection of geometric changes in urban environments. In *IEEE International Conference on Computer Vision* (Barcelona, 2011), IEEE, pp. 2336–2343. 2, 3

[TBP13] TANEJA A., BALLAN L., POLLEFEYS M.: City-scale change detection in cadastral 3d models using images. In *IEEE Conference on Computer Vision and Pattern Recognition* (Columbus, 2013), IEEE, pp. 113–120. 2, 3

[UM14] ULUSOY A., MUNDY J.: Image-based 4-d reconstruction using 3-d change detection. In *European Conference on Computer Vision* (Zurich, 2014), Fleet D., Pajdla T., Schiele B., Tuytelaars T., (Eds.), Springer International Publishing, pp. 31–45. 3

[XVP13] XIAO W., VALLET B., PAPARODITIS N.: Change detection in 3d point clouds acquired by a mobile mapping system. In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* (Antalya, Turkey, 2013), vol. II-5/W2, ISPRS, pp. 331–336. 3

[YSL*14] YAN F., SHARF A., LIN W., HUANG H., CHEN B.: Proactive 3d scanning of inaccessible parts. *ACM Transaction on Graphics 33*, 4 (July 2014), 1–8. 2, 3

[ZBVH09] ZAHARESCU A., BOYER E., VARANASI K., HORAUD R.: Surface feature detection and description with applications to mesh matching. In *IEEE Conference on Computer Vision and Pattern Recognition* (June 2009), IEEE, pp. 373–380. 2

[ZF07] ZEIBAK R., FILIN S.: Change detection via terrestrial laser scanning. In *Proceedings of the ISPRS Workshop on Laser Scanning* (Espoo, Filand, 2007), ISPRS, pp. 430–435. 3