https://doi.org/10.2352/ISSN.2470-1173.2019.7.IRIACV-455

Change Detection in Cadastral 3D Models and Point Clouds and Its Use for Improved Texturing

Sander Klomp; Eindhoven University of Technology, SPS-VCA group of Electr. Eng., Eindhoven, the Netherlands
Bas Boom; CycloMedia Technology, Zaltbommel, the Netherlands
Thijs van Lankveld; CycloMedia Technology, Zaltbommel, the Netherlands
Peter H.N. de With; Eindhoven University of Technology, SPS-VCA group of Electr. Eng., Eindhoven, the Netherlands

Abstract

By combining terrestrial panorama images and aerial imagery, or using LiDAR, large 3D point clouds can be generated for 3D city modeling. We describe an algorithm for change detection in point clouds, including three new contributions: change detection for LOD2 models compared to 3D point clouds, the application of detected changes for creating extended and textured LOD2 models, and change detection between point clouds of different years. Overall, LOD2 model-to-point-cloud changes are reliably found in practice, and the algorithm achieves a precision of 0.955 and recall of 0.983 on a synthetic dataset. Despite not having a watertight model, texturing results are visually promising, improving over directly textured LOD2 models.

Introduction

Since increasingly more municipalities use 3D cadastral data, the correctness of 3D models is becoming crucial for various city-management applications. Today, many modern cities maintain databases of 3D Level Of Detail 2 (LOD2) [1] models of buildings for cadastral applications, such as real-estate evaluation, city planning, or property visualization. However, many of these models are either poorly constructed, or no longer represent reality because cities continuously change. By combining terrestrial images and aerial imagery, or using LiDAR, large 3D point clouds can be generated to update existing 3D city models.

Large-scale image and LiDAR data is highly suited for detecting changes at city scale, as has been shown for aerial photographs [2], LiDAR [3], terrestrial images [4], or combinations [5]. Especially LiDAR is useful for detecting small geometric differences, due to the geometric accuracy of LiDAR point clouds [5]. Post-processing of changes, by filtering out uninteresting changes such as pedestrians, cars and vegetation, is a further possibility [6][7]. For cadastral data, change detection has also been performed, by using 2D image projections [8]. Opposed to using projections in 2D photographs, performing the change detection on 3D point clouds has the advantage of being less sensitive to the accuracy of the 3D data [9]. This is especially important if aerial imagery are used for creating the point cloud.

This paper explores the possibility of using 3D point clouds for accurate change detection in cadastral city models. Directly meshing the point cloud for creating an LOD2 model leads to noisy surfaces and model imperfections. Our first contribution is the detection of coarse-scale changes from 3D point clouds at the level of 3D cadastral models of buildings. The second, most novel contribution, directly extends the LOD2 model with planar surfaces found in point clouds. A new implementation of generalized alpha hull [10] is employed, which creates a simplified polygon with a minimal number of vertices. Both contributions apply to Method 1. The third contribution is a point-cloud-based change detection in urban environments, which is resilient to noise (Method 2). The algorithm for detecting extensions to LOD2 models is evaluated both visually on real data, and quantitatively on a synthetic dataset, yielding attractive visual results.

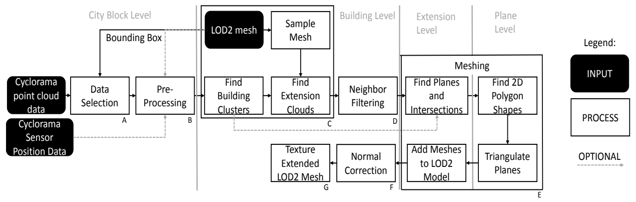


Figure 1. *Extension detection and texturing pipeline. Capitals below processes are noted as 'stages' in the text.*

Method 1: Cadastral change detection and model correction

Given a cadastral LOD2 3D model of a building and a set of aerial and terrestrial images, the objective of our algorithm is to find an extended LOD2 model, which has been corrected using the urban point cloud from the images. To achieve this, we find the significant differences between the LOD2 model and urban point cloud and create simple planar representations of these differences. The algorithm employs the framework as depicted in Fig. 1. First a building of interest is selected, i.e. an LOD2 mesh, whose bounding box is used to filter the point cloud. After that, the ground plane is removed to reduce false positives, and point cloud clusters of buildings are extracted. The LOD2 model is point-sampled and compared to the detected building clusters to find extensions. Then, planes are fitted to the extensions, converted to simple meshes, added to the LOD2 model, and finally textured. Each of the pipeline stages of Fig. 1 is explained below.

Data selection

The LOD2 models are triangular meshes of simple building blocks, while the input city data is a point cloud. Relevant regions are selected, based on their intersection within an extended axis-aligned bounding box around the LOD2 model. Note that this means that extensions having dimensions larger than the bounding box extension are not guaranteed to be entirely detected. We assume that such large extensions are special cases, and are not relevant to the rest of the algorithm.

Pre-processing

The LOD2 model location and point cloud coordinates are accurate enough without registration. This leaves two pre-processing steps: downsampling of the point clouds to ensure that further processing thresholds work well, and ground removal to individually cluster building point clouds.

First, downsampling is applied if needed. The framework uses many thresholds that depend on the density of the data. Hence, downsampling the point cloud to match the density for which all default parameter values are intended, is simpler than retuning all thresholds. As only coarse differences are desired, data loss due to downsampling is not an issue. The downsampling is performed using a voxel grid, where each downsampled point is the centroid of all points contained in the voxel. Second, ground removal is required to prevent region growing beyond the building of interest. Hence, the ground points are removed from the point cloud. This process consists of three steps.

1. Compute point normals by applying PCA on the covariance matrix of the neighboring points within radius r.
2. Remove points that have insufficient neighbors to compute the normals. This serves as outlier removal.
3. Remove points whose z-coordinate is within a margin around an estimated ground level and simultaneously the normal is close to a vertical orientation. Both the margin and allowed normal range are computed from a maximum ground slope.

Extension detection

Finding the separate extension point clouds consists of first splitting the point cloud into separate buildings, then sampling the LOD2 model to allow comparison, and finally detecting extension clusters by comparing the buildings to the sampled LOD2 model. These three processes use various Euclidean distance-based techniques to cluster and filter points.

From the groundless point cloud, the building of interest can be separated from the other buildings with Euclidean clustering [11]. Connected buildings will still be included in this cluster, which will be compensated at a later phase. Directly comparing a point cloud to a mesh can be expensive without intelligent and complex use of octrees. Instead, the LOD2 mesh is converted to a point cloud by simply sampling its faces. This sampling adds points randomly on all triangular mesh faces, with uniform distribution. Theoretically, this means that 'holes' may occur in a triangle when the added points are randomly placed only near the edges of the triangle. However, when sampling is sufficiently dense, this is not a problem in practice. All mesh vertices are included in the sampled model, as these corner points are often relevant when computing distances between point clouds. Now that all data are in point cloud format, the LOD2 model cloud can be 'subtracted' from the building cloud, by removing all points from the real point cloud within a thresholded distance to the sampled LOD2 model point cloud. This leaves only points that are extensions, strong noise, or connected neighboring buildings.

Neighboring building filtering

Discovered extensions may contain neighboring buildings due to the Euclidean clustering. Connected neighboring buildings are removed by using their corresponding LOD2 models. It is assumed that any extension that is above a neighbor's ground face should be part of that neighbor, and not the current building. The extension points are projected to the ground plane, and filtered based on their distance to the neighbor ground face. After filtering, floating extensions are removed. 'Floating' refers to point clusters that no longer connect to the original LOD2 model after neighbor filtering. It is still possible that an extension includes (part of) a neighbor extension, if it is directly connected to the main building and not above the neighbor's footprint. This limitation could be remedied if full parcel data is available.

Extension meshing

Meshing the detected extension clusters consists of the four processes shown in Fig. 1 Stage E. Note that we are not interested in a standard mesh from point cloud, e.g. directly triangulating the cluster point cloud, but desire a simplified representation that fits the supplied LOD2 model. Hence, the extension is split into simple planar polygon shapes prior to triangulation.

Find planes and intersections

A standard method for finding planes (or other shapes) in point cloud data, is Random Sample Consensus (RANSAC) [12]. This performs well if only a single plane in the data should be found, but fails if a split of complex data into local planes is desired. A solution to this problem has been implemented by Schnabel *et al.* [13], and is called 'Efficient RANSAC'. This algorithm splits the extension into several plane-like point clouds.

Optionally, the full building cluster can be used for the Efficient RANSAC plane fitting, represented by the dashed arrow in Fig. 1. This ensures that extension planes approximately aligned with an existing wall will also be fitted to match this existing wall, instead of risking skew due to only fitting on the smaller noisy extension data.

For better meshing and texturing results, we should ensure that the discovered planes will, when meshed, connect to their neighboring planes and the LOD2 model. For each plane in the extension, the neighboring planes are found and plane intersections are computed. Similarly, each plane is compared to neighboring planes in the LOD2 model. Splitting the sampled LOD2 model into planes can be done directly from the original mesh, by merging neighboring triangles based on their normals until only non-coplanar polygon faces remain.

Find 2D polygon shapes

Creating a simplified planar mesh from the point cloud requires two more steps after finding RANSAC plane inliers. First, all plane inliers are projected onto the plane, and second, a shape description is desired, to reduce the amount of triangles created by triangulation. The outline of a 2D point cloud can be found using the Alpha Hull algorithm [14]. However, the projection of points on their planes causes some plane clouds to become visually disjoint from other plane clouds, which would introduce gaps during meshing. These gaps can be reduced using the 'generalized alpha shape' [10], with computed plane intersections. Our method is inspired by this generalized alpha shape, though differs slightly, because we maintain a fully point-based implementation of the alpha shape. Fig. 2 portrays the extra steps taken compared to a normal alpha shape. We project points within distance 2α of the plane intersections, resulting in an alpha hull that is guaranteed to touch neighboring alpha hulls on the plane intersections.

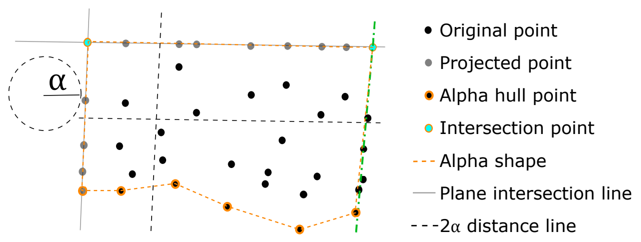


Figure 2. *Example of our implementation of the generalized alpha shape. The circle shows the value of alpha (α) used in this example.*

It may happen that a single plane is described by multiple simple polygons, if said plane contains very large holes. To improve the simplicity of the resulting polygons, any points of a polygon that are not already part of a plane intersection are approximated by a RANSAC line, provided that sufficient inliers can be found, shown by the green line in Fig. 2. This reduces the prevalence of serrated edges in the mesh. Furthermore, to reduce the amount of triangles after triangulation, all non-end points of the set of points on a plane intersection line are removed from the alpha hull. Finally, if an intersection of lines, either RANSAC lines or plane intersection lines, is close to the plane-projected point cloud, this intersection point is added to the alpha hull, as shown in light blue in Fig. 2. This step closes holes in the mesh, primarily around the corners of buildings.

There are many options for triangulating simple polygons, and we choose to use the simple Ear Clipping (EC) [15] method, with an implementation based on FIST [16][17].

Triangular mesh normal correction

An additional challenge is determining the face direction of the newly triangulated planes, because the extension meshes are not watertight, hence there is no inherent definition of 'outside'. All the points in the point cloud have been generated from either terrestrial or aerial images, hence we can use the location of the camera from where a point has been generated, as described in Algorithm 1.

```
Algorithm 1 Algorithm for plane triangle facing correction
Input: triangulated plane, thresholds $T_1$, $T_2$
Output: corrected plane
1: for each triangle do
2:     Compute centroid $C$ and its normalized vector to sensor $s$
3:     Find closest point cloud point to centroid and sensor location generating this point
4:     Compute dot product $d$ of triangle normal $n_{xyz}$ and $s$
5:     if $(d < 0)$ then $n_{xyz} = -n_{xyz}$ end if
6: end for
7: Flip triangles to match orientation of largest $|d|$ and compute average dot product $d_{av}$
8: if $d_{av} < T_1$ and $|n_z| < T_2$ and $n_z < 0$ then Flip all triangle normals end if
9: return Corrected plane
```

Texturing

Once the extended LOD2 model mesh has correct triangle normals facing outward, it can be textured. The texturing is performed by mapping parts of the terrestrial panorama images and aerial images to triangles of the model using texture atlases, and UV-coordinates per vertex. The resulting textures are not ortho-rectified, and have a pre-determined pixel size. Loopy belief propagation is used to select the best textures from multiple cameras, while reducing seams and small occlusions. The full texturing method was developed by the company CycloMedia and has not been published.

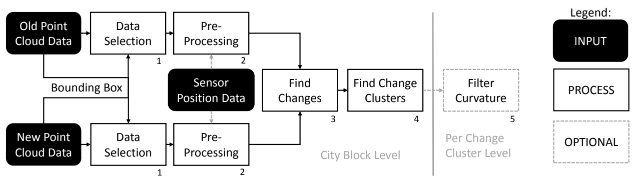


Figure 3. *Flowchart of the change detection algorithm. The numbers denote 'stages'.*

Method 2: Point cloud change detection

Parts of our algorithm for extension detection are also suited for change detection on point clouds. The framework for change detection is shown in Fig. 3, having strong similarities with Fig. 1. Data selection adopts only the points that are present in the bounding boxes of both datasets in each point cloud. Pre-processing entails optional ground removal and optional downsampling. In Stage 3 of Fig. 3, an old point cloud is subtracted from a new one or vice versa, similarly to how the LOD2 model was 'subtracted' from the point cloud earlier.

The resulting differences are often noisy, which is solved by applying Euclidean clustering with minimum cluster size. Executing the change detection both ways (new versus old and vice versa) results in 'added change' and 'removed change' with respect to the old cloud. Various thresholds make this algorithm dependent on point cloud density, and the cloud should thus be downsampled if necessary. Curvature filtering can be applied to remove chaotic changes. Examples of changes that often can be removed by this are trees or irregular bushes, while planar changes such as new buildings or cars, are preserved.

Experimental setup

The LOD2 models for the experiments have been supplied by the Dutch Cadastre, Land Registry and Mapping Agency (Kadaster, Apeldoorn, The Netherlands), and the 3D point clouds by the company CycloMedia (Zaltbommel, The Netherlands), generated from their panoramic and aerial images. For the extension detection, ground truth is absent, hence we have created a synthetic dataset to quantitatively evaluate the extension detection algorithm. Change detection is tested on both 3D point cloud data from images and LiDAR data. The entire framework is implemented in C++, and makes use of the Point Cloud Library [11].

Synthetic dataset generation

The synthetic dataset consists of three parts: (1) synthetic point cloud to resemble the 3D point clouds from CycloMedia (2) synthetic LOD2 model without extensions, created by simplifying a real LOD2 model, and (3) ground-truth building with extensions, for which we use the non-simplified LOD2 model.

The synthetic point cloud is created in several steps:

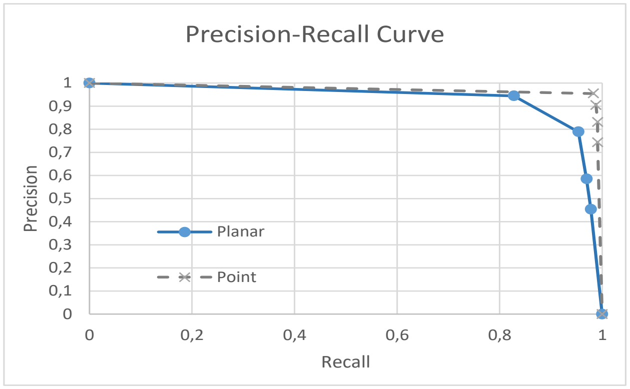


Figure 4. *Synthetic data Precision-Recall curve. 'Planar' refers to recall and precision computed on per-plane basis, 'point' refers to recall and precision computed on per-point basis.*

1. Select an LOD2 building mesh of interest,
2. Collect neighbors (also LOD2 meshes) of the model of interest,
3. Sample all meshes to achieve point clouds with similar density to point clouds from images,
4. Create a randomly sampled plane at ground height,
5. Add Gaussian noise and outliers.

Note that in this synthetic dataset various sources of real-world errors are not present. Some examples of absent sources are: trees and plants touching the building, cars parked very close to a building, or rather noisy points from aerial images. Manually creating synthetic buildings with and without extensions is time consuming. Instead, we apply Quadric Mesh Simplification [18] on the building, which cuts-off multiple extrusions from the building, to create a synthetic extension-less building. The difference between an unsimplified and simplified building is the extension ground truth.

Results

Extension detection and the LOD2 model extension are verified on the synthetic dataset, and textured results are validated by visually comparing 167 3D-textured models, out of which a complicated case is shown in this paper. Change detection on point clouds is evaluated on both point clouds from terrestrial and aerial images, and LiDAR data.

Extension detection on synthetic data

All tests have been performed on a set of 2,793 different LOD2 models, taken from a small city center. Here, most buildings are row houses and flats, although special buildings like churches occur. Generally, we are interested in whether all planes making up the extension are reconstructed, not the individual points. Hence, we define an extra precision (P_r) and recall (R_c) metric, based on the generated planes. A ground-truth plane is considered a false negative, when it is covered for less than 50% by detected extension planes. A detected plane is considered a true positive, if at least 50% of its points are within the distance margin to a ground-truth plane. Determining plane overlap requires a distance threshold, which is set to 50 cm.

Calculating planar P_r and R_c for various minimum extension-cluster size thresholds (100, 200, 400, 800), results in the P_r-R_c curve as shown in Fig. 4. These thresholds give only 4 performance points, because going below 100 or above 800 is empirically not useful. Point-by-point P_r and R_c values are significantly higher than their planar counterparts. Because the data points are found reliably, low planar scores are a result of the plane-fitting part of the algorithm, not the extension detection. The fact that primarily P_r scores are worse, means that many small spurious planes are fitted, which is supported by real-world results.

To determine more about the nature of the errors, the additional metrics shown in Table 1 are computed. The metric 'average squared distance' is a distance measure from each detected point to the nearest ground-truth plane point. When planes are shifted or mis-oriented in space with respect to each other, this metric shows a larger value. The 'weighted average inlier fraction' measures the fraction of points per found extension plane that is within the distance threshold to its ground-truth plane. This is a measure of plane overlap, and equals unity as long as the entire plane is within 50 cm to the ground truth. Finally, 'average planar angle deviation' computes the angle between found plane and ground-truth plane. Table 1 shows increasing errors for growing minimum extension sizes.

Meshing and texturing

Various issues arise when fitting planes and meshing the discovered extensions. Fig. 5a shows several of these issues on a complicated extension.

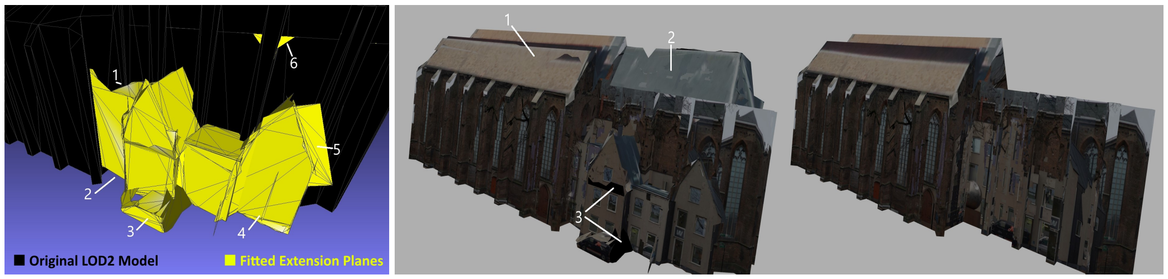
Markers 1 and 2 indicate cases where a plane does not quite extend to the black LOD2 model or ground, leaving a hole in the mesh. This tends to happen wherever point cloud data is especially sparse in a large area, so that the plane intersection points along with the alpha shape can no longer close the hole. Marker 3 shows a simple error caused by Euclidean clustering, where a car has been included in the extension point cloud. Marker 4 shows a common case of a plane fitted on noise. After fitting the front of the building, enough points may remain to fit a plane perpendicular to the wall surface and into the window 'alcove'. This may be partially resolvable by fine-tuning the plane RANSAC-inlier threshold and minimum inlier count. Markers 5 and 6 show cases of plane over-extension, which result from plane intersection computations that are supposed to fill holes.

Textured models reveal some other errors more clearly, as shown in Fig. 5b. Incorrect triangle facing is common, and causes missing textures due to back-face culling, as shown in Fig. 5b Marker 3. Another common issue is a roof that has a slightly different incline than its LOD2 counterpart. The extension will only be detected from the points where the distance between the two becomes too large, at which point-plane intersections between the two will be too far away from the detected plane for the hole to be closed by the alpha shape. This happens in Fig. 5b Marker 1, though the result is still arguably better than the original. Marker 2 shows a large roof discovered solely from aerial points, which has a small triangle missing, due to low density.

Figs. 5b and 5c allow for visual comparison of extended and base textured model. This example is a complicated case, though the resulting extended model represents the average improvement for many buildings. Buildings with extensions connecting to neighbor extensions will generally lead to poor cases, while simple extensions such as dormers, are generally processed without problems.

Metrics for finding extensions on the synthetic point cloud dataset. Table left side: point metrics, right side: plane-based metrics.

Min. cloud size	Point precision	Point recall	Average squared distance	Average inlier fraction	Average plane angle deviation	Planar precision	Planar recall
100	0.955	0.983	0.565	0.949	5.07	0.944	0.828
200	0.905	0.988	2.04	0.894	14.16	0.789	0.954
400	0.831	0.991	3.99	0.813	28.39	0.586	0.969
800	0.743	0.991	4.97	0.789	39.09	0.454	0.978



(a) (b) (c)

Figure 5. *Examples of textured building blocks. The numbers indicate various types of undesirable algorithm behavior. (a) Meshed complicated extension. (b) Extended textured building block. (c) Original textured building block. (Best viewed in color.)*

Change detection

Point-cloud-based change detection is a secondary result of the extension detection algorithm, and thus not the focus of this paper. However, testing has revealed some interesting results that are worth analyzing.

The change detection algorithm is tested on datasets taken from the same area, one year apart. For this, two types of data have been used in tests: point clouds created from multiple terrestrial images, and colored LiDAR point clouds. Aerial data was deliberately not included, due to the large error margin of up to several meters. The result of the change detection on image-based point clouds, with ground removal, is shown in Fig. 6. Markers 1 and 4 show undesirable changes, caused by strong wind and edge effects, respectively. The red points near Marker 2 show a small alley of which no updates were available, hence the algorithms determine that a part of a wall must have been removed here. Marker 3 shows a type of change one likely wants to detect, namely a large block that was placed on the street. Whether these detections are a problem depends on the application. Post-filtering, for example based on curvature, could remove some undesired types of detections. The method is equally well applicable to LiDAR point clouds, provided that they are downsampled first.

Conclusions and discussion

We have described an algorithm for change detection in point clouds, including 3 new contributions: change detection for LOD2 models compared to 3D point clouds, the application of detected changes for creating extended and textured LOD2 models, and change detection between point clouds of different years.

First, the detection of significant differences between a point cloud and LOD2 model has been performed based on various Euclidean distance-based metrics and repeated clustering. This point cloud-based part of the algorithm performs well, although it tends to find more than extensions. It achieves a recall of 0.983 with a precision of 0.955 on a synthetic dataset, with the precision dropping sharply if slightly more recall is desired.

Second, the extension of the LOD2 models using simple planar shapes, based on detected changes, has been investigated. Texturing these extended models yields visually promising results, and provides a clear improvement over directly-textured LOD2 models. However, the models are visually still far from perfect. Additional work is required to achieve fully connected, water-tight textured meshes.

Third, a modified version of the extension detection algorithm is applied for change detection between two point clouds. This algorithm is highly robust to noise, and capable of finding large, significant changes.

The methods used also contain several notable limitations. First, the synthetic dataset applied to evaluate the extension point cloud finding algorithm lacks many sources of real-world artifacts (vegetation, vehicles etc.), so that the precision results are too optimistic. A true quantitative measurement of the effectiveness of the algorithm requires LOD2 models with and without extensions, which are not freely available. Second, ground removal assumes approximately horizontal ground, which can be remedied by fitting a ground plane.

Future work should focus on two aspects. First, finding a better discrimination between extensions of the current building and neighboring buildings, so that any extension that is detected in the point cloud is guaranteed to be correct. Second, water-tightening post-processing techniques could be applied to allow for applications of the extended LOD2 model, as if it were a normal LOD2 model.

Acknowledgements

The authors would like to thank CycloMedia for providing high-quality terrestrial and aerial images, as well as the cadastral LOD2 models.

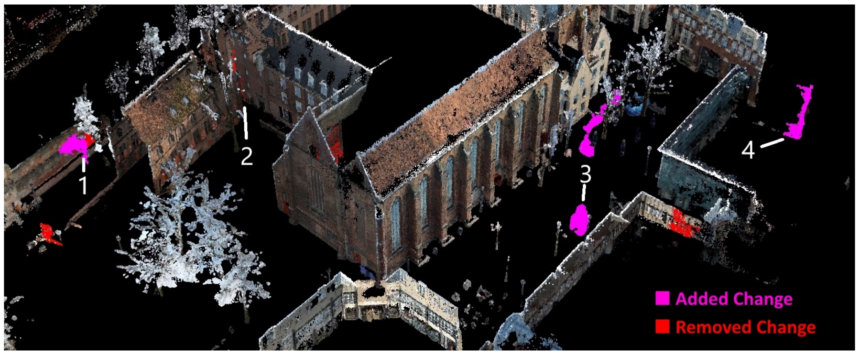


Figure 6. *Change detection result on a point cloud, where the ground has been removed.*

References

[1] G. Gröger, T. Kolbe, C. Nagel, and K.-H. Häfele, "OGC City Geography Markup Language (CityGML) Encoding Standard," *Ogc*, pp. 11 – 12, 2012.

[2] R. Qin, "Change detection on LOD 2 building models with very high resolution spaceborne stereo imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 96, pp. 179–192, 2014.

[3] 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*, vol. 36, no. 3, p. W19, 2005.

[4] F. Pédrinis, M. Morel, and G. Gesquière, "Change Detection of Cities," in *3D Geoinformation Science: The Selected Papers of the 3D GeoInfo 2014*, M. Breunig, M. Al-Doori, E. Butwilowski, P. V. Kuper, J. Benner, and K. H. Haefele, Eds. Cham: Springer International Publishing, 2015, pp. 123–139.

[5] R. Qin and A. Gruen, "3D change detection at street level using mobile laser scanning point clouds and terrestrial images," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 90, pp. 23–35, 2014.

[6] A. Taneja, L. Ballan, and M. Pollefeys, "Image based detection of geometric changes in urban environments," *2011 International Conference on Computer Vision*, pp. 2336–2343, 2011.

[7] a. K. Aijazi, P. Checchin, and L. Trassoudaine, "Detecting and updating changes in Lidar point clouds for automatic 3D urban cartography," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-5/W2, no. November, pp. 7–12, 2013.

[8] A. Taneja, L. Ballan, and M. Pollefeys, "City-scale change detection in cadastral 3D models using images," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 113–120, 2013.

[9] R. Qin, J. Tian, and P. Reinartz, "3D change detection Approaches and applications," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 122, pp. 41–56, 2016.

[10] T. van Lankveld, "Large Scale Shape Reconstruction from Urban Point Clouds," Ph.D. dissertation, Utrecht University, Netherlands, 2013.

[11] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," *KI - Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

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

[14] H. Edelsdrunner and D. G. Kirkpatric, "On the shape of a set of points in the plane," *IEEE Transactions On Inforamtion Theory*, vol. 29, pp. 551–559, 1983.

[15] H. ElGindy, H. Everett, and G. Toussaint, "Slicing an ear using prune-and-search," *Pattern Recognition Letters*, vol. 14, no. 9, pp. 719–722, 1993.

[16] M. Held, *FIST: Fast Industrial-Strength Triangulation of Polygons*, 2001, vol. 30, no. 4.

[17] "Earcut Github Repository," 2016. [Online]. Available: https://github.com/mapbox/earcut.hpp

[18] S. Forstmann and C. Rorden, "Fast Quadric Mesh Simplification," 2015. [Online]. Available: https://github.com/sp4cerat/Fast-Quadric-Mesh-Simplification

Author Biography

Sander Klomp received both his BS and MS from the Eindhoven University of Technology (2016,2018) with the designation Cum Laude. He is now pursuing a PhD degree at TU/E in collaboration with ViNotion, with a focus on efficient deep learning algorithms.

Thijs van Lankveld received his MSc in knowledge engineering from the University of Maastricht (2007) and his PhD in computational geometry from Utrecht University (2012). After working at INRIA - Sophia Antipolis and Geometry Factory, he now works at Cyclomedia Technology. His work has focused on 3D scene processing; specifically point clouds, shapes, surfaces, textured meshes, and scene projections.

Bastiaan Boom graduated at the Free University of Amsterdam in Computer Science in 2005. He received his PhD at the University of Twente (2010) in Electrical Engineering, where his thesis was entitled Face recognition's grand challenge: uncontrolled conditions under control. From 2011 until 2015 he worked as postdoc at University of Edinburgh on the Fish4Knowledge project. Currently within Cyclomedia, he is cluster lead of the group where they develop computer vision and machine learning algorithms.

Peter de With is Full Professor of the Video Coding and Architectures group in the Department of Electrical Engineering at Eindhoven University of Technology, which he leads since 2000. De With is a Fellow of the IEEE, has (co-)authored over 400 papers on video coding, analysis, architectures, 3D processing and their realization and has received multiple papers awards. He is a program committee member of the IEEE CES and ICIP and holds some 30 patents.

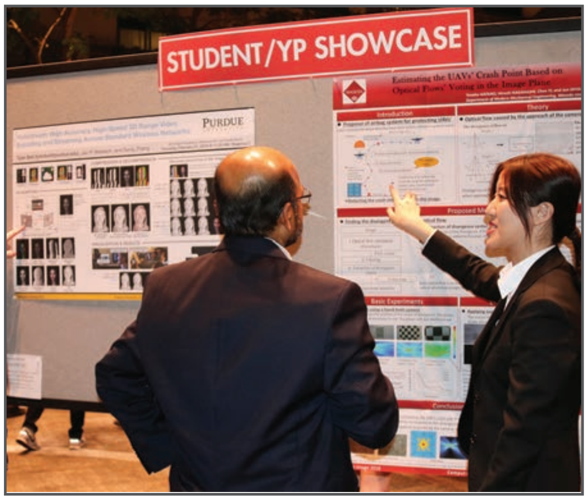
JOIN US AT THE NEXT EI!

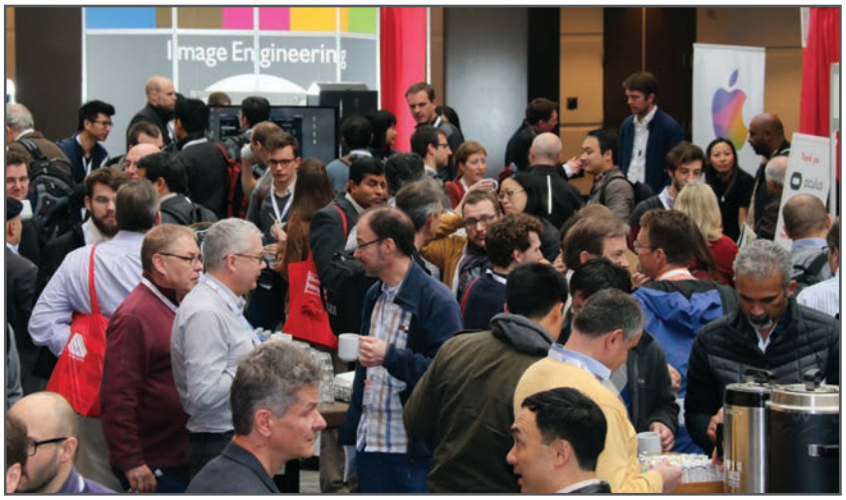
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





AR is a fundamental and transformative interface and platform.
20+ years from my initial working demo to today's AR displays
Optimistic that it won't take another 20 to reach ubiquitous consumer AR!
My first working AR prototype
Today's platforms
Ubiquitous consumer AR

• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org