Point Cloud Data Cleaning and Refining for 3D As-Built Modeling of Built Infrastructure

Abbas RASHIDI[1] and Ioannis BRILAKIS[2]

[1]Visiting Assistant Professor, College of Engineering and Information Technology, Georgia Southern University, Email: arashidi@georgiasouthern.edu

[2]Laing O'Rourke Lecturer, Department of Engineering, University of Cambridge, UK, Email: ib340@cam.ac.uk

ABSTRACT

Spatial sensing of built infrastructure is now a common practice within the AEC industry and results are commonly encapsulated in the form of dense Point Cloud Data (PCD). PCD of built infrastructure might consist of millions of spatial points and it is well known that processing all these points is neither necessary nor computationally feasible. In addition, due to several reasons including hardware and/or software deficiencies, there might be several outliers that need to be removed from the PCD before further processing. As the result, cleaning and refining PCD is a paramount step in the process of spatial sensing and object-oriented modeling of built infrastructure scenes.
This research work entails two parts: The first part provides an in-depth literature review on current states of practice and research on the concept of PCD cleaning. The second part presents the authors' suggested framework for cleaning and refining PCD of built infrastructure. This prototype mainly consists of three major components: 1) removing outliers; 2) filling holes and gaps on surfaces of PCD and 3) balancing the density of different areas of PCD based on a plane recognition approach. Several case studies are presented to demonstrate the efficiency of the proposed framework.

KEYWORDS: Point Cloud Data, 3D edge, Filling holes, Removing outliers, Built infrastructure

1. INTRODUCTION

Generating BIM models of built infrastructure during the design phase of projects is becoming a standard practice within the AEC industry. The as-design BIM models are based on initial design parameters and assumptions and in several cases, there are deviations between the designed project and the final built product. As the result, in most cases, it is necessary to generate the as-built BIM models of built infrastructure as well. The as-built BIM models can also be used as the technical archive of final project by facility managers.
The common state-of-the practice process for generating as built models of built infrastructure consists of two major stages:
The first step is spatial sensing of the built environment using either active (e.g. laser scanners) or passive (e.g. photo/videogrammetry) sensors. The outcome of this stage is encapsulated in the form of thousands or millions of spatial points, also known as Point

Cloud Data (PCD). The second step is processing the generated PCD and converting it into an informative, object-oriented 3D model.
While the first step (spatial sensing and PCD generation) could be handled automatically, the second stage is mainly manual and operators need to spend significant amounts of time and efforts to convert a raw PCD to an informative 3D model. To tackle this issue, developing algorithms for automatically processing PCD, extracting 3D objects and converting them into a BIM model have been an active field of research within the last decade.
The other issue with generating as-built BIM models of built environment is that generated PCD are not always ready for processing and modeling practices in their raw format. Depending on the type of sensors (i.e. active versus passive sensors), there might be several outliers, poorly scanned areas, gaps and holes within raw PCD. The size of PCD is another issue since scanning a simple residential building might consists of 10-100 million points. It is not necessary – and feasible- to process all those points and a down-sampling stage would be necessary to reduce the size of the PCD and prepare it for the next modeling stages. That being said, a post processing step, for cleaning and handling raw PCD and converting it to an acceptable format, ready for further modeling steps is necessary (Figure 1).

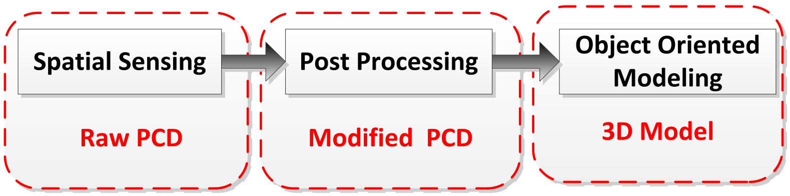

Figure 1: The process of generating 3D models of built infrastructure

This paper first provides a thorough literature review on common states of practice and research for PCD cleaning and modifying tools. As the next step, the proposed PCD cleaning sets of algorithms, devised by the authors for dealing with raw PCD of built infrastructure scenes are presented and validated. The proposed PCD cleaning tools address three major issues within built infrastructure PCD: Removing outliers, filling holes and balancing the point density at various locations. The validation procedure contains various datasets collected from real built infrastructure scenes.

2. POINT CLOUD DATA CLEANING: LITERATURE REVIEW

Manually modifying PCD is a common practice within engineering fields and a number of commercial software packages already exist in the market (Polyworks PIFEdit, Pix4D, SAFE). These software packages are all heavily based on user interaction and provide a range of modifying tools including visual removal of outliers, manual segmentation of points into clusters and adding points into sparse areas. Manually processing PCD is extremely beneficial; however when it comes to PCD of civil infrastructure objects, considering the size and complexity of the generated PCD, the process would be tedious and labor intensive. Researchers have attempted to address this issue by proposing semi or fully automatic algorithms for modifying various aspects of PCD. Teutsch et al (2011) proposed a clustering algorithm for subset segmentation and outlier removal of PCD. The suggested algorithm

performs on the basis of k-d tree data structure as the efficient search structure among millions of points. They also considered outliers as single or sets of few points that have certain distance with the neighborhood points. Köhler et al (2004) suggested an outlier removal algorithm for PCD acquired by using structured light. In the proposed method, they utilized 2D information of points which can be provided by calibrating the scanning devices.
He et al (2004) considered the outlier removal issue as an optimization problem and suggested a local search heuristic based algorithm to find optimal solutions.
In order to dealing with the complexities involved in the process of outlier detection, some authors have developed more advanced statistical models. Shen et al (2009) employed kd-tree to manage the outlier removal from airborne LiDAR data. Their approach is based on the elimination of the obvious low and high outliers using the elevation histogram analysis. Under their algorithm, for each point, the average of the distances between the central point and its k-neighborhood points are calculated. If the average distance is larger than an adaptively preset value, the point is regarded as an outlier.
Filling holes on surfaces of PCD is another important component of a robust PCD cleaning algorithm. It is well known that PCD generated by using spatial sensing technologies are typically incomplete and the surfaces might contain several undesired holes, artifacts or poorly-reconstructed areas. To tackle the issue, several hole filling algorithms have been suggested by researchers. One simple, yet widely employed technique is to triangulate connected components of the boundary points of the hole and then embedding the hole with a patch which has the topology of a disk (Curless and Levoy, 1996). For more complicated topologies, more advanced techniques are required. These techniques can be divided into multiple groups. From perspective of representation of the surfaces, there are two major categories of algorithms:
i) *Embedding holes using explicit surfaces:* Explicit methods represent the surface location and geometry in an explicit manner. This category can be classified into several sub-categories including computational geometry (Edelsbrunner and Mucke 1994; Edelsbrunner 1998) and parametric surface approaches (Rogers 2003), which usually utilize Voronoi diagrams and Delaunay triangulations to identify connections between various 3D points and consequently reconstructing triangulated patches by connecting adjacent vertices.
ii) *Fitting holes using implicit surfaces:* Generally speaking, reconstruction of an implicit surface is equal to segmenting into two interior and exterior regions. As the result, implicit surface reconstruction algorithms typically fit surfaces as a co-dimension of a scalar-valued function. The global implicit surface representations as well as the incorporation of volumetric or prior information make the implicit surface approach more efficient in dealing with noisy, general datasets. The implicit function the scalar function could be constructed either on a grid (Zhao et al. 2001; Lempitsky and Boykov 2007) or without a grid and by using mesh-free approximation functions such as the poly-harmonic radial basis interpolating function (Morse et al. 2001; Wendland 2002).
In terms of the required data for embedding holes, different algorithms are classified into three categories:
iii) *Mesh- based hole filling algorithms:* According to Davis et. al, mesh-based algorithms are classified as the algorithms which treat the set of points as an ordered 2D

array of range samples (with known depth). These samples therefore can be triangulated to shape a polygon meth.

iv) *Point Cloud based hole filling algorithms:* Unlike the mesh-based algorithms, point cloud based algorithms are implemented on unorganized sets of 3D points without any topological prior information. This category of algorithms treats the hole area as a set of unorganized 3D points to be fit with a continuous surface (Davis et.al, Vang and Oliviera).

The recent category, also known as region growing techniques, usually begins with a seed triangle, then targeting a new point and adding it to the existing region boundary and finally, repeating the procedure until all candidate points have been considered ((Bernardini 99).

While the presented algorithms perform well on smooth surfaces, they are not able to reconstruct the true shape of the objects in areas close to sharp edges and corners. Sharp edges and corners are very common elements of built infrastructure PCD. To handle this problem, a number of algorithms for filling holes while restoring sharp features have been proposed (Chen and Cheng, 2005; Li et al, 2010).

3. THE PROPOSED POINT CLOUD DATA CLEANING ALGORITHM FOR BUILT INFRASTRUCTURE SCENES

As shown in Ffigure 2, the proposed PCD cleaning algorithm in this research mainly consists of three stages:

1. Removing outliers 2. Filling hoes and gaps 3. Balancing the density of different parts of the point cloud using a plane recognition approach. Details of each stage are summaries below:

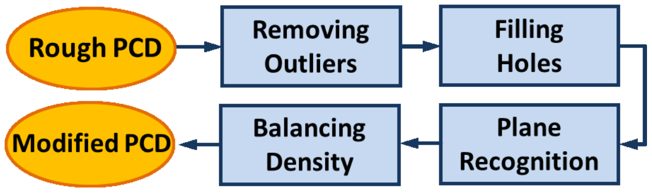

Figure 2: Overview of the proposed algorithm for point cloud data cleaning

3.1. Removing outliers:

Removing outliers is the primitive step within the PCD cleaning pipeline. Outliers or unwanted points mainly include unattached points into surfaces of objects and might occur due to several reasons including multiple reflections on shiny surfaces during laser scanning of built infrastructure scenes or possible errors in implementing Bundle Adjustment algorithm for image/video based PCD generator pipelines (Rashidi et al, 2015).

As discussed earlier, several criteria have been suggested for identifying outliers; three of which are the most popular:

i) *Distance-to-plane factor:* Deviation of the point from a manifold generated by the neighboring points. An isolated point while its neighbors approximate a surface is likely an outlier. Determining an efficient "point to plane distance" threshold will properly handle this criterion. Under this criterion, a point is considered as an outlier, if its distance from the plane approximated by its k nearest neighbor points exceeds a certain threshold (Figure 3, a).

ii) *Spherical distance factor:* A point which is located somewhere in space far from a cluster formed by its k neighboring points is appeared to be an outlier. In other words, the distance of an inlier point from a sphere built by the k-nearest neighbor points should not exceed a certain threshold. This maximum allowable distance is a factor of the sphere's diameter (d) (Figure 3, b).

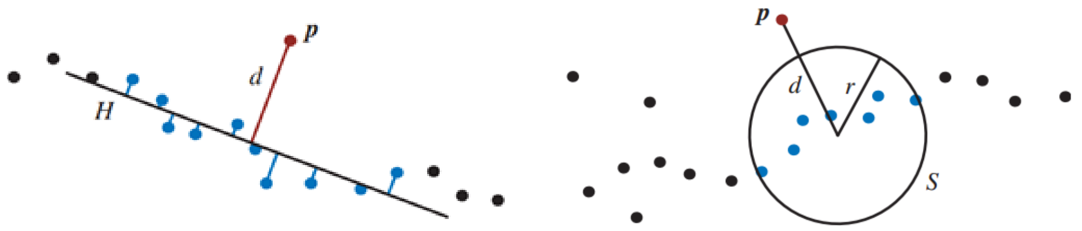

Figure 3: a) The distance (d) of an outlier (p) from a surface generated by its neighbors (H) exceeds a certain threshold. The k- nearest neighbors are highlighted in blue and the H surface is generated based on minimizing the squared distances of k-nearest neighbor points (min $\sum distance\ (p, H)^2$). Distance values (d) should be normalized to handle possible noise and surface deviations.
b) Spherical distance factor: the outlier p is located farm from sphere S

iii) *K nearest-neighbor reciprocity factor:* In a well sampled vicinity, an inlier point q might be among the K nearest neighbor of an outlier p but an outlier p is most likely not within the k nearest neighbor points of an inlier q. A directed graph of nearest neighbor relations can handle this criterion. More information could be found at:

3.2. Filling holes on surfaces of PCD

In this research, an edge detection approach for filling holes on surfaces of PCD is proposed. This approach consists of the following steps (Figure 4):

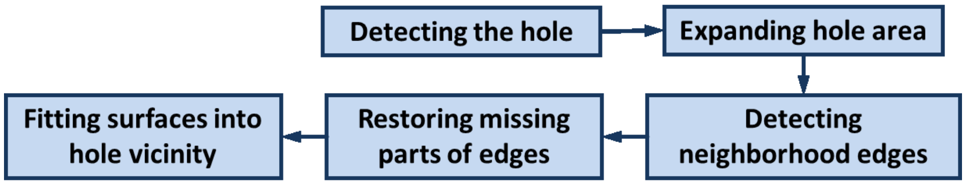
Detecting the hole
Expanding hole area
Detecting neighborhood edges
Restoring missing parts of edges
Fitting surfaces into hole vicinity

Figure 4: Overview of the proposed algorithm for point cloud data cleaning

A brief explanation about each stage is outlined below:

i) Detection of holes and expansion of the hole area:

Automated detection of holes on surfaces of PCD is an ill-posed problem since it is not possible to automatically differentiate between natural holes on objects (such as an opening on a wall or the hole along a pipeline) and undesired holes appeared on surfaces of PCD. Furthermore, the size of holes needed to be covered is a serious concern. On one hand, very small holes do not play an important role during processing stages and can be ignored; on the other hand, it is not possible to fill very large gaps and non-reconstructed areas merely by

implementing post processing techniques. In this research, we refer to Wang and Oliveira (2007) for the description of a hole on the surface of PCD. According to their definition, a hole consists of a loop of boundary edges. By definition, a boundary edge only belongs to one triangle in the mesh, while a shared edge is shared between two triangles.
After detecting the hole, the next step is expanding the hole area to include possible surrounding sharp edges. In this research, the area within a certain distance from the centroid point of the hole vicinity is considered as the expanded hole area and the search for detecting sharp edges would happen within this area.

ii) Detection of sharp edges and restoring missing parts of those edges:

Sharp edges are defined as the intersection lines between two different surfaces or a corner where two or more surfaces are connected. Sharp edges can be recognized by looking for abrupt changes in surfaces orientations. As the result, we need to search for 3D points with significant changes in the directions their normal vectors compared to the neighboring points.
In this research, we implemented the sharp feature detection algorithm suggested by Walsh et al. (2013). Their approach is mainly based upon clustering a Gauss map on local neighborhoods. A summary of the above mentioned approach is presented here. More details could be found at Walsh et al. (2013):
The first step of the sharp feature detection is to select a local neighborhood of points around each point. In this work, the common "k nearest neighbors" algorithm is employed. The neighborhood is defined as the *k* closest points and is defined adaptively by the method suggested by (Hoppe et al., 1992). It is necessary to mention that the local surface properties related to each 3D point are highly reliant on the size of the neighborhood *k*. A larger size of *k* is ideal for noisy data points to avoid the negative influence of the noise on the sharp feature selection. In contrast, smaller neighborhoods ensure that estimates reflect local surface properties and minimize computational expenses. To handle this situation, Hoppe et. al suggested to set the initial value of k as minimum. As the next step, the size of the neighborhood for each point would increase until reaching the maximum possible value. The maximum value of k is achievable once the eigenvalues of the covariance matrix for the neighboring points become distinct. By following this strategy we can ensure that the neighborhood is large enough to deal with noisy data, yet small enough to predict the local properties.
After determining the local neighborhood for each point, the next step is to form a Gauss map for each point. First, all possible *k(k–1)* triangulations of the chosen point *p* with its neighborhood points are generated. The normal vector of each triangle is then computed and mapped onto a unit sphere centered at *p*. Once the Gauss map is formed, each point undergoes a flatness test to determine whether the examined point fits into a smooth surface or a sharp edge:
The basic idea is that in ideal situation and for completely flat surfaces, the computed normal vectors mapped into the unit sphere are all almost parallel. On the other hand, for points located on sharp features, there would be an angle between those lines. By calculating the angles between every normal vector, we can run the flatness test. If the angle between these two angles is lower than a predefined threshold, then the two vectors belongs to the same cluster. The clustering process is repeated until we achieve the minimum number of clusters.

Upon completion of the clustering process, all clusters including few points are disregarded (these points are considered as noisy data). Then we count the number of clusters for each point. If there is only one cluster left, the point is located on a flat surface; otherwise, in the cases that the number of remaining clusters is more than one, the point is located on a sharp edge. More details regarding detection of sharp features on PCD could be found at (Walsh et al. 2013; and Weber et al., 2010). Finally, after detecting the existing 3D edges on the extended area, the next step is to restore the missing parts of the edges possibly located on the surface of the hole. To achieve this goal, we simply extrapolate each sharp edge using a RANSAC algorithm and insert extra points on missing parts.

iii) Fitting proper surfaces into the hole area:
The last stage of the proposed hole filling algorithm is to cover the hole vicinity using a proper surface. As explained in the literature review section, different approaches have been suggested for this purpose. In this research work, we followed the standard ball pivoting algorithm (BPA) suggested by Bernardiny et al. (1999). The reason behind choosing BPA is its robustness in terms of dealing with noisy data; simplicity in terms of the concept and implementation; and efficiency in terms of computational demands (Figure 5).

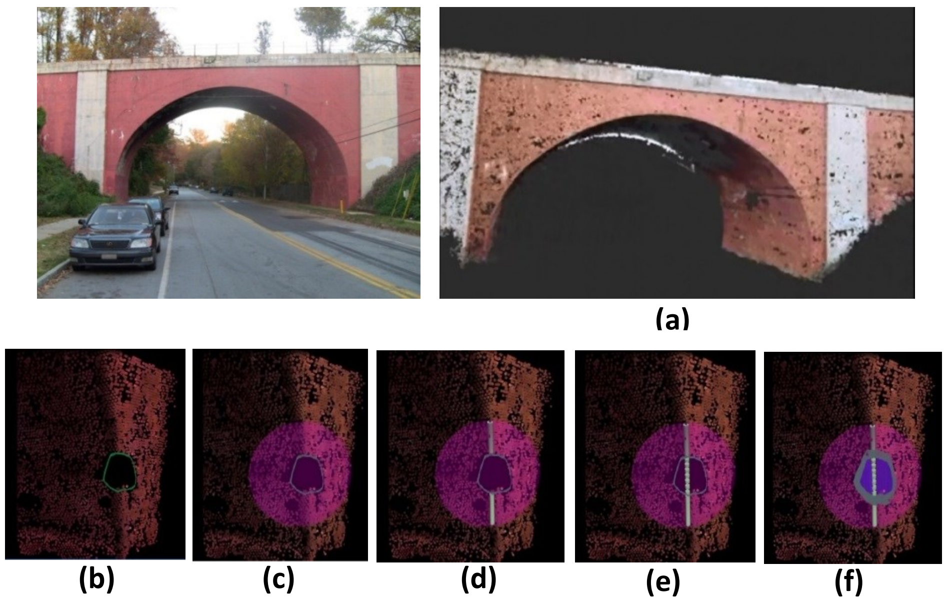

Figure 5: Implementing the proposed hole filling algorithm on a bridge PCD

3.3. Balancing density of point clouds

One mutual problem with most spatial sensing devices is that the captured 3D points are not equally distributed. This phenomena might be the result of several factors including possible arbitrary movements of camera while taking videos and images (in the case of using photo/videogrammetric techniques for spatial sensing) or changes in the default density of PCD during multiple scans (in the case of using LiDAR technologies). Despite the importance of balancing the density of 3D points on different areas of PCD, it has been neglected by most of exiting PCD cleaning algorithms. Balancing density of points on surfaces of PCD is also required due to other reasons:

1- Considering the large scales of civil infrastructure, the generated PCD usually consist of millions of 3D points. It is not computationally feasible to process all those 3D points. As the result, it is desired to run a preliminary processing algorithm to reduce the number of 3D points on different dense surfaces to an optimum level and by following a down sampling procedure.
2- Sparse or poorly reconstructed surfaces of PCD might not be appropriate for further 3D modeling stages.
3- From perspective of visualization purposes, uniform PCD with balanced density of points on different surfaces are more pleasant.

Civil infrastructure objects incorporate multiple flat or curvy surfaces. That being said, an efficient approach for balancing the density of points at different parts of civil infrastructure PCD is through a robust algorithm for recognizing different surfaces and then inserting/removing 3D points by implementing an up/ down sampling process. The above mentioned approach has been incorporated in this research work. To this end, we implemented the plane recognition algorithm suggested by Zhang et.al, (2015) and then balanced the point density of different surfaces by removing/inserting 3D points following a simple up/down sampling approach (Figure6). Adaptive thresholds need to be considered for identifying desired densities.

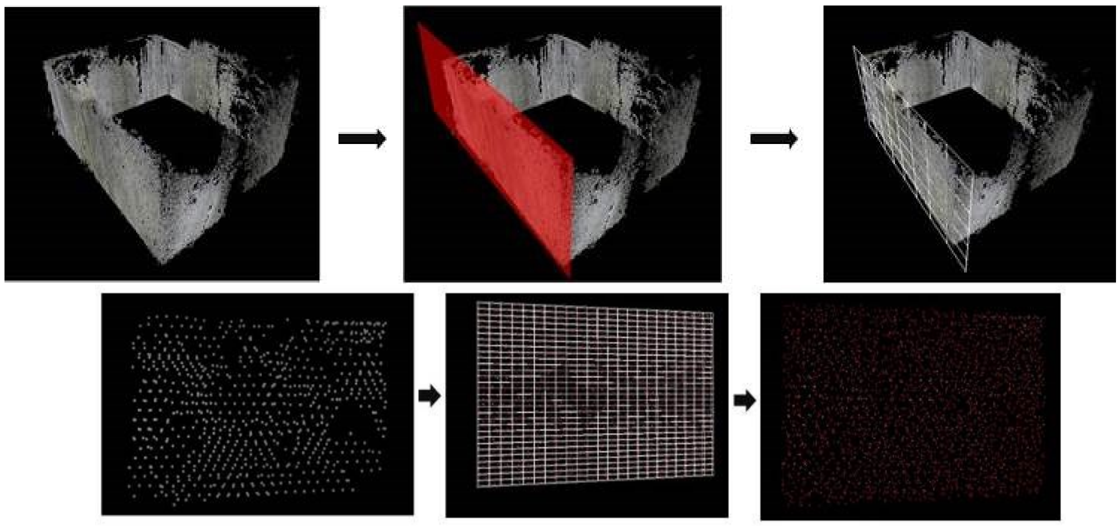

Figure 6: Different stages for balancing densities on surfaces of PCD

7. IMPLEMENTATION, EXPERIMENTAL SETUP AND RESULTS

A C# based prototype was implemented to test the validity of the proposed algorithm. It was written in Visual Studio 2010 using Windows Presentation Foundation (WPF) and publicly available libraries such as OpenCV 2.0 (wrapped by EmguCV) for access to computer vision tools and DirectX 10 for the graphical display of results. Ten different built infrastructure settings were selected and their corresponding PCD were generated by capturing and processing video streams (Rashidi et al 2015). Proper thresholds for various stages of algorithms were determined based on the experimental results.

For outlier detection algorithm, the results were just evaluated visually. A comprehensive validation, including calculations of True Positive, True Negative, False Positive, and False Negative values will take place as the next step of the research (Figure 7).

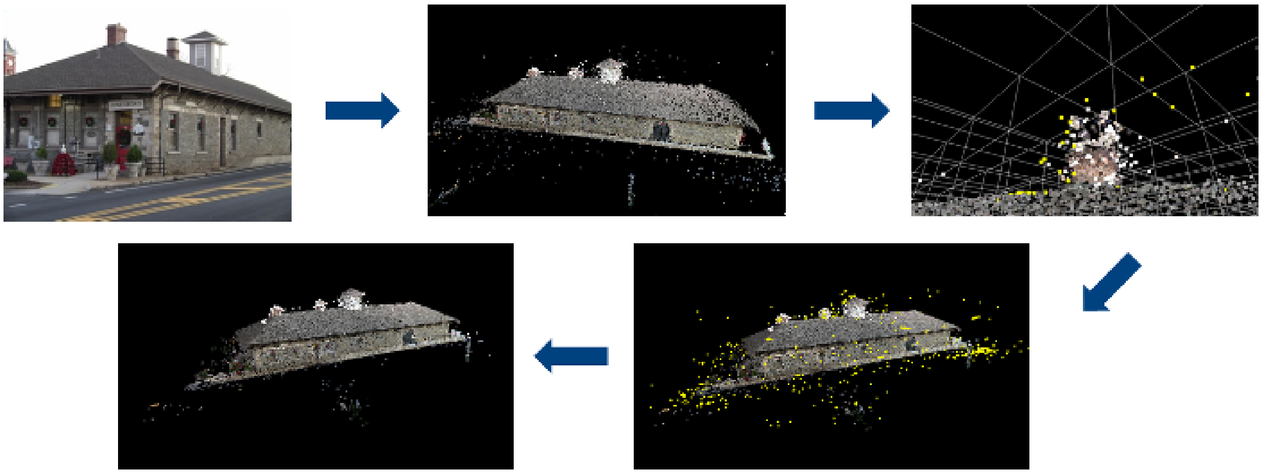

Figure 7: Detecting and removing outliers from PCD of a building as a case study

In order to validate the performance of the PCD density modifying and hole filling algorithms, other than visual evaluation, proper ground truth data was generating manually and obtained results were compared with the ground truth and the errors were computed. Part of obtained results is summarized in table 1.

Table 3: Summary of the average results obtained by implementing the presented algorithms on ten case studies

Outlier detection algorithm		**Filling holes algorithm**	
Total number of points in PCD	234,681	Maximum percentage of error (%)	11.8
Number of detected outliers	6,076	Average percentage of error (%)	5.23
Balancing density algorithm			
Average number of detected surfaces within the PCD	17	Number of surfaces under desired density	4
Number of added/removed points	8,604	Number of surfaces under desired density	8

8. SUMMARY AND CONCLUSION REMARKS

PCD cleaning and modifying is a vital post-processing step after spatial sensing of built infrastructure and prior to further 3D modeling steps. In this paper, efficient algorithms for dealing with three common issues with the PCD of built infrastructure objects were presented. The performances of the presented algorithms were evaluated using several case studies from real built environment settings. In addition to the presented case studies, future experiments will also be required to quantitatively measure the accuracy of the presented algorithms especially for the case of outlier removal. Developing robust algorithms for automatically recognizing 3D objects throughout the built infrastructure PCD and therefore enhancing the object oriented modeling stage is another possible direction for future research.

REFERENCES

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G., (1999) ‘‘The Ball-Pivoting Algorithm for Surface Reconstruction,’’ IEEE Transactions on Visualization and Computer Graphics,

Chen C.Y. and Cheng, K.Y. (2005), "A sharpness dependent filter for mesh smoothing," Computer Aided Geometric Design, 22(5), 376-391,

Davis , R. Marschner, Garr, Levoy, (2008), “Filling Holes in Complex Surfaces using Volumetric Diffusion”,

EDELSBRUNNER, H. (1998). “Shape reconstruction with delaunay complex. Springer-Verlag”, Proc. of LATIN98: Theoretical Informatics 1380, 119–132

He, Z., Xu, X., and Deng, S., (2008) “An Optimization Model for Outlier Detection in Categorical Data”

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. & Stuetzle, W. (1992), Surface reconstruction from unorganized points, SIGGRAPH ‘92 Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, ACM New York City, New York, 26(2), 71–8.

Köhler, J., Nöll, T., Reis, G., and Stricker, D., (2012). “Robust Outlier Removal from Point Clouds Acquired with Structured Light, EUROGRAPHICS 2012 Conference.

Li, Z., Meek, D.S. and Walton, D.J. (2010) "Polynomial blending in a mesh hole-filling application," Computer-Aided Design, 42, pp. 340-349, 2010.

MORSE, B. S., YOO, T. S., C HEN, D. T., R HEINGANS, P., AND SUBRAMANIAN, K. R. (2001). “Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions”. IEEE Computer Society Press, 89–98

Rashidi, A., Brilakis, I., and Vela, P., (2015) “Generating Absolute-Scale Point Cloud Data of Built Infrastructure Scenes Using a Monocular Camera Setting” Journal of Computing in Civil Engineering, in press

Rashidi, A., Dai, F., Brilakis, I., and Vela, P., (2013) “Optimized Selection of Key Frames for Monocular Videogrammetric Surveying of Civil Infrastructure”, Journal of Advanced Engineering Informatics

Shen, J., Liu, J., Zhao, R., and Lin, X., (2012). “A Kd-tree-based OutlierDetection Method for Airborne LiDAR Point Clouds”

Teutsch, C., Trostmann, E., and Berndt, D., (2010). “A parallel point cloud clustering algorithm for subset segmentation and outlier detection.”

WENDLAND, H. (2002). “Fast evaluation of radial basis functions: Methods based on partition of unity. In Approximation Theory X: Wavelets, Splines, and Applications”. Vanderbilt University Press, 473–483

ZHAO, H. K., O SHER, S., AND FEDKIW, R. (2001). “Fast surface reconstruction using the level set method”. Proceedings of IEEE workshop, 194–201