OCTREE-BASED, AUTOMATIC BUILDING FAÇADE GENERATION FROM LIDAR DATA

Linh Truong-Hong[(1)] and Debra F. Laefer[(2)*]

[(1)] Research Fellow, Urban Modelling Group (UMG), School of Civil, Structural and Environmental Engineering (SCSEE), University College Dublin (UCD), Newstead G27, Belfield, Dublin 4, Ireland. Email: linh.truong.hong@ucd.ie

[(2)*] Associate Professor, UMG, SCSEE, UCD, Newstead G25, Belfield, Dublin 4, Ireland. Email: debra.laefer@ucd.ie, corresponding author

ABSTRACT:

This paper introduces a new, octree-based algorithm to assist in the automated conversion of laser scanning point cloud data into solid models appropriate for computational analysis. The focus of the work is for typical, urban, vernacular structures to assist in better damage prediction prior to tunnelling. The proposed FacadeVoxel algorithm automatically detects boundaries of building façades and their openings. Next, it checks and automatically fills unintentional occlusions. The proposed method produced robust and efficient reconstructions of building models from various data densities. When compared to measured drawings, the reconstructed building models were good agreement, with only 1% relative errors in overall dimensions and 3% errors in openings. In addition, the proposed algorithm was significantly faster than other automatic approaches without compromising accuracy.

KEYWORDS: Light Detection and Ranging (LiDAR), Terrestrial Laser Scanning, Masonry Buildings, Geometric Modelling, Computational Modelling, Finite Element Modelling

1. INTRODUCTION

Laser scanning, also known as Light Detection and Ranging (LiDAR), rapidly and accurately acquires topography of object surfaces and generates a set of data known as a point cloud. Laser scanning is gaining popularity in reverse engineering [1-3] and visualization [4, 5], yet exploiting the data for structural analysis remains a topic for development [6, 7]. Developing automatic, robust, and efficient methods to reconstruct geometric building models for computational modelling has the potential to save money and time for projects needing city-scale modelling.

Computational models are especially important in structural engineering, when assessing the status of existing buildings or any risks related to adjacent construction works. For tunnelling projects this can be highly problematic, because the measured drawings used as the basis for the solid models needed for Finite Element Modelling (FEM) are rarely available for the vast majority of structures that comprise the architectural fabric of many historic cities. Overcoming this data gap has traditionally required the manual surveying of each structure. Along a single tunnel route, this may include hundreds of structures for each kilometre of tunnelling, thereby making the financial and temporal requirements of such surveying cost-prohibitive. For example, a study of the area along the first kilometre of the upcoming Dublin Metro which included 449 buildings potentially at risk [8]; mostly small (2-4 storey), slender, load-bearing, rectangular masonry buildings with regular features and little, if any, ornamentation [9]. Of 449 buildings, measured drawings of some type (not necessarily complete) were available for only 27.6% of the buildings. These types of buildings are extremely vulnerable to tunnel-induced subsidence damage. During Dublin's last tunnelling project, 1 in 8 buildings along the route were damaged at the cost of nearly €5 million [10]. As such, the aims of this paper are to introduce an efficient and reliable approach for

reconstructing the accurate geometry from LiDAR data of the load-bearing masonry buildings in a form that is compatible with FEM analysis. The proposed method can automatically eliminate unrealistic holes due to incomplete data or occlusions. The derived building models were benchmarked against a selection of other currently available solutions.

Reconstruction often involves two main steps: segmentation and feature reconstruction. Segmentation extracts point clouds on planar features or of the facade objects in order to remove irrelevant points. The remaining portion of the point cloud is then used to reconstruct building models. In this paper, the segmentation step was conducted by employing software associated with the LiDAR scanner, whereas the building reconstruction with the necessary feature detection is the focus of the contribution herein.

2. RELATED WORKS

To date, several approaches have been developed to semi-automatically (e.g. [11]) or automatically (e.g. [12, 13]) reconstruct the building geometry from LIDAR data [both airborne (ALS) and terrestrial (TLS)] and is sometimes combined with photogrammetry. The processes can be divided into either model-driven or data-driven approach. In the former, geometric primitives are initially described and building features are subsequently fitted based on point cloud data [14, 15]. This technique has limited applicability to complex buildings and results in relatively low geometric accuracy, thereby preventing easy usage in computational models. With the data-driven technique, building boundaries and features are generated directly from the point cloud [13, 16, 17]. Such methods can be applied to arbitrary shapes by fitting polygons or surfaces, as is often done in reconstructing building roofs [11, 18]; however, this approach is more sensitive to noise from the input data than the model-

driven approach. Additionally, there are some interactive tools to support users in the fast reconstruction of building models (e.g. [19, 20]).

Amongst data-driven techniques, outlines of buildings and their features (e.g. doors and windows) are commonly generated from points lying on a dataset's boundaries. To assist in this end, Pu and Vosselman [21] identified points on boundaries of building features as the end points of triangles in triangulated irregular networks (TINs), which have lengths exceeding a specified threshold. The boundary points of each feature are categorized into upper, lower, left and right groups, and a minimum-bounding rectangle was subsequently fitted to each feature. However, for the overall façade, an upper boundary line was generated from upper contour points by least-squares fitting, and the extreme vertices of the upper line were projected onto the ground plane to create two vertical boundary lines [11]. In related research, Boulaassal et al. [22] segmented points on planes by the RANdom Sample Consensus (RANSAC) algorithm and extracted contour boundary points of openings from a two-dimensional (2D) Delaunay triangulation similar to the work by Pu and Vosselman [11]. Those boundary points were transformed into parametric objects [22]. While these efforts successfully extracted sufficient boundary points to generate outline polygons of major features, they were highly sensitive to user-defined length thresholds and generated varying levels of geometric accuracy.

The quality of fitting polygons to building features has been improved by combining various data sources. For example, Becker and Haala [12] generated boundary lines derived from a photograph using a Sobel filter algorithm and from TLS data using a half-disc approach. These boundary lines were then combined together to improve accuracy, and airborne LiDAR and 2D ground plans provide the building's outlines. The façade details were improved by integrating boundary lines of openings into the building models [12, 23]. In that

approach, reconstructing building models involved integrating the boundary lines for cell decomposition. Opening details were subsequently refined by integrating image data. Similarly, Pu and Vosselman [11] used a Canny extractor for detecting feature edge from a photograph and Delaunay triangulation for extracting boundary points from the TLS data. The quality of the resulting models depended on the accuracy in mapping the boundary lines and the availability of such data. Despite these significant advances, a fully automatic approach for reconstructing highly detailed and accurate building models strictly from LiDAR data still remains a challenge [1].

Alternatively, volumetric methods have been employed to reconstruct object surfaces. In such approaches, the object's surfaces can be determined from a signal function distance (i.e. the distance from an arbitrary point to its oriented target plane [24]), an oriented-charge (e.g. the linear local distance field and a normal field between a point and its neighbours [25]), or a radial basis function, which approximates multivariable functions by a linear combination of terms based on a single, univariate function [26].

Using an altogether different method, Curless and Levoy [27] employed a volumetric method to reconstruct surfaces from range images by applying a cumulative, weighted, signed distance function. This method successfully filled gaps from curving spaces but could not, however, generate surfaces for arbitrary objects and required a voxel size smaller than any anticipated feature for reliable detection. Consequently, the method was rather computationally expensive. To overcome this, Pulli et al. [28] used an octree representation in which the whole input data set became the initial cubical volume and was recursively subdivided into eight smaller cubes until reaching a predefined sub-division depth (Figure 1). In that approach, each voxel was classified as "inside", "boundary", or "outside" by comparing the voxel's location to the sensor and range data. Consequently, the mesh of the

object's surface was created as the interface shared between outside cubes and other ones. Notably the octree has been used for the storage and compression of huge TLS point clouds [29], as well as for visualisation [30].

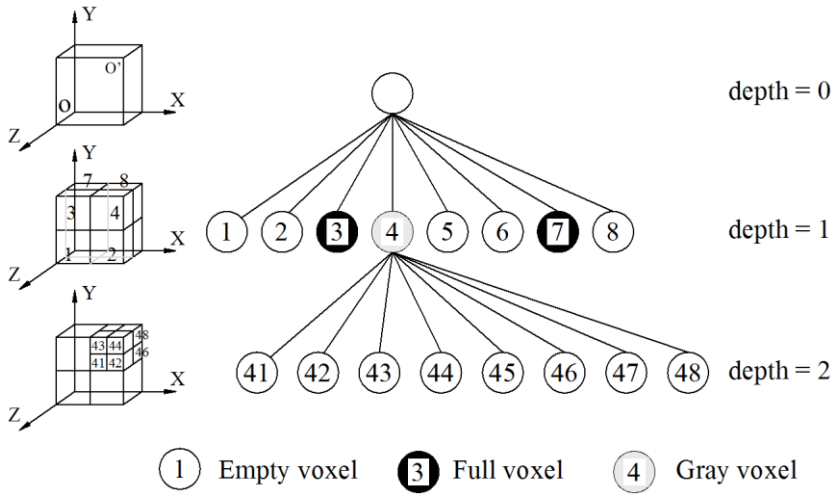


Figure 1. Octree representation

In another approach to compute an approximate distance field, Wang et al. [25] used oriented charges of each voxel of an octree representation, which was determined by points within neighbour voxels. However, surface features smaller than the smallest voxel size risked going undetected. To help overcome this, Dalmasso and Nerino [31] used compact radial support functions to estimate a surface based on a multi-scale, volumetric approach. Child voxels were classified as surface, inner, or outer voxels based strictly on the voxel's position with respect to the sensor. Surface voxels were continuously divided until reaching a specific scale or until the voxel contained only one data point. In an example of using octree representations in building reconstruction, Wang et al. [32] extracted sample points on boundary openings by examining adjoining voxels of a selected voxel along the vertical and horizontal directions. Data points within the selected voxel were classified as boundary points, if at least one adjoining voxel was empty. This method was successful in consistently detecting all openings but with relatively low geometric accuracy.

In summary, although many approaches have been developed to generate geometric models of building façades from LiDAR data, most are of limited use for computational modelling because of one or more of the following: (1) their primary aim is visualisation; (2) they are largely manual or semi-automatic, and thereby rely on user experience and/or supplemental data; (3) they need extensive user interaction before being fully usable within a commercial FEM package; or (4) they lack the geometric accuracy for meaningful engineering applications [33]. In structural analysis, allowable error depends on various factors including structure type, analysis purpose, and cost. While the geometry for a structural analysis model is traditionally presumed to be completely accurate, a 5% deviation is generally considered acceptable [34]. Therefore, a viable automated approach must be able to discern object boundaries, locate major features of structural interest, and overcome localized data occlusions. The FacadeVoxel (FV) algorithm is proposed by the authors for such a purpose and benchmarked herein for load-bearing, unreinforced masonry building façades.

3. THE PROPOSED FACADEVOXEL ALGORITHM

The FV algorithm has three main steps: (1) use of a hierarchical data structure to build a volumetric representation of an object by employing an octree representation; (2) extraction of boundary points from data within the voxels on boundaries of holes and the façade, along with the generation of their boundary lines; and (3) re-determination of voxel properties by comparing a voxel's position to a set of derived boundary lines (Figure 2). The geometric building model generated from this algorithm is subsequently imported directly into a finite element package for structural analysis. To achieve this, both topology and geometry of the

building model are converted into suitable data formats by a Boundary-Representation (B-Rep) [6].

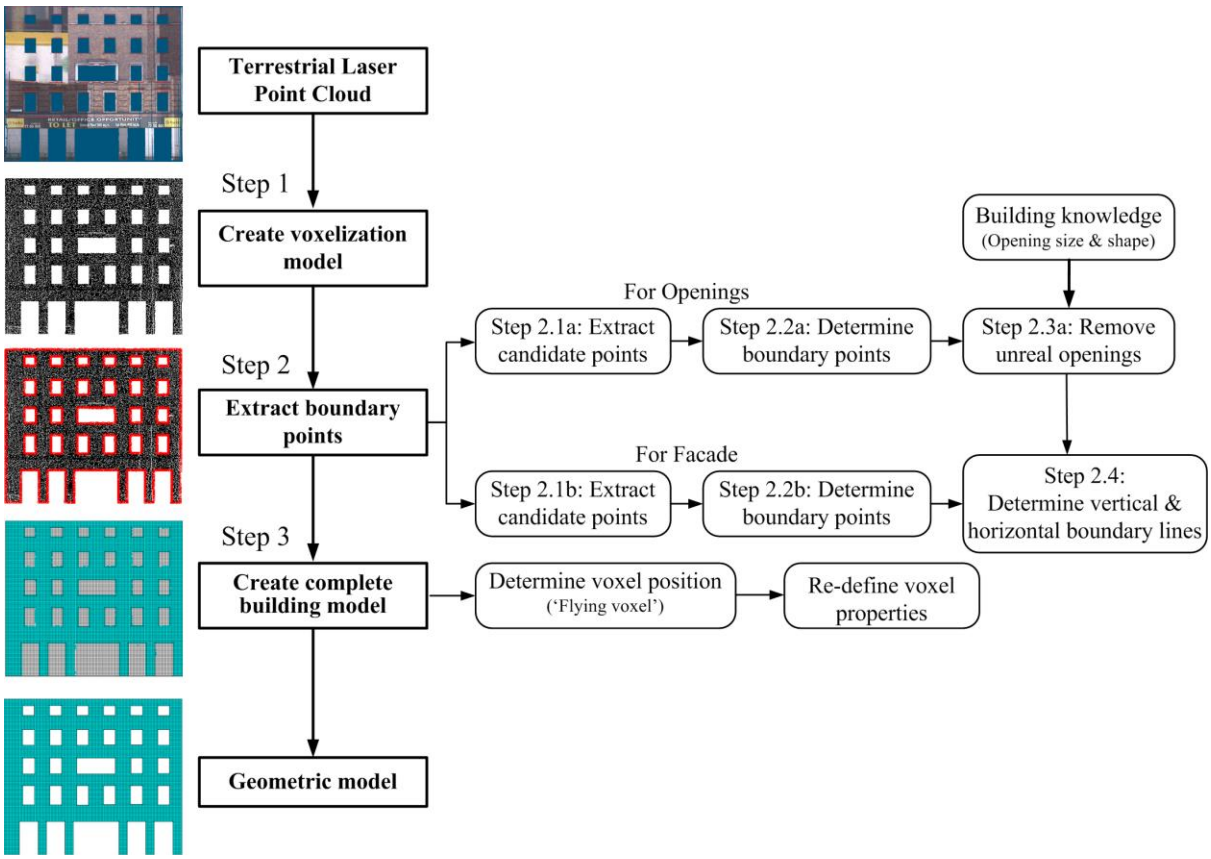


Figure 2. Workflow of the façade algorithm

3.1. Octree representation (Step 1)

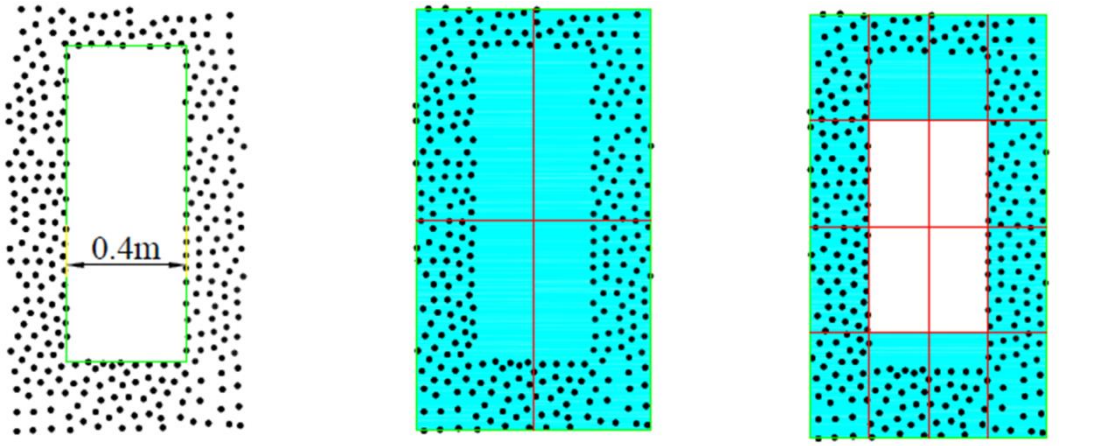
Sample points are defined as a set of point, $\mathbf{P}_i$ (i = 1,…,n), which have three coordinates, (x_i, y_i, z_i) $\in R^3$ in Cartesian coordinate system. Step one uses an octree representation as a volume initially bounding all sample points of the façade, where scanned data are projected onto a vertical fitting plane (Figure 2). As a terrestrial laser scanner cannot collect the facade thickness from a single scan of a building's exterior, a preselected voxel dimension along the depth direction of the façade is currently assigned. Thus, the subsequent subdivision mechanism of the octree representation is analogous to that of a quadtree [35], where a voxel is subdivided into four smaller voxels along the length and height directions of the façade.

However, as the long-term goal of this work is for three-dimensional (3D) building reconstruction and for its application to highly articulated buildings, as well as those of limited decoration, the structure is herein described as an octree, even though the current appearance is that of a quadtree.

Each voxel is described by its geometry, address, and status. Geometry is defined by coordinates of each voxel's two opposite corners; for example the corners O and O' at the octree depth of 0 (Figure 1). Similar to Tang [36], each node in the octree structure in the proposed algorithm has an address that reflects the relationship between adjacent nodes and the path from a current node to the root, which is represented by a string of integers (Figure 1). Finally, the voxel's status is determined based on a number of sample points contained within the considered voxel. The voxel is classified as "full", if the voxel contains at least one sample point. All others are classified as "empty".

Selecting an appropriate termination criterion of an octree is not a trivial task for automatic building reconstruction. Previously, several approaches have been employed. Ayala et al. [37] selected termination based on a minimal voxel size, while Pulli et al. [28] used a predefined maximum tree depth, and Wang et al. [25] adopted a maximum threshold of sample points contained within a voxel. In the proposed FV algorithm, the minimum voxel size is considered as the termination condition for recursively sub-dividing the octree representation, where the shortest side (either horizontal or vertical) of the voxel is less than half of the expected minimum opening size. The criterion is based on an assumption that the inside of an opening is represented either by an empty voxel or by a cluster of empty voxels. In the example presented in Figure 3, when a voxel's dimension is defined as greater than half of the minimum opening size of 0.4m [38], no empty voxels appear, thereby incorrectly implying the presence of no openings (Figure 3b). By applying the criterion, four empty

voxels appear, thereby clearly indicating a potential opening (Figure 3c). Thus, based on these premises, the required octree depth along the x- and y-directions can be expressed as equations (1) and (2).



a) Expected opening in TLS data shown in green lines

b) Octree with a voxel dimension of more than half of the expected minimum opening dimension

c) Octree with a voxel dimension of less than half of the expected minimum opening dimension

Figure 3. Octree representation with various voxel sizes

$$depth\_x = \left\lceil \log_2 \frac{x_{max} - x_{min}}{MinVoxelSize} \right\rceil \qquad \text{Equation 1}$$

$$depth\_y = \left\lceil \log_2 \frac{y_{max} - y_{min}}{MinVoxelSize} \right\rceil \qquad \text{Equation 2}$$

where x_{max}, x_{min}, y_{max}, y_{min} are the minimum and maximum coordinates of the sample input points. The term MinVoxelSize is set equal to half of the smallest façade feature expected to be detected, which in this case is a window; the sign [] means that the value is rounded up toward the nearest integer. In these equations, (x_{max}-x_{min}) and (y_{max}-y_{min}) are, respectively, the length and height of the bounding box. The maximum octree depth is defined as being equal to the smaller value of depth_x or depth_y.

3.2. Extract boundary points (Step 2)

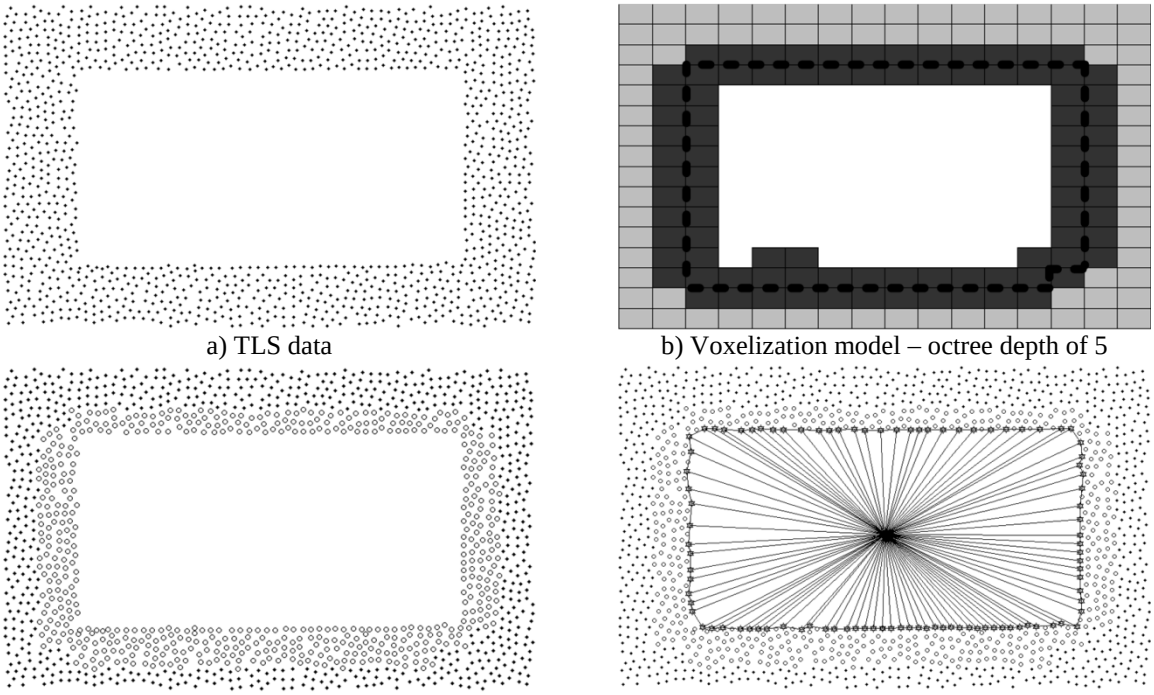
Glass or highly reflective materials tend not to return a LiDAR signal. Thus, it was hypothesized that boundary points on openings could be extracted from data points within the point cloud that lie on the perimeter of any empty voxel and that boundary points of the building are on the outer perimeter of the given dataset. Specific boundary point detection strategies are presented below.

3.2.1. Boundary point detection for openings (steps 2.1a-2.3a)

The portion of the FV algorithm for detecting boundary points of realistic openings is comprised of three subparts: (1) extract candidate points, (2) determine boundary points and (3) remove un-real openings (Figure 2). For extracting candidate points (step 2.1a), from point clouds (Figure 4a), the voxelization model was generated, and then the full voxels along the entire perimeter of the hole that was represented by an empty voxel cluster are detected by a flood-filling technique [39] (Figure 4b). The sample points within these full voxels on the proximity boundaries of openings are identified as candidate points. However, if only a few candidate points lie on the voxel's surfaces (for example see the dashed black line segments representing the surfaces projected onto a 2D plane in Figures 4b), boundary points may be missed. To overcome this problem, other full voxels adjoining the full voxels on the opening's boundaries are added to the set of full voxels containing candidate points (dark grey rectangles in Figures 4b). The candidate points are then segregated from the set of full voxels (Figure 4c).

After a set of the candidate points are determined, in step 2.2a, boundary points are extracted one-by-one (result shown in Figure 4d). In this process, the centre of the empty voxel group (which is the average of the centres of all voxels in the group) is considered as its rotational

centre, called P_0. From the candidate points, the closest point to P_0 in the Euclidean metric is defined as the initial reference point, P_1 (Figure 5a). A point in the neighbouring points $\{\mathbf{P}_2\}$ of the reference point P_1 is the target point, if this point has the smallest score resulting from combining the angle at P_1 between two vectors P_1P_0 and P_1P_{2i} and the perimeter of the triangle $P_0P_1P_{2i}$ (Figure 5b) as accomplished through comparing other points in the neighbourhood. Details of searching boundary points are described below.

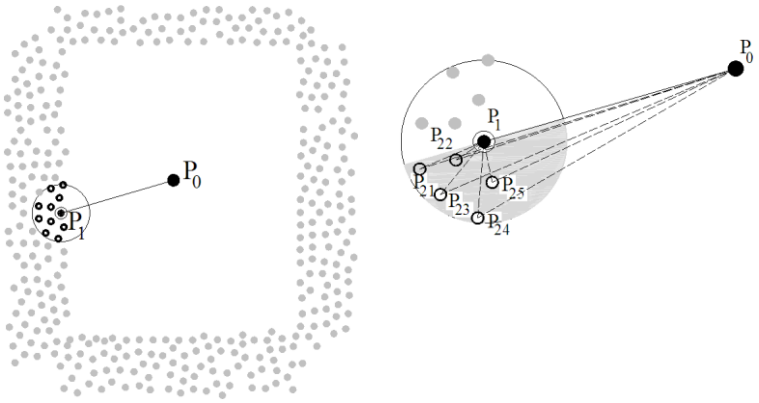


a) TLS data b) Voxelization model – octree depth of 5

c) Candidate boundary points d) Resulting boundary points

Notes:
- Black and unfilled holes are, respectively, point clouds and candidate points, while diamond points are target points.
- Dark and light grey rectangles are full voxels containing candidate points and others, while the white space represents an interior opening.
- Dashed black lines are voxel surfaces projected onto a 2D plane.

Figure 4. Process of window reconstruction



a) Select neighbouring points of the reference point, P_1 (*)

b) Determine target points from the sub-candidate points (+)

Notes:
(*) Black points are neighbouring points of P_1
(+) Sub-candidate boundary points are unfilled points in a grey hemisphere.

Figure 5. Searching boundary points of openings (doors and windows)

To determine a target point, the neighbouring points of the reference point, $\mathbf{P}_1$ were extracted from the full voxels around the hole (Figure 4b) by a k-nearest neighbour (kNN) technique [40]. The vector P_0P_1 rotates about the point P_0 to search the boundary points around the hole. Thus, only sample points within the neighbouring points lying on a hemisphere (the grey area with filled points-Figure 5b), which called the sub-candidate points $P_{2i} \in \{\mathbf{P_2}\}$, were investigated (as expressed by Equation 3). As the target point was determined from the sub-candidate points of the reference point, the searching process required at least 3 points, thereby making 3 triangles ($P_0P_1P_{2i}$). This implies that with an arbitrary location of P_0 and a uniform regular distribution within the point cloud the neighbourhood size should be 8 kNNs. However, as uniform distribution cannot be assured due to variations in the environment, the quantity of neighbouring points for P_1 was experimentally chosen as 10 points (Figure 5a).

$$\mathbf{C}_{\mathbf{P}_2} = \left\{ \mathbf{P_2} \in \mathbf{P} \mid \mathbf{P_2} \in \text{a hemisphere defined by } P_0P_1 \right\} \quad \text{Equation 3}$$

where **P** is the neighbourhood of the reference point, P_1.

The boundary score for the angle $\angle P_1$ and a triangle perimeter of each triangle are given as Equations 4 and 5:

$$\prod_{\angle \mathbf{P}_1} = \frac{\angle \mathbf{P}_1}{\max(\angle \mathbf{P}_1)}$$ Equation 4

$$\prod_{\Delta(\mathbf{P}_2)} = \frac{\Delta(\mathbf{P}_2)}{\max(\Delta(\mathbf{P}_2))}$$ Equation 5

where $\angle \mathbf{P_1}$ is the angle at P_1 made by the two vectors P_1P_0 and P_1P_{2i}, and $\mathbf{\Delta(P_2)}$ is the perimeter of the triangle $P_0P_1P_{2i}$, where P_{2i} is the point in the sub-candidate points, $\{\mathbf{P_2}\}$ (Figure 5b).

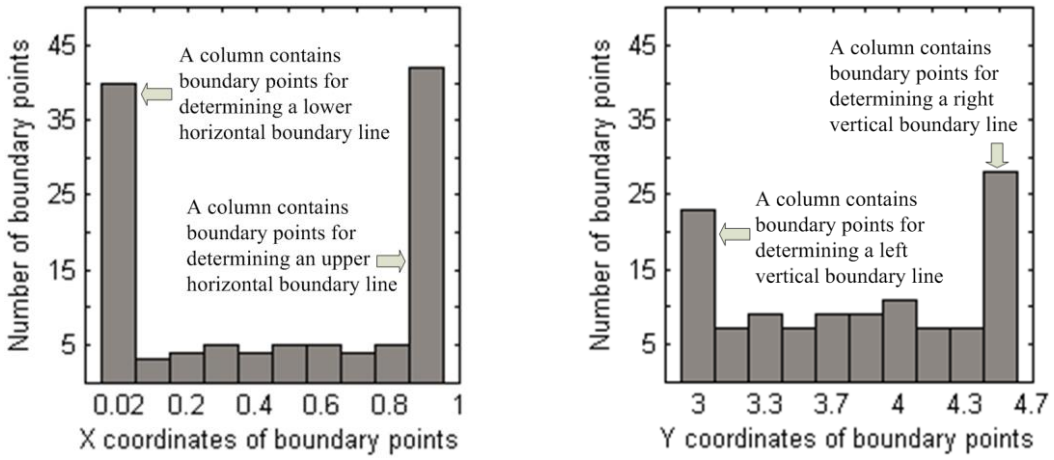
To increase the robustness of determining the boundary point, these boundary scores are combined into a weighted sum using Equation 6:

$$\prod_{(\mathbf{P}_2)} = w_{\angle} \prod_{\angle \mathbf{P}_1} + w_{\Delta} \prod_{\Delta(\mathbf{P}_2)}$$ Equation 6

in which $w_{\angle}$ and w_{Δ} are, respectively, the weights for a score of the angle at P_1 and the triangle perimeter, where $w_{\angle} + w_{\Delta} = 1$. In the proposed algorithm, a uniform weighting was selected as the default value distribution. Once the target point is determined, it is assigned as the reference point for the next iterative detection cycle, and the process continues, until the cumulative angle $\angle P_0$ equals to 360˚.

Additionally, to eliminate unrealistic openings due to missing data or occlusions (step 2.3a), equivalent dimensions of openings are computed from boundary points of each building feature using a histogram. Employing the proposed approach outlined by Truong-Hong et al. [6], the pre-extracted boundary points of each opening are analysed by histograms along the x- and y- coordinates. For a window, the boundary points are visible as the two histogram peaks along the x- and y-directions corresponding to the vertical and horizontal boundary lines, respectively (Figure 6). However, for ground floor doors and floor-to-ceiling plate glass

windows, only one peak (along the histogram's y-direction) appears. Subsequently, the equivalent dimensions of the opening can be calculated according to Equation 7 and 8.



a) Histogram for x-coordinates of boundary points along x-direction

b) Histogram for y-coordinates of boundary points along y-direction

Figure 6. Using histograms to determine height and length of a window

$$H_o = \left| \frac{\sum_{i=1}^{n} y_i^{up}}{n} - \frac{\sum_{j=1}^{m} y_j^{down}}{m} \right| \qquad \text{Equation 7}$$

$$L_o = \left| \frac{\sum_{i=1}^{l} x_i^{left}}{l} - \frac{\sum_{j=1}^{k} x_j^{right}}{k} \right| \qquad \text{Equation 8}$$

where n and m, and k and l are, respectively, the number of boundary points belonging to two peaks ("up" and "down") along the y-histogram, and to two peaks ("left" and "right") along the x-histogram. Notably, for a hole along the ground surface, the down peak is the smallest column in the y-histogram, with respect to y-coordinate boundary points.

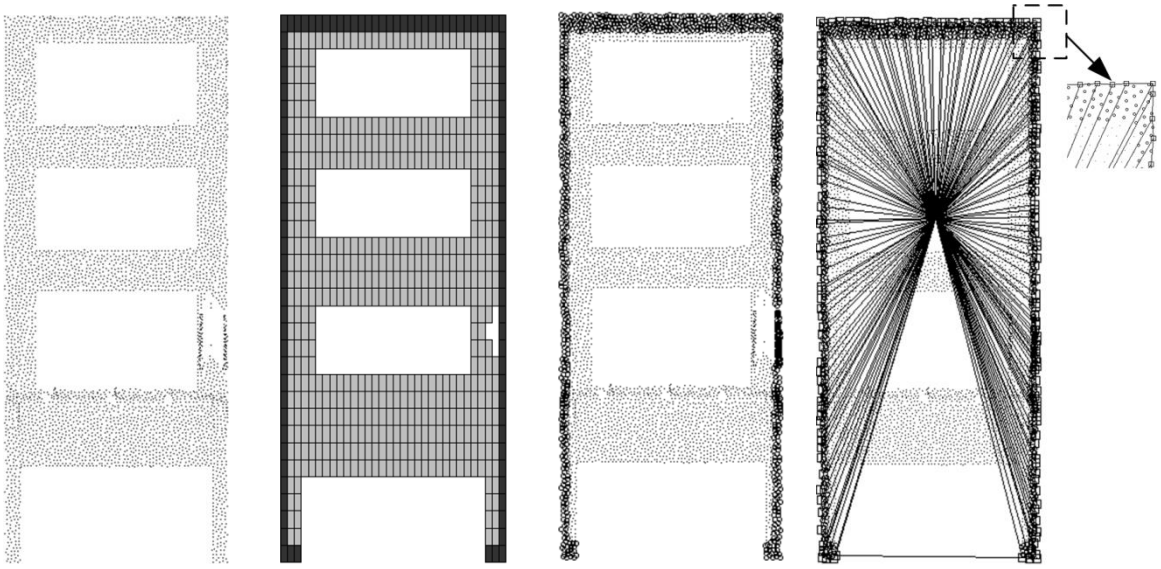
Finally, the holes are considered real openings (e.g. doors or windows), if their characteristics satisfy those of common openings in building façades involving minimum lengths and widths equal or greater than 0.4m [6, 7, 12, 21] and a height-to-length ratio between 0.25 and 5.0

[38, 41], as per the function $f(H_o, L_o, H_o/L_o)$, where H_o and L_o are the height and length of the opening, respectively (Equation 9).

$$f(H_o, L_o, H_o / L_o) = \begin{cases} \text{if } H_o \geq 0.4; L_o \geq 0.4; 0.25 \leq H_o / L_o \leq 5.0 : \text{Opening} \\ \text{Otherwise} : \text{non - opening} \end{cases} \quad \text{Equation 9}$$

3.2.2. Boundary point detection for the façade (steps 2.1b-2.3b)

Similar to boundary point extraction for openings, the façade boundary point detection process involves the following: extracting the candidate boundary points (Step 2.1b) and determining the target boundary points from the candidate points (Step 2.2b) (Figure 2). As the boundary points of the façade are on the envelope of the input data (either a convex or concave hull) (Figure 7a), the process initially extracts full voxels, either connected to the bounding box or those that are outermost along the vertical and horizontal grids (dark gray voxels in Figure 7b). Subsequently, the sample points contained within full voxels are considered to be candidate points for the searching of boundary points (Figure 7c). The results are shown in Figure 7d. Details of this process are provided below.



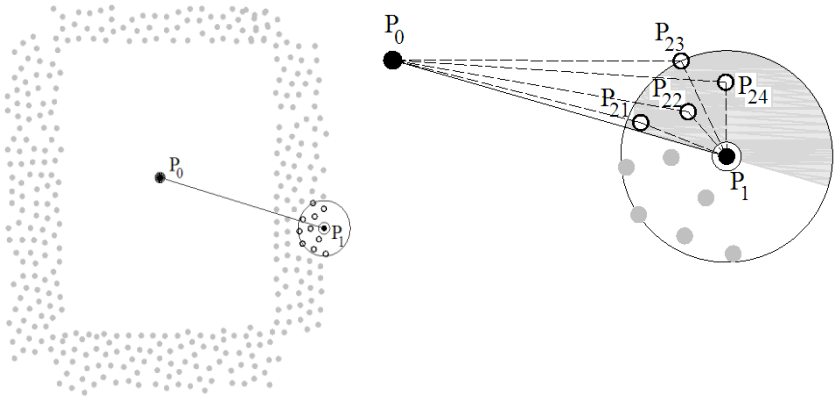
a) Input point cloud data of the façade | b) Full voxels containing candidate points | c) Resulting candidate points | d) Resulting target points

Note:

- Black points are input data set; unfilled circle points are candidate points; and unfilled square points are target points
- Dark grey rectangles are full voxels containing candidate points; and light grey rectangles are other full voxels.

Figure 7. Process of detecting boundary points for a building façade

Analogous to detecting boundary points of the openings, with the rotational centre, P_0, is an average of all candidate sample points. The process starts with an initial reference point, P_1, which is the closest point to the minimum x-, y-coordinates of the input data in the Euclidean metric. From the reference point (P_1), the neighbouring points are determined by using kNN searching with 10 kNNs (Figure 8a). However, only sample points in the neighbouring points lying on a hemisphere (as determined by the ray P_0P_1) are considered for the searching of target points. These are shown as unfilled points in Figure 8b and are called the sub-candidate points $\{\mathbf{P_2}\}$. The point $P_{2i} \in \{\mathbf{P_2}\}$ is the target point, if the angle $\angle P_1$ of the triangle $P_0P_1P_{2i}$ is the greatest in a set of angles $\angle P_1$ of triangles established by P_0, P_1, and P_{2i} (Figure 8b). Next, the target point is assigned as the reference point for the next iterative detection, and the process is repeated, until the accumulated angle at P_0 equals 360 degrees (Figure 8b). The resulting detection is shown in Figure 7d.



a) Select neighbouring points of the reference point, P_1 [(a)]

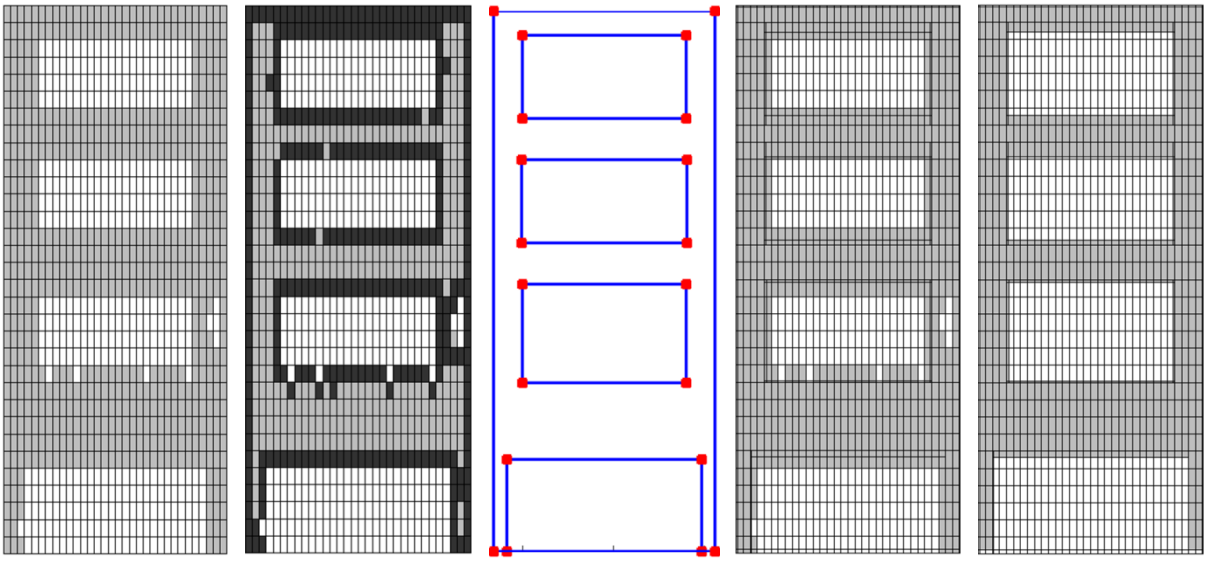
b) Determine target points [(b)]

Notes:
[(a)] Black points are neighbouring points of P_1
[(b)] Sub-candidate boundary points are unfilled points in a grey hemisphere

Figure 8. Identification of target points of a facade's boundary

3.2.3. Determine vertical and horizontal boundary lines of the façade and its openings (step 2.4)

Since, an area of a convexhull of boundary points of each opening is always smaller than that of the façade, the group of the boundary points forming the largest area of the convexhull constitutes the façade's boundary points. In the FV algorithm, boundary lines of a building façade and its openings, which are assumed to involve only vertical and horizontal lines (orthogonal to each other around boundaries and openings) are directly generated from boundary points detected in the previous sections (steps 2.2b and 2.3a). From those, the full voxels in the octree representation generated from the TLS data (Figure 9a) that contained boundary points are extracted (dark grey rectangles in Figure 9b).



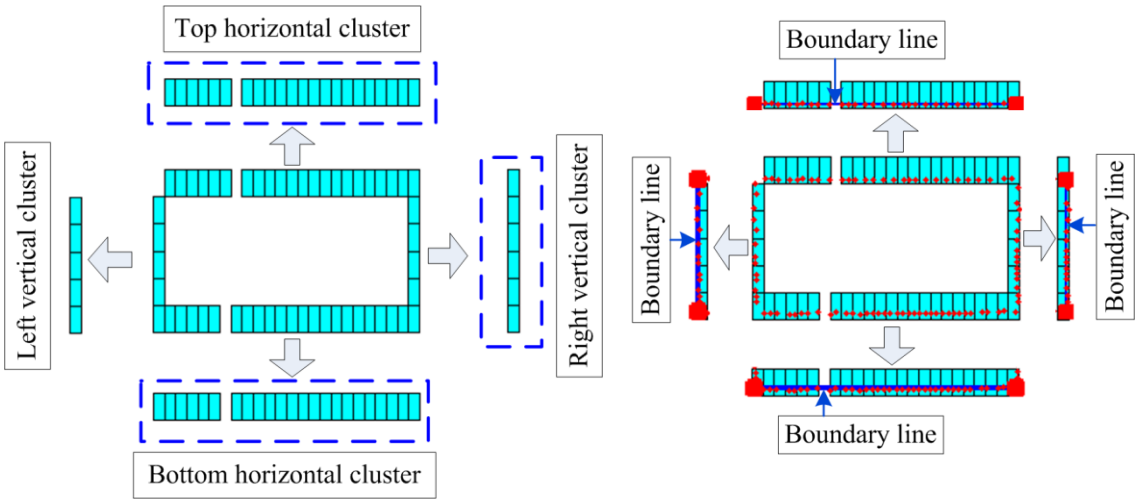
a) Initial voxelization model (octree depth 6) | b) Characterized voxel model (octree depth 6) | c) Detected boundary lines of façade and its openings | d) New voxelization model (octree depth 7) | e) Final voxelization model (octree depth 7)

Notes:
- Light grey rectangles are full voxels; dark grey rectangles are full voxels containing boundary points of the façade and its openings; and white rectangles are empty voxels
- The reconstructed building model is generated from input data of Figure 7a

Figure 9. A building model reconstruction based voxelization

Next, the extracted full voxels of each building feature are divided into sub-clusters of full voxels corresponding to horizontal and vertical boundary lines by using a grid clustering technique. The voxels containing boundary points determined in a previous step are on the

same horizontal or vertical grid to be extracted. From that the grid voxels are only considered in order to re-extract the boundary points possessed by these voxels for reconstructing boundary lines. However the boundary points in the grid must satisfy the following conditions: (1) the maximum distance between two boundary points in the grid is not less than the minimum opening size, and (2) the minimum distance between two adjacent boundary points belonging to adjoining full voxels is not greater than half of the opening size (herein 0.4m is the adopted minimum opening size). As such, Figure 10a illustrates the results of horizontal and vertical sub-clusters of full voxels for an opening. Finally, the boundary lines are determined from the boundary points within the full voxels of these voxel sub-clusters by using a least-squares method (Figure 10b) [42]. Additionally, the boundary lines of each building feature are adjusted to ensure continuity by extending or trimming the boundary lines at their points of intersection (Figure 9c).



a) Grouped vertical and horizontal clusters containing sample points on the boundary

b) Proposed boundary lines of the opening

Figure 10. Modified grid clustering technique

3.3. Create complete building model (Step 3)

As only full voxels belonging to the solid wall are converted into the solid model for direct importation into FEM packages, voxel properties in the initial octree representation must be

re-assessed. For that, a new voxelization model is generated by dividing the initial voxelization model by the boundary lines (Figure 9d). The number of child voxels depends on the number of boundary lines intersecting the parent voxel. There are two child voxels, if one boundary line intersects the parent voxel, and four child voxels, if there are two intersections. However, no subdivision occurs, if no boundary lines intersect the voxel, or if the boundary lines are on the boundaries of the parent voxel.

Subsequently, each voxel in the re-voxelization model is characterized by using the Flying Voxel method proposed by Truong-Hong et al. [6], in which voxels inside of openings or outside of the façade are labelled either as empty or as full. Figure 9e shows the re-characterized voxels of the octree voxelization in Figure 9d. Subsequently, the full voxels in the octree nodes are stored in a neutral file (with the file extension .ANF), whereas the topology and geometry are described as a B-Rep scheme [6].

In summary, the FV algorithm was designed as a novel approach to efficiently and accurately detect sample points on an object's boundaries. In this, a sample point drawn from a voxel on an object's boundaries can be declared as a boundary point, only if its score is higher than those of its neighbouring points. Therefore, the proposed approach does not trawl through all input data points to identify boundary points, as is done for the other fully automated, processes [e.g. 6,7]. As such, the FV algorithm is proposed both to improve accuracy of the resulting building models, while significantly reducing the computational cost.

4. EXPERIMENTAL TESTS, RESULTS, AND DISCUSSION

4.1. Experimental data

To test the FV algorithm, three building façades in Dublin, Ireland were scanned with a terrestrial laser scanner. These buildings were selected for three reasons. Firstly, if the proposed algorithm cannot be shown to work reliably with relatively simple structures, there is no use trying them on more complicated ones; notably 2D facades have to date been preferable for use in structural analysis of masonry building rather than fully 3D ones. Secondly, this building type represents the vast majority of the architectural fabric of many urban centres. Thirdly, unlike most existing structures, independent measured drawings were readily available for these structures, which were used for ground truth verification.

Point cloud data of the building facades were acquired by a Trimble GS200 unit [43]. The data collection processing was controlled by the affiliated propriety software RealWorks Survey Advanced (RWS) V6.3, which was installed on a laptop linked to the scanner [44]. Although the scanner can collect data across a wide range, typically two scanner positions were needed up to acquire point clouds of each façade because of limitations caused by traffic, terrain, and footpath space. The maximum standoff distance was approximately 10 m for Building 1 (2 Anne St. South, Dublin City) and Building 2 (5 Anne St. South, Dublin City), and 30 m for Building 3 (2 Westmoreland St., Dublin city) (Figure 11a, 12a and 13a). The TLS data were acquired with a sampling step of 10 mm at a 100 m range for Buildings 1 and 2, and of 20 mm for Building 3. Additionally, in order to register the point clouds of each facade, there were three reference points defined for each scanner station. Geometries on the façade (e.g. corners of the windows or window ledges) were selected as the reference points, instead of using special targets or nails to mark the reference points. The reference regions were scanned with a sampling step of 2 mm at 100m in Building 1 and 2, and with 10 mm sampling step at 100m for Building 3. The acquired data included x-, y-, z-coordinates, intensity, and RGB values. The intensity value varies according to the material surface

reflectivity measured and the optical wavelength of the laser used; for more details of the data acquisition see Truong-Hong [45].

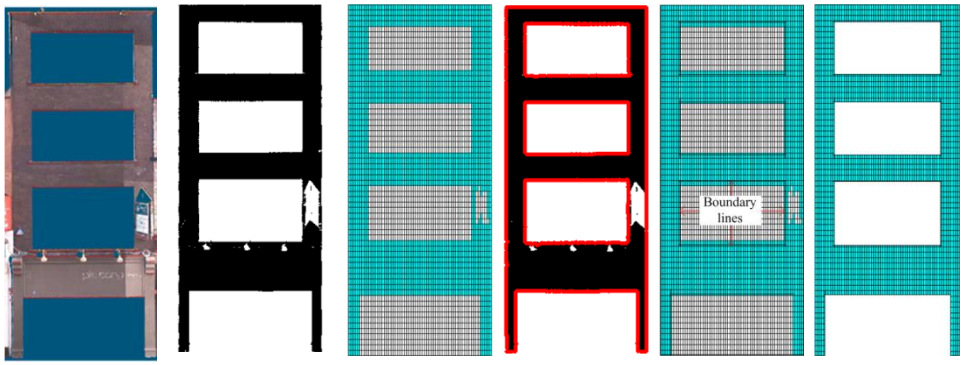
Pre-processing point clouds by use of the RWS V6.3 involved two main steps: (1) merging the scans from the multiple scanner stations, and (2) removing all irrelevant data points. Point clouds from two scanner stations of each facade were manually merged by using reference points of both the source and target stations. A trial and error process was undertaken by selecting a pair of points from the source and target stations until the average error between a pair of data sets could be expressed in terms of a distance error less than 5 mm. As data from internal walls/objects or occluding elements (e.g. trees and buses) are often in different planes from the facade, those data points were removed subsequently using the in-built segmentation technique within the RWS V6.3 [44]. Details of the data pre-processing are fully available in Truong-Hong [45]. Notably, processing time for this step done within the RWS V6.3 software was typically less than 5 minutes for each of the three data sets. Finally, in order to evaluate efficiency and robustness of the proposed algorithm and to validate reliability of reconstructed building models, four sampling densities data were tested for each building facade (Table 1), in which NS00 was the point cloud data set obtained directly from the pre-processing step, and the other three data sets (NS20, NS50, NS75) were re-sampled from NS00 by using the random re-sampling data function built into the RWS V6.3 software [44].

Table 1. Dataset sizes

Building	Sampling dataset			
	NS00 (*)	S20 (2500 points/m^2)	S50 (400 points/m^2)	S75 (175 points/m^2)
B1-2 Anne St. South	264,931	51,171	9,909	4,643
B2-5 Anne St. South	190,865	51,884	11,119	5,366
B3-2 Westmoreland St.	650,306	353,848	71,155	35,468

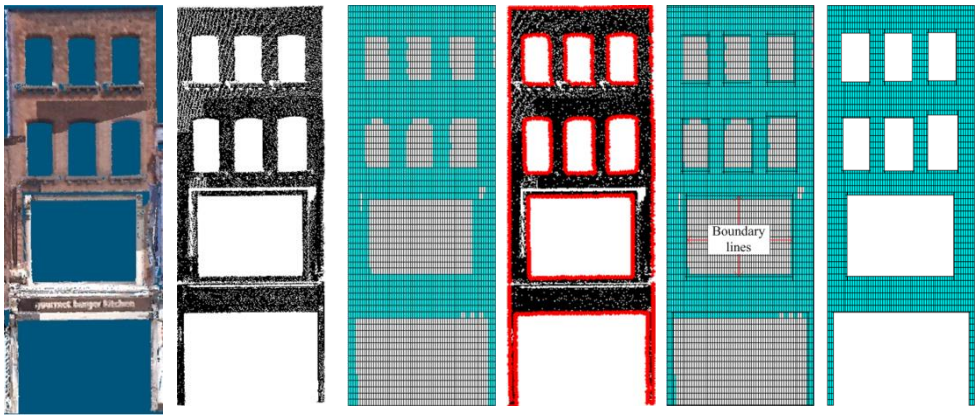
* Sampling density of the original datasets of Buildings 1, 2 and 3 were respectively around 9.0k, 6.5k and 2.8k points/m^2.

The proposed approach was implemented in MATLAB scripts [46], and experimental tests were run on a Dell Precision Workstation T5400 with a main system configurations as follows: Intel (R) Pentium (R) Xeon (8CPU) CPU speed 2GHz with 24 Gb RAM. Automatic generation of solid models for the three building façades are shown in Figures 11-13 (each showing a distinct, sample data density).



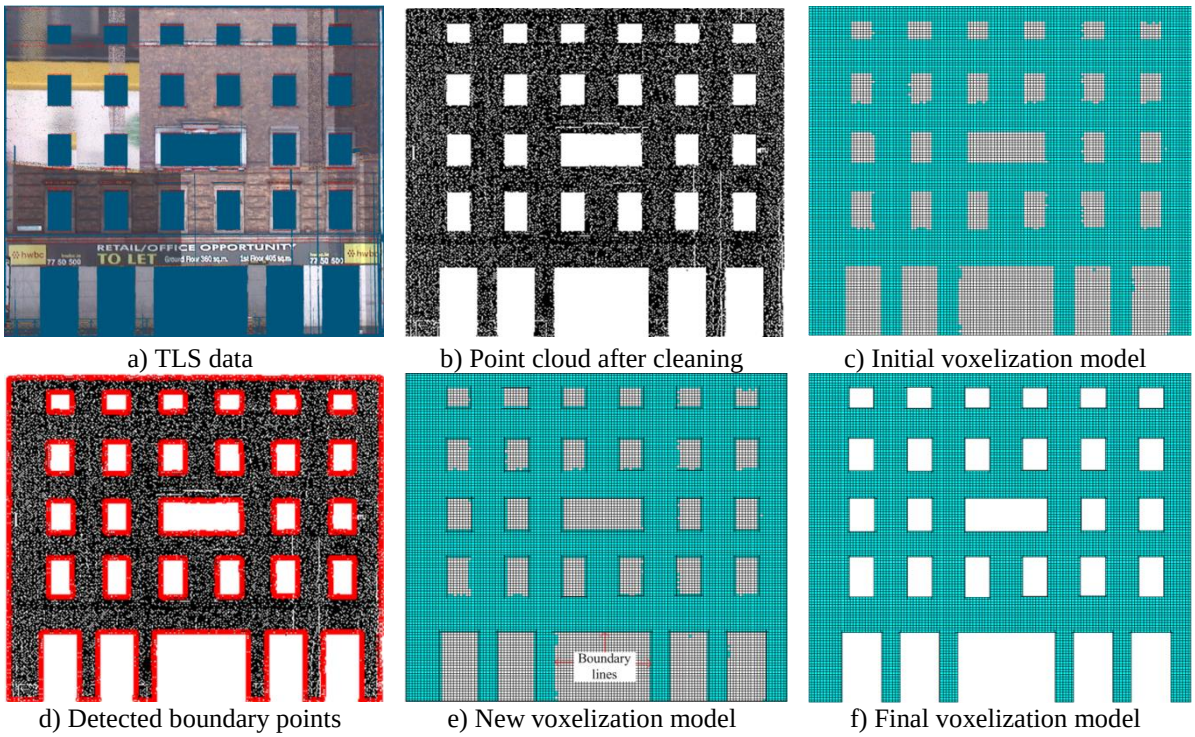
a) TLS data | b) Point cloud after cleaning | c) Initial voxelization model | d) Detected boundary points | e) New voxelization model | f) Final voxelization model

Figure 11. Façade reconstruction of Building 1 (4.95m length by 12.16m height) based on the NS00 dataset (Table 1)



a) TLS data | b) Point cloud after cleaning | c) Initial voxelisation model | d) Detected boundary points | e) New voxelization model | f) Final voxelization model

Figure 12. Façade reconstruction of Building 2 (4.90m length by 13.28m height) based on the S20 dataset (Table 1)



a) TLS data b) Point cloud after cleaning c) Initial voxelization model

d) Detected boundary points e) New voxelization model f) Final voxelization model

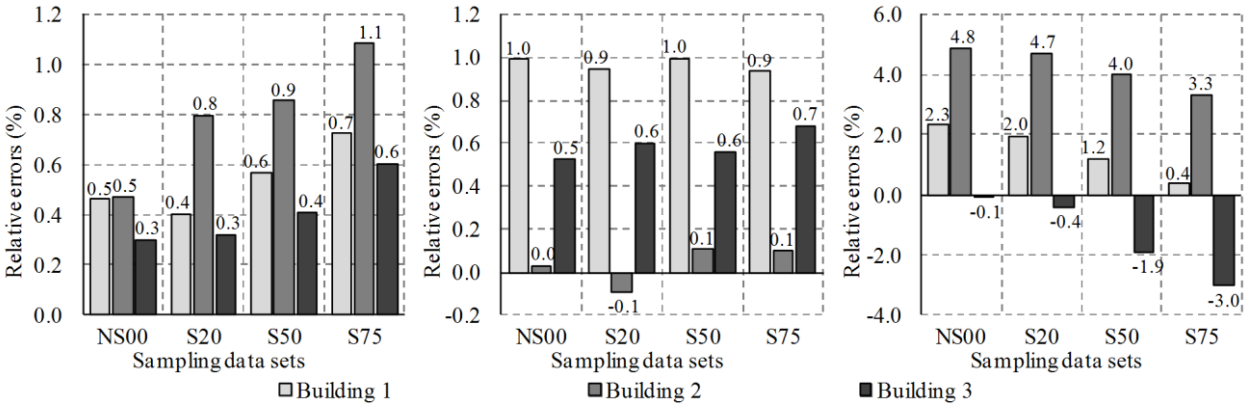
Figure 13. Façade reconstruction of Building 3 (19.36m length by 17.0m height) based on the S50 dataset (Table 1)

4.2. Results

The FV algorithm was successful in reconstructing the building façades and all openings, as well as automatically filling occlusion-based openings (Fig.s 11-13). Auto-generated façade dimensions and opening areas were compared to CAD models obtained from on-site surveys in order to assess geometric accuracy of those building models (Tables 2). The algorithm slightly underestimated lengths and heights – generally less than 1.1% (<121 mm-B2S75) and 1.0% (<53 mm-B1NS00), respectively, (Figure 14a and b). For Building 3, the relative errors of the length and height were only 0.6% (<117 mm-B3S75) and 0.7% (<116 mm-B3S75) (Figures 14a and b). Additionally, the opening areas in the reconstructed models of Building 1 and 2 underestimated those apertures by 2.3% (<0.72 m^2–B1NS00) and 4.8% (<1.67 m^2-B2NS00) in terms of relative errors, respectively (Figure 14c). The opposite trend was observed in Building 3, where the opening area in the reconstructed models was 3% greater (>2.91 m^2-B3S75) (Figure 14c and Table 2).

Table 2: Derived overall dimensions and opening areas of the 3 facades

Building	Aspect	CAD	Resulting geometries from the various sampling datasets			
			NS00	S20	S50	S75
B1-2 Anne St. South	Length (m)	4.95	4.93	4.93	4.92	4.91
	Height (m)	12.16	12.04	12.05	12.04	12.05
	Opening area (m^2)	30.70	29.98	30.10	30.33	30.58
B2-5 Anne St. South	Length (m)	4.90	4.88	4.86	4.86	4.85
	Height (m)	13.28	13.28	13.29	13.27	13.27
	Opening area (m^2)	34.46	32.79	32.84	33.07	33.32
B3-2 Westmoreland St.	Length (m)	19.36	19.30	19.30	19.28	19.24
	Height (m)	17.00	16.91	16.90	16.91	16.88
	Opening area (m^2)	96.20	96.28	96.59	98.02	99.11



a) Facade length b) Facade height b) For façade opening area

Figure 14. Relative error of quantities of interest for three building facades

4.3. Discussion

The boundary point detection approach proposed in the FV algorithm extracted sufficient boundary points to consistently reconstruct the three building facades and their actual openings (Figure 11-13). The FV algorithm introduced a procedure to classify a hole as either a proper opening or an unrealistic hole due to missing or occluded data. The procedure succeeded in recognizing arbitrarily shaped holes such as those produced by trees and automatically remove them (Figure 15a) and cars (Figure 15b). Notably, segmentation also results small holes.

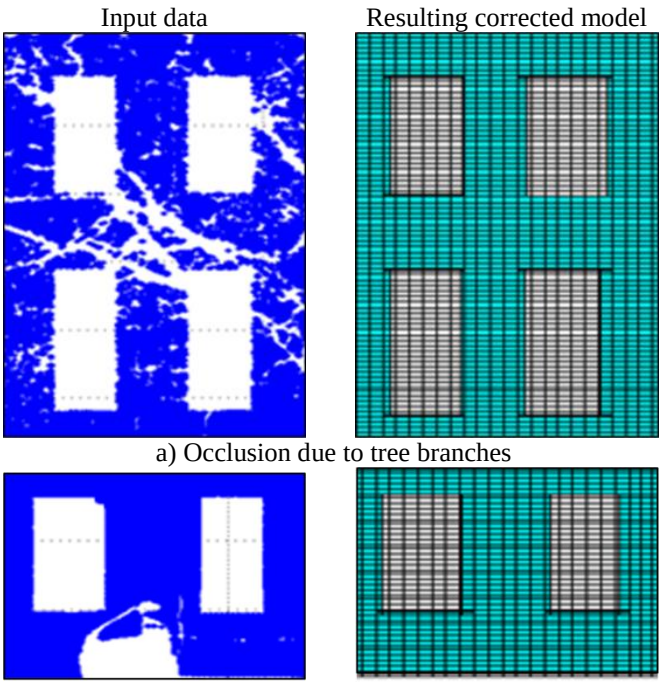
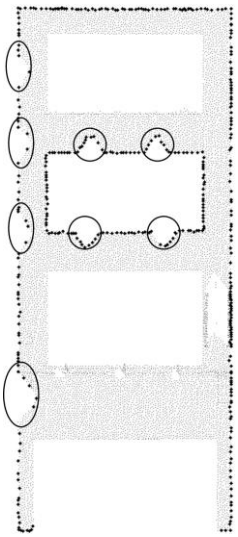
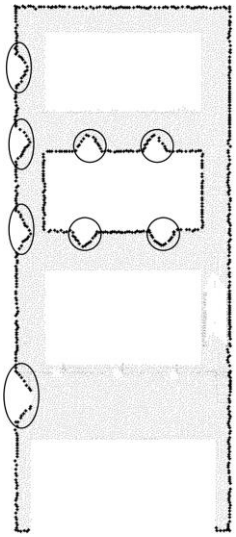


Figure 15. Example of automatically filling un-expected holes

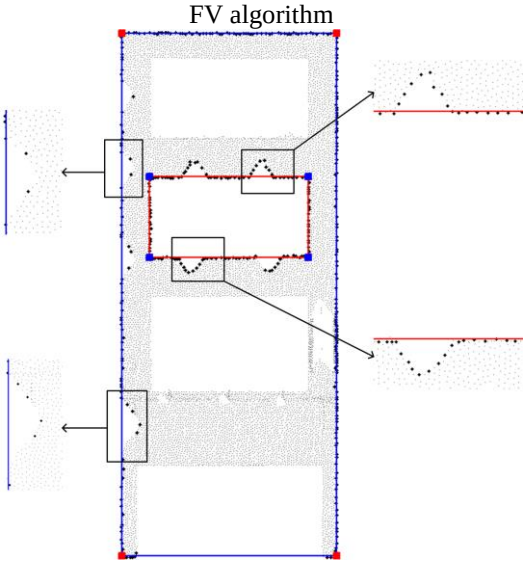
In the proposed algorithm, constrained conditions through a grid clustering technique are implemented to eliminate incorrect boundary points before generating boundary lines. The technique is based on the observation that voxels that possess incorrect boundary points lie on a different grid from voxels containing the correct boundary points. The proposed algorithm was evaluated against the use of an alpha shape with line fitting [47]. For example, if the input data of Building 1 (2 Anne street South) contains several defects on the left boundary of the façade and lower and upper sides of the window in second story, which mark as circles (Figure 16a and b). Obviously, boundary points from both FV algorithm and a comparative method contained un-expected boundary points (Figure 16a and b) and the boundary lines were then generated based on the boundary points (Figures 16c and d). Notably, a threshold radius ($1/\alpha$) of 75mm implemented in the alpha shape method was selected experimentally to optimize output results.



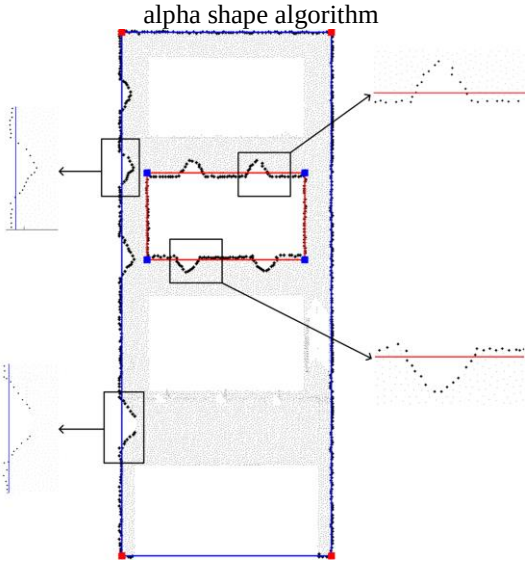
a) Boundary points of the window and façade from FV algorithm



b) Boundary points of the window and façade from alpha shape algorithm



c) Boundary points and lines of the window and façade from FV algorithm



d) Boundary points and lines of the window and façade based alpha shape algorithm and fitting lines

Figure 16. Reconstructing boundary lines of the façade and window by combining alpha shape method and fitting line and by FV method

(*) circles showed defect data and un-expected boundary points

Clearly, the comparative method gave negatively impact on reconstructed boundary lines. The boundary lines at the left side of the façade and ones at lower and upper side of the window migrate from the actual boundaries (Figure 16d). This problem does not occur in the FV algorithm (Figure 16c). Furthermore, the relative error in the overall façade length of the comparative method is increased up to 1.88% (as compared to only 0.59% for the FV method (Table 3)). Additionally, the window dimensions from the FV were also better than ones derived from the comparative method. Of which, the window length in the FV method

generated relative errors of only 1.11% (FV method) vs. 1.33% (the comparative method). For the window height, the difference in results was even more profound. The relative error for the FV method was -7.97% vs -16.40% for the comparative method. In conclusion, the FV algorithm can eliminate incorrect boundary points during reconstructing boundary lines.

Table 3. Evaluation of FV algorithm with defect input data

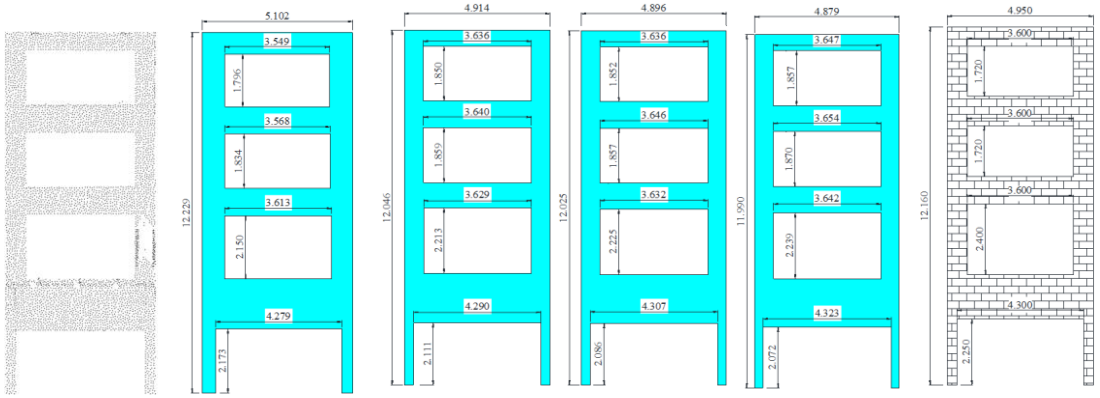
Aspect	Measured drawings	Competitive method		Relative error (%)	
		FV algorithm	MBR	Measured drawings vs. FV	Measured drawings vs. MBR
Façade: Length (m)	4.95	4.92	4.86	0.59	1.88
Façade: Height (m)	12.16	12.04	12.05	1.00	0.89
Window: Length (m)	3.6	3.64	3.65	-1.11	-1.33
Window: Height (m)	1.72	1.86	2.00	-7.97	-16.40

The efficacy of the FV approach and its compatibility with computational modelling were benchmarked against the commercial program Kubit [19] and two automatic approaches developed by the authors (the FacadeDelaunay (FD) algorithm [6] and the FacadeAngle (FA) [7]). The Kubit software allows users to use AutoCAD tools to create building models based on point clouds. The FD and FA approaches automatically reconstruct building, where the boundary points were extracted based on characteristic of triangles in triangulation mesh and an angle criterion, respectively. These algorithms are described elsewhere [6,7].

For evaluation, the S75 datasets with 175 pts/m^2 (Figure 16a, 17a and 18a) were selected to generate the building models to test robustness; denser datasets require longer acquisition times and are, thus, less economical.

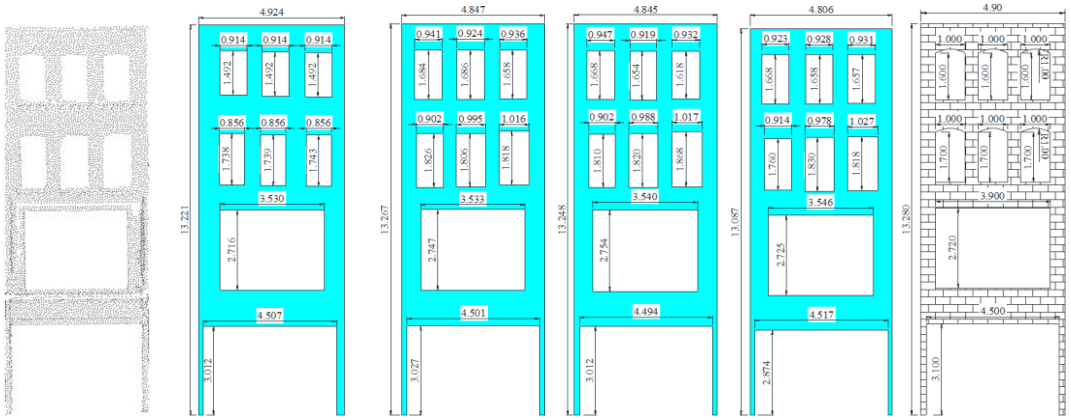
In Kubit, the building models were created within an AutoCAD program semi-automatically, by manually identifying boundaries of the building and its openings using various, in-built toolbox aids. For this, separate building photographs are needed to define openings

realistically; see [33] for further details. The FD and FA algorithms are described previously elsewhere [6, 7].



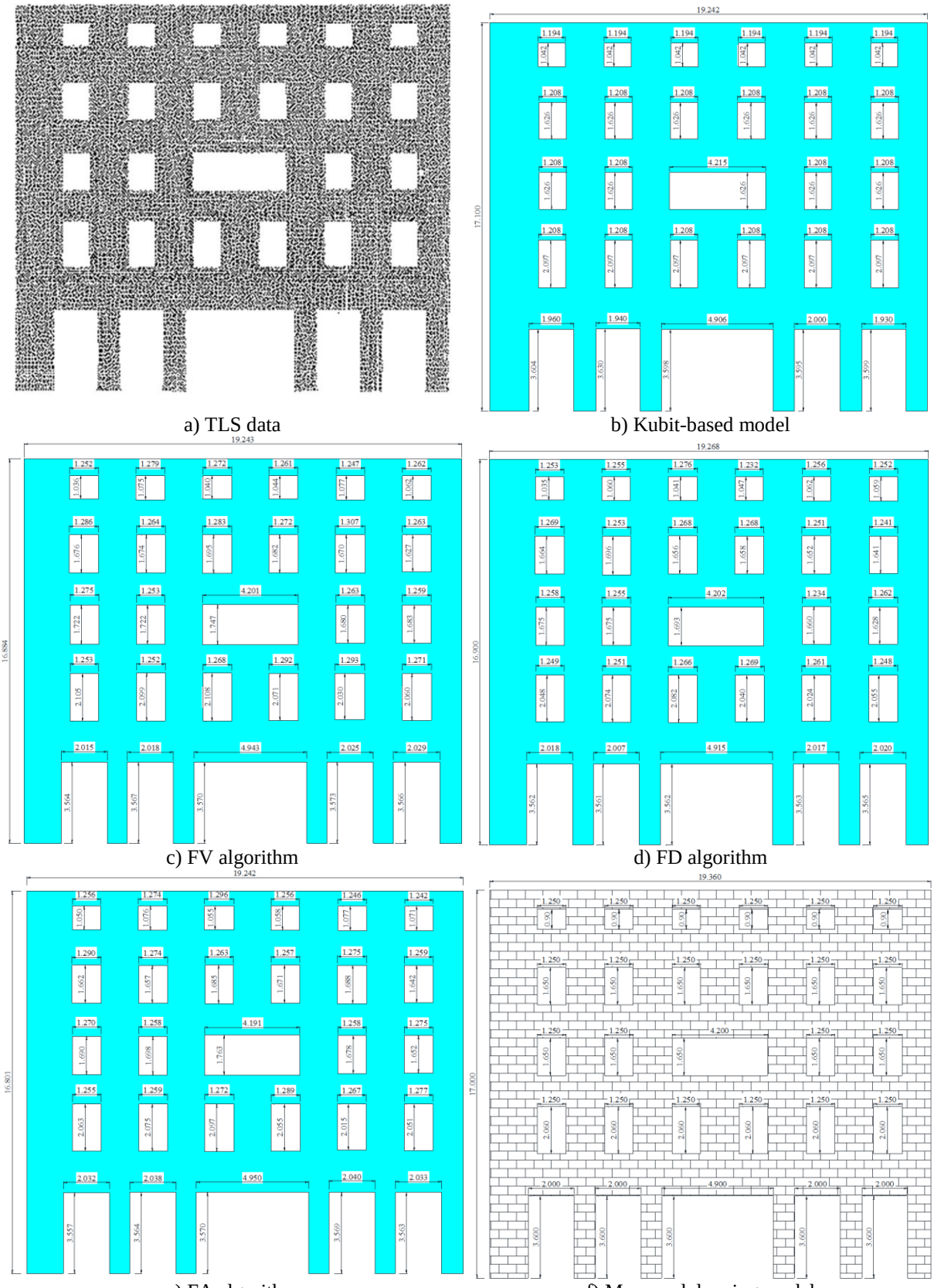
a) TLS data b) Kubit-based model c) FV Algorithm d) FD algorithm e) FA algorithm f) Measured drawing model

Figure 17. Solid models of Building 1 reconstructed from 175 points/m^2 by using various approaches



a) TLS data b) Kubit-based model c) FV algorithm d) FD algorithm e) FA algorithm f) Measured drawing model

Figure 18. Solid models of Building 2 reconstructed from 175 points/m^2 by using various approaches



a) TLS data

b) Kubit-based model

c) FV algorithm

d) FD algorithm

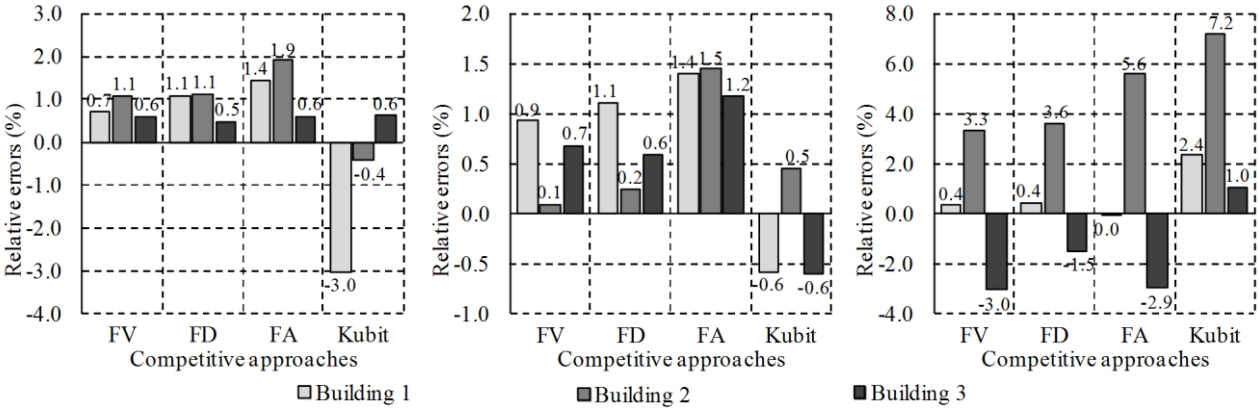
e) FA algorithm

f) Measured drawing model

Figure 19. Solid models of Building 3 reconstructed from 175 points/m^2 by using various approaches

With the Kubit program, realistic building models with smooth boundary lines were created rapidly (Figures 16b, 17b, and 18b). While the resulting models were compatible for FEM importation, independent knowledge of the existing buildings' façades (e.g. the number, size, and position of openings) was fully required. The other approaches did not need this information. Resulting building models of each approach are shown in Figures 16b-e, 17b-e, and 18b-e. The reported dimensions were obtained by importing the solid models in AutoCAD. These dimensions were compared to ones from independently produced, on-site manual surveys (Figure 16f, 17f and 18f).

In general, the FV algorithm reconstructed building models more accurately than the other approaches. The maximum relative errors of the overall dimensions (height and length) were 1.1% (a length of Building 2) for the FV algorithm, and 1.9% (a length of Building 2) for other automatic approaches and -3.0% for the Kubit program (Figure 19a and b). Furthermore, the relative error of opening areas in FV-based models were also smaller than other approaches, in which the maximum error was 3.3% in FV-based models, while it was 5.6% for other automatic approaches-based models and 7.2% in the Kubit-based ones. However, in Building 3, the relative error in the FV-based model was slightly higher than those in the other automatic-based models (-3.0% in FV-based model vs. -2.9% in FA-based ones). That is because boundary points of the openings derived from FV algorithm contained sample points that were slightly further from the actual boundaries of the opening. When working with lower density datasets, the accuracy of the openings can be improved by increasing the number of neighbouring points. For this model and this density the kNN was increased to 20, which was equal to that used in the FA algorithm.



a) Facade length b) Facade height c) Façade opening area

Figure 20. Relative errors of overall dimensions between CAD drawings versus the FV, FD, FA algorithms and Kubit-based solid models

Other geometric discrepancies were checked in terms of the opening dimensions. In two small buildings (Building 1 and 2), the average absolute errors of opening dimensions in the building models derived from the FV algorithm were generally smaller than those from the other three approaches, while an opposite trend was found in the larger building (Building 3) (Table 4). As such, the best result from the FV algorithm was -4.6 mm (Std = 116 mm), while ones from other automatic algorithms and commercial software were 7.5 mm (Std = 121.5 mm) and 4.7 mm (Std = 61.2 mm), respectively. However, in Building 3, the error with the FV-based model was slightly higher than ones from the automatic algorithms. So while the FV algorithm herein did not significantly improve the geometric accuracy of the building models, it did provide a major reduction in computational time, as is presented in Figure 21.

Table 4. Dimensional discrepancies of openings between CAD drawings versus the point cloud based solid models

Aspects	Building 1				Building 2				Building 3			
	Automatic			Semi-automatic	Automatic			Semi-automatic	Automatic			Semi-automatic
	FV	FD	FA	Kubit	FV	FD	FA	Kubit	FV	FD	FA	Kubit
Av., mm	-4.6	-6.4	-14.0	28.8	7.5	11.0	22.8	75.6	-28.3	-14.5	-23.8	4.7
Min. error, mm	-139.0	-137.0	-149.0	-110.0	-126.0	-168.0	-130.0	-40.0	-177.0	-162.0	-177.0	-140.0
Max. error, mm	187.0	175.0	178.0	250.0	366.0	360.0	354.0	370.0	117.0	39.0	45.0	120.0
Std., mm	116.0	118.1	122.1	110.0	121.5	121.9	126.1	102.7	56.9	50.1	54.1	61.2

Using all of the datasets in Table 1, the FV algorithm proved clearly superior for datasets greater than 35.5 k points, as was required for even the lease dense version of the 19.36m wide by 17.0m high Building 3 (Figure 20). In very dense datasets such as those directly obtained from the laser scanner, the FV algorithm was consistently more than an order of magnitude faster and was up to 25 times faster for the 650.3k point dataset.

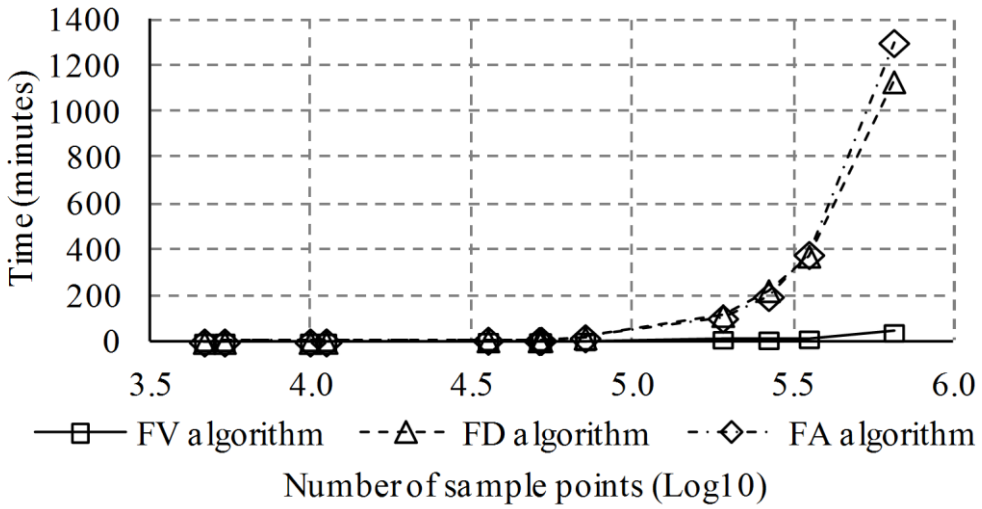


Figure 21. Processing time of FV algorithm vs. of FD and FA algorithms data from Truong-Hong et al. [6, 7]

5. CONCLUSIONS AND FUTURE WORK

The FacadeVoxel algorithm is proposed as a rapid and robust, fully-automated approach to detect boundary points of facade features for many urban structures that otherwise lack existing geometric documentation. This is done by examining only sample points in the

vicinity of detected boundary features. Sample points within full voxels around a boundary feature are extracted and examined based on a local score. Subsequently, these boundary points are used to generate fitting boundary lines, and all voxels belonging to a solid portion of the wall are determined. Afterwards, the complete solid model of the building was described as a neutral model as input for a commercial computational modelling programme. The algorithm consistently detected all openings for three sample buildings and also succeeded in automatically filling non-openings, even at data densities as low as 175 pts/m^2. This was achieved without any supplemental datasets, user knowledge, or manual intervention. Furthermore, the resulting solid models were fully compatible with a common, commercial FEM package. When compared to two previous research-based approaches and a commercial program, building models based on the FV method were generally more accurate, and up to 25 times faster for datasets of up to 650.3k points.

Like the three comparative approaches, the FV algorithm is currently limited to relatively simple 2D morphologies. A full 3D model can be obtained theoretically by segmentation and subsequent reassembly of individual exterior walls. Future work is planned in this direction. Additionally, further work is needed to address non-rectangular window openings.

ACKNOWLEDGMENTS

This work was generously supported by Science Foundation Ireland Grant 05/PICA/I830 and the European Union Grant ERC StG 2012-307836-RETURN. Further thanks to Donal Lennon of UCD's Earth Institute for his assistance with terrestrial data acquisition.

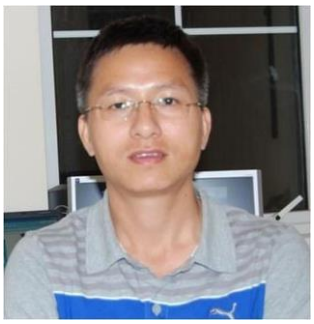
REFERENCES

[1] N. Haala, M. Kada, An Update on Automatic 3D Building Reconstruction, ISPRS Journal of Photogrammetry and Remote Sensing, 65 (2010) 570-580.
[2] H.J. Koelman, Application of a photogrammetry-based system to measure and re-engineer ship hulls and ship parts: An industrial practices-based report, Computer-Aided Design, 42 (2010) 731-743.
[3] D.-J. Yoo, Three-dimensional surface reconstruction of human bone using a B-spline based interpolation approach, Computer-Aided Design, 43 (2011) 934-947.
[4] M. Peternella, T. Steinerb, Reconstruction of piecewise planar objects from point clouds, Computer-Aided Design, 36 (2004) 333–342.
[5] E.H. Lim, D. Suter, 3D terrestrial LIDAR classifications with super-voxels and multi-scale Conditional Random Fields, Computer-Aided Design, 41 (2007) 701-710.
[6] L. Truong-Hong, D. Laefer, T. Hinks, H. Carr, Flying Voxel Method with Delaunay Triangulation Criterion for Façade/Feature Detection for Computation, ASCE, Journal of Computing in Civil Engineering, 26 (2012) 691-707.
[7] L. Truong-Hong, D.F. Laefer, T. Hinks, H. Carr, Combining an angle criterion with voxelization and the flying voxel method in reconstructing building models from LiDAR data, Computer-Aided Civil and Infrastructure Engineering, 28 (2012) 112-129.
[8] RPA, Environmental impact statement-metro north, in, 2011.
[9] J. Clarke, D.F. Laefer, Generation of a building typology for risk assessment due to urban tunneling, in: BCRI conference Dublin, Ireland, 2012.
[10] F. Cassidy, Port Tunnel compo payout, in: Independent, Dublin, 2006.
[11] S. Pu, G. Vosselman, Knowledge based reconstruction of building models from terrestrial laser scanning data, ISPRS Journal of Photogrammetry and Remote Sensing, 64 (2009) 575-584.
[12] S. Becker, N. Haala, Refinement of Building Facades by Integrated Processing of LIDAR and Image Data, PIA07 - Photogrammetric Image Analysis, Munich, Germany, 19-21 September, 2007 (2007) 36(33/W49A), 37-12.
[13] Q.-Y. Zhou, U. Neumann, 2.5D Dual Contouring: A Robust Approach to Creating Building Models from Aerial LiDAR Point Clouds, in: 11th European Conference on Computer Vision (ECCV 2010), Crete, Greece, 2010, pp. 1-14.
[14] N. Haala, C. Brenner, K.h. Anders, 3D Urban GIS From Laser Altimeter And 2D Map Data, International Archives of Photogrammetry & Remote Sensing, 32 (1998) 339-346.
[15] J. Hu, S. You, U. Neumann, K.K. Park, Building Modeling from LiDAR and Aerial Imagery, in: ASPRS 2004, Denver, Colorado, USA, 2004, pp. 1-6.

[16] G. Vosselman, S. Dijkman, 3D Building Model Reconstruction from Point Clouds and Ground Planes, in: International Archives of Photogrammetry and Remote Sensing, Annapolis, MA, USA, 2001, pp. 37-43.
[17] P. Dorninger, N. Pfeifer, A Comprehensive Automated 3D Approach for Building Extraction, Reconstruction, and Regularization from Airborne Laser Scanning Point Clouds, Sensors, 8 (2008) 7323-7343.
[18] J. Chen, B. Chen, Architectural Modeling from Sparsely Scanned Range Data, Journal International Journal of Computer Vision, 78 (2008) 223-236.
[19] Kubit, PointCloud, in, 1999.
[20] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, B. Chen, SmartBoxes for interactive urban reconstruction, Journal ACM Transactions on Graphics, 29 (2010) Article 93.
[21] S. Pu, G. Vosselman, Extracting windows from terrestrial laser scanning, in: ISPRS Workshop on Laser Scanning and SilviLaser 2007, Espoo, Finland, September 12-14, 2007, 2007, pp. 320-325.
[22] H. Boulaassal, T. Landes, P. Grussenmeyer, Automatic extraction of planar clusters and their contours on building façades recorded by terrestrial laser scanner, International Journal of Architectural Computing, 7 (2009) 1-20.
[23] N. Haala, S. Becker, M. Kada, Cell decomposition for the generation of building models at multiple scales in: Symposium of ISPRS Commission III, Photogrammetric Computer Vision PCV ' 06 Bonn, Germany, 2006, pp. 6.
[24] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, ACM SIGGRAPH, (1992) 71-78.
[25] J. Wang, M.M. Oliveira, H. Xie, A.E. Kaufman, Surface reconstruction using oriented charges, in: Computer Graphics International, Stony Brook, New York, USA, June 22-24, 2005, 2005, pp. 122-128.
[26] M.D. Buhmann, Radial Basis Functions: Theory and Implementations Cambridge University Press Cambridge, 2003.
[27] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: The 23rd annual conference on Computer graphics and interactive techniques, ACM, New Orleans, LA, USA, 1996, pp. 303-312.
[28] K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, W. Stuetzle, Robust Meshes from Multiple Range Maps, in: International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, Canada, 1997, pp. 205-211.

[29] J. Elseberg, D. Borrmann, A. Nuchter, One billion points in the cloud – an octree for efficient processing of 3D laser scans, ISPRS Journal of Photogrammetry and Remote Sensing, 76 (2013) 76-88.
[30] K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems, in: ICRA 2010 workshop Best Practice in 3D Perception and Modeling for Mobile Manipulation, Alaska, USA, 2010, pp. 8.
[31] P. Dalmasso, R. Nerino, Hierarchical 3D surface reconstruction based on radial basis functions, in: Proceedings of 2nd International Symposium on 3D Data Processing, Visualization and Transmission, Thessaloniki, Greece, 2004, pp. 574-579.
[32] R. Wang, J. Bach, F.P. Ferrie, Window detection from mobile LiDAR data, in: IEEE Workshop on Applications of Computer Vision, Kona, Hawaii, 2011, pp. 58-65.
[33] D.F. Laefer, L. Truong-Hong, M. Fitzgerald, Processing of Terrestrial Laser Scanning Point Cloud Data for Computational Modelling of Building Facades, Recent Patents on Computer Science, 4 (2011) 16-29.
[34] L. Truong-Hong, D.F. Laefer, Validating Computational Models from Laser Scanning Data for Historic Facade, Journal of Testing and Evaluation, 41 (2013) 481-496.
[35] H. Samet, R.E. Webber, Hierarchical Data Structures and Algorithms for Computer Graphics. Part I: Fundamentals, IEEE Comput. Graph. Appl., 8 (1988) 48-68.
[36] Z. Tang, Octree representation and its applications in CAD, Journal of Computer Science and Technology, 7 (1992) 29-38.
[37] D. Ayala, P. Brunet, R. Juan, I. Navazo, Object representation by mean of mominal division duadtrees and octrees, ACM Transaction on Graphics, 4 (1985) 41-59.
[38] N. Ripperda, Determination of Facade Attributes For Facade Reconstruction, in: ISPRS Congress, Proceedings of Commission III, Beijing, China, 2008, pp. 285-290.
[39] M.K. Agoston, Computer Graphics and Geometric Modeling: Implementation & Algorithms, Springer Verlag London Limited, 2005.
[40] G.T. Toussaint, Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining, International Journal of Computational Geometry and Applications, 15 (2005) 101-150.
[41] H. Mayer, S. Reznik, Building Façade Interpretation From Image Sequences, in: R.F. Stilla U, Hinz S (Ed.) CMRT05. IAPRS, Vienna, Austria, 29-30 August, 2005, 2005, pp. 55-60.

[42] F. Pighin, J.P. Lewis, Practical least-squares for computer graphics, ACM SIGGRAPH 2007 courses, (2007) 1-57.
[43] Trimble, GS200 3D Scanner, in, 1999.
[44] Trimble, RealWorks Survey: Technical notes-RealWroks Survey, in, 2005.
[45] L. Truong-Hong, Automatic Generation of Solid Models of Building Façades from LiDAR Data for Computational Modelling, in: School of Architecture, Landscape and Civil Engineering, University College Dublin, 2011.
[46] The MathWorks, MATLAB Function Reference, in, 2007.
[47] H. Edelsbrunner, D.G. Kirkpatrick, R. Seidel, On the shape of a set of points in the plane, IEEE Transactions on Information Theory, 29 (1983) 551-559.



Linh Truong-Hong is a research fellow at Urban Modelling Group, University College Dublin, Ireland. He received PhD in Structural Engineering from University College Dublin in 2011. His research interests include geometric modelling, structural health monitoring, soil-structure interaction modelling and structural modelling.

Debra F. Laefer is an associate professor at the University College Dublin where she leads the Urban Modelling Group. Her research interests include protection of the built environment from natural and manmade activities. In recognition of her contributions, she is a recent recipient of a European Research Council award in this area.