Digital Twinning of Existing Reinforced Concrete Bridges from Labelled Point Clusters

Ruodan Lu[1,2]* A.M. ASCE; Ioannis Brilakis[3], M. ASCE

*corresponding author

[1]School of Architecture, Building and Civil Engineering, Loughborough University, United Kingdom, E-mail: r.lu@lboro.ac.uk,

[2]Darwin College, University of Cambridge, United Kingdom, E-mail: rl508@cam.ac.uk

[3]Laing O'Rourke Reader, Department of Engineering, University of Cambridge, United Kingdom, E-mail: ib304@cam.ac.uk

Abstract

The automation of digital twinning for existing reinforced concrete bridges from point clouds remains an unresolved problem. Whilst current methods can automatically detect bridge objects in point clouds in the form of labelled point clusters, the fitting of accurate 3D shapes to point clusters remains largely human dependent largely. 95% of the total manual modelling time is spent on customizing shapes and fitting them correctly. The challenges exhibited in the fitting step are due to the irregular geometries of existing bridges. Existing methods can fit geometric primitives such as cuboids and cylinders to point clusters, assuming bridges are comprised of generic shapes. However, the produced geometric digital twins are too ideal to depict the real geometry of bridges. In addition, none of the existing methods have explicitly demonstrated how to evaluate the resulting Industry Foundation Classes bridge data models in terms of spatial accuracy using quantitative measurements. In this article, we tackle these challenges by delivering a slicing-based object fitting method that can generate the geometric digital twin of an existing reinforced concrete bridge from four types of labelled point cluster. The quality of the generated models is gauged using cloud-to-cloud distance-based metrics. Experiments on ten bridge point cloud datasets indicate that the method achieves an average modelling distance of 7.05 cm (while the manual method achieves 7.69 cm), and an average modelling time of 37.8 seconds. This is a huge leap over the current practice of digital twinning performed manually.

Key words: Digital Twin, IFC, BrIM, BIM, Point Cloud Data

1 Introduction

Highway authorities have a duty to manage and maintain the majority of bridges. Therefore, it is crucial that bridge management minimizes disruption, risk and consequent costs to road users and makes economic and efficient use of resources (FHWA, 2012). However, every year, the United States (US) spends roughly \$12.8 billion to address deteriorating bridge conditions (ASCE, 2013). The reasons behind these massive costs are in part because bridge owners face a major challenge with structuring and managing the data needed for rapid repair, maintenance, and retrofit of their bridges. The data available in Bridge Management Systems (BMS) does not meet the standard of information needed for sound decision-making (ASCE, 2017). There is a need for at least 315,000 bridge inspections per annum across the US and the UK, given the typical two-year inspection cycle (ASCE, 2017; Network Rail, 2015). Visual inspection is still the most common form of condition monitoring. The resulting physical condition information from the visual assessment is then entered into a BMS, such as the US's AASHTOWare (AASHTOWare, 2018) or the UK's NATS (Flaig & Lark, 2000), to rate the deterioration of the bridge. However, these BMSs are geared primarily to make system-wide prioritization decisions based on high-level comparisons of condition data (Vassou, 2010). They do not assess the actual condition of a particular bridge component and of a particular location of the component. Having a Geometric Digital Twin (gDT) would be quite useful for this purpose as texture and damage information can then be properly integrated with the geometry at the component-level of the virtual 3D representation of a bridge (Hüthwohl et al., 2018).

A Digital Twin (DT) is defined as a digital replica of a real-world asset (Parrott & Lane, 2017). It differs from, and is much more than, traditional Computer-Aided Design. A DT is based on massive, cumulative, real-time, real-world data measurements across an array of dimensions, and the consequent use of a digital model across the entire lifecycle of an infrastructure (Buckley & Logan, 2017). The model comprises 3D geometry of the infrastructure components as well as a comprehensive set of semantic information, including materials, functions, and relationships between the components. The use of a DT is greatest during the design stage, while little use is made in the closeout stage, and is almost absent in the maintenance stage (as-is) (Buckley & Logan, 2017). Hereafter, the "DT" specifically refers to the "as-is DT", generated for existing infrastructure, except as otherwise noted. Bridge owners today do not generate DTs for existing bridges, because they perceive the cost of doing so to outweigh their benefits. The fundamental feature of DTs is the 3D geometry, without which many DT applications do not exist. We use the adjective "geometric" (gDT) to highlight a DT with only geometry data, i.e. gDT. The following texts review the current practice of digital twinning from point clouds, i.e. the process to acquire a gDT for an existing asset. This explains why the DT implementation is so limited.

Major vendors such as Autodesk, Bentley, Trimble, AVEVA and ClearEdge3D, and so on, provide the most advanced digital twinning software solutions. For example, ClearEdge3D (2017) can automatically extract pipes in a plant point cloud as well as specific standard shapes like valves and flanges from industry catalogues followed by fitting built-in models to them through a few clicks and manual adjustments. This means ClearEdge3D can realize a certain degree of automation. However, the spec-driven component library of ClearEdge3D can only fit point cloud subparts with standardised shapes such as rectangular walls, pipes, valves, flanges, and steel beams, based on an industry specification table. Other commercial applications cannot automate the fitting task for either generic or arbitrary shapes. Modellers must manually fit 3D shapes to the segmented point cloud subparts. Fitting accurate 3D shapes to the point clusters is challenging because the set of allowable primitives is limited in most software applications (Wang et al., 2015). Real-world reinforced concrete (RC) bridge components usually have complicated shapes, containing complex skews and imperfections, and cannot be simply fitted using idealized generic shapes. Modellers must manually create an accurate solid form to fit each point cluster as none of the existing software packages can do this automatically. Modelling software such as Revit provides a high degree of flexibility that allows users to design a shape in a freeform manner via Revit's Family editor (Figure 1). The so-called "families" are parametrized object types controlled by parameters, constraints, and dependencies. Modellers first draw a 2D sketch assigned with geometric and dimension constraints. Then, the 2D sketch is used for extruding or rotating to produce a final parametric 3D model. Features (Sacks et al., 2018; Sacks et al., 2004), such as chamfers in a pier, windows in a wall, and connections on a steel beam, can also be added. Although parametric modelling is powerful, a well-designed modelling plan is required due to the ambiguous and complex nature of parametric modelling (Lee et al., 2006). 95% of the total modelling time is spent on customizing shapes and fitting them to point clusters (Lu & Brilakis, 2017).

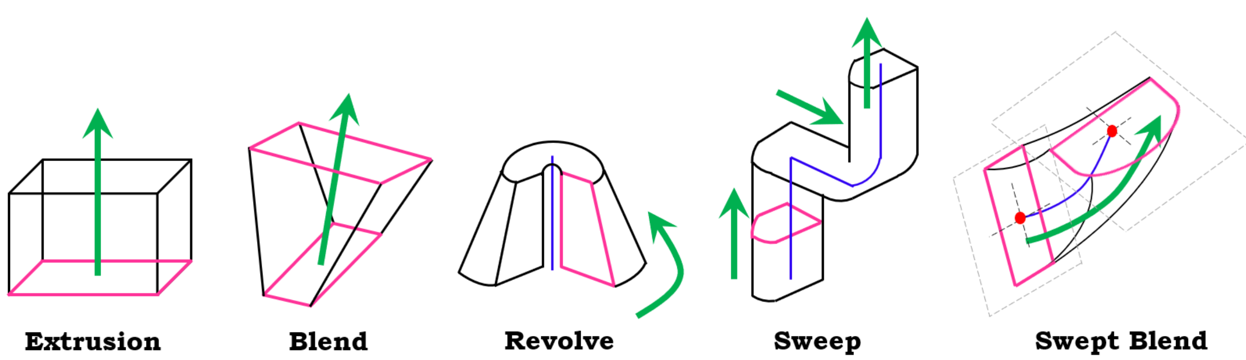


Figure 1 Forms available in Revit Family editor

In this paper, we propose to tackle this challenge with a novel automatic fitting method to generate gDTs.

It follows a slicing strategy to generate 3D shapes using an established data format, Industry Foundation Classes (IFC), followed by fitting them to the labelled bridge point clusters. The novelty of this method lies in the fact that multiple local topological configurations derived from the slicing scheme provide good characterization to approximate the global topology of the underlying bridge in a point cloud. We provide a review of existing work in Section 2 and outline the proposed method in Section 3. We then elaborate on the experiments in Section 4. Finally, we interpret the results and draw conclusions in Section 5.

2 Background

The use of existing software packages for digital twinning of existing bridges is human dependent to a great extent. Unlike building geometries which are generally developed in a grid system (Thomson & Boehm, 2015), real-world bridge geometries are defined with curved alignments, vertical elevations, and varying cross-sections (Wai-Fah & Lian, 2014). Extensive manual effort is required for practitioners to manually customize 3D accurate models to fit underlying bridge components to arbitrary shapes. We define "model fitting" in this context as leveraging computer graphic techniques to form the 3D shape of a point cluster, a subpart of a point cloud. The 3D shape is approximate, in the sense that it describes the geometry or the shape of a point cluster to produce its digital 3D representation to an acceptable quality based on the specific required level of detail.

There is no universal solution to describe a 3D object. Different representation methods have their advantages and disadvantages. How to choose a representation depends on (1) the nature of the object being modelled, (2) the particular modelling technique that we choose to use, and (3) the application scenario where we bring the object to life. The most commonly used existing shape representation methods can be categorized into four groups: Implicit Representation, Boundary Representation, Constructive Solid Geometry, and Swept Solid Representation. The following texts describe each in turn.

Implicit Representation is a solid modelling approach, which is based on the representation of 3D shapes using mathematical formulations, i.e. implicit functions. For example, a point cluster can be described as a plane (Limberger & Oliveira, 2015), a sphere, a torus (Schnabel et al., 2007), and so on. Implicit shape representations have difficulty with describing sharp features such as edges and vertices, although they can check whether a point lies inside, outside, or on the surface (Song & Jüttler, 2009). Given that only a very limited number of primitives can be represented exactly by algebraic formulations, implicit functions are of limited usefulness when modelling bridge components, as they usually do not take idealized shapes. In addition, the as-weathered and as-damaged condition of a bridge further reduces the effectiveness of implicit representations. There is a trade-off between the accuracy of the representation and the bulk of information used for shapes that cannot be represented by mathematical formulations. We present three other shape representation methods in the following texts.

Boundary Representation (B-Rep) is a method that describes shapes using their limits. The model represented using B-Rep is an explicit representation, as the object is represented by a complicated data structure giving information about each of the vertices, edges, and loops and how they are joined together to form the object. Both Tessellated Surface Representation (TSR) and Polygon/Mesh Representation can be considered as types of B-Rep. For example, a flat quadrilateral is made up of four vertices joined by four straight lines or a bi-cubic parametric patch (Zhang et al., 2015). A curvilinear quadrilateral is made up of four vertices joined by four cubic curves (Dimitrov et al., 2016). Kwon et al. (2004) introduced a local spatial modelling algorithm to fit planes, cuboids, and cylinders to point clouds in B-Rep, assuming that a construction site consists of these primitives. Valero et al. (2012) developed a method to yield B-Rep models for indoor planar objects (e.g. walls, ceilings, and floors). Oesau et al. (2014) leveraged a graph-cut formulation to reconstruct a synthetic building point cloud into a mesh-based model. However, simply representing an object embedded in point clouds using TSR or polygon facets/mesh is still a low-level machine representation, although it is the most popular representation in computer graphics. Problems with polygon mesh B-Rep models include (1) Level of detail. High-resolution results can be unduly complex and unnecessary. An option is to reduce the polygon resolution without degrading the rendered presentation (Chen et al., 2017). However, by how much should it be reduced? (2) Occlusions. Large occluded regions are hardly smoothed so that PR/MP does not guarantee a group of polygons facets can form a closed mesh model (Carr et al., 2003).

Constructive Solid Geometry (CSG) is a high-level volumetric representation that works both as a shape representation and a record of how an object was built up (Deng et al., 2016). The final shape can be represented as the combination of a set of elementary solid primitives, which follow a certain "logic". The primitives can be cuboids, cylinders, spheres, cones, and so on. When building a model, these primitives are created and positioned, then combined using Boolean set operators such as union, subtract, intersect, and so on. The methods proposed by Rabbani (2006) and Patil et al. (2017) can be used for modelling piping systems using generic shapes, such as cylinders. The random sampling method of Schnabel et al. (2007) can be used to model objects composed of five basic shapes: plane, sphere, cylinder, cone, and torus. Walsh et al. (2013) developed a shape library containing generic objects (e.g. cuboid, cylinder) to fit point clusters using surface fitting in the least squares sense. Rusu et al. (2008) proposed a model fitting module to fit kitchen objects (e.g. cupboards and appliances) using 3D cuboids. Similarly, Xiao and Furukawa (2014) introduced an algorithm called "inverse CSG" to reconstruct large-scale indoor environments using cuboids, assuming that they are the most common shapes found in indoor walls. Zhang et al. (2014) designed a multi-class Adaboost decision tree classifier from surface primitive features to classify both infrastructure components (pier, beam, deck, etc.) and 3D shape entity labels (cuboid, cylinder, sheet, etc.) (Figure 2). However, this method is tailored for idealized or simplified topology designs that do not consider the real geometries of bridge components. For example, a real sloped slab with varying vertical elevation cannot be simply modelled by a single sheet. Modelling non-generic shapes using the CSG approach demands a well-thought-out modelling plan. We thus contend that CSG is less suitable for representing real bridge components, which are more complex than simple primitives, such as cuboids and cylinders.

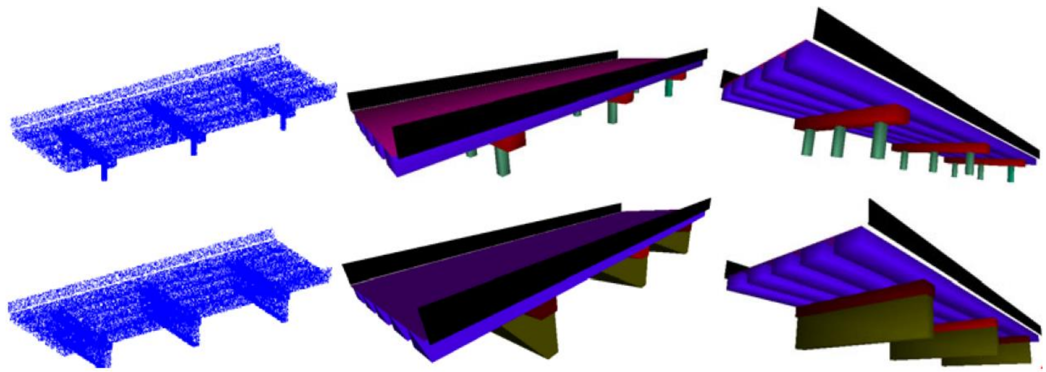


Figure 2 Fitted IFC entities in synthetic bridge point clouds (Zhang et al., 2014)

Swept Solid Representation (SSR) or Extrusion is a representation model which creates a 3D shape by sweeping a 2D profile that is completely enclosed by a contour line along a specific path in the third dimension. Budroni and Böhm (2010) suggested a plane-sweep-based method to extrude planar elements (e.g. walls) in indoor environments. Similarly, Ochmann et al. (2016) presented an approach for reconstructing parametric planar building elements from indoor point clouds. Thomson & Boehm (2015) extruded the footprint of office walls by specifying the length, width, and height. The reconstructed geometry was compared against the reference model using quality metric, which, however, was specifically designed for walls in cuboid shapes. Laefer & Truong-Hong (2017) introduced a kernel-density-estimated-based method to reconstruct standardized steel beams in point clouds. The sweeping approach has been studied in building/industry settings to generate cuboids or standardised beams. Its implementation has not yet been investigated for twinning bridge elements.

IFC Geometric Representation

In order to support the use of a gDT in the construction industry, all the associated geometric and property information should be represented in platform-neutral data format, i.e. IFC. This section focuses on the principles involved in representing IFC geometry and the most important geometry representations. According to Borrmann et al. (2018), all geometry representations in IFC data model can be grouped into four classes: 1) Bounding Boxes; 2) Curves; 3) Surface models; and 4) Solid models. Bounding Boxes can be represented using *IfcBoundingBox*. Bounding Boxes are highly simplified geometric representations for 3D objects that are often used as placeholders. *IfcBoundingBox* is defined by a placement corner point and the dimensions of the three sides as a cuboid. Then, *IfcCurve* and its subclasses *IfcBoundedCurve*, *IfcLine*, and *IfcConic* can be used to model line objects. Freeform curved edges (splines) and curved surfaces are required to model sophisticated and complex geometries. A freeform 3D curve is mathematically described as parametric curves, meaning that the x, y, z coordinates are functions tracing a 3D curve at common parameters. Next, surface models are used to represent composite surfaces comprised of sub-surfaces. They can be curved surfaces (e.g. NURBS) or flat surfaces (e.g. mesh). TSR is a very simple geometric representation that can be interpreted by almost all visualization software applications. *IfcTriangulatedFaceSet* can be used to represent the tessellated surfaces, i.e. polygons with an arbitrary number of edges, or triangular mesh. TSR cannot represent curved surfaces ideally but approximates them into triangular facets. In this case, the curved surface can be described using a finer mesh size if accuracy is a concern. *IfcBSplineSurface* can be used to represent curved surfaces, such as NURBS surfaces. One classic way to generate 3D objects as solid models is through the CSG approach. *IfcCsgPrimitive3D* and its subclasses such as *IfcBlock*, *IfcRightCircularCylinder*, *IfcSphere*, and so on can be used. Combination operations can be performed using *IfcBooleanResult*. However, as previously mentioned, the use of CSG is very limited due to the fact that the use of primitives is very restrictive. By contrast, SSR (or Extrusion) is widely used for creating 3D objects in IFC. Possible representations include, but are not limited to, the classes summarized in the following. In general, *IfcSweptAreaSolid* and its subclasses *IfcExtrudedAreaSolid*, *IfcRevolvedAreaSolid*, *IfcFixedReferenceSweptAreaSolid*, and *IfcSurfaceCurveSwptAreaSolid* can be used to present extruded solids. A closed profile *IfcArbitraryClosedProfileDef*, which is the most common subclass of *IfcProfileDef*, is necessary for this representation. For example, when using *IfcExtrudedAreaSolid*, the ExtrudedDirection is defined so that *IfcArbitraryClosedProfileDef* can be extruded along the direction. When using *IfcRevolvedAreaSolid*, both ExtrudedDirection and the axis are defined so that *IfcArbitraryClosedProfileDef* can rotate around the axis up to a given angle. Then, *IfcFixedReferenceSweptAreaSolid* allows the extrusion to be done along any curve in space through the attribute Directrix. That is to say, the profile is extruded along a specific axis defined by the attribute FixedReference.

Gaps in knowledge, Objectives, and Research Questions

Digital twinning for existing assets using point clouds is still in an early stage. Existing methods concentrate on generating building and industrial components, such as walls, ceilings, floors, and standardized industrial elements. These objects are simply represented as planar elements, cuboids, and cylinders using a set of limited constraints. The problem of fitting 3D solid models in IFC format to real bridge point clusters in non-standardized shapes has yet to be addressed. In addition, no standardized metric has been specified for the quantitative evaluation of the resulting gDTs.

We aim to fill the above-mentioned knowledge gaps by delivering a method that can automatically fit 3D solid models in IFC format to labelled point clusters making up a real-world RC bridge. We also gauge the quality of the generated gDTs using distance-based metrics, which can be applied to other infrastructure types other than bridges. These objectives are achieved by answering the following research questions: (1) how to extract and use the geometric features to reconstruct the labelled bridge point clusters in arbitrary shapes into 3D solid models in IFC format? and (2) how to evaluate the spatial accuracy of a bridge gDT reconstructed from a point cloud?

Hypothesis

The hypothesis of this research is that the slicing-based bridge-component fitting method can generate high-quality gDT of an existing RC bridge in IFC format and there is no significant difference in the spatial accuracy for different RC bridges. In addition, the twinning time is much less compared to the manual practice. This hypothesis will be tested with a point cloud dataset of ten highway RC bridges in the UK.

3 Proposed Solution

3.1 Scope

We focus on typical RC slab and beam-slab bridges because 73% of existing highway bridges and 86% of planned future bridges are of these two types (Kim et al., 2016). We only deal with the four most important and highly detectable components of the two types of bridges: slab, pier, pier cap, and girder (Kedar, 2016). In addition, we focus only on the non-textured geometric representation part of the bridge DT, including the semantic meaning of its components, namely a labelled bridge gDT. The enrichment of other semantic information such as materials, defects, additional relationships, and so on, are beyond the scope of this research.

3.2 Overview

Figure 3 illustrates the workflow of the proposed method. We assume that the object detection task is properly done. This means that the inputs of the proposed method are four types of labelled point cluster, namely the outputs of the authors' previous work (Lu et al., 2018). The output of this paper is an IFC file, containing various *IfcObjects* making up a bridge gDT and corresponding to a level of detail LOD 250 – 300. The method consists of two major steps: Step 1, geometric feature extraction and shape detection in the four types of component point cluster; and Step 2, *IfcObjects* fitting for the extracted features and identified shapes. Defining and specifying the level of geometric detail required for twinning gDTs in accordance with the end user requirements is beyond the scope of this research. Thus, we generate a bridge gDT based on the existing very broad guidance (Table 1) such that it is flexible to adapt to current and future needs. As shown, LOD 200 uses a bounding box to represent each component. It is a coarse representation, meaning that all components are represented as generic placeholders with approximate geometry. Thus, it cannot fully support the construction course and the post-construction process. The LOD increases as the project requirement proceeds. A LOD 300 gDT is graphically represented as a specific system, object, or assembly accurate in terms of size, shape, location, and so on. Note that, LOD 300 does not include information such as detailing, fabrication, installation, and detailed assemblies, which are necessary to reflect the actual status of existing infrastructure (Table 1). LOD 350 and higher LODs contain enriched information that reflects the as-is status of existing infrastructure. However, various additional sensors are required to capture this embedded information that is invisible to a laser sensor. Extracting this information is beyond the scope of this research. We therefore only focus on generating a LOD that can be achieved through laser scanning alone. In this paper, the method generates a bridge gDT with a LOD that is higher than LOD 200 but may not be fully in line with LOD 300, as some components may be represented in a stacked way (e.g. pier). Thus, we use LOD 250 – 300 to denote the expected LOD of the output gDT. Specifically, the geometry of a slab point cluster is approximated using multiple oriented slice models along with its horizontal alignment. The geometry of a pier cap point cluster is represented by extruding its projected outline. For a pier point cluster, the method first checks its shape and then decides whether to represent it as a generic shape primitive or to represent it using stacked slices. Last, for a girder point cluster, the method uses a template matching method to fit it with a specific profile from a precast concrete catalogue. The proposed method uses current IFC standards, aggregation relationship, and the Model View Definition suggested by Sacks et al. (2018) to encode geometric features taken to describe a bridge component. The expected contribution of the proposed method is that it is the first method of its kind to efficiently generate an accurate gDT in IFC format using labelled point clusters making up an existing RC bridge.

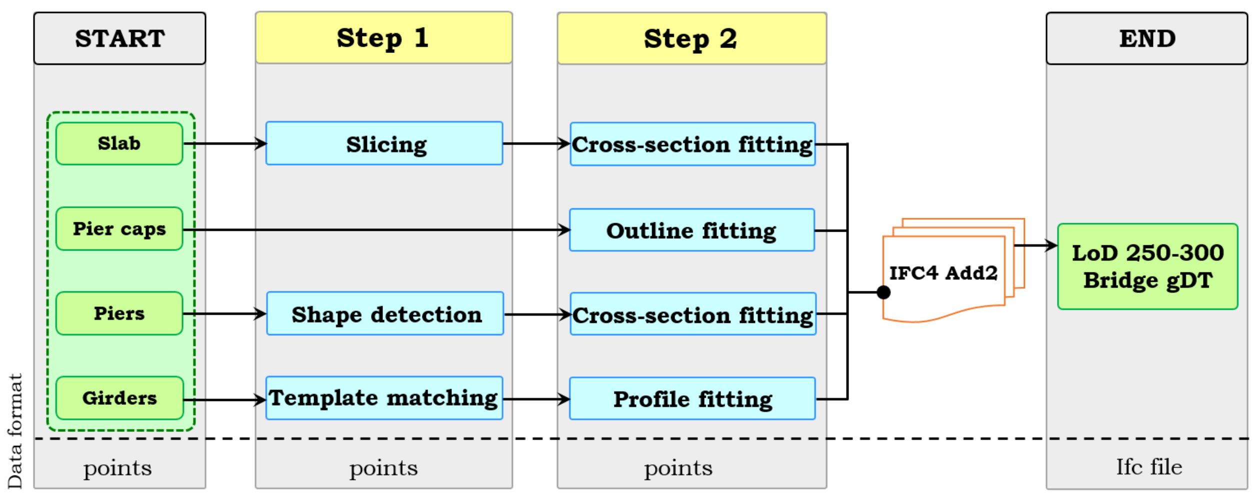
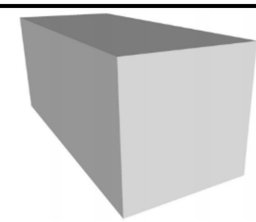
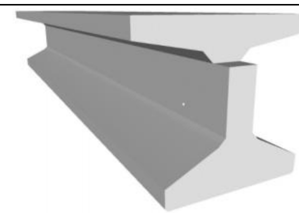
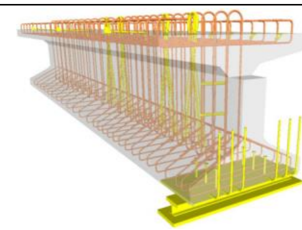


Figure 3 Workflow of the proposed method

Table 1 LOD Specification for Highway Bridge Precast Structural Girder (BIMForum, 2018)

LOD	Interpretation	Schema
200	Elements are generic placeholders. They may be recognizable as the components they represent, or they may be volumes for space reservation. Any information derived from the elements must be considered approximate.	
300	The quantity, size, shape, location, and orientation of the element as designed can be measured directly from the model without referring to non-modelled information.	
350	Parts necessary for coordination of the element with nearby or attached elements are modelled. These parts will include such items as supports and connections.	

3.3 LOD 250 – 300 gDT generation

In this twinning phase, a bridge is represented by the four types of point cluster with detailed geometries. We first assign a specific IFC entity to one corresponding point cluster based on its semantic label. Specifically, *IfcSlab* is used for slabs, *IfcBeam* for both pier caps and girders, and *IfcColumn* for piers. We use SSR (or Extrusion) to create the stacked slice models for each component. Solid extrusions are preferred wherever possible if one dimension of a component is larger than the other two, or if each extruded cross-section is deemed to be constant. The general thrust behind the LOD 250 – 300 representation is that the geometry of a bridge component can be approximated using multiple stacked slices. This stems from Cavalieri's principle (Kern & Bland, 1948), which serves as the theoretical guidance of our method. We elaborate on how to twin each of the four types of point cluster in the following texts.

3.4.1 Slab – *IfcSlab*

The topology of a bridge usually depends on its horizontal and vertical alignment, such as the straightness and flatness of the deck. Real-world bridges are neither straight nor flat. To circumvent or be compatible with the existing constraints of road geometry, many highway bridges carrying roads are on a curved alignment and the supporting structure follows that curved alignment (Highways England, 2018b). The presented method aims to approximate the real horizontal (and/or vertical) alignment by using multiple straight segments, such that different gap-freedom horizontal alignment segments can be concatenated to a single horizontal alignment, with the same also true for the vertical alignment. This information can be assigned in the future into the *IfcAlignment* entity as the list of slab segments generated from the proposed method can deduce the necessary information required for *IfcAlignment*.

According to Kobryń (2017), we assume that a circular curve is used for the horizontal alignment of bridges investigated in this research, such that the general function of the horizontal alignment is a degree two parabola. This assumption is based on the highway bridge design rule that it is preferable to locate bridges on the tangent positions of the alignment. Large horizontal curves should be avoided on bridges whenever possible. Yet, often, it is necessary to locate a bridge on a curve due to road geometry and on-site constraints. Where a curve is necessary, a simple curve should be used on the bridge and any necessary curvature or super-elevation transitions ought to be placed on the approaching roadway (Highways England, 2018a).

We use a similar but not identical slicing method to that proposed in (Lu et al., 2018) to slice the deck slab into J slices. The slicing does not take a parallel pattern but is rather oriented along the normal direction of the curved alignment. The deck slab point cluster normally contains most of the scanned points of an entire bridge point cloud, attributed to its large upper and bottom surface being exposed to the laser sensor. We use only 10% of them being randomly chosen for fitting a parabola. To this end, we project the randomly down-sampled slab point cluster onto the XY-plane followed by fitting a unique second-degree polynomial to the projected n points (x_i, y_i) by minimizing the square error, provided that the X-axis is the principal direction (Lu et al., 2018):

$$E = \sum_{i=0}^{n} |y_i - p(x_i)|^2, \qquad \text{Eq.1}$$

where $p(x_i)$ is the interpolant of a kth degree polynomial that can be expressed in the system of linear equations with polynomial coefficients $a_{0,\ldots,}a_k$:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^k \\ 1 & x_2 & x_2^2 & \ldots & x_2^k \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix}, \qquad \text{Eq.2}$$

i.e. $y = \mathrm{X}a$. This can be solved by pre-multiplying by the transpose of X^T, i.e. $\mathrm{X}^\mathrm{T}y = \mathrm{X}^\mathrm{T}\mathrm{X}a$. We can then yield this system for a_k for a second-degree polynomial to construct the interpolant $p(x)$ by inverting directly the matrix equation:

$$a = (\mathrm{X}^\mathrm{T}\mathrm{X})^{-1}\mathrm{X}^\mathrm{T}y, n > k. \qquad \text{Eq.3}$$

Finally, we acquire the parabola of the deck slab $f(x) = Ax^2 + Bx + C$ with A, B, $C \in \mathbb{R}$, $a \neq 0$. Next, we compute the tangent at each interpolant of the parabola (Figure 4). The derivative of the parabola gives the slope of the line tangent: $\mathrm{tangent}_j = f(x)' = 2Ax + B$. The normal is given by $\mathrm{normal}_j = \frac{-1}{\mathrm{tangent}_j}$. The deck slab is then segmented along the direction of the normal of each interpolated position into J slices.

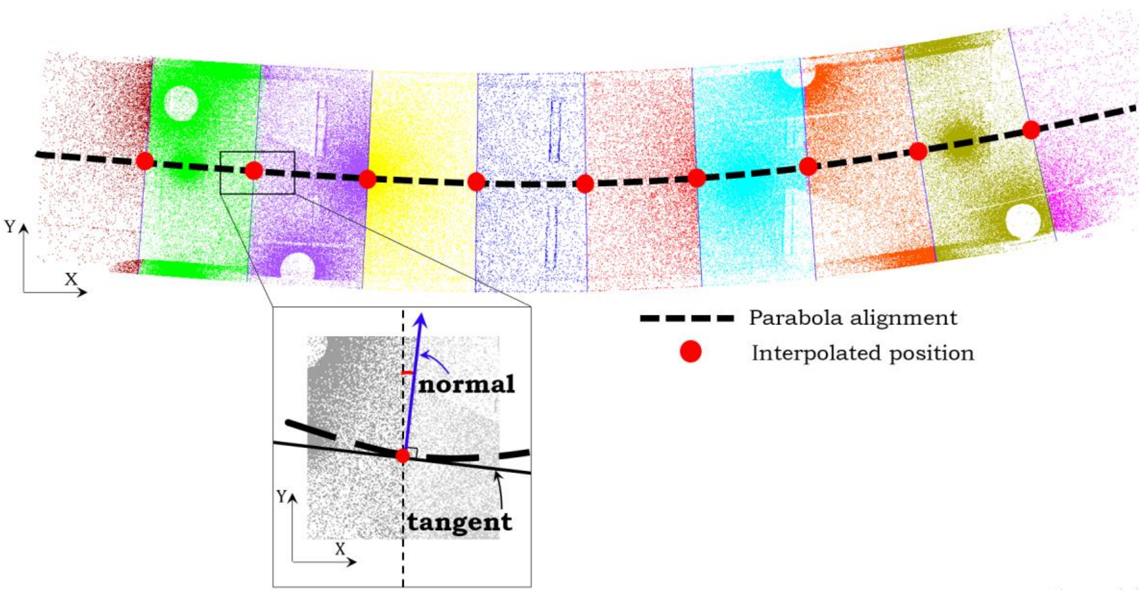


Figure 4 Slicing deck slab along the normal of the interpolated positions

We then assume that each slice runs straight along its tangent direction and that its cross-section is constant. This way, the problem of modelling the whole deck slab is transformed into modelling each straight slab slice. For each slice, the method first rotates the slice around the Z-axis using:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(-\varphi_j) & \sin(-\varphi_j) & 0 & 0 \\ -\sin(-\varphi_j) & \cos(-\varphi_j) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad \text{Eq.4}$$

where the rotated angle φ_j is derived from the angle between the normal direction of the alignment of the slice j and the global Y-axis. Specifically, the normal direction of each slice is computed using the mid-x value of each slice j. We use a 2D *ConcaveHull* α-shape (Moreira & Santos, 2006) to describe the outline of the slice cross-section using the updated points (x', y', z'). Each concave hull of the local XY-plane projection of the slice j is stored as a 2D Cartesian point *IfcCartesianPoint* (Figure 5 (a)). These *IfcCartesianPoint* elements map the cross-section with a list of *IfcPolyline* objects (Figure 5 (b)). A 2D profile *IfcArbitraryClosedProfileDef* is therefore used to describe the slice cross-section. The slab slice geometry is then represented using an extruded geometry model through *IfcExtrudedAreaSolid* and *IfcShapeRepresentation*, expressing it as a Swept Solid. The extruded area solid defines the extrusion of a 2D area (given by a profile definition) by two attributes. One is the ExtrudedDirection, defining the direction in which the profile is to be swept; the other is the Depth, defining the distance over which the profile is to be swept. The ExtrudedDirection is derived from the tangent direction at the mid-x value position of each slice. The depth is derived from the maximum and minimum x'-coordinates of each oriented slab slice.

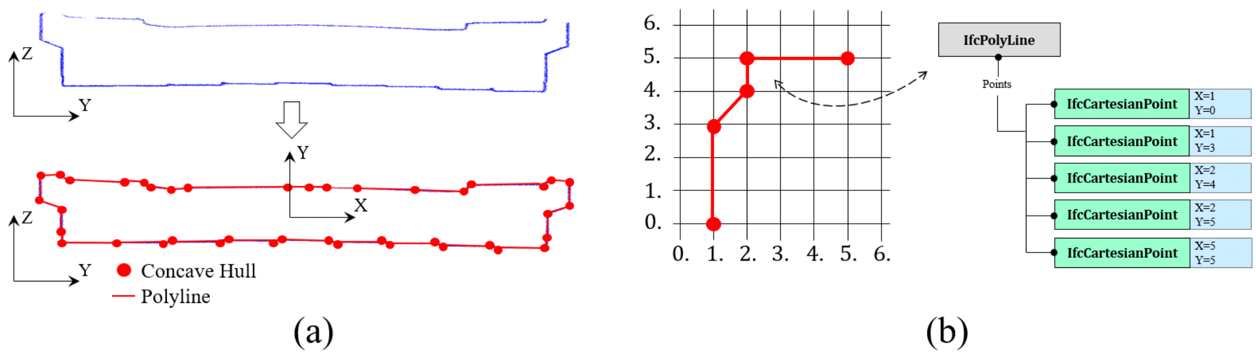


Figure 5 (a) concave hulls of the local XY-plane of slice j; (b) an example of IfcPolyline object

Figure 6 shows an example of a snippet of the IFC data file of a slab slice, defined by 92 concave hulls that are connected by 93 polylines. IFC has a flexible extension mechanism that allows for custom defined attributes through *IfcPropertyset* without modifying the underlying schema. *IfcPropertyset* is a set of IFC properties which store the actual data as triplets including name, data type, and value. We introduce a property set *Pset_SlabSliceProperties*, in which the method adds the attributes (e.g. cross-section area, length, and orientation) of each slab slice and composes them into an *IfcPropertyset*.

```
/***********************************/
/*                Slab 1           */
/***********************************/
#100001=   IFCCARTESIANPOINT((-1655.0,269561.6));
…
#100091=   IFCCARTESIANPOINT((-1243.2,269571.9));
#100092=   IFCCARTESIANPOINT((-1431.6,269563.1));
#101=      IFCPOLYLINE((#100001,#100002,#100003,#100004,
           … ,#100090,#100091,#100092,#100001));
#102=      IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,'deckSlab',#101);
#103=      IFCCARTESIANPOINT((37646.700000000004,0.,0.));
#104=      IFCSLAB('7IfdS9ZAQku4vN074Zp8',$,'deckSlab',$,'deckSlab',$,#107,'deckSlab',
           $);
#105=      IFCEXTRUDEDAREASOLID(#102,#108,#114,3904.3);
#106=      IFCSHAPEREPRESENTATION(#1,'Body','SweptSolid',(#105));
#107=      IFCPRODUCTDEFINITIONSHAPE($,$,(#106));
#108=      IFCAXIS2PLACEMENT3D(#103,#2,#3);
#109=      IFCPROPERTYSINGLEVALUE('Property A:',$,IFCIDENTIFIER('N/A'),$);
#110=      IFCPROPERTYSINGLEVALUE('Property B:',$,IFCIDENTIFIER('N/A'),$);
#111=      IFCPROPERTYSINGLEVALUE('Property C:',$,IFCIDENTIFIER('N/A'),$);
#112=      IFCPROPERTYSET('Q4aFfLsjxKvYYYQNpxfR',$,'Pset_SlabSliceProperties',$,(#10
           9, #110, #111));
#113=      IFCRELDEFINESBYPROPERTIES('IzbZOghGrptNbGu3FayF',$,$,$,(#104),#112);
#114=      IFCDIRECTION((-0.02355817626597581,0.,1.));
```

Figure 6 Snippet of the IFC data file of a slab slice

3.4.2 Pier cap – *IfcBeam*

Similar to how the slab slice is extruded, when modelling a pier cap point cluster, we project its points onto the XY-plane. We then use a 2D *ConcaveHull* α-shape to describe the projected contour such that each concave hull of the local XY-plane projection of the pier cap is stored in a 2D Cartesian point *IfcCartesianPoint* followed by mapping the contour with a list of *IfcPolyLine* objects. Like the slab slice, a pier cap is also represented as a Swept Solid through *IfcArbitraryClosedProfileDef* and *IfcExtrudedAreaSolid*. Specifically, the extruded direction is assumed to be vertical for pier caps and the depth is defined as the height of the pier cap, which is calculated using the maximum and minimum of its z-coordinates. Likewise, we introduce the property set *Pset_PierCapProperties*, for which the method can flexibly add attributes.

3.4.3 Pier – *IfcColumn*

Piers support the weight of a bridge against gravity and serve as retaining walls to resist lateral movement. Defining a generic parametric pier object is difficult because piers can take many configurations. In general, its cross-section, whose scale may vary over its height, defines the shape of a pier. Figure 7 illustrates a collection of the most typical cross-section shapes of piers for modern highway bridges (Wai-Fah & Lian, 2014). However, in reality, piers can also take many other irregular shapes.

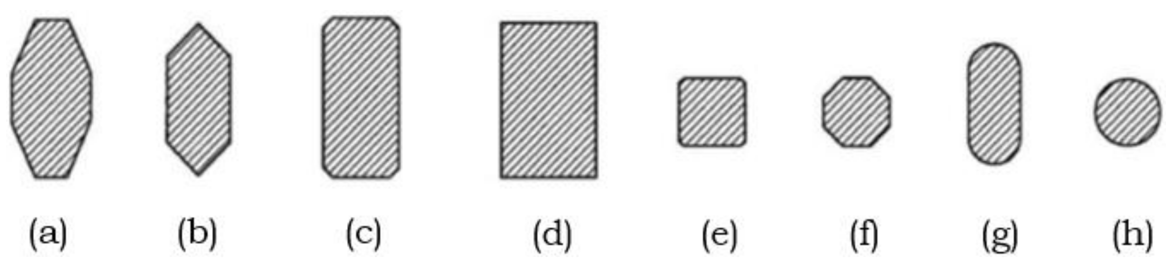


Figure 7 Typical cross-section shapes of piers (Wai-Fah & Lian, 2014)

To simplify the problem, we group the cross-sections of typical pier shapes into 3 classes of primitives: circular (cylindrical piers), quadrilateral (cuboid or trapezoidal prism piers), and the others:

- *Shape group* 1 – Circular (Figure 7 (h));
- *Shape group* 2 – Quadrilateral (Figure 7 (d));
- *Shape group* 3 – Other shapes: the rest, Figure 7 (a), (b), (c), (e), (f) and (g).

Unlike simplified scenarios and synthetic data, the underlying real objects in point clouds are similar to hand-drawn geometric shapes that usually contain imperfections. A shape detection method is needed to tackle different situations. It should be invariant and robust to scaling, distortion, occlusion, and the jagged edges produced by imperfect boundaries. We use a fuzzy-logic-based shape descriptor to achieve this goal. It can handle ambiguity in imperfect point cloud projections in a natural manner, thereby recognizing cross-section shapes independently of noise, edge effect, size, unevenly distributed points, and occlusions. We elaborate on this method in the following.

Piers are not necessarily perfectly vertical, although we assume that the piers investigated in this research are quasi-vertical. First, we project a pier point cluster onto the global XY-plane followed by calculating the perimeter of the projected points (denoted P_{ch}) and the bounded area (denoted A_{ch}) using their concave hulls. We then compute the area of the enclosing rectangle of the concave hulls, i.e., the 2D oriented-bounding-box (denoted A_{er}) and the area of their inner largest-quadrilateral (denoted A_{lq}). Figure 8 (a) and (b) illustrate examples of a cylindrical pier and a trapezoidal prism pier, respectively. As shown, the cross-section of a cylinder is close to a circle while the cross-section of a trapezoidal prism pier is close to a rectangle. If the cross-section is detected as a circle, then the perimeter of the concave hulls P_{ch} (Figure 8 (a.3)), the enclosing rectangle A_{er} (Figure 8 (a.4)), and the inner largest quadrilateral A_{lq} (Figure 8 (a.5)) are distinctly different from each other, whereas if the cross-section is a quadrilateral, these three geometric features are similar to each other (Figure 8 (b)).

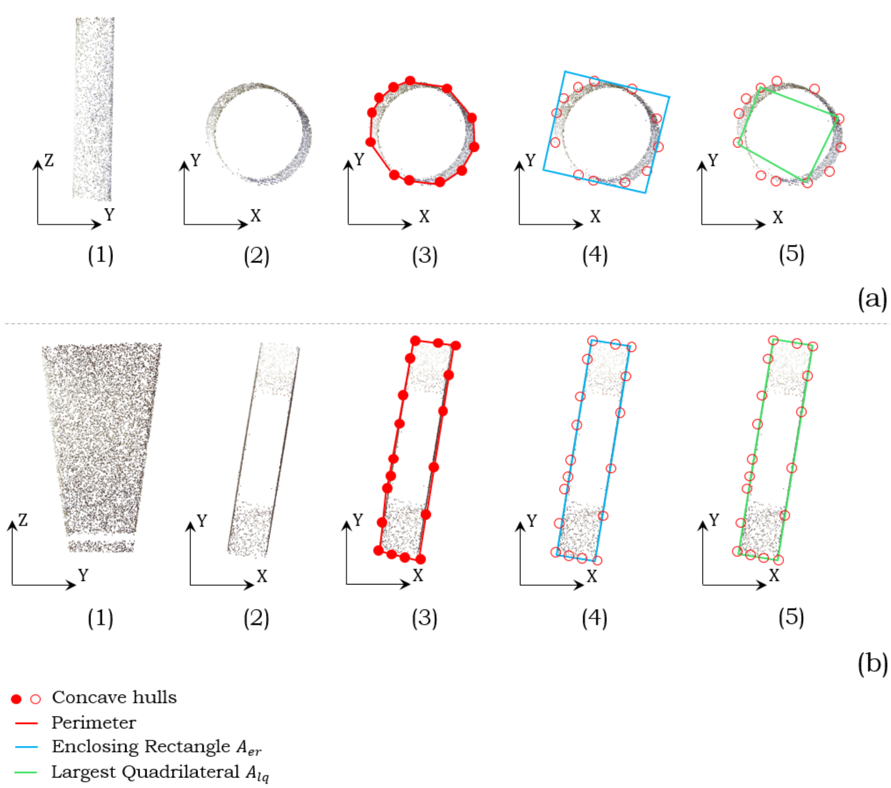


Figure 8 (1) YZ-plane projection; (2) XY-plane projection; (3) concave hulls of XY-plane projected points; (4) enclosing rectangle of concave hulls; (5) largest quadrilateral of concave hulls

Define the thinness ratio as ${P_{ch}}^2/A_{ch}$:

$$\text{if } {P_{ch}}^2/A_{ch} \cong 4\pi, \qquad \text{Eq.5}$$

then, the cross-section ← circle,

The thinness of a circle is minimal since it is the planar figure with the smallest perimeter enclosing a given area, yielding a value around 4π. Next:

$$\text{else if } A_{ch}/A_{er} \cong A_{lq}/A_{er} \cong 1, \qquad \text{Eq.6}$$

then, the cross-section ← rectangle.

Specifically, we use Bretschneider's formula (Eq.7) to calculate the area of a quadrilateral inside a set of 2D points (Figure 9):

$$A_q = \sqrt{(sp-a)(sp-b)(sp-c)(sp-d) - abcd \cdot \cos^2(\frac{\alpha+\gamma}{2})}, \qquad \text{Eq.7}$$

where sp is the semi-perimeter. The inner largest-quadrilateral A_{lq} is the maximum value of A_q found.

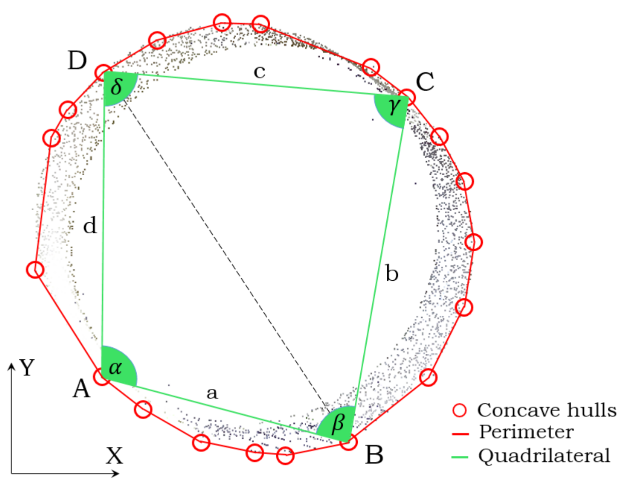


Figure 9 A quadrilateral inside concave hulls of the projected points of a cylindrical pier

Otherwise, then the features satisfy neither Eq.5 nor Eq.6, the cross-section takes another shape. For a shape that is identified as a *group 1* shape (circular), we describe the pier using a small number of parameters. Otherwise, we conduct a slicing procedure followed by using 2D α-shape to describe the cross-section. The following texts elaborate the steps of twinning these classified shapes into 3D *IfcObjects*.

Cylindrical pier

If a cross-section shape is identified as a circle, then it is a cylindrical pier. We need a minimum of three parameters to define a cylindrical pier in 3D space: radius (or diameter), location, and direction. To keep consistent, we use an efficient slicing method to twin a cylinder. It is first conducted along the Z-axis. Then, *IfcAxis2Placement3D* is used to define a location point and the orientation. The coordinates of the location point are stored in a 3D Cartesian point *IfcCartesianPoint* as an attribute Position. The pier direction information in the 3D coordinates system is stored in *IfcDirection*, which is defined by the vector computed by the bottom and upper slice centre of the cylinder, i.e. point A (x_A, y_A, z_A) and point B (x_B, y_B, z_B): $\frac{\overrightarrow{AB}}{|\overrightarrow{AB}|} = \left(\frac{x_B - x_A}{|\overrightarrow{AB}|}, \frac{y_B - y_A}{|\overrightarrow{AB}|}, \frac{z_B - z_A}{|\overrightarrow{AB}|}\right)$. The slicing procedure is then conducted again along the pier direction followed by computing the radius for each slice. The radius of the entire cylinder is calculated by averaging the radii obtained from the multiple slices. The average radius value is stored in *IfcCircleProfileDef* as an attribute Radius. Next, like the deck slab and pier cap, the geometry of the cylindrical pier is represented using the extruded model through *IfcExtrudedAreaSolid* and *IfcShapeRepresentation*, expressing it as a Swept Solid along its extruded direction *IfcDirection*. We introduce the property set *Pset_CylinderProperties*, in which four attributes are defined: Position, Direction, Diameter, and Length. The method then composes them into an *IfcPropertyset*.

Quadrilateral and other piers

If a pier cross-section shape is identified as a quadrilateral or other shape, we follow a similar strategy but use a stacked representation to approximate the overall pier shape through multiple slice models. For each slice, we apply the same method used for twinning the pier cap. That is to say, each slice of the pier is considered a pier cap, so that again we use a 2D α-shape to describe the cross-section of the pier slice using *IfcArbitraryClosedProfileDef* and *IfcExtrudedAreaSolid*.

3.4.4 Girder – *IfcBeam*

The majority of beam-slab bridges to be built in the near future in the UK select precast concrete components for the primary structural elements (Kim et al., 2016). Therefore, we assume that the girders studied in this research are precast, standardized bridge beams. A template matching method is suggested to find the best-match girder type in existing precast bridge beam catalogues. We use the girder sections provided by the standard products of the American Association of State Highway and Transportation Officials (AASHTO) and the Bridge Beam Manual provided by BANAGHER Precast Concrete (BANAGHER, 2018) (BANAGHER, 2018), which is the largest precast concrete Bridge Beam manufacturer in Ireland and the UK. According to Lu et al. (2018), the specific girder type in each span can be inferred using three criteria: 1) Span length sl; 2) Girder bottom flange b_f; and 3) Web depth d. The span length sl can narrow down a possible range of girder types. This is because, often, the creation of a typical girder section begins with the calculation of the structure depth for a given span length (AASHTO, 2017). Then, the girder bottom flange b_f and the web depth d can be used to select a specific girder type from the possible girder types. Lu et al. (2018) have given the slope l of each segmented slab so that we can derive angle θ. Then, sl is approximately calculated using the maximum and minimum x- and y-coordinates of each slab: i.e. $sl \approx \Delta x - \Delta y * tan\theta$ (Figure 10 (a)). Given that the girders are already segmented in each span, we can calculate the bottom flange of each girder such that b_f is the average value. The web depth d is also given by Lu et al. (2018) using the projection histograms of the girders in each span along Z-axis. Figure 10 illustrates an example of girder type determination using the three criteria, where sl_3 ≈28 m, $\overline{b_f}$ ≈760 mm, and d ≈1600 mm (Figure 10 (a)). The closest precast girder type found in the BANAGHER Manual is type SY2 from SY Beams (Figure 10 (b)).

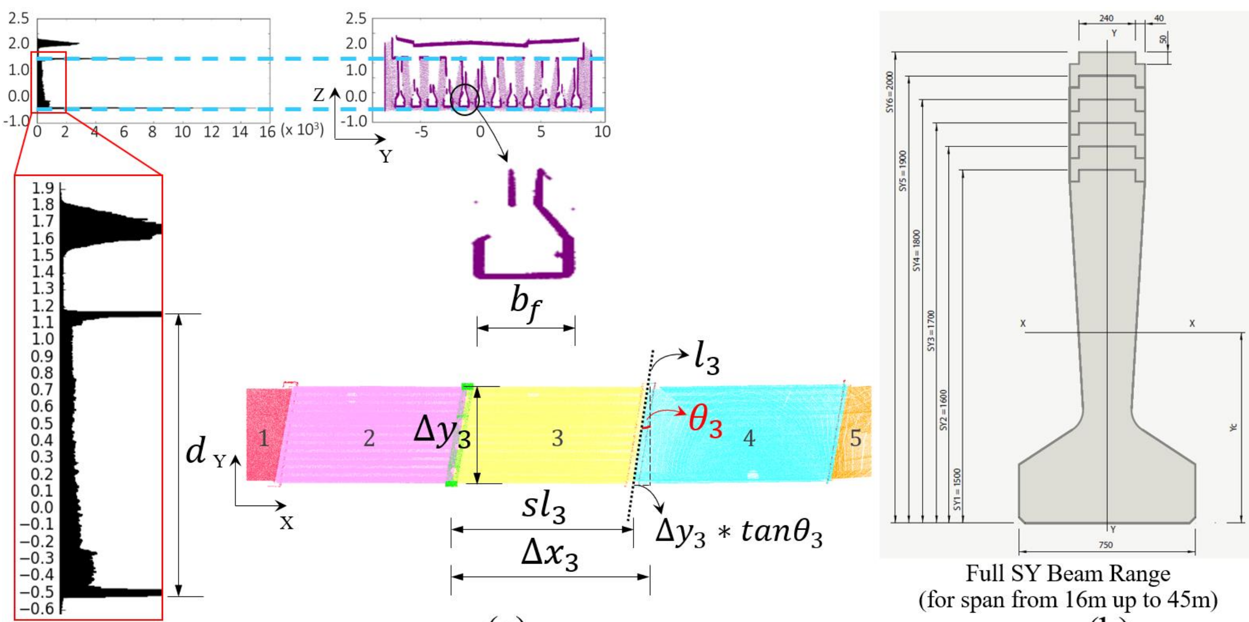


Figure 10 (a) matching criteria; (b) best matching type from catalogue

Next, we encode the identified profile using IFC standards. The profile feature points are used to describe the geometry of the girder. For instance, a girder point cluster is matched with a standard pre-stressed wide flange concrete girder, e.g. WF50G (Figure 11). Given the coordinates of the starting middle bottom point (green point pt_start in Figure 11 (b)), and the dimensions of WF50G, each feature point (red point in Figure 11 (b)) can be defined accordingly with the exact coordinate information. Then, we store the coordinates of each feature point in a 2D Cartesian point *IfcCartesianPoint* in its local XY-coordinates, followed by mapping the contour with a list of *IfcPolyline* objects. A 2D profile *IfcArbitraryClosedProfileDef* is used to describe the girder profile. The girder is then represented as a Swept Solid. Assuming that the girders in each span are straight, the extruded direction is defined by the starting and end middle bottom points of a girder point cluster. Again, we introduce the property set *Pset_GirderProperties*, in which the attributes such as Girder Type, Length, and Slope are added. The length and slope information of a girder can be computed using its Oriented Bounding Box representation.

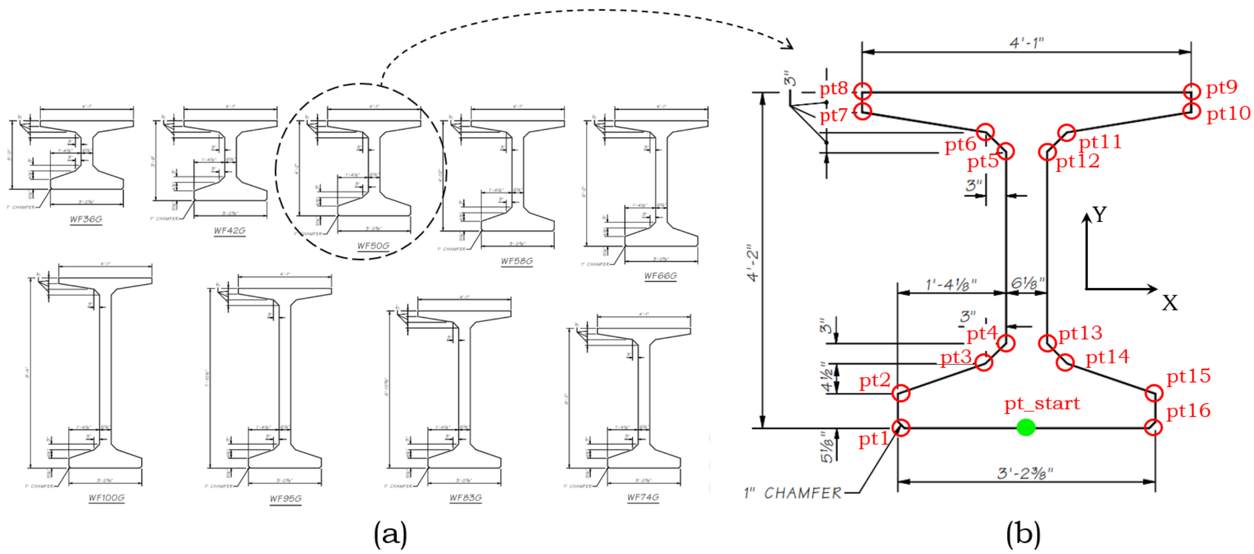


Figure 11 (a) Example of standard pre-stressed wide flange concrete girders (WSDoT, 2009); (b) WF42G and the feature points (in total 16 points)

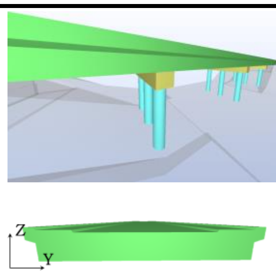
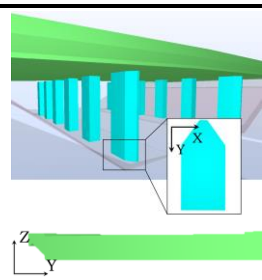
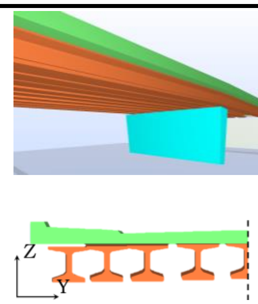
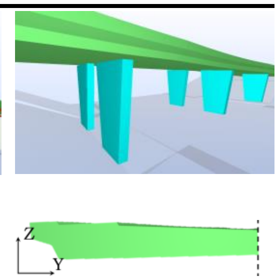
4 Experiments and Results

4.1 Ground Truth Data

In order to test the hypothesis of this research, we used the ten bridge point clouds collected by Lu et al. (2018) to conduct the experiments. The raw data is available at http://doi.org/10.5281/zenodo.1233844. First, we prepared point clusters of the four component types, serving as the input of the proposed method for all the ten bridges, such that each bridge dataset consists of labelled point clusters. Next, a set of ground truth (GT) gDTs was manually generated and exported into IFC files using Autodesk Revit (Table 2).

GT: The four types of bridge components in this set of models were represented within their precise dimensions. These models were considered in line with and were compared against the automatically generated LOD 250 – 300 gDTs using the proposed method. The average time spent on manually creating one such GT gDT was 27.6 (±16.4) hours (around 1656 minutes).

Table 2 Manual modelling of GT gDTs in IFC format

gDT	***Bridge 1***	***Bridge 4***	***Bridge 7***	***Bridge 9***
GT				
Time (h)	50	26	27	20

4.2 Implementation & Results

The proposed IFC object fitting method was implemented on Gygax (https://github.com/ph463/Gygax/) as a software prototype module, on a desktop computer (CPU: Intel Core i7-4790K 4.00 GHz, Memory: 32 GB, SSD: 500 GB). We designed the module in a flexible way so that one can acquire an IFC file containing a bridge gDT according to a given LOD. This is achieved by the FineLevel class, representing a list of LODs. That is to say, we produced an IFC file of a bridge with a specific LOD by generating a subclass of IFCBaseGenerator. For example, a LoD250300Generator class inherited from IFCBaseGenerator was generated to produce a LOD 250 – 300 bridge gDT (Figure 12). This way, we can extend the module to accommodate future needs for generating higher LOD gDTs. The Unified Modelling Language (UML) Diagram and the Graphic User Interface (GUI) of Gygax are shown in Figure 12.

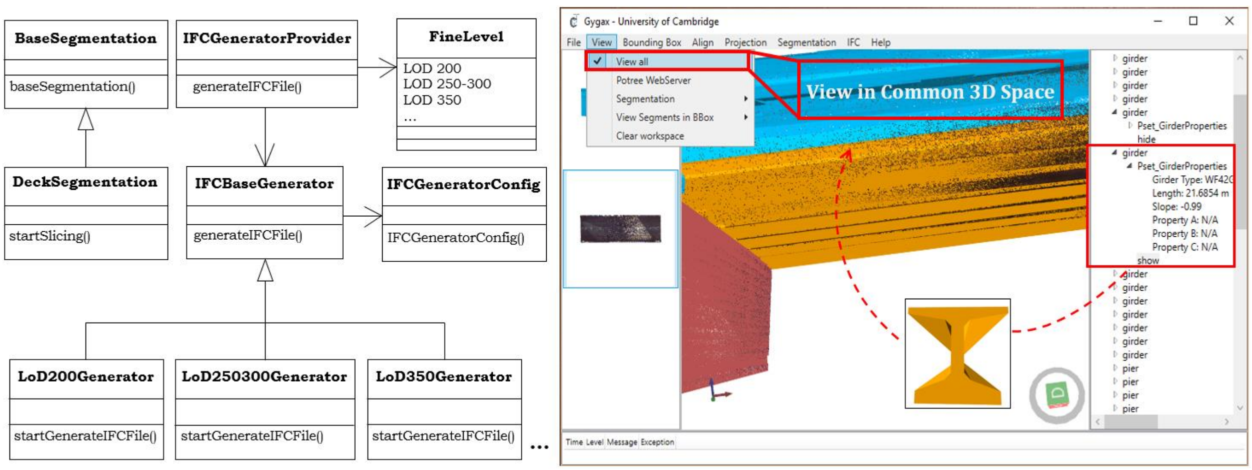
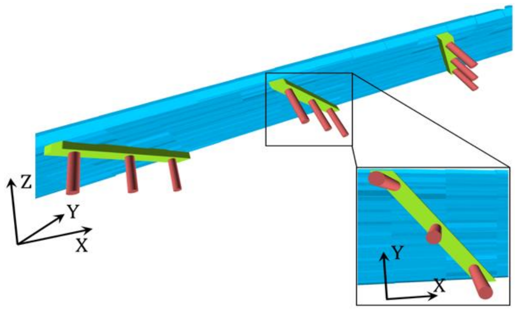
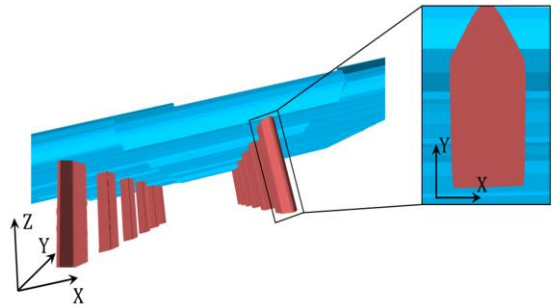
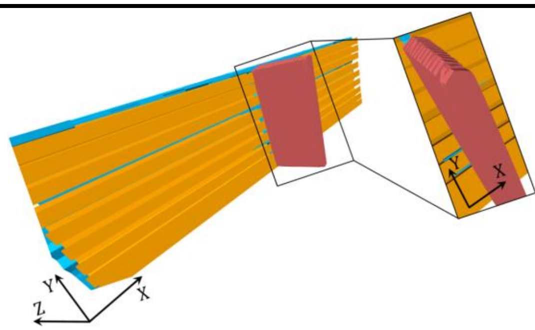
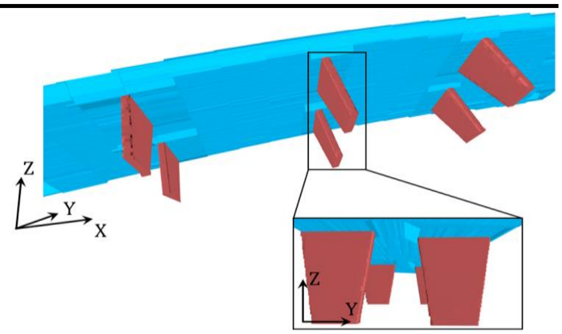


Figure 12 UML diagram of the IFC object-fitting module (L); LOD 250-300 gDT implementation (R)

Table 3 illustrates the results of the LOD 250 – 300 gDTs in IFC format generated by the proposed method (we show only four bridge examples due to limited space). The number of deck slab slices and pier slices were both set to be 20. The value α in the 2D *ConcaveHull* α-shape algorithm was set to be 0.98. The twinning time was recorded. For a bridge dataset of four types of point cluster containing less than one million points together, the average twinning time was 37.8 (±28.4) seconds for LOD 250 – 300 gDT generation. The time spent on generating the LOD 250 – 300 gDT for *Bridge 4* (58.1 seconds) and for *Bridge 10* (65.5 seconds) was 53.7% and 73.3% higher than the average, respectively. This is mainly because *Bridge 4* has large sparse regions in the slab point clusters and *Bridge 10* contains roughly 70% more points than other bridges. Both situations took more processing time. In summary, compared to the manual modelling process, *GT* (27.6 hours = 99360 seconds), the time cost of the proposed method is trivial. This means a direct time saving of 100%.

Table 3 LOD 250 – 300 gDTs generated from the proposed method

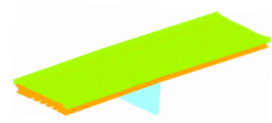
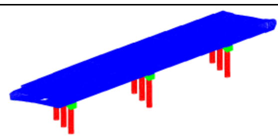
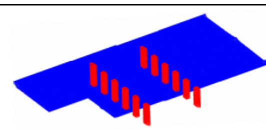
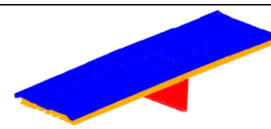
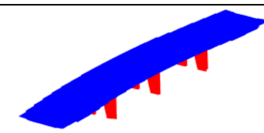
gDT	***Bridge 1***	***Bridge 4***
LOD 250-300		
Time (s)	25.5	58.1
gDT	***Bridge 7***	***Bridge 9***
LOD 250-300		
Time (s)	31.1	37.3

4.3 Evaluation

The nature of the *ConcaveHull* α-shape algorithm used in the proposed fitting method makes it impossible to evaluate the resulting gDTs using vertex-based metrics. This is because the vertices of the manual gDTs and that of the automated ones do not correspond. Normally, the number of hulls found in the automated gDTs by the proposed method is much greater than that of the vertices of the manual models. This is because when we use a modelling software interface to assist with the act of creating a 3D object embedded in point clouds, almost every object description is approximate in the sense that it describes the geometry of the 3D object only to the extent that inputting this description into the modelling software module produces a 3D model of acceptable quality. Thus, the surfaces of the manually generated gDTs are smooth planes without local undulations. To this end, we chose distance-based cloud-to-cloud (C2C) metrics to evaluate the automated LOD 250 – 300 gDTs by comparing the twinning quality between the manual gDTs and the automated gDTs.

To do so, we converted both the manual gDTs and the automated ones in IFC format into point clouds. This was achieved by converting the geometry in .ifc file format into .obj file format using IfcOpenShell (2018). The .obj format is a data format which represents only the 3D geometry information, such as the vertex position, vertex normal, and the faces that define each polygon as a list of vertices. Next, we randomly sampled points using the generated polygons for each manual gDT as well as each automated LOD 250 – 300 gDT. The number of the sampled points from the polygons was in line with the original size of the point cloud of each bridge. We acquired two sets of point cloud data (PCD): *GT* PCDs and *Auto* PCDs (Table 4). Thus, the problem of comparison of the twinning quality (between the manual gDTs and the automated gDTs) is transformed into measuring the difference between the two sets of point clouds, compared against the original real (reference) point cloud of each bridge, respectively. It is worth noting that the laser scanner (Faro Focus 3D X330) we used for the data collection can sample an object's surface highly accurately in the form of point clouds. The theoretic ranging error can be up to ±2 mm. This is a systematic measurement error of around 10 m. However, several factors may affect the measuring accuracy, such as low/high temperature, dust, rain, bright sunshine, and highly reflective surfaces. These factors were not considered in this research. Herein, we assume that the original real point cloud has a very high degree of spatial accuracy. We elaborate the comparison in the following.

Table 4 Sampled point clouds of GT gDTs and of Automated LOD 250 – 300 bridge gDTs

	Bridge 1	*Bridge 4*	*Bridge 7*	*Bridge 9*
GT **PCD**				
Auto **PCD**				

One central problem in computer graphics is measuring the extent to which one shape differs from another. The Hausdorff distance is a commonly used shape comparison method that can measure the difference between two different representations of the same 3D object (Aspert et al., 2002; Cignoni et al., 1998). Given two point sets $A = \{a_1, \dots, a_p\}$ and $B = \{b_1, \dots, b_q\}$, the Hausdorff distance is defined as:

$$H(A, B) = \max(h(A, B), h(B, A)), \qquad \text{Eq.8}$$

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|_2, \qquad \text{Eq.9}$$

where $\|.\|_2$ denotes the usual Euclidean norm on the point sets A and B. The function $h(A, B)$ is called the directed Hausdorff distance from A to B. It determines the point $a \in A$ that is farthest from any point of B and measures the distance from a to its nearest neighbour in B (using $\|.\|$). In other words, $h(A, B)$ ranks each point of A based on its distance to the nearest point of B and uses the largest ranked point as the distance. The Hausdorff distance $H(A, B)$ is the maximum of $h(A, B)$ and $h(B, A)$. However, the issue that needs to be noted is that the nearest neighbour is rarely, in reality, the actual nearest point on the surface represented by the point cloud. This is especially true if the reference point cloud is non-uniformly distributed or contains occlusions. That is why we first kept within an order of magnitude of at least 4 million points for the sampled points for each bridge to conduct the distance calculations. However, defects in real-world point clouds cannot be totally avoided. In this scenario, a local distance strategy was leveraged to compute a local model using neighbouring points to get a better estimation of the "real" distance (Figure 13). We used a quadratic model Q, which can be expressed as $Q(x, y, z) = ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + iz + j = 0$ to fit the neighbouring points in the reference point cloud on a smooth surface within a radius of 0.3 m. This means that we not only compute the distance of a single point, we also take into account a local tendency. Given a point q_i of the compared point cloud that is not on the quadratic model Q, the Euclidean distance from this point q_i to Q can be expressed as:

$$d(q_i, Q) = \min\{\|q_i - p\| : Q(p) = 0\}. \quad \text{Eq.10}$$

Hence, the estimated average local distance from a compared point cloud to a reference point cloud is:

$$\overline{\text{dist}} = \frac{1}{n}\sum_{i=1}^{n} \min\{d(q_i, Q)\}. \quad \text{Eq.11}$$

The overall estimated distance between a compared point cloud and a reference point cloud is then the bigger one of the mutual $\overline{\text{dist}}$, that is:

$$\text{C2C} = \max(\overline{\text{dist}}_{A/B}, \overline{\text{dist}}_{B/A}). \quad \text{Eq.12}$$

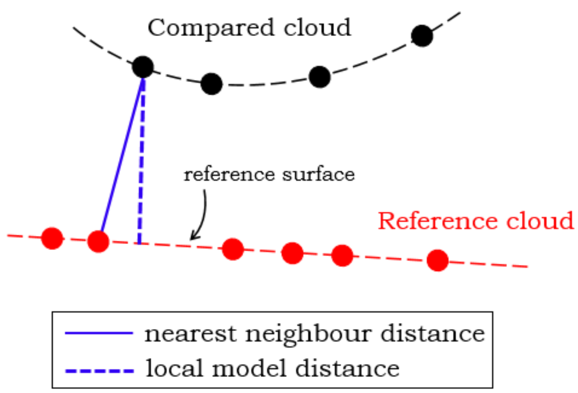


Figure 13 Nearest neighbour distance and local surface model distance

Table 5 summarizes the C2C distances of:

- *GT* PCDs against the real world PCDs (i.e. *GT*/*Real* & *Real*/*GT*); and
- *Auto* PCDs against the real world PCDs (i.e. *Auto*/*Real* & *Real*/*Auto*)

in colour scalar field for four bridge datasets. An automated gDT is deemed to be better modelled if its C2C (denoted C2C_{Auto}) is smaller compared to that of the manual model (denoted C2C_{GT}), and vice versa. In total, six out of ten bridge point cloud datasets were modelled better using the proposed method than by manual modelling (the better C2C result was highlighted in green). The C2C of the remaining four *Auto* PCDs were found to be close to those of their corresponding *GT* ones. The overall C2C_{Auto} of ten bridge automated gDTs was 7.05 cm while the C2C_{GT} was 7.69 cm. Note that these results contain challenging scenarios, details of which are discussed in the next section. Table 6 illustrates the histograms of the C2C distribution (*Auto*) of the four bridges using the colour map, where the horizontal axis presents the C2C distance in metres while the vertical axis presents the point counts. We also calculated the number of matched points (in percentage) of each bridge derived from their automated gDTs, compared to the corresponding real point cloud (Table 7). We define “matched” at different levels, i.e. C2C<10 cm, C2C<7.5 cm, C2C<5 cm, and C2C<2.5 cm. On average, 78.6% of points representing the automated gDTs had a C2C distance less than 10 cm, 72.5% inferior to 7.5 cm, 61.6% inferior to 5 cm, and 41.3% inferior to 2.5 cm. Full results of the C2C distances of the ten bridges and the histograms of the C2C distribution of the other six bridges are given in the Appendix.

Table 5 Comparison of C2C distance between GT PCDs and Auto PCDs against Real world PCDs

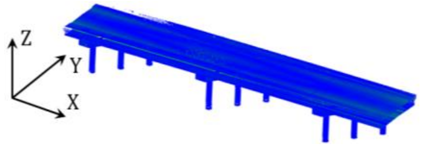
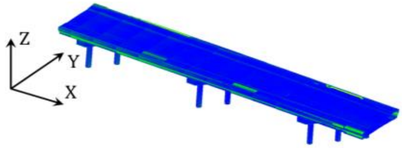
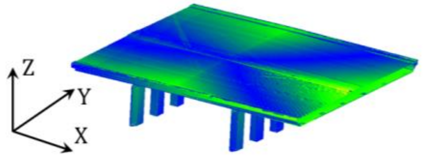
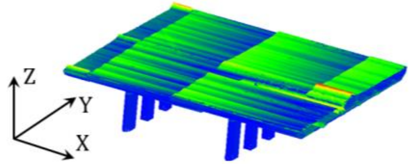
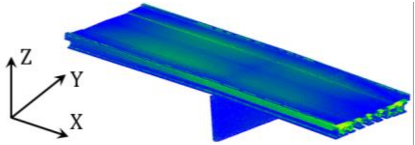
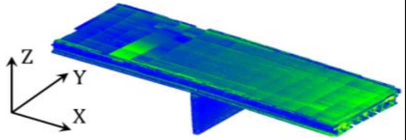
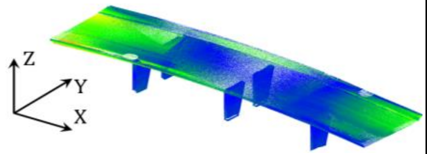
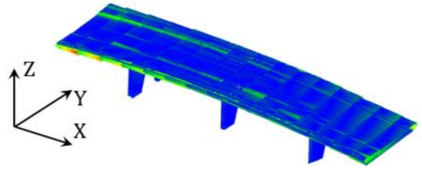
Bridge 1		***Bridge 4***	
C2C - *GT/Real*	C2C - *Auto/Real*	C2C - *GT/Real*	C2C - *Auto/Real*
4.0 cm	4.3 cm	7.3 cm	9.4 cm
			
Bridge 7		***Bridge 9***	
C2C - *GT/Real*	C2C - *Auto/Real*	C2C - *GT/Real*	C2C - *Auto/Real*
15.7 cm	12.5 cm	9.8 cm	5.6 cm
			

Table 6 C2C distance of Auto PCDs in histogram colour map

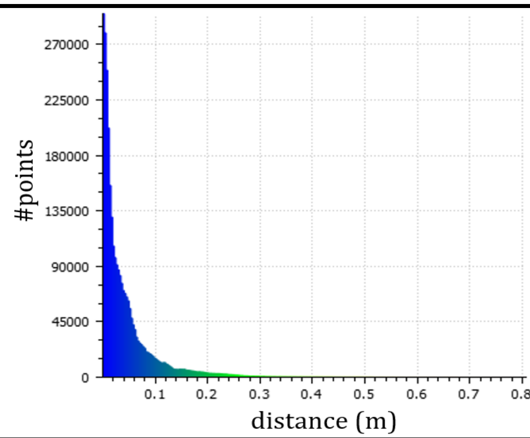
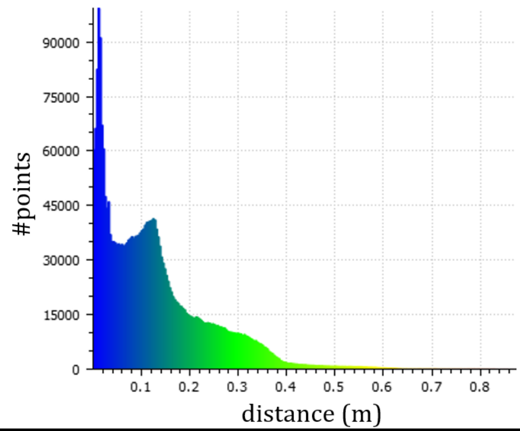
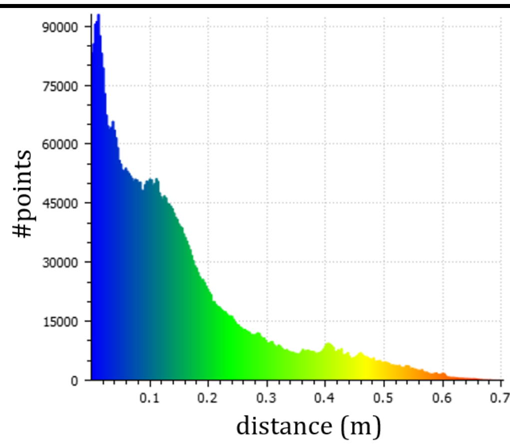
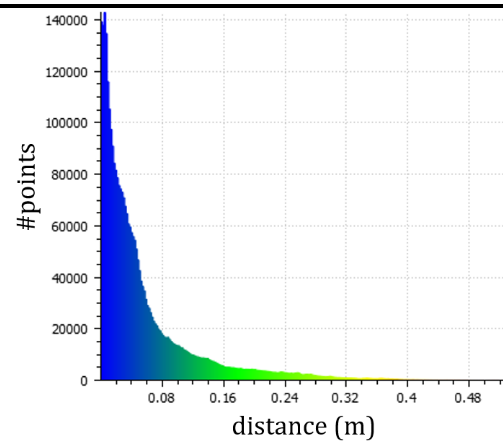
	Bridge 1	***Bridge 4***
Histogram colour map		
	Bridge 7	***Bridge 9***
Histogram colour map		

Table 7 C2C distance in percentage of points between Auto PCDs and Real world PCDs

	< 10 cm	**< 7.5 cm**	**< 5 cm**	**< 2.5 cm**
Bridge 1	89.1%	83.6%	73.2%	53.3%
Bridge 2	73.8%	62.8%	47.2%	29.1%
Bridge 3	94.6%	90.3%	69.5%	34.9%
Bridge 4	59.3%	53.2%	46.7%	37.7%
Bridge 5	95.8%	89.7%	75.4%	43.0%
Bridge 6	87.2%	82.3%	75.0%	50.7%
Bridge 7	56.2%	49.7%	40.5%	28.7%
Bridge 8	93.8%	89.5%	77.1%	55.2%
Bridge 9	83.4%	77.3%	66.5%	43.8%
Bridge 10	52.7%	47.0%	44.4%	36.6%
Avg.	**78.6%**	**72.5%**	**61.6%**	**41.3%**

5 Conclusions

To answer the research questions, this paper proposes a novel object fitting method to generate gDTs of existing RC bridges in IFC format, using four types of point clusters. The method produces a bridge gDT with LOD 250 – 300, which uses a stacked slice representation. The resulting gDTs are evaluated in terms of spatial accuracy using distance-based metrics. We discuss in the following texts how well the research questions have been addressed through interpreting the experiment outcomes in detail. The experimental results of the LOD 250 – 300 gDTs generated using the proposed method showed that six out of ten bridges (*Bridges 3*, *5*, *6*, *7*, *8*, and *9*) were better modelled ($\mathrm{C2C}_{Auto} < \mathrm{C2C}_{GT}$). The Represented Accuracy (the standard deviation range that is to be achieved once the point cloud is processed into some other form such as a model) of most bridges was roughly in line with LOA20 (Level of Accuracy 20: 15 mm – 5 cm) (USIBD, 2016), independent of other errors introduced when the measured data (point cloud) was generated and processed into a model. Compared to their *GT* PCDs, the *Auto* PCDs of *Bridge 3*, *5*, *6*, *8*, and *9* had only a small portion of mismatched points, attributed to local small indentions on the deck slab surfaces. The $\overline{\mathrm{C2C}}_{Auto}$ of *Bridges 3*, *5*, *6*, *7*, *8*, and *9* was 4.7 (±0.5) cm while their $\overline{\mathrm{C2C}}_{GT}$ was 7.6 (±2.4) cm. The small indentions the concentrated in areas where sparse data was present. Specifically, the whole slab surface points in the *GT* PCD of *Bridge* 5 were found to be mismatched (several centimetres higher) to the *Real* PCD. This suggested that the quality of the manually generated gDTs was not consistent, depending largely on the modeller's rigorousness. The topologies of *Bridge 8* and *Bridge 9* were quite similar. Both deck slabs contain obviously curved alignments. The proposed method correctly depicted their geometries and outperformed the manual operation: for *Bridge 8*, the $\mathrm{C2C}_{Auto}$ was 3.7 cm while the $\mathrm{C2C}_{GT}$ was 7.2 cm; for *Bridge 9*, the $\mathrm{C2C}_{Auto}$ was 7.2 cm while the $\mathrm{C2C}_{GT}$ was 9.8 cm. Most of the mismatched points in the *GT* PCDs of these two bridges were found on the upper surface of the slab and the boundaries of the extremities, where local undulations were present, and the alignment curves become strong.

By contrast, *Bridge 7* was a challenging scenario, and both its $\mathrm{C2C}_{GT}$ and $\mathrm{C2C}_{Auto}$ were not insignificant. It is not surprising that this was mainly due to the largely missing girder points in the real point cloud, whereas the missing points did not actually affect the manual operation or the proposed method, because both the modeller and the proposed method used engineering inference to overcome the problem of occlusions and produced the girders with complete dimensions. This explains why both C2C distances of the *GT* PCD and *Auto* PCD to the *Real* PCD were large and the tail of the error histogram was long (Table 6).

For the remaining four bridges, the C2C of the *Auto* PCDs were found close to that of their corresponding *GT* ones, except for *Bridge 10*. For *Bridge 1*, the $\mathrm{C2C}_{GT}$ was 4.0 cm while the $\mathrm{C2C}_{Auto}$

was 4.3 cm. For *Bridge 2*, the $\mathrm{C2C}_{GT}$ was 6.4 cm while the $\mathrm{C2C}_{Auto}$ was 7.3 cm. Only a limited number of mismatched points were concentrated locally at the boundaries or on the undulating surfaces. By contrast, *Bridge 4* was a challenging case. A large portion of its slab points in the input data was very sparse. The proposed method did not extract enough concave hulls to capture the slab geometry in that region so that the automated gDT was incomplete, and no points were sampled. We therefore evaluated *Bridge 4* after removing the partially modelled slice to avoid incorrect calculation of the C2C distance. The big value of $\mathrm{C2C}_{Auto}$ (9.4 cm) was again mainly attributed to the locally generated indentions on the slab surface. This explains why Bridge 4 had a long-tail error histogram (Table 6). By contrast, the manual gDT of *Bridge 4* was better modelled ($\mathrm{C2C}_{GT}$ = 7.3 cm), but there were still many mismatched points in the slab. This was due to the varying deck slopes, which are difficult to effectively describe manually. Lastly, *Bridge 10* was the most challenging case. The spatial accuracy of its *Auto* gDT ($\mathrm{C2C}_{Auto}$ = 13.5 cm) was not as good as its GT gDT ($\mathrm{C2C}_{GT}$ = 5.5 cm). Many mismatched points in the *Auto* PCD were found under the deck slab. This is due to the complex geometry of its superstructure. *Bridge 10* is a diaphragm bridge, containing upstand diaphragms (embedded pier caps), which lie on the same level as the integrated beams. The upstand diaphragms are oriented based on the pairwise piers. The proposed method did not properly capture and describe these complex geometries. Thus, the *Auto* PCD were not well matched to the *Real* PCD, leading to a large $\mathrm{C2C}_{Auto}$. This demonstrated that human assistance is still necessary in some really challenging scenarios that the current automated method cannot handle.

Contributions. **Con 1.** The proposed method can effectively twin four types of concrete bridge elements from point clusters in non-standardized shapes. **Con 2.** Although imperfections exist, the experimental results on the ten bridge point clouds proved that, compared to a human modeller, the overall performance of the proposed method is consistent and less liable to human errors ($\overline{\mathrm{C2C}}_{Auto}$=7.05 cm, $\overline{\mathrm{C2C}}_{GT}$=7.69 cm). If *Bridge 7* and *Bridge 10* are not taken into account, the $\overline{\mathrm{C2C}}_{Auto}$ was 5.6 (±1.7) cm while the $\overline{\mathrm{C2C}}_{GT}$ was 7.0 (±2.1) cm. This means that the proposed method realized an improvement of 20% on spatial accuracy. **Con 3.** The average processing time (37.8 seconds) demonstrated the unprecedented ability of the proposed method to rapidly twin bridge concrete elements, significantly overriding the current manual practice. The hypothesis of this research has been experimentally validated. **Con 4.** The use of this method will reduce the repetitive work of the manual gDT generation and provide a basis that could be integrated into the BMS currently used in practice. The entire digital twinning process will then be streamlined, and the cost and benefit ratio will be improved.

Future work will focus on 1) developing gap-less slab segments that will keep the tangential continuity of the alignment and can be mapped to *IfcAlignment*; 2) taking more bridge configurations and component types into account; 3) investigating the effect of different parameters on the overall performance. For example, we will study how much the number of slices, the alpha value of

ConcaveHull, and the level of surface smoothness affects the performance of the proposed method.

Acknowledgements

This research work is supported by EPSRC, Infravation SeeBridge project under Grant Number No. 31109806.0007, and Cambridge Trimble Fund. We would like to thank for their supports. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of EPSRC, Infravation SeeBridge, or Trimble.

References

AASHTO. AASHTO LRFD Bridge Design Specifications, American Association of State Highway and Transportation Officials, 8th Edition (2017). Available at https://store.transportation.org/Common/DownloadContentFiles?id=1648, accessed 2 May, 2019.

AASHTOWare. (2018). AASHTOWare. Available at https://www.aashtoware.org/wp-content/uploads/2018/03/Bridge-Rating-Product-Brochure-FY-2019-11022018.pdf, accessed 2 May, 2019.

ASCE. (2013). 2013 Report Card for America's Infrastructure, Bridges. Available at http://2013.infrastructurereportcard.org/bridges/, accessed 2 May, 2019.

ASCE. (2017). 2017 Report Card for America's Infrastructure, Bridges. Available at https://www.infrastructurereportcard.org/wp-content/uploads/2017/01/Bridges-Final.pdf, accessed 2 May, 2019.

Aspert, N., Santa-Cruz, D., & Ebrahimi, T. (2002). MESH: Measuring errors between surfaces using the Hausdorff distance. In *Proceedings - 2002 IEEE International Conference on Multimedia and Expo, ICME 2002*. http://doi.org/10.1109/ICME.2002.1035879

BANAGHER. (2018). Bridge Beam Manual, 2nd Edition. Available at https://bancrete.com/bridge-beam-manual/, accessed 2 May, 2019.

BIMForum. (2018). Level of Development Specification. Available at https://bimforum.org/wp-content/uploads/2018/07/BIMForum-LOD-2018_Spec-Part-1_and_Guide_PUB-DRAFT.pdf, accessed 2 May, 2019.

Borrmann, A., Beetz, J., Koch, C., Liebich, T., & Muhic, S. (2018). Industry Foundation Classes: A Standardized Data Model for the Vendor-Neutral Exchange of Digital Building Models. In A. Borrmann, M. König, C. Koch, & J. Beetz (Eds.), *Building Information Modeling* (pp. 81–126). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-92862-3_5

Buckley, B., & Logan, K. (2017). The Business Value of BIM for Infrastructure 2017. *Dodge Data & Analytics*, 1–68. Available at https://www2.deloitte.com/content/dam/Deloitte/us/Documents/finance/us-fas-bim-infrastructure.pdf, accessed 2 May, 2019.

Budroni, A., & Boehm, J. (2010). Automated 3D Reconstruction of Interiors from Point Clouds. *International Journal of Architectural Computing*, *08*(01), 55–73. http://doi.org/10.1260/1478-0771.8.1.55

Carr, J. C., Beatson, R. K., McCallum, B. C., Fright, W. R., McLennan, T. J., & Mitchell, T. J. (2003).

Smooth surface reconstruction from noisy range data. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Austalasia and South East Asia - GRAPHITE '03* (pp. 119–297). http://doi.org/10.1145/604492.604495

Chen, J., Zhang, C., & Tang, P. (2017). Geometry-Based Optimized Point Cloud Compression Methodology for Construction and Infrastructure Management. *In Proceedings of the ASCE International Workshop on Computing in Civil Engineering 2017* (pp. 377–385). http://doi.org/10.1061/9780784480823.045

Cignoni, P., Rocchini, C., & Scopigno, R. (1998). Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum*. http://doi.org/10.1111/1467-8659.00236

ClearEdge3D. (2017). Structure Modelling Tools. Available at https://www.clearedge3d.com/, accessed 2 May, 2019.

Deng, Y., Cheng, J. C. P., & Anumba, C. (2016). Mapping between BIM and 3D GIS in different levels of detail using schema mediation and instance comparison. *Automation in Construction*, *67*(July), 1–21. http://doi.org/10.1016/j.autcon.2016.03.006

Dimitrov, A., Gu, R., & Golparvar-Fard, M. (2016). Non-Uniform B-Spline Surface Fitting from Unordered 3D Point Clouds for As-Built Modeling. *Computer-Aided Civil and Infrastructure Engineering*, *31*(7), 483–498. http://doi.org/10.1111/mice.12192

FHWA - Federal Highway Administration. (2012). Estimated 2012 Costs to Replace or Rehabilitate Structurally Deficient Bridges. Available at https://www.fhwa.dot.gov/bridge/nbi/sd2012.cfm, accessed 2 May, 2019.

Flaig, K. D., & Lark, R. J. (2000). The development of UK bridge management systems. *Proceedings of the Institution of Civil Engineers - Transport*, *141*(2), 99–106. http://doi.org/10.1680/tran.2000.141.2.99

Highways England. (2018a). Design Manual for Roads and Bridges: Volume 6 Road Geometry. Available at http://www.standardsforhighways.co.uk/ha/standards/dmrb/vol6/index.htm, accessed 2 May, 2019.

Highways England. (2018b). Design Manual for Roads and Bridges. Available at http://www.standardsforhighways.co.uk/ha/standards/dmrb/index.htm, accessed 2 May, 2019.

Hüthwohl, P., Brilakis, I., Borrmann, A., & Sacks, R. (2018). Integrating RC Bridge Defect Information into BIM Models. *Journal of Computing in Civil Engineering*, *32*(3), 04018013. http://doi.org/10.1061/(ASCE)CP.1943-5487.0000744

IfcOpenShell.org. (2018). IfcOpenShell, the open source ifc toolkit and geometry engine. Available at http://ifcopenshell.org/, accessed 2 May, 2019.

Kedar, A. (2016). *SeeBridge Project Document: Criteria for evaluation of complete SeeBridge System*. Available at https://technionmail-my.sharepoint.com/personal/cvsacks_technion_ac_il/Documents/Virtual%20Construction%20Lab/SeeBridge/WP1/Criteria/Deliverable12SeeBridgeEvaluationCriteria%20.pdf, accessed 2 May, 2019.

Kern, W. F., & Bland, J. R. (1948). Spherical segment. *Solid Mensuration with proofs, 2nd ed.* (pp. 97–102). New York, NY: J. Wiley & sons, inc.. Retrieved from https://www.worldcat.org/title/solid-mensuration-with-proofs-2d-ed/oclc/757456815?referer=di&ht=edition#borrow, accessed 2 May, 2019.

Kim, M.-K., McGovern, S., Belsky, M., Middleton, C., & Brilakis, I. (2016). A Suitability Analysis of Precast Components for Standardized Bridge Construction in the United Kingdom. *Procedia Engineering*, *164*, 188–195. http://doi.org/10.1016/j.proeng.2016.11.609

Kobryń, A. (2017). *Transition Curves for Highway Geometric Design*. *Springer Tracts on Transportations and Traffic* (Vol. 14). Cham: Springer International Publishing. http://doi.org/10.1007/978-3-319-53727-6

Kwon, S.-W., Bosche, F., Kim, C., Haas, C. T., & Liapi, K. A. (2004). Fitting range data to primitives for rapid local 3D modeling using sparse range point clouds. *Automation in Construction*, *13*(1), 67–81. http://doi.org/10.1016/j.autcon.2003.08.007

Laefer, D. F., & Truong-Hong, L. (2017). Toward automatic generation of 3D steel structures for building information modelling. *Automation in Construction*, *74*, 66–77. http://doi.org/10.1016/j.autcon.2016.11.011

Lee, G., Sacks, R., & Eastman, C. M. (2006). Specifying parametric building object behavior (BOB) for a building information modeling system. *Automation in Construction*, *15*(6), 758-776 http://doi.org/10.1016/j.autcon.2005.09.009

Limberger, F. A., & Oliveira, M. M. (2015). Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition*, *48*(6), 2043–2053. http://doi.org/10.1016/j.patcog.2014.12.020

Lu, R., & Brilakis, I. (2017). Recursive Segmentation for As-Is Bridge Information Modelling. In *Lean and Computing in Construction Congress - Volume 1: Proceedings of the Joint Conference on Computing in Construction* (pp. 209–217). Edinburgh. http://doi.org/10.24928/JC3-2017/0020

Lu, R., Brilakis, I., & Middleton, C. R. (2018). Detection of Structural Components in Point Clouds of Existing RC Bridges. *Computer-Aided Civil and Infrastructure Engineering*. http://doi.org/10.1111/mice.12407

Moreira, A., & Santos, M. Y. (2006). Concave Hull: A k-Nearest Neighbours Approach for The Computation of The Region Occupied By A Set of Points. Available at http://repositorium.sdum.uminho.pt/bitstream/1822/6429/1/ConcaveHull_ACM_MYS.pdf, accessed 2 May, 2019. *Proceedings of the 2nd International Conference on Computer Graphics Theory and Applications (GRAPP 2007), Barcelona, Spain*.

Network Rail. (2015). Network Rail Bridge List. Available at https://www.whatdotheyknow.com/request/list_of_bridges_on_network_rail, accessed 2 May, 2019.

Ochmann, S., Vock, R., Wessel, R., & Klein, R. (2016). Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics*, *54*, 94–103. http://doi.org/10.1016/j.cag.2015.07.008

Oesau, S., Lafarge, F., & Alliez, P. (2014). Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut. *ISPRS Journal of Photogrammetry and Remote Sensing*, *90*, 68–82. http://doi.org/10.1016/j.isprsjprs.2014.02.004

Parrott, A., & Lane, W. (2017). Industry 4.0 and the digital twin. Available at https://www2.deloitte.com/insights/us/en/focus/industry-4-0/digital-twin-technology-smart-factory.html, accessed 2 May, 2019. *Deloitte University Press*.

Patil, A. K., Holi, P., Lee, S. K., & Chai, Y. H. (2017). An adaptive approach for the reconstruction and modeling of as-built 3D pipelines from point clouds. *Automation in Construction*, *75*, 65–78. http://doi.org/10.1016/j.autcon.2016.12.002

Rabbani, T. (2006). Automatic Reconstruction of Industrial Installations Using Point Clouds and Images. Available at http://resolver.tudelft.nl/uuid:0012068e-93b4-4bd9-a9b3-9c579ae7c91a, accessed 2 May, 2019. *Publications on Geodesy*, *62*(May), 7401–7410.

Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., & Beetz, M. (2008). Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems*, *56*(11), 927–941. http://doi.org/10.1016/j.robot.2008.08.005

Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). *BIM Handbook:* A Guide to Building *Information Modeling for Owners, Designers, Engineers, Contractors, and Facility Managers, Third Edition. Wiley.* ISBN: :9781119287537. http://doi.org/10.1002/9781119287568

Sacks, R., Eastman, C. M., & Lee, G. (2004). Parametric 3D modeling in building construction with examples from precast concrete. *Automation in Construction*, *13*(3), 291-312. http://doi.org/10.1016/S0926-5805(03)00043-8

Sacks, R., Kedar, A., Borrmann, A., Ma, L., Brilakis, I., Hüthwohl, P., … Muhic, S. (2018). SeeBridge as next generation bridge inspection: Overview, Information Delivery Manual and Model View Definition. *Automation in Construction*, *90*, 134–145. http://doi.org/10.1016/j.autcon.2018.02.033

Schnabel, R., Wahl, R., & Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, *26*(2), 214–226. http://doi.org/10.1111/j.1467-8659.2007.01016.x

Song, X., & Jüttler, B. (2009). Modeling and 3D object reconstruction by implicitly defined surfaces with sharp features. *Computers & Graphics*, *33*(3), 321–330. http://doi.org/10.1016/j.cag.2009.03.021

Thomson, C., & Boehm, J. (2015). Automatic geometry generation from point clouds for BIM. *Remote Sensing*. http://doi.org/10.3390/rs70911753

USIBD. (2016). Level of Accuracy (LOA) Specification version 2.0. Available at https://usibd.org/product/level-of-accuracy-loa-specification-version-2-0/, accessed 2 May, 2019. *U.S. Institute of Building Documentation*.

Valero, E., Adán, A., & Cerrada, C. (2012). Automatic Method for Building Indoor Boundary Models from Dense Point Clouds Collected by Laser Scanners. *Sensors*, *12*(12), 16099–16115. http://doi.org/10.3390/s121216099

Vassou, V. (2010). Structures Condition Suevey of Borough Principal Road Network. Technical report. Available at http://www.bridgeforum.org/bof/meetings/bof33/BCI Study_Report_Final.pdf, accessed 3 April, 2019. *London Bridges Engineering Group, Bridge Condition Indicators Project*.

Wai-Fah, C., & Lian, D. (2014). *Bridge Engineering Handbook, Construction and Maintenance*. *CRC Press*. ISBN: 9780849316845. Available at https://www.academia.edu/25903166/BRIDGE_ENGINEERING_HANDBOOK_CONSTRUCTION_AND_MAINTENANCE, accessed 2 May, 2019.

Walsh, S. B., Borello, D. J., Guldur, B., & Hajjar, J. F. (2013). Data Processing of Point Clouds for Object Detection for Structural Engineering Applications. *Computer-Aided Civil and Infrastructure Engineering*, *28*(7), 495–508. http://doi.org/10.1111/mice.12016

Wang, C., Cho, Y. K., & Kim, C. (2015). Automatic BIM component extraction from point clouds of existing buildings for sustainability applications. *Automation in Construction*, *56*, 1–13. http://doi.org/10.1016/j.autcon.2015.04.001

WSDoT. (2009). Precast Prestressed Wide Flange Girders. Available at http://www.wsdot.wa.gov/eesc/bridge/designmemos/14-2009.htm, accessed 2 May, 2019.

Xiao, J., & Furukawa, Y. (2014). Reconstructing the World's Museums. *International Journal of Computer Vision*, *110*(3), 243–258. http://doi.org/10.1007/s11263-014-0711-y

Zhang, G., Vela, P. A., & Brilakis, I. (2014). Automatic Generation of As-Built Geometric Civil Infrastructure Models from Point Cloud Data. In *Computing in Civil and Building Engineering (2014)* (pp. 406–413). Reston, VA: American Society of Civil Engineers. http://doi.org/10.1061/9780784413616.051

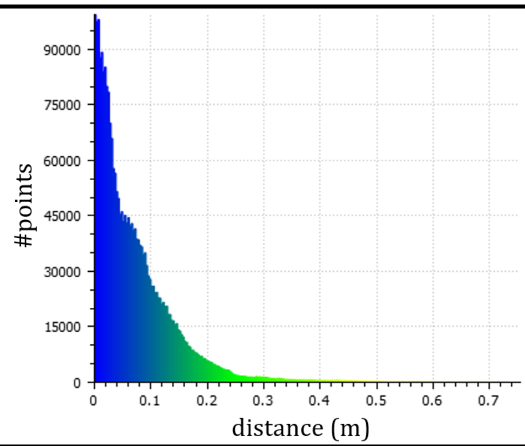
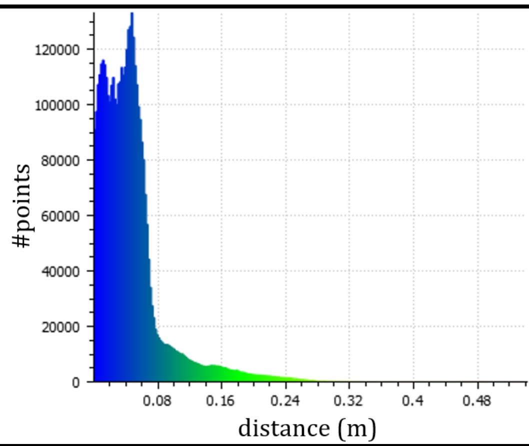
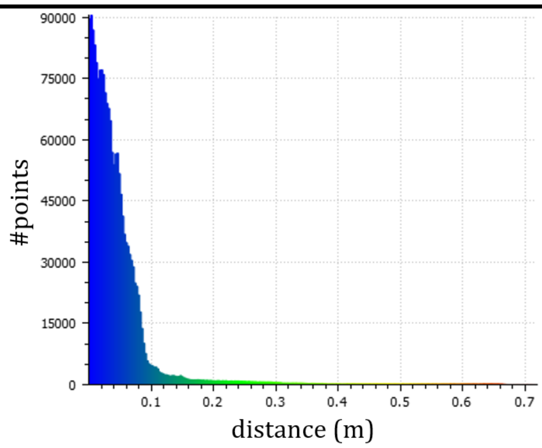
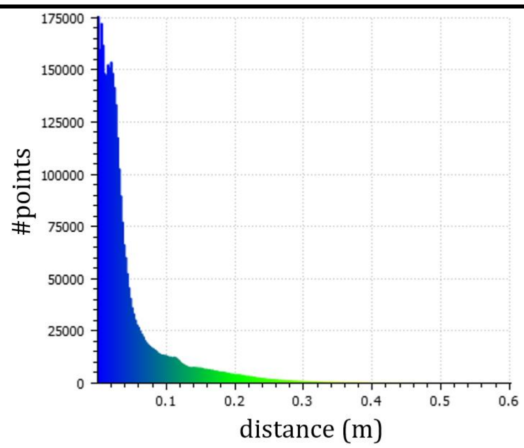
Zhang, G., Vela, P. A., Karasev, P., & Brilakis, I. (2015). A Sparsity-Inducing Optimization-Based Algorithm for Planar Patches Extraction from Noisy Point-Cloud Data. *Computer-Aided Civil and Infrastructure Engineering*, *30*(2), 85–102. http://doi.org/10.1111/mice.12063

Appendix

Table 8 Comparison of C2C distance of ten bridges

(m)	**Bridge 1**				**Bridge 2**				**Bridge 3**			
α/β	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*
$\overline{\mathrm{dist}}_{\alpha/\beta}$	0.040	0.039	0.043	0.041	0.064	0.060	0.073	0.067	0.052	0.050	0.044	0.047
C2C	0.040		0.043		0.064		0.073		0.050		0.047	
(m)	**Bridge 4**				**Bridge 5**				**Bridge 6**			
α/β	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*
$\overline{\mathrm{dist}}_{\alpha/\beta}$	0.073	0.065	0.094	0.074	0.109	0.098	0.049	0.036	0.049	0.023	0.046	0.042
C2C	0.073		0.094		0.109		0.049		0.049		0.046	
(m)	**Bridge 7**				**Bridge 8**				**Bridge 9**			
α/β	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*
$\overline{\mathrm{dist}}_{\alpha/\beta}$	0.157	0.042	0.125	0.055	0.072	0.064	0.037	0.030	0.076	0.098	0.056	0.044
C2C	0.157		0.125		0.072		0.037		0.098		0.056	
(m)	**Bridge 10**											
α/β	*GT/Real*	*Real/GT*	*Auto/Real*	*Real/Auto*								
$\overline{\mathrm{dist}}_{\alpha/\beta}$	0.055	0.036	0.135	0.080								
C2C	0.055		0.135									

Table 9 C2C distance of Auto PCDs in histogram colour map

	Bridge 2	***Bridge 3***
Histogram colour map		
	Bridge 5	***Bridge 6***
Histogram colour map		
	Bridge 8	***Bridge 10***
Histogram colour map