GENERATING BRIDGE GEOMETRIC DIGITAL TWINS FROM POINT CLOUDS

Abstract

The automation of digital twinning for existing bridges from point clouds remains unsolved. Extensive manual effort is required to extract object point clusters from point clouds followed by fitting them with accurate 3D shapes. Previous research yielded methods that can automatically generate surface primitives combined with rule-based classification to create labelled cuboids and cylinders. While these methods work well in synthetic datasets or simplified cases, they encounter huge challenges when dealing with real-world point clouds. In addition, bridge geometries, defined with curved alignments and varying elevations, are much more complicated than idealized cases. None of the existing methods can handle these difficulties reliably. The proposed framework employs bridge engineering knowledge that mimics the intelligence of human modellers to detect and model reinforced concrete bridge objects in imperfect point clouds. It directly produces labelled 3D objects in Industry Foundation Classes format without generating low-level shape primitives. Experiments on ten bridge point clouds indicate the framework achieves an overall detection F1-score of 98.4%, an average modelling accuracy of 7.05 cm, and an average modelling time of merely 37.8 seconds. This is the first framework of its kind to achieve high and reliable performance of geometric digital twin generation of existing bridges.

Introduction

A Digital Twin (DT) is defined as a digital replica of a real-world asset (Parrott & Lane, 2017). The asset could be a tunnel, a building, a bridge, or any other man-made asset of the built environment. A DT differs from and is much more than the traditional Computer-Aided Design. It is based on massive, cumulative, real-time, real-world data measurements across an array of dimensions, and consequent use of a digital model across the entire lifecycle of an infrastructure (Buckley & Logan, 2017). The model comprises both 3D geometry of the infrastructure components as well as a comprehensive set of semantic information, including material, functions, and relationships between the components. It could be further enriched with other information, such as sensor data (Davila Delgado et al., 2017) and damage information (Hüthwohl et al., 2018). This is particularly useful for the asset inspection practice, which is currently based on on-site manual data collection and visual assessment. There is a need for at least 315,000 bridge inspections per annum across the United States and the United Kingdom (UK) given the typical two-year inspection cycle. This explains why there is a huge market demand for less labour-intensive bridge documentation techniques that can efficiently boost bridge management productivity. The greatest value of using DTs is that they are projected to save substantial costs for global infrastructure owners (unlock 15—25% savings by 2025) by automating the inspection process and enabling accurate condition assessments and timely maintenance decisions (Barbosa et al., 2017).

The use of a DT is greatest during the design stage (as-designed), while little use is made in the closeout stage (as-built), and almost absent in the maintenance stage (as-is) (Buckley & Logan, 2017). Almost no as-is DTs are generated, so no expected value is realized (0% US, 2% UK, 1% France, and 0% Germany) (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 that the cost of doing so outweighs their benefits. The following text reviews the current practice of digital twinning. This explains why the DT implementation is so limited.

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 the DT with only geometry data, i.e. gDT. A gDT is generated using raw spatial data, such as the point clouds collected with laser scanners. The adoption of DT is very limited even though there are many capable laser scanning hardware solutions. This is mainly because the manual digital twinning from point clouds is a daunting task. We outline the end-user requirements (*EURs*) of DTs and then provide a brief review of existing software solutions to check their degree of automation regarding the *EURs*.

End-user requirements (EURs)

Developing detailed *EURs* of DTs is outside the scope of this study. This section summarizes the fundamental information that a DT must contain. The end-users of DTs are inspectors, engineers, and the decision makers. The *EURs* define the information that will be required by the end-users from both their own internal team and from suppliers. The *EURs* should clearly articulate the information requirements and describe the expected information deliverables. However, the nature of the *EURs* depends on the complexity of the

project, the experience, and the requirements of the end-users. Experienced end-users may develop detailed *EURs*, whilst others may only set out high-level requirements, and basic rules. Broadly, a DT includes:

- *EUR 1*: Component-level digital representation. A DT should contain the main structural component types of a sensed asset with a component-level resolution (Sacks et al., 2017).
- *EUR 2*: Component's explicit geometry representation and property sets. The full geometry should represent as-is conditions of the sensed asset (Borrmann & Berkhahn, 2018).
- *EUR 3*: Component's taxonomy. The components should be labelled by their element types (Koch & König, 2018).
- *EUR 4*: Component's implicit information such as structural relationships, material, cost, schedule and so on. A DT should be sufficiently semantically meaningful (Sacks et al., 2018; Sacks et al., 2018).
- *EUR 5*: Component's damage information. Damage type (crack, spalling, scaling, efflorescence and others), location, and orientation should be exactly identified and embedded into the DT along with the texture data (Hüthwohl et al., 2018).
- *EUR 6*: All above-listed *EURs* should be presented in a platform neutral data format, such as Industry Foundation Classes (IFC) (Borrmann et al., 2018; Koch & König, 2018).

Major vendors such as Autodesk, Bentley, Trimble and ClearEdge3D, etc. 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 as the *EUR 1 & EUR 2* can be partially automated. However, the spec-driven component library of ClearEdge3D can only recognize and fit point cloud subparts with standardised shapes based on an industry specification table. For other commercial applications, none of them can automate any one of the *EURs*. Modellers must first manually segment a point cloud into subparts, and then manually fit 3D shapes to them (*EUR 1 & EUR 2*). Fitting accurate 3D shapes to the segmented point clusters is challenging because the set of allowable primitives is limited in most software applications (Wang et al., 2015). Modellers need to enrich the resulting gDT with other explicit and implicit information, such as component's taxonomy (*EUR 3*), connectivity and aggregation (*EUR 4*), and defects (*EUR 5*) to meet the *EURs*. Then, all *EURs* need to be exported in IFC format (*EUR 6*).

Bridge Digital Twinning

Real world reinforced concrete (RC) bridge components usually have complicated shapes, containing complex skews, and cannot be simply fitted using idealized predefined shapes. We investigate the entire workflow of digital twinning of a typical RC bridge point cloud using CloudCompare 2.6.2 and Autodesk Revit 2016. Revit provides excellent flexibilities that allow users to design a shape in a more freeform manner. Geometry in Revit's Family consists of solid and void forms in five varieties: Extrusion, Blend, Revolve, Sweep and Swept Blend (Figure 1). Up until the end of the manual operation, only *EURs 1, 2, 3*, and *6* are satisfied. 95% of the total modelling time is spent on customizing shapes and fitting them to the point clusters. The "bottlenecks" of digital twinning using current software applications are listed as follows:
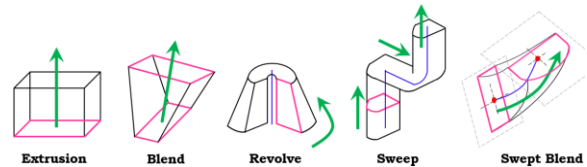

Figure 1: Forms available in Revit Family editor

1) Software packages can semi-automatically extract standardized shapes in point clouds but cannot automatically extract non-canonical shapes. Manual shape customization is necessary, but laborious and time-consuming.
2) *EUR 2* can only be manually achieved. The presence of occlusions and varying density in the data adds hours of adjustment.
3) *EURs 1, 3, and 6* can only be manually achieved and *EURs 4* and *5* are unavailable within existing applications.
4) None of existing software packages can offer a one-stop digital twinning solution. Modellers have to shuttle intermediate results in different formats, giving rise to possibility of information loss.

The following texts investigate existing automated methods in the literature related to *EURs 1, 2, 3*, and *6*, i.e. *EURs* required to generate a gDT with component-level semantic labels. *EURs 4* and *5* are beyond the scope of this paper.

State of Research

We divide existing methods of digital twinning using point clouds into two groups: (1) object detection methods (*EURs 1* and *3*); and (2) 3D solid model fitting methods (*EURs 2* and *6*).

Object Detection in Point Clouds

We define "detection" in the context as the combination of *clustering* (from a point cloud to point clusters) and *classification* (labelling the point clusters). Current methods of point cloud clustering

generally follow a "*bottom-up*" approach, which goes from points to surfaces or patches followed by semantic labelling to derive objects. Most point cloud classification methods follow a "*top-down*" approach, which employs human visual perception such as relationships and contexts to detect specific instances embedded in point clouds or to infer the semantics of components in a geometric model. Real point clouds are imperfect data with many problems, such as *occlusions* and *varying point density*. We review both bottom-up and top-down detection methods and investigate how far they have solved these challenges. Specific limitations are also identified.

Bottom-up detection

The bottom-up approach pieces together low-level primitive features like points to generate higher-level features successively until a top-level system is formed. The higher-level features are surface normal (Sampath, 2010), meshes (Marton et al., 2009), surface planes/patches (Zhang et al., 2015), non-uniform B-Spline surfaces (NURBS) (Dimitrov et al., 2016), and voxels (Vo et al., 2015). Three main methods arise from the literature: RANdom Sample Consensus, Region Growing, and the Hough-Transform paradigm.

RANdom Sample Consensus (RANSAC) is especially used for detecting planar surfaces. Jung et al. (2014) and Arikan et al. (2013) used RANSAC to detect planar surfaces such as walls, floors, and ceilings in building point clouds. Whilst the RANSAC algorithm is effective in the presence of noise and outliers, it has some limitations. First, it suffers from spurious-planes, which are frequently produced around the boundaries (Jung et al., 2014). Second, RANSAC requires prior knowledge about the data, meaning that the selection of a fixed number of shape hypotheses implies that a prior estimate of the inlier ratio is available. This is often not the case in practice. For example, Schnabel et al. (2007) detected plane, sphere, cylinder, cone, and tori with RANSAC using random sampling of minimal sets in a point cloud. Yet, given its computationally-expensive nature, it is unrealistic to use RANSAC to detect complex geometries. Recently, Zhang et al. (2015) presented a novel RANSAC method to detect planar patches in bridge point clouds. Although the experiments indicated this method outperforms baseline methods, it cannot detect pier patches when the point densities of those regions are low.

Region Growing (RG) is also a widely used scheme for point cloud clustering. It starts with a set of initial seeds and then adds in neighbouring points based on similarity of the surface normal (Macher et al., 2017), curvature and so on, until an edge is reached. Walsh et al. (2013) presented an RG algorithm to detect both planar and curved surfaces in bridge point clouds. However, the segmentation was finally achieved after manually choosing key points around the boundaries. Dimitrov & Golparvar-fard (2015) suggested an upgraded RG method which excels when the point cloud does not suffer from substantive occlusions. However, it over-segments objects when non-trivial occlusions are present. The persistent occlusion problem was addressed by Xiong et al. (2013) through a learning-paradigm that can detect occluded planar surfaces and estimate their shapes in building point clouds. However, their method cannot be directly applied to bridge settings, whose occluded surfaces do not follow a specific pattern as in a building point cloud. In general, RG-based methods suffer from occlusion effects, and also have the boundary weakness. These limitations give rise to issues such as over-/under-segmentation, which often requires a certain amount of manual adjustment.

Hough-Transform (HT) is another commonly used clustering method. The major use of HT is in 2D and 3D, where the number of parameters is small. For example, Díaz-Vilariño et al. (2015) used HT to detect the strong horizontal and vertical lines in range image for building opening boundary detection. Adan & Huber (2011) proposed effective HT methods to detect walls in building point clouds. However, HT becomes computationally prohibitive when the number of dimension increases. For example, the HT requires a 5D Hough parameter space for cylinder detection. Rabbani (2006) suggested a two-stage approach to reduce the computational complexity as well as the number of dimension. In general, HT is powerful for detecting simple geometric objects in point clouds. However, HT is sensitive to parameter dimensions and cannot be applied in practice to shapes characterized by too many parameters. This constraint impedes its use in the detection of bridge objects, which often contain skews and imperfections, and cannot be described using generic shapes with limited parameters. The following paragraph reviews a computationally more efficient method.

Octree-Based (OB) methods have been proposed to tackle the issue of computational complexity and reduce the original point cloud size. Su et al. (2016) presented an OB segmentation method designed for piping systems. Truong-Hong et al. (2013) introduced an OB-based technique to automatically extract building façade features in point clouds. Xu et al. (2018) suggested an OB probabilistic segmentation model for construction sites. However, the segmentation accuracy of this method is quite sensitive to the voxel size. This problem was discussed by Vo et al. (2015), who proposed an octree RG-based algorithm for surface patch segmentation in urban environments. Their method can semi-automatically adjust the voxel size using an adaptive octree. However, this method faces the difficulty of patch generation for low point density regions. In general, voxel-based clustering is more computationally efficient than point-based clustering. Yet, voxel size determination remains largely a user-defined task.

Top-down detection

Bottom-up detection schemes are rarely suitable for point cloud classification. Classification through low-level primitives is insufficient since local surfaces, patches or voxels can be labelled as such, but it is difficult to determine whether they belong to the same instance. The intervention of high-level information is required to overcome such challenges. The top-down approach usually combines a set of engineering criteria and classifies objects in point clouds that meet the criteria. Prior studies show that knowledge-based classification methods are robust, as domain-specific information such as known parameters (Ahmed et al., 2014), object instances (Dore & Murphy, 2014), and topological relationships (Koppula et al., 2011), are invariant to factors such as pose and appearance. Su et al. (2016) used a set of connectivity criteria to merge and label industrial components across voxels. Son et al. (2013) proposed a knowledge-based method for detecting industrial plant objects based on the known surface curvature and size of the pipelines. Perez-Gallardo et al. (2017) suggested a semantic model-based system to detect four object classes in an industrial scene using topological information. Laefer & Truong-Hong (2017) leveraged the steel standard library to identify and match the cross-sections of steel frames in point clouds. Recently, Riveiro et al. (2016) used specific topological constraints to segment masonry bridge point cloud through normal clustering. However, this algorithm largely depends on data quality so that it is difficult to generalize it to adapt to large-scale point clouds, which usually suffer from occlusions and non-uniformly-distributed points. Ma et al., (2017) leveraged relationship knowledge and shape features to classify bridge 3D solid objects (Figure 2). However, the input of this method needs to be a geometric bridge model (not a point cloud) without any semantic meaning. In addition, it assumes that the bridge model is developed in a grid system and the pairwise relationship between two objects is well defined. These assumptions are restrictive and make the method less feasible for real-world linear constructions, as bridges, roads, and tunnels usually possess curved horizontal/vertical alignments.
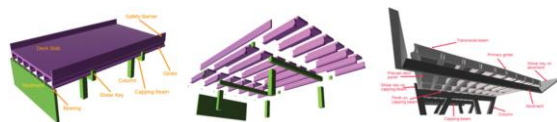

Figure 2: Bridge object classification (Ma et al., 2017)

Other detection methods

Data-driven, learning-based methods have been widely used to detect objects in point clouds. Numerous volumetric Convolutional Neural Network (CNN) and Deep Learning frameworks have been proposed by transforming points into voxel grids. Maturana & Scherer (2015) proposed a supervised 3D CNN called *VoxNet* to classify objects from the volumetric data. Likewise, Qi et al. (2016) suggested a multi-view CNN. Instead of transforming a point cloud into 3D voxel grids, Qi et al. (2017) introduced a deep neural network called *PointNet*, which can directly consume points. However, the major restrictions to applying these learning schemes to infrastructure component detection tasks include: (1) the lack of a sufficient number of labelled large-scale real point clouds to train a good model, and (2) the high computing burden. These methods usually require a substantial down-sampling task before they can be used even in high performance computing systems (e.g. TensorFlow).

Model Fitting to Point Cluster

Model fitting techniques

Model fitting aims to use computer graphic techniques to form the 3D shape of a point cluster. There is no universal solution to describe an object. How to choose a representation totally 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. Existing shape representation methods can be categorized into four groups: Implicit Representation, Boundary Representation, Constructive Solid Geometry, and Swept Solid Representation. We review each of these in the following texts.

Implicit Representation. One solid modelling approach is based on the representation of 3D shapes using mathematical formulations, i.e. implicit functions. Common implicit surface definitions include, but are not limited to Plane, Sphere, Ellipsoid, Torus, Elliptic Paraboloid, and Hyperbolic Paraboloid.

Given that only a very limited number of primitives can be represented exactly by algebraic formulations, implicit functions are of limited usefulness when modelling real-world bridge objects.

Boundary Representation (B-Rep) is a method to describe 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. The geometry of a vertex is given by its coordinates. The edges are straight or curved lines. A face is represented by some description of its surface (algebraic or parametric forms can be used). Valero et al. (2016) developed a method to yield B-Rep models for indoor planar objects (walls, ceilings and floors).

Both Tessellated Surface Representation (TSR) and Polygon/Mesh Representation (PR/MR) can be considered as B-Rep types. A final model of TSR or PR/MR, is represented as a collection of connected surface elements. Oesau et al. (2014) leveraged a graph-cut formulation to reconstruct a synthetic

building point cloud into a mesh-based model. Representing an object using polygonal facets or mesh is the most popular representation in computer graphics. However, there are some problems with polygon mesh models: 1) Level of detail. High resolution could be unduly complex. An option is to reduce the polygon resolution without degrading the rendered presentation. But by how much? 2) Missing data, i.e. occlusions. Large occluded regions are hardly smoothed. Thus, PR/MP does not guarantee that a group of polygons facets can form a closed mesh model. 3) No sense of volume. It is difficult to extract geometric properties such as the radius of a cylindrical column on a mesh representation.

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. The final shape can be represented as the combination of a set of elementary solid primitives, which can be cuboids, cylinders, spheres, cones, and so on. Xiao and Furukawa (2012) introduced an algorithm to reconstruct large-scale indoor environments with a CSG representation consisting of volumetric primitives. However, this method uses only cuboids as volumetric primitives, assuming that they are the most common shapes found in indoor walls. Zhang et al. (2014) (Figure 3) designed a classifier to classify infrastructure components (e.g. pier, beam, deck) and fit them with 3D shape entities (e.g. cuboid, cylinder, sheet) However, this method is tailored for simplified topology designs that do not consider the real bridge geometries. For example, a real sloped slab with varying vertical elevation cannot be simply modelled by a sheet.
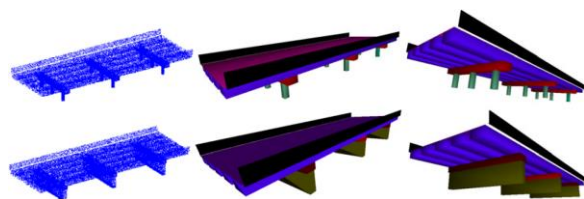

Figure 3: Fitted IFC entities in synthetic bridge point clouds (Zhang et al., 2014)

Swept Solid Representation (SSR) or Extrusion is a representation of model which creates a 3D solid shape by sweeping a 2D profile that is completely enclosed by a contour line along a specific path. Ochmann et al. (2016) presented a method to reconstruct 3D building models from indoor point clouds. Laefer & Truong-Hong (2017) used a kernel-density-estimated-based method to identify and extrude the cross-sections of steel beams in point clouds. The following texts outline the principals involved in representing geometry in IFC standards.

IFC Geometric Representation

The IFC coverage increases along with the end users'

needs. The fundamental feature of DTs is the 3D geometry. According to Borrmann et al. (2018), an object in a DT is initially described as a semantic identity and can then be linked with one or more geometric representations. This ability allows objects in a DT to use application-specific geometric representations. This also provides flexibility to link one or more geometric representations with a semantic object. All geometry representations can be grouped into four classes: Bounding Boxes, Curves, Surface models, and Solid models.

Bounding Boxes are highly simplified geometric representation for 3D objects that are usually used as placeholders. They can be represented using *IfcBoundingBox*. Then, *IfcCurve* and its subclasses can be used to model line objects. Freeform curved edges (i.e. splines) and curved surfaces are required to model complex geometries. *IfcTriangulatedFaceSet* can be used to represent the tessellated surfaces, i.e. polygons with an arbitrary number of edges, or triangular mesh. Curved surface can be described using a finer mesh size, if accuracy is a concern. Specifically, *IfcBSplineSurface* can be used for representing curved surfaces, e.g. NURBS. Then, one classic way to generate 3D objects as solid models is through CSG. *IfcCsgPrimitive3D* and its subclasses can be used. However, the use of CSG is very limited. By contrast, SSR is widely. *IfcSweptAreaSolid* and its subclasses *IfcExtrudedAreaSolid*, *IfcRevolvedAreaSolid*, *IfcFixedReferenceSweptAreaSolid*, and *IfcSurfaceCurveSwptAreaSolid* can be used to present extruded solids. A closed profile *IfcArbitraryClosedProfileDef* is necessary for this representation.

Gaps in knowledge & Objectives

The problem of detecting bridge objects in the form of labelled point clusters from point clouds featuring defects has yet to be solved. Likewise, the problem of fitting 3D solid models in IFC format to real bridge point clusters has yet to be addressed. In addition, the problem of evaluating the quality of a generated gDT has yet to be studied in depth. Therefore, the objective of this research is to devise, implement, and benchmark a framework that can generate labelled geometric models of constructed bridges comprising concrete elements in IFC format.

Proposed framework

We propose a novel top-down framework which exploits bridge engineering knowledge as guidance to directly extract labelled point clusters corresponding to bridge components without generating surface primitives, and then to efficiently reconstruct these labelled point clusters into 3D IFC components. Real-world bridges are neither perfectly straight nor flat. Bridge geometries are defined by horizontal straight or curved alignments, vertical elevations, and varying cross-sections. A slicing-based algorithm is proposed

to tackle these difficulties. The algorithm is repeatedly used throughout the whole framework until all the components are detected and modelled. The algorithm can deal with the skew complexity and can quickly select a set of candidate locations for target objects. The global topology of a bridge can also be well approximated using multiple slices.

We only focus on typical RC slab and beam-slab bridges, because 73% of existing highway bridges and 86% of planned future bridges are RC slab and beam-slab bridges (Kim et al., 2016). In addition, we only deal with four most important and highly detectable components of the two types of bridges, i.e. slab, pier, pier caps, and girders. The framework (Figure 4) consists of two processes: **Process 1.** detection of four bridge component types in point clouds, aiming to meet *EURs* 1 and 3, and **Process 2.** run-time model fitting to the point clusters using IFC standards, aiming to meet *EURs* 2 and 6.

The framework starts with a registered point cloud of an RC bridge. Irrelevant points such as vegetation, trees, traffic, etc. are manually removed. We then align the cropped point cloud using Principal Analysis Component (PCA) such that its centre axis, i.e. horizontal alignment, is roughly parallel to the X-axis of the global coordinate system.
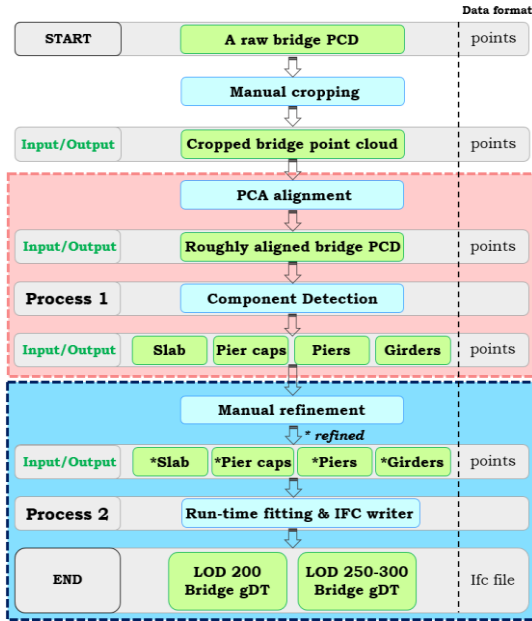

Figure 4: The proposed framework of this research

Process 1 – Bridge component detection

This section (Process 1) proposes a four-step top-down object detection method through which the key components in an RC bridge point cloud are detected. The input is a roughly aligned bridge point cloud. The outputs are labelled point clusters of four component types, i.e. slab, pier cap, pier and girder. The novelty of this method lies in the fact that it directly extracts RC bridge components in point clouds without

generating low-level surface primitives. We use the point cloud of the Nine Wells bridge (Figure 5) in Cambridge, UK to demonstrate the development of the method.
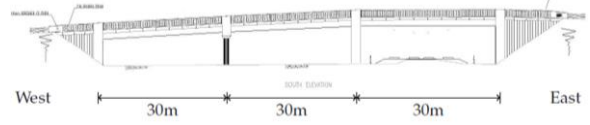

Figure 5: Side view of the Nine Wells bridge

The method breaks down a large bridge point cloud into sub-datasets through a recursive slicing algorithm. That is, it chops the point cloud by means of a 'virtual parallel scalpel' with a specified equal thickness. The first two steps are recursive. The first step segments a whole aligned bridge point cloud (D_N) into the pier assembly (denoted α_M) and deck assembly. The second and third steps detect pier areas (denoted β_{mp}) and pier caps in the pier assembly and deck assembly. The last step detects girders and slab in a merged deck assembly. Note that pier caps and girders may not exist in some bridge point clouds.

Specifically, in Step 1, we chop D_N into multiple slices along the X-axis and extract $range_{j\langle z \rangle}$ which is the height of each slice j (denoted $S_{j\langle x \rangle}$) (Figure 6). We classify $S_{j\langle x \rangle}$ as a pier assembly slice if Eq. (1) is satisfied; otherwise, it is a deck assembly slice:

$$range_{j\langle z \rangle} > \rho_1 |\max\{z_i | D_N\} - \min\{z_i | D_N\}|, \quad (1)$$
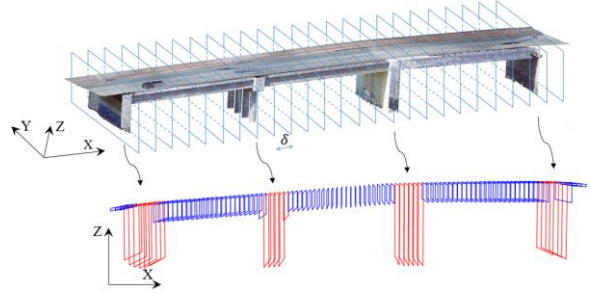

Figure 6: Slicing along X-axis - Step 1

where ρ_1 is discrimination parameter. Each extracted pier assembly α_m is considered as a miniature of D_N, so that Step 2 follows exactly the same strategy as the first step. We replace the right side of Eq. (1) with $\rho_2 |\max\{z_i | \alpha_m\} - \min\{z_i | \alpha_m\}|$, where ρ_2 is another discrimination parameter used to detect the pier area β_{mp} in α_m. Next, Step 3 detects pier caps in $\{\beta_{mp}\}$. This is achieved by removing the uninformative deck points in β_{mp} first followed by investigating the normal direction of the top part of β_{mp}. Up until now, we have detected pier, slab, and pier cap. The final Step 4 aims to detect girders. We first segment the entire deck assembly cluster into several spans

according to the direction of expansion joints. We then use density histograms to detect girders in each span. We merge all over-segments and finally acquire the four labelled point clusters of bridge key components (*EURs* 1 and 3). The details of Process 1 and the four-step object detection method can be found in (Lu et al., 2018).

Process 2 – IFC object fitting

The problem of automatic conversion from the labelled point clusters into 3D solid IFC models remains unsolved, although Process 1 can directly produce labelled point clusters of four component types. Process 2 aims to solve this problem. We propose a slicing-based object fitting method that can twinning an RC bridge into IFC format, using the four types of point cluster constituting the bridge. The inputs are the refined point clusters generated from Process 1. The outputs are two IFC files corresponding to two different levels of detail (LOD 200 and LOD 250-300). The novelty of this method lies in the fact that multiple local topological configurations derived from the slicing scheme can provide good characterization to approximate the global topology of the underlying bridge in a point cloud.

Figure 7 illustrates the workflow of the proposed method, which consists of two major steps: Step 1, geometric feature extraction of point clusters; and Step 2, *IfcObjects* fitting to the extracted features.
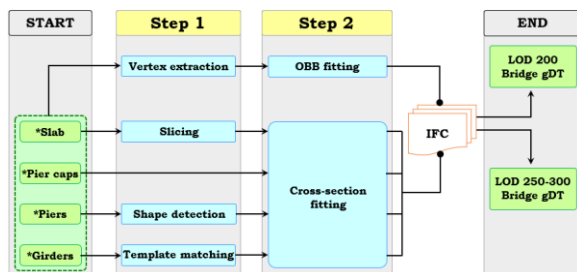

Figure 7: Workflow of the IFC object fitting method

LOD 200 gDT generation. In this twinning phase, the bridge is represented at a coarse level. We use TSR to generate Oriented Bounding Box (OBB) for each point cluster. TSR is an explicit way to present an OBB, whose parallelepiped geometry can be represented using the tessellated geometry model, expressing it as a triangulated tessellation using vertex coordinates. The attributes such as the length, width, and height of each OBB are given and composed into a property set.

LOD 250-300 gDT generation. In this twinning phase, the four point cluster types are represented with detailed geometries through multiple slice models. Solid extrusions are preferred wherever possible if the cross-section of each slice is deemed to be constant. Specifically, for slab point cluster, we implement similar but not identical slicing method to that proposed in Process 1 to slice the deck slab into J
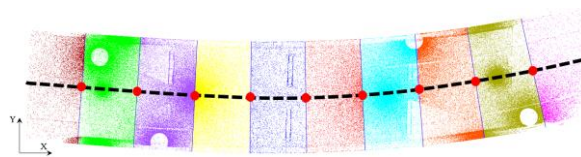

Figure 8: Slab slicing

slices. The slicing does not take a parallel pattern but is rather oriented along the normal direction of the curved alignment of the slab (Figure 8). Then, the problem of modelling the entire slab is transformed into modelling each straight slab slice assuming each slice is straight along the tangent direction and the cross-section of each slice is constant. Similar to how the slab slice is extruded, pier cap point clusters are represented as Swept Solid using the outline of the 2D α-shape on the XY-plane. Next, for pier point clusters, we use a fuzzy-logic algorithm to first classify the cross-section into three categories: circular, quadrilateral, and others. Then, cylindrical piers are represented as Swept Solid, while quadrilateral and other shape piers are represented using multiple slice models. Finally, for girder point clusters, we use a template matching method to find the best-match girder type in existing beam catalogue, assuming that the girders are precast beams. In addition, we create a property set for each component, in which the method can flexibly add the attributes for future use.

Experiments & Results

Data collection & Research platform

We used a Faro Focus 3D X330 laser scanner to collect point clouds of ten RC highway bridges around the city of Cambridge, UK. These point clouds are available at http://doi.org/10.5281/zenodo.1233844. The detailed statistics of the data are given in (Lu et al., 2018). The proposed framework and all relevant algorithms were implemented on Gygax (research coding platform developed by the Construction IT group at the University of Cambridge) on a desktop computer (CPU: Intel Core i7-4790K 4.00GHz, Memory: 32GB, SSD: 500GB).

Implementation & Ground Truth Preparation

We developed a user-defined 2D clipping polygon function on Gygax using Viewport3DX in Helix Toolkit to manually delete irrelevant points. Then, the PCA alignment procedure, object detection method, and IFC object fitting method, were implemented on Gygax as different modules, respectively. We show the implementation of LOD 250-300 gDT generation of two bridges using Gygax GUI in Figure 9.

Three ground-truth (GT) datasets: *GT A*, *GT B*, and *GT C*, were created by manually conducting Step 1, Step 2, and the entire method of Process 1, respectively. The GT datasets are optimally desired outputs to compare against those generated from the proposed method. Then, two sets of models: *GT D* and *GT E*
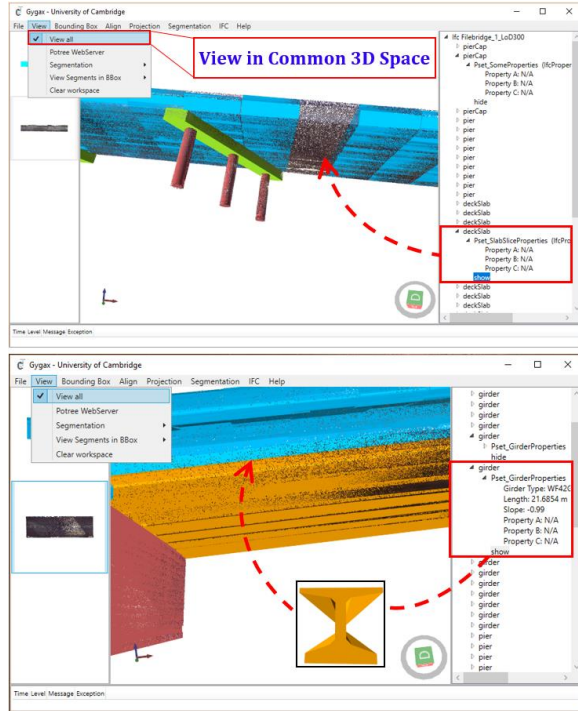
were manually created to compare against the resulting

Figure 9: LOD 250-300 gDT generation. Top: Bridge 1; Bottom: Bridge 7

LOD 200 gDTs and LOD 250-300 gDTs, generated from the automated IFC object fitting method of Process 2, respectively. The modeling times were also recorded. On average, 0.92 and 27.6 hours were spent on generating one gDT in *GT D* and *GT E*, respectively.

Experiments of Process 1 – object detection

The two parameters ρ_1 and ρ_2 were estimated by comparing against the manual detection results of *GT A* and *GT B* using point-wise performance metrics Precision (Pr), Recall (R), and F1-score (F1):

$$\mathrm{Pr}_s = \frac{\mathrm{TP}_s}{\mathrm{TP}_s+\mathrm{FP}_s} = \frac{\text{\# of correctly labelled points in cluster } s}{\text{total \# of points in cluster } s}, \quad (2)$$

$$R_s = \frac{\mathrm{TP}_s}{\mathrm{TP}_s+\mathrm{FN}_s} = \frac{\text{\# of correctly labelled points in cluster } s}{\text{total \# of points in GT cluster } s}, \quad (3)$$

$$F1_s = 2 * \frac{\mathrm{Pr}_s * R_s}{\mathrm{Pr}_s + R_s}, \quad (4)$$

where TP refers to True Positive, FP refers to False Positive, and FN refers to False Negative. '*s*' represents a specific point cluster in Step 1 and Step 2, respectively. We conducted a grid-search over the value space (0, 1) and computed the empirical receiver operating characteristic (ROC). The optimal values of ρ_1^* and ρ_2^* were identified when the distance to the perfect classification in the ROC was minimized. Once the other parameters were deduced based on ρ_1^* and ρ_2^*, we evaluated the entire method. For each bridge, the evaluation was conducted using bounding-box-wise metrics and using similar point-wise metrics as (2), (3), and (4). The average Pr, R and F1 of

bounding-box-wise component detection for all ten bridges were 100%, 98.5%, and 99.2%. For point-wise evaluation, we also computed the micro-average scores:

$$\mathrm{Pr}_{\mathrm{micro}} = \frac{\sum_{s=1}^{|s|} \mathrm{TP}_s}{\sum_{s=1}^{|s|} \mathrm{TP}_s + \sum_{s=1}^{|s|} \mathrm{FP}_s}, \quad (5)$$

$$R_{\mathrm{micro}} = \frac{\sum_{s=1}^{|s|} \mathrm{TP}_s}{\sum_{s=1}^{|s|} \mathrm{TP}_s + \sum_{s=1}^{|s|} \mathrm{FN}_s}, \quad (6)$$

The F1-score is simply the harmonic mean of $\mathrm{Pr}_{\mathrm{micro}}$ and R_{micro}. The micro-average of P/R/F1 was 98.4% for the ten bridge data. Figure 10 only illustrates detection results of *Bridge 1* and *Bridge 7*, due to limited space. To learn how many occlusions are exactly acceptable, we re-conducted experiments using *Bridge 1* by creating arbitrary occlusions. The method achieved high detection rate (F1-score 96.6%), despite the presence of large occlusions (30-40%).
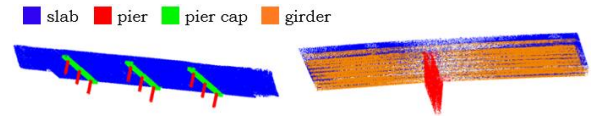

Figure 10: Detection results. Bridge 1 (L) & Bridge 7 (R)

Experiments of Process 2 – IFC object fitting

First, we used the clipping function to manually refine the results from Process 1. This is because the labelled point clusters generated from the object detection method were not perfect. FP positives retained around boundaries between adjacent point clusters. These points need to be removed before implementing the fitting method. The resulting LOD 200 and LOD 250-300 gDTs (Figure 11) were compared against the *GT D* and *GT E*, respectively. Specifically, the evaluation
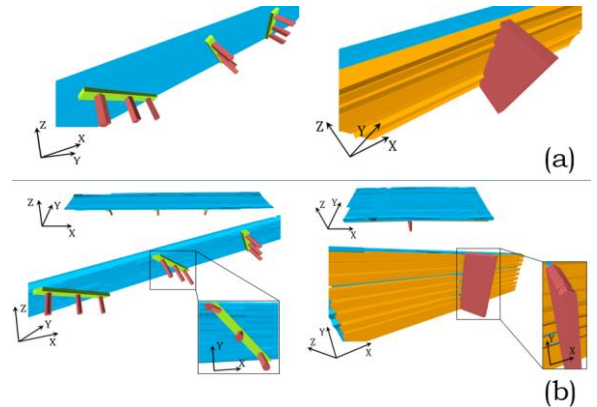

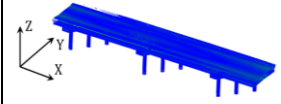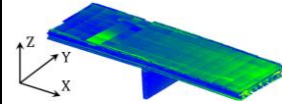*Figure 11: (a) LOD 200 gDTs and (b) LOD 250-300 gDTs. Bridge 1 (L) and Bridge 7 (R)*

of LOD 200 gDTs was based on point-to-point (P2P) distance metrics. We computed the volume and centroid of each GT bounding-box and the automated one. We then computed the Euclidean distance (E_{dc}) between the centroids, the false volume ratio (FVR) between the volumes, and the Euclidean distance between each corresponding vertex. The average value of $\overline{\mathrm{FVR}}$, $\overline{E_{dc}}$, and $\overline{\mathrm{P2P}}$ for ten bridges were 16.5%, 11

cm, and 23 cm, respectively. The average twinning time was 10.2 seconds. By contrast, the evaluation of LOD 250-300 gDTs was based on cloud-to-cloud (C2C) distance metrics. Specifically, the LOD 250-300 gDTs and the *GT E* were first converted into .obj files followed by random sampling dense points from the rendered surface polygons. For each bridge, both its sampled Auto point clouds and GT point clouds were compared against the bridge's original point cloud. The estimated C2C distance between two clouds is the bigger one of the mutual $\overline{\text{dist}}$:

$$\texttt{C2C} = \max(\overline{\text{dist}}_{\alpha/\beta}, \overline{\text{dist}}_{\beta/\alpha}), \tag{7}$$

where α and β represent a compared point cloud and a reference point cloud, respectively. An automated gDT is deemed bettered modelled if its C2C (denoted $\texttt{C2C}_{\text{Auto}}$) is smaller than that of the manual model (denoted $\texttt{C2C}_{\text{GT}}$) and vice versa. Table 1 illustrates the C2C results of *Bridge 1* and *Bridge 7*. The overall $\overline{\texttt{C2C}}_{\text{Auto}}$ of ten bridges gDTs was 7.05 cm while the $\overline{\texttt{C2C}}_{\text{GT}}$ was 7.69 cm. This implies the proposed method outperforms the manual operation. In addition, the average twinning time for ten bridges was 37.8 seconds, reducing the manual time by 95.8%.

Table 1: LOD 250-300 gDTs C2C evaluation

Bridge 1	Bridge 7
$\texttt{C2C}_{\text{GT/Real}}$: 4.0 cm	$\texttt{C2C}_{\text{Auto/Real}}$: 12.5 cm

Conclusions

This paper presents a framework of gDT generation for existing RC bridges using point clouds to meet *EURs 1*, *2*, *3* and *6*. It follows a top-down strategy to directly generate labelled point clusters followed by fitting them with IFC objects. Experiments on ten real bridge point clouds demonstrate the efficiency and robustness of the framework. This is a huge leap over the current practice of digital twinning, which is performed manually. The presented research activities create the foundations for future generating information enriched DTs of existing bridges that can be used over the whole lifecycle of a bridge.

Acknowledgments

References

Ahmed et al. (2014). Automatic Detection of Cylindrical Objects in Built Facilities. *Journal of Computing in Civil Engineering*, 28(3), 1–11.

Arikan, M. et al. (2013). O-snap:Optimization-based snapping for modeling architecture. *ACM Transactions on Graphics*, 32(1), 1–15.

Barbosa, F. et al. (2017). Reinventing Construction: A Route to Higher Productivity.

Borrmann, A. et al. (2018). Industry Foundation Classes: A Standardized Data Model for the Vendor-Neutral Exchange of Digital Building Models. In *Building Information Modeling: Technology Foundations and Industry Practice* (pp. 81–126). Cham: Springer International Publishing.

Borrmann, A., & Berkhahn, V. (2018). Principles of Geometric Modeling. In A. Borrmann, M. König, C. Koch, & J. Beetz (Eds.), *Building Information Modeling: Technology Foundations and Industry Practice* (pp. 27–41). Cham: Springer International Publishing.

Buckley, B., & Logan, K. (2017). The Business Value of BIM for Infrastructure 2017. *Dodge Data & Analytics*, 1–68.

ClearEdge3D. (2017). Structure Modelling Tools.

Davila Delgado, J.M. et al. (2017). *Proceedings of the Institution of Civil Engineers - Bridge Engineering*. 170:3, 204-218

Díaz-Vilariño, L. et al. (2015). 3D modeling of building indoor spaces and closed doors from imagery and point clouds. *Sensors*, 15(2), 3491–3512.

Dimitrov, A., & Golparvar-Fard, M. (2015). Segmentation of building point cloud models including detailed architectural/structural features and MEP systems. *Automation in Construction*, 51(C), 32–45.

Dimitrov, A. et al. (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.

Dore, C., & Murphy, M. (2014). Semi-automatic generation of as-built BIM façade geometry from laser and image data. *Journal of Information Technology in Construction*, 19(January), 20–46.

Hüthwohl, P. et al. (2018). Integrating RC Bridge Defect Information into BIM Models. *Journal of Comp in Civil Eng*, 32(3), 04018013.

Jung, J. et al. (2014). Productive modeling for development of as-built BIM of existing indoor structures. *Automation in Construction*, 42, 68–77.

Kim, M.-K. et al. (2016). A Suitability Analysis of Precast Components for Standardized Bridge Construction in the United Kingdom. *Procedia Engineering*, 164, 188–195.

Koch, C., & König, M. (2018). Data Modeling. In *Building Information Modeling* (pp. 43–62). Cham: Springer International Publishing.

Laefer, D. F., & Truong-Hong, L. (2017). Toward automatic generation of 3D steel structures for building information modelling. *Automation in Construction*, 74, 66–77.

Lu, R. et al. (2018). Detection of Structural

Components in Point Clouds of Existing RC Bridges. In Journal of Computer-Aided Civil and Infrastructure Engineering, 0(0). 2018

Ma, L. et al. (2018). 3D Object Classification Using Geometric Features and Pairwise Relationships. *Computer-Aided Civil and Infrastructure Engineering*, 33(2), 152–164.

Macher, H. et al. (2017). From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings. *Applied Sciences*, 7(10), 1030.

Maturana, D., & Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 922–928). IEEE.

Ochmann, S. et al. (2016). Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics*, 54, 94–103.

Parrott, A., & Lane, W. (2017). Industry 4.0 and the digital twin. *Deloitte University Press*.

Perez-Gallardo et al. (2017). GEODIM: A Semantic Model-Based System for 3D Recognition of Industrial Scenes. In *Intelligent Systems Reference Library* (pp. 137–159).

Qi, C. et al. (2016). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR 2017*.

Qi, C. R. et al. (2016). Volumetric and Multi-View CNNs for Object Classification on 3D Data. *CVPR 2016*.

Rabbani, T. (2006). Automatic Reconstruction of Industrial Installations Using Point Clouds and Images. *Publications on Geodesy*, 62(May), 7401–7410.

Riveiro, B. et al. (2016). Automated processing of large point clouds for structural health monitoring of masonry arch bridges. *Automation in Construction*, 72(3), 258–268.

Sacks, R. et al. (2017). Semantic Enrichment for Building Information Modeling: Procedure for Compiling Inference Rules and Operators for Complex Geometry. Journal of Computing in Civil Engineering.

Sacks, R. et al. (2018). BIM Handbook: A Guide to Building Information Modelling For Owners, Managers, Designers, Engineers and Contractors Third Edition. *John Willey & Son Inc*.

Sacks, R. et al. (2018). SeeBridge as next generation bridge inspection: Overview, Information Delivery Manual and Model View Definition. *Automation in Construction*, 90, 134–145.

Sampath, A., & Jie Shan. (2010). Segmentation and Reconstruction of Polyhedral Building Roofs From Aerial Lidar Point Clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 48(3), 1554–1567.

Schnabel, R. et al. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2), 214–226.

Son, H. et al. (2015). 3D reconstruction of as-built industrial instrumentation models from laser-scan data and a 3D CAD database based on prior knowledge. *Automation in Construction*, 49, 193–200.

Su, Y. et al. (2016). Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 113, 59–74.

Truong-Hong, L. et al. (2013). 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(2), 112–129.

Valero, E. et al. (2016). Semantic 3D Reconstruction of Furnished Interiors Using Laser Scanning and RFID Technology. *Journal of Computing in Civil Engineering*, 30(4), 04015053.

Vo, A.-V., Truong-Hong, L., Laefer, D. F., & Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104, 88–100.

Walsh, S. B. et al. (2013). Data Processing of Point Clouds for Object Detection for Structural Engineering Applications. *Computer-Aided Civil and Infrastructure Engineering*, 28(7), 495–508.

Wang, C. et al. (2015). Automatic BIM component extraction from point clouds of existing buildings for sustainability applications. *Automation in Construction*, 56, 1–13.

Xiao, J., & Furukawa, Y. (2014). Reconstructing the World's Museums. *International Journal of Computer Vision*, 110(3), 243–258.

Xiong, X. et al. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, 31, 325–337.

Xu, Y. et al. (2018). Voxel-based segmentation of 3D point clouds from construction sites using a probabilistic connectivity model. *Pattern Recognition Letters*, 102, 67–74.

Zhang, G. et al. (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.

Zhang, G. et al. (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)