Highlights

Automated Geometric Digital Twinning of Bridges from Segmented Point Clouds by Parametric Prototype Models

M. Saeed Mafipour, Simon Vilgertshofer, André Borrmann

- A reverse engineering approach is proposed for the parametric modeling of bridges.
- Parametric Prototype Models (PPMs) are introduced to describe bridge point clouds.
- Local and global optimization problems are defined to adjust and assemble PPMs.
- Metaheuristic optimization algorithms are utilized to derive parameter values.
- The method is validated with the point cloud of six bridges in Bavaria, Germany.

Automated Geometric Digital Twinning of Bridges from Segmented Point Clouds by Parametric Prototype Models

M. Saeed Mafipour*, Simon Vilgertshofer, André Borrmann

Chair of Computational Modeling and Simulation, Leonhard Obermeyer Center, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany

---

Abstract

Digital Twins (DTs) provide a promising solution for the maintenance and operation of bridges, thanks to their ability to mirror physical/structural conditions. A bridge DT generally consists of a geometric-semantic model whose creation, however, requires extensive manual effort. This paper presents an automated framework to generate the parametric model of bridges from their segmented point clouds. Following the concept of reverse engineering with parametric modeling, Parametric Prototype Models (PPMs) are proposed as tools to extract parameter values from point clouds. A local and global optimization problem is defined to adjust and assemble PPMs into an integrated model. The proposed approach has been validated by applying it to the point cloud of bridge components as well as point clouds captured from six concrete bridges in Bavaria, Germany. The results show that the proposed approach can generate the parametric model of bridges with a mean absolute error (MAE) of 8.71 cm.

Keywords: Digital Twin, Parametric Modeling, Reverse Engineering, Metaheuristic Algorithms

---

*Corresponding author

1. Introduction

The transportation system of countries generally relies on road infrastructure, including bridges that have often been constructed decades ago. To enable the long-term operation of bridges, the National Bridge Inspection Standards (NBIS) require transportation agencies to evaluate the status of bridges over their service life [1]. In current practice, condition assessment and bridge evaluation are primarily conducted manually, which, in turn, increases the operation and management costs. The current ASCE report card [2] asserts the deterioration rate of existing bridges has exceeded the rate of repair and rehabilitation as the conventional methods cannot adequately provide a mechanism for efficient coverage of all bridges. To reduce the costs associated with the maintenance, management, and operation of bridges, the conventional methods for bridge evaluation and quality assurance can be supported by digital methods [3, 4].

Building Information Modeling (BIM) plays a prominent role in the Architecture, Engineering, and Construction (AEC) industry by providing the geometric-semantic representation of assets. In the infrastructure domain, bridges, as critical structures, have been widely investigated for developing bridge information modeling (BrIM) in the as-designed, as-built, and as-is phases [5, 6, 7]. BrIM provides a comprehensive 3D demonstration for Accelerated Bridge Construction (ABC), Virtual Design and Construction (VDC), and structural analysis. A detailed comparison by Kumar et al. [8] illustrated the significant advantage of using BrIM over conventional approaches by implementing three bridge projects by spending five times less time. In addition to the as-designed and as-built phases of bridges, BrIM has been highly beneficial in the as-is phase for the inspection and structural health monitoring (SHM) [9, 10]. BrIM facilitates the identification of the exact location of sensors and enables automated sensor data inventory into the model [11, 12]. It presents a connector to systematically interpret and visualize SHM data on a 3D model that can be used appropriately for the instant analysis of the structure. The same applies to manual inspections and the localization of identified defects and damages. Compared with traditional 2D drawings, BrIM provides a more comprehensive representation in a 3D environment with the capability of continuous semantic enrichment at various levels. This model can be shared with the involved teams in the project and is used for more accurate decision-making on the possible rehabilitation of the structure.

Most recently, bridge information models have been extended to the concept of "Digital Twin" (DT) models [13]. The DT concept, established originally in the manufacturing industry [14], promises a substantial improvement in extending the life cycle of bridges by providing a coherent digital replica mirroring the physical reality, including the current status of the actual asset [15]. A DT is defined purposefully based on its anticipated applications, serving the use cases and requirements that generate a DT for a specific domain. The prominent feature of a DT is its capability to be linked with the actual asset through an access point to handle bidirectional updates. The interval of these updates might differ depending on the asset type and the desired use cases [15].

Nonetheless, a DT must be capable of receiving and handling the required updates to provide an up-to-date representation of the actual asset. A bridge DT can be as simple as a 2D map representing the general but up-to-date information of the bridge or as complicated as a 3D geometric model that includes all the cracks and spalling on the structure, as well as the state of the interior systems, such as pre-spanning cables. The DT will typically inherit all the features of BrIM, is linked with the Bridge Management System (BMS), and reflects the impact of the external factors on the structure [16]. All these features enable DT to perform as an efficient digital representation for supporting and facilitating the operation and maintenance of bridges.

Photogrammetry and Terrestrial Laser Scanning (TLS) are two primary geodetic techniques commonly used to capture existing bridges due to the low manual effort required. Both techniques produce point cloud data (PCD), however, with varying levels of accuracy and density. A comparative analysis of accuracy and reliability by Mohammadi et al. [17] demonstrated the capability of both methods in the digital twinning of bridges. TLS can generate PCD

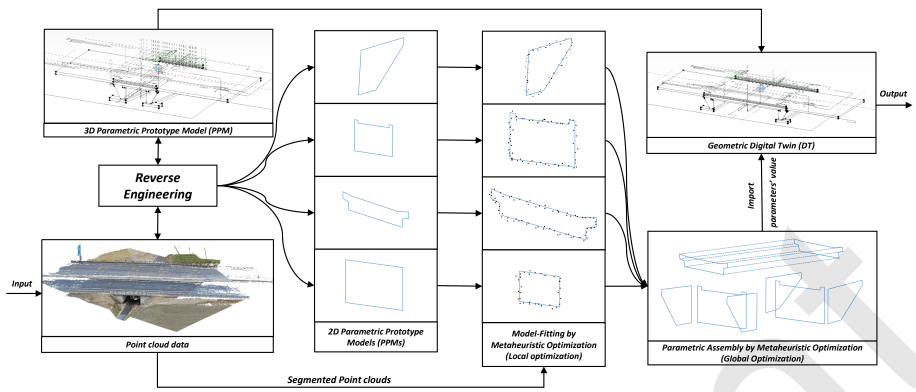


Figure 1: Proposed pipeline for geometric digital twinning of bridges.

of bridges with very high measurement accuracy and level of density. At the same time, aerial photogrammetry is more cost-effective and appropriate for capturing hard-to-reach or unreachable areas of an asset.

Despite the significant benefits of bridge DTs, the manual PCD-based creation of the required geometric model is labor-intensive and error-prone. To handle this challenge, this paper presents an automated framework, as shown in Figure 1, to generate the parametric model of existing bridges from their segmented PCD. Following a reverse engineering approach with parametric modeling, Parametric Prototype Models (PPMs) are proposed to represent the bridge or the bridge component geometry. These dummy models are created based on a set of parameters as well as constraints and fed by analyzing the bridge point clouds. PPMs are constant in type; however, their geometry can be adjusted/updated based on the input value of parameters. They are created purposefully to end up with the anticipated geometric DT model at the start of the process. Leveraging the parametric design of PPMs, a list of candidates is generated and adjusted through a local metaheuristic optimization to fit them into the point cloud of bridge elements. To assemble the fitted PPMs, the extracted parameters from the pieces are integrated through a global metaheuristic optimization. To generate the model of the entire bridge, the extracted parameter values are injected into the 3D PPM of the bridge. As a result, an inherently consistent geometric-semantic model is obtained that not only resembles the input bridge point cloud but also preserves all the relations and dependencies between the bridge components. The prospected benefit of this approach for end users is the massive reduction of the effort to create the geometric model of the bridge from PCD.

The key contributions are highlighted as follows:

- The proposal of PPMs as tools to extract the value of parameters from bridge components that cannot be defined simply in a closed-form formulation.
- The definition of local and global optimization problems over the PPMs to handle the model-fitting problem even in cases with large occlusion.
- The introduction of metaheuristic/evolutionary algorithms as techniques to solve the model-to-cloud fitting optimization problems.
- The description of a framework for the parametric assembly of bridge elements to achieve a parametric and highly flexible model for handling geometric updates and further refinements.

This paper is structured as follows: Section 2 outlines related works in the scope of geometric digital twinning or modeling bridges and the theoretical background of the proposed method. Section 3 describes a novel method for the piece-wise parametric modeling of bridge elements from PCD. This section further addresses the assembly problem for geometric digital twinning of the entire bridge. Section 4 develops the required algorithms to process segmented point clouds and proposes a metaheuristic algorithm to solve the model-to-cloud fitting problem. Section 5 demonstrates the real-world applications of the proposed approach and quantifies its precision in the parametric modeling of six single-span bridges as well as other bridge components. Section 6 compares the proposed method with other existing methods and evaluates its performance in point clouds with occlusion. This section also demonstrates the editability of the model for further refinements. The paper finally ends with a conclusion in Section 7 discussing the development of our research, the significant findings, including known limitations, possible generalizations, and topics for future research.

2. Background and related research

This section presents an overview of the techniques used to create a parametric prototype model (PPM) as a basis for model fitting. Furthermore, a summary of various existing methods to automate the generation of 3D geometry and parametric models from PCD is provided. On this basis, the novelty of the presented approach is highlighted in comparison to similar methods.

2.1. Parametric and procedural modeling

Parametric modeling is a solid modeling approach used in creating geometric models. This concept was developed in the 1990s [18] to capture design intent based on a set of features and constraints. While applied primarily in mechanical engineering, the concept has also been increasingly used to create adaptable models of infrastructure facilities [19, 16]. Two-dimensional parametric sketches form the basis of a parametric model. They are composed of geometric objects and parametric constraints. In a parametric model, particular dimensions such as positions, heights, and widths are defined using variables instead of fixed numerical values. This feature aids designers in altering a design or exploring different variants immediately, as shown in Figure 2. The set of parametric constraints that all major constraint solvers implement is defined as the standard geometric constraint language [20]. It comprises the dimensional constraints for distances and angles and geometric constraints to preserve the geometric shape.

The core concept of procedural modeling is to store not only the outcome of a modeling process but also the sequence of creating sketches and modeling operations, called the model construction history. Models created this way are called procedural models or construction history models. They use the concept of parametric modeling to create flexible 2D sketches. These sketches form the

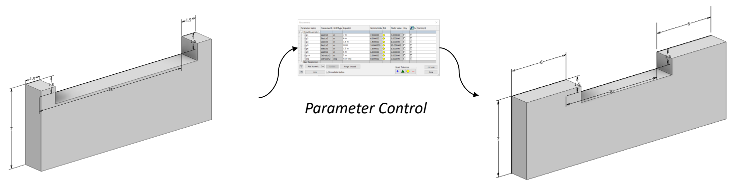


Figure 2: Adjusting parameters value of a parametric model.

basis for the procedural operations that generate 3D geometry by Extrusions, Sweeps, Lofts, or Boolean operations [21].

2.2. Reverse engineering in CAD

Reverse engineering is the process of dismantling a system or model to realize how it accomplishes a task. In computer-aided design (CAD), reverse engineering has been a fundamental problem addressed with various techniques over the years [22, 23]. All reverse engineering processes consist of three basic steps: *Information Extraction*, *Modeling*, and *Review*. Information extraction is the process of gathering information from the desired system. Modeling is acquiring and combining data to create the geometry, and review is the testing process of the resulting model. Reverse engineering can facilitate the model creation process from scanned data through parametric modeling. Depending on the model type the scan data represents, the parametric model of the object can be created. Due to the parametric design of the model, it can be compared (reviewed) with the scanned data and be further altered to reach a higher level of similarity. Recently, this CAD approach has also been of interest to leverage prior knowledge about the topological and existing rules in PCD to model the geometry of objects [24, 25].

2.3. Metaheuristic and evolutionary optimization

Metaheuristic and evolutionary computation is a sub-field of artificial intelligence and soft computing to solve optimization problems, especially with incomplete or imperfect data information [26]. The evolution of biological and natural systems has inspired most metaheuristic algorithms [27, 28]. Contrary to gradient-based optimization algorithms, metaheuristic algorithms are derivative-free and not dependent on the closed-form formulation of the objective function. This feature enables them to optimize nonlinear, multi-modal, and multivariate functions whose derivatives are not computable. Metaheuristic algorithms, similarly to other optimization techniques, require an objective/fitness function to evaluate the quality of the model. Most metaheuristic algorithms such as Particle Swarm Optimization (PSO) [29], Genetic Algorithm (GA) [30], Teaching Learning Based Optimization (TLBO)[31], Grey Wolf Optimizer (GWO) [32], and Firefly Algorithm (FA) [33] are population-based. This means a list of solutions/candidates is proposed initially based on the problem space (discrete or continuous) and the ranges of the parameters. This list is further improved by considering the fitness function value and the algorithm strategy. Finally, the best solution is reported as the global optimum location in the space of the problem.

2.4. State of the Art

Various methods have been proposed to model the geometry of three-dimensional bodies from PCD automatically or semi-automatically. The proposed approaches generally provide the inputs for solid modeling approaches to represent a geometry with a desired level of abstraction or details. Leveraging the closed-form description of primitive shapes and providing an objective function to evaluate the closeness of primitives to points, various techniques have been proposed to address the model-to-cloud fitting problem. On top of them, the RANdom SAmple Consensus (RANSAC) algorithm [34], Hough transform [35], and least squared optimization algorithms [36] can be mentioned. Most recently, deep learning models have also been capable of using the objective function of primitive shapes to automate the simultaneous semantic segmentation and geometric modeling of primitive shapes [37]. B-rep methods have also been used to construct low-semantic and generic models such as meshes/patches from point clouds to address the emerging challenges of modeling more complex shapes whose description by a closed-form formula is cumbersome. To reduce the unwanted complexity of meshes in modeling and storing the geometry, bounding hulls such as convex hull [38], α-shape [39], x-hull [40], concave hull [41], crust [42], etc. have been introduced as well. These methods generally result in the explicit representation of the boundary points. They can illustrate the geometry of complicated shapes solely or in combination with CAD functionalities such as extrude, loft, rotate, and sweep. They, however, cannot simply/directly provide meaningful information about the required parameters to create a volumetric model.

Lu and Brilakis [43] created the geometric digital twin of bridges from point clouds using a 2D ConcaveHull α-shape method [41] and generated 3D shapes using Industry Foundation Classes (IFC). Zhang et al. [44] detected the planar patches from noisy point clouds and determined the boundaries of each patch by the α-shaped algorithm. Wang et al. [45] employed the M-estimator SAmple Consensus (MSAC) algorithm to detect the planar faces and extracted the value of parameters from regular and irregular shapes through a line detection algorithm. Yang et al. [46] employed the principal component analysis (PCA) algorithm to detect the alignment of elements and extracted the value of parameters using the RANSAC algorithm [47]. Dimitrov et al. [48] proposed an approach for successively fitting uniform B-Spline curves to the two-dimensional cross-section of point clouds. Kwon et al. [49] described a heuristic method for extracting the value of parameters from primitive shapes such as cuboids and cylinders. Justo et al. [50] generated the IFC model of truss bridges using bounding boxes of instance-segmented point clouds and collision of elements. Valero et al. [51] detected the planer surfaces in the point clouds and determined the value of parameters by measuring the distance between planes. Oesau et al. [52] proposed a rough feature preserving multi-scale line fitting and a graph-cut formulation to reconstruct a building point cloud into a mesh-based model. Rabbani [36] proposed a method based on least-squared optimization to model a piping system from its point cloud. Patil et al. [53] suggested an area-based adaptive hough transform to estimate single and multiple cylinder orientations and reconstructed piping networks by finding the connection relationships between pipes. Walsh et al. [54] segmented the point cloud of structural elements using features such as normal vectors, curvature, and connectivity of points and extracted the value of parameters from primitive shapes using a least-squares optimization algorithm. Laefer and Truong-Hong [55] proposed a kernel density estimation (KDE) algorithm to detect the density signal of steel profiles and match them with the standard sections in a catalog. Yan and Hajjar [56] employed the RANSAC algorithm to detect the plane surfaces of steel profiles and model the super-structure components of bridges. Kim et al. [25] presented an approach based on reverse engineering for the segmentation of pipe point clouds through deep learning models and employed a 3D matching system to reconstruct 3D plant models. Li et al. [37] described a deep learning model to segment and estimate the parameter values of primitive shapes from point clouds. Barazzetti [57] proposed an approach for the parametric as-built model generation of complex shapes from point clouds using NURBS curves and surfaces.

2.5. Research gaps

Despite the impressive progress in the geometric digital twinning of bridges, several research gaps still exist. Some of the limitations and the parts requiring further investigation are mentioned below:

- Modeling complicated geometries in bridges, such as the deck, abutment, and parapet, has not been addressed parametrically.
- The proposed algorithms have been mostly following a bottom-up approach. They, thus, require many problem-specific thresholds, and their performance is affected by occlusion.
- The final 3D model is not a parametric model in most similar works, i.e., the model cannot receive geometric updates while this is the core feature of a geometric DT.

- It has not been adequately investigated how the elements are assembled into a coherent model. This aspect is even more relevant when the components are parametric, and the final model must preserve its parametric consistency.

This paper addresses these research gaps by proposing a reverse engineering approach to creating PPMs and optimizing them to achieve the desired model of the entire bridge.

3. Methodology

Reverse engineering with parametric modeling is a technique commonly used in the industry to convert scanned data to a CAD model. Reverse engineering proposes the desired final model to achieve at the beginning of the process, while parametric modeling keeps the model adjustable for the required reviews. Through these techniques, the initial model can be compared and become closer in shape to the scanned data by adjusting the value of parameters. Considering the desired model of the bridge, parametric prototype models (PPMs) are designed in this section and used to extract the value of parameters from point clouds. The optimized PPMs are then assembled, and the resulting parameters are imported into the initial model to generate the parametric model of the entire bridge.

3.1. Parametric prototype model

A parametric model includes several parameters through which it can be altered. Also, it comprises a set of constraints that control and preserve the object's shape while being updated. In 3D modeling software, the parametric modeling process is started mainly by drawing 2D sketches on reference/working planes. These 2D sketches are refined and used by functionalities such as extrude, sweep, loft, and rotation to create a volumetric 3D model. Inspired by this process, we define a Parametric Prototype Model (PPM) as a dummy model comprising human-definable parameters and constraints that can update

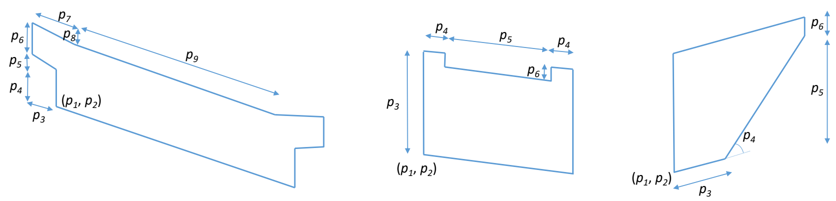


Figure 3: Various examples of PPMs.

the shape. Figure 3 shows three typical PPMs constructed by a set of parameters and constraints describing the geometric shapes. Parameters include the coordinate of origin, length of the edges, and angles, while geometric constraints might consist of horizontal, vertical, perpendicular, coincident, etc., constraints to restrict the geometry.

In particular, a PPM has three features. It contains a finite number of parameters and constraints, has a specific object type, and is a function of parameter values. For instance, a 2D rectangular PPM must be described with only four parameters, including the coordinate of origin (O_x, O_y), length, and width, as this object type has these parameters in the definition. It must also be a function of the parameter values, i.e., it can update its shape with new values of a parameter, such as width.

Contrary to the conventional model fitting methods, PPMs pave the way to fitting into not only the point cloud of simple geometries but also more complicated geometries that commonly exist in bridges. The programming process of a PPM is started from an origin and extended to other vertices based on the value of parameters. Concurrently, constraints such as parallelism, connectivity, perpendicularity, and symmetry are implicitly applied to the prototype model. Using Object-Oriented Programming (OOP) as an analogy, the PPM of an element is the instance of a class containing attributes such as dimensional values (i.e., parameter values) and constraints. Objects generated from the class will have different parameter values.

Figure 4 shows the PPM of a typical bridge deck described by a set of parameters. As can be seen, any change in the value of parameters leads to

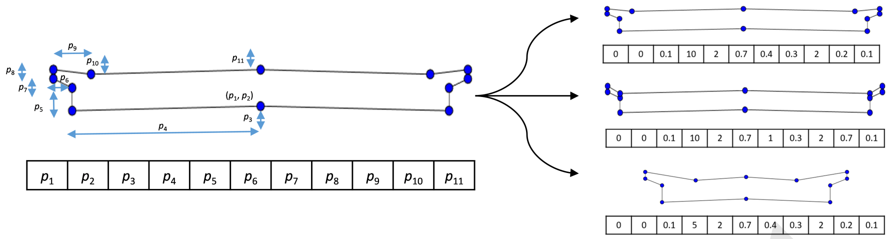


Figure 4: PPM of a typical bridge deck.

an instance of the bridge deck class with new dimensions. Considering a point cloud associated with this bridge deck, a list of candidates/solutions can be created and proposed for the value of dimensions the point cloud represents. To determine the value of parameters through a PPM, each candidate needs to be quantified based on its similarity to the point cloud. To this end, a fitness function is defined in the next section and optimized by a metaheuristic algorithm.

3.2. Model-to-cloud fitting

A PPM is defined numerically based on a set of parameters and constraints. Therefore, the mathematical model of the PPM cannot be expressed and derived simply by a gradient-based algorithm. To address this issue, metaheuristic algorithms can be employed to adjust PPMs and fit them into the point cloud of elements. To instantiate a PPM, random values can be generated in predefined ranges inspired by bridge engineering knowledge. To fit a PPM, the shortest Euclidean distance of the edges to the point cloud must be minimized.

Considering a set of points $\mathcal{S} = \{s_i | i = 1, ..., n\}$, where $s_i \in \mathbb{R}^2$, and a 2D PPM described by a set of parameters $\mathcal{X} = \{x_r | r = 1, ..., m\}$ with the lower bound l_r and upper bound u_r, in which $x_r \in [l_r, u_r]$, the following objective/fitness function can be defined in the term of mean absolute error (MAE):

$$F(x_1, ..., x_m) = \frac{1}{n} \sum_{i=1}^{n} e_i, \tag{1}$$

where e_i is a positive value describing the shortest distance of the i^{th} point to the edges and vertices of the PPM.

A PPM can typically have any position with respect to points set in the space of the problem. The aforementioned function is capable of minimizing the distance of points to the edges of the PPM. However, it cannot guarantee all the edges are fitted into the point cloud. This is because some edges might not find any point in their vicinity. Thus, no point exists to apply a value of error to such edges, and the corresponding parameters to the location of these edges cannot be adjusted during the optimization process. In other words, these edges have a redundant degree(s) of freedom that must be closed.

This case is even intensified in occluded point clouds in which some parts are empty of points. To improve the performance of the optimization algorithm and enable it to handle occlusion, the concept of *active* and *passive* edges is proposed.

Definition: An edge is called *active* when it has at least one of the following conditions: 1. It possesses at least two points, or 2. It possesses at least a point and has a slope constraint. In any other conditions, the edge is called *passive* as it does not have enough points or constraints to contribute to the optimization process.

To activate the passive edges of a PPM with the number of k edges, a new penalty term $(\lambda_j e'_j)$ is defined for each edge j and added to the previous fitness function as follows:

$$F(x_1, ..., x_m) = \frac{1}{n}\sum_{i=1}^{n} e_i + \frac{1}{k}\sum_{j=1}^{k} \lambda_j e'_j, \tag{2}$$

where λ_j is a binary value controlling the activity of edges, i.e., 0 for active edges and 1 for passive edges, and e'_j is the value of error required to activate the passive edges.

Considering the shortest distance of points to the edges, subsets of points can be created and assigned to each edge. Thus, the first term of the fitness function can be rewritten for the edges, and the following simplified fitness function is achieved:

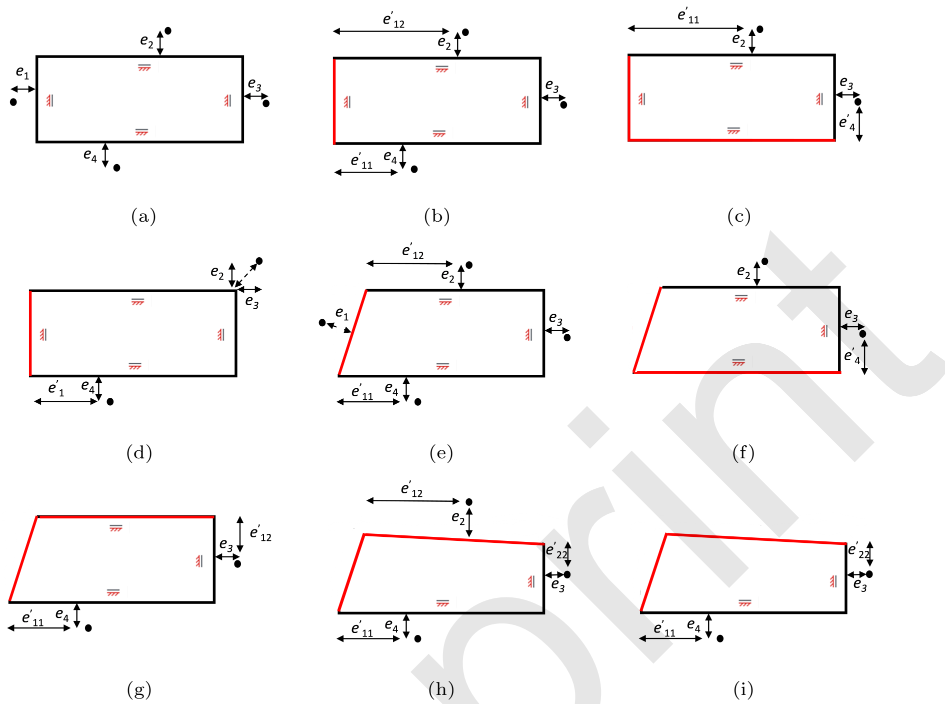
$$F(x_1, ..., x_m) = \frac{1}{k}\sum_{j=1}^{k}(e_j + \lambda_j e'_j), \tag{3}$$

where e_j is the total distance of the edge j to its nearest points.

To determine whether an edge is active or passive, the number of points assigned to the edge must be counted during the optimization process; this number might vary as the PPM moves onto the plane and updates its shape. Also, the slope constraints of the edge, such as vertical and horizontal constraints, must be controlled; these constraints are constant. Using such information and considering the definition of the active edges, the passive edges can be detected and activated.

To activate a passive edge, the two neighboring edges of the passive edge are considered, and the value of e'_j is calculated accordingly. Figure 5 shows a PPM with four edges in different model-fitting scenarios. Assume the edges of the PPM have been assigned the index $j = \{1, 2, 3, 4\}$ from the left edge in clockwise order. Figure 5a depicts a rectangular PPM as all the edges of the PPM have horizontal or vertical constraints. As can be seen, there is a point close to each edge of the PPM; thus, the edges possess a point. Considering the relative position of points with respect to the edges and constraints controlling the slope (one point and a slope constraint), all the edges are active, and the value of error is only the shortest distance of edges to the points, i.e., no additional value of error (penalty) is required to be added ($\lambda_j = 0$). Figure 5b shows another scenario in which the left edge has no point and only has a vertical constraint. Since this edge has only a constraint and no point, it cannot be involved in the optimization process (a passive edge).

To activate this edge and close its translational degree of freedom, a single point needs to be assigned to this edge from the neighboring edges to meet the condition of one point and a slope constraint. As both neighboring edges have a point and the edge has a vertical constraint, a value of error equal to the minimum distance of the left edge (passive edge) to the closest point of the



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 5: Various scenarios of fitting a typical PPM into a set of points.

neighboring edges is added ($e'_1 = min(e'_{11}, e'_{12})$). Figure 5c illustrates another case in which the bottom and left edges are passive. However, they both have at least a neighboring edge with a point through which they can be activated (one point and a slope constraint). In Figure 5d, the left, top, and right edges have no points; however, the bottom edge, the right endpoint of the top edge, and the top endpoint of the right edge possess a point. In this case, the point belonging to the endpoints can activate the corresponding edges, i.e., the top and right edges are still active. Nonetheless, this point cannot be used for activating the neighboring edges. Thus, the left edge is only activated based on the point belonging to the bottom edge.

Figure 5e demonstrates a PPM in which the left edge has no slope constraint. Even though this edge possesses a point, it is still passive, as it needs one more point to satisfy the condition of two points. This edge can be activated by adding the mean value of errors ($e'_1 = mean(e'_{11}, e'_{12})$). Figure 5f also illustrates a PPM in which the left and bottom edges are passive due to a lack of points. Although the bottom edge has a constraint and only a point from the neighboring edge is sufficient to activate it, the left edge has neither a constraint nor two points from each neighboring edge to reach the activity condition of two points. Thus, model fitting is impossible in this case, as the slope of the left edge cannot be recognized. As a result, a high error value ($e'_1 = 10e^3$) is added to decrease the selection probability of this PPM.

Figure 5g illustrates a PPM whose left and top edges are passive. Even though the top edge only needs a point to reach the condition of a point and a constraint, the left edge cannot find two points from each neighboring edge to become activated. In the next case (Figure 5h), the left and top edges both have no constraint, and they are passive. The top edge already has a point, and it needs only a point from the neighboring edges to be activated. The left edge, however, has no point and needs two points, each one from a neighboring edge. As can be seen, the neighboring edges can give a point to this edge; thus, it can be activated as well. The last case shown in Figure 5i is similar to the previous case, while the left and top edges can only take a point from one of the neighboring edges. Therefore, model-fitting, in this case, is impossible as well.

While Equation 3 is capable of model-fitting and handling occlusion to a large extent, it cannot ensure the equal contribution of edges to the optimization process. The current definition of the objective function is based on the distribution of points across the edges of the PPM. This distribution might vary from a slight bias to a severe imbalance where some edges have one point, and others have hundreds of points. This results in a lack of sensitivity to the movement of edges with a lower number of points. To address this challenge, the weighted summation of errors resulting from each edge is calculated.

Considering a point cloud with n points and a PPM with a number of k edges whose edge j possesses t_j points, the edge weight ω_j can be calculated as follows:

$$\forall j : 1 \leq j \leq k \quad \rightarrow \quad \alpha_j = \frac{t_j}{n} \quad \& \quad \omega_j = \frac{1}{\alpha_j + \beta}, \tag{4}$$

where $1 \leq t_j \leq n$, $\sum_{j=1}^{k} t_j = n$, and β is a constant value (0.02) preventing a zero denominator.

The weighted fitness function of the problem can also be rewritten, and an optimization/minimization problem is defined for fitting a PPM into a point cloud as below:

$$\begin{gathered} \textit{To minimize:} \quad F(x_1, ..., x_m) = \frac{1}{k}\sum_{j=1}^{k} \omega_j (e_j + \lambda_j e'_j), \\ \textit{Subjected to:} \quad l_r \leq x_r \leq u_r \end{gathered} \tag{5}$$

After the initialization process, a list of candidates (population) is randomly generated from a PPM by a metaheuristic optimization algorithm. This list will be then improved by adjusting the initial value of parameters and minimizing the value of error resulting from Equation 5. As can be seen in Figure 6, this optimization process leads to a PPM that resembles the input point cloud, and its value of parameters is a close approximation of the values the point cloud represents.

The approach presented here provides an element-wise model-fitting, i.e., each PPM can extract the value of parameters from a single component (face/cross-section). In the next section, a global optimization problem is defined to assem-

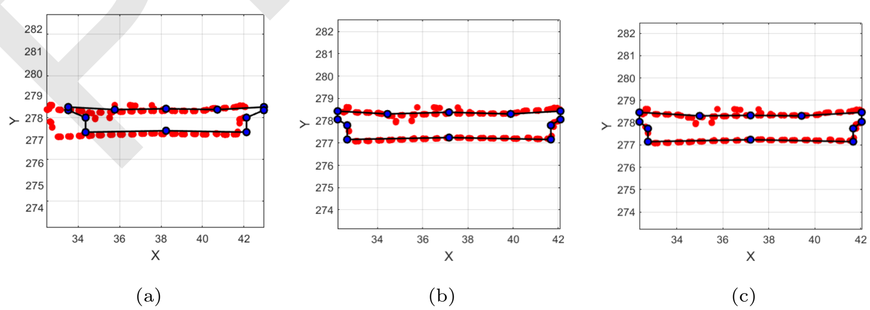


Figure 6: PPM of a typical bridge deck during the optimization process: (a) iteration 1; (b) iteration 20; (c) iteration 100.

ble and integrate all the pieces and create the parametric model of the entire bridge.

3.3. Parametric assembly

The model-fitting process through PPMs leads to a list of parameters representing the point cloud of elements. To create the parametric model of the entire bridge, these components must be assembled consistently, e.g., dimensions of shared edges and faces must be equal.

For this purpose, snapping algorithms have been generally proposed to connect and integrate pieces [58, 59]. These algorithms discover matches between polygons and search for adjacent vertices considering various conditions. The neighboring vertices are then replaced with a new vertex representing all the vertices. Snapping algorithms can be practical for model reconstruction and 3D representation of bridges. However, the model cannot stay parametric in those algorithms as the location of vertices is a function of parameters, and this function needs to meet a set of constraints. Furthermore, snapping algorithms generally follow a bottom-up approach, starting from vertices and edges, and mostly require setting problem-specific thresholds. To handle this challenge, a top-down approach is proposed, and a global optimization problem is defined to assemble the bridge components.

Figure 7 illustrates the point cloud of an abutment comprising two wing walls and a retaining wall. Following the proposed method in Section 3.2, a set of parameters can be obtained for each face/cross-section by solving element-wise optimization problems associated with the 2D PPMs. Herein, the value of parameters has been shown by x_{ij}, where i and j are indices devoted to the face and parameter number, respectively. In a parametric assembly problem, sets containing common parameters among components can be found that logically need to be represented by a single parameter. For instance, $A_2 = \{x_{13}, x_{24}, x_{33}\}$ is a set including the values of height resulting from the initial model-fitting process. Considering a top-down approach, the 3D PPM of an abutment can be created with a group of unique parameters, among which there is only a sin-

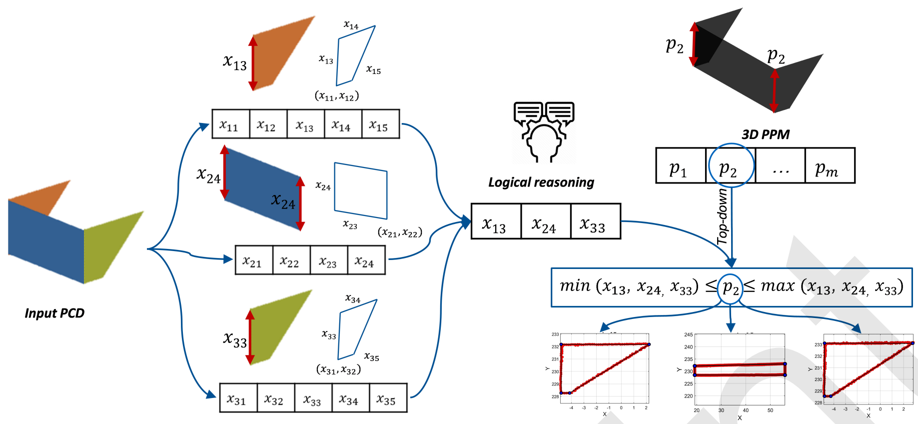


Figure 7: Assembly process of a typical abutment.

gle parameter, such as p_2 controlling the height of the abutment. To integrate PPMs, a representative value must be generated from the set A_2 and applied to the parameter p_2. Although averaging the set A_2 can provide a single representative value, it cannot lead to a permanent solution.

This results from the fact that a parametric model generally contains complicated dependencies and relations, and it is not apparent how the dependent parameters are affected by the average function. Considering the results of the initial element-wise model-fitting, each member of the set A_2 can be a proper candidate for the parameter p_2. The discrete set A_2 can be converted to a continuous interval by using the min and max functions, and each value in this range is considered a possible value for p_2 as well ($min(A_2) \leq p_2 \leq max(A_2)$).

Conversely, the value of the parameter p_2 should apply to the PPMs associated with the set A_2 and still retain them as close as possible to their corresponding point clouds. To satisfy these conditions, random values of the parameter p_2 can be generated in the interval resulting from the initial model-fitting, and their impact is evaluated on all the involved PPMs. In doing so, the value leading to the best fitting of all the PPMs can be approximated. This top-down method is only dependent on the proposed list of candidates for a parameter. This example can be extended and is expressed as an optimization problem for the parametric modeling of the entire bridge.

Let $\mathcal{X} = \{x_i | i = 1, ..., n\}$ be the set of all the possible parameter values resulting from fitting several PPMs into their corresponding point clouds. Following the reverse engineering (top-down) approach, assume $\mathcal{P} = \{p_j | j = 1, ..., m\}$ is also the target set of parameter values required to create the parametric model of the entire bridge. Considering the label of parameters, the initial set $\mathcal{X}$ can be divided into smaller sets of parameters that need to be assembled. Thus, a family of sets is obtained $\mathcal{A} = \{A_j | j = 1, ..., m\}$, where $A_j \subseteq \mathcal{X}$ and contains all the possible candidate values for the corresponding parameter p_j. The parametric assembly process of the number of h PPMs can be described as an optimization/minimization problem as follows:

$$\begin{aligned} \textit{To minimize:} \quad & G(p_1, ..., p_m) = \frac{1}{h} \sum_{v=1}^{h} \omega_v F_v, \\ \textit{Subjected to:} \quad & min(A_j) \leq p_j \leq max(A_j) \end{aligned} \tag{6}$$

where F_v is the fitness function described in Equation 5 and ω_v is the weight assigned to each PPM to balance the model-fitting errors. The value of ω_v can be calculated using Equation 4 based on the total number of points and the number of assigned points to each PPM.

This objective function receives a set of parameter values, randomly generated in ranges obtained by the initial element-wise model-fitting. It adjusts all the involved PPMs and fits them into the point cloud of the entire bridge.

3.4. Model generation

The proposed algorithms in the previous sections extract the value of parameters following a reverse engineering paradigm to achieve a 3D model satisfying the expected applications in practice. The 2D PPMs have also been set up to generate the final model after assembly. To deduct the design features of modeling the entire bridge, the 3D PPM can be created based on a set of parameters. End users can define these parameters following a level of detail (LoD) satisfying the anticipated applications from the model.

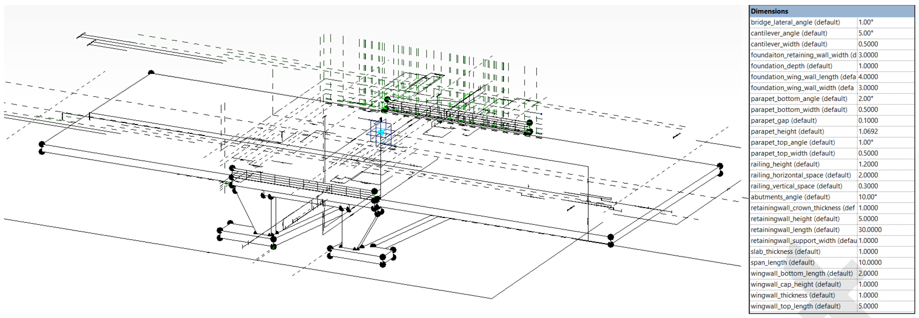


Figure 8: 3D PPM of a single-span concrete bridge created following reverse engineering.

This user-dependent definition of the model is highly close to the definition of a bridge DT as it is also created based on a set of desired use cases and requirements. Figure 8 demonstrates the 3D PPM of a single-span RC bridge created through a set of parameters to meet a desired LoD. This 3D PPM is completely parametric and dependent on the value of parameters. This model can be defined in most of the existing BIM-authoring tools. To create the model of the entire bridge from the bridge point cloud, the value of parameters extracted by the optimization algorithms after assembly can be imported into this model. As a result, a 3D PPM is generated that resembles the point cloud of the entire bridge.

4. Developed algorithms for processing bridge point clouds

Various algorithms are required to process segmented bridge point clouds and prepare them for applying PPMs. This section introduces these techniques and provides more details about them. In the next section, the application of each part is shown in the geometric digital twinning of bridge point clouds.

4.1. Clustering and de-noising

Multiple instances generally exist in the segmented point cloud of classes such as railings and abutments. To enable piece-wise model-fitting, the point cloud of these classes needs to be further clustered and de-noised. Density Based Spatial Clustering of Applications with Noise (DBSCAN) [60] is an automatic clustering algorithm proposed for discovering clusters in large spatial databases. This algorithm starts from a random point and expands the region based on the local density of data points. DBSCAN can be used to cluster and refine points in bridges [61].

However, setting a threshold value for density in bridges is challenging, especially in bridge point clouds with different resolutions. Also, it is computationally expensive and slow to process large datasets, which is common in bridges. To address these issues, a modified version of DBSCAN is proposed to cluster and de-noise segmented point clouds of bridges. As shown in Algorithm 1, this clustering method starts from a random query point and expands the region based on the connectivity of points. To reduce the complexity order of DBSCAN from $O(n2)$ to $O(k\ log(n))$, kd-tree is used as a data structure, and the neighboring points are obtained by KNN search. Any neighbor of the query

Algorithm 1 Clustering & de-noising algorithm

Input pc: point cloud; n: number of clusters (1 for the de-noising task); r: radius; k: number of neighbors; $label$: points label, initially undefined; KNN: K-nearest neighbors search; $Dist$: function to calculate Manhattan distance

```
foreach p ∈ pc do
    if label(p) undefined then
        next cluster label ← c
        label(p) ← c
        Neighbors N ← KNN(K, pc)
        Neighbors of the query point Q ← N/{p}
        foreach q ∈ Q do
            if label(q) undefined then
                Distance d ← Dist(q,p)
                if d < r then
                    label(q) ← c
                    Neighbors of the neighboring point S ← N/{q}
                    Q ← S ∪ Q
return label
```

point located within a predefined distance (radius) is added to the cluster of the query point, and its neighbors are also added to the list of the query point neighbors.

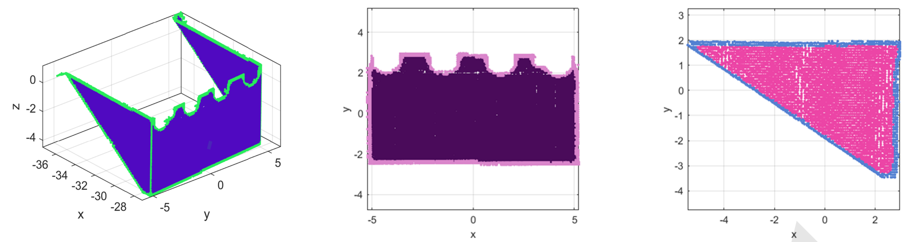
This process is repeated for any neighboring point in the list and continues until all the points are evaluated and assigned to a cluster. Similarly to DB-SCAN, this clustering method might result in many clusters in each of which the connectivity conditions have been satisfied. This algorithm is used in two applications: clustering and de-noising. In the clustering task, the largest n clusters are selected as the smaller clusters are more likely to represent noise clusters. In the de-noising task, the first largest cluster is only extracted as the points in this cluster satisfy the connectivity conditions and are far from the points belonging to other clusters.

4.2. Boundary points detection

A point cloud represents the external surfaces of objects in a scene. It also implicitly contains semantic and geometric information about the objects. Depending on the use case, a point cloud can be abstracted, simplified, and purposefully represented with a lower number of points. In a model-fitting process, boundary points mostly contain the geometric information of elements. Hence, the detection of these points seems necessary for fitting PPMs.

Boundary points generally have different features than interior points. Mean shift is one of those features proposed for detecting boundary points [62]. This point-level feature is expressed as each point's distance to its neighboring points' mean point. In general, boundary points show a higher shift value toward their mean point as they cannot find neighboring points all around their vicinity. To detect these points, a threshold has been defined in [62], which is based on the distance of the query point to its nearest neighbor. However, setting the value of this threshold is difficult, especially in point clouds with different resolutions.

To address this problem, a Fuzzy C-Means (FCM) algorithm is employed to automate the detection process of boundary points. FCM is an unsupervised clustering algorithm and an extension of the K-means algorithm in which the



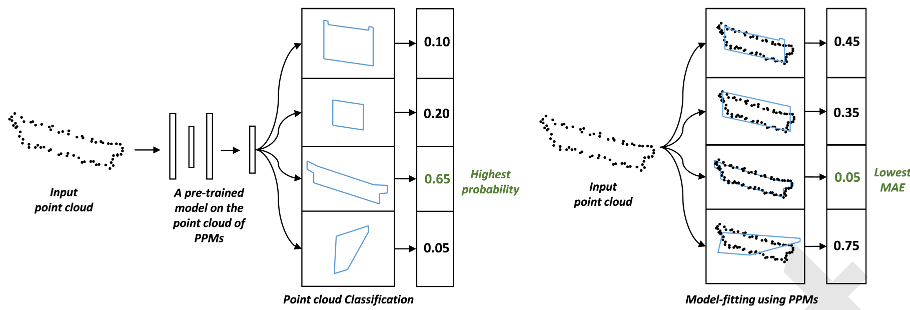
(a) (b) (c)

Figure 9: Boundary points detection by FCM clustering in 3D/2D: (a) an abutment; (b) a retaining wall; (c) a wing wall.

membership degree of data samples to clusters is expressed by fuzzy logic [63]. Considering the value of the mean shift, points can be divided into two clusters with sharp features (boundary points) and points with soft features (interior points). To detect boundary points, the nearest neighbors of each point are obtained by applying KD-tree and KNN search, and the value of the mean shift is computed. This feature is then passed through an FCM with two clusters. Since the value of the mean shift is higher for boundary points, the resulting cluster with the higher mean value is selected as boundary points. As a result, the proposed threshold can be eliminated, and the required points to fit PPMs are detected automatically, as shown in Figure 9.

4.3. Selection of PPMs

Given the point cloud of a bridge component (face/cross-section), a proper PPM needs to be selected to describe the input sample. For instance, the PPM of a bridge deck cannot be used for deriving the parameter values from an abutment point cloud as these elements are different in type. To address this problem, a library/catalog of bridge elements is created in which various types of PPMs exist. To select the appropriate PPM, the similarity of the input point cloud to all the PPMs is checked. For this purpose, two methods, called supervised and unsupervised selection, are proposed to determine the PPM required for model fitting. As shown in Figure 10, both of the methods



(a) (b)

Figure 10: Selection of PPMs based on the input point cloud: (a) supervised selection; (b) unsupervised selection.

are classifiers, however, with different levels of supervision.

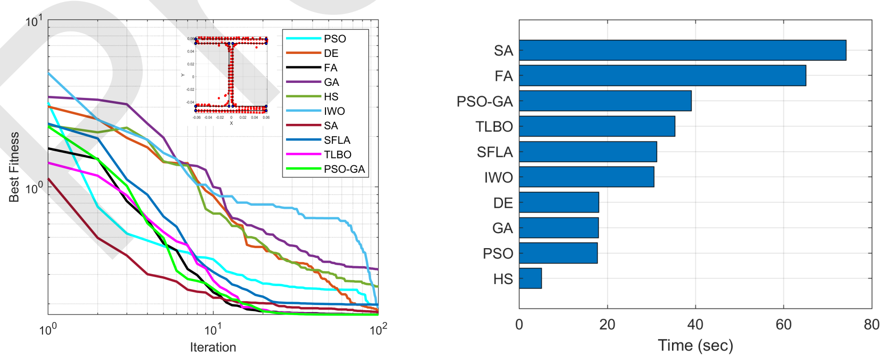
The supervised selection method requires a machine/deep learning model to be trained on the point cloud of the existing bridge elements in the catalog. There are many models in the literature that can be used as a point cloud classifier [64, 65, 66, 67]. The trained model can receive the point cloud of bridge elements and determine the type of PPM required for model fitting. This approach needs a large dataset of point clouds as well as an annotation process. However, the trained model can instantly select and call the appropriate PPM from the catalog.

The unsupervised selection method fits each existing PPM in the library/catalog to the input point clouds by solving a piece-wise optimization problem. Each model-fitting process leads to a value of model-fitting error describing the similarity of the input point cloud to the PPM. At the end of the process, the PPM with the lowest value of error is selected as it is more likely to represent the input point cloud. In comparison with the supervised selection, this method does not require a dataset for training and can directly classify the point cloud of bridge components. However, it requires more time to test each PPM on the input point cloud. The supervised and unsupervised selection methods can both be used interchangeably for the selection of PPMs.

4.4. Selection of the metaheuristic algorithm

Various metaheuristic algorithms can be used for fitting PPMs into point clouds. To evaluate the impact of the algorithms on the performance of the model, ten different metaheuristic algorithms, including Particle Swarm Optimization (PSO) [29], Genetic Algorithm (GA) [30], Harmony Search (HS) [68], Differential Evolution (DE) [69], Invasive Weed Optimization (IWO) [70], Shuffled Frog Leaping Algorithm (SFLA) [71], Teaching Learning Based Optimization (TLBO) [31], Firefly Algorithm (FA) [33], Simulated Annealing (SA) [72], and hybrid PSO-GA [73] are tested.

Each algorithm is run ten times to fit an I-shaped beam PPM into a point cloud, and the resulting mean convergence diagrams, as well as the average time required for model fitting, are presented. The hyperparameters of each algorithm have been tuned such that the best results are achieved for a specific number of iterations in a reasonable time interval. Figure 11a shows the obtained convergence diagrams from the metaheuristic algorithms in a logarithmic scale. As can be seen, three algorithms of PSO-GA, TLBO, and FA have been capable of gaining the lowest model-fitting errors, respectively. Figure 11b also illustrates the average required time for fitting the PPMs in which the HS algorithm has achieved the lowest modeling time.



(a) (b)

Figure 11: Comparing the performance of 10 different metaheuristic algorithms in a PPM-to-cloud fitting task: (a) Convergence diagram; (b) Convergence time.

Comparing the results of PSO-GA, TLBO, and FA in terms of time demonstrates the faster performance of TLBO in the model-fitting task. Among these three algorithms, TLBO only needs one hyperparameter (number of particles/population) and stopping criteria, while the two other algorithms have more problem-dependent hyperparameters for tuning. Therefore, TLBO can provide a higher level of automation with minimal user intervention. Considering the algorithm's stability in convergence, the required time for model-fitting, and the number of hyperparameters, TLBO is selected while all other algorithms can also be utilized.

5. Experiments with real-world data

This section employs the techniques introduced in the previous sections and evaluates the performance of the proposed method in creating the geometric model of six single-span concrete bridges as well as other components that generally exist in multi-span bridges.

5.1. Experiment 1: Geometric modeling of six single-span concrete bridges

The point cloud data of six single-span reinforced concrete (RC) highway bridges in Bavaria, Germany, is used for evaluation and model reconstruction. This dataset has been acquired through aerial photogrammetry by flying a drone around the structure and underneath the bridge deck to take photographs from various angles to meet a minimum 75% frontal and 60% side overlap. All the captured images have the same resolution of 5472 × 3078. This dataset has been processed by Agisoft based on Structure from Motion (SfM) to generate the point cloud of the structure. All the bridge samples have been subsampled by the uniform grid subsampling method with a grid size of 5 cm to decrease the processing load of the algorithms. This step led to point clouds with an average density of 252 points/m^2 and around 2 million points per sample. As shown in Figure 12, the samples comprise a bridge deck, abutments (retaining walls and wing walls), railings, and background.

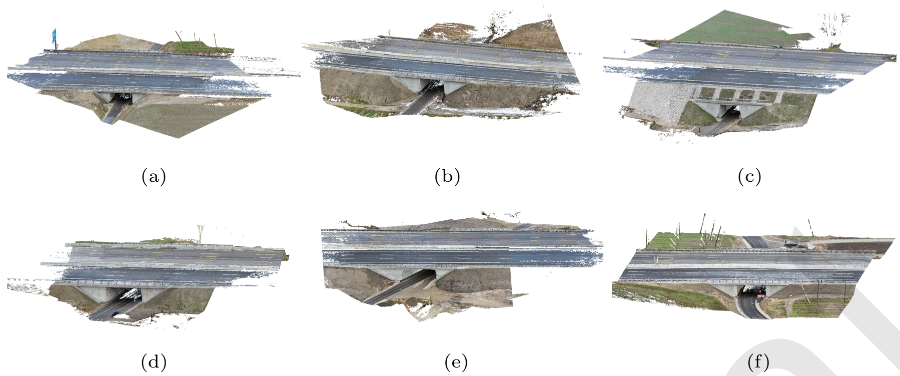


Figure 12: Photogrammetric PCD of six single-span RC bridges. (a-f) shows the bride sample 01-06 with a density of 246, 250, 254, 252, 251, and 257 points/m^2, respectively.

Bridge point clouds need to be prepared prior to applying the PPMs and deriving the value of parameters. Figure 13 depicts the required preprocessing steps, including semantic segmentation, transformation, instance segmentation (clustering), and face/cross-section detection.

Semantic segmentation is the initial step in enriching the input raw bridge point clouds, as shown in Figure 13a. This step separates the input point cloud into the point cloud of bridge elements such as abutments, bridge deck, and railings, as well as the background, by predicting a class label for each point. Semantic segmentation narrows down the initial problem from the entire bridge point cloud to the point cloud of bridge elements and determines the type of each component from which the type of the PPM can be recognized as well. This step has not been covered in the paper as its focus is on parametric modeling of bridges. However, there are various methods for semantic segmentation of point clouds, such as bottom-up [74, 75, 76], top-down [77, 78, 79], or deep learning-based [61, 80, 64]. All these research works, as well as the previous work by the authors of this paper [81], can be used.

The raw bridge point clouds are not generally along the x-axis and have some degrees of rotation around the z-axis. For bridge point clouds with a straight deck (without a large horizontal curvature), it is more suitable to ro-

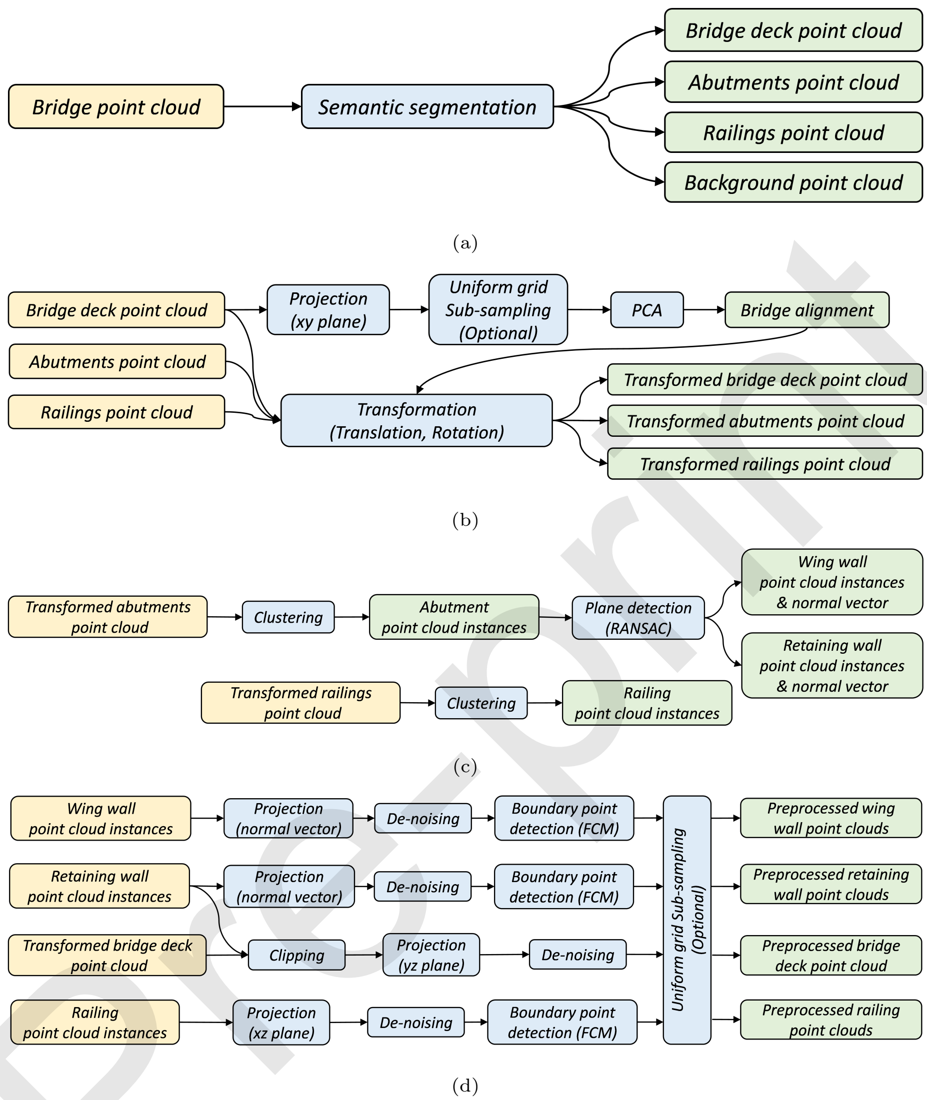


Figure 13: Required preprocessing steps for the proposed pipeline: (a) semantic segmentation; (b) transformation; (c) clustering (instance segmentation); (d) cross-section/face and boundary points detection.

tate the point cloud around the z-axis and make it along the x-axis. Thus, transformation (translation and rotation) of the segmented point clouds is the next preprocessing step, as shown in Figure 13b. As the variance of points along the length of the bridge deck is significantly higher than in the other directions, principal component analysis (PCA) is employed to detect the alignment of the bridge. To this end, the point cloud of the bridge deck is projected onto the xy plane, and a uniform grid subsampling is applied to remove the impact of overlying points resulting from the projection. Then, PCA is executed, and the segmented point clouds are translated and rotated around the z-axis as much as the angle between the principal component obtained by PCA and the x-axis.

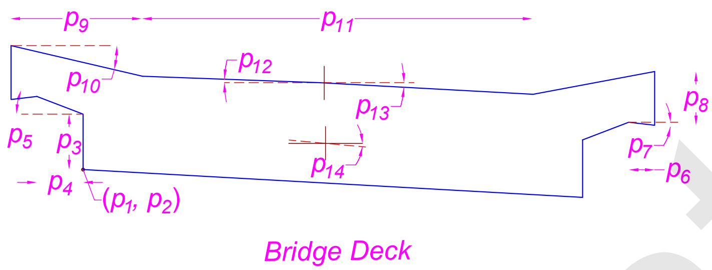
There is generally more than one point cloud instance in the classes of abutments and railings. Also, abutments consist of sub-elements, including a retaining wall and two wing walls. Therefore, these classes need to be further segmented/clustered, as shown in Figure 13c. To this end, the clustering algorithm described in Section 4.1 is employed to detect the two instances in each class. As mentioned, this algorithm clusters the point cloud instances following the connectivity rules. As the point cloud instances, such as abutments and railings, generally stand far from each other, a connectivity radius of $r = 1$ m is considered for the instance segmentation. In order to detect the point cloud of the retaining wall and the wing walls, the RANSAC algorithm is employed. As the number of existing faces in each abutment point cloud is known (two wing walls and a retaining wall), the number of existing thresholds in RANSAC is reduced and limited to only a distance threshold from the planes that can be reasonably selected (herein 10 cm), over a number of iterations (herein 300) for each plane instance.

The last remaining step is the detection of cross-section or boundary points of faces, which are required for fitting PPMs. For this purpose, a combination of projection, de-noising, FCM clustering, and subsampling functions/methods is employed, as shown in Figure 13d. The point cloud of wing walls and retaining walls is projected onto 2D planes using their normal vectors detected by the RANSAC algorithm in the previous step. The boundary points are then detected by the FCM clustering algorithm proposed in Section 4.2. As the bridge deck point cloud is the part between the retaining walls, it is clipped and projected onto the yz plane. The railing point clouds are also projected onto the xz plane, and their boundary points are detected using the FCM clustering algorithm. All these point clouds are de-noised after projection, as described in Section 4.1, and subsampled by uniform grid subsampling (grid size $\simeq$ 5 to 20 cm). In all the steps, the subsampling module is optional and can be eliminated. This module has been only used to decrease the processing loads of the algorithms and remove the impact of overlying points due to the projection. The de-noising module also checks the connectivity rules to ensure no point exists far from the target points.

To derive the value of parameters, the corresponding PPM to each preprocessed point cluster is selected. As shown in Figure 13, the semantic segmentation and the clustering modules generally determine the type of the required PPMs for model fitting. However, in scenarios where the type of PPMs is not known, the supervised and unsupervised selection methods (Section 4.3) can be employed.

Figure 14 shows the details of PPMs used for model fitting all the bridge samples. These PPMs include a bridge deck, wing wall, retaining wall, and railing obtained by analyzing the bridge point cloud samples to reach a desired LoD. All the PPMs have been initialized only once and used for the geometric digital twinning of all the bridge samples, i.e., no user intervention is applied to the PPMs from sample to sample. Most of the parameter intervals have been obtained by analyzing a large number of bridge data provided by the German bridge database "SIB-Bauwerke" as well as empirical knowledge. For parameters such as the origin or width of the bridge deck that might largely vary in bridges, the axis-aligned bounding box (AABB) of the point clouds, with the lower left corner (ll) and upper right corner (ur), has been used to relatively set the initial values. All these intervals have also been shown in Figure 14.

To adjust the instantiated PPMs and fit them into their corresponding point clouds, TLBO is employed as it showed promising performance in Section 4.4. This algorithm only needs a number of population/particles (75 particles) and a stopping criteria (300 iterations). Piece-wise optimization problems are solved by TLBO for each point cluster representing a bridge element. Each optimization process starts with a list of candidates randomly generated by the opti-



p_1: x-origin $\in [x_{ll}, x_{ur}]$

p_2: y-origin $\in [y_{ll}, y_{ur}]$

p_3: deck depth $\in [0.05, 1.30]$

p_4: cantilever width $\in [0.10, 1.80]$

p_5: cantilever slope $\in [5°, 75°]$

p_6: parapet bottom width $\in [0.05, 1.00]$

p_7: parapet bottom slope $\in [-45°, 0°]$

p_8: parapet height $\in [0.05, 1.20]$

p_9: parapet top width $\in [0.50, 3.50]$

p_{10}: parapet top slope $\in [-45°, 0°]$

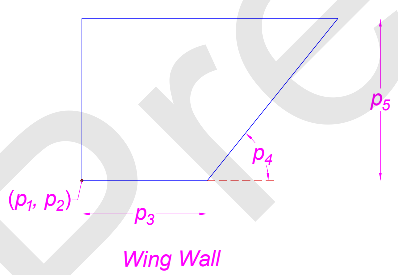
p_{11}: deck width $\in [4.00, x_{ur} - x_{ll}]$

p_{12}: deck right slope $\in [-5°, 5°]$

p_{13}: deck left slope $\in [-5°, 5°]$

p_{14}: deck inclination $\in [-10°, 10°]$

(a)



p_1: x-origin $\in [x_{ll}, x_{ur}]$

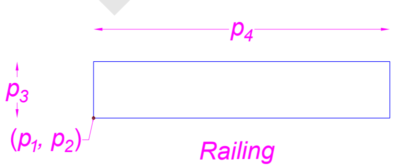
p_2: y-origin $\in [y_{ll}, y_{ur}]$

p_3: width $\in [0.30, x_{ur} - x_{ll}]$

p_4: slope $\in [30°, 80°]$

p_5: height $\in [2.00, y_{ur} - y_{ll}]$

(b)



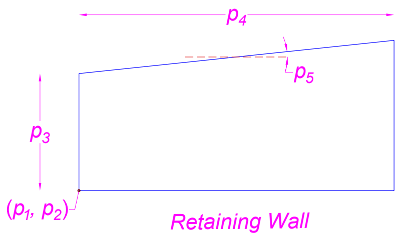
p_1: x-origin $\in [x_{ll}, x_{ur}]$

p_2: y-origin $\in [y_{ll}, y_{ur}]$

p_3: height $\in [0.50, 1.80]$

p_4: length $\in [0.80 \times (x_{ur} - x_{ll}), x_{ur} - x_{ll}]$

(c)



p_1: x-origin $\in [x_{ll},\, x_{ur}]$

p_2: y-origin $\in [y_{ll},\, y_{ur}]$

p_3: height $\in [2.00,\, y_{ur} - y_{ll}]$

p_4: length
$\in [0.80 \times (x_{ur} - x_{ll}),\, x_{ur} - x_{ll}]$

p_5: slope $\in [-10°,\, 10°]$

(d)

Figure 14: List of PPMs used for geometric digital twinning of the bridge point clouds: (a) bridge deck; (b) wing wall; (c) railing; (d) retaining wall. x_{ur}, y_{ur} and x_{ll}, y_{ll} are the x- and $y-$ coordinate of the upper right and lower left corner of the axis-aligned bounding box (AABB) surrounding the input point cloud. $x_{ur} - x_{ll}$ and $y_{ur} - y_{ll}$ are also the length and the height of the AABB, respectively. The dimensions of the AABB are used for the relative initialization of PPMs. All values are in meter (m).

mization algorithm. This list is further refined by TLBO such that the PPMs can be fitted into the point cloud. This process leads to a close approximation of the parameter values after optimization. Considering the number of four wing walls, two retaining walls, two railings, and a bridge deck that exists in each bridge point cloud, the optimization algorithm must be capable of extracting the value of 52 parameters. To evaluate the accuracy of the resulting models, the mean absolute error (MAE) of PPMs is calculated by Equation 1 that shows the distance of points to PPMs.

Table 1 illustrates the MAE of PPMs after the model fitting process. Averaging the resulting values of error from the bridge samples shows that TLBO has been capable of modeling bridges with an MAE of 8.71 cm. Note that noises and other imperfections in the entire pipeline have been considered in the calculation of MAE. Therefore, these error values show the worst case in which some noises or wrongly classified points still exist in the problem space. In addition, no external intervention has been made in the modeling process of bridges from the point clouds. This table also demonstrates a class-wise comparison of each element's error value. As can be seen, the class Retaining Wall (RW) has re-

Table 1: MAE of the fitted PPMs into the point cluster of bridge elements (cm).

Sample	Wing Wall (WW)				Retaining Wall (RW)		Railing (R)		Deck	Mean
	WW1	WW2	WW3	WW4	RW1	RW2	R1	R2		
Bridge 01	1.87	2.65	2.26	2.45	11.73	9.76	6.00	5.65	13.00	6.15
Bridge 02	6.67	5.48	8.08	5.75	35.26	37.06	7.89	9.91	12.97	14.34
Bridge 03	6.30	6.71	6.63	6.19	12.82	15.19	15.71	17.86	7.61	10.56
Bridge 04	3.28	3.28	4.83	4.83	4.74	9.65	17.84	17.85	8.05	8.26
Bridge 05	2.94	2.94	3.15	2.88	7.03	7.54	16.09	7.99	15.51	7.34
Bridge 06	2.82	2.37	2.39	3.07	9.05	4.25	10.32	9.95	5.98	5.58
Mean	4.16				13.67		11.92		10.52	8.71

sulted in the highest value of MAE. Figure 15 shows the fitted model to the retaining wall of Bridge 02 after optimization. As can be seen, the bottom and vertical edges of the PPM have horizontal and vertical constraints, thus preventing them from rotating and becoming closer to the points. This instance shows that the governing reverse engineering approach and the injected bridge engineering knowledge enforce the algorithm to generate PPMs that necessarily end up with the anticipated 3D model. Although the rotational degrees of freedom can be given to such edges, the 3D model must also be capable of accepting these new parameters as the process has been started from the final model. In this example, our presumption has been to generate a bridge model whose retaining walls have constraints on the bottom and lateral edges.

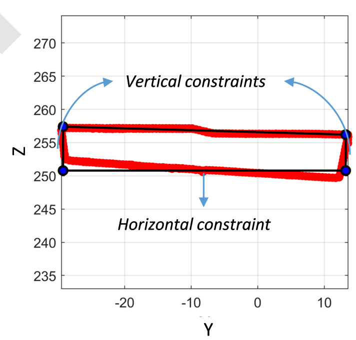


Figure 15: Fitted retaining wall of Bridge 02 by the PPM.

To generate the 3D model of the entire bridge, all the extracted parameters have been assembled by solving another optimization problem as described in Section 3.3. In this process, all the involved PPMs are refined again so that the common parameters among elements are integrated, and the PPMs still remain as close as possible to their corresponding point cloud.

Figure 16 depicts the histogram of each bridge sample after the geometric modeling process. The vertical axis of the diagram shows the number of points assigned to all the involved PPMs, and the horizontal axis shows the distance of points to the PPMs in terms of MAE. As can be seen, a large portion of points has a distance of less than 5 cm from the fitted PPMs in all the samples. However, in samples Bridge 02 (Figure 16b) and Bridge 03 (Figure 16c), the variation range of MAE is larger than a sample such as Bridge 06 (Figure 16f). This observation is also compatible with Table 1 in which the value of MAE is higher in Bridges 02 and 03. Comparing the point cloud of these two samples with

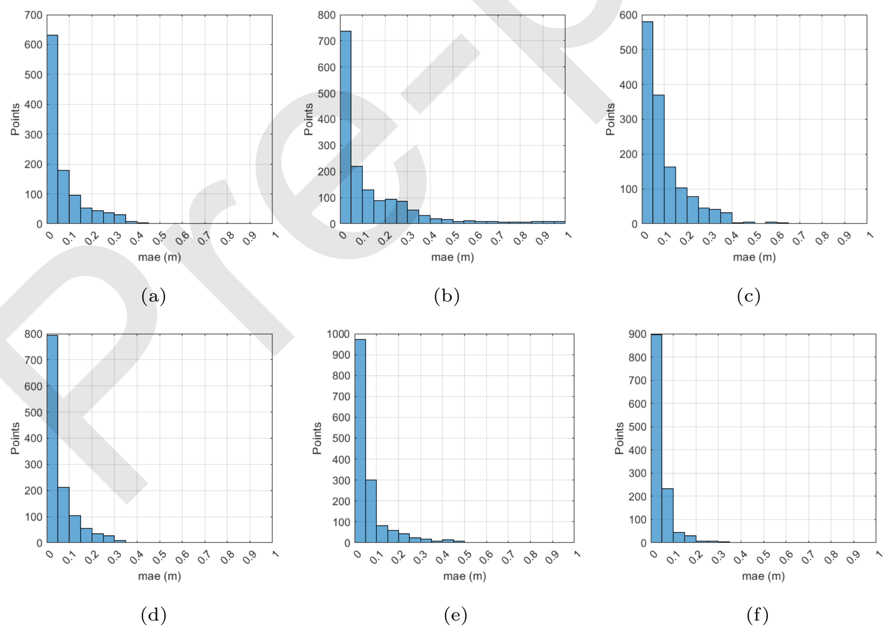


Figure 16: Histogram of fitted bridges into the point clouds. a-f show the bridge samples 01-06, respectively.

other samples shows that Bridges 02 and 03 have more differences in type with respect to the desired model. Hence, the generated PPMs from the model at the beginning of the process have not been able to completely describe/capture differences beyond the imposed presumptions/restrictions. This means the four PPMs used for modeling all the bridge components of the six samples have been more compatible in type (not dimensions; the same setup/initialization has been used for all the samples) with the point cloud of samples 01 and 04-06. As a result, in Bridges 03 and 04, the edges and vertices of the PPMs have not been able to move closer to the bridge point cloud, which, in turn, has led to a higher value of error.

Table 2 shows the overall time required for preprocessing segmented point clouds, extracting the value of parameters, and assembling them into an integrated model. As can be seen, the modeling time of all the samples with around 2 million points is less than 370 sec (6.16 min). This shows the massive reduction of the modeling time in comparison with the manual modeling processes, which usually take several days. To visualize the 3D model of each bridge sample, the parameter values are imported into the 3D PPM of the bridge according to Section 3.4. This process leads to the 3D geometric model of each bridge sample as shown in Figure 17.

Table 2: Required time for modeling bridges from point clouds.

Sample	Bridge 01	Bridge 02	Bridge 03	Bridge 04	Bridge 05	Bridge 06
Time (sec)	285.41	367.07	328.16	311.76	350.18	305.01

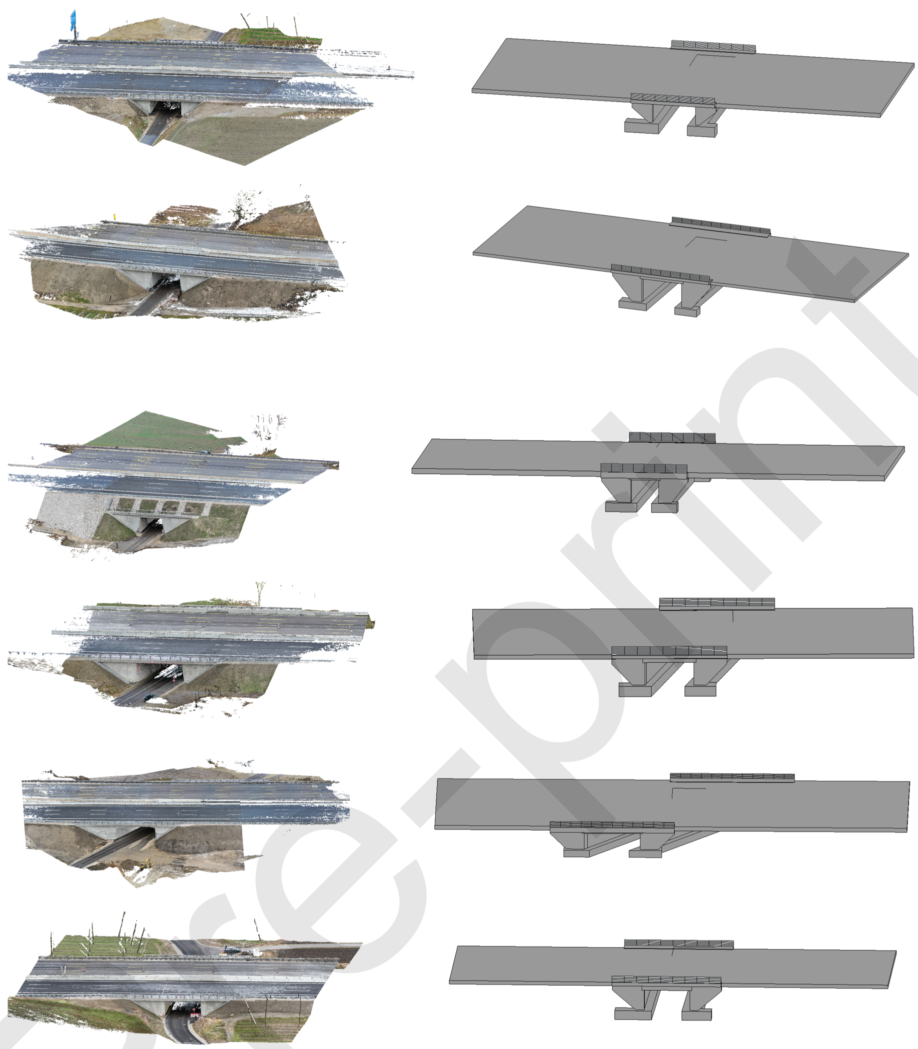


Figure 17: Point cloud of bridges and their corresponding fitted geometric model. Each row shows a bridge sample (01-06).

5.2. *Experiment 2: Bridge elements*

Contrary to single-span bridges, multi-span bridges have a longer deck supported by piers. The deck of multi-span bridges generally has vertical and horizontal curvatures and cannot be described properly by a single extrude function.

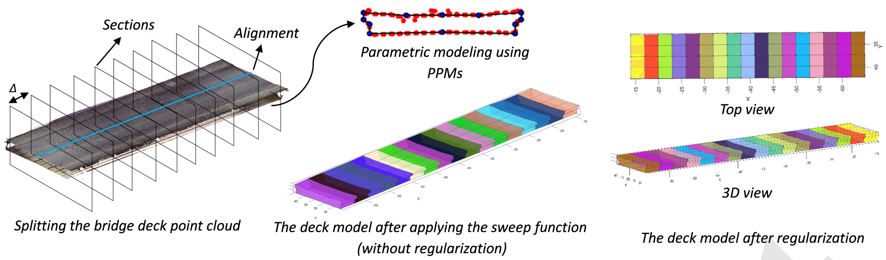
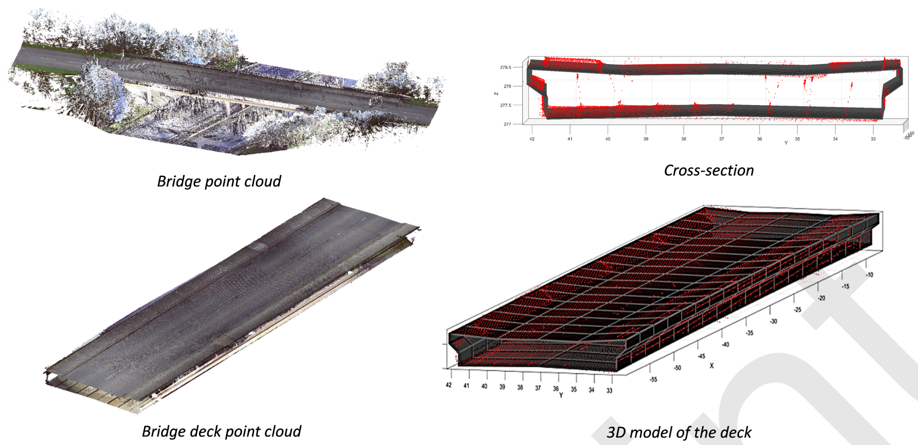


Figure 18: Modeling process of a typical multi-span bridge deck.

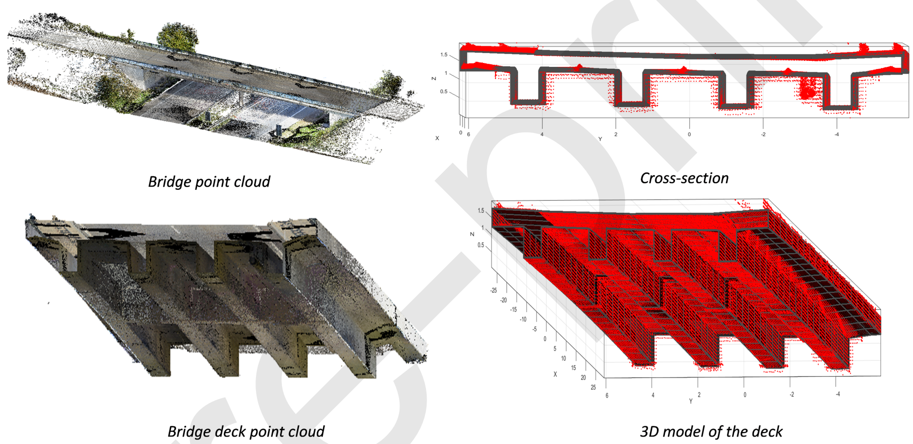
Figure 18 shows the process of modeling the deck of a typical multi-span bridge. To capture the curvature and changes of such bridges, the alignment of the deck needs to be detected in the initial step. The alignment of straight bridges/decks can be recognized using the PCA algorithm similar to Experiment 1. In the case of bridges with horizontal curvature, a polynomial can be fitted to the deck point cloud after projection onto the xy plane. Using the bridge alignment, the deck point cloud can be split into smaller segments, each of which is placed between two planes/sections in a pre-defined distance (Δ) (see Figure 18). These segmented point clouds can be projected onto a 2D plane and fitted using a PPM by solving multiple piece-wise optimization problems. Note that a single PPM with the same initialization is used for fitting all the slices of the bridge deck point cloud. This model-fitting process leads to a list of parameters obtained from each slice. Sweeping/connecting all the PPMs along the length of the bridge deck results in the 3D model of the deck. However, this model might not be smoothed as the extracted values from PPMs have differences along the length of the bridge deck (Figure 18). To address this issue, the values of each parameter are regularized separately in three steps. First, assuming a normal distribution for parameter values, the outliers are removed by calculating the mean value (μ) of the parameter and its standard deviation (σ). Second, a polynomial is fitted to the values. Third, the value of the parameter is read from the fitted polynomial using its location.

To clarify, assume a deck point cloud with a starting point at $x = 0$ and an endpoint at $x = 3$. Considering four sections at locations such as $x = \{0, 1, 2, 3\}$, three segments of the point cloud can be obtained and fitted by a PPM. Let $p = \{2.40, 2.50, 2.60\}$ be the extracted values for a parameter such as the parapet width by fitting the three sequential PPMs. After removing outliers from the set p, for ex., values more and less than $\mu \pm \sigma$ (68% of data), a polynomial, such as $ax^2 + bx + c$, can be fitted to the values of the set p. Using this polynomial whose coefficients are known after fitting, regularized values of the parameter can be extracted by inserting the values of the set x into the fitted polynomial (Figure 18).

Figure 19a shows the results of applying PPMs in the parametric modeling of two multi-span bridges. The first bridge sample is Bridge 01 from the Cambridge bridge point cloud dataset [77], which shows a concrete bridge point cloud acquired by laser scanning. As the bridge deck is straight, PCA can be applied to this sample similarly to the single-span bridges. To generate the geometric model, the bridge deck is split into intervals of 2 m, and a PPM is fitted to each segment of the point cloud. The PPM of this sample is similar to the PPM used for the deck of single-span bridges. After extracting the value of parameters, outliers are removed, and a polynomial of degree two is fitted. As can be seen in Figure 19a, the model has been fitted into the point cloud completely. Calculating the distance of points to the PPMs along the length of the bridge deck shows an MAE of 1.67 cm/m, while noises have also been considered in calculating the value of error; thus, the computed value shows the worst case. Figure 19b also demonstrates the application of PPMs in the geometric modeling of another multi-span bridge captured in Munich, Germany. Contrary to Experiment 1, this sample has been acquired through laser scanning. In comparison with the previous bridge sample, this bridge deck is more complicated in shape as it has four T-shaped concrete girders. To select the appropriate type of PPM for describing the point cloud sample, the unsupervised selection method proposed in Section 4.3 has been used, and T-shaped bridge decks with 3-6 girders are tested. As the value of MAE resulting from PPM with four girders has been lower, this type of PPM is selected. This PPM



(a)



(b)

Figure 19: Parametric modeling of two bridge decks from their point clouds: (a) first sample; (b) second sample.

can also be initialized similarly to the deck of single-span bridges. The only difference is the existence of girders whose dimensions can be logically set up based on bridge engineering knowledge. In this example, a width of [0.1, 1.5] and a depth of [0.2, 1.5] have been considered for the girders. Also, they have a distance of [0.5, 3] with respect to each other, half of which belongs to the flange of a girder. The flange can also have a value of slope in the range of [-3°, 3°]. This bridge deck point cloud has a length of around 80 m, and it has been sliced every 2.5 m. This means $80/2.5 = 32$ distinct optimization problems that need to be solved to model this bridge deck. As can be seen in Figure 19b, a single PPM has been capable of deriving the parameter values from all the slices and generating the 3D model of the deck. Averaging the values of MAE over the length of this bridge shows a value of 1.05 cm/m while noises still exist in the problem.

Various types of piers can be seen in bridges. A common type of bridge pier is shown in Figure 20, which consists of a pier cap and two pier columns. This pier can be modeled by PPMs if the pier cap is separated from the pier column. To this end, an FCM clustering algorithm is used for two clusters (pier cap and pier column). As the feature vector of the FCM, three features are calculated that represent differences between these two elements. First, the pier cap is generally over the pier columns; thus, its points have higher values of z-coordinate. Second, the pier cap is a horizontal element while the pier column is vertical; therefore, the z-component of the points' normal vector is higher for the pier cap. Third, if the pier is projected onto the xy plane, the 2D density of the points belonging to the pier column is higher as it is a vertical element. The 2D density can be calculated by counting the number of neighboring points placed within a circle with a predefined radius. Using these three features, the point cloud can be segmented.

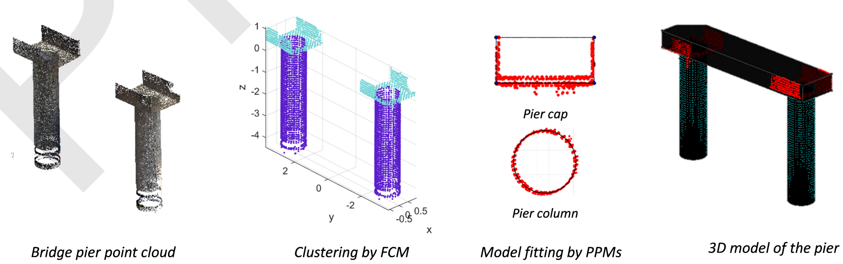


Figure 20: Modeling process of a typical bridge pier.

To extract the value of parameters from the pier column, the points of each pier are projected, and a circular PPM is fitted. Note that a circle is a primitive shape; however, it can still be represented with three parameters, a center (x, y) and a radius (r), and its distance to the points is minimized by a metaheuristic algorithm. The pier cap can also be modeled by a rectangular PPM. As can be seen in Figure 20, the pier cap has occlusion and some noise; however, the PPM can still perform properly. After extracting the value of parameters, the elements can be assembled, and the 3D model of the pier is obtained. Averaging the value of MAE from fitting the pier columns and the pier cap shows an MAE of 1.43 cm.

6. Discussion

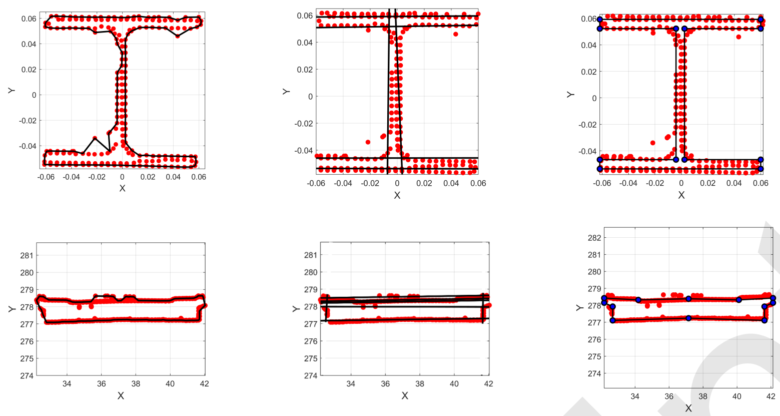
The performance of the proposed method can be evaluated in various scenarios and compared with other existing algorithms. This section further discusses the model fitting process by PPMs and highlights its advantages in geometric modeling.

6.1. Comparison with other methods

For comparison, the point clouds of an I-shaped beam and a bridge deck are fitted by PPMs, α-Concave hull [82], and RANSAC algorithm [82]. Figure 21 visually compares these methods after applying each algorithm.

As can be seen, PPMs have been more successful in model-fitting, thanks to the reverse engineering strategy governing the optimization algorithm. The other two methods cannot provide an exact number of parameters after model fitting and require another heuristic algorithm to refine their results. Therefore, these methods cannot directly provide a meaningful parametric model without any post-processing step. The proposed algorithm, however, results in a finite number of parameters with a close approximation of their values. It also preserves constraints such as orthogonality, parallelism, and symmetry in model fitting to meet the anticipated requirements.

Table 3 compares the proposed method with the most recent methods [56, 55, 43] in the geometric modeling of bridges or structural elements. Each column



(a) (b) (c)

Figure 21: Comparison the results of the model fitting approaches: (a) α-Concave hull [82]; (b) RANSAC [47]; (c) PPM (ours).

of the table represents a feature that can be a basis for comparison. The second column demonstrates the bridge components addressed by the methods. As can be seen, the two first methods are limited in covering all the components that generally exist in bridges and have mainly focused on steel profiles/sections, while the third method, in addition to steel girders, covers piers and the bridge deck. The third column in the table shows the core model-fitting algorithm used to model the geometry from point clouds.

The first method utilizes the RANSAC algorithm for estimating the dimensions of steel profiles, while the second method uses a kernel density estimation (KDE) algorithm to detect the type of cross-section from a catalog. The third method also employs α-concave hull for more complicated geometries, such as the bridge deck, and a density estimation algorithm to detect the type of girders from a catalog. The column named "Modeling Level" shows the coverage level of the proposed approaches. The two first methods have been limited to modeling bridge components, while the next two methods have generated the entire model of the bridge. The Assembly column demonstrates whether the assembly process of elements has been described or not. As can be seen, none of

Table 3: Comparison of the proposed method with three state-of-the-art methods.

Method	Covered Elements	Core Algorithm	Modeling Level	Assembly	Accessibiliy to Dimensions	Parametric Modeling
1. Yan and Hajjar [56]	super-structure components: I-shaped girders and cross-frames	RANSAC	Components	No	Yes (only steel sections)	No
2. Laefer and Truong-Hong [55]	Steel sections: I-, L-, T- and C-shape sections	KDE	Components	No	Yes (only steel sections)	No
3. Lu and Brilakis [43]	super- and sub-structure components: bridge girders, piers, and decks	α-concave hull & density estimation	Entire bridge	No	Yes (only circular pier columns and steel sections)	No
4. Ours	super- and sub-structure components: retaining walls, wing walls, parapets, bridge girders, piers, railings, and decks	PPM	Entire bridge	Yes	Yes (all elements)	Yes

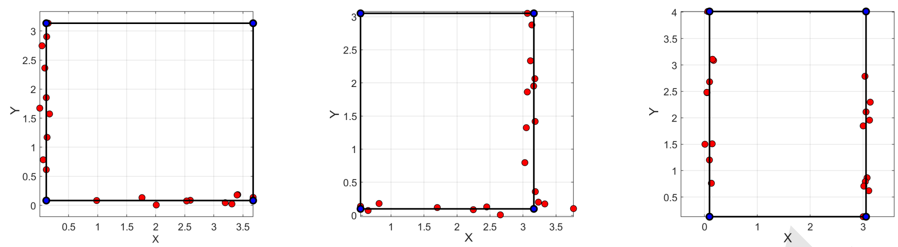
the other methods have addressed this problem. The next column (Accessibility to Dimensions) represents whether the value of parameters/dimensions has been extracted from point clouds. The first two methods have been capable of obtaining the value of parameters. However, these methods have only covered steel sections such as girders or cross-frames. The third method has also been limited and only extracted the value of parameters for circular pier columns and steel girders. This method uses the α-concave hull for describing the more complicated geometries, and as discussed in Figure 21, this algorithm cannot solely result in the parameter values. The last column also shows whether the resulting model is parametric and can accept geometric updates. As can be seen, none of the other methods have included this feature in the geometric modeling.

6.2. Occlusion resistant model-fitting

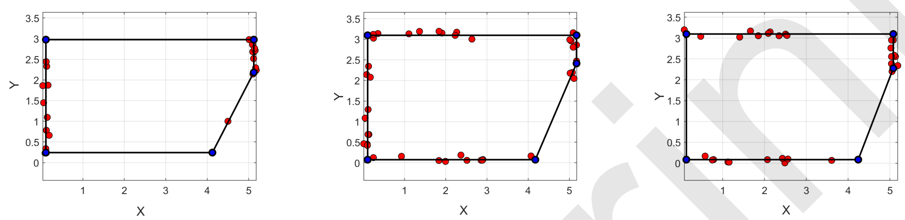
The proposed concept of *active* and *passive* edges can improve the algorithm's performance in fitting PPMs into occluded point clouds. This new fitness function definition can generate results at a competitive level with human recognition in modeling. Figure 22 shows the results of the model fitting a rectangular and a trapezoidal PPM into the occluded point clouds. In some cases, the edges of the PPMs cannot find any point in their vicinity. Nonetheless, these edges can still be fitted into the point clouds. Note that a simple fitness function definition such as Equation 1 cannot provide meaningful results in these cases as the optimization algorithm cannot realize the correct placement of the passive edges.

6.3. Editability of the resulting model

One of the advantages of the proposed approach is the editability of the resulting model, which is required to enable design work in the frame of rehabilitation or modification measures. This feature enables users to modify each element by adjusting the value of parameters as shown in Figure 23. Note that a



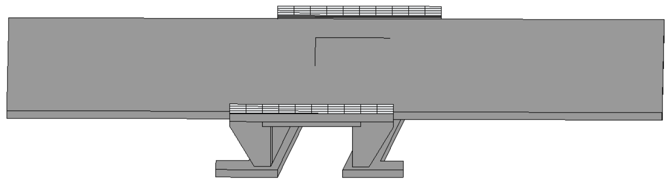
(a)



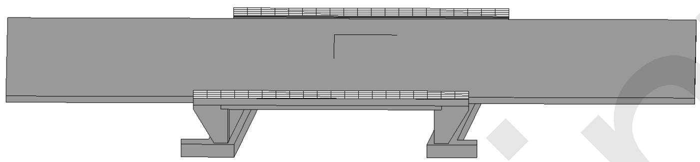
(b)

Figure 22: Performance of the algorithm in sustaining a large amount of occlusion: (a) a rectangular PPM; (b) a trapezoidal PPM.

point cloud only represents the object's outer shell. For instance, it cannot provide any information about the foundation of abutments or the inner thickness of the bridge deck. Therefore, external resources are still required from which the related parameters can be extracted and imported into the model. The resulting model of the defined pipeline preserves all the existing relationships and dependencies between elements, thanks to its parametric design. Also, all the existing parameters can be adjusted or unchanged during optimization. For example, a default value for the depth of the foundation can be assumed and remained unchanged throughout the optimization process. After optimization, this parameter can be read or extracted from structural drawings and imported into the model separately. The resulting model can also be connected to various algorithms for further enrichment. This is highly compatible with the definition of geometric DTs, which need to stay connected to the actual asset for handling bidirectional updates.



(a)



(b)

Figure 23: Editability of the model: (a) the resulting model; (b) edited model with a new span length and foundation depth.

7. Conclusions

This paper presents an automated approach to creating the geometric digital twin (DT) of concrete bridges from segmented point clouds. Parametric prototype models (PPMs) have been introduced as tools to extract the value of parameters from point clouds. The PPM of bridge elements can be created following a reverse engineering approach and the desired model to achieve from the bridge point cloud. PPMs can be instantiated with random values and fitted into the point cloud of elements through an optimization problem solvable by metaheuristic algorithms.

To improve the model-fitting accuracy and enable the model to handle a large amount of occlusion, new fitness/cost functions have been introduced. Each PPM after optimization/adjustment results in a list of parameters whose values show the parameter values the point cloud represents. To generate the parametric model of the entire bridge, the resulting PPMs from the piece-wise model-fitting problems need to be assembled. A global optimization problem has been defined to integrate PPMs and generate a list of parameters compatible with the anticipated model. Following the reverse engineering approach, these parameter values are finally injected into the 3D PPM of the bridge to create the geometric digital twin model reflecting the input point cloud.

The results of testing the proposed pipeline on the point cloud of six single-span concrete bridges as well as bridge components show that the proposed approach can generate the model of bridges with a mean absolute error (MAE) of 8.71 cm. The resulting model from this method still stays connected to the actual asset in reality through an access point and is still editable for any further refinement or enrichment.

Considering the results of the paper, the digital twinning process of the existing bridges can be automated to a large extent. However, the proposed algorithm is still limited in covering more complicated bridge types, even though most of the algorithms in the paper are extendable to such bridges.

Apart from that, the proposed pipeline requires bridge engineering knowledge and statistical study for setting the range of parameters that might make the algorithm limited in modeling highly complicated/arbitrary shapes that are too diverse in shape. Sing-span bridges form a large portion of the existing bridges in Germany (more than 50 %, according to a database received from the Federal Highway Research Institute). The ultimate goal is to have a library of parametric bridge models, including single-span, multi-span, etc., and a classifier to select the desired model from the library in the initial step. This can be achieved using a deep learning model that receives the input point cloud and calls the desired model from the library. However, this remains future research. In future works, the proposed pipeline will be tested on more samples of bridges, and the resulting models will be connected to other resources, such as technical drawings, for further geometric-semantic enrichment.

Acknowledgement

The research presented has been performed in the frame of the TwinGen project funded by the German Ministry for Digital and Transport (BMDV).

References

[1] Federal Highway Administration (FHWA), National Bridge Inspection Standards (NBIS), Rule, U.S. Department of Transportation (DOT), 2022. URL: https://www.federalregister.gov/documents/2022/05/06/2022-09512/national-bridge-inspection-standards.

[2] American Society of Civil Engineers (ASCE), ASCE's 2021 Infrastructure Report Card, Report, American Society of Civil Engineers, 2021. URL: https://infrastructurereportcard.org/.

[3] R. Sacks, A. Kedar, A. Borrmann, L. Ma, D. Singer, U. Kattel, SeeBridge Information Delivery Manual (IDM) for Next Generation Bridge Inspection, in: A. A. U. Sattineni, S. A. U. Azhar, D. G. T. U. Castro (Eds.), Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC), International Association for Automation and Robotics in Construction (IAARC), Auburn, USA, 2016, pp. 826–834. doi:https://doi.org/10.22260/ISARC2016/0100.

[4] R. Sacks, A. Kedar, A. Borrmann, L. Ma, I. Brilakis, P. Hüthwohl, S. Daum, U. Kattel, R. Yosef, T. Liebich, et al., SeeBridge as next generation bridge inspection: overview, information delivery manual and model view definition, Automation in Construction 90 (2018) 134–145. doi:http://dx.doi.org/10.1016/j.autcon.2018.02.033.

[5] M. Marzouk, M. Hisham, Bridge information modeling in sustainable bridge management, American Society of Civil Engineers (ASCE), 2012, pp. 457–466. doi:http://dx.doi.org/10.1061/41204(426)57.

[6] C. Shim, N. Yun, H. Song, Application of 3D bridge information modeling to design and construction of bridges, Procedia Engineering 14 (2011) 95–99. doi:http://dx.doi.org/10.1016/j.proeng.2011.07.010.

[7] A. Rashidi, E. Karan, Video to BrIM: Automated 3D as-built documenta-

tion of bridges, Journal of performance of constructed facilities 32 (2018) 1–11. doi:http://dx.doi.org/10.1061/(ASCE)CF.1943-5509.0001163.

[8] B. Kumar, H. Cai, M. Hastak, An assessment of benefits of using BIM on an infrastructure project, in: International Conference on Sustainable Infrastructure, 2017, pp. 88–95. doi:http://dx.doi.org/10.1061/9780784481219.008.

[9] B. McGuire, R. Atadero, C. Clevenger, M. Ozbek, Bridge information modeling for inspection and evaluation, Journal of Bridge Engineering 21 (2016) 04015076. doi:http://dx.doi.org/10.1061/(ASCE)BE.1943-5592.0000850.

[10] S. Jeong, R. Hou, J. P. Lynch, H. Sohn, K. H. Law, An information modeling framework for bridge monitoring, Advances in engineering software 114 (2017) 11–31. doi:http://dx.doi.org/10.1016/j.advengsoft.2017.05.009.

[11] V. Saback de Freitas Bello, C. Popescu, T. Blanksvärd, B. Täljsten, Framework for bridge management systems (BMS) using digital twins, in: Proceedings of the 1st Conference of the European Association on Quality Control of Bridges and Structures: EUROSTRUCT 2021 1, Springer, 2022, pp. 687–694. doi:https://doi.org/10.1007/978-3-030-91877-4_78.

[12] M. Futai, T. Bittencourt, R. Santos, C. Araújo, D. Ribeiro, A. Rocha, R. Ellis, Utilization of digital twins for bridge inspection, monitoring and maintenance, in: Proceedings of the 1st Conference of the European Association on Quality Control of Bridges and Structures: EUROSTRUCT 2021 1, Springer, 2022, pp. 166–173. doi:https://doi.org/10.1007/978-3-030-91877-4_20.

[13] I. Brilakis, Y. Pan, A. Borrmann, H.-G. Mayer, F. Rhein, C. Vos, E. Pettinato, S. Wagner, Built Environment Digital Twining, Report, International Workshop on Built Environment Digital Twinning presented by

TUM Institue for Advanced Study and Siemens AG, 2019. doi:https://doi.org/10.17863/CAM.65445.

[14] E. Negri, L. Fumagalli, M. Macchi, A review of the roles of digital twin in CPS-based production systems, Procedia Manufacturing 11 (2017) 939–948. doi:http://dx.doi.org/10.1016/j.promfg.2017.07.198.

[15] I. Osadcha, A. Jurelionis, P. Fokaides, Geometric parameter updating in digital twin of built assets: A systematic literature review, Journal of Building Engineering 73 (2023) 106704. doi:https://doi.org/10.1016/j.jobe.2023.106704.

[16] S. Vilgertshofer, M. Mafipour, A. Borrmann, J. Martens, T. Blut, R. Becker, J. Blankenbach, A. Göbels, J. Beetz, F. Celik, TwinGen: Advanced technologies to automatically generate digital twins for operation and maintenance of existing bridges, in: Proc. of European Conference on Product and Process Modeling, 2022, pp. 213–221. doi:https://doi.org/10.1201/9781003354222.

[17] M. Mohammadi, M. Rashidi, V. Mousavi, A. Karami, Y. Yu, B. Samali, Quality evaluation of digital twins generated based on UAV photogrammetry and TLS: Bridge case study, Remote Sensing 13 (2021) 3499. doi:http://dx.doi.org/10.3390/rs13173499.

[18] J. J. Shah, M. Mäntylä, Parametric and feature-based CAD/CAM: Concepts, Techniques and Applications, ohn Wiley & Sons, New York, 1995.

[19] Y. Ji, A. Borrmann, J. Beetz, M. Obergrießer, Exchange of Parametric Bridge Models Using a Neutral Data Format, Journal of Computing in Civil Engineering 27 (2013) 593–606. doi:10.1061/(ASCE)CP.1943-5487.0000286.

[20] C. Schultz, M. Bhatt, A. Borrmann, Bridging qualitative spatial constraints and feature-based parametric modelling: Expressing visibility and

movement constraints, Advanced Engineering Informatics 31 (2015) 2–17. doi:10.1016/j.aei.2015.10.004.

[21] D. Mun, S. Han, J. Kim, Y. Oh, A set of standard modeling commands for the history-based parametric approach, Computer-Aided Design 35 (2003) 1171–1179. doi:http://dx.doi.org/10.1016/S0010-4485(03)00022-8.

[22] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton, W. Puech, A comprehensive process of reverse engineering from 3D meshes to CAD models, Computer-Aided Design 45 (2013) 1382–1393. doi:https://doi.org/10.1016/j.cad.2013.06.004.

[23] T. Varady, R. R. Martin, J. Cox, Reverse engineering of geometric models—an introduction, Computer-aided design 29 (1997) 255–268. doi:https://doi.org/10.1016/S0010-4485(96)00054-1.

[24] A. Durupt, S. Remy, G. Ducellier, B. Eynard, From a 3D point cloud to an engineering CAD model: a knowledge-product-based approach for reverse engineering, Virtual and Physical Prototyping 3 (2008) 51–59. doi:http://dx.doi.org/10.1080/17452750802047917.

[25] H. Kim, C. Yeo, I. D. Lee, D. Mun, Deep-learning-based retrieval of piping component catalogs for plant 3D CAD model reconstruction, Computers in Industry 123 (2020) 103320. doi:https://doi.org/10.1016/j.compind.2020.103320.

[26] P. Singh, S. K. Choudhary, Introduction: optimization and metaheuristics algorithms, in: Metaheuristic and evolutionary computation: algorithms and applications, Springer, 2021, pp. 3–33. doi:http://dx.doi.org/10.1007/978-981-15-7571-6_1.

[27] M. Shariati, M. S. Mafipour, B. Ghahremani, F. Azarhomayun, M. Ahmadi, N. T. Trung, A. Shariati, A novel hybrid extreme learning machine–grey wolf optimizer (ELM-GWO) model to predict compressive strength of con-

crete with partial replacements for cement, Engineering with Computers 32 (2020) 757–779. doi:https://doi.org/10.1007/s00366-020-01081-0.

[28] M. Shariati, M. S. Mafipour, P. Mehrabi, A. Shariati, A. Toghroli, N. T. Trung, M. N. Salih, A novel approach to predict shear strength of tilted angle connectors using artificial intelligence techniques, Engineering with Computers 37 (2020) 2089–2109. doi:https://doi.org/10.1007/s00366-019-00930-x.

[29] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-international conference on neural networks, volume 4, IEEE, 1995, pp. 1942–1948. doi:10.1109/ICNN.1995.488968.

[30] J. H. Holland, Genetic algorithms, Scientific american 267 (1992) 66–73. doi:http://dx.doi.org/10.1038/scientificamerican0792-66.

[31] R. V. Rao, V. J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-aided design 43 (2011) 303–315. doi:http://dx.doi.org/10.1016/j.cad.2010.12.015.

[32] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, Advances in engineering software 69 (2014) 46–61. doi:http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.

[33] X.-S. Yang, Firefly algorithms for multimodal optimization, in: International symposium on stochastic algorithms, Springer, 2009, pp. 169–178. doi:http://dx.doi.org/10.1007/978-3-642-04944-6_14.

[34] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, in: Computer graphics forum, volume 26, Wiley Online Library, 2007, pp. 214–226. doi:http://dx.doi.org/10.1111/j.1467-8659.2007.01016.x.

[35] T. Rabbani, F. Van Den Heuvel, Efficient hough transform for automatic detection of cylinders in point clouds, Proc ISPRS Workshop Laser Scan 2005, ISPRS Arch 36 (2005) 60–65.

[36] T. Rabbani, Automatic reconstruction of industrial installations using point clouds and images, Report, TU Delft: Faculty of Civil Engineering and Geosciences, 2006. URL: http://resolver.tudelft.nl/uuid:0012068e-93b4-4bd9-a9b3-9c579ae7c91a.

[37] L. Li, M. Sung, A. Dubrovina, L. Yi, L. J. Guibas, Supervised fitting of geometric primitives to 3d point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2652–2660. doi:http://dx.doi.org/10.1109/CVPR.2019.00276.

[38] F. P. Preparata, S. J. Hong, Convex hulls of finite sets of points in two and three dimensions, Communications of the ACM 20 (1977) 87–93. doi:http://dx.doi.org/10.1145/359423.359430.

[39] H. Edelsbrunner, E. P. Mücke, Three-dimensional alpha shapes, ACM Transactions On Graphics (TOG) 13 (1994) 43–72. doi:https://doi.org/10.1145/174462.156635.

[40] M. Duckham, L. Kulik, M. Worboys, A. Galton, Efficient generation of simple polygons for characterizing the shape of a set of points in the plane, Pattern Recognition 41 (2008) 3224–3236. doi:https://doi.org/10.1016/j.patcog.2008.03.023.

[41] A. Moreira, M. Y. Santos, Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points, in: Proceedings of the Second International Conference on Computer Graphics Theory and Applications, INSTICC Press, 2007, pp. 61–68. doi:10.5220/0002080800610068.

[42] N. Amenta, M. Bern, D. Eppstein, The Crust and the β-Skeleton: Com-

binatorial Curve Reconstruction, Graphical Models and Image Processing 60 (1998) 125–135. doi:https://doi.org/10.1006/gmip.1998.0465.

[43] R. Lu, I. Brilakis, Digital twinning of existing reinforced concrete bridges from labelled point clusters, Automation in Construction 105 (2019) 102837. doi:https://doi.org/10.1016/j.autcon.2019.102837.

[44] G. Zhang, P. A. Vela, P. Karasev, I. Brilakis, A sparsity-inducing optimization-based algorithm for planar patches extraction from noisy point-cloud data, Computer-Aided Civil and Infrastructure Engineering 30 (2015) 85–102. doi:http://dx.doi.org/10.1111/mice.12063.

[45] B. Wang, C. Yin, H. Luo, J. C. Cheng, Q. Wang, Fully automated generation of parametric BIM for MEP scenes based on terrestrial laser scanning data, Automation in Construction 125 (2021) 103615. doi:http://dx.doi.org/10.1016/j.autcon.2021.103615.

[46] L. Yang, J. C. Cheng, Q. Wang, Semi-automated generation of parametric BIM for steel structures based on terrestrial laser scanning data, Automation in Construction 112 (2020) 103037. doi:https://doi.org/10.1016/j.autcon.2019.103037.

[47] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (1981) 381–395. doi:https://doi.org/10.1145/358669.358692.

[48] A. Dimitrov, R. Gu, M. Golparvar-Fard, Non-uniform B-spline surface fitting from unordered 3D point clouds for as-built modeling, Computer-Aided Civil and Infrastructure Engineering 31 (2016) 483–498. doi:http://dx.doi.org/10.1111/mice.12192.

[49] S.-W. Kwon, F. Bosche, C. Kim, C. T. Haas, K. A. Liapi, Fitting range data to primitives for rapid local 3D modeling using sparse range point

clouds, Automation in Construction 13 (2004) 67–81. doi:https://doi.org/10.1016/j.autcon.2003.08.007, the best of ISARC 2002.

[50] A. Justo, D. Lamas, A. Sánchez-Rodríguez, M. Soilán, B. Riveiro, Generating IFC-compliant models and structural graphs of truss bridges from dense point clouds, Automation in Construction 149 (2023) 104786. doi:https://doi.org/10.1016/j.autcon.2023.104786.

[51] E. Valero, A. Adán, C. Cerrada, Automatic method for building indoor boundary models from dense point clouds collected by laser scanners, Sensors 12 (2012) 16099–16115. doi:http://dx.doi.org/10.3390/s121216099.

[52] S. Oesau, F. Lafarge, P. Alliez, Indoor scene reconstruction using feature sensitive primitive extraction and graph-cut, ISPRS journal of photogrammetry and remote sensing 90 (2014) 68–82. doi:http://dx.doi.org/10.1016/j.isprsjprs.2014.02.004.

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

[54] S. B. Walsh, D. J. Borello, B. Guldur, J. F. Hajjar, Data processing of point clouds for object detection for structural engineering applications, Computer-Aided Civil and Infrastructure Engineering 28 (2013) 495–508. doi:http://dx.doi.org/10.1111/mice.12016.

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

[56] Y. Yan, J. F. Hajjar, Geometric models from laser scanning data for superstructure components of steel girder bridges, Automation in Construc-

tion 142 (2022) 104484. doi:https://doi.org/10.1016/j.autcon.2022.104484.

[57] L. Barazzetti, Parametric as-built model generation of complex shapes from point clouds, Advanced Engineering Informatics 30 (2016) 298–311. doi:https://doi.org/10.1016/j.aei.2016.03.005.

[58] M. Arikan, M. Schwärzler, S. Flöry, M. Wimmer, S. Maierhofer, O-snap: Optimization-based snapping for modeling architecture, ACM Transactions on Graphics (TOG) 32 (2013) 1–15. doi:http://dx.doi.org/10.1145/2421636.2421642.

[59] W. Sui, L. Wang, B. Fan, H. Xiao, H. Wu, C. Pan, Layer-wise floorplan extraction for automatic urban building reconstruction, IEEE transactions on visualization and computer graphics 22 (2015) 1261–1277. doi:http://dx.doi.org/10.1109/TVCG.2015.2505296.

[60] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: kdd, volume 96, 1996, pp. 226–231.

[61] T. Xia, J. Yang, L. Chen, Automated semantic segmentation of bridge point cloud based on local descriptor and machine learning, Automation in Construction 133 (2022) 103992. doi:http://dx.doi.org/10.1016/j.autcon.2021.103992.

[62] S. M. Ahmed, Y. Z. Tan, C. M. Chew, A. Al Mamun, F. S. Wong, Edge and corner detection for unorganized 3d point clouds with application to robotic welding, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 7350–7355. doi:http://dx.doi.org/10.1109/IROS.2018.8593910.

[63] J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Springer Science & Business Media, 2013. doi:http://dx.doi.org/10.1007/978-1-4757-0450-1.

[64] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4558–4567. doi:http://dx.doi.org/10.1109/CVPR.2018.00479.

[65] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, Advances in neural information processing systems 30 (2017). doi:https://doi.org/10.48550/arXiv.1706.02413.

[66] H. Zhao, L. Jiang, J. Jia, P. H. Torr, V. Koltun, Point transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 16259–16268. doi:http://dx.doi.org/10.1109/ICCV48922.2021.01595.

[67] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, L. J. Guibas, Kpconv: Flexible and deformable convolution for point clouds, in: Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 6411–6420. doi:https://doi.org/10.48550/arXiv.1904.08889.

[68] Z. W. Geem, J. H. Kim, G. V. Loganathan, A new heuristic optimization algorithm: harmony search, simulation 76 (2001) 60–68. doi:https://doi.org/10.1177/003754970107600201.

[69] A. K. Qin, V. L. Huang, P. N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE transactions on Evolutionary Computation 13 (2008) 398–417. doi:https://doi.org/10.1109/TEVC.2008.927706.

[70] B. Xing, W.-J. Gao, B. Xing, W.-J. Gao, Invasive weed optimization algorithm, Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms 62 (2014) 177–181. doi:https://doi.org/10.1007/978-3-319-03404-1_13.

[71] M. Eusuff, K. Lansey, F. Pasha, Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, Engineering optimization 38 (2006) 129–154. doi:https://doi.org/10.1080/03052150500384759.

[72] P. J. Van Laarhoven, E. H. Aarts, Simulated Annealing: Theory and Applications, volume 1, Springer Dordrecht, 1987. doi:https://doi.org/10.1007/978-94-015-7744-1.

[73] H. Garg, A hybrid PSO-GA algorithm for constrained optimization problems, Applied Mathematics and Computation 274 (2016) 292–305. doi:https://doi.org/10.1016/j.amc.2015.11.001.

[74] L. Truong-Hong, R. Lindenbergh, Automatically extracting surfaces of reinforced concrete bridges from terrestrial laser scanning point clouds, Automation in Construction 135 (2022) 104127. doi:http://dx.doi.org/10.1016/j.autcon.2021.104127.

[75] D. Lamas, A. Justo, M. Soilán, M. Cabaleiro, B. Riveiro, Instance and semantic segmentation of point clouds of large metallic truss bridges, Automation in Construction 151 (2023) 104865. doi:https://doi.org/10.1016/j.autcon.2023.104865.

[76] B. Riveiro, M. DeJong, B. Conde, Automated processing of large point clouds for structural health monitoring of masonry arch bridges, Automation in Construction 72 (2016) 258–268. doi:https://doi.org/10.1016/j.autcon.2016.02.009.

[77] R. Lu, I. Brilakis, C. R. Middleton, Detection of structural components in point clouds of existing RC bridges, Computer-Aided Civil and Infrastructure Engineering 34 (2019) 191–212. doi:http://dx.doi.org/10.1111/mice.12407.

[78] Y. Yan, J. F. Hajjar, Automated extraction of structural elements in steel girder bridges from laser point clouds, Automation in Con-

struction 125 (2021) 103582. doi:http://dx.doi.org/10.1016/j.autcon.2021.103582.

[79] Y. Pan, Y. Dong, D. Wang, A. Chen, Z. Ye, Three-dimensional reconstruction of structural surface model of heritage bridges using UAV-based photogrammetric point clouds, Remote Sensing 11 (2019) 1204. doi:http://dx.doi.org/10.3390/rs11101204.

[80] X. Yang, E. del Rey Castillo, Y. Zou, L. Wotherspoon, Y. Tan, Automated semantic segmentation of bridge components from large-scale point clouds using a weighted superpoint graph, Automation in Construction 142 (2022) 104519. doi:http://dx.doi.org/10.1016/j.autcon.2022.104519.

[81] M. Mafipour, S. Vilgertshofer, A. Borrmann, Digital twinning of bridges from point cloud data by deep learning and parametric models, in: ECPPM 2022-eWork and eBusiness in Architecture, Engineering and Construction 2022, CRC Press, 2023, pp. 543–550. doi:10.1201/9781003354222-27.

[82] S. Asaeedi, F. Didehvar, A. Mohades, α-Concave hull, a generalization of convex hull, Theoretical Computer Science 702 (2017) 48–59. doi:https://doi.org/10.1016/j.tcs.2017.08.014.