Automation in Construction 149 (2023) 104786

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Generating IFC-compliant models and structural graphs of truss bridges from dense point clouds

Andrés Justo [a,*], Daniel Lamas [a], Ana Sánchez-Rodríguez [b], Mario Soilán [a], Belén Riveiro [a]

[a] CINTECX, Universidade de Vigo, GeoTECH Group, Campus Universitario de Vigo, As Lagoas, Marcosende, 36310 Vigo, Spain
[b] ICITECH, Universitat Politècnica de València, Camino de Vera s/n, 46022, Valencia, Spain

ARTICLE INFO

Keywords:
Point cloud
IFC
BIM
Truss bridge
Bounding box

ABSTRACT

The IFC schema has been evolving towards the infrastructure domain. Furthermore, the use of laser scanning technologies as means to digitalize and monitor infrastructures has also significantly increased. This work presents an automated modelling approach for truss bridges that utilizes laser scanning data as its source for geometrical information. The methodology takes a partially instance-segmented point cloud of a truss bridge and generates both an IFC-compliant information model of the truss and the corresponding structural graph. This process uses bounding boxes and their collisions to overcome the missing data from the partial segmentation to create the truss model, as well as to identify the nodes that connect the different truss members. The methodology was tested on a use case made of 272 members and obtained the truss model and structural graph files.

1. Introduction

Infrastructure systems are directly linked to the economy and society of a nation. In particular, Critical Infrastructure Systems (CIS) such as water supply, transport or power supply, are driving components of its development. In addition, CIS also play a central role in the mitigation, management and recovery from disaster scenarios. The needs of a population grow along with its size, which means that the infrastructure backbone that support them must expand as well. This increase in scale and complexity results in a dependency relationship between CIS that enables them to function properly. For instance, transport infrastructure distributes the resources used by other systems, and power supply is needed in almost every scenario. While this synergy improves the overall quality and efficiency of CISs, it also increases its vulnerabilities. The collapse or operational shutdown of a system could create a domino effect that impacts all related CISs. Therefore, the resilience of these systems is a key priority, as they should be able to withstand or mitigate the consequences of these scenarios, protect their users, reduce costs derived from them, and hasten the recovery from such cases [1–3].

Transport infrastructure is identified as a CIS because the transport of both goods and people is crucial to the well-functioning of any society. Similarly to other CIS, the increase in demand is translated into an increase in the size and complexity, while also exceeding expected traffic values for existing assets. In the case of bridges, many of the 1234 km of road bridges over 100 m long in the EU were built during the 1950s and have reached the end of their design life, surpassing the traffic load that was expected at design [4]. This trend carries the need for efficient and cost-effective technologies to support infrastructure management throughout their entire lifecycle [5]. This is particularly accentuated in bridges, where manual visual inspection is still the most common method to assess their condition.

Interoperability and digitalization are also key concerns in the construction industry. The use of paper documents or fragmented information in different formats results in the loss of information and high costs and delays [6,7]. Gallaher et al. presented a cost analysis report of inadequate interoperability in the U.S. capital facilities industry across the entire life-cycle, estimating a loss of 15.8 billion dollars per year [8]. A digital model of the bridge can serve as a single source of truth, where all the digital information is centralized in a clear and accessible manner. The digital model also serves as a foundation for interoperability and integration with other technologies that can harness that data. In this context, Building Information Modelling (BIM) presents a collaborative workflow where the interested parties can work together in a Common Data Environment (CDE) [9]. The different specialized agents can exchange information using a unique federated model described using a data model, such as the Industry Foundation Classes

* Corresponding author.
E-mail addresses: andres.justo.dominguez@uvigo.gal (A. Justo), daniel.lamas.novoa@uvigo.gal (D. Lamas), asanrod3@upv.es (A. Sánchez-Rodríguez), msoilan@uvigo.gal (M. Soilán), belenriveiro@uvigo.gal (B. Riveiro).

https://doi.org/10.1016/j.autcon.2023.104786
Received 27 October 2022; Received in revised form 31 January 2023; Accepted 3 February 2023
Available online 13 February 2023

(IFC) presented by buildingSMART [10]. This is particularly beneficial in large scale projects that combine multiple disciplines and domain-specific teams, such as transport infrastructure.

IFC has been evolving towards the infrastructure domain over the last years with its 4.X releases [11]. IFC 4.0 first enabled the extension of IFC to infrastructure and is a full ISO standard (ISO 16739-1:2018 [12]), while its predecessor IFC 2 × 3 is an ISO/PAS. IFC 4.1 introduced the key component of an infrastructure information model, the alignment, to serve as a linear reference system for the positioning of elements and interlinkage with other infrastructures of the network. IFC 4.2 presented extensions to the schema that enabled it to describe bridges. However, buildingSMART decided to harmonize and unify all the infrastructure domains under a single release, instead of including them in individualized versions. The newly released IFC 4.3 represents that vision, which encompasses bridges, road, railways, ports and waterways. It introduced several hierarchy and nomenclature changes, as well as some new functionalities such as the lateral profile inclination of the alignment. At the time of writing, IFC 4.3 is under ISO voting and receiving continuous updates. Also, IFC4.4 is set to be an extension of IFC 4.3 to mainly include tunnel functionalities [13]. However, since IFC 4.3 is still new and was subjected to many changes both during development and in its release, many programming libraries and viewers do not properly support it yet. Due to this, or because IFC 4.3 is still not an ISO standard, many existing efforts use pre-existing IFC versions. Koo et al. (2020) mentioned how, due to the lack of ISO standardization for infrastructure elements of IFC, these entities needed to be mapped to similar architectural entities, or to proxy ones [14]. Kwon et al. (2020) presented an extension to IFC 4.2 to model alignment-based railway tracks [15].

In this setting, point clouds obtained using laser scanning technologies offer a robust basis for the geometrical definition of information models for infrastructures. Bariczová et al. (2021) used Terrestrial Laser Scanning (TLS) data to verify the geometry of walls defined using IFC 4.0 [16]. Barazzetti et al. (2020) states that the integration of geospatial information along with other data, such as LiDAR (Light Detection And Ranging) point clouds, is key in the generation of BIM-GIS models of infrastructure [17]. Ariyachandra et al. (2020) present a method that detects railway mast from air-borne LiDAR data and delivers an IFC model of the results [18]. It also serves as means to obtain information models of the assets, as point clouds can provide the needed geometrical information. There are several works and reviews that detail the current state of this technology [19–22] in the transport infrastructure domain. In the case of bridges, existing works using point clouds and IFC often deal with non-truss bridges and use meshes for their representation. Sánchez-Rodríguez et al. (2020) presented the case of a masonry bridge, where the point cloud was translated into meshes that were then formatted following the IFC schema [23]. Isailović et al. (2020) described a procedure to update the as-built IFC models through the incorporation of damage meshes [24]. As for truss structures, current works using point clouds often target wooden structures. For instance, Prati et al. (2019) presented a 3D model of the wooden roofing of St Peter's Cathedral, generated from TLS data [25]. Hermida et al. (2020) proposed an algorithm to obtain 2D models of variable inertia from LiDAR data of timber trusses [26]. To the author's knowledge, no existing works were found that dealt with the fully automated generation of truss models and the corresponding structural graph from point cloud data.

Given this context, the core objective of this work is to create an IFC-compliant model of a truss, as well as a structural graph that represents it, from partially instance-segmented point cloud data. The key concept behind it is the use of the bounding boxes that encompass each of the partially segmented truss members. As will be explained throughout this work, this representation itself overcomes the partial segmentation, while bounding box collisions are used to determine the connection relationships between truss members, obtaining a structural graph. Therefore, the contribution of this work is threefold:

1. Automated generation of a truss bridge IFC-compliant model.
2. Automated generation of a structural graph representing the truss.
3. Overcoming the missing data from the truss bridge point cloud segmentation.

This work is the second part of an automated pipeline that takes the raw point cloud and outputs both the IFC model, and the structural graph made of nodes and edges. The first part oversees the point cloud segmentation, while the second deals with the automated generation of the model and its correction, as well as the obtention of the structural graph. To better illustrate the workflow, Fig. 1 presents a diagram where the truss can be seen evolving from the raw point cloud to its segmented form, and then to the IFC model and the structural graph.

This work is structured as follows: In Section 2, the context of this work is introduced. It tackles the point cloud segmentation (Section 2.1), the bounding boxes (Section 2.2), and the IFC entities and relationships used to build the model (Section 2.3). Section 3 explains the methodology used for the truss model (Section 3.1) and for the structural graph (Section 3.2). Section 4 shows the results of the methodology by presenting both the final IFC model (Section 4.1) and the structural graph (Section 4.2) in their respective receiving software/viewer. Section 5 discusses the results, addressing the limitations and shortcomings. Finally, Section 6 offers the conclusions along with future lines of work to overcome the shortcomings and further improve the methodology.

2. Context

2.1. Point cloud segmentation

The starting point of the methodology is a partially instance-segmented point cloud of a truss bridge, which is depicted in Fig. 2. The overall dimensions of this truss are 64 × 5.6 × 4.8 m and is made of 272 members. It was extracted from a 594 m structure that rests on 11 pillars.

This type of truss can be divided into three types of faces: vertical, horizontal, or interior. The two vertical faces contain the vertical posts, as well as the diagonals. The bottom horizontal face contains the struts and bottom lateral braces. The seventeen interior faces, which can be seen as cross-sections of the truss, contain interior braces and interior lateral braces. Furthermore, there are four chords that delimit the bounds of the truss. These are not assigned to a face type since they are located in the intersection between the horizontal and vertical faces. This nomenclature can be seen in Fig. 3, where the front view corresponds to a vertical face, the bottom view to a horizontal face, and the side view shows the projection of all interior faces. Nevertheless, the methodology only distinguishes between chords, straight members (vertical posts or struts), and diagonals (bottom lateral brace and diagonals), to be as generalized as possible.

In the truss shown in this work, the top horizontal face could not be properly segmented due to severe occlusions and the density of the point cloud. This is a common problem when dealing with transport infrastructure point clouds, since they are usually acquired from long distances and/or occluded by other structural members. Therefore, the top horizontal face was deemed unfit for analysis and excluded from the model.

To process this truss type, the point cloud processing was tailored to meet a certain input criterion for the methodology:

- There must be at least a labelled set of points for each member.
- The labelled set of points must be aligned with the direction of the member they belong to.
- The chords are to be segmented as completely as possible since they delimit the bounds of the truss and are used as reference.
- Unwanted elements must be omitted (e.g., the handrail).

To fulfil those requirements, the point cloud processing assesses the

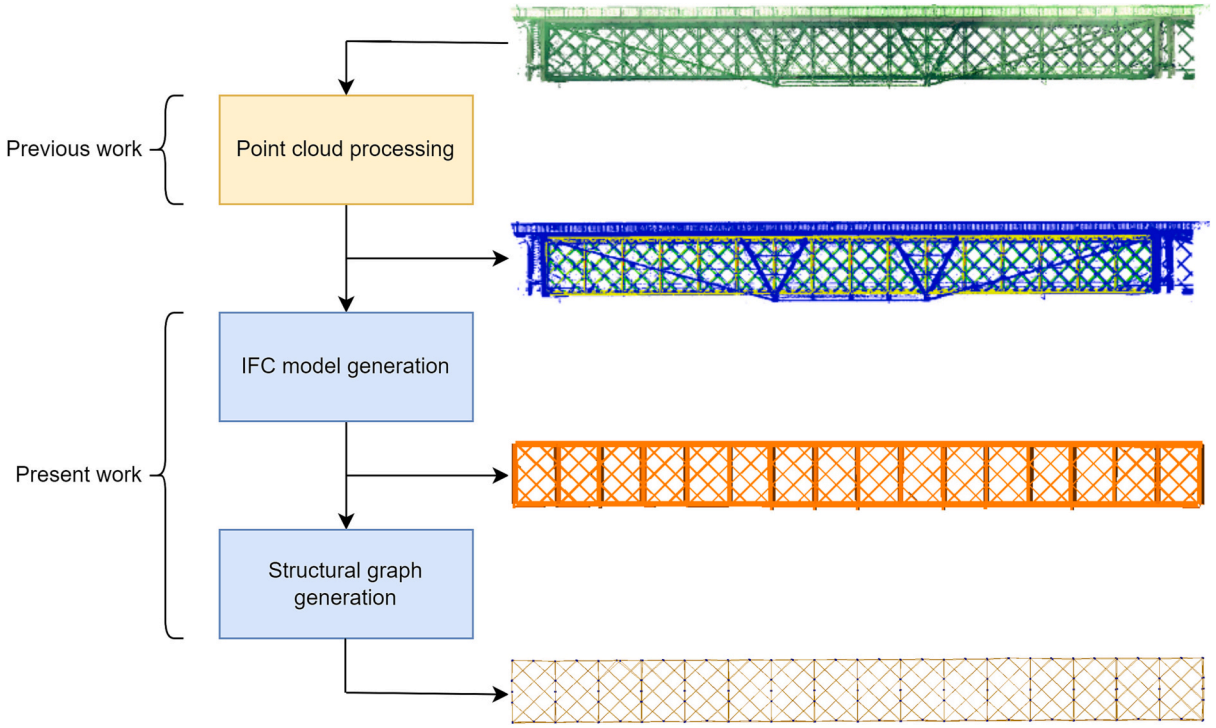

Fig. 1. Pipeline workflow.

Fig. 2. Truss bridge studied.

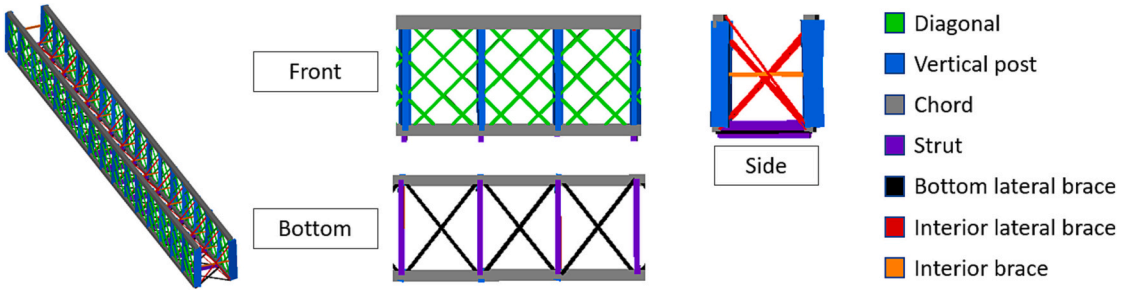

Fig. 3. Different types of members.

distance between points and the expected direction of each type of member, since they follow a certain pattern inside their group. For instance, all diagonals follow either one of two directions, and all the vertical posts are aligned with the Z axis. By comparing those expected directions with the main component obtained from applying Principal Component Analysis (PCA) to a set of points, it is possible to exclude

those which are not guaranteed to belong to that element. The use of this criterion, in return, results in a partially instance-segmented point cloud, since a lot of the points are excluded through this process, as seen in blue in Fig. 4.

Once all the members have been processed, the final step is to format the information to be usable in the next step. This is done via the generation of a .csv file that feeds the software developed. There are three columns per member, which represent the X, Y and Z coordinates of its points, respectively. The headers of the columns provide the identification and classification of the member that they are describing. This is done by following the naming schema of: "**TrussName_Face_-MemberType_MemberName**". This gives information about the face to which the member belongs (e.g., horizontal bottom face), its type (e.g., diagonal, strut…) and its identification (e.g., member ID and truss name). Since in this scenario there is only one truss, the truss name is equal for every column.

2.2. Bounding boxes

This section aims to explain what a bounding box is in the context of this work, and the reason behind its use in the methodology. In general terms, a bounding box is the box region that delimits a set of objects or points, in such a way that the entire set is confined inside it. A bounding box is defined by a centre, a set of orthogonal axes, and the extent along those axes to reach the face normal to them. In this work, the axis which has the largest associated extent is called the main axis of the bounding box. A representation of these parameters can be seen in Fig. 5.

Usually, when referring to a bounding box, what is actually referred to is the minimum bounding box, which represents the minimum volume of space that completely contains an object or set of objects. This box can either be axis aligned, meaning that its edges are parallel to one of the XYZ axis, or oriented, where their axes are arbitrary orthogonal vectors. Fig. 6 presents these definitions in a simplified 2D scenario.

For an axis aligned scenario, the minimum bounding box is usually simple to define. All that it takes is the maximum and minimum coordinate in each axis of each of its contained objects. In an oriented scenario, however, this calculation is much more challenging, since there is a possible solution for each set of orthogonal axes. Furthermore, this complexity is accentuated in the 3D space. The calculation of a minimum oriented bounding box is outside of the scope of this paper. On a first version of the work, the GeometricToolsEngine [27] was used to compute this minimum oriented bounding box for each of the elements. However, the iteration nature of the calculation led to complex scenarios and unexpected behaviours. Fig. 7 presents the main issue using these minimum oriented bounding boxes, where two boxes with almost identical volumes present completely different rotations along their main axis.

To avoid this scenario, an approximation that was consistent in the rotation along the main axis of the bounding box was used. The generation of this approximated box, based on Principal Component Analysis (PCA), is described in Section 3.1.1. Fig. 7 also shows how the bounding boxes aid to overcome the partial segmentation. Two segmented parts of the truss member might be disconnected due to noise, occlusions, or purposely being omitted from the segmentation to avoid conflicting information in intersections. However, the bounding box formed by the existing segments includes any possible space that the missing parts in between might have occupied.

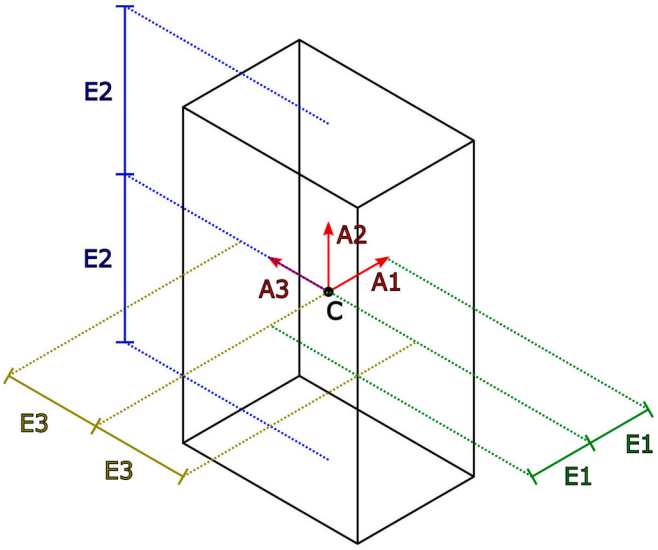

Fig. 5. Bounding box parameters. C – center. A – axis. E – extent. A2 as main axis.

2.3. IFC

The purpose of this section is to describe the different IFC entities and relationships used to build the model. The first thing to define is the IFC schema version to be used. As IFC 4.3 is still not widely available in programming libraries and visualization software, IFC 4.1 was used instead [28]. Nevertheless, the development took into account existing documentation about IFC 4.3 in order to make software as upwards compatible as possible, so that it can be updated in the future. To aid in the following explanation, Fig. 8 presents a simplified diagram of the different IFC entities used.

The entity used to model each member is *IfcMember*. In the case of the truss in its entirety, it is described using an *IfcElementAssembly* that aggregates all the truss members into a single instance using *IfcRelAggregates*. This assembly is placed in the model world coordinate system using *IfcLocalPlacement*. The truss members are also placed using an *IfcLocalPlacement*, but relative to the placement of the truss assembly, instead of the model world coordinate system. This way, in the case of a full bridge model that uses an alignment, the only element that must change to linear placement (*IfcLinearPlacement*) is the truss assembly. The solid used to represent each member is an *IfcExtrudedAreaSolid*, which requires a profile and a length. This representation will be used to form rectangular prisms that match the shape of the bounding boxes. The extrusion length is obtained from the biggest dimension of the bounding box obtained from the member points. The extruded profile is represented as a *IfcRectangleProfileDef* that uses the two remaining bounding box dimensions. A more detailed comment on the members profiles is given in the discussion, in Section 5.

The model also includes the relationships between members, more specifically, the connections amongst themselves. This is modelled using series of *IfcRelConnectsElements* that are generated through bounding box collisions, as explained in Section 3.1.3.

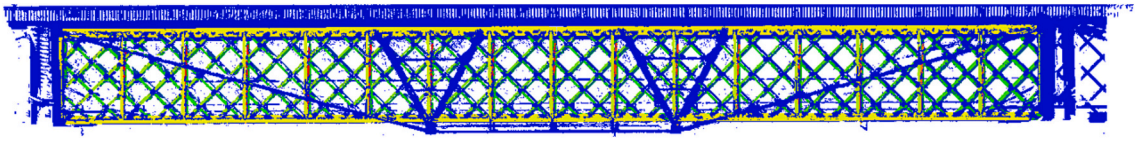

Fig. 4. Segmented point cloud. Blue – points omitted from segmentation. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

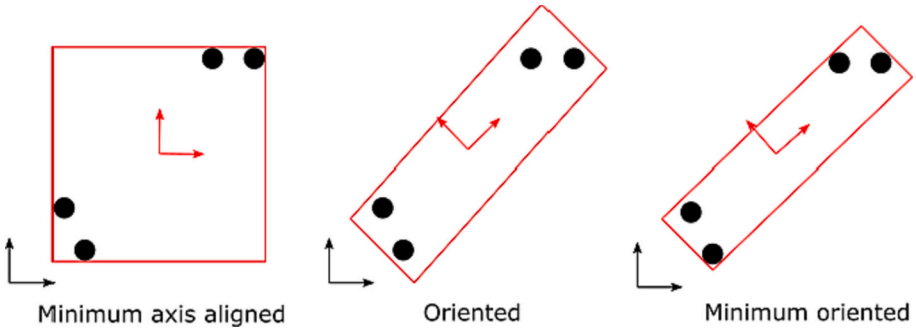

Fig. 6. Axis aligned vs Oriented bounding box.

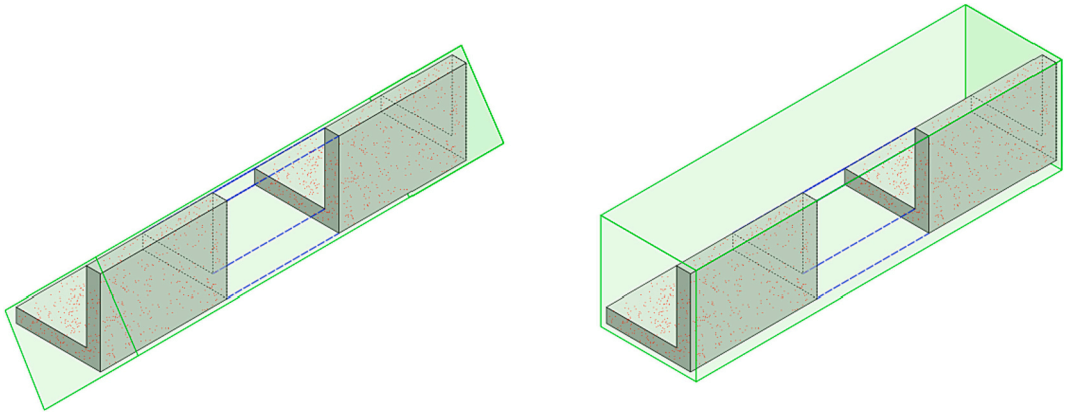

Fig. 7. Minimum oriented bounding box – Two different rotations.

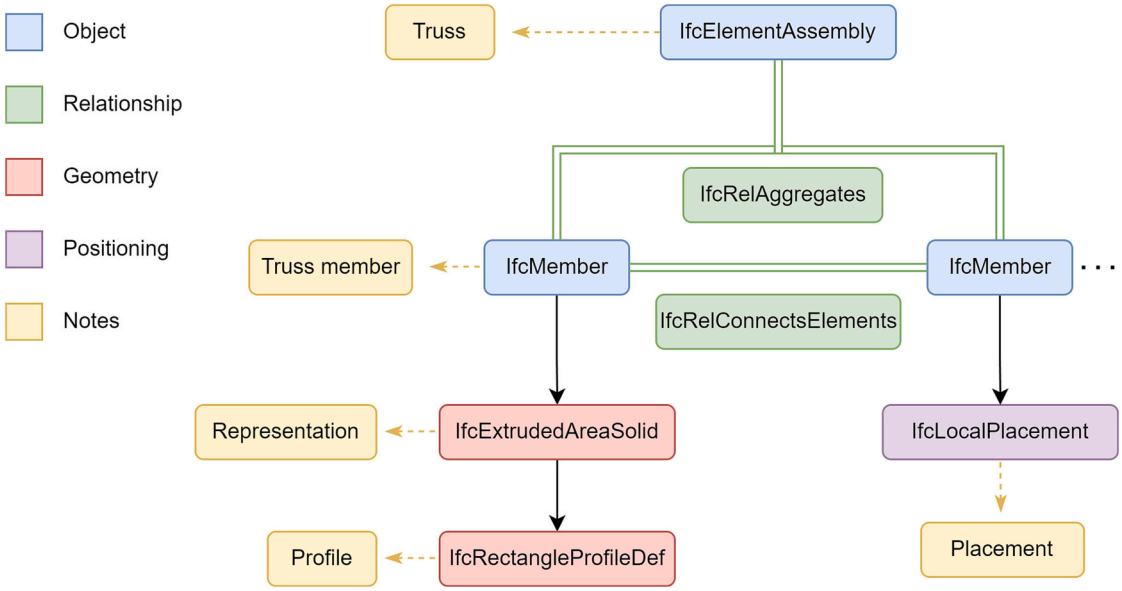

Fig. 8. IFC entity diagram.

3. Methodology

As mentioned in the introduction, the methodology generates both the IFC model of the truss and its structural graph. Therefore, this section is split following that pattern. Section 3.1 presents the truss model generation, while Section 3.2 describes the construction of the structural graph. To aid in the explanation, Fig. 9 presents an overall view of the methodology following the steps of the different sub-sections.

3.1. Truss model

The aim of this section is to cover the entire model generation of the truss, represented by "IFC model generation" in Fig. 1. This process can be split into four steps, resulting in different versions of the truss.

The first step processes the input data described in Section 2.1 and instantiates the different member objects. These objects contain the identification data as marked by the headers of the .csv, as well as the

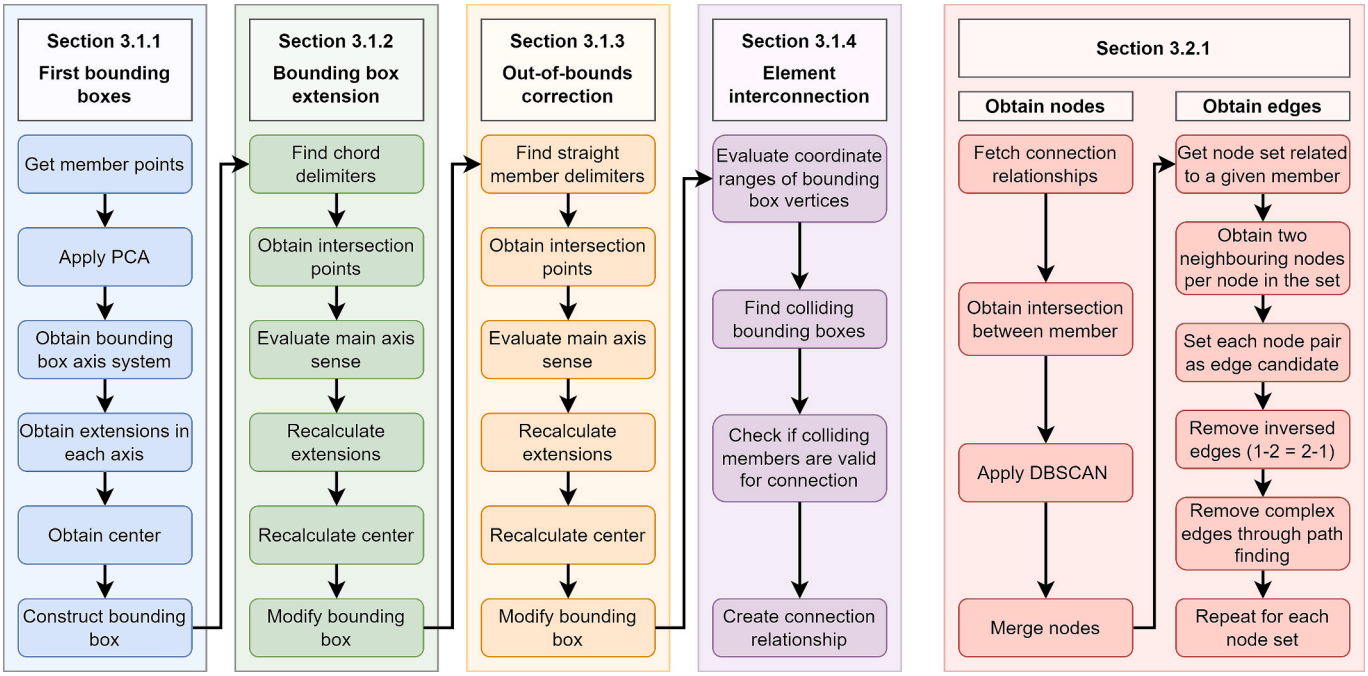

Fig. 9. Overall methodology view.

bounding boxes created directly from the given points. The second step deals with the extension of the bounding boxes until they reach the chords. The third step corrects the edge cases, as they go out of bounds during this extension. Finally, the fourth step obtains the connection relationships between members and generates the IFC model. Therefore, this section is split following that criterion. To illustrate these steps, Fig. 10 presents a simplified flow from the initial state to the desired end state. It must be noted that this approach deals with an isolated truss, and therefore needs to use some parts of it as a reference. As such, the chords were chosen as the reference, as they delimit the bounds of the truss and its members. This means that the chords of the truss are considered as appropriately segmented in terms of length and are therefore not corrected. In the case of analysing the truss in the context of a full bridge, the chords could be corrected using the piers as reference.

3.1.1. *First step – Input processing and first bounding boxes*

The purpose of this step is to move from the input data to the state **A** of the truss as seen in Fig. 10. This means that, for each member, a bounding box is to be computed from the member points contained in the input file. As mentioned in Section 2.1, the input data .csv contains both the member points and the identification information of each member. Therefore, to proceed with the bounding box generation, the data contained in the file are to be extracted. For each member, the identification is obtained by splitting the headers using the predefined delimiting character, in this case "_", and the three columns of the coordinates are joined together into a list of points.

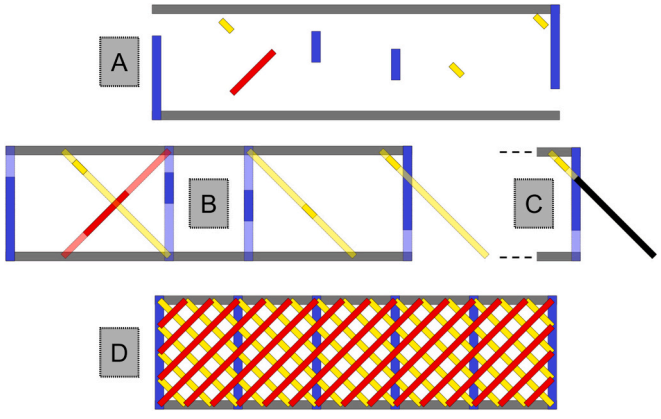

Fig. 10. Correction process overview. A – First state. B – Second state. C – Third state. D - Final state.

Afterwards, the member points are subjected to a Principal Component Analysis (PCA). The resulting main component vector will represent the main axis of the bounding box. To obtain the remaining two axes, the main axis is used as a guide to rotate a XYZ axis system so that the Z' axis matches the main axis. This procedure can be seen in Fig. 11. At first, the system rotates **Az** in the +Z axis, which makes X' parallel to the projection of the main axis in the XY plane (**MAh**). Then, the system rotates **Ay'** in the +Y′ axis, resulting in the Z' axis being parallel to the main axis (**MA**). By performing the rotation in this manner, the Y′ axis is contained in the XY plane. Through this restriction, a consistent orthogonal system is obtained for each member, and is used as the bounding box axis system for the member.

Once the orthogonal axis system has been established, the points are transformed to this new axis system. Then, the coordinate ranges in each axis are used to obtain the extent of the bounding box in each of the three directions, as well as its centre. Finally, the centre is transformed back into the original X, Y, Z coordinate system.

These parameters (centre, axes, and extent) are also used to set the placement of the member, which is made of translation and rotation. The translation is obtained using the centre and the extent of the main axis, since, by our criterion, the member is placed using the centre of one of the faces normal to the main axis, not the centre of the bounding box itself. The rotation is directly extracted from the axis system of the bounding box.

At this point, both the bounding box and the placement of the member have been defined. The process is then repeated for the rest of the members. If no further processing is done, the truss is at the first state (**A**) of Fig. 10. For better understanding of this state, Fig. 12 shows what the IFC model of the truss looks like at the end of this step. As it can be seen, the members are not interconnected due to the use of a partially instance-segmented point cloud. Nevertheless, since the main axis of the bounding boxes matches the expected direction of the members, the

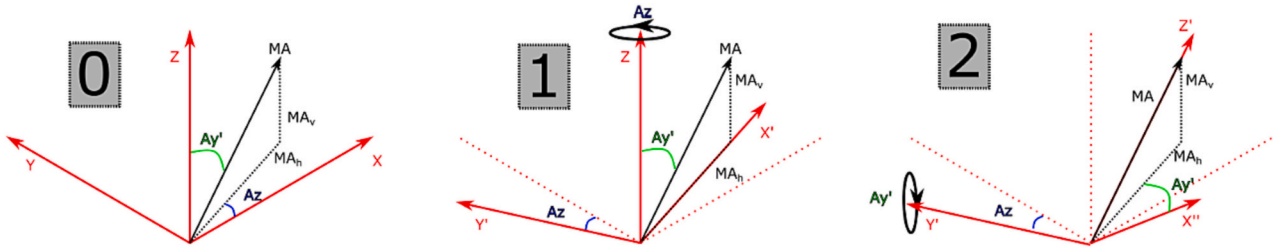

Fig. 11. Obtention of the bounding box axis system. MA – main axis. A – rotation angle around a given axis.

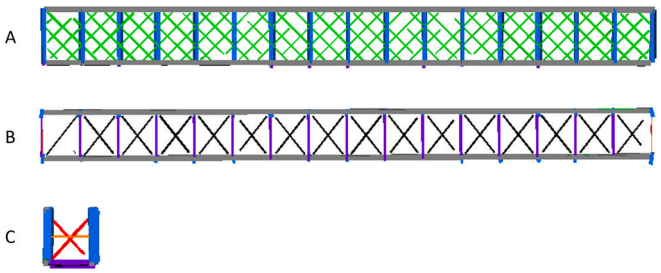

Fig. 12. First state truss. A - Front view. B - Bottom view. C - Side view.

following steps are able to correct the lack of data.

3.1.2. *Second step – Bounding box extension*

The aim of this step is to move the truss from state **A** to **B** of Fig. 10. At the end of the previous step, the members were not interconnected. In order to solve this scenario and complete the geometry of the truss, the bounding boxes of the members are extended in the direction of their main axis. This extension is the reason behind the criterion "*The labelled set of points must be aligned with the direction of the member they belong to*" mentioned in Section 2.1. Through this restriction, the bounding boxes are aligned with the real direction of the member and, by extending the bounding box in that direction, they are able to represent the entirety of the member from just a segment.

Nevertheless, it is necessary to set the boundary conditions that will delimit their extension. Such boundaries can be described via other members of the truss, called delimiting members or delimiters. In this step, the delimiting members for extension are the chords of the truss, since almost every member has both ends connected to chords. Therefore, the task at hand is to correctly identify which chords are to be used as delimiters. This selection is based on the classification of the member. For example, the members of the bottom face use the two chords with the lowest Z value in their centre.

This procedure is applicable to all members, with the exception of those belonging to the interior faces of the truss. Interior lateral braces can connect to any of the four chords. Therefore, the delimiting chord selection for interior lateral braces is based on the intersection between the main axis of the brace, and the main axis of the delimiting candidate chord. In the case of interior braces, their boundary conditions for extension are defined by two vertical posts, so the intersection check must be performed for the vertical posts, instead of chords.

After the delimiting members have been selected, the extension of the member can be done. Fig. 13 presents the extension scenario using a vertical post and a chord as example. First, the intersection points between the main axis of the member ($\mathbf{AX_M}$) and the axis of the delimiting members ($\mathbf{AX_{ch1}}$ and $\mathbf{AX_{ch2}}$) are calculated ($\mathbf{Int_{M\text{-}ch1}}$ and $\mathbf{Int_{M\text{-}ch2}}$). Then, the distance between the centre of the bounding box of the member ($\mathbf{C_M}$) and each of the intersection points is obtained (**E1** and **E2**). Using these distances, the extent of the bounding box along its main axis is recalculated (**E'**). This also prompts a correction of its centre ($\mathbf{C_M'}$), as the extension might not be symmetrical.

It must be noted that the main axis vectors of two bounding boxes representing members of the same type (e.g., diagonals) might have the same orientation but different sense. In Fig. 13 this would be seen as the yellow arrow next to $\mathbf{C_M}$ to be pointing downwards instead of upwards. The orientation is the same, but the sense is different. This possibility must be taken into account to properly calculate the new bounding box extent (**E'**). Once these values are obtained, both the bounding box and the placement are redefined using the new parameters.

If this process is repeated for the rest of the members, the truss reaches the second state (**B**) of Fig. 10. Fig. 14 shows what the IFC model would look like if no further processing was done and the model was to be generated at this point. As it can be seen, the members are now interconnected and give a better representation of the truss geometry. However, some of the members were extended out-of-bounds, and are to be corrected in the next step.

3.1.3. *Third step – Out-of-bounds correction*

The objective of this step is to move the truss from state **B** to **C** as per Fig. 10. This is achieved by correcting the overextended members

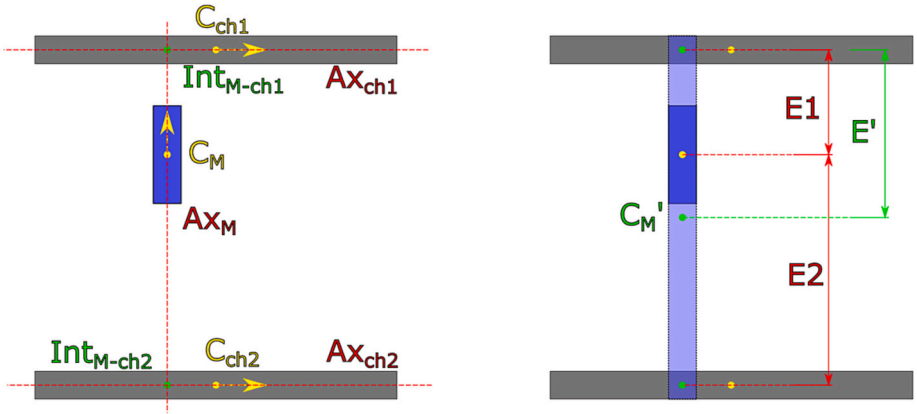

Fig. 13. Extension scenario.

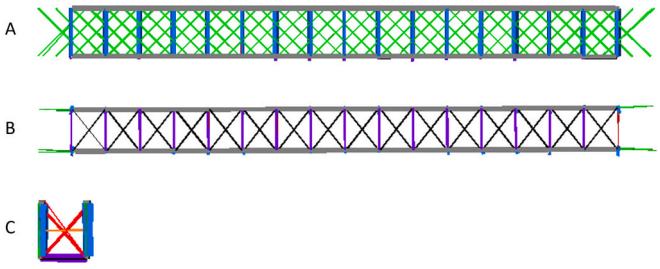

Fig. 14. Second state truss. A - Front view. B - Bottom view. C - Side view.

through the truncation of their bounding box using a newly defined delimiting member.

In the previous step, the delimiting members were set to chords, with the exception of interior braces. However, this is not the case for all diagonal members (diagonals and bottom lateral braces, depending on the face). Fig. 15 presents an example on how some of these members only connect to one chord, while the others connect to one of the two straight members situated at opposite ends of the truss face (vertical post or strut, depending on the face). Since the previous step forced the extension towards chords, the diagonals were extended outside of the boundaries of their respective face, as shown in Fig. 14.

This problem is easier to tackle if each face is analysed individually. To identify which diagonal members need correction, they are sorted using their bounding box centre coordinates. In the example of Fig. 15, this would be from left to right. Then, they are looped through both in ascending (left to right) and descending order (right to left), checking for intersection with the respective first or last straight members. By performing the loop in this manner, the need to check every diagonal is removed since once a diagonal which does not intersect is found, the rest will not intersect either. Through this process, the bounding box of the diagonals that do intersect with a straight member are truncated at the point of intersection, effectively correcting their geometry. Once this process has been completed for all members and faces, the truss has reached state **C** as seen in Fig. 10. At this point, the geometrical aspect of the truss model has been completed, whose result can be seen in Fig. 16.

3.1.4. Fourth step – Element interconnection

The goal of this section is to obtain the final state of the model from state **C**, as seen in Fig. 10. This final state is the complete IFC-compliant model of the truss, including the different relationships between its members.

The IFC generation of the truss is done following the entities presented in Section 2.3. For the most part, this model has already been shown in the previous step, as state **C** has the same geometry as the final state of the truss. However, an IFC model goes further than the geometry. It can contain information about the semantics of the object (e.g., name and description), relationships between entities, materials, and other properties (e.g., maintenance history, thermal properties, and costs).

In this work, the only source of data is the partially instance-segmented point cloud described in Section 2.1. Therefore, the truss

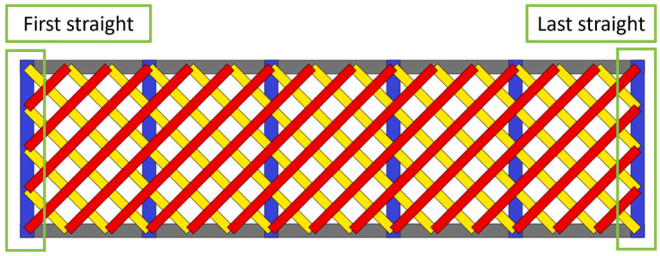

Fig. 15. Example of straight and diagonal members.

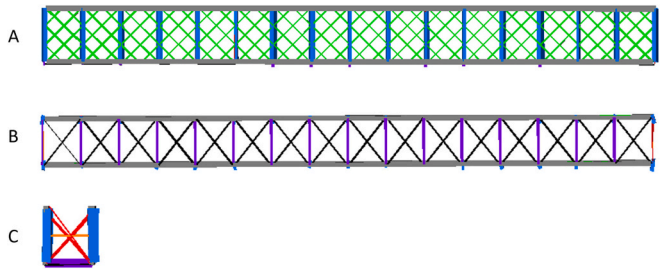

Fig. 16. Third state truss. A - Front view. B - Bottom view. C - Side view.

model contains some semantics (bridge name, brief description, etc.), its geometry, and the topology relationships that can be extracted from such data (aggregation and connection). If additional information was to be present in the form of documentation or other accessible sources, it would be possible to include it in the model.

The truss as a whole was generated as a single element assembly which only included the semantics of the truss and a placement. Without linking the truss with its members, it does not have a representation. However, when the *IfcRelAggregates* relationship defines the truss as the aggregation of all the truss members, both the truss and the members gain new information. On one side, the truss representation is now defined as the union of all the representation of its members. On the other, each member has gained context in the project and a point of reference for their placement. This implies that if the truss assembly is moved, all the members follow it, since they are placed relative to it. Therefore, if the truss is to be placed in a full bridge model which uses a different placement system or reference, such as the use of an *IfcAlignment* for linear placement, only the truss assembly placement needs to be redefined.

The connections between members are a key factor in the structural analysis of a truss, as they define the different points where the loads are distributed to the different elements. For two elements to be connected, they must touch one another. Therefore, a fitting procedure is the use of bounding box collisions to determine which elements are candidates to be linked together. The collision of bounding boxes is done through a mathematical C++ engine, called GeometricToolsEngine [27], that allows to check whether two oriented bounding boxes intersect one another.

Using this engine, all the members are automatically checked for collision with other members. To speed up the process, for every member being examined, all impossible options are not sent to the engine for collision detection. This is done by checking the coordinate ranges of the vertices of both bounding boxes. If the coordinate ranges do not overlap, a collision is not possible. Fig. 17 presents this filtering in a 2D scenario with axis-aligned boxes. Even if two rectangles overlap in the Y axis, if they do not overlap in the X as well, they will never collide.

At this point, it is possible to relate any member to the members it collides to. However, another rule is added due to the nature of the truss. The members are only fixed together at the chords and straight members. On the other hand, diagonals often touch each other but are not fixed to one another. Therefore, all collisions that are not produced by two diagonals are used to set a *IfcRelConnectsElements* relationship between the colliding members.

With the inclusion of these relationships, the model has reached the final state and can now be used to obtain a structural graph, which will be explained in Section 3.2.

3.2. Structural graph

The objective of this section is to cover the generation of the structural graph of the truss, represented by the “Structural graph generation” in Fig. 1. A graph is a data structure of nodes and edges, where the edges link nodes to each other. In this work, the nodes are used to

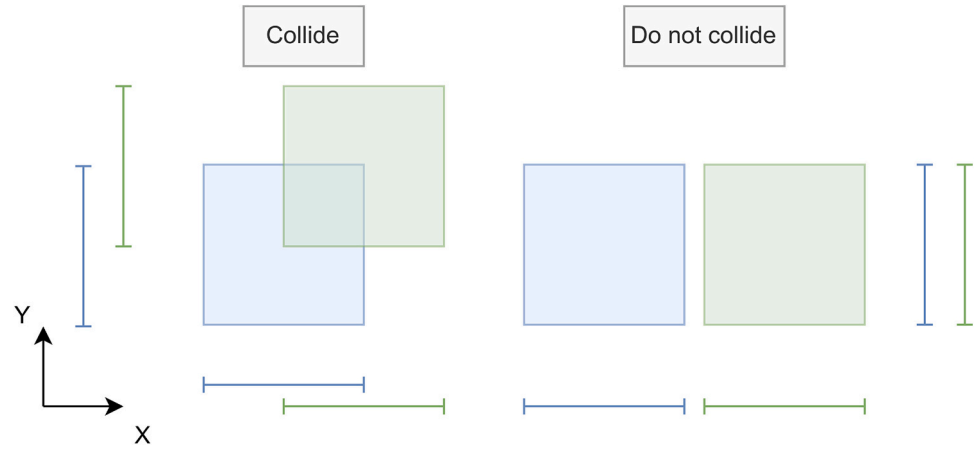

Fig. 17. Coordinate overlapping for collision detection.

represent the connexion points between members, while the edges are used to represent the members themselves. The flexibility of this representation is a great fit for structural analysis, since is possible to include any information of interest to these entities. For instance, a property setting the node as fixed, a distributed load being applied to an edge, or member properties such as material and profile.

This process is performed after the entire IFC model has been generated following the procedure explained in Section 3.1. The reason behind this is that the IFC model itself can be used as an import in some structural analysis software. Therefore, the obtention of the structural graph is treated as an optional step in the process and, if adapted, could be performed on already existing models. The overall idea is to use the *IfcRelConnectsElements* relationships present in the models to generate nodes that are linked together with edges that represent the members themselves. A simplified example of this can be seen in Fig. 18. Following this, this section is further divided between the construction of the graph (Section 3.2.1) and its export as a file compatible with structural analysis software (Section 3.2.2).

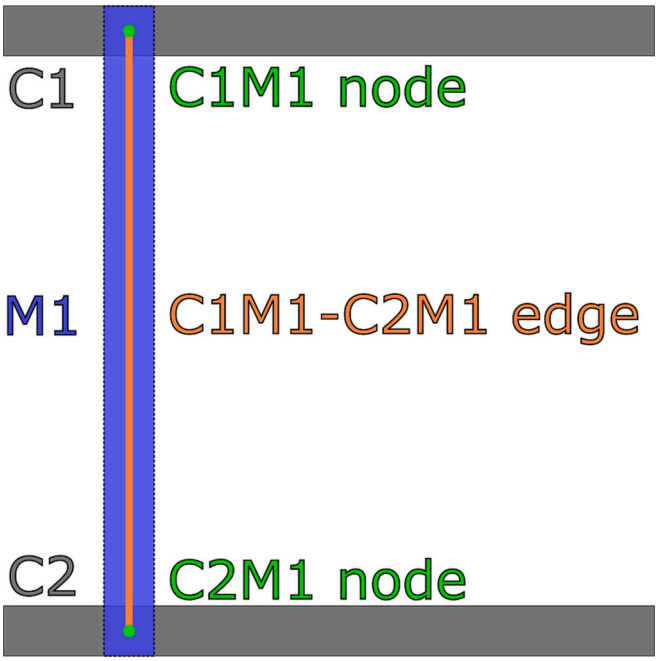

Fig. 18. Node and edge example.

3.2.1. Graph construction

As mentioned, the structural graph is made of nodes and the edges that connect them. In this case, the nodes carry the spatial coordinates of the intersection, while the edges contain information about the members that they represent, such as profile or material. Since the edges are defined through a start and an end node, the first step is to obtain the nodes themselves.

To do so, the connection relationships described in Section 3.1.4 are analysed. By fetching each of the connection relationships and calculating the intersection between the main axis of the bounding boxes of the members, the nodes are obtained. However, due to the variability of the point cloud and therefore, the bounding boxes, the nodes which are theoretically the same, appear in different positions, as seen in Fig. 19A. To solve this situation, the nodes are clustered using a Density-Based Spatial Clustering of Applications with Noise (DBSCAN [29]) algorithm in order to obtain a single node for such occurrences, as shown in Fig. 19B. The merged node will carry all the information of the previous nodes. This includes which elements are related in its creation. For instance, in Fig. 19A, one node might be obtained from the collision of a diagonal and the chord, while other from the vertical post and the chord. The merged node is therefore related to the diagonal, the chord, and the vertical post.

The following step is to generate the edges connecting the nodes. Since the nodes carry the information of the members they are related to, the approach taken is to evaluate the node set of each member individually. Here, the set refers to every node related to the member, which implies that the node is situated on the member itself. This aids in the definition of rules for processing since the nodes are almost aligned, as seen in Fig. 20.

Each node in the set is evaluated, obtaining its two closest nodes of the set as candidates for the definition of an edge. In the fragment shown in Fig. 20, this would mean that the chosen nodes for node 2 are node 1 and node 3. These edge candidates can be valid, inversed, or complex. These types of candidates are represented in Fig. 21 and are defined as follows:

- **Valid**. The candidate correctly defines a path between two nodes, without crossing any other node, and has not been included yet.
- **Inversed**. The candidate has already been included in its inverse sense. The edges are non-directed, meaning that the edge that connects node 1 to 2 is the same that connects 2 to 1.
- **Complex**. The path of the edge crosses an intermediate node. Therefore, the candidate can be expressed as the sum of two edges. In the example of Fig. 21, the two closest nodes to node 1 are node 2 and node 3. However, the edge from node 1 to node 3 can be expressed as the sum of the edge from node 1 to 2 and the edge from node 2 to 3.

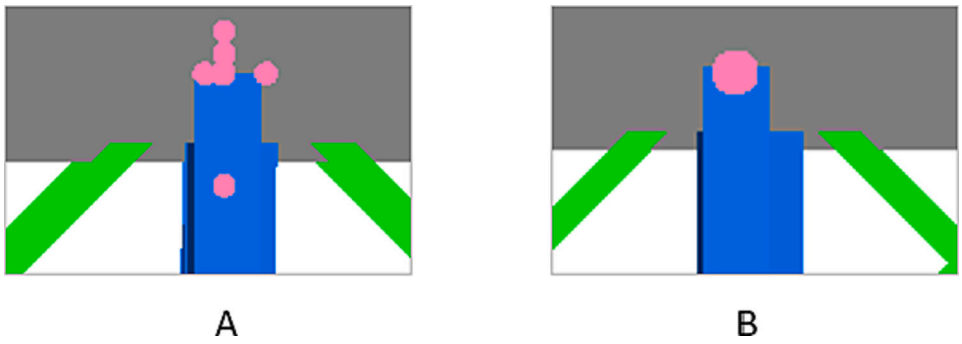

Fig. 19. Collision nodes (Pink). A - Unmerged nodes. B - Merged node. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

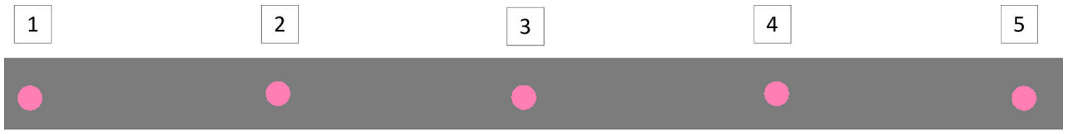

Fig. 20. Fragment of a chord node set.

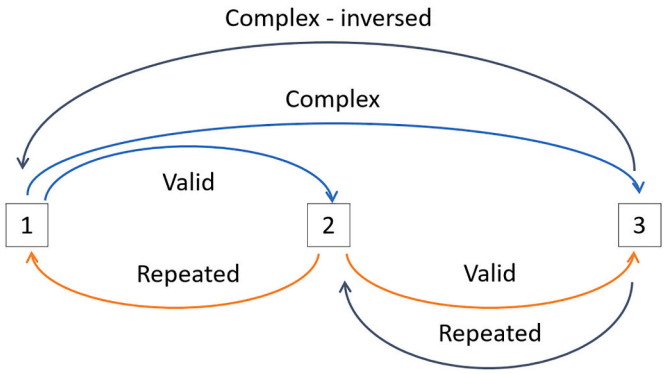

Fig. 21. Example of types of edge candidates.

The node set is indexed and looped through in an ordered manner. Therefore, it allows the use of the indexes themselves to analyse the edges. For a given node under study, its two neighbours are checked. If the edge that would link the studied node to one of its neighbours, or its inverse, has not been marked as an edge candidate, it is marked. In the example of Fig. 21, the following would occur:

1. **Node 1**. Neighbours: Node 2 and Node 3
   a. Edge 1–2. It has not been marked, do so.
   b. Edge 1–3. It has not been marked, do so.
2. **Node 2**. Neighbours: Node 1 and Node 3
   a. Edge 2–1. Edge 1–2 has already been marked, skip.
   b. Edge 2–3. It has not been marked, do so.
3. **Node 3**. Neighbours: Node 1 and Node 2
   a. Edge 3–1. Edge 1–3 has already been marked, skip.
   b. Edge 3–2. Edge 2–3 has already been marked, skip.

In this first part of the process, edges 1–2, 1–3, 2–3 has been marked as possible candidates. However, edge 1–3 is a complex edge, as it can be expressed by the sum of edge 1–2 and 2–3. One possible solution is to calculate a member-specific length between nodes and use it to filter edges that are above that threshold. However, this would only work in a perfect scenario. If there are any missing members due to the point cloud acquisition or because it is the actual state of the truss, this method would fail. Therefore, all edge candidates must be examined to ensure that they cannot be broken down into simpler candidates. Since only two neighbouring nodes are explored per node in the set, and the inversed edges have already been eliminated, the number of checks needed is greatly reduced.

This evaluation is done in the form of paths branching from the starting node of the examined edge. The rules for the path-finding are the following: (i) To not use the examined edge; (ii) To not go backwards (using the inverse of an already taken path); and (iii) To not create a path of higher length than the examined edge. If it is possible to reach the end node in two steps without breaking the aforementioned conditions, the examined edge is complex and therefore removed. This process can be seen in Fig. 22 for the edges 1–2 and 1–3 of the previous example. Edge 2–3 is analogous to 1–2 so is therefore not presented.

This process is then repeated for each of the node sets, obtaining all edges of the graph.

3.2.2. *Export*

At this point, both the geometrical IFC model and the structural graph have been generated. Therefore, the final action is their export towards a structural analysis software. The software chosen is DIANA [30], since it provides a clear way to input the structural graph as a text file that can be automatically generated from the graph characteristics. Furthermore, DIANA also supports the import of the geometrical IFC model directly and the use of Python commands to generate variables. As such, there are three available export options in the developed software:

- **Geometrical export**. The IFC file itself. Albeit this removes the need to calculate the structural graph described through Section 3.2, it also requires the user to manually set the analysis conditions. Therefore, this option is best fit whenever the user is experienced and wants to perform a thorough analysis, since it removes the monotonous task of manually setting the geometry.
- **Text file graph export**. Generates a DIANA-compatible text file that describes the nodes and edges and their properties. It is also possible to define a mesh-like sectioning of the edges. This option is best fit when applying an automated pipeline analysis, meaning that the whole procedure is expected to be performed as automatically as possible, from point cloud import to structural analysis.
- **Python command export**. Similarly to the text export, it generates a .py file that contains series of variable declarations for the nodes and edges. This is the most minimalistic representation of the graph and

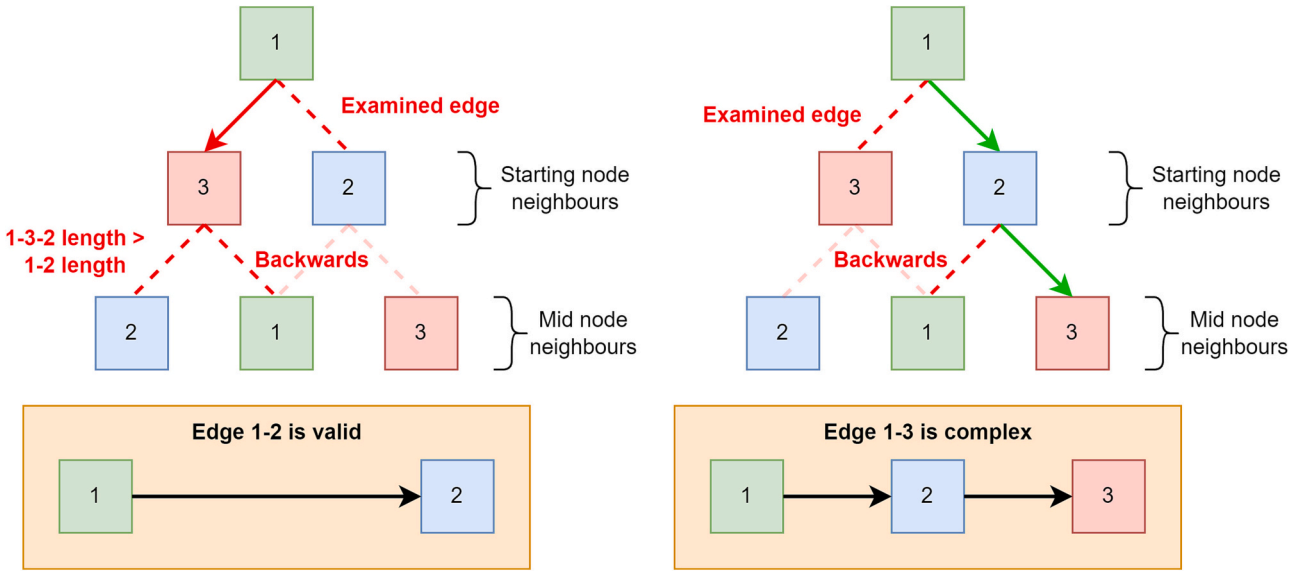

Fig. 22. Complex edge evaluation. Left: Edge 1–2. Right: Edge 1–3.

could be further enriched by adding other commands to set up materials and profiles. The application scenario is equal to the text file export. Therefore, in case of using DIANA, it is simply a matter of preference for the end user. If the structural graph is to be imported in a different software, or further processed using other tools, a Python file might be a better option due to its wide adoption.

4. Results

The methodology explained throughout Section 3 was applied to a partially instance-segmented point cloud of a truss bridge, as described in Section 2.1. The truss contains 272 members, divided amongst the analysed faces (two vertical faces, a horizontal bottom face and seventeen interior faces). These members can be further broken down into their classes as per the nomenclature of Fig. 3: 136 diagonals, 4 chords, 34 vertical posts, 16 struts, 32 bottom lateral braces, 33 interior lateral braces, and 17 interior braces. The software developed in this work uses the 4.1 version of the IFC schema. It was programmed using C# and used the xBIM 5.1.323 toolkit [31] for the creation of the model, along with the GeometricToolsEngine [27] to check whether two bounding boxes intersect each other. For reference, the viewer used to visualize the IFC models, and the one from where the figures of Section 3.1 were extracted, is FZKViewer 6.4 [32].

4.1. Truss model

The truss model generation has been explained in Section 3.1. This process is able to automatically generate an IFC-compliant model of a truss bridge using a partially instance-segmented point cloud as the source of information. Furthermore, it does not only create the model using the provided data as-is, but actively overcomes missing or faulty data. The driving factor behind this result is the use of bounding boxes as

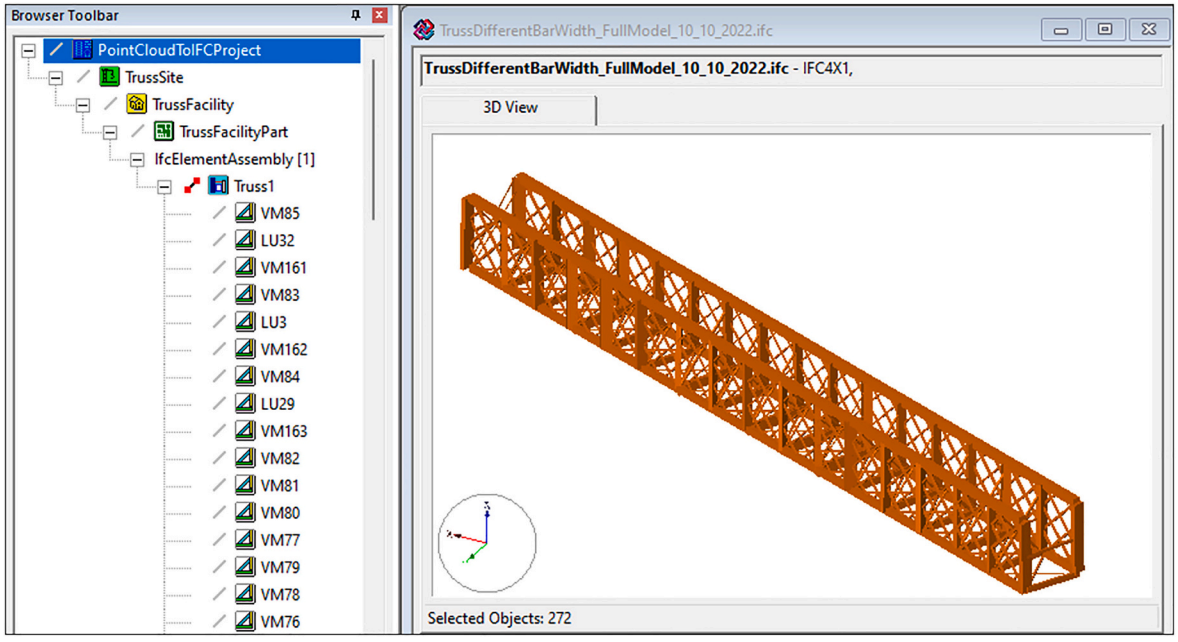

Fig. 23. Full IFC model in FZKViewer.

both a way to fill the possible gaps and to further evolve the model while providing meaningful information using their collisions. The evolution of the model has been shown throughout Section 3.1, presenting the coloured geometry of the model in its first state in Fig. 12, and its final state in Fig. 16. In this section, however, the full IFC model will be presented through different figures taken from the FZKViewer. Fig. 23 presents a global view of the model, including the hierarchy of the project. It must be noted that the colouring scheme for the different members used until now is no longer present. Since this section will cover the final truss model, all members are represented using *IfcMember*, and have the same colour.

As it can be seen, the model is made of 272 objects, in this case *IfcMembers* that are assembled as an *IfcElementAssembly* called "**Truss1**" in the model. Inside the "**PointCloudToIFCProject**" other three entities along with the truss: "**TrussSite**", "**TrussFacility**" and "**TrussFacilityPart**". These are instances of *IfcSpatialStructureElement* and are used to organize a project. If a single entity is selected, its properties can be explored, this includes identification and context information, as well as what kind of geometry and placement it has. Fig. 24 presents the properties of a vertical post with name "**VM77**".

More importantly, the relations tab allows us to explore the different relations to other entities, such as the aggregation relationship with the truss assembly. As "**VM77**" is a vertical post, it has a total of 16 connection relationships to other members. Fig. 25 shows its relations tab, highlighting some of the members connected through these relationships.

4.2. Structural graph

The structural graph construction has been described in Section 3.2. It started off with the node assessment through the bounding box collisions, represented by the *IfcRelConnectsElements* relationships. Then, the edges were generated member wise, targeting their node sets in a way that simulates their alignment.

The software developed outputs two files. The first one is always an IFC file (.ifc) that describes the truss model. The second is a file that contains the information of the structural graph. As mentioned in Section 3.2.2, this graph information can be expressed through a text file that follows the DIANA guidelines for imports, or through a Python file that contains instructions to generate the nodes and edges as variables. Whichever option is chosen, the amount of data that is possible to be included in both formats is the same.

To present these exports, the following figures represent all three types once imported into DIANA. First, Fig. 26 is the direct import of the IFC file. Then, Fig. 27 is the text file import, which also includes a simple mesh. Finally, Fig. 28 is the python import with only nodes and edges.

As shown, the graph export, either through text or Python instructions, contains much simpler and direct information. This is an important factor to consider when thinking about scaling the model to include more elements and details, or include it into an automated structural analysis pipeline.

5. Discussion

The results described in Section 4 show that the proposed methodology is able to fulfil the objective and contributions of this work, which were presented in Section 1. The purpose of this section is to address the shortcomings of the paths taken to achieve those contributions.

As first contribution, the software developed outputs an IFC-compliant file that contains the truss information model, including the connection relationships between its members. The generation of such model has been explained in Section 3.1.4, with the final IFC model being shown in Section 4.1. The information model follows the IFC 4.1 schema instead of IFC 4.3, which is under ISO DIS voting at the time of

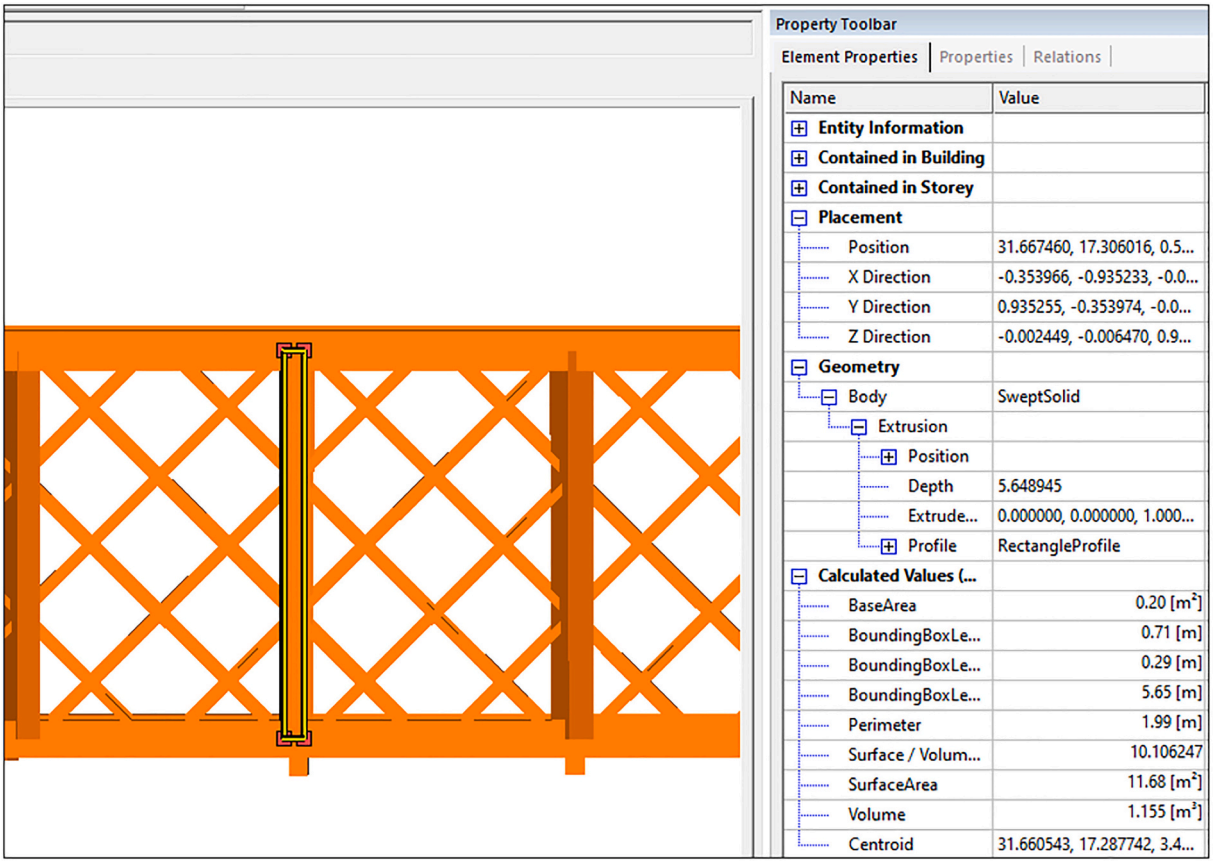

Fig. 24. VM77 - Element properties tab.

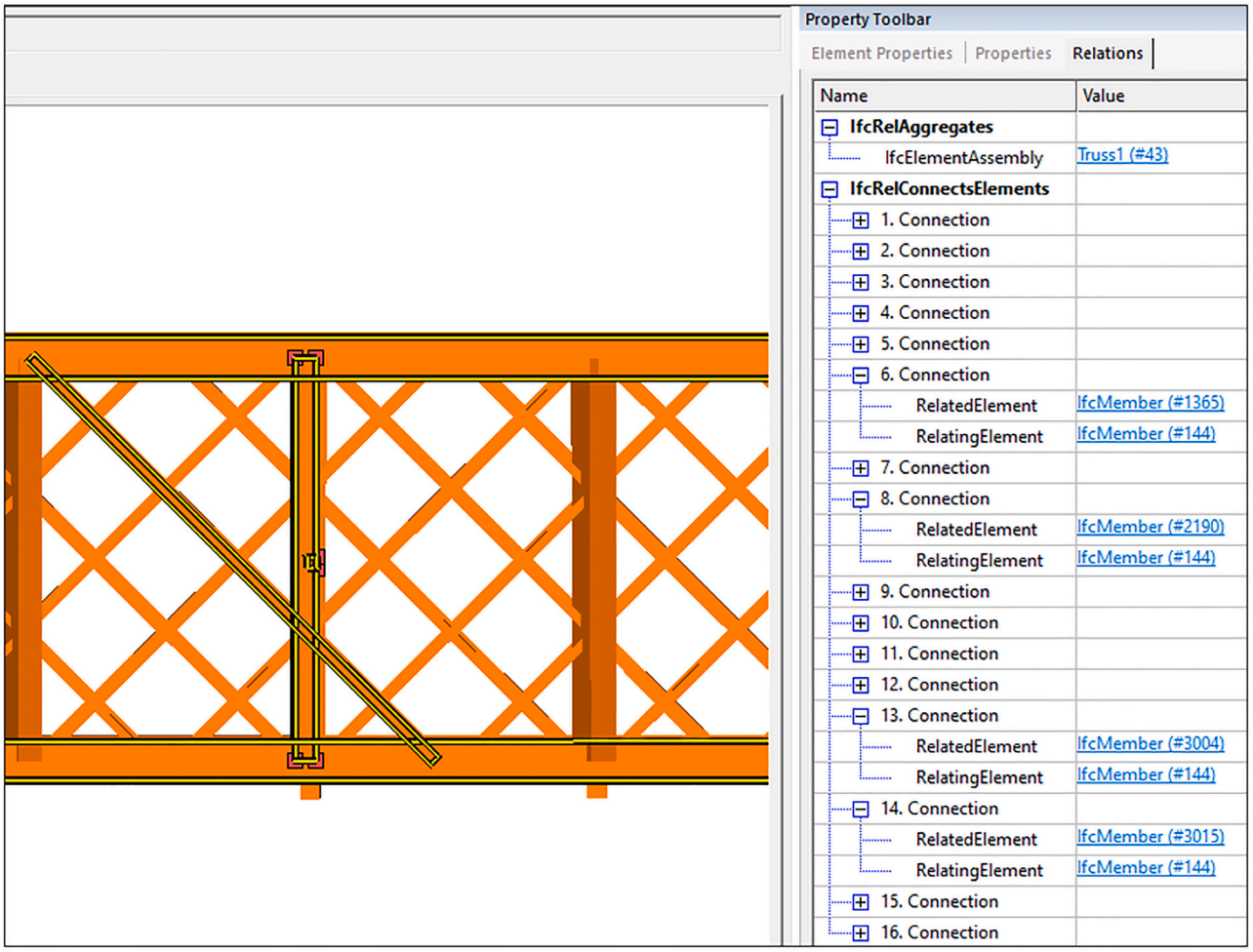

Fig. 25. VM77 - Relations tab.

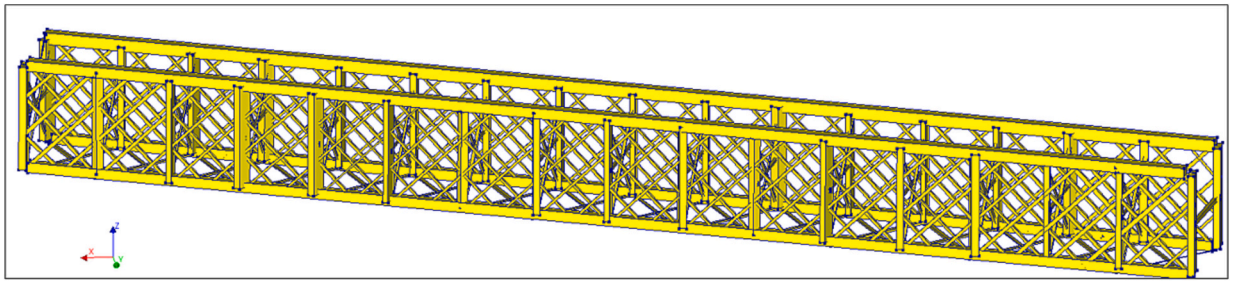

Fig. 26. Direct .ifc import - DIANA.

writing. Therefore, some IFC entities related to infrastructure are non-existing or have a different name. For instance, the"**TrussFacility**" mentioned in Section 4.1, which works as a tool to better organize a project, is expressed with *IfcBuilding* instead of the *IfcFacility* of IFC 4.3.

As second contribution, a structural graph representing the truss geometry is generated and exported. This process has been detailed in Section 3.2, while the visualization of the imported files was presented in Section 4.2. The construction of this graph relies on the previously introduced *IfcRelConnectsElements* relationship between members whose bounding boxes collide with one another. Through this work, two ways of working with the bounding box were used. On one side, the intersection between the main axis of two bounding boxes, used in Section 3.1.2–3.1.3. On the other, checking if two bounding boxes collide to one another, used in Section 3.1.4. The issue with the former is that two lines almost never intersect to one another in a 3D scenario. Therefore, the closest point of one line with respect to the other is used instead. The problem with the latter, on the other hand, is that it only checks whether two bounding boxes are colliding, it does not provide the shape of intersection of the bounding boxes. Therefore, if geometrical data is required, additional operations are needed once it is known that they collide.

As third contribution, the methodology is able to overcome the partial segmentation. This is achieved through the use of bounding boxes and their modification according to the truss frame of reference, the chords. This has been described in Sections 3.1.1–3.1.3. Since a partially instance-segmented point cloud is the only source of data, there are some issues and challenges that are to be addressed. Infrastructure point clouds are acquired outdoors and from large distances.

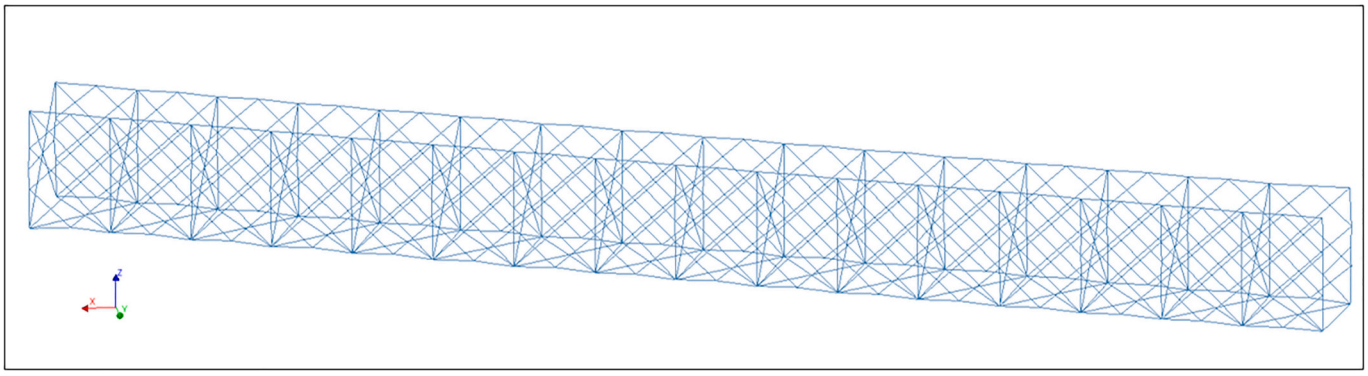

Fig. 27. Text graph import - DIANA.

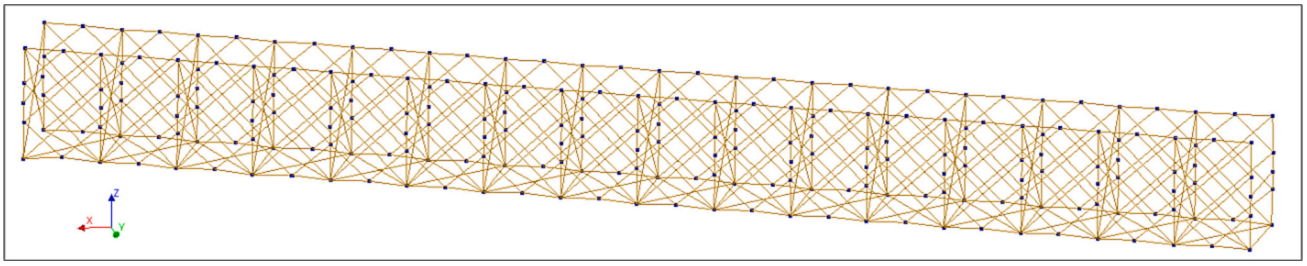

Fig. 28. Python graph import - DIANA.

Furthermore, occlusions from vegetation, obstacles, or the asset itself might be present, as well as unfavourable weather conditions. These aspects often result in point clouds with suboptimal quality when compared to indoor point clouds. As the proposed methodology aims to operate in an automatic manner, any issues with the point cloud are translated into the model itself. For instance, some members of the same type and face present slightly different rotations and positions to one another, which is not the case in the real truss. This is visible in the front and bottom views of the final truss geometry, in Fig. 16. Another example is the profile of the members. It was not possible to extract the profile measurements or shapes from the point cloud, which in turn resulted in the use of rectangular profiles that took advantage of the bounding box measures. Also, as stated in Section 2.1, the top horizontal face was deemed unfit for analysis and excluded from the model. Unfortunately, the point cloud quality did not allow for its inclusion. The occlusions from all the other members, combined with existing noise and proximity of the girder, made the top section barely recognizable with the human eye and was therefore excluded from being processed. Nevertheless, if included as input, the software is ready to correct the top face members in the same manner as the others. This would increase the quality of the model, completing the full truss and improving the accuracy of any possible analysis performed on it.

6. Conclusions

This work represents the second part of a fully automated pipeline that transforms a raw point cloud of a bridge truss into an IFC-compliant model and a structural graph. In the case presented in this paper, a total of 272 members were modelled, connected, and processed into a structural graph that was later exported into DIANA. While the structural analysis itself falls outside of the scope of this paper, the methodology presents the capabilities of the pipeline that is being developed. Nevertheless, there are still areas that can be improved. The graph construction, albeit quite generalized and abstract, is still mainly designed for truss members. The underlying bounding box methodology is a level higher in abstraction, as it uses the concept of general physical elements, instead of truss members specifically. Therefore, as the program develops, and more types of trusses and elements are included, some of these aspects are bound to change in order to account for the flexibility required for such a task. For instance, if piers were to be added, they could be used as the frame of reference in order to also correct the chord members. It would also be possible to use the connection with the piers to set up certain nodes as anchor points. Nevertheless, the approach presented is focused on the truss, which is one of the most geometrically complex elements of a truss bridge. If a different category of elements were to be added, such as elements of the road situated above the bridge (e.g., guardrails), its inclusion in the model would follow the approach taken by the authors in a previous research [33]. Also, as mentioned in the discussion in Section 5, the point cloud introduced certain issues, such as the slight variations in position and orientation of the members, as well as no information about their profile types. The positioning and orientation could be solved by analysing the entire truss to obtain the dominant orientation vectors, as well as the distance between the members, since these structures usually follow a pattern. However, setting this kind of restrictions would significantly reduce the abstraction capability of the methodology, and might encounter difficulties when applied in other scenarios. In the case of the member profiles, the input of an additional document containing the member types could be used to determine the overall shape of each member (e.g., IPE), while their bounding box dimensions could be used to determine their size (e.g., IPE220). As for the IFC model generation, new iterations of the software developed could target IFC 4.3 instead IFC 4.1 if the programming libraries and viewers adopt it. Nevertheless, the key component of this work, the bounding boxes, is unaltered by the change. This is because the IFC formatting is performed at the last stage of the process. Therefore, if the schema or its nomenclature changes, only a small fraction of the developed software is to be adapted. The mentioned bounding boxes were used in two different manners throughout the work. The first being main axis intersection, and the second being the check of collision between two bounding boxes. The main axis intersection performs very well when the members are expected to intersect, such as diagonals with chords. The bounding box

collision application is broader and more general, allowing to check collisions with any entity that has a bounding box, which is ideal for creation of *IfcRelConnectsElements* relationships.

The authors believe that the evolution of IFC towards the infrastructure domain, coupled with the capabilities of point clouds as geometrical source of information, will bring new developments for infrastructure information models. The results presented in this work are promising and set the basis for future work on not only on trusses, but on other types of structures.

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 769255. This work has been partially supported by the Spanish Ministry of Science and Innovation through the PONT3 project Ref. PID2021-124236OB-C33. This work has been partially supported by the University of Vigo through the human resources grant: "Axudas para a contratación de personal investigador predoutoral en formacion da Universidade De Vigo 2021" (PREUVIGO-21) and by the Spanish Ministry of Science and Innovation through the grant FJC2020–046370-I and the grant RYC2021–033560-I.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This document reflects only the views of the authors. Neither the Innovation and Networks Executive Agency (INEA) nor the European Commission is in any way responsible for any use that may be made of the information it contains.

References

[1] J.J. Magoua, F. Wang, N. Li, High level architecture-based framework for modeling interdependent critical infrastructure systems, Simul. Modell. Pract. Theory. 118 (2022), https://doi.org/10.1016/J.SIMPAT.2022.102529.
[2] M. Ouyang, Review on modeling and simulation of interdependent critical infrastructure systems, Reliab. Eng. Syst. Saf. 121 (2014) 43–60, https://doi.org/10.1016/J.RESS.2013.06.040.
[3] A. Boin, A. McConnell, Preparing for critical infrastructure breakdowns: the limits of crisis management and the need for resilience, J. Conting. Crisis Manag. 15 (2007) 50–59, https://doi.org/10.1111/J.1468-5973.2007.00504.X.
[4] Keeping European Bridges Safe, (n.d.). https://joint-research-centre.ec.europa.eu/jrc-news/keeping-european-bridges-safe-2019-04-05_en (accessed June 28, 2022).
[5] A. Costin, A. Adibfar, H. Hu, S.S. Chen, Building Information Modeling (BIM) for transportation infrastructure – Literature review, applications, challenges, and recommendations, Autom. Constr. 94 (2018) 257–281, https://doi.org/10.1016/J.AUTCON.2018.07.001.
[6] A. Borrmann, M. König, C. Koch, J. Beetz, Building Information Modeling: Why? What? How?, Building Information Modeling: Technology Foundations and Industry Practice, 2018, pp. 1–24, https://doi.org/10.1007/978-3-319-92862-3_1.
[7] U. Isikdag, G. Aouad, J. Underwood, S. Wu, in: Building information models: a review on storage and exchange mechanisms, in: Proceedings of 24th W78 Conference: Bringing ITC Knowledge to Work, Maribor, 2007, pp. 135–144 (accessed February 8, 2023), http://itc.scix.net/paper/w78_2007_97.
[8] M.P. Gallaher, A.C. O'connor, J.L. Dettbarn, L.T. Gilday, Cost analysis of inadequate interoperability in the U.S. Capital Facil. Indus., (n.d.). doi:https://doi.org/10.6028/NIST.GCR.04-867.
[9] C. Preidel, A. Borrmann, H. Mattern, M. König, S.E. Schapke, Common Data Environment, Building Information Modeling: Technology Foundations and Industry Practice, 2018, pp. 279–291, https://doi.org/10.1007/978-3-319-92862-3_15.
[10] buildingSMART - The International Home of BIM, (n.d.). https://www.buildingsmart.org/ (accessed June 28, 2022).
[11] IFC Release Notes - buildingSMART Technical, (n.d.). https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/ifc-release-notes/ (accessed June 28, 2022).
[12] ISO - ISO 16739-1:2018 - Industry Foundation Classes (IFC) for Data Sharing in the Construction and Facility Management Industries — Part 1: Data schema, (n.d.). https://www.iso.org/standard/70303.html (accessed September 15, 2022).
[13] IFC Schema Specifications - buildingSMART Technical, (n.d.). https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/ (accessed December 13, 2022).
[14] B. Koo, R. Jung, Y. Yu, I. Kim, A geometric deep learning approach for checking element-to-entity mappings in infrastructure building information models, J. Comput. Design Eng. 8 (2021) 239–250, https://doi.org/10.1093/JCDE/QWAA075.
[15] T.H. Kwon, S.I. Park, Y.H. Jang, S.H. Lee, Design of railway track model with three-dimensional alignment based on extended industry foundation classes, Appl. Sci. 10 (2020) 3649, https://doi.org/10.3390/APP10103649.
[16] G. Bariczová, J. Erdélyi, R. Honti, L. Tomek, Wall structure geometry verification using TLS data and BIM model, Appl. Sci. 11 (2021) 11804, https://doi.org/10.3390/APP112411804.
[17] L. Barazzetti, M. Previtali, M. Scaioni, Roads detection and parametrization in integrated BIM-GIS using LiDAR, Infrastructures 5 (2020) 55, https://doi.org/10.3390/INFRASTRUCTURES5070055.
[18] M.R.M.F. Ariyachandra, I. Brilakis, Detection of railway masts in airborne LiDAR data, J. Constr. Eng. Manag. 146 (2020) 04020105, https://doi.org/10.1061/(ASCE)CO.1943-7862.0001894.
[19] M. Soilán, A. Sánchez-Rodríguez, P. del Río-Barral, C. Perez-Collazo, P. Arias, B. Riveiro, Review of laser scanning technologies and their applications for road and railway infrastructure monitoring, Infrastructures 4 (2019) 58, https://doi.org/10.3390/INFRASTRUCTURES4040058.
[20] R. Wang, J. Peethambaran, D. Chen, LiDAR point clouds to 3-D urban models: a review, IEEE J. Select. Top. Appl. Earth Observ. Remote Sens. 11 (2018) 606–627, https://doi.org/10.1109/JSTARS.2017.2781132.
[21] S. Gargoum, K. El-Basyouny, Automated extraction of road features using LiDAR data: A review of LiDAR applications in transportation, in: 2017 4th International Conference on Transportation Information and Safety, ICTIS 2017 - Proceedings, 2017, pp. 563–574, https://doi.org/10.1109/ICTIS.2017.8047822.
[22] L. Ma, Y. Li, J. Li, C. Wang, R. Wang, M.A. Chapman, Mobile laser scanned point-clouds for road object detection and extraction: a review, Remote Sens. 10 (2018) 1531, https://doi.org/10.3390/RS10101531.
[23] A. Sánchez-Rodríguez, S. Esser, J. Abualdenien, A. Borrmann, B. Riveiro, From point cloud to IFC: a masonry arch bridge case study, EG-ICE 2020 Works. Intell. Comput. Eng. Proceed. (2020) pp. 422–431.
[24] D. Isailović, V. Stojanovic, M. Trapp, R. Richter, R. Hajdin, J. Döllner, Bridge damage: detection, IFC-based semantic enrichment and visualization, Autom. Constr. 112 (2020), 103088, https://doi.org/10.1016/J.AUTCON.2020.103088.
[25] D. Prati, G. Zuppella, G. Mochi, L. Guardigli, R. Gulli, wooden trusses reconstruction and analysis through parametric 3d modeling, ISPRS Ann. Photogram. Remote Sens. Spat. Inform. Sci. 42 (2019) 623–629, https://doi.org/10.5194/ISPRS-ARCHIVES-XLII-2-W9-623-2019.
[26] J. Hermida, M. Cabaleiro, B. Riveiro, J.C. Caamaño, Two-dimensional models of variable inertia from LiDAR data for structural analysis of timber trusses, Constr. Build. Mater. 231 (2020), 117072, https://doi.org/10.1016/J.CONBUILDMAT.2019.117072.
[27] Geometric Tools, (n.d.). https://www.geometrictools.com/index.html (accessed September 29, 2022).
[28] buildingSMART, IFC 4.1 Documentation, (n.d.). https://standards.buildingsmart.org/IFC/RELEASE/IFC4_1/FINAL/HTML/ (accessed October 13, 2022).
[29] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 226–231. https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf (accessed January 17, 2023).
[30] DIANA FEA, (n.d.). https://dianafea.com/ (accessed October 26, 2022).
[31] S. Lockley, C. Benghi, M. Černý, Xbim., Essentials: a library for interoperable building information applications, J. Open Source Softw. 2 (2017) 473, https://doi.org/10.21105/joss.00473.
[32] KIT - IAI - Downloads - FZKViewer, (n.d.). https://www.iai.kit.edu/english/1648.php (accessed July 11, 2022).
[33] A. Justo, M. Soilán, A. Sánchez-Rodríguez, B. Riveiro, Scan-to-BIM for the infrastructure domain: generation of IFC-compliant models of road infrastructure assets and semantics using 3D point cloud data, Autom. Constr. 127 (2021), 103703, https://doi.org/10.1016/j.autcon.2021.103703.