Bridge Damage: Detection, IFC-Based Semantic Enrichment and Visualization

Dušan Isailović*[1], Vladeta Stojanovic*[2], Matthias Trapp[2], Rico Richter[2], Rade Hajdin[1] and Jürgen Döllner[2]

[1]*Department for construction project management, Faculty of Civil Engineering, University of Belgrade, Bul. kralja Aleksandra 73, 11120 Belgrade, Serbia*

[2]*Hasso Plattner Institute, Faculty of Digital Engineering, Computer Graphics Systems Group, University of Potsdam, Prof.-Dr.-Helmert-Strasse 2-3, 14482 Potsdam, Germany*

Abstract

Building Information Modeling (BIM) representations of bridges enriched by inspection data will add tremendous value to future Bridge Management Systems (BMS). This paper presents an approach for point cloud-based detection of spalling damage, as well as integrating damage components into a BIM via semantic enrichment of an *as-built* Industry Foundation Classes (IFC) model. An approach for generating the *as-built* BIM, geometric reconstruction of detected damage point clusters and semantic-enrichment of the corresponding IFC model is presented. Multiview-classification is used and evaluated for the detection of spalling damage features. The semantic enrichment of *as-built* IFC models is based on injecting classified and reconstructed damage clusters back into the *as-built* IFC, thus generating an accurate *as-is* IFC model compliant to the BMS inspection requirements.

Keywords: Damage Detection, Building Information Modeling, 3D Point Clouds, Multiview Classification, Bridge Management Systems

*First two authors contributed equally to the presented paper.
Emails: disailovic@grf.bg.ac.rs, vladeta.stojanovic@hpi.de, matthias.trapp@hpi.de, rico.richter@hpi.de, rade.hajdin@grf.bg.ac.rs, juergen.doellner@hpi.de

1. Introduction

The cornerstone of the modern, global society is the affordable transportation of people and goods. According to [1] and [2], more than 90% of inland passenger transport and more than 75% of inland freight transport in the EU in 2015 is carried out using road infrastructure. Proper management of the road infrastructure is the prerequisite for good quality transportation. Being a critical component of the road infrastructure, bridges require special treatment. Management and maintenance of bridges is an important concern in countries that depend heavily on road transport infrastructure to accommodate increasing volumes of commercial transportation as a result of economic growth and globalization [3].

Currently, most countries use highly sophisticated information systems specialized for managing bridges, named Bridge Management Systems (BMSs). A BMS comprises an inventory, inspection and intervention database equipped with analytical tools to perform various predictions (e.g., maintenance cost, bridge condition). Although the rigid structure of a relational database provides high system robustness, the alphanumeric data format in BMS fails in describing geometric information. According to Mirzaei *et al.*, none of the existing BMSs include geometric representation of bridges [4].

The idea of Building Information Modeling (BIM) is a few decades old [5], however, only recent mandates in certain countries promoted it in the construction industry as the most efficient technology to create, store and modify data for the built environment throughout its entire lifecycle [6]. BIM is envisioned as a comprehensive, accurate and up-to-date digital representation of a building. As shown by Sacks *et al.*, such a comprehensive data repository could easily provide a BMS with an inventory of data as well as structural component visualization and enhanced decision making (using the 3D model in order to enhance decision making through visualization) [7].

Inspections are periodical quality assessment procedures. Even though the implementation of these procedures varies among the countries, the basics are common all around the world. Visual inspections are the most frequent and the most cost-effective ones. They can trigger an in-depth investigation, or even maintenance intervention. During the inspection, the inspector examines each element of the bridge, searching for visible damages. Apart from equipment related defects (e.g., bearings, expansion joints), concrete spalling, cracks, and reinforcement corrosion are the most frequent damages on the reinforced concrete bridges.

1.1. Problem Statement

Updated after each inspection, a BIM can be used to represent a current state of the bridge. However, while commercial BIM software is capable of creating 3D bridge models with highly accurate geometry, there is a paucity for IFC compatible software tools for updating model content. The IFC format is a neutral format for the exchange of digital building models. The use of IFC as a BIM standard file format aims to solve the interoperability issues — though the current version of the format does not include provisions for inspection findings by default.

With the advances in deep-learning within the field of AI, sophisticated methods based on computer vision principles can be implemented for detection of potential damage elements contained in the *as-is* representation of structural components of bridges. Specifically, image-based multiview classification can be used to detect potential damage features contained in the point cloud representation of a bridge.

1.2. Research Contributions

The presented research addresses the problem of generating *as-is* IFC model representations of bridges with structural damage features obtained from a point cloud. The proposed solution to this problem is using image-based multiview classification to detect and extract concrete macro damage features. Furthermore, an IFC semantic enrichment framework is proposed to inject the extracted and reconstructed damage features into the *as-is* IFC model. Finally, the proposed damage detection, feature extraction, and semantic enrichment approach are validated in the presented case study. The contributions can be summarized as follows:

1. Image-based, multiview classification is used, where point cluster regions are spatially divided using an *octree* data structure. Each of the octree nodes is used to generate a cubemap representation of the point cluster inside it. These cubemap images are then classified using a retrained Convolutional Neural Network (CNN), and the damage-classified clusters can then be extracted.
2. Geometric reconstruction methods of points from damage-detected clusters are evaluated and discussed.
3. Geometric and semantic enrichment of the IFC model is achieved by injecting the reconstructed 3D meshes representing damaged regions and corresponding BMS catalog-based damage information. The proposed

method uses Constructive Solid Geometry (CSG) Boolean operations to geometrically enrich the IFC geometry elements, which align with corresponding damage regions from the *as-is* point cloud. Damage information (e.g., type, extent, and severity) is structured so that it complies to the BMS data structure.

2. Related Work

2.1. Bridge Point Cloud Inspection

Currently, bridges are inspected mostly manually. The idea of substituting human visual perception with an automated, systematic and quantitative 3D point cloud assessment is currently an intensively investigated topic. This extends to the idea of using Unmanned Aerial Vehicles (UAVs) to acquire point cloud representations for further evaluation. Several commercial UAVs specialized for inspection were developed in the last ten years. Wells and Lovelace analyzed a state of the art hardware and software solutions for photogrammetry-based bridge inspection [8]. Although the improvements of UAVs are remarkable (e.g., protected propellers, multi-directional camera, high-resolution image acquisition, distance lock, additional thermographic camera, etc.), the output data is not post-processed. The UAVs are usually provided with additional software for the 3D scene geometric reconstruction. None of the analyzed software includes damage detection. Instead, the damage is manually detected and modeled as a pinned location with attached photos of the damaged region. Even though such *as-is* point cloud representation can be considered semantically poor, it still significantly decreases inspection costs. Wells and Lovelace further compared traditional and UAV-based inspection in terms of cost and duration, inspecting 12 bridges of different types and sizes [8]. Whereas both approaches took roughly the same time, in the same report, it was claimed that the UAV-based inspection was averagely 40% cheaper than the traditional one.

Damage detection for concrete bridges has been exhaustively investigated in the past two decades. Jahanshahi and Masri developed a state of the art method for extracting an accurate two-dimensional geometry of a concrete crack from the image [9], whereas German *et al.* established a concrete spalling detection method providing the length and depth of the spalling region [10]. None of the image-based damage detection techniques provides the damage location relative to the inspected structure. The latest research

tends to systematize imagery acquisition techniques with damage detection and feature extraction methods into an automated bridge inspection system. Morgenthal *et al.* proposed a conceptual framework for utilizing the state of the art UAV-based bridge inspection techniques [11]. Instead of inspection standardization, the authors suggest defining tasks and assessment criteria for each inspection. After the UAV flight path planning, the authors propose the use of multi-scale crack centerline detection, also proposed by Sironi *et al.* [12], as well as the structural condition assessment by integrating the detected spalling damage into the previously generated finite element model of the bridge. Research by Hüthwohl and Brilakis focused on the image-based classification of concrete surfaces of highway bridges for damage detection, and used morphological operators to highlight the damage in surface textures that are then projected on to the given *as-is* model [13]. Research by Xu and Turkan (2019) propose a framework for implementation and integration of BIM and UAV technologies for bridge maintenance. The proposed framework makes use of an image-based processing technique for detecting concrete cracks, and links this information with a string description field of the bridge IFC representation [14].

The benefits and challenges of using point-cloud data alongside BIM has been researched by Qu and Sun [15] and Tuttas *et al.* [16]. In these studies, the automated generation of an semantically-rich model for further geometric reconstruction to *as-is* BIM models, are noted as the main benefits. These semantically enriched models can further be converted into BIM data or used for other type of data analysis. Additionally, Anil *et al.* state that the use of point clouds alongside the *as-designed* or *as-is* BIM representation allows for the assessment of any conflicting differences [17]. The use of point clouds can benefit the generation of digital documentation of new building features that are added in the post-construction phase [18].

2.2. Multiview Classification

The use of machine learning applications has gained momentum in the last few years, especially in the field of construction automation [19]. The most notable progress is with the use of *deep-learning* methods that rely on the use of 2D or 3D CNNs [20]. CNNs can be trained on 3D geometry or 2D image data to classify elements from the built environment [21]. The increased use of CNNs for other important computer vision solutions has led to the development and release of Google's Inception V3 CNN model and the

TensorFlow API, which allow for more practical implementation and application of deep-learning-based methods for classification of 2D and 3D data [22], [23]. 3D CNNs make use of voxelized representations of 3D geometry models for training and classification [24], while 2D CNNs can be trained to classify grayscale or RGB images - most commonly photographs, including those featuring structural damage [25], [26]. Research by Wang *et al.* has shown that the use of 2D image-based classification, particularly 2D CNNs, is optimal in terms of performance and classification accuracy [27]. A specific method for classifying 3D objects using 2D images is known as multiview classification [28]. Multiview classification is based on the concept of taking several consecutive images of a 3D object, or part of the 3D scene, and using the classification results generated by a 2D CNN to classify the 3D object. Such an approach can be used for classification of both 3D geometry, as well as point cloud representations of built environment features [29], [30].

2.3. Point Cloud Geometry Reconstruction

Geometric reconstruction of 3D point cloud geometry to a 3D triangular mesh representation requires the use of various 3D spatial approximation algorithms that can detect a minimum number of vertex primitives in order to triangulate mesh surfaces. The simplest and least accurate method is the approximation of a *3D convex hull* [31]. This approach is useful for approximating the overall bounding-shape of the 3D shape represented by point extremities in a given cluster, but does not preserve any required details such as surface curvature. A more robust method that can preserve the surface curvature to varying level of detail is the *Poisson surface geometric reconstruction* method [32], which is generally more suited towards reconstructing organic shapes due to its nature to *smoothen* hard edges in its 3D shape approximation. Alternatively, the *Ball-Pivoting Algorithm* (BPA) is another surface geometric reconstruction algorithm that approximates a triangular mesh by connecting every three vertices that touch the radius of a rolling sphere [33]. The BPA method can preserve the hard edges often found in the physical built environment, therefore it is more suitable for 3D mesh geometric reconstructions. Another common geometric reconstruction method is based on *Delaunay triangulation*, where the circumcircle of a triangle is used to detect $3 + n$ vertices for triangulation. A modern version of this approach has been adapted in various configurations, including a meshing algorithm suited for reconstructing partial point clouds [34].

The final important element of 3D triangular geometric reconstruction from point clouds are the ability to use the reconstructed meshes in adding, subtracting or merging operations with *as-designed* BIM geometry. This requires the use of CSG operations [35]. CSG allows for the approximation of cavities and merged geometry using Boolean operations. The quality of CSG results depends largely on the implemented partitioning and tessellation schemes, but often 3D meshes that have CSG operations performed on them will increase in geometric and visual complexity. The use of CSG allows for the introduction of explicit 3D geometries into the implicit building component geometry representations, which is used in the IFC file representation (*B-Reps* — Boundary Representations) [36].

2.4. Semantic Enrichment of IFC Files with Bridge Damage

Integrating the geometry and features of the detected damage into the Bridge Information Model (BrIM) has been a subject of research for a decade. Some researchers try to use the existing BIM software solutions to model damage elements and therefore commit to the proprietary data modeling formats. Others develop openBIM-based data models. There are differences in damage data input as well. Whereas some tend to keep the manual input based on inspection reports, others use digitally captured images and/or point clouds.

McGuire *et al.* investigated the damage modeling capabilities of commercial BIM software [37]. They tried to model damage features such as location, type, severity, and volume by using LEAP Bridge [38], Tekla Structures [39] and Revit [40]. However, since none of the analyzed software had an embedded functionality to model damage, they developed a Revit plugin. The proposed plugin models damage as a parametric solid — a *parallelepiped*. Relying on the traditional inspection procedure, the inspector is asked to detect damage and estimate the location and dimensions of a corresponding parallelepiped. Additionally, the inspector is expected to rate the damage severity according to AASHTOWare Bridge condition state ratings [41]. McGuire *et al.* afterwards proposed an Excel-based structural condition assessment tool [37].

As opposed to a proprietary format (such as the one used by Revit), IFC is a neutral format for the exchange of digital building models developed by buildingSMART International (bSI). The content and structure of each IFC file must comply with the IFC schema, written in the EXPRESS data modeling language, defined in STEP standard (ISO, 2004) [42]. The IFC schema

specifies the definitions of all the objects, their properties, and mutual relationships. The IFC schema strictly separates the semantic and geometric representation of objects. Physical objects are defined in the *Product Extension* of the Core layer of IFC data model. Objects defined in the Product Extension can have single or multiple geometric representations [43]. Targeting buildings, IFC schema is not appropriate for the description of bridges. For this reason, efforts on extending the existing schema for bridge modeling are ongoing. The development of the *IFCBridge* specification, containing definitions of bridge-specific entities started by an initiative of Yabuki *et al.* [44], is planned for future finalization and release [45].

Hüthwohl *et al.* describe both the inspection details, as well as the defect type, nature, and properties [46]. They distinguish the defect as a deterioration process (*defect*) from the defect as visually observable damage on a surface of the bridge structure (*element defect*). Therefore, the *defect* semantics are modeled as *IfcElementAssembly*, capable of containing the aggregation of several *element defects*. The condition rating of an element defect is represented by *IfcPropertySet* of predefined type *Pset_ Condition*. The defect is connected with a damaged IFC element using the relationship *IfcRelAggregates*, so that the assignment of a single defect to multiple IFC elements is possible. For an element defect geometric representation, the *IfcSurfaceFeature* entity is proposed. This work was further expanded by Hüthwohl et al. (2019) , where image-based CNNs were used to detect and categorize the severity of structural damages and defects in bridges (e.g., spalling, cracks and varying combinations of both) [47]. Recent research by Isailović *et al.* proposes the method for feeding the inspection database of BMS by BIMs enriched with damage information [48]. They insert the manually detected point cloud-based damage geometry into the existing BIM by performing the CSG boolean difference operation on damaged bridge elements.

3. Proposed method

The method for generating the *as-is* BrIM is described using the Business Process Modeling and Notation (BPMN) [49] (Fig. 1). The prerequisite input for the proposed process is an *as-designed* IFC and point clouds. The first two activities prepare the input data for further processing. To be mutually comparable, the *as-built* bridge geometry representation should be aligned to the *as-designed* one. For this reason, the BrIM geometry is converted to a 3D triangular mesh (Wavefront OBJ file format) by using IFCOpenShell

library [50], and the point cloud is manually registered to the mesh by using CloudCompare software tool [51].

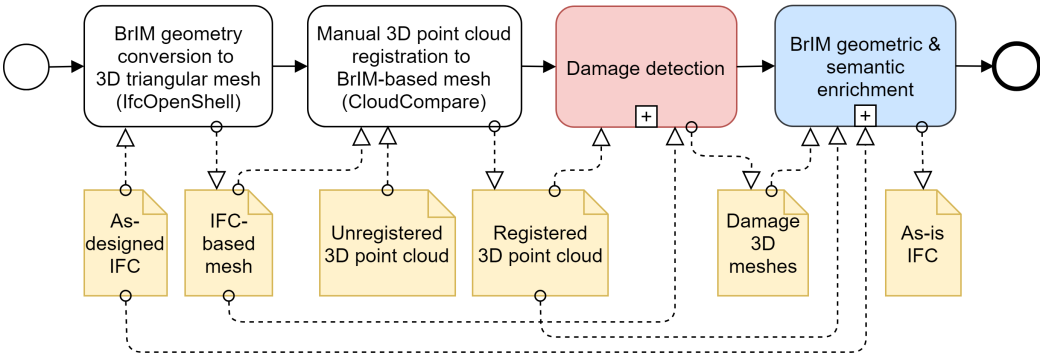


Figure 1: Proposed BPMN process map of *as-is* BrIM generation.

3.1. Damage detection

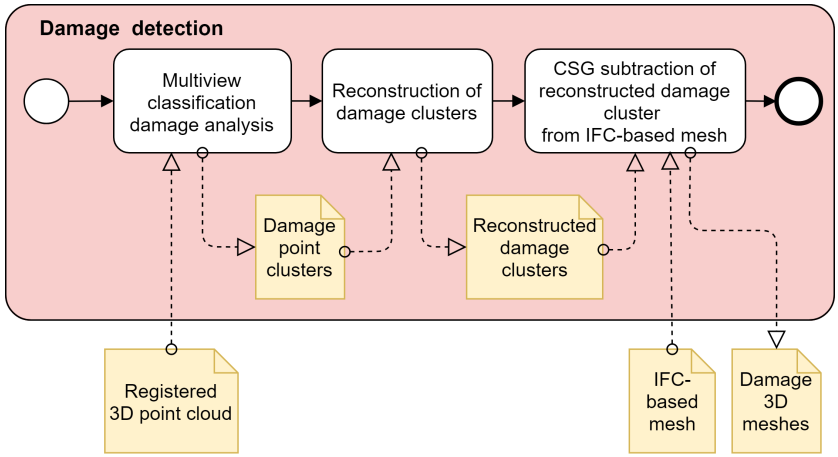


Figure 2: Damage detection sub-process.

3.1.1. Point Cloud Preparation

Before further analysis of the bridge point cloud, it needs to be verified to enable more accurate damage analysis. The acquired point cloud model should meet the following criteria:

1. The point cloud is spatially aligned with the *as-designed* or *as-built* IFC model Level-of-Detail (LOD) 200 [52]).

2. The point cloud is complete, and no major components are missing.
3. The quality of the point cloud is acceptable, and the surface noise is acceptable.
4. The point is of high resolution, so as to portray important graphical elements required for classification.

The most critical criteria items are (1) the correct alignment of the IFC model and the point cloud, as well as (4) using a point cloud with a high resolution (to preserve the visual fidelity required for correct multiview classification). Incorrect alignment can lead to the injection of semantics into false spatial regions of the *as-is* IFC model, while using a point cloud of low resolution can obscure and distort the possible damage features required for the identification of damaged elements.

3.1.2. Image Classification-based Damage Analysis

Since 3D geometry types can be classified using their 2D image projections, the use of multiview classification can easily be extended for classification of images of point clouds. The presented approach is focused on classifying images of point clouds containing RGB color attributes, in addition to their spatial positions. A 2D CNN can be trained using real-life photographs of various built environment elements or artefacts, and used to detect these features contained in a given 3D RGB point cloud representation. The classification accuracy when using a multiview approach to classify 3D point clouds largely depends on the quality of the point cloud, the amount of visual noise and clutter, as well as the entropy of the given image used for classification (e.g., how much of useful visual information can be captured in a given multiview image).

Using a 3D point cloud model of a given bridge, the proposed multiview approach can detect spalling damage regions from partitioned 3D point cluster regions of the model - by generating consecutive images of the given point cluster region, classifying them using a retrained CNN, averaging the classification result by the number of multiview images of the given point cluster, and streaming the classification result back to the point cluster as a new semantic. This enables the detection of spalling damage regions, and semantically enriches the corresponding point cloud model at the same time (Sec. 5.2).

3.1.3. Multiview-Classification Implementation

The approach described by Stojanovic *et al.* is used to generate a cubemap representation of the contained point clusters [53]. The use of an octree-partitioning scheme that generates multiview cubemap was selected, as it is adaptable to varying point cloud complexity, and provides acceptable classification results for RGB point cluster classification [53]. At each octree node that contains a point cluster, a virtual camera position is computed. This virtual camera generates a cubemap of the location, while the visibility of other octree nodes is disabled. This approach allows us to capture the complete environment around each node center as a single image. Cubemap faces, whose average RGB value is higher than 250 are disregarded, meaning that they contain mostly white space. If the scene contains bright surfaces, the RGB average value for discarding faces can be lowered (usually to an average RGB color channel value of 123). The summation of the classified cubemap images (six in total), is then used to create the final classification score for the given node. If the majority of cubemap images feature elements that are classified as damaged, then the given node will be classified as containing point clusters that represent potentially damaged elements.

3.1.4. Multiview CNN Retraining and Classification

A CNN (Inception V3) that is retrained on numerous photo examples of damaged and non-damaged reinforced concrete elements is used for spalling damage detection. The Inception V3 CNN model was retrained using TensorFlow in Python 3.6. Only the last bottleneck layer of the CNN was retrained with the new image categories. The training data input vector size is $300 \times 300 \times 3$ elements (3 RGB color channels). Random distortion of training data (brightness, scale, and cropping) was not utilized. A linear *softmax* function for generating the classification probability scores for the input image data was used.

Several factors influence the classification result outcome, and these factors concern both the way the 3D point cloud clusters are presented as 2D images, and the way the CNN model is retrained. First, the size of the RGB points in 3D space needs to be considered. Since the classification approach captures images of point cloud clusters and uses them for classification, the selection of an optimal point size for the representation of the point cloud is required. This means that the selected size of the points does not introduce too much space between points, so that the scene background color (whitespace) is not pronounced or dominant in the generated multiview images of

the point cluster — but at the same time does not make the points too large so that aliasing artifacts become pronounced.

Once the selected point cloud segments have been processed and the optimal point size selected, the density of the octree partitioning scheme of the 3D point cloud needs to be decided. Since cubemap images for classification from each of the octree nodes are generated, the density of the octree directly affects how many points we will be able to capture and project as perspective-view 2D images in each of the generated cubemap faces for a given node. A cubemap is projected from the center of each octree node in all six directions. If a low octree resolution is chosen (resulting in coarse octree partitioning), the generated 3D perspective-view cubemap faces will feature large portions of the point cloud captured within the horizontal and vertical field of view of 90 degrees. Therefore, if a large portion of the point cloud in the perspective view of a cubemap face is captured and classified, it may contain both damaged and non-damaged regions. Depending on the physical size of the damage features in the training data, the classification system may miss the damaged regions because they are too small in the captured images. In that case, a higher resolution octree has to be used, and the point cloud has to be partitioned into smaller node regions for cubemap image generation (thus increasing the processing time).

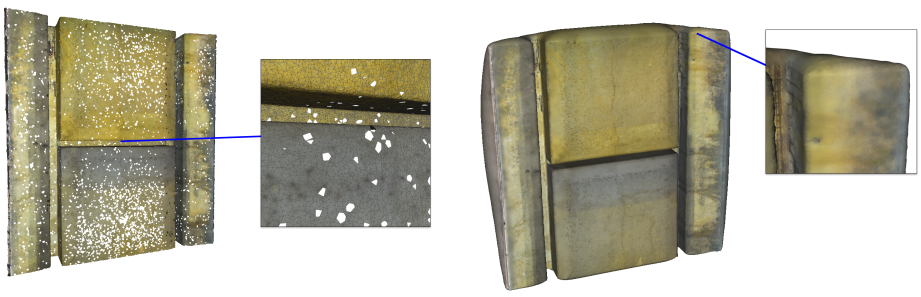
3.1.5. Damaged Cluster Geometric Reconstruction

Regions of the point cloud model that have been classified as damaged can be extracted as point clusters. The classification value of an octree node, which is set either as damaged or healthy, can be used to determine if the points contained in the Axis Aligned Bounding Box (AABB) of that node are to be extracted. The points in each node that is classified as damaged can be copied into system memory and exported as tabulated Cartesian world coordinates and RGB values encoded in a simple ASCII text file. This file format can easily be interpreted by point cloud processing software such as CloudCompare. The extracted points representing clusters that contain potential damage features can further be reconstructed as triangulated geometry, and used for CSG operations on the IFC geometry. Otherwise, they can simply be used alongside either the point cloud or IFC geometry for comparative visualization.

Normal vectors for each of the point clusters need to be pre-computed prior to any kind of geometric reconstruction. This can be accomplished by analyzing the local neighborhood of a point [54], where the normal vector is

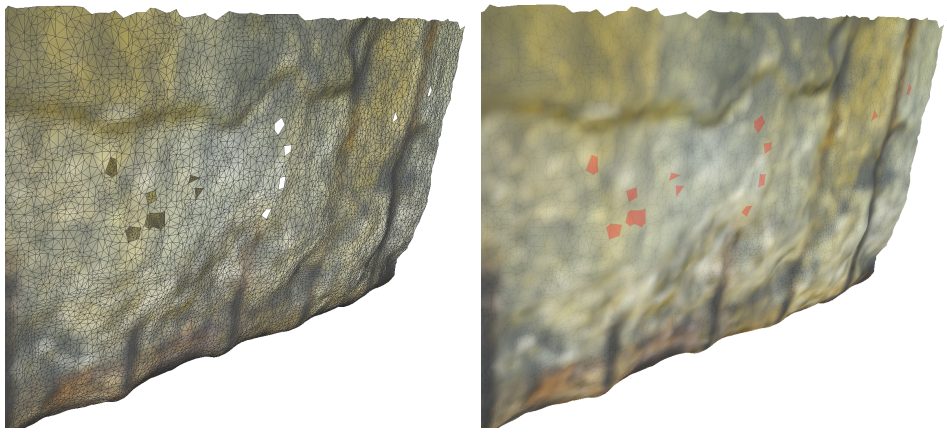
oriented according to the represented neighboring points. The neighborhood of a point can be computed using the co-variance matrix of the k-nearest neighbors, and corresponding eigenvectors and eigenvalues [55].

The two evaluated geometric reconstruction algorithms (Poisson and BPA (Sec. 2.4), have advantages and disadvantages when it comes to reconstructing point cloud representations of bridge structural elements. BPA can preserve the sharper edge features in the reconstructed geometry result, but often also introduces holes into the mesh as a consequence of generating non-manifold edges (Fig. 3(a)). Poisson geometric reconstruction does not preserve sharp edges typically found in man-made structures such as bridges, thus cannot be used for complete model geometric reconstruction but is potentially suitable for smaller damaged element geometric reconstruction (Fig. 3(b)). Both geometric reconstruction methods usually require further manual editing of the reconstructed geometry. This can be seen in Fig. 4, where an automated hole-filling method is used [56], to try and reduce the number of surface holes in the reconstructed mesh using the Ball Pivoting algorithm.



(a) (b)

Figure 3: comparison of geometric reconstruction using the BPA and Poisson methods. (a) BPA surface geometric reconstruction example, and (b) Poisson surface geometric reconstruction example.



(a) (b)

Figure 4: Example of post-processing the geometric reconstruction mesh in order to close any holes. (a) Reconstructed mesh with open holes, and (b) closed hole regions (red).

3.2. BrIM geometric and semantic enrichment

The term *semantic enrichment* used here complies to the definition given by Sacks *et al.* [57]. Once the damage geometry is inserted into the BrIM, it is enriched with the corresponding semantic information (e.g., damage type, severity, and extent). This information adheres to the BMS damage classification.

The *as-built* BrIM represents the bridge at the time of completion. Although the geometry of the constructed bridge should comply with the designed one, this rarely happens in reality. Newly constructed concrete bridges contain various imperfections, mainly caused by the construction inaccuracy either due to slightly misplaced formwork, or the formwork deformation due to the weight of the fresh concrete. The settlement of the foundation can be also a less common cause. Before the *as-designed* IFC is adjusted to the *as-built* one, a triangular mesh is reconstructed based on the registered 3D point cloud using the MeshLab software tool [58].

The proposed sub-process for geometric and semantic enrichment of IFC representation of the bridge is shown in Fig. 5, where damage is represented as a voided volume in IFC geometry. The damage is previously detected on the *as-is* point cloud. Therefore, the damage geometry must be injected into the *as-built* IFC, rather than into the *as-designed* one. The damage mesh that is subtracted from the bridge element can be unfortunately outside the

element (due to the deviation of the *as-built* comparing to the *as-designed* geometry).

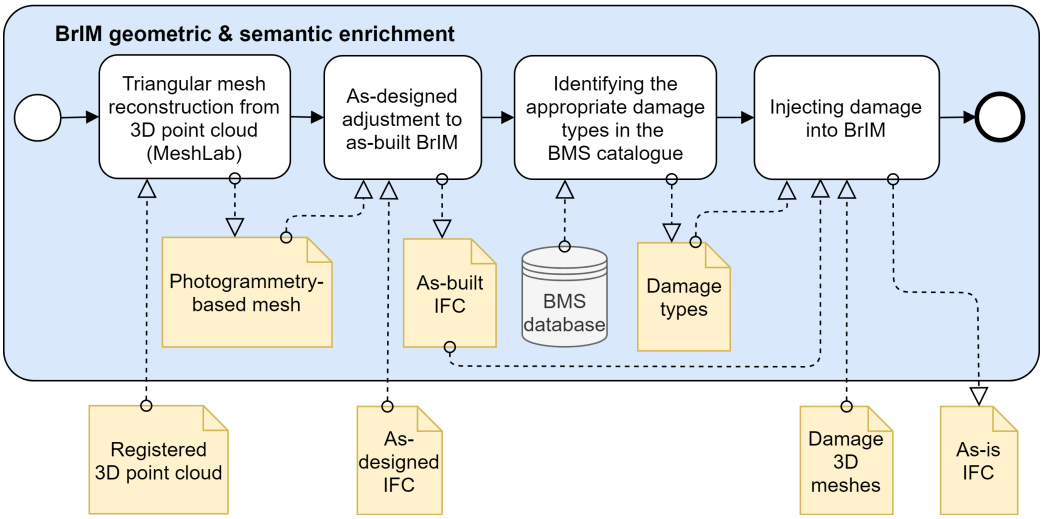
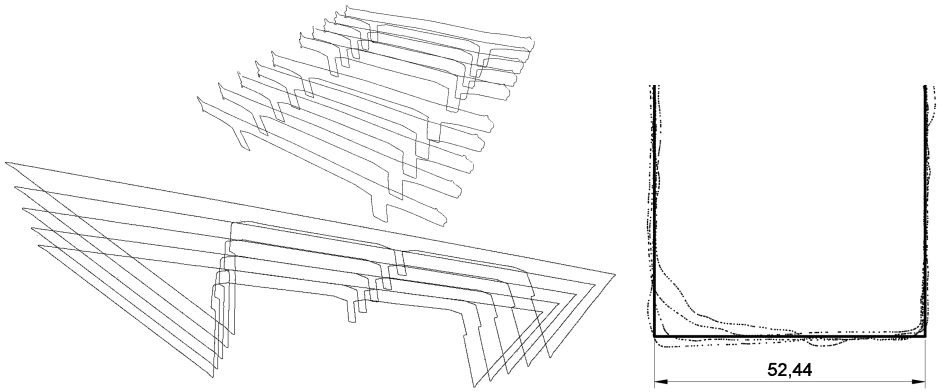


Figure 5: BrIM geometric & semantic enrichment sub-process.

Once the *as-built* IFC is generated, the detected damages are manually paired with appropriate damage catalog types from the database of a specific BMS. Finally, the previously generated damage meshes with its catalog types and accompanying semantics are injected into the *as-built* IFC, producing the *as-is* IFC.

3.2.1. As-designed adjustment to as-built BrIM

The *as-built* geometry of the bridge is assessed by slicing the reconstructed photogrammetry-based mesh equidistantly in two orthogonal directions using MeshLab. The transverse and longitudinal bridge cross-sections are shown in (Fig. 6(a)). Exported in the DXF format, the cross-sections are overlapped and the centerline is manually fitted using Autodesk AutoCad. The cross-sections centerline represents the actual bridge contour in two orthogonal directions. Detail of cross-section overlap and fitted centerline is shown in Fig. 6(b). The actual bridge dimensions are measured, and the BrIM is remodeled using Autodesk Revit [40]. Finally, the BrIM is exported to IFC format (*as-built* IFC).



(a) (b)

Figure 6: (a) Transverse and longitudinal bridge cross-sections: slices of photogrammetry-based mesh. (b) Detail of beam cross section contours overlap (dotted lines) and fitted centerline (continuous line). The actual beam width is measured 52.44 cm, whereas the designed width is 50 cm.

3.2.2. Injecting damage into BrIM

Once the damage elements are detected and their geometry extracted as triangular meshes, the *as-built* IFC can be enriched with damage information to form the *as-is* IFC. The enrichment of *as-built* IFC is two-fold: geometric and semantic. Semantic enrichment needs to meet two requirements: damage features need to comply with BMS damage classification, and semantic data structure need to comply with IFC schema.

BMS-based damage semantics.

The damage data structure is implemented differently in various BMSs around the world. Each BMS has a unique condition rating system, the format of an inspection report, damage types, etc. The method for including BMS-based damage semantics is inspired by the Swiss Federal Roads Authority BMS named KUBA [59]. Without going into details of Swiss bridge inspection procedure, only the required damage data will be listed and explained. According to [60] and [61], for each noticed damage, the following information needs to be assessed and documented:

- Damage type: classification of a visible surface defect, selected from the BMS catalog.

- Deterioration process: physical-chemical process causing surface defects, selected from the BMS catalog (KUBA distinguishes nine deterioration processes).

- Damage position: rough distance measure, relative to the dimension of the inspected element.

- Damage extent: an approximate measure of the damaged region (areal dimension or percentage of the damaged region relative to the overall element surface).

- Damage severity: damage condition rating complying with BMS damage rating system (KUBA distinguishes five ratings: 1 (good), 2 (acceptable), 3 (defective), 4 (poor), 5 (alarming)).

The position and the extent will be implicitly determined by the *as-is* IFC geometry. State of the art tools for BIM analysis, such as spatial query language QL4BIM [62], are capable of sophisticated analysis of mutual relationships between IFC objects. Nevertheless, no straightforward solution appropriate for damage severity assessment for bridges is currently available. In most cases, due to the complexity of the task and the required expertise, the deterioration process has to be manually assessed by an experienced and in some countries licensed structural engineer.

Observations made in course of inspections are not necessarily damages. These observations can be thought of as symptoms of damage processes, eventually leading to structural damage. It is therefore that in KUBA 5.0 the term *Inspection finding* is used instead of damage. Fig. 7 depicts the proposed class diagram of *Inspection finding*, compliant to the data structure of KUBA 5.0. Attributes of *Inspection finding* are the textual description and 3D *geometricRepresentation*. It is associated with *Inspection*, described by date and type. Furthermore, the *Inspection finding* is associated with *Damage severity*, as well as *Catalog type of damage* and *Damage property*. Whereas the first two are catalog entries, defined by *hierarchyCode*, the *Damage property* is optional, added only if the damage extent cannot - at the moment - be precisely derived (e.g., crack width) from the *geometricRepresentation* of *Inspection finding*. BMS Catalog type includes information both on damage type, as well as corresponding deterioration process.

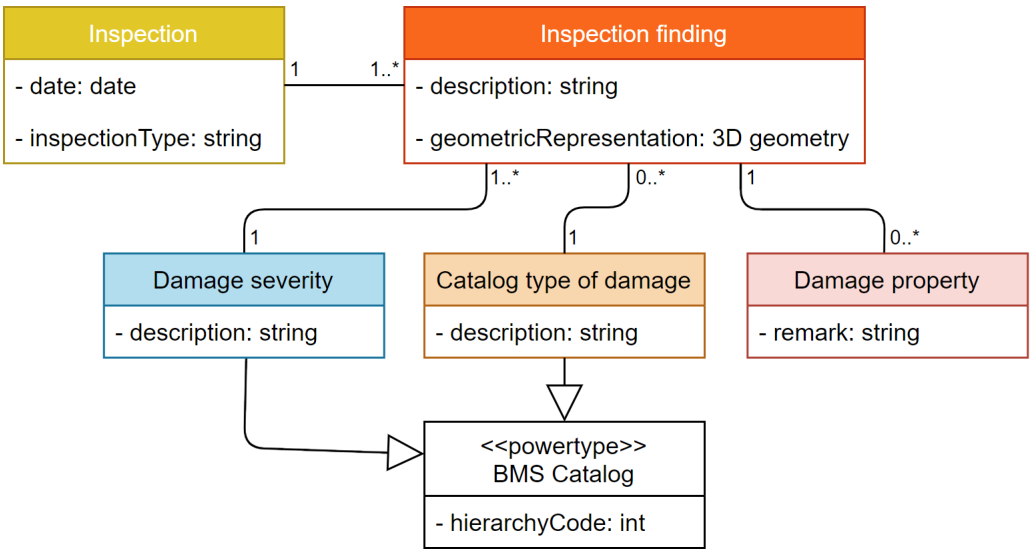


Figure 7: Inspection finding class diagram.

Proposed IFC structure.
Rather than proposing the schema extension, the existing schema (IFC4 Version 4.0 - Addendum 2 [63]) definitions are used. KUBA 5.0 groups mutually close damages related to the same deterioration process. Therefore, the *Inspection finding* is modeled as *IfcElementAssembly* of the user-defined type *INSPECTIONFINDING*, an aggregation of damages represented by *IfcSurfaceFeature* instances of user-defined type *DAMAGE*. The *IfcSurfaceFeature* is able to accurately and meaningfully represent damage geometry. Therefore, the missing volume due to damage can be subtracted from the original volume of the bridge elements. Damage (*IfcSurfaceFeature* instance) is connected with the damaged element of the bridge (represented by the instance of *IfcElement*) by the relationship *IfcRelVoidsElement*. The proposed IFC structure is shown in Fig. 8. Colors of IFC entities correspond to the colors of the classes in Fig. 7.

Any of the bridge elements can be damaged, so *IfcElement*, an abstract superclass of all the structural components, is used to describe damaged elements in Fig. 8. The way the damage (*IfcSurfaceFeature*) is associated to the damaged element is shown in Fig. 9.

Damage severity and *Catalog type of damage* are modeled as instances of *IfcPropertySingleValue*, members of *IfcPropertySet*, connected with damage by relationship *IfcRelDefinesByProperties*. KUBA partitions each bridge

element based on damage groups found at that element. Furthermore, the condition rating (e.g., *Damage severity*) refers to the damage extent, which includes damages of the same group. Thus, *Damage severity* is represented by *AssesmentCondition*, *IfcPropertySingleValue* of the predefined *IfcPropertySet* named *Pset_ Condition*, connected with *Inspection finding* by relationship *IfcRelDefinesByProperties*. The best IFC match for modeling *Inspection* is found to be the *IfcTask*. Its attributes *TaskTime* and user-defined *ObjectType* perfectly correspond to the proposed definition of *Inspection* class. *Inspection* (*IfcTask*) is connected with *Inspection finding* (*IfcElementAssembly*) by relationship *IfcRelAssignToProcess*.

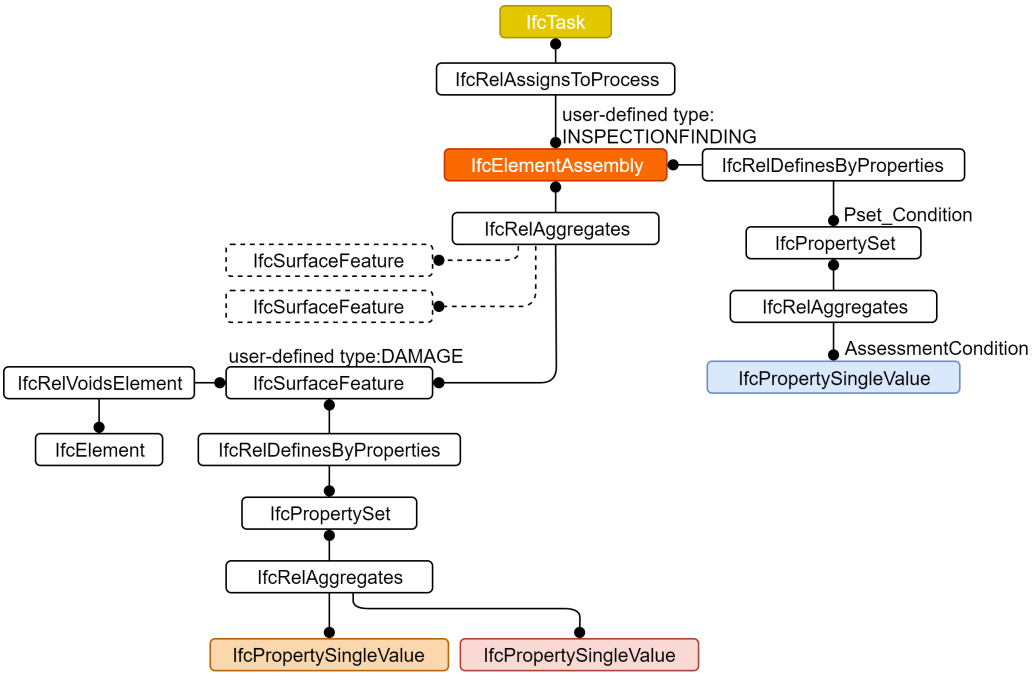


Figure 8: IFC structure for description of inspection findings.

Damage geometry in IFC. Once the point clusters have been reconstructed and adjusted, they can be used as CSG elements to perform Boolean operations on a given extracted geometry mesh from the BIM dataset. The triangulated mesh also needs to be generated from the BIM representation, and each of the CSG operations need to be performed for each bridge element separately. Geometric representations in the IFC file are stored as *boundary representations* (B-Reps), so specified geometry segments are subsequently

extracted at the desired LOD. The parsing and extraction of the BIM geometry from the IFC file were enabled by using the IFCOpenShell library [50]. In the final pre-processing step, the CloudCompare software tool can be used to align and verify the transformed point cloud data with the extracted BIM geometry data using its built-in deviation analysis features.

Fig. 9 shows the IFC structure for damage representation. The chosen IFC entity for the geometry representation of damage is *IfcTriangulatedFaceSet*. Defined this way, geometric representation of the defect is self-sufficient for any kind of further structural analysis or condition assessment. Furthermore, *IfcTriangulatedFaceSet* is a straightforward IFC entity for the description of triangular meshes. Whereas the *CoordIndex* attribute represents the indices to three vertex points of the same triangular face, the *IfcCartesianPointList3D* stored in the *Coordinates* attribute represents coordinates of the indexed mesh vertices.

Fig. 9 illustrates the IFC representation of concrete spalling on the bridge girder. The IFC objects describing spalling are colored green, whereas the objects describing girder are colored grey. The IFC objects defining the relationship between the spalling and the girder are colored red. The *IfcSurfaceFeature* instance must be hosted by an instance of a child of an *IfcElement* (in this case *IfcBeam*). This is implemented by the *IfcRelVoidsElement* relationship between the *IfcSurfaceFeature* and *IfcBeam*. This relationship ensures an automatic computation of the result of CSG difference between geometric representations of those two objects every time the model is to be rendered in IFC viewer. The local placement (*IfcLocalPlacement*) of both objects refers to the same instance of *IfcCartesianPoint*, so the previously performed alignment between IFC and point cloud model representation is preserved (e.g., the volumes of the objects overlap). Although nested in the geometric representation of a girder (*IfcBeam*), reinforcement bars (*IfcReinforcingBar*) are not voided. Instead, they stick out of damaged elements. They also mostly correspond to real spalling geometry, therefore this representation allows computation of the extent of exposed reinforcement.

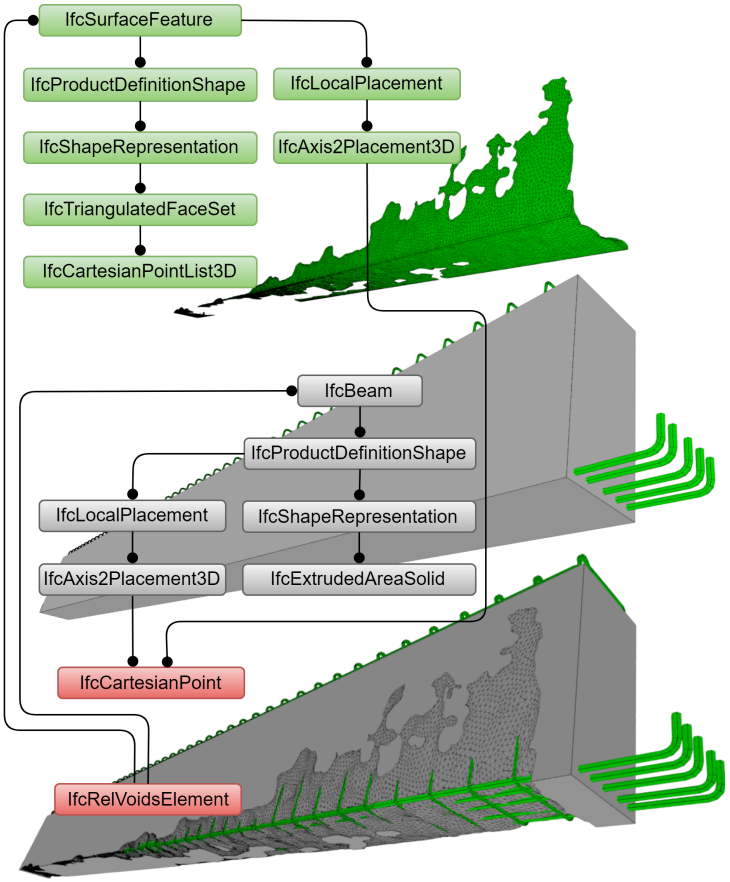


Figure 9: IFC structure for geometric representation of damage.

4. Case Study

The main objective of the case study was to evaluate the use of the octree-based, multiview classification approach, for the detection of spalling damage point clusters. These damage point clusters can then be injected into the *as-is* IFC model (Sec. 3.2.2). The proposed approach is applied for a photogrammetrically acquired point cloud of a bridge over the river Gročica, located in the Grocka municipality of the city of Belgrade, Serbia (Fig. 10). It is a 12.5 meter spanned simply supported double girder bridge built in the 1930s. Neglect in addition to an inappropriate designed and poorly maintained drainage system has caused large spallings, accompanied by extensively corroded reinforcement on girders, abutments, and curbs. Thus,

extensive defects (larger than approximately 10 cm), detectable by the proposed method are present. Detected damage corresponds to the following damage types from KUBA catalog [59]:

- Cracks due to reinforcement corrosion
- Spalling
- Chipped off patched spots
- Fractured reinforcement
- Chipping-missing pieces
- Loss of chippings
- Slightly corroded Reinforcement
- Strongly rusted reinforcement

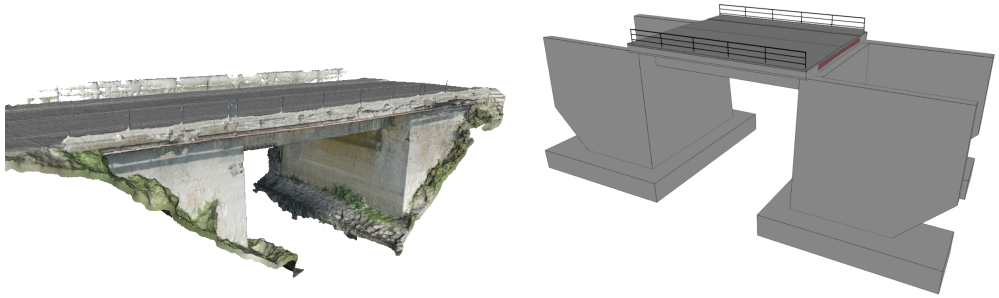
The bridge is owned and maintained by the public enterprise Roads of Serbia (RoS).

(a) (b)

Figure 10: Case study: Bridge over river Gročica, located in Grocka municipality of the city of Belgrade, Serbia. (a) Aerial photograph showing the location of the bridge, and (b) Perspective view of the bridge.

The point cloud of the bridge (30 708 690 points) is shown in Fig. 11(a). It was generated using aerial photogrammetry, where sequential images of the bridge were captured and aligned. The *as-designed* BrIM is modeled using

Autodesk Revit [40] and exported to IFC format (Fig. 11(b)). Once the IFC-based triangular mesh is generated and the 3D point cloud is registered to it, the proposed methods for *Damage detection* and *BrIM geometric and semantic enrichment* are evaluated.



(a) (b)

Figure 11: Inputs for the proposed method: (a) Photogrammetry-based 3D point cloud, (b) *As-designed* BrIM.

4.1. Damage detection

Data Preparation.

Firstly, the point cloud model is aligned with the updated *as-built* IFC geometry model. This is accomplished manually using the CloudCompare software tool. The 3D model geometry from the *as-is* Grocka bridge model was extracted as a Wavefront OBJ model using the IFCOpenShell. Next, the 3D point cloud is visually inspected in order to determine the appropriate point size for rendering the 3D point cloud (and generating the cubemap images that encode the 3D point representations). In the last preparation step, manual segmentation of components from the *as-is* bridge point cloud is performed (in order to enable the spalling damage evaluation of each of the main structural components separately). This kind of bridge component segmentation can potentially be automated [7]. The initial manual segmentation scheme chosen for the Grocka bridge model divides the bridge into six different point clusters (Fig 12). Specific structural elements are then segmented further before damage assessment (e.g., girder).

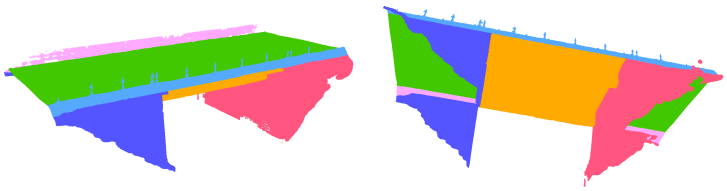


Figure 12: The initially segmented *as-is* bridge point cloud.

Damage Detection System Implementation.
The damage feature detection method using multiview classification was implemented as a prototypical web-based application using a service-oriented paradigm. The prototype web-based application is mostly implemented in JavaScript [53]. This approach was adopted in order to deal with larger and more complex bridge point cloud models, and to specifically detect spalling damage using multiview classification. The main visualization and classification result display systems are implemented within a client web-based application. The client application communicates with the back-end server application by sending the generated cubemap images for classification. The server is implemented using Node.js and communication with the server is established using the Sockets.io and Express.js software libraries. The server listens to any communication by the client from a given port, such as incoming responses for receiving data and responses for sending classification results. The server calls the image classifier script implemented in Python 3.5 using Tensorflow. Once the classification results have been generated in JSON format from Python and stored in a text file on the server, the server loads in these text files before parsing them and sending them to the client. The client application then averages the result for each corresponding node, from which the valid cubemap faces were generated (sec. 3.1.3). Finally, the server removes the generated cubemap faces and any text files containing classification values once the image classification has been completed.

Three.js was used as the main software component for the client-side rendering system [64], as it allows for the use of the OpenGL ES 2.0 and 3.0 API specifications within a compatible client web-browser. One limit of Three.js for visualizing point clouds is the lack of support for out-of-core rendering of massive amounts of point-cloud data, and therefore it can only be used to visualize point-cloud scenes in real-time with approximately 4.5 million points, without resorting to the use of more sophisticated scene and memory management methods [65]. The use of out-of-core rendering refers

to fetching and processing data that is too large to fit into main system RAM or graphics card memory, and therefore has to be streamed in using various algorithms and techniques.

Multiview Classification.
A version of the Inception V3 CNN to detect spalling damage was retrained, where a custom spalling damage dataset was created and used for retraining. The spalling CNN consists of a total of 4804 images (2494 of which are examples of healthy structural surfaces), while the rest of the 2310 images are examples of spalling (Fig. 13). As opposed to cracks, spalling is usually accompanied by visible reinforcement bars. After the careful examination of the photos capturing various deteriorated bridges in the north-west region of Serbia, the examples of concrete spalling were extracted and used for the retraining of the spalling damage detection CNN. For retraining the spalling damage detection CNN, 4000 training steps were used with a learning rate of 0.01, with the final predicted accuracy of the spalling CNN is 79.6%. Although the resolution of the Grocka bridge point cloud is insufficient for detection of the miniature artifacts such as cracks, the retrained CNN detects any unhealthy part of the concrete surface within an acceptable minimum accuracy range of approximately 25 cm, which can include the minimum size of the detected damage elements proposed in the case study requirements (approximately 10 cm).

Segmented parts of the Grocka bridge point cloud (e.g., structural elements of the bridge), were used to test the classification accuracy of the retrained spalling detection CNNs (Fig. 14). An effort was made to establish a correlation between the spatial accuracy of the classification of spalling damage, and the octree nodes used to generate images for classification from each discretized region of the point cloud (Sec. 3.1.4). Through experimentation, it was decided that the resolution of the octree should be between 100-150 nodes that can be used for a point set of approximately 100 000 points. This in turn translates to sampling between 700 to 500 points per octree node, based on the given complexity of the point cloud model. The approximate average physical cubic size of an octree node is 50 cm.

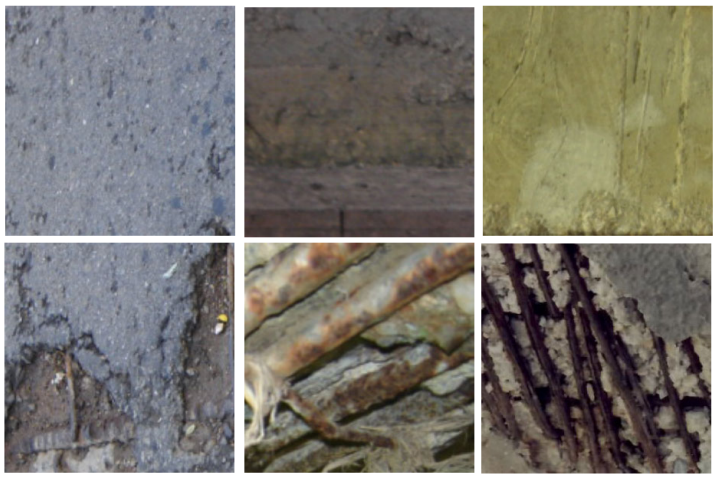


Figure 13: Examples of spalling (bottom) and healthy (top) surface images used for re-training the CNN.

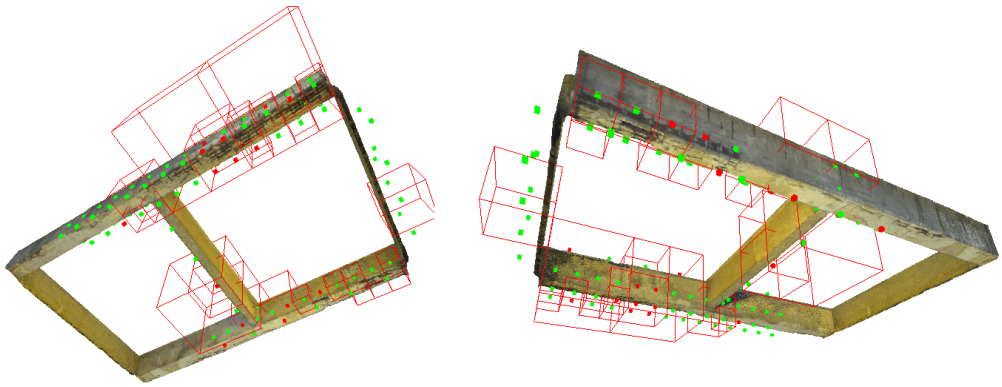


Figure 14: Example spalling damage classification using multiview classification with a girder bridge component. The larger red cubes indicate damage clusters that will be extracted as damage features and used for geometric reconstruction and semantic enrichment of the *as-buit* IFC model.

As-designed adjustment to as-built BrIM.
The dimensions of the photogrammetry-based mesh are assessed in equidistant cross-sections. Neglecting the missing volume of the structural elements due to damage, the centerline of both transverse and longitudinal cross-section is manually fitted (Fig. 15). The only criteria were to keep the cross-

section symmetric after the dimension adjustment. The automation of this step is possible, however, it was out of the scope of this paper.

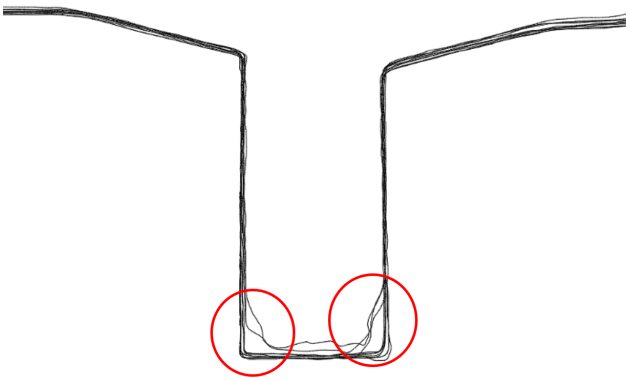


Figure 15: Main girder equidistant cross-sections overlapped (missing volume due to damage is circled in red).

After the analysis, the shape of the bottom console contours is slightly changed and translated for approximately 5 cm upwards. The adjustments of girder dimensions are shown in Table 1. Finally, the BrIM is remodeled according to updated bridge dimensions using Autodesk Revit [40]. The remodeled BrIM is exported as an updated *as-built* IFC model.

Table 1: Adjustment of girder dimensions from *as-designed* to *as-built*.

Element	**Main Girders**	**Middle Cross Girder**	**End Cross Girders**
As-designed cross section dimensions (cm)	50×110	30×106	50×125
As-built cross section dimensions (cm)	52.44×110	32×110	40×125

Injecting damage into BrIM.
Once each damage mesh is associated with single or multiple catalog types, a prototype software is used to enrich the *as-built* IFC with damage data. The prototype software is implemented using IFCEngine [66]. This is a toolbox written in C++, featuring an application programming interface (API) with a set of functions and methods for reading and writing STEP files.

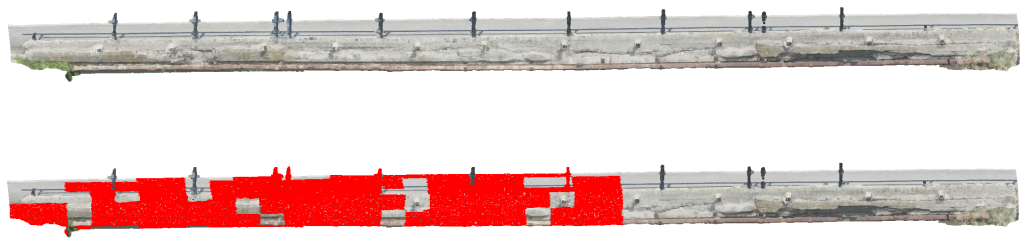
To insert the damage geometry into IFC, a custom OBJ parser was written in C++. The parser reads vertices and faces data from an OBJ file and writes it into the IFC, as *IfcCartesianPointList3D* and *IfcTriangulatedFaceSet*. The final geometry is a result of CSG difference between the *as-built*

IFC and damage geometry, implemented through the *IfcRelVoidsElement* relationship between damage object (*IfcSurfaceFeature*) and damaged bridge element (*IfcElement*). For the model visualization, the existing IFC viewer developed by RDF is embedded in the prototype software. The software is also written in C++, using the Microsoft Foundation Class library (MFC) to provide basic GUI functionality.

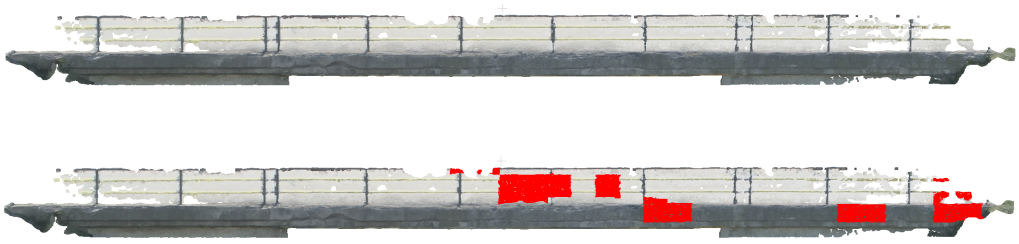
5. Results

5.1. Damage detection

The classification approach was tested using the segmented regions of the north and south bridge abutments, the left and right consoles and the support structure girders. All of these elements feature visible spalling damage. The damage regions were classified using the octree-based multiview classification method, and the extracted damage regions for each of the selected structural elements are shown in Fig. 16 - Fig. 19. The areas highlighted in red indicate possible spalling damage detected by the multiview classification system.

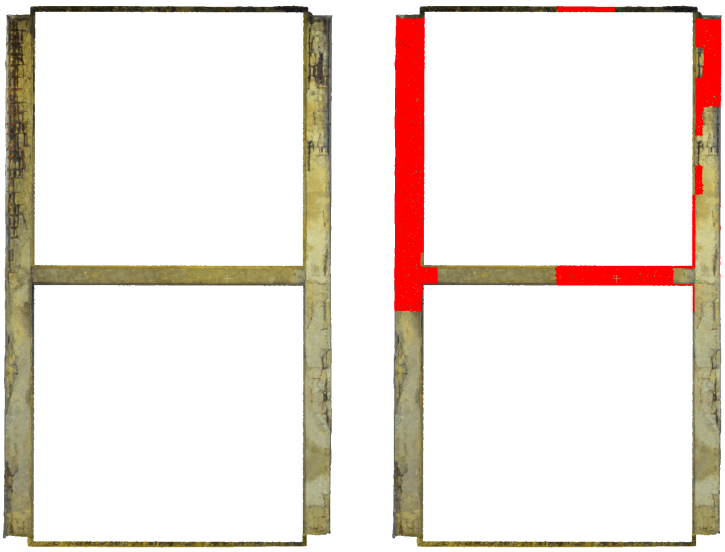


(a)

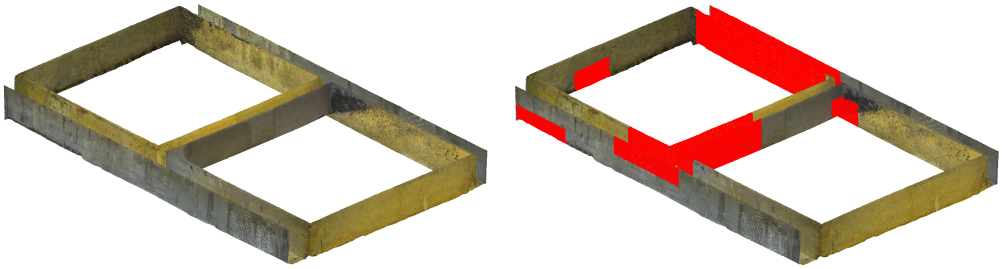


(b)

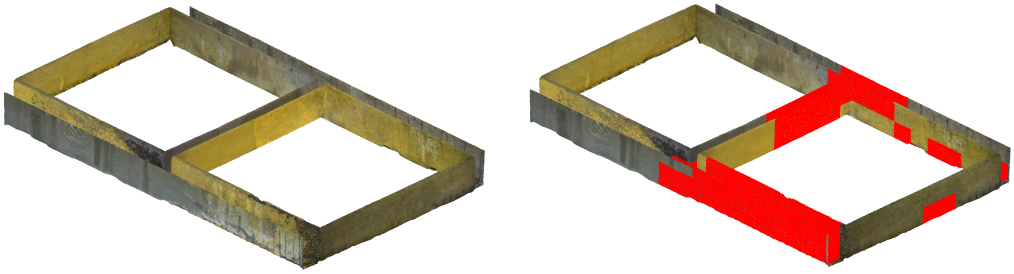
Figure 16: (a) Detected spalling damage in the left bridge console, and (b) detected spalling damage in the right bridge console.



(a)

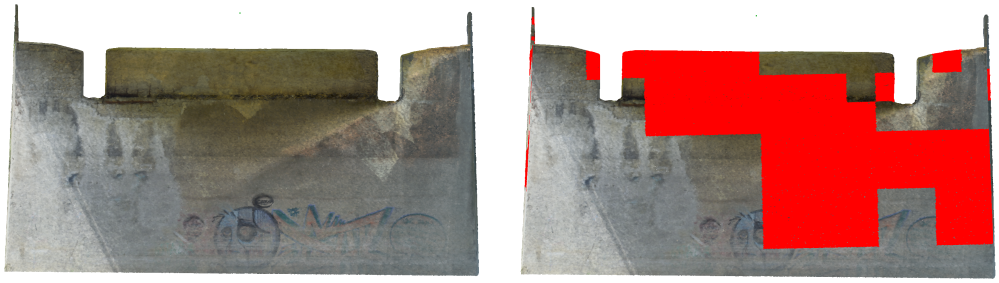


(b)

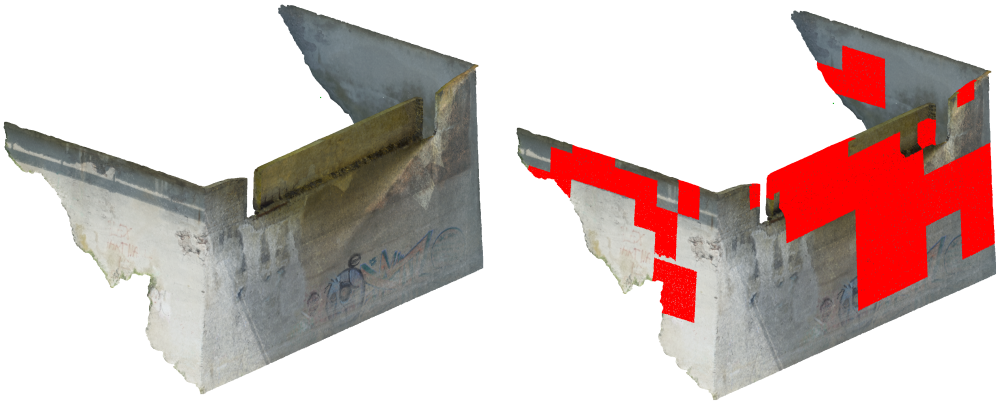


(c)

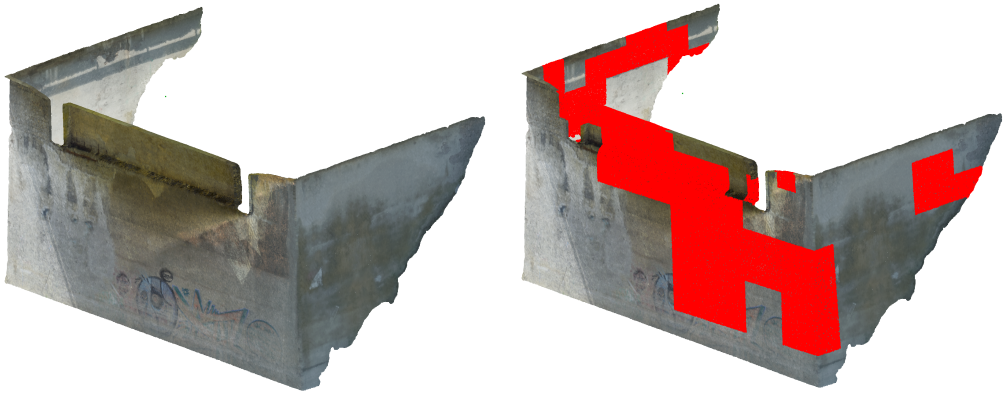
Figure 17: (a) Bottom view of the girder with detected spalling damage. (b) Back view of the same girder with detected spalling damage, and (c) the front view.



(a)

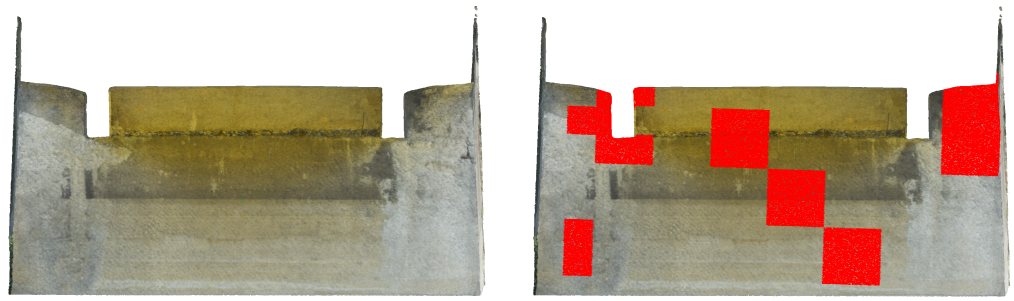


(b)

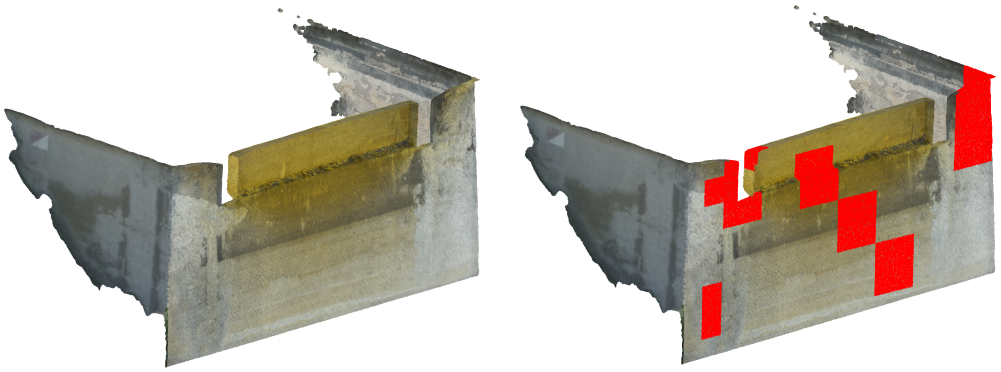


(c)

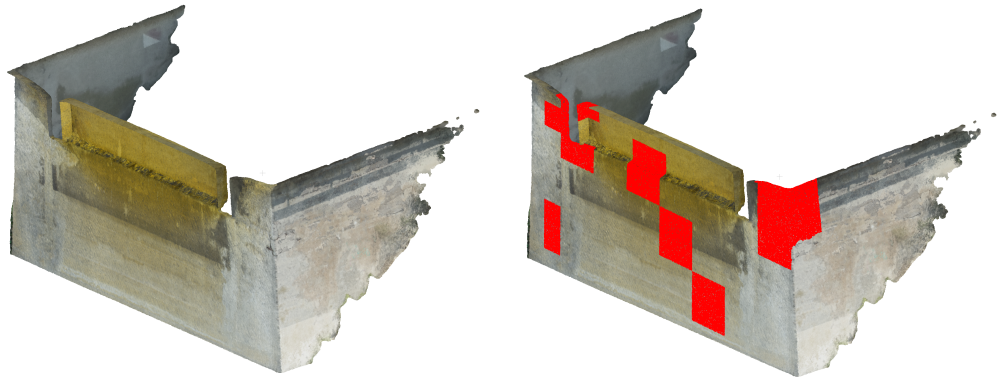
Figure 18: (a) Front view of the north abutment with detected spalling damage. (b) Left view of the north abutment with detected spalling damage, and (c) the right view.



(a)



(b)



(c)

Figure 19: (a) Front view of the south abutment with detected spalling damage. (b) Left view of the south abutment with detected spalling damage, and (c) the right side view.

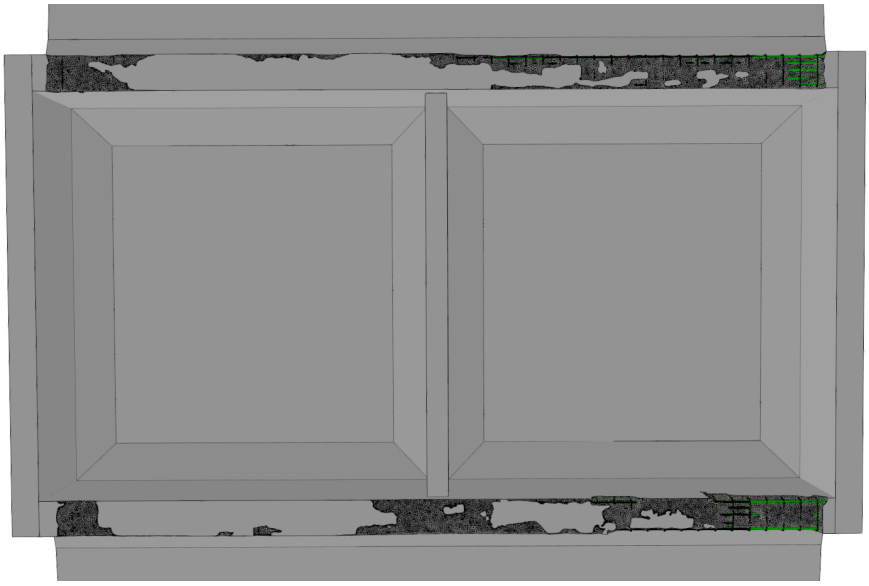
5.2. BrIM geometric and semantic enrichment

The geometric enrichment results were evaluated using an *as-is* baseline IFC, where spalling damage regions were manually detected. This baseline IFC was compared against the second *as-is* IFC, where spalling damage elements were detected automatically and injected using the presented method. The IFC STEP code snippet describing the girder damage, its geometric representation and relationship with the girder is shown in Fig. 20. Visualizations of the baseline IFC and the one with automatically detected damages are shown in Fig. 21 -Fig. 23.

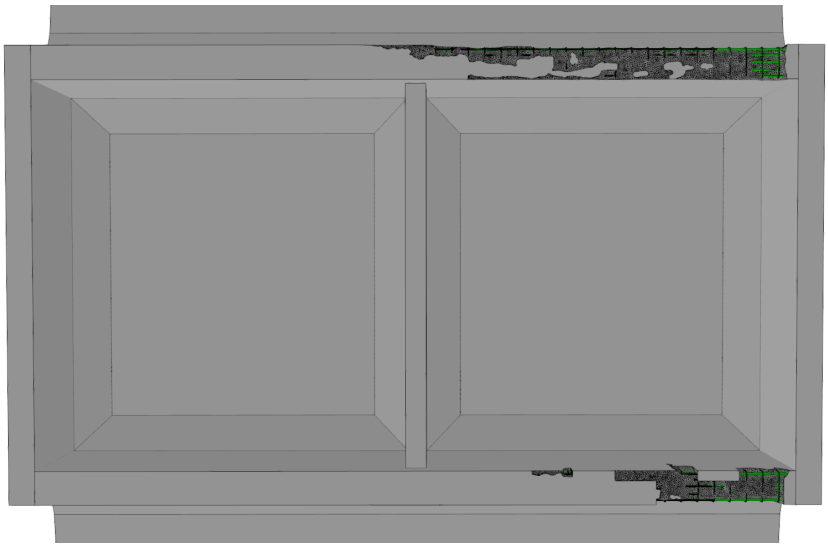
```
/*Bridge girder (IfcBeam)*/
#6 = IFCCARTESIANPOINT((0., 0., 0.));
#67962 = IFCAXIS2PLACEMENT3D(#6, $, $);
#67963 = IFCLOCALPLACEMENT($ , #67962);
#67985 = IFCBEAM('1_T5$UV0vAqBl32$8LDvNv', #41, 'Concrete-Rectangular Beam:52 x 110:
         316571', $, 'Concrete-Rectangular Beam:52 x 110', #67963, #67983, '316571');

/*Damage (IfcSurfaceFeature) geometic representation: IfcTriangulatedFaceSet*/
#121005 = IFCSURFACEFEATURE('3k6jCKxz5BRuSE0w84Zh$V', $, 'Bridge damage', $,'DAMAGE',
          #121006, #121009, $, .USERDEFINED.);
#121006 = IFCLOCALPLACEMENT($, #121007);
#121007 = IFCAXIS2PLACEMENT3D(#6, $, $);
#121009 = IFCPRODUCTDEFINITIONSHAPE($, $, (#121010));
#121010 = IFCSHAPEREPRESENTATION(#101, 'Body', 'Tesselation', (#121011));
#121011 = IFCTRIANGULATEDFACESET(#121012, $, .T., ((6, 8, 9), (12, 13, 366), ...)
#121012 = IFCCARTESIANPOINTLIST3D(((-893.999, -352.476, -186.071), ...)
#121021 = IFCRELVOIDSELEMENT('0sIPpA3CH9ivdmNTQ6k0Gd', $, $, $, #67985, #121005);
```

Figure 20: STEP code snippet describing the geometric representation of girder damage.

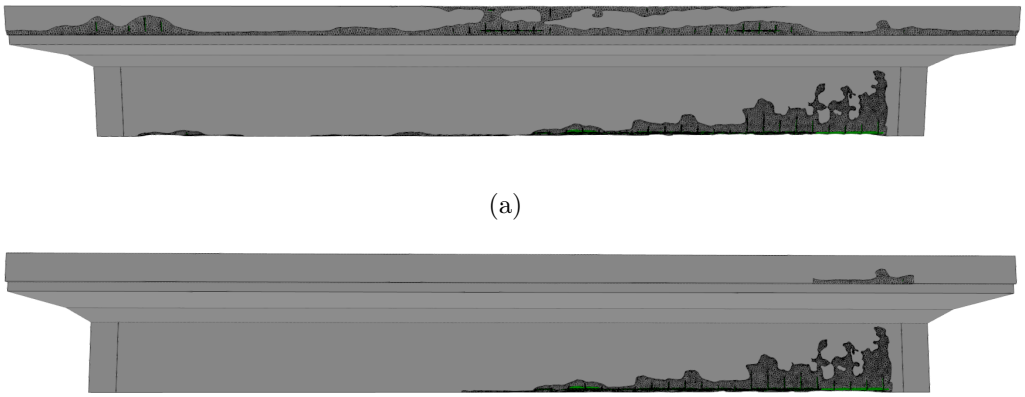


(a)



(b)

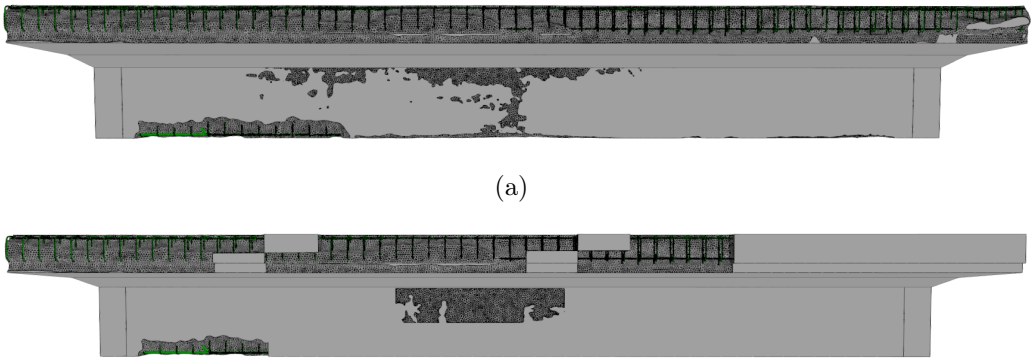
Figure 21: (a) Bottom of the bridge superstructure with baseline results representation, and (b) the IFC with automatically detected spalling damage geometry.



(a)

(b)

Figure 22: (a) East girder and curb of the bridge with baseline results, and (b) the IFC with automatically detected spalling damage geometry.



(a)

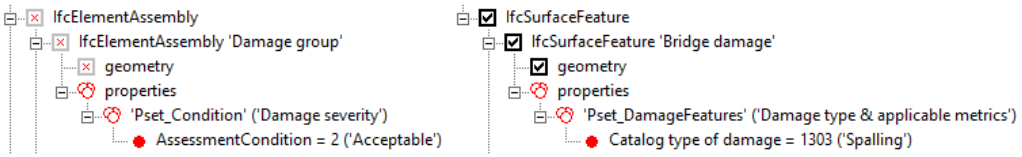
(b)

Figure 23: (a) West girder and curb of the bridge with baseline results, and (b) the IFC with automatically detected spalling damage geometry.

The STEP code snippet and the IFC tree of the damage semantics are shown in Fig. 24, whereas Fig. 25 shows the STEP code snippet describing bridge inspection results.

```
#121013 = IFCPROPERTYSET('1S5u9YpID6kBOhVrFxchDd', $, 'Pset_DamageFeatures',
          'Damage type & applicable metrics', (#121014));
#121014 = IFCPROPERTYSINGLEVALUE('Catalog type of damage', 'Spalling', '1303', $);
#121015 = IFCRELDEFINESBYPROPERTIES ('1kZOnhb4v4fx$Vt$Yar7v4', $, $, $, #121005, #121013);
#121016 = IFCELEMENTASSEMBLY('1IuystymbElBpwLStuzqYP', $, 'Damage group', $,
          'INSPECTIONFINDING', $, $, $, $, .USERDEFINED.);
#121017 = IFCRELAGGREGATES('2XBTDOJsbDkhXwE82N67ph', $, $, $, #121016, (#121005, #121022,
          #121026, #121030));
#121018 = IFCPROPERTYSET('0XWwlHQT1FQgX3NYFEYPIi', $, 'Pset_Condition', 'Damage severity',
          (#121019));
#121019 = IFCPROPERTYSINGLEVALUE('AssessmentCondition', 'Acceptable', '2', $);
#121020 = IFCRELDEFINESBYPROPERTIES('1v7lckcjf7lgeYumWM_tX_', $, $, $, #121016, #121018);
```

(a)



(b)

Figure 24: Damage semantics: (a) IFC STEP code snippet, (b) IFC tree of the embedded viewer.

```
#121059 = IFCTASK('1K9d8_vIX4j86qFmmaKUF3', $, 'Inspection', $, 'Routine inspection',
          $, $, $, $, $, $, #121060, .USERDEFINED.);
#121060 = IFCTASKTIME('Inspection time', 'MEASURED', $, $, $, $, $, $, $, $, $, $, $,
          $, $, $, '2018-08-10T10:00:00', '2018-08-10T15:00:00', $, $);
#121061 = IFCRELASSIGNSTOPROCESS('1KCiqvVunBaBWqdRuHbhWY', $, $, $, (#120829, #120854,
          #120879, #120904, #120933, #120962, #120991, #121016, #121045), $, #121059, $);
```

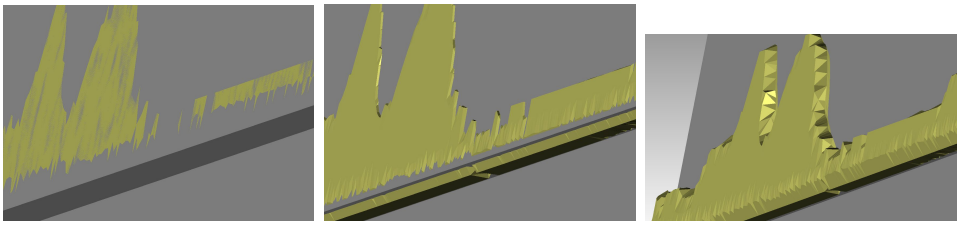
Figure 25: IFC code snippet describing the bridge inspection.

6. Discussion

Multiview Classification. The multiview classification approach for spalling damage detection has successfully detected spalling damage for the core bridge elements, without any user input. However, it can still detect false-negative features (e.g., defects and other environmental features such moss, graffiti, posters, water/rust stains, litter, shadows, etc (Fig. 18(a))). The detected damage clusters are then extracted and reconstructed, before being injected as semantics into the *as-built* IFC model. The presented approach aims to detect macro damage features, as the resolution of the point cloud is too low to attempt to detect finer damage elements such as surface cracks.

The classification accuracy of the CNN could be improved if a larger and more varied spalling image dataset was used for training. The web-based prototype application can classify a scene of fewer than 50 nodes and 50 000 points within two minutes average. Classification of each of the selected bridge components with up to 250 nodes can take on average seven minutes to generate the cubemap images and classify them.

Representation of Damage Semantics in IFC Model.
After being reconstructed as triangulated geometry, the extracted damage clusters are injected into the BrIM, using the proposed IFC geometric representation and prototype software. To generate a visible void, the damage mesh needs to enclose the damaged element. However, the damage mesh resulting from damage detection sub-process fails to fulfill this criterion for two reasons. Firstly, the outer surfaces of the mesh and damaged element coincide. Secondly, the mesh edges are chamfered as a result of a CSG computation. As the most straightforward way to overcome these issues, the authors chose to enlarge the damage mesh before injecting it into IFC, by scaling it. The scale is determined based on empirical tests and the results of these tests are shown in Fig. 26. Fig. 26(a) shows that the damage mesh in original size does not produce any visible void, whereas Fig. 26(b) shows the artifacts due to the insufficient enlargement of the mesh (e.g., damage mesh does not completely enclose the damaged element). The smallest scale which does not produce any artifacts is 100.5 % of original mesh size (Fig. 26(c)). Thus, each damage mesh is scaled to this percentage before introducing to the *as-built* IFC.



(a) (b) (c)

Figure 26: Analysis of different damage mesh scales (the last one is used as the damage geometry representation): (a) original size, (b) scaled to 100.2%, (c) scaled to 100.5%.

The proposed IFC structure succeeded in representing all the detected damages. Using the simple tree structure in IFC viewer, the damage structure

and semantics can easily be navigated and selected for further inspection by the user. Due to a large amount of added data (e.g., coordinates of mesh vertices, mesh indices), the IFC file size increased considerably after the semantic enrichment. The size of the *as-built* IFC file is 4.39 MB, whereas the size of the *as-is* file is 13.1 MB.

Detection System Accuracy.
For the system accuracy assessment, the BrIM enriched by automatically detected damages is compared against the baseline BrIM. Comparison of damages in BrIM was twofold: semantic and geometric. The semantics of damage information is analyzed by comparing the number of damages of the same type in each model, whereas the geometry is analyzed by comparing the missing volume due to damages for each bridge element. Results of empirical observation are presented in Tables 2 and 3.

Table 2: Detection accuracy assessment, with respect to occurrences of specific damages.

Damage Catalog Hierarchy Code	Damage Description	Baseline Occurrences	Automatically Detected Occurrences	Detection Success
1303	Spalling	9	7	78%
1304	Chipped off patched spots	1	0	0%
2204	Slightly corroded Reinforcement	3	1	33%
1314	Loss of chippings	7	5	71%
2205	Strongly rusted reinforcement	6	5	83%
1313	Chipping-missing pieces	1	1	100%
1307	Fractured reinforcement	4	4	100%

Table 3: Detection accuracy assessment, with respect to elements missing due to damages.

Damaged Element	Missing volume (cm^3) - Baseline	Missing volume (cm^3) - Automatically Detected	Detection Success
West curb	721426.6	430511.0	60%
East curb	87018.3	5049.8	6%
West girder	151670.6	61570.7	41%
East girder	127294.6	112051.0	88%
North abutment	17481.4	7236.2	41%
South abutment	1485.0	1485.0	100%

Economic viability of the proposed approach.
The inspection duration analysis presented in Table 4 includes the photogrammetric survey and point cloud generation. The analysis does not include the *as-designed* BrIM generation because it is a one-time procedure. The total duration of the inspection was approximately 68.1 hours, of which 56.4 hours were spent on activities dominantly performed by computer. All computer processing is performed on a workstation PC with the following hardware specifications:

- CPU: AMD FX 8350

- GPU: ASUS STRIX-GTX980
- Memory: 32GB

The traditional routine inspection significantly varies in duration, depending on the type of bridge structure, the number of spans, deck area, minimum applicable condition rating, time of the year, access equipment and inspector. The average normalized duration of an inspection of reinforced concrete slab bridges is 462 min/1000 ft^2 [67]. The inspection of the Bridge over river Gročica (133 m^2 = 1440 ft^2) therefore, would have taken 665 min, or 11 hours. Besides the fieldwork, this duration refers to the report writing.

Table 4: Inspection duration (proposed approach).

Process activity	Software components	Dominantly engaged	Domain of expertise	Duration (h)
Photogrammetric survey	None	Human labor	UAV photogrammetry	5.0
Point cloud generation	Agisoft Photoscan	Computer	Photogrammetry	38.0
BrIM geometry conversion to 3D triangular mesh	IfcOpenShell	Computer	BIM	5.0
Point cloud registration to BrIM-based mesh	CloudCompare	Human labor	Computer graphics	1.5
Multiview classification	Prototype web-based app	Computer	Computer graphics, machine learning	0.4
Geometric reconstruction of damage clusters	MeshLab	Computer	Computer graphics	1.0
CSG subtraction of reconstructed damage cluster from IFC-based mesh	MeshLab	Computer	Computer graphics	2.0
Triangular geometric reconstruction from 3D point cloud	MeshLab	Computer	Computer graphics	10.0
As-designed adjustment to as-built BrIM	MeshLab, AutoCad, Revit	Human labor	Structural engineering, BIM	4.0
Identifying the appropriate damage types in the BMS catalog	None	Human labor	Structural engineering	0.2
Injecting the damage into BrIM	Prototype app	Human labor	BIM	1.0
			Total:	68.1

To perform the proposed process in this case study took six times longer than the traditional one would. Nevertheless, whereas the traditional process requires permanent human labor engagement, most of the work (83%) in the proposed process is done by a computer. The generation of the photogrammetry-based point cloud is certainly the most time-consuming activity in the process, taking more than half of the total inspection time. Using the terrestrial laser scanning (TLS) instead of photogrammetry to acquire the point cloud is indeed much faster and more accurate. One drawback when using a TLS to capture a point cloud is a lack of encoded RGB values, which is crucial for damage detection using multiview classification. Additionally, the duration of a UAV-based inspection undoubtedly depends on the bridge size. Using UAVs is efficient in reducing the inspection time when inspecting large bridges, whereas small bridges are faster inspected in the traditional manner [8].

In 2012, 16.4 billion USD was spent on rehabilitation or replacement of existing highway bridges in the USA [68]. According to Zulfiqar *et al.* [69], the USA spends only 2.7 billion USD per year on routine bridge inspections. Rather than reducing the inspection cost, this research intended to reduce

the several times higher maintenance cost, by providing the inspection deliverable with enough information for proper maintenance prioritization. The current inspection deliverable is a paper report with condition ratings, loosely related to the bridge KPIs (e.g., safety and serviceability). Even the point cloud with marked damages is a much more reliable basis for making decisions on maintenance interventions or bridge closure [8]. In comparison to a simple point cloud, the output of the proposed process, *as-is* BrIM is the object-oriented model of both the bridge and damages, with accompanying BMS semantics. Structured this way, the damage information is ready to be utilized in the calculation of damaged bridge KPIs, the base for a more objective maintenance prioritization.

Finally, there are certain legal limitations for the application of the proposed inspection process. Most country's aviation authorities, such as the Federal Aviation Administration (FAA), forbid UAV operating over a moving vehicle [70]. Due to this kind of regulation, although technically possible, the utilization of UAV in bridge inspection is a bit limited. The first limitation is related to the pavement inspection, and the second one refers to the inspection of the underside of an overpass. However, the traffic lane closure during the inspection eliminates all legal issues of this kind.

7. Conclusions and Future Work

The paper shows a rational and practice-oriented method to develop a new generation of BMS by incorporating inspection findings into a BrIM model. Updated with *as-is* information about the bridge, the BrIM would reflect the current bridge condition more accurately. Targeting large spallings, accompanied by extensively corroded reinforcement, abutments, and curbs, the presented approach managed to detect occurrences of five damage types out of seven, with the success rate greater than 70% (Table 2). Furthermore, the previously generated *as-built* BrIM of the case study bridge is successfully enriched by the damage information according to the proposal. The detected damage clusters are injected into the *as-built* IFC geometry via CSG Boolean operations, and related textual semantics following the requirements of a damage classification from the Swiss BMS KUBA — including the requirements of the IFC 2x4 schema. Rather than proposing a schema extension, the existing schema definitions are used to describe the damage type, extent, and severity — as well to group them according to the location and causing deterioration process.

To detect damages other than spalling, the current binary classification could be refined to include other damage types. However, the resolution of the bridge point cloud should be reasonably higher than the one used in the case study. Such a point cloud could be generated using a combination of laser scanning and photogrammetry. Damages detected this way could then be geometrically represented in a BrIM, by mapping image segments depicting damages onto damaged elements as textures [46].

Since the damage data structure and semantics definitions proposed in the presented approach comply with the selected BMS structure, there are two ways to utilize it in the existing BMS: (1) It is possible to either apply the proposed approach on an external IFC file and simply link it with BMS, and (2): One can insert IFC representation of every specific bridge into the BMS. The latter, a more robust method, would require certain extensions of the current BMS software, such as an IFC viewer or custom tools for point cloud processing and damage detection.

Once the data is acquired and properly integrated into BrIM, it can be used as a basis for the straightforward assessment of bridge KPIs such as safety and serviceability. Research by Isailović *et al.* proposes using Bayesian nets to assess the probability of bridge failure based on inspection findings [71]. Damage location and severity are roughly estimated and introduced to Bayesian mesh by manually updating node values. Having the BrIM that contains all the damage information, nodes of the Bayesian mesh can be automatically updated.

Therefore, accurate and exhaustive damage information contained in the BrIM can be introduced to a finite element model representation of the bridge (eventually resulting in an accurate and up-to-date structural analysis of the bridge). Such an analysis would directly provide the most important bridge KPIs: *safety* and *serviceability*.

Besides the benefits listed above, the limitations of the proposed approach should be pointed out. Due to both the legal and physical limitations, UAVs cannot always be used in bridge inspection without road(s) closure, implying additional costs. In the proposed approach, the *as-designed* BrIM is a pre-requisite. However, a minority of existing bridges have BrIM representation, so the creation of such models by using BIM authoring tools is required. Lack of digital drawings or even the paper ones makes this task rather difficult. Another requirement of the proposed approach is highly precise registration of point cloud to IFC-based geometry. For that reason, manual registration is proposed, rather than the automated one. Although the registration pre-

cision is achieved, this task increased the total inspection duration for 1.5 hours. Even with precise registration, to be able to detect fine damage, such as crack, the proposed approach would require an ultrahigh-resolution point cloud. Last, but not least, the case study showed that the proposed approach in the bridge inspection lasts approximately seven times longer than the traditional inspection.

In terms of future work concerning the multiview classification approach, there are improvements and advancements that can be made:

First, the amount of training data used to retrain the CNN can be greatly increased by either using existing images of various types of structural damage featured in bridges, or using the approach of synthetic data generation. The use of synthetic training data generation can be beneficial in increasing the classification accuracy, as a larger number of training images can be generated using photo-realistic renderings of structural damage features. Such approaches are already being used in the autonomous driving research community for generating training data containing various built environment features (e.g., buildings, road signs, traffic lights, road markings, etc) [72].

Second, while the use of a 2D CNN has proven to be practical along with acceptable accuracy, it could be argued that another approach would be to use a 3D CNN for training and classification of point cloud representations of structural damage. Modern 3D CNN aimed at semantic segmentation, such as PointNet++, can be utilized for such tasks [73].

Third, the image-based classification could theoretically be performed right after the image capture stage and before the 3D point cloud generation stage. Therefore the generated point cloud model could already contain point with visual or encoded semantics indicating the presence of potential damage features. The system would then just need to extract these clusters without needing to classify the point cloud itself, and reconstruct them into geometric representations that would be used for semantic injection into as-is BIM model via CSG operations.

Fourth, the use of viewpoint entropy can be combined with the existing multiview classification approach, to better evaluate the visual information contained in each multiview image prior to classification (so that only images containing adequate and useful visual information get classified) [74].

Finally, the methods presented in this research are applicable to a much broader range of challenges in designing, building and maintaining the built environment, including the processing of large point clouds to compile BIM models and to detect damage, a compilation of *as-built* and *as-is* models with

explicit geometry and semantics. Further development of this research aims to contribute to the acquiring and use of Digital Twins (DTs) for managing the built environment.

Acknowledgements

Work contributed by Dušan Isailović is partly funded by the Ministry of Education, Science and Technological Development of the Republic of Serbia under grant TR-36038. It is a part of the project *Development of the method for the production of MEP design and construction documents compatible with BIM process and related standards.*

Work contributed by Vladeta Stojanovic is partially funded by the Research School on Service-Oriented Systems Engineering of the Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam.

The authors thank Peter Bonsma and RDF for providing the IFCEngine software library along with support. The authors also thank the Public Enterprise Roads of Serbia for providing the graphic documentation of the case study bridge, and Dr. Nikola Tanasić for providing photos of various Serbian bridges, which were used as a source for the CNN training set.

References

[1] Eurostat. Modal split of inland passenger transport. URL: `https://ec.europa.eu/eurostat/statistics-explained/index.php?title=File:Modal_split_of_inland_passenger_transport,_2015_(%25_of_total_inland_passenger-kilometres).png`, last accessed: 13/9/2019, 2015.

[2] Eurostat. Freight transport in the EU-28 modal split of inland transport modes. URL: `https://ec.europa.eu/eurostat/statistics-explained/index.php?title=File:Freight_transport_in_the_EU-28_modal_split_of_inland_transport_modes_(%25_of_total_tonne-kilometres)_2016.png#file`, last accessed: 13/9/2019, 2018.

[3] Stefan Schultz and Patrick Stotz. Marode Fernstrassen: Hier zerbröseln Deutschlands Brücken, 2016. URL: `https://www.spiegel.de/wirtschaft/soziales/`

deutschland-hier-sind-deutschland-bruecken-marode-a-1080431.html, last accessed: 15/9/2019.

[4] Zanyar Mirzaei, Bryan T Adey, L Klatter, and PD Thompson. The IABMAS bridge management committee overview of existing bridge management systems. Technical report, International Association for Bridge Maintenance and Safety (IABMAS), 2014. URL: https://www.researchgate.net/publication/322754699_Overview_of_existing_Bridge_Management_Systems_-_Report_by_the_IABMAS_Bridge_Management_Committee, last accessed: 18/9/2019.

[5] Graphisoft. BIM Curriculum. URL: https://www.graphisoft.com/learning/bim-curriculum/, last accessed: 13/9/2019, 2015.

[6] Rafael Sacks, Ury Gurevich, and Prabhat Shrestha. A review of building information modeling protocols, guides and standards for large construction clients. *Journal of Information Technology in Construction (ITcon)*, 21(29):479–503, 2016. URL: = https://www.itcon.org/papers/2016_29.content.00509.pdf, last accessed: 13/9/2019.

[7] Rafael Sacks, Amir Kedar, André Borrmann, Ling Ma, Ioannis Brilakis, Philipp Hüthwohl, Simon Daum, Uri Kattel, Raz Yosef, Thomas Liebich, et al. SeeBridge as next generation bridge inspection: overview, information delivery manual and model view definition. *Automation in Construction*, 90:134–145, 2018. DOI: 10.1016/j.autcon.2018.02.033.

[8] Jennifer Wells and Barritt Lovelace. Improving the Quality of Bridge Inspections Using Unmanned Aircraft Systems (UAS). Technical report, Minnesota Department of Transportation, 2018. URL: https://www.dot.state.mn.us/research/reports/2018/201826.pdf, last accessed: 12/9/2019.

[9] Mohammad R Jahanshahi and Sami F Masri. Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures. *Automation in Construction*, 22:567–576, 2012. DOI: 10.1016/j.autcon.2011.11.018.

[10] Stephanie German, Ioannis Brilakis, and Reginald Desroches. Rapid entropy-based detection and properties measurement of concrete

spalling with machine vision for post-earthquake safety assessments. *Advanced Engineering Informatics*, 26(4):846–858, 2012. DOI: 10.1016/j.aei.2012.06.005.

[11] Guido Morgenthal, Norman Hallermann, Jens Kersten, Jakob Taraben, Paul Debus, Marcel Helmrich, and Volker Rodehorst. Framework for automated UAS-based structural condition assessment of bridges. *Automation in Construction*, 97:77–95, 2019. DOI: 10.1016/j.autcon.2018.10.006.

[12] Amos Sironi, Vincent Lepetit, and Pascal Fua. Multiscale centerline detection by learning a scale-space distance transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2697–2704, 2014. DOI: 10.1109/CVPR.2014.351.

[13] Philipp Hüthwohl and Ioannis Brilakis. Detecting healthy concrete surfaces. *Advanced Engineering Informatics*, 37:150–162, 2018. DOI: 10.1016/j.aei.2018.05.004.

[14] Yiye Xu and Yelda Turkan. Bridge inspection using bridge information modeling (brim) and unmanned aerial system (uas). In *Advances in Informatics and Computing in Civil and Construction Engineering*, pages 617–624. Springer, 2019. DOI: 10.1007/978-3-030-00220-6_74.

[15] Tan Qu and Wei Sun. Usage of 3D Point Cloud Data in BIM (Building Information Modelling): Current Applications and Challenges. *Journal of Civil Engineering and Architecture*, 9(11):1269–1278, 2015. DOI: 10.17265/1934-7359/2015.11.001.

[16] Sebastian Tuttas, Alexander Braun, André Borrmann, and Uwe Stilla. Acquisition and consecutive registration of photogrammetric point clouds for construction progress monitoring using a 4D BIM. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 85(1):3–15, 2017. DOI: 10.1007/s41064-016-0002-z.

[17] Engin Burak Anil, Pingbo Tang, Burcu Akinci, and Daniel Huber. Deviation analysis method for the assessment of the quality of the as-is Building Information Models generated from point cloud data. *Automation in Construction*, 35:507–516, 2013. DOI: https://doi.org/10.1016/j.autcon.2013.06.003.

[18] Hichem Barki, Fodil Fadli, Ahmed Shaat, Pawel Boguslawski, and Lamine Mahdjoubi. BIM models generation from 2D CAD drawings and 3D scans: an analysis of challenges and opportunities for AEC practitioners. *Building Information Modelling (BIM) in Design, Construction and Operations*, 149:369–380, 2015. DOI: 10.2495/BIM150311.

[19] Charles R Farrar and Keith Worden. *Structural health monitoring: a machine learning perspective*. John Wiley & Sons, 2012. ISBN: 978-1-119-99433-6.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436 – 444, 2015. DOI: 10.1038/nature14539.

[21] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3d data: A survey. *ACM Computing Surveys (CSUR)*, 50(2):20, 2017. DOI: 10.1145/3042064.

[22] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*, pages 265–283, 2016. ISBN: 978-1-931971-33-1.

[23] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. ISBN: 9781467388528.

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. ISBN: 9781538607343.

[25] Hyunjun Kim, Eunjong Ahn, Myoungsu Shin, and Sung-Han Sim. Crack and noncrack classification from concrete surface images using machine learning. *Structural Health Monitoring*, 2018. DOI: 10.1177/1475921718768747.

[26] Christian Koch, Zhenhua Zhu, Stephanie German Paal, and Ioannis Brilakis. Machine vision techniques for condition assessment of civil infrastructure. In *Integrated Imaging and Vision Techniques for Industrial Inspection*, pages 351–375. Springer, 2015. ISBN: 978-1-4471-6741-9.

[27] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In *Proceedings of British Machine Vision Conference (BMVC)*, volume 12, 2017. URL: `https://dblp.org/rec/bib/conf/bmvc/2017`, last accessed: 18/9/2019.

[28] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. ISBN: 978-1-4673-8392-9.

[29] Vladeta Stojanovic, Matthias Trapp, Rico Richter, and Jürgen Döllner. A service-oriented approach for classifying 3d points clouds by example of office furniture classification. In *Proceedings of the 23rd International ACM Conference on 3D Web Technology*, page 2. ACM, 2018. DOI: 10.1145/3208806.3208810.

[30] Andreas Dietze, Marcel Klomann, Yvonne Jung, Michael Englert, Sebastian Rieger, Achim Rehberger, Silvan Hau, and Paul Grimm. Smulgras: a platform for smart multicodal graphics search. In *Proceedings of the 22nd International Conference on 3D Web Technology*, page 17. ACM, 2017. DOI: 10.1145/3055624.3075942.

[31] C Bradford Barber, David P Dobkin, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996. DOI: 10.1145/235815.235821.

[32] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. ISBN:3-905673-36-3.

[33] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. DOI: 10.1109/2945.817351.

[34] Julie Digne, David Cohen-Steiner, Pierre Alliez, Fernando De Goes, and Mathieu Desbrun. Feature-preserving surface reconstruction and simplification from defect-laden point sets. *Journal of mathematical imaging and vision*, 48(2):369–382, 2014. DOI: 10.1007/s10851-013-0414-y.

[35] James D Foley, Foley Dan Van, Andries Van Dam, Steven K Feiner, John F Hughes, J Hughes, and Edward Angel. *Computer graphics: principles and practice*. Addison-Wesley Professional, 1996. ISBN: 0201848406.

[36] Nouha Hichri, Chiara Stefani, Livio De Luca, Philippe Veron, and Gael Hamon. From point cloud to BIM: a survey of existing approaches. In *XXIV International CIPA Symposium*. Proceedings of the XXIV International CIPA Symposium, 2012. DOI: 10.5194/isprsarchives-XL-5-W2-343-2013.

[37] Brendan McGuire, Rebecca Atadero, Caroline Clevenger, and Mehmet Ozbek. Bridge information modeling for inspection and evaluation. *Journal of Bridge Engineering*, 21(4):04015076, 2016. DOI: 10.1061/(ASCE)BE.1943-5592.0000850.

[38] Bentley Systems, Exton, PA. LEAPBridge [Computer software]. 2019.

[39] Tekla, Espoo, Finland. Tekla Structures 17.0 [Computer software], 2011.

[40] Autodesk, San Rafael, CA. Autodesk Revit version 19.0.1.1 [Computer software]. 2019.

[41] Steve B Chase, Y Adu-Gyamfi, AE Aktan, and E Minaie. Synthesis of National and International Methodologies Used for Bridge Health Indices. Technical report, Office of Infrastructure Research and Development, Federal Highway Administration, 2016. URL: `https://www.fhwa.dot.gov/publications/research/infrastructure/structures/bridge/15081/15081.pdf`, last accessed: 12/9/2019.

[42] International Organization for Standardization. ISO 10303-11:2004 Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual. URL: `https://www.iso.org/standard/38047.html`, last accessed: 13/9/2019, 2004.

[43] André Borrmann, Markus König, Christian Koch, and Jakob Beetz. Building Information Modeling: Technology Foundations and Industry Practice. Springer, 2018. ISBN: 978-3-319-92862-3.

[44] Nobuyoshi Yabuki, Eric Lebegue, Jean Gual, Tomoaki Shitani, and Li Zhantao. International Collaboration for Developing the Bridge Product Model "IFC-Bridge". In *Proceedings of the 11th Joint International Conference on Computing and Decision Making in Civil and Building Engineering*, pages 1927–1936, 2006. URL: `http://itc.scix.net/pdfs/w78-2006-tf289.pdf`, last accessed: 12/9/2019.

[45] Andre Borrmann, Sergej Muhic, Juha Hyvarinen, Tim Chipman, Stefan Jaud, Christophe Castaing, Claude Dumoulin, Thomas Liebich, and Laura Mol. The IFC-Bridge project - Extending the IFC standard to enable high-quality exchange of bridge information models. In *Proceedings of 2019 European Conference on Computing in Construction*, pages 377 – 386, 2019. DOI: 10.35490/EC3.2019.193.

[46] Philipp Hüthwohl, Ioannis Brilakis, André Borrmann, and Rafael Sacks. Integrating RC bridge defect information into BIM models. *Journal of Computing in Civil Engineering*, 32(3):04018013, 2018. DOI: 10.1061/(ASCE)CP.1943-5487.0000744.

[47] Philipp Hüthwohl, Ruodan Lu, and Ioannis Brilakis. Multi-classifier for reinforced concrete bridge defects. *Automation in Construction*, 105:102824, 2019. DOI: 10.1016/j.autcon.2019.04.019.

[48] D Isailović M Petronijević and R Hajdin. The future of BIM and Bridge Management Systems. In *IABSE Symposium 2019: Towards a Resilient Built Environment - Risk and Asset Management*, pages 1673 – 1680. International Association for Bridge and Structural Engineering (IABSE), 2019. ISBN: 978-1-5108-8445-8.

[49] Object Management Group. Business Process Model and Notation (BPMN 2.0). URL: `http://www.bpmn.org/`, last accessed: 18/9/2019, 2011.

[50] Thomas Krijnen. IfcOpenShell. URL: `http://ifcopenshell.org/`, last accessed: 18/9/2019, 2018.

[51] Daniel Girardeau-Montaut. Cloudcompare - open source project [computer software]. URL: https://www.danielgm.net/cc/, last accessed: 18/9/2019, 2011.

[52] BIM Forum. Level of Development Specification. URL: https://bimforum.org/lod/, last accessed: 12/9/2019, 2014.

[53] Vladeta Stojanovic, Matthias Trapp, Rico Richter, and Jürgen Döllner. Service-oriented semantic enrichment of indoor point clouds using octree-based multiview classification. *Graphical Models*, page 101039, 2019. DOI: 10.1016/j.gmod.2019.101039.

[54] Niloy J Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328. ACM, 2003. DOI: 10.1145/777792.777840.

[55] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. 26(2), 1992. URL: http://hhoppe.com/recon.pdf, last accessed: 13/9/2019.

[56] Hossam ElGindy, Hazel Everett, and Godfried Toussaint. Slicing an ear using prune-and-search. *Pattern Recognition Letters*, 14(9):719–722, 1993. DOI: https://doi.org/10.1016/0167-8655(93)90141-Y.

[57] Rafael Sacks, Charles Eastman, Ghang Lee, and Paul Teicholz. *BIM handbook: a guide to building information modeling for owners, designers, engineers, contractors, and facility managers*. John Wiley & Sons, 2018. ISBN: 9781119287544.

[58] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136, 2008. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[59] Federal Roads Office FEDRO, Bern, Switzerland. KUBA 5.0 [Computer software], 2019.

[60] Rade Hajdin. KUBA Version 4.0. In *IABSE Symposium Report*, volume 91, pages 9–16. International Association for Bridge and Structural Engineering, 2006. DOI: 10.2749/222137806796235962.

[61] Rade Hajdin. KUBA 4.0: The Swiss Road Structure Management System. In *Proceedings of 10th International Bridge and Structure Management Conference, Buffalo, New York, October 20–22. Buffalo, New York*, 2008. DOI: 10.17226/17628.

[62] S Daum and A Borrmann. Simplifying the Analysis of Building Information Models Using tQL4BIM and vQL4BIM. In *Proceedings of the EG-ICE workshop 2015*, 2015. ISBN: 9781510809567.

[63] buildingSMART International. IFC4 Addenum 2 Specification. URL: `https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/`, last accessed: 13/9/2019, 2016.

[64] Ricardo Cabello et al. Three. js. URL: `https://github.com/mrdoob/three.js`, last accessed: 18/9/2019, 2010.

[65] Sóren Discher, Rico Richter, and Jürgen Döllner. Concepts and techniques for web-based visualization and processing of massive 3d point clouds with semantics. *Graphical Models*, page 101036, 2019. DOI: 10.1016/j.gmod.2019.101036.

[66] RDF. IFC Engine [dynamic library]. URL: `http://rdf.bg/product-list/ifc-engine/`, last accessed: 13/9/2019, 2018.

[67] Emal Masoud, Abigail Clarke-Sather, and Jennifer McConnell. Lean construction applications for bridge inspection. Technical report, 2017. URL: `https://cait.rutgers.edu/wp-content/uploads/2018/05/cait-utc-nc24-final.pdf`, last accessed: 18/9/2019.

[68] Robert S. Kirk and William J. Mallett. Highway bridge conditions: Issues for congress. Technical report, Congressional Research Service, 2018. URL: `https://fas.org/sgp/crs/misc/R44459.pdf`, last accessed: 9/11/2019.

[69] Ahsan Zulfiqar, Miryam Cabieses, Andrew Mikhail, and Namra Khan. Design of a bridge inspection system (bis) to reduce time and cost. Technical report, Department of Systems Engineering and Operations

Research George, Mason University, 2014. URL: https://catsr.vse.gmu.edu/SYST490/490_2014_BI/BIS_FinalReport.pdf, last accessed: 19/12/2019.

[70] Small Unmanned Aircraft System Aviation Rulemaking Committee. Comprehensive Set of Recommendations for sUAS Regulatory Development. URL: https://www.faa.gov/regulations_policies/rulemaking/committees/documents/media/SUASARC-4102008.pdf, last accessed: 13/9/2019, 2009.

[71] D Isailović R Hajdin, and J Matos. Bridge quality control using Bayesian net. In *40th IABSE Symposium 2018: Tomorrow's Megastructures*, pages S27–51. International Association for Bridge and Structural Engineering (IABSE), 2018. ISBN: 978-1-5108-7385-8.

[72] David Griffiths and Jan Boehm. Synthcity: A large scale synthetic point cloud. *CoRR*, abs/1907.04758, 2019. URL: http://arxiv.org/abs/1907.04758.

[73] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5105–5114, 2017. ISBN: 978-1-5108-6096-4.

[74] Vladeta Stojanovic, Matthias Trapp, Jürgen Döllner, and Rico Richter. Classification of indoor point clouds using multiviews. In *The 24th International Conference on 3D Web Technology*, pages 1–9. ACM, 2019. DOI: 10.1145/3329714.3338129.