Zhang, Youqi; Lin, Weiwei

Computer-vision-based differential remeshing for updating the geometry of finite element model

Published in:
Computer Aided Civil and Infrastructure Engineering

DOI:
10.1111/mice.12708

Published: 01/02/2022

DOI: 10.1111/mice.12708

RESEARCH ARTICLE

COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING WILEY

Computer-vision-based differential remeshing for updating the geometry of finite element model

Youqi Zhang | **Weiwei Lin**

Department of Civil Engineering, School of Engineering, Aalto University, Espoo, Finland

Correspondence
Weiwei Lin, Department of Civil Engineering, School of Engineering, Aalto University, P.O. Box 12100, 00076 Aalto, Finland.
Email: weiwei.lin@aalto.fi

Abstract
This article presents a method that enables a finite element (FE) model to remesh itself for updating the geometric changes caused by structural damages, using computer vision (CV) techniques and geometric analyses. Currently, there is no mature automatic approach to utilize the information of structural damage detection (SDD) for structural state awareness. Thus, the purpose of this study is to automate the pipeline from the accomplishment of SDD to numerical analyses for fast structural capacity evaluation. CV techniques are used to determine the shapes and dimensions of both structural components and damages. Geometric analyses are used to develop the algorithms for automatically deleting, generating, and splitting the elements of FE models for updating the geometric changes. Experiments are performed on a plate and a C-shaped steel crossbeam of a bridge to demonstrate the effectiveness of the proposed method and algorithms.

1 | INTRODUCTION

The aging civil infrastructures have already become a serious problem for many countries, which keep threatening public safety and request a large amount of cost for rehabilitation (ASCE, 2017; MLIT, 2018; VÄYLÄ, 2018). Thus, the need for quantifying the reserve capacity of the aging civil infrastructures and investigating their structural health state is growing rapidly, and such need boosts the development of structural health monitoring (SHM) technology to overcome the aging problem of civil infrastructures. In general, two main steps are required to estimate the structural capacities of civil infrastructure in the existing SHM schemes: (1) Identify the damages of the civil infrastructure, (2) quantify the reserved capacity of the civil infrastructure via numerical analysis based on the results of damage identification.

Currently, most of the research in the SHM field is focused on the first step: Developing the structural damage detection (SDD) methodologies with higher accuracy and robustness via various approaches, like computer vision (CV), vibration, natural modes, and strain. For instance, Rafiei and Adeli (2017, 2018) proposed vibration-based unsupervised machine learning models for global and local health condition assessment of structures. Li and Kurata (2019) proposed a mode-based method for assessing seismic damage on beam-column connections. Vibration-based damage detection methods for highrise buildings are also developed via signal processing (Amezquita-Sanchez & Adeli, 2015; Jiang & Adeli, 2007). Sajedi and Liang (2020) proposed a fully convolutional encoder-decoder neural network for vibration-based semantic damage segmentation for large-scale SHM. Comparing the CV-based SHM systems to the other SHM systems, visual information of structures is more intuitive since structural behaviors like displacement, damage, and deformation can be directly identified from the visual data with certainty. By using CV algorithms, researchers have already developed several measurement methodologies in the SHM systems, such as the measurement methodologies of displacement (Dong et al., 2019; Feng & Feng, 2017; Khuc & Catbas, 2017b; Lydon et al., 2019), vibration and natural modes (Chen et al., 2018; Dong et al., 2019; Khuc & Catbas, 2017a). By now many CV-based SDD methodologies have also been developed. Especially after the deep learning (DL) wave in 2015 (LeCun et al., 2015), developing CV-based SDD methods using DL is in the spotlight of the researchers' interests. Many successful DL models have been proposed for different damage detection tasks (Cha et al., 2017, 2018; Huang et al., 2018; Jahanshahi et al., 2017; S. Li et al., 2019; Pan & Yang, 2020; Xu et al., 2019; A. Zhang et al., 2017, 2019; C. Zhang et al., 2020). Those DL models can complete SDD tasks in an end-to-end manner. In other words, if a raw photo is fed into such a DL model, damage(s) can be detected and presented as the output without any intermediate steps or human intervention. Meanwhile, DL models normally show sound performance in complex engineering scenarios. For example, when using a moving camera (either on a drone or a vehicle) to capture images or videos of some structures (Gopalakrishnan et al., 2018; Lee et al., 2018; Liu et al., 2020; Salaan et al., 2018), the rapidly changing foreground and background greatly increase the complexity of image or video data, which makes DL almost become a standard solution for SDD tasks. In a word, DL makes the CV-based damage detection process simpler and more practical to use in real-life scenarios.

However, only SDD methods are not sufficient to quantify the structural health states. Questions such as "how severe the detected damages will affect the functionality of the bridge" and "how much capacity is left for the bridge to use" are still rarely answered. If evaluating the residual loading capacities and identifying the health states of the structures are the purposes, further developments (step 2 as mentioned) are needed based on the damage detection results. For instance, prognostics and health management methodologies (Chiachío et al., 2015, 2019; Cristiani et al., 2021) were developed based on probabilistic models, which can account for the model uncertainty and predicting structural remaining life. Currently, the most common way is manually creating finite element (FE) models and updating the detected damages to quantify the degradation of structural capacity, which is very time-consuming and may delay the awareness of safety risks. Therefore, it is in high demand to consider the two main steps as a whole to develop algorithms for automatically updating the FE model as the digital twin of the real structure, with the ability to present the health state of the structure in real time.

Nevertheless, such an idea is still at a very early stage since automatic damage modeling and updating on existing FE models request the works on damage identification, quantification, localization, alignment, meshing of the identified structural damages, and even further development of FE modeling methods to solve complex engineering problems (Adeli & Yu, 1995; Yu & Adeli, 1993). As a start, some studies attempt to utilize CV techniques such as using 3-D point cloud data to generate the FE model of the entire structure with tetrahedral solid elements, where only a set of image data are needed. Generally, 3-D point cloud data can be obtained either from images or video data by the structure from motion algorithms (Agarwal et al., 2011; Schönberger & Frahm, 2016) or laser scanners. Fernandez et al. (2016) created an FE model of a corroded steel bar by using its 3-D point cloud data. Castellazzi et al. (2015) proposed a semi-automatic procedure to transform the 3-D point cloud data of a building to an FE model and validated the performance of the generated FE model. Korumaz et al. (2017) presented an approach to transfer point cloud data into an FE model for the health assessment of a historical structure. Clement et al. (2004) developed a workflow to generate the FE model of a human single-rooted tooth by using computed tomography (CT) image data. Tajima et al. (2009) also transferred the CT data of a tooth into an FE model. Such methods require: (1) Large amounts of computation and processing time; (2) many manual operations and interventions; and (3) specific monitoring equipment or strategy to collect suitable data for point cloud data generation, which makes such methods difficult to be fully automatic and infeasible in real-time analysis.

To the best of the authors' knowledge, using CV techniques for updating the geometric changes caused by damages on FE models directly is rarely investigated. Ghahremani et al. (2018) proposed a method using two sets of point cloud data (before and after the damage occurs) to detect the structural damage, quantify the structural change, and generate a solid model of the damage. By importing the solid model into an FE mesh generator, tetrahedral solid elements of the damage were obtained. Then the FE model of the damage was used to replace the corresponding region of the FE model. The result is very encouraging since the method can update the geometry of the FE model to consider the structural damage. However, the method requests a point cloud data of the original structure, and it still needs several manual works to obtain the updated FE model. These requirements became obstacles to make the method fully automatic due to the complexity of the computation of point cloud data.

Point cloud data can well present the depth information, which is essential for analyzing the depths of structural damages on concrete structures. However, if the structures or structural components (e.g., steel beams, concrete shells, etc.) are modeled by shell elements in the corresponding FE models, the depth information could be redundant in many cases, because the in-depth behavior is not relevant regarding crack propagation. When there

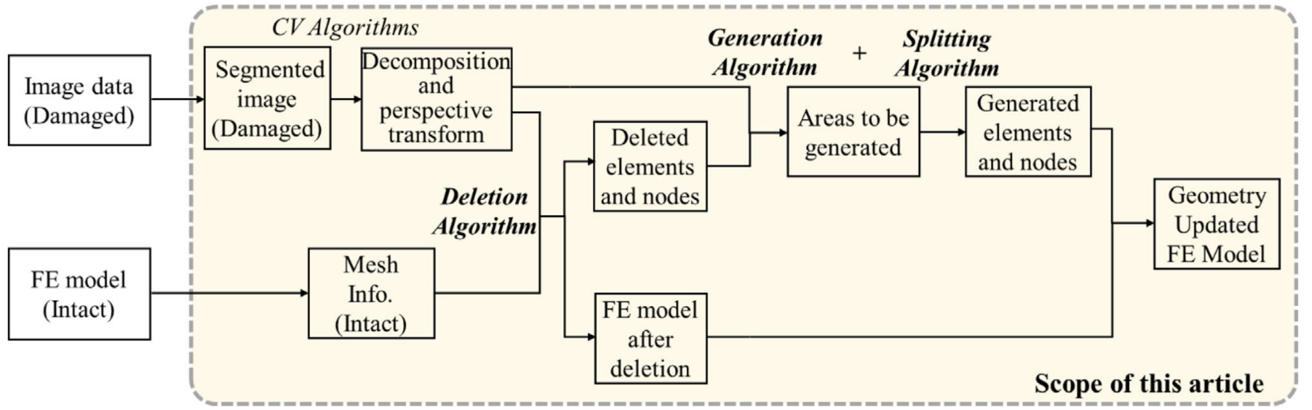


FIGURE 1 Architecture of the proposed real-time geometry updating method for finite element (FE) models

are cracks on such structures, the structural damages will throughout the entire thicknesses of the structures. In the above cases, there is no need to extract the depth information of structures or structural components from the visual data, and 2-D image data is the best option to capture the characteristics of the damages and structural components. Hence, it is necessary to develop a method to automatically update the geometric changes for the 2-D structures that can be simulated with shell elements by using image data. It can avoid the additional computational consumption of the point cloud, improve the computational efficiency, and make it possible to fully automate the procedure of estimating the remaining structural capacity.

Therefore, toward the goal of real-time damage updating on FE models, as the initial stage, this study aims to develop an automatic method to locally remesh FE models for updating the geometric changes of FE models, which are simulated or partially simulated by shell elements, via CV techniques and geometric analyses. To automate the pipeline from the accomplishment of SDD to numerical analyses for fast structural capacity evaluation, a method for automatically updating the geometric changes is proposed. To introduce and validate the proposed method, the article is organized into four main sections. First, the background of the research is presented. Second, the proposed method is described, including the applied CV techniques and the proposed algorithms for automatic deletion, generation, and splitting of shell elements. Third, three experiments are conducted to validate the performance of the proposed methodologies. Finally, the paper is finalized with some conclusions and discussions.

2 | METHODOLOGIES

2.1 | Overview of the method

To achieve CV-based differential remeshing, a method, as shown in Figure 1, is proposed. The method requests (1) segmented image data that represent the structural damages and components and the background, and (2) an FE model of the structure in an intact (or original) state. The segmented image can be obtained by training standard DL models for image segmentation, for instance: Fully convolutional networks (Long et al., 2015; S. Li et al., 2019; Narazaki et al., 2020), U-Net (X. Li et al., 2018), or by using labeling tools. Three algorithms in total, including the deletion, the generation, and the splitting algorithms are developed to constitute the main part of the proposed method.

The pipeline of the proposed method can be summarized into four steps. First, obtain the (1) segmented image representing the structural damages and components and the background and (2) the node list (NL) and element list (EL) of the intact FE model. Second, if the monitored structure is 3-D, decompose the structure or structural components in the segmented image into 2-D elements and perform the CV algorithms, respectively. Third, based on the meshing information of the FE model and the perspective-transformed image, use the proposed deletion algorithm to calculate the elements and nodes that need to be deleted, and obtain the FE model after deletion. Finally, use the proposed generation and splitting algorithms to obtain the FE model after new nodes and elements are generated.

The proposed method aims to be applied to all types of structures that can be simulated by using shell elements, for example, C-, T-, I-shaped, hollow steel beams, and so forth. In real engineering application scenarios, cameras will be installed to monitor the critical structural components, which can be determined by either structural redundancy analysis (Lin et al., 2017) or engineering experiences. If any damage is detected by the camera, the proposed method will use the result of damage detection for updating the geometry of the corresponding FE model directly. No doubt that extracting the geometric information of structure and damage accurately from image data is crucial for a successful modeling updating. Therefore, in actual engineering scenarios, multiple cameras can be

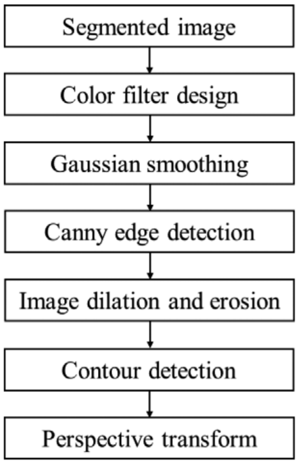


FIGURE 2 Flowchart of image processing

installed to capture image data of the critical structural component(s) from different perspectives and positions. Then by using the image data acquired from the camera sets, more accurate geometries of structure and damage can be obtained, compared to the cases using a single camera for data acquisition.

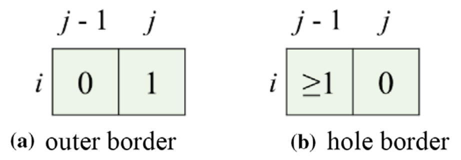


FIGURE 3 Condition of contour following starting point

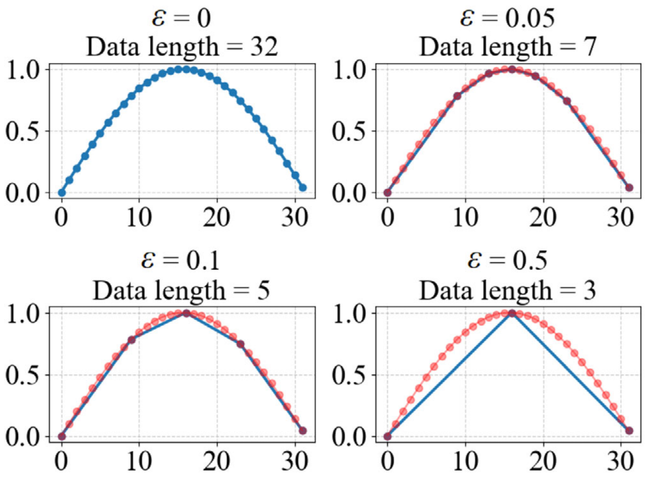


FIGURE 4 Example of the Ramer–Douglas–Peucker method

2.2 | CV algorithms

To extract the geometric information of both structural components and damages from the segmented image, a series of CV algorithms are applied in the proposed method. The CV algorithms include (1) color filter design by extracting a single channel of red green blue (RGB) images or hue, saturation, value (HSV) color space transform (Smith, 1978), (2) Gaussian smoothing to remove Gaussian noise from the image after filtering, (3) Canny algorithm (Canny, 1986) to detect the edges in images, (4) image dilation and erosion to improve the quality of the detected edges, (5) contour detection and simplification algorithms (Ramer, 1972; Suzuki & Abe, 1985) to determine the areas of both structural components and damages in the segmented images, and (6) perspective transform to display the structure and damages with almost undistorted scales. The flowchart of the CV algorithms is summarized in Figure 2. Because most of the utilized CV algorithms are standard methods, only contour detection, simplification algorithms, and perspective transform are introduced in this article.

2.2.1 | Contour detection

Contour is a continuous curve along the boundary of an object. The contour detection method (Suzuki & Abe, 1985) defines the starting point determination rule of the border following the outer and hole borders as shown in Figure 3. By scanning each pixel of an image from left to right and from top to bottom, the method can detect contours and find their topological structures.

Since there are too many redundant nodes in the detected contour, the Ramer–Douglas–Peucker algorithm (Ramer, 1972) is applied to simplify the contour with fewer points and line segments. The algorithm connects the first and last points of the curve with a straight line, calculates the distance between all the intermediate points and the straight line, and finds the maximum distance value d_{max}. Then d_{max} is compared with the threshold value: If $d_{max} <$ threshold ε, all the intermediate points on this curve are discarded. Otherwise, if $d_{max} \geq$ threshold ε, this point is taken as the boundary, the curve is divided into two parts, and the above process is repeated for the two parts of the curve until all points are processed. Figure 4 demonstrates the Ramer–Douglas–Peucker algorithm on a half period of sine waveform. The red curve is the original data, and the blue curve is after simplification. The assignment of ε will be very influential to the simplification result. The higher the ε, the fewer the nodes representing the curve.

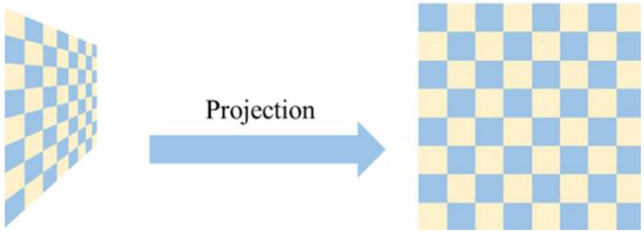


FIGURE 5 Example of perspective transform

2.2.2 | Perspective transform

Image taken from a casual perspective will lead to distortion of objects in the image. When using image data for damage quantification, the image data should be transformed into a better view to display the geometric relationship of the structure with more accurate dimensions. The procedure of perspective transform is illustrated in Figure 5. Perspective transform of images can be realized by applying the projection as Equation (1), in which x_i' and y_i' are the coordinates of pixels in the transformed image, and x_i and y_i are the coordinates of the corresponding pixels in the original image. $\boldsymbol{M}$, a 3×3 matrix as shown in Equation (2), is the coefficient matrix for perspective transform, which maps x_i and y_i to x_i' and y_i'. In $\boldsymbol{M}$, $M_{3,3}$ is determined to be 1, and the other eight coefficients can be obtained by solving the linear system in Equation (3), in which the coordinates of four pixels in both images are needed. Note that three of the pixels in each image should not be collinear. Then the corresponding coordinates of each pixel in the transformed image x_i' and y_i' can be calculated by using Equation (4). Finally, pixel interpolation is used to complete the perspective transform.

$$\left(x_i', y_i', 1\right)^T = M \cdot (x_i, y_i, 1)^T \tag{1}$$

$$M = \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -x_1x_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -x_2x_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -x_3x_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -x_4x_4' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1y_1' & -y_1y_1' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2y_2' & -y_2y_2' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y_3y_3' & -y_3y_3' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y_4y_4' & -y_4y_4' \end{bmatrix} \begin{bmatrix} M_{1,1} \\ M_{1,2} \\ M_{1,3} \\ M_{2,1} \\ M_{2,2} \\ M_{2,3} \\ M_{3,1} \\ M_{3,2} \end{bmatrix} = \begin{bmatrix} x_1' \\ x_2' \\ x_3' \\ x_4' \\ y_1' \\ y_2' \\ y_3' \\ y_4' \end{bmatrix} \tag{3}$$

ALGORITHM 1 The deletion algorithm of elements and nodes on FE models

```
Require: PDA, the polygon(s) of the damaged area(s)
Require: EL, element list of the intact FE model
Require: NL, node list of the intact FE model
Require: DEL, empty list for saving the deleted elements
  for element i in EL:
    if i ∩ PDA ≠ ∅:
      Add i to DEL
      Delete i from EL
  end for
  for j in DEL:
    for node k in j:
      if k does not constitute any element in EL:
        Delete k from NL
    end for
  end for
  Return EL, NL, DEL,
```

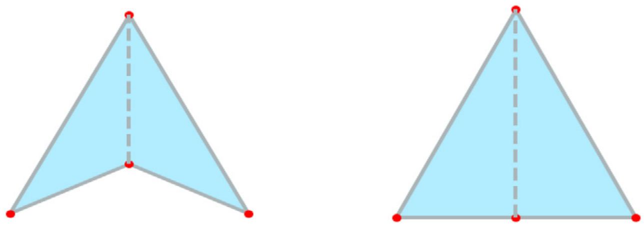
$$\begin{cases} x_i' = \dfrac{M_{1,1}x + M_{1,2}y + M_{1,3}}{M_{3,1}x + M_{3,2}y + M_{3,3}} \\ y_i' = \dfrac{M_{2,1}x + M_{2,2}y + M_{2,3}}{M_{3,1}x + M_{3,2}y + M_{3,3}} \end{cases} \tag{4}$$

2.3 | Algorithms for geometric analyses

2.3.1 | Deletion algorithm of nodes and elements

When deleting the nodes and elements in the damaged areas manually, we must first accurately measure the dimensions of the damage, then find the elements and nodes that have intersections with the damage, and finally delete those nodes and elements. The above three steps are very time-consuming if done manually. To achieve a more efficient way for deleting nodes and elements in FE models, the deletion algorithm is proposed, and the procedure is specified in Algorithm 1 .

The polygon represents the intact FE model and can be calculated by transferring all the elements into polygons and then calculating the union of all the polygons. The polygon(s) of the damaged area(s) (PDA) is acquired directly from the perspective-transformed segmented image through the CV algorithms. The EL and the NL of the intact FE model can be exported directly from the FE model software or script.



(a) (b)

FIGURE 6 Splitting rules for (a) concave quadrilaterals and (b) degenerate quadrilaterals

2.3.2 | Generation algorithm of nodes and elements

When generating a shell element, the coordinates of the nodes that constitute the element must be determined. In general, a shell element in an FE model must obey two rules: (1) The three nodes of a three-node shell element are not collinear, and (2) the shape of a four-node shell element is convex (neither concave nor degenerate quadrilateral). Based on the above requirements for shell elements, the proposed generation algorithm is explained in Algorithm 2 .

The proposed generation algorithm determines each newly generated element through the following concept: (1) Calculate the polygon(s) that needs to generate new element(s) within the range of each deleted element; (2) analyze whether the polygon meets the geometric requirements of shell elements. If the requirements are met, the element is directly generated. Otherwise, the polygon will be divided into multiple smaller polygons that meet the requirements. When the polygon is a concave or a degenerate quadrilateral, it will be directly split as shown in Figure 6. If the polygon is more complex (with five or more than five nodes), it will be split by using the proposed splitting algorithm (Algorithm 3). Then based on the above operations, a list of temporary elements (TEL) will be generated. All the generated temporary elements meet the general requirements of shell elements; hence, the updated FE model can be used for analyses. However, a certain proportion of low-quality elements are generated. When investigating the local performance of the component, these low-quality elements cannot be ignored.

To alleviate the above problem, the operations including (1) element quality check, (2) quadrilateral element splitting, and (3) node merging have been developed in Algorithm 2. The element quality check operation checks the larger corner angle and aspect ratio (the ratio between the longest and shortest edge of an element) of the generated elements. If an element has a larger corner angle or higher aspect ratio than the criteria, the element will be determined as a low-quality element. Otherwise, the element is a high-quality element. Referring to Abaqus 2016 (Dassault Systèmes, 2015), the default larger corner angle for element quality check is 160° for quadrilateral elements and 170° for triangular elements, and the default aspect ratio is 10 for both quadrilateral and triangular elements. In this study, the larger corner angle is set to 160°, and the aspect ratio is 10 for both quadrilateral and triangular elements. Since many quadrilateral elements do not pass the quality check owing to their large corner angles, the quadrilateral element splitting operation is proposed to divide the quadrilateral elements by a line that connects the node of the large corner angle and the opposite node. Then the element quality will be improved by obtaining the two new smaller triangular elements. The low-quality elements that have a higher aspect ratio than the criteria normally have two nodes very close. Merging the two nodes can effectively solve the problem of element quality caused by the above situation. In the pair of the nodes to be merged, the node that also constitutes other original element(s) or high-quality element(s) will absorb the other node. Then triangular elements whose aspect ratio exceeds the criteria will be degenerated into a line and be removed from the EL, and quadrilateral elements with an aspect ratio exceeding the criteria will degenerate into a triangular element with better quality. Based on the above operations, the qualities of the elements in TEL will be optimized.

Subsequently, whether the nodes that constitute the element in the TEL are in the NL, in other words, whether the nodes that constitute the element in the TEL are used in the FE model after deletion will be checked. If the answer is yes, it is passed. Otherwise, a new node should be registered in the NL. The node registration must be performed before the element generation.

Performing the proposed generation algorithm on the area of each deleted element iteratively is very beneficial. Theoretically, the splitting operation can be directly performed in the area of the polygons representing the damaged structure (PDS)∖polygon of the intact FE model (PFEM). However, in actual cases, direct splitting the area of PDS∖PFEM will easily generate elements that do not fulfill the geometric requirements of shell elements. On the contrary, performing the proposed generation algorithm on the area of each deleted element iteratively can avoid the problem.

2.3.3 | Splitting algorithm of elements

The proposed splitting algorithm is a core step in the generation algorithm (Algorithm 2). Because of the complexity

ALGORITHM 2 The generation algorithm of elements and nodes

Require: *PDS*, the polygon represents the damaged structure
Require: *EL*, element list of the FE model after applying the Algorithm1 (deletion)
Require: *NL*, node list of the FE model after applying the Algorithm1 (deletion)
Require: *DEL*, list of the deleted elements
Require: *GPL*, an empty list to save the polygons where need generating new elements
Require: *TEL*, an empty list to save the newly generated elements temporarily
  for element *i* in *DEL*:
    $GP \leftarrow i \cap PDS$, GP is the polygon(s) where needs to create new element(s) within the area of element *i*
    Add *GP* to *GPL*
  end for
  for *j* in *GPL*:
    if *j* fulfills the geometric requirements of the shell element:
      Create a shell element based on *i*'s coordinates and add it to *TEL*
    else if *j* is a concave quadrilateral or a degenerate quadrilateral:
      Split *j* into two triangles as the way illustrated in Fig. 6
      Create two triangular shell elements based on the splitting results and add them to *TEL*
    else if *j* is a polygon that consists of 5 or more than 5 nodes:
      Splitting is performed based on the splitting policy proposed in the later content (Algorithm 3)
      Create shell elements based on the splitting results and add them to *TEL*
  end for
  for element *k* in *TEL*:
    if *k* is a low-quality quadrilateral element with a large corner angle:
      Split *k* into 2 triangular elements by a line which connects the node of the large angle and the opposite node
      Add the obtained 2 triangular elements to *TEL* and delete *k* from *TEL*
  end for
  for element *k* in *TEL*:
    if *l* is a low-quality element and the shortest distance of its two adjacent nodes is closer than a specific value:
      Merge the two nodes into one node, the node which also constitutes other adjacent original element(s) or high-quality element(s) will absorb the other node of the node pair
      Delete the node being merged from *NL*
      Update *k* in *TEL*
  end for
  for element *k* in *TEL*:
    if area(*k*) = 0:
      Delete *k* from *TEL*, this operation is to remove the degenerated elements cause by node merging
  end for
  for element *l* in *TEL*:
    for node *m* in *l*:
      if *m* not in *NL*:
        Add *m* to *NL*
    end for
  end for
  $EL \leftarrow EL + TEL$
Return *EL*, *NL*

of the splitting algorithm, it is introduced in an individual subsection. Algorithm 3 describes the proposed splitting algorithm.

The algorithm is designed for splitting the polygons, which have five or more than five nodes, into quadrilaterals and (or) triangles. The split quadrilaterals and (or) triangles will fulfill the geometric requirements of the shell element. There are two principles of the splitting operation. On one hand, the split line must be fully within the

ALGORITHM 3 The splitting algorithm for the polygons that consist of five or more than five nodes

Require: *GPL*, a list of the polygons where need generating new elements
Require: *TEL*, a list to save the newly generated elements temporarily
for polygon *i* in *GPL* whose number of nodes ≥ 5:
while exists a split line, which links two nodes of *i* and is fully within the area of *i*, can divide *i* into a convex quadrilateral *Q* and another polygon *P*:
Create a quadrilateral shell element based on *Q* and add it to *TEL*
i ←*P*, assign *P* to *i*
end while
if *i* fulfills the geometric requirements of the shell element as stated in Subsection 2.3.2:
Create a shell element based on *i*'s coordinates and add it to *TEL*
else if *i* is a concave quadrilateral or a degenerate quadrilateral:
Split *i* into two triangles, as illustrated in Fig. 6
Create two triangular shell elements based on the splitting results and add them to *TEL*
else:
while the number of nodes of *i* ≥ 4:
Split *i* into a triangle *T* and another polygon *P* with a split line which links two nodes of *i* and is fully within the area of *i*
Create a triangular shell element based on *T* and add it to *TEL*
i ←*P*
end while
Create a triangular shell element based on *i* and add it t
end for
Return *TEL*

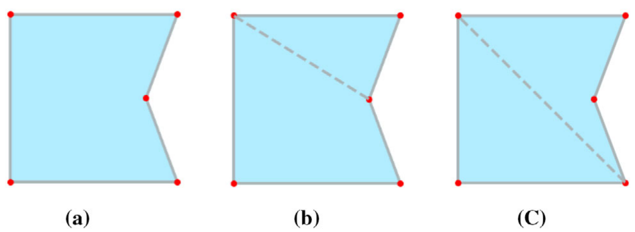


FIGURE 7 Splitting a concave pentagon: (a) Concave pentagon, (b) prioritized splitting pattern, (c) unacceptable splitting pattern

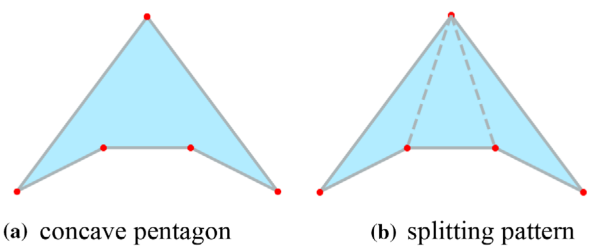


FIGURE 8 Splitting a more complex concave pentagon

polygon and not cross any borderline. On the other hand, the two endpoints of the split line must be two vertices (nodes) of the polygon. No new node is generated during the polygon splitting process.

When splitting a polygon that has five or more than five nodes, priority is given to the splitting pattern that can obtain a convex quadrilateral. This is because using four-node quadrilateral elements can normally achieve higher accuracy than that of the three-node triangular elements if they are all high-quality elements. For example, when splitting a concave pentagon as shown in Figure 7a, the splitting pattern in Figure 7b is prioritized since the splitting results in a convex quadrilateral and a triangle. The splitting pattern in Figure 7c is not acceptable because it returns a concave quadrilateral and a triangle.

If no splitting pattern can return a convex quadrilateral that meets the geometric requirement of shell elements, the triangular splitting will start. For instance, the concave pentagon as shown in Figure 8a can only be split in the way as shown in Figure 8b.

The proposed splitting algorithm is different from the normal meshing methods because new nodes inside the polygon do not need to be generated. If meshing a polygon for generating new elements directly, a connection problem can easily occur between the existing elements and the newly generated elements. As illustrated in Figure 9, the smaller elements obtained from meshing will lose connection with the larger elements. To overcome the above problem, the proposed splitting algorithm does not generate new nodes and only uses the existing nodes for generating split lines.

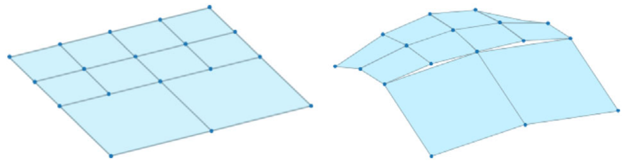


FIGURE 9 Connecting problem of shell elements

3 | CASE STUDIES

To validate the proposed method for automatically updating the geometries of FE models, three experiments in total are performed. The first two experiments are conducted on one of the simplest shell structures—a rectangular paperboard. In the third experiment, the proposed method is examined on an actual bridge.

Segmented images can be obtained by either using standard DL models or using label tools as introduced in Section 2.2. In this study, the segmented images of the three cases are generated by using labeling tools.

All the proposed algorithms are written and validated in Python 3.6 environment. The geometric analyses are performed by using the Shapely package, which is deeply rooted in the conventions of the geographic information system field. If an FE model is regarded as a map, the spatial relationship between nodes and elements can be analyzed by using Shapely.

3.1 | Case 1: Small damage on a board

The intact shell structure in case 1 is shown in Figure 10a. The shell is 297 mm long and 210 mm wide. An FE model of the structure is also established by using 121 nodes and 100 quadrilateral shell elements as shown in Figure 10b. Small damage is made on the short side of the structure as presented in Figure 10c, and the corresponding segmented image is shown in Figure 10d.

Based on the proposed automatic remeshing method, a series of CV algorithms introduced in Section 2.2 are applied to the segmented image to obtain the contours of the damaged structure and the damage. The output of each step is demonstrated sequentially in Figure 11. In the beginning, the green channel of the segmented image is extracted to filter the components of damage in the segmented image as shown in Figure 11a. Assigning a 5×5 all-one matrix as the Gaussian kernel, Gaussian smoothing is applied to remove the Gaussian noise image after the color filtering. By now the image is ready for edge detection, the suitable thresholds for edge detection are 10 and 200 determined by trial-and-error. Through the Canny algorithm, an image of Figure 11b is outputted. Note that the edge is a 1-pixel-width continuous curve, and it may look

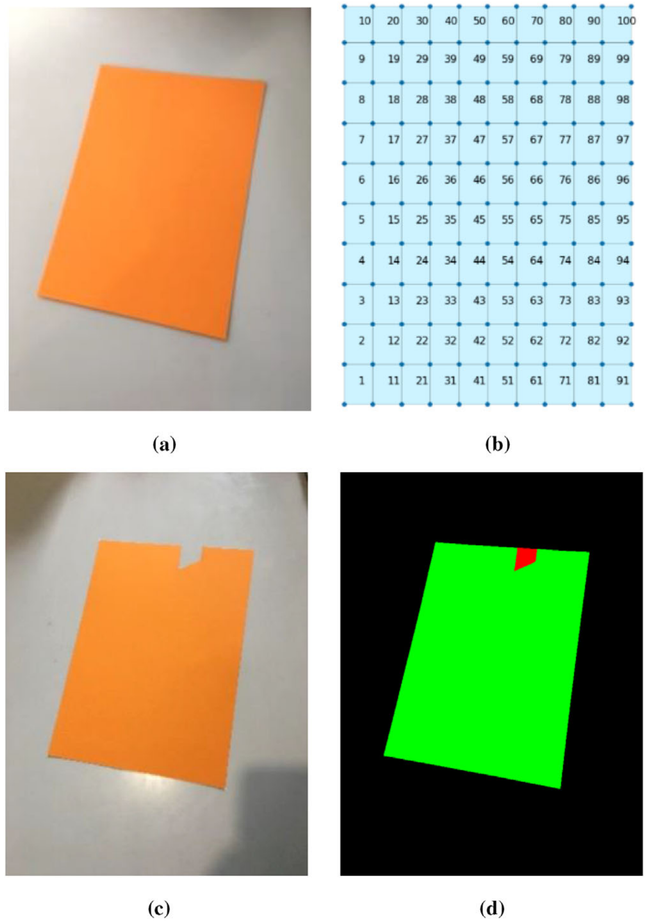


FIGURE 10 Initial data of case 1: (a) Shell structure for tests, (b) FE model of the intact shell structure, (c) damaged shell structure in case 1, and (d) segmented image of the damaged structure in case 1

broken when the image is too small. Afterward, dilation and erosion operations are performed on the detected edge to make the boundary of the structure neater as can be seen in Figures 11c and d. Then the rough contour of the structure can be determined as shown in Figure 11e. However, the rough contour consists of 8114 points. Such a large number of points are redundant for further processing. Therefore, simplification is essential. Assigning ε to 0.03 and simplifying the rough contour by using the Ramer–Douglas–Peucker algorithm as introduced in Section 2.2.1, four points in total are reserved to express the contour of the structure as shown in Figure 11f. The obtained four outer points in the contour are used for alignment when performing the automatic perspective transform. By changing the ε to 0.002 and simplifying the rough contour again by using the Ramer–Douglas–Peucker algorithm, the eight nodes shown in Figure 11g are obtained. Comparing the two simplification results, the eight nodes can represent the contour of the damaged structure more accurately than the four nodes, and the additional four nodes

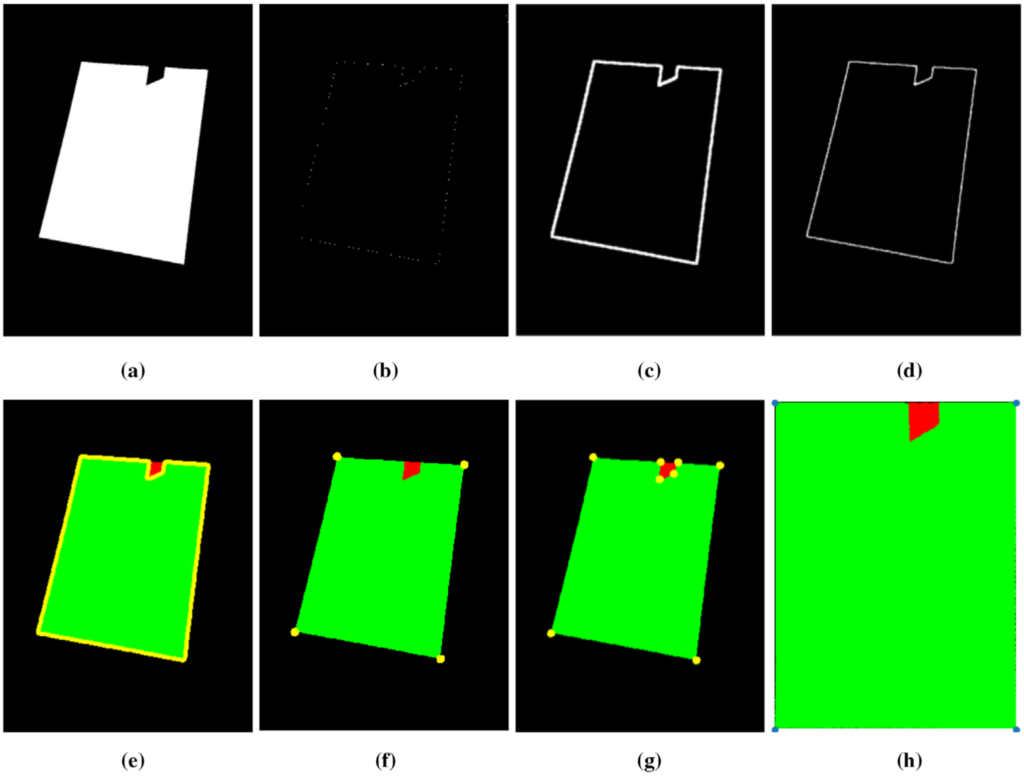


FIGURE 11 Procedure of computer vision (CV) processing in case 1: (a) Color filtering, (b) Canny edge detection result, (c) edge of the structure after dilation, (d) edge of the structure after erosion, (e) contour before simplification, (f) contour after simplification when ε is 0.03, (g) contour after simplification when ε is 0.002; (h) image after perspective transform

can also be used to represent the contour of the damage. As introduced in Section 2.2.1, ε is a parameter to describe how rough a curve will be simplified. It can be determined by either calculation based on its definition to meet the required accuracy of simplification or just several rounds of trial-and-error. Finally, the image data after perspective transformation is presented in Figure 11h, in which the structure is displayed in a perfect top view. The detailed procedure include: (1) Substituting the four points into the x_i' and y_i' in Equation (1); (2) assigning the desired locations after perspective transform into the x_i and y_i in Equation (1); (3) calculating the transform matrix $\boldsymbol{M}$ by Equation (3); and (4) transforming each pixel in the original picture and performing interpolation to fill in the blank pixels in the generated images. The coordinates of each pixel in the perspective-transformed image can be easily converted to the corresponding position on the structure. Similarly, by extracting the red channel of the segmented image in the first step and processing it as the above steps, the contour of the damage can also be obtained. The parameters of the CV algorithms (except perspective transform) are mainly determined by trial-and-error. This is because the pixel values in images vary greatly from scene-to-scene and object-to-object, and in most cases, trial-and-error can be an efficient way to obtain suitable parameters. Up to now, the CV-based image processing of the structure has been completed, and the contours of the structure and the damage have been obtained. It is ready to use the contour information to update the geometry of the FE model.

The FE model updating is started by deleting unnecessary nodes and elements. Based on the obtained contour of the damaged structure as well as the NL and EL of the FE model, the proposed deletion algorithm makes the deletion process fully automatic. First, PDS and PDA can be generated by using the contour information obtained from the CV algorithms as shown in Figure 12. By using the proposed deletion algorithm, four elements (elements 59, 60, 69, and 70) in the FE model that have intersections with PDA can be identified. Thus, the four elements are removed, and the FE model becomes in the state as shown in Figure 13. The nodes that constituted the deleted elements are analyzed to check whether they also contribute to the reserved elements. Then the nodes with no contribution will be removed.

After deleting the nodes and elements involved in the damage, by using the proposed generation and splitting algorithms, new nodes and elements are automatically generated for updating the damage on the FE model. By computing element $i \setminus$ PDS for i in the list of the deleted elements (DEL), the polygons that need new nodes and

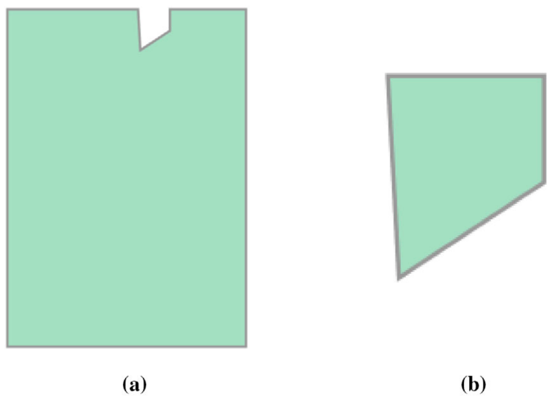


FIGURE 12 Polygons extracted from the image in case 1: (a) Damaged structure, (b) damage area

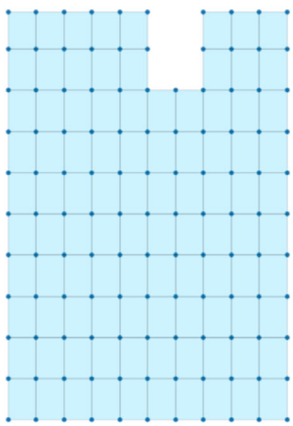


FIGURE 13 FE model after deletion

elements are outputted as shown in Figure 14. The number on each polygon is the identity (ID) of the deleted element. The polygon within element 60 meets the geometric requirement of a shell element, so it can be transformed into a shell element directly. The other polygons do not fulfill the geometric requirements of a shell element, so splitting operation is performed by using the proposed splitting algorithm. Finally, the new nodes and elements to the FE

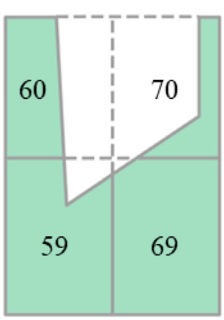


FIGURE 14 Elements have an intersection with damage

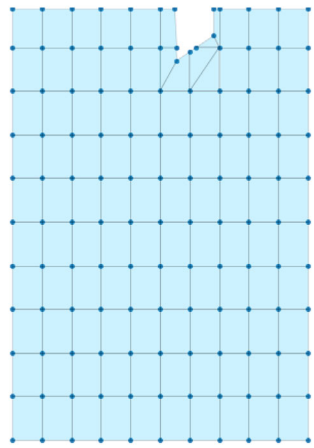


FIGURE 15 Geometry updated FE model (case 1)

model after the deletion are assembled, and the geometry of the FE model has been updated as shown in Figure 15.

3.2 | Case 2: Large damage on a board

Since the CV processing in case 2 is identical to case 1, the procedure of the CV processing in case 2 has been briefly summarized in Figure 16. The photo of the damaged structure is shown in Figure 16a, and the corresponding segmented image is shown in Figure 16b. The damage crosses over half of the structure, and it is more complex than the damage in case 1. Through the same CV algorithms, the contour of the damaged structure can be simplified to 13 points, as shown in Figure 16c, and the segmented image after the perspective transform is shown in Figure 16d. Here, ε has been assigned to 0.0035 and 0.1 to obtain the 13-node and 4-node contours. Then PDS and PDA can be generated as shown in Figure 17. Since the shape of damage obtained by perspective transform may cause an error, the obtained shape of damage is compared to the actual structure to verify the accuracy of contour recognition. The ground truth of the perimeter and area of the damage are 409.93 mm and 3761.5 mm^2, respectively. The perimeter and area of the identified damage are 407.31 mm and 3667.5 mm^2, respectively. Thus, the error rates of the perimeter and area of the obtained damage are 0.64% and 2.50%. The area of the intersection between the ground truth and obtained damage is 3623.53 mm^2, which is 96.33% of the area of ground truth. Therefore, the above investigation shows that the shape of the damage has been captured accurately and is feasible to be further used for updating the geometry of the FE model.

The proposed deletion algorithm can delete nodes and elements in the FE model for the damage accurately. First, whether the elements in the FE model have intersection(s) with PDA are analyzed, and all the intersected

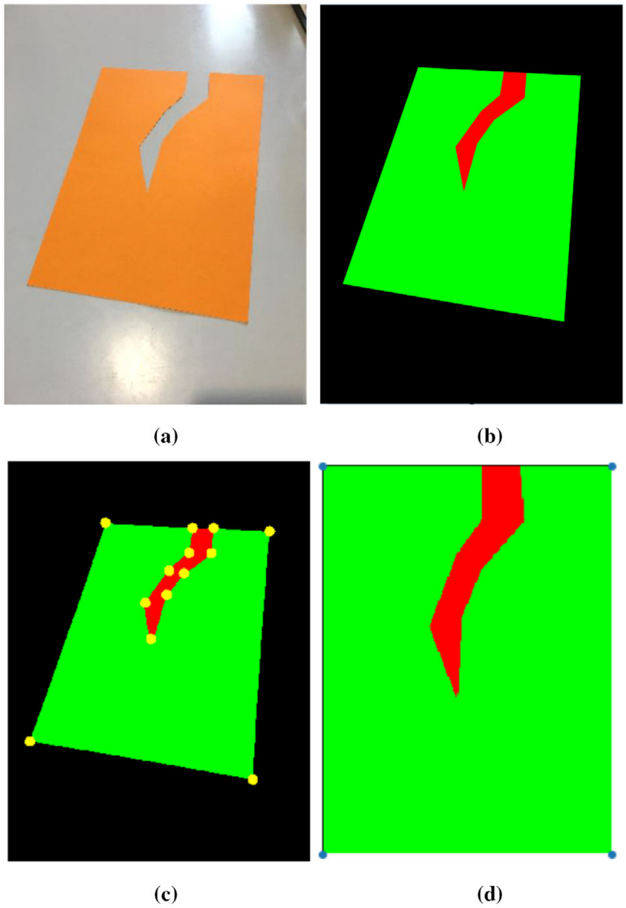


FIGURE 16 Photo, segmented image, and the outputs of the CV algorithms in case 2: (a) photo of the damaged shell structure, (b) segmented image, (c) simplified contour of the damage structure, (d) segmented image after perspective transform

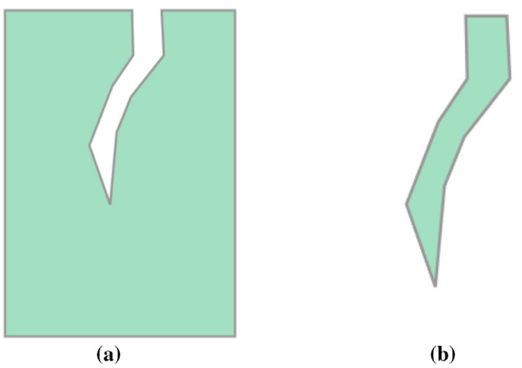


FIGURE 17 Polygons extracted from the image in case 2: (a) damaged structure, (b) damage area

elements are deleted automatically. Second, the nodes in the delated elements are removed if they do not constitute any reserved element. Afterward, the FE model after deletion is obtained as shown in Figure 18.

A comparison between the deleted elements and the damage is shown in Figure 19 for better visualizing their

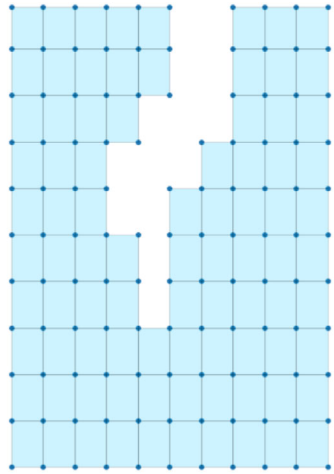


FIGURE 18 FE model after deletion (case 2)

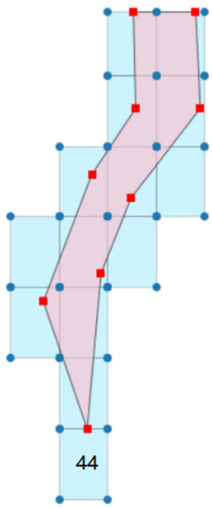


FIGURE 19 Relationship between the delated elements and damage (case 2)

geometric relationship. All the nodes of the FE model within the area of the damage are removed. New nodes should be generated on (1) the vertices of the contour of the damage and (2) intersections between the damage and the boundary of each element. The unmasked blue parts in Figure 19, which can be obtained by computing the union of each $i \setminus$ PDA for i in the DEL, should have new elements generated. There is one special situation that needs additional explanation. The lowest corner of the damage just touches the boundary of an element below it (element 44), and it does not cross the boundary of the element. In this case, the touched element is first deleted by using the deletion algorithm since its intersection with the damage is a point and not empty. In the generation process, the generation algorithm will consider element 44 as a degenerate pentagon, which will be split by the proposed splitting algorithm. The splitting strategy of the above

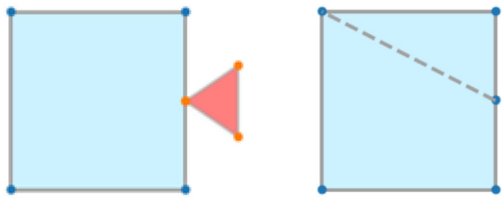


FIGURE 20 Splitting strategy of point touching cases

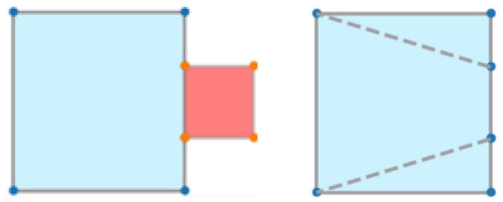


FIGURE 21 Splitting strategy of line-touching cases

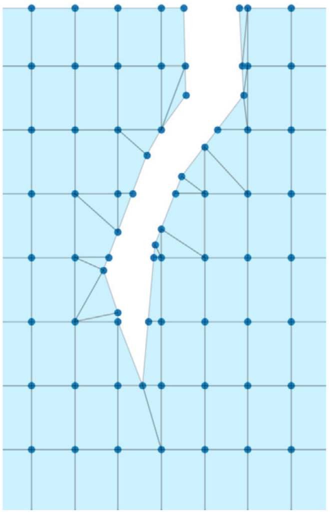


FIGURE 23 Damage area in the updated FE model (Case 2)

one-point-touching situation can be simply illustrated in Figure 20. Similarly, when a line segmentation of damage touches the boundary of an element, the element will be split according to a degenerate hexagon, with the splitting strategy shown in Figure 21.

When the geometry of damage is complex, generating the new elements by only splitting may result in a certain proportion of low-quality elements. To overcome this problem, the proposed generation algorithm (Algorithm 2) also consists of the operations of element quality check, splitting of large corner angle of quadrilateral elements, and node merging. As introduced in Section 2.3.2, referring to Abaqus 2016 (Dassault Systèmes, 2015), the criteria of the element quality check are the larger corner angle of 160° and the aspect ratio of 10 for both quadrilateral and triangular elements in this study. Figure 22 demonstrates the effectiveness of the proposed node-merging operation for improving the element quality of the updated FE model.

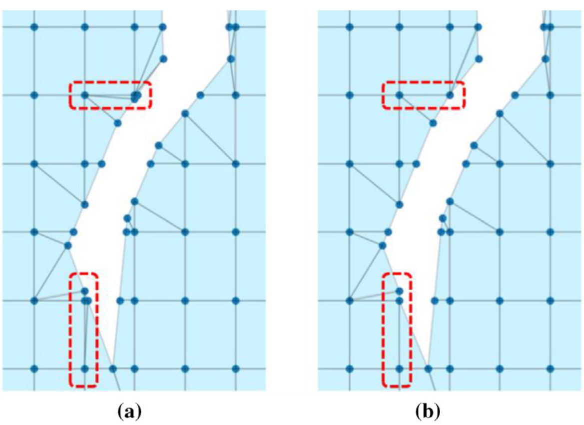


FIGURE 22 Visualization of node-merging operation: (a) Before merging, and (b) after merging

Two pairs of nodes in total are merged into two nodes as shown in the red boxes in Figure 22. In each red box, there is a low-quality element with a very high aspect ratio. By merging the two nodes of the short edge of the element, the element degenerates into a line segment and then is removed from the EL. The node-merging operation also affects the adjacent element of the low-quality element because they share the same edge that has been merged into a node. For each red box in Figure 22, a small triangular element beside the low-quality element is also degenerated into a line segment and is removed from the EL. As a result, the problem of the low-quality elements has been alleviated effectively.

The geometry of the FE model is successfully updated by using the proposed method as can be seen from the final damage updating result (zoomed damaged area in Figure 23). The method can present the geometry of the damage accurately in the updated FE model. Small angular changes of the damage contour can be precisely captured. At the same time, unnecessary nodes are not generated to avoid updating the damage-uninvolved elements. Furthermore, all the new elements generated by the proposed algorithms fulfill the geometric requirements of the shell element, though two low-quality elements are generated. Since the experiment was designed to demonstrate the method, the element sizes are very large, which made it easier to obtain low-quality elements. In actual engineering cases, element sizes will be much smaller, and the possibility of generating low-quality elements by the proposed algorithms will be much lower.

To validate the performance of the proposed FE model updating method, the FE model updated by the proposed algorithms is compared to the manually updated FE model (shown in Figure 24), which has a higher quality of

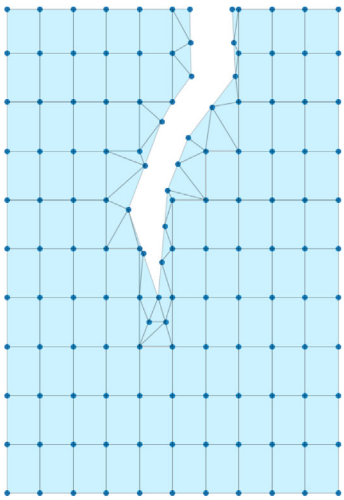


FIGURE 24 Manually updated FE model

TABLE 1 Natural frequencies of both finite element (FE) models (Hz)

Mode	Automatically updated FE model	Manually updated FE model	Difference
1	14.55	14.63	0.55%
2	30.57	31.15	1.86%
3	32.87	33.09	0.66%
4	43.80	43.88	0.18%
5	53.11	53.50	0.73%
6	82.98	83.05	0.08%
7	88.87	89.40	0.59%
8	109.24	109.03	0.19%
9	139.71	142.62	2.04%
10	147.97	148.29	0.22%

elements. The manually updated FE model is obtained by the following three steps: (1) Using the mesh generating function of a commercial finite element analysis (FEA) software to generate the meshing information of the damaged area, (2) manually using the meshing result to generate elements, and (3) integrating these newly generated elements to the FE model after the deletion operation. The boundary condition is to simulate the board being fixed at the four vertices by four bolts. The translational movement in the x, y, and z directions and the rotational movement in the x and y directions are fixed, and the rotational movement in the z-direction (perpendicular to the board surface) is free. Eigenvalue analyses of both FE models are performed because natural modes can reflect the overall structural capacity and are commonly used for evaluating the structural health state. The dimensions of the structure are $297 \times 210 \times 0.5$ mm. Assuming the material of the structure is SS400 steel, the material properties are as follows: Elastic model of 2.05×10^5 N/mm^2, Poisson's ratio of 0.3, the linear expansion coefficient of 0.2×10^{-5} N/mm^3, and density of 7.85×10^{-9} N/mm^3/g. In total 10 natural modes are calculated. The natural frequencies of both FE models are shown in Table 1, and the mode shapes of the automatically updated FE model are shown in Figure 25 as an example. Table 1 shows that the natural frequencies of both FE models are from about 14 to 148 Hz. The natural frequencies of the two FE models are very close. For example, the difference in mode 6 is only 0.08%, and the largest difference is 2.04% at mode 9. The average difference of the 10 modes is 0.71%. The result shows that the FE model updated by the proposed algorithms can represent the dynamic characteristics of the structure accurately, and it also proves that a limited number of low-quality elements will not affect the analytical result of the entire structure.

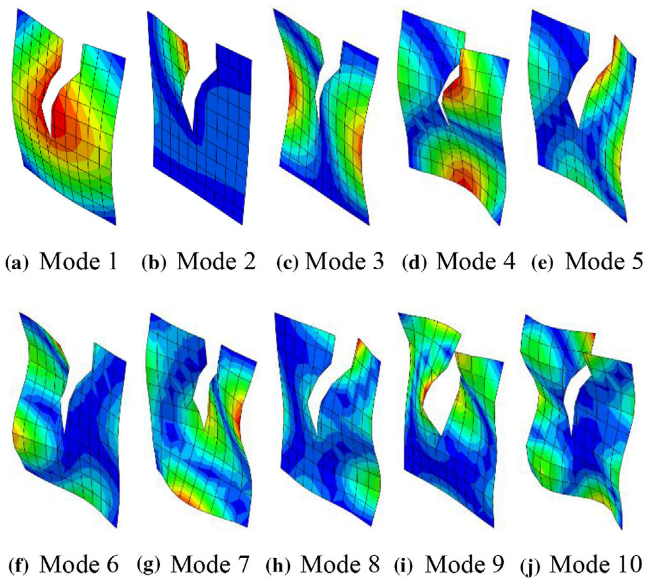


FIGURE 25 Mode shapes of the manually updated FE model

3.3 | Case 3: Damaged steel crossbeam of a bridge

The proposed method is validated on a damaged C-shaped crossbeam of the Funatonishi bridge. The bridge is a five-span simply supported composite bridge in Japan, and a photo of it is shown in Figure 26 (TORNH, 2003). The bridge is 100.0 m long and 10.2 m wide. The general drawing and sectional view of the bridge are shown in Figures 27 and 28, in which the location of the damaged crossbeam is also indicated. The construction of the bridge was completed in 1968, and the fatigue damage on the crossbeam, as shown in Figure 29 (TORNH, 2003), was identified in 2003. The damage crosses half of the upper

FIGURE 26 Photo of the Funatonishi bridge

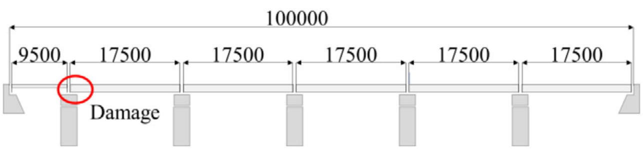


FIGURE 27 General drawing of the Funatonishi bridge

flange, the entire web, and the entire lower flange. Because of lack in the design drawings of the C-shaped beam, the height of the web is assumed as 300 mm, the width of the flanges as 100 mm, and the thickness as 10 mm. These dimensions help to determine the scales in the FE model. The corresponding FE model in the intact state is shown in Figure 30.

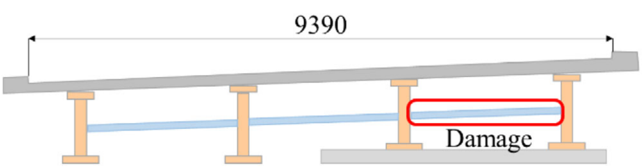


FIGURE 28 Sectional view of the Funatonishi bridge

FIGURE 29 Damage on the crossbeam

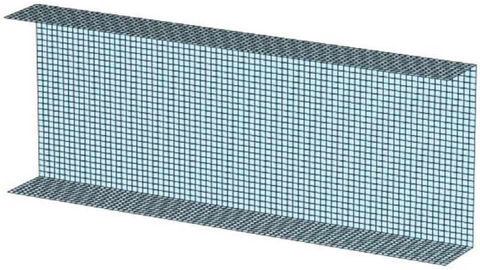


FIGURE 30 FE model of the crossbeam in the intact state

The procedure of updating the geometry of the FE model of the crossbeam is summarized in Figure 31. First, since both the C-shaped steel beam and the fatigue damage are 3-D, 2-D decomposition of the beam is necessary to use the proposed geometry updating method. The 2-D decomposition of the segmented image can be simply accomplished by using color filters, and three individual segmented images representing the web and flanges will be generated. As a result, the 3-D C-shaped crossbeam is decomposed into three 2-D quadrilateral components. Subsequently, by applying the CV algorithms introduced in Section 2.2 to the segmented image of each component, the four points of the web and flanges for perspective transform can be obtained, and then the perspective-transformed images are computed. The two common points of the four points for perspective transform between the web and each flange will be used for the assemblage after updating the geometries of all the three components.

By using the proposed deletion, generation, and splitting algorithms, the geometries of the decomposed FE-models of web and flanges are updated, respectively. The detailed information of (1) the shape of damage in each component, (2) the polygons that require splitting, and (3) the updated FE model in each component are shown in Figure 32. Finally, the three individual 2-D FE models are assembled into a 3-D FE model. There is one potential problem, that is, due to the limitation of image quality and error of the CV algorithms, the generated nodes on the boundaries of the damage from two adjacent decomposed FE models may not be perfectly matched at the junction. For instance, at the upper junction of the crossbeam, the x coordinate of the damage on the right side is 372 on the upper flange but 374 on the web. If the coordinates of the damage on the junction on each decomposed FE model are not identical, the decomposed FE models cannot be assembled into the final 3-D FE model. This error is very small, and the solution can be using the average of the coordinates of both sides at the junction. Thus, in this case, the x coordinate of the node on the right boundary of the damage on the upper junction uses 373 on both the upper flange and the web. As a result, the geometry updated FE model is obtained as

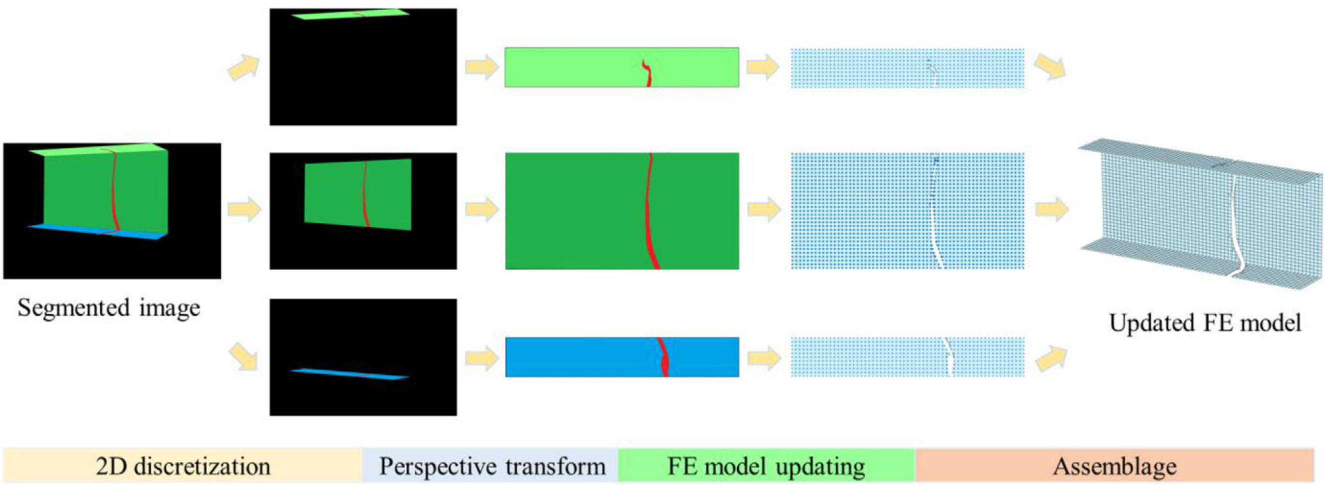


FIGURE 31 Procedure of geometry updating of the FE model in case 3

shown in Figure 33. In total 138 shell elements are generated by the proposed algorithms during the procedure of model updating, and all the generated elements can pass the element quality check. The results show the effectiveness of the optimization operations in Algorithm 2. Overall, the crack is accurately updated on the FE model, and it enables the fast quantification of the degradation on the loading capacity of the bridge caused by the fatigue damage through numerical analyses.

4 | CONCLUSION

In this study, a novel method for automatically updating the geometric changes of the FE models simulated with shell elements is proposed via CV techniques and geometric analyses. The contributions include the developments of the algorithms for deleting, generating, and splitting shell elements, as well as the integration of the proposed algorithms with CV techniques. The proposed method can greatly simplify the process of updating the geometric changes of FE models, allowing the FE models to be used for rapid analyses and quantification of the degradation in loading capacity caused by structural damages.

Through the three experiments, four conclusions can be drawn as follows: First, the proposed method can update the geometric changes of structures successfully by using a single segmented image data captured after damage occurs and an FE model in the original state. The method does not need to rebuild the entire FE model. Only the nodes and elements involved in the damaged area(s) will be processed by the proposed method.

Second, the proposed algorithms for automatic deletion, generation, and splitting of shell elements obtained promising validation results in the experiments. In the real-world application in case 3, all the generated elements can pass the element quality check, demonstrating the high performance of the proposed method for automatically updating the geometric changes caused by damages.

Third, the automatically updated FE model, by using the proposed method, is comparable to the manually updated FE model for eigenvalue analysis. The differences between eigenvalue analysis results are remarkably small, and it proves that the FE models updated by the proposed method can present the dynamic characteristics of structures accurately.

Finally, the proposed method can not only be used on pure 2-D structures like shell structures but can also be applied on all the structures that can be simulated with shell elements and can be decomposed into several 2-D FE models, for example, C-, T-, I-shaped and hollow beams, and so forth. In actual engineering cases, many scenarios can use the proposed method with their SHM system for fast assessment of the changes of structural capacity.

There are also several limitations of the proposed FE model in geometry updating method. First, the proposed method cannot update the changes of material and mechanical properties caused by structural damages. In the current stage, it focuses on updating the geometric changes caused by damages on the corresponding FE models. Evaluating the decrease of material and mechanical properties of structures affected by damages will be investigated in the future study. Second, the method requests a segmented image of the structure, indicating the regions of structural components, damage, and background. Currently, the existing DL models for image segmentation are only for damage detection or only for structural components segmentation. One integrated DL model is required and will be developed in future works to simplify the computation. Finally, the proposed method

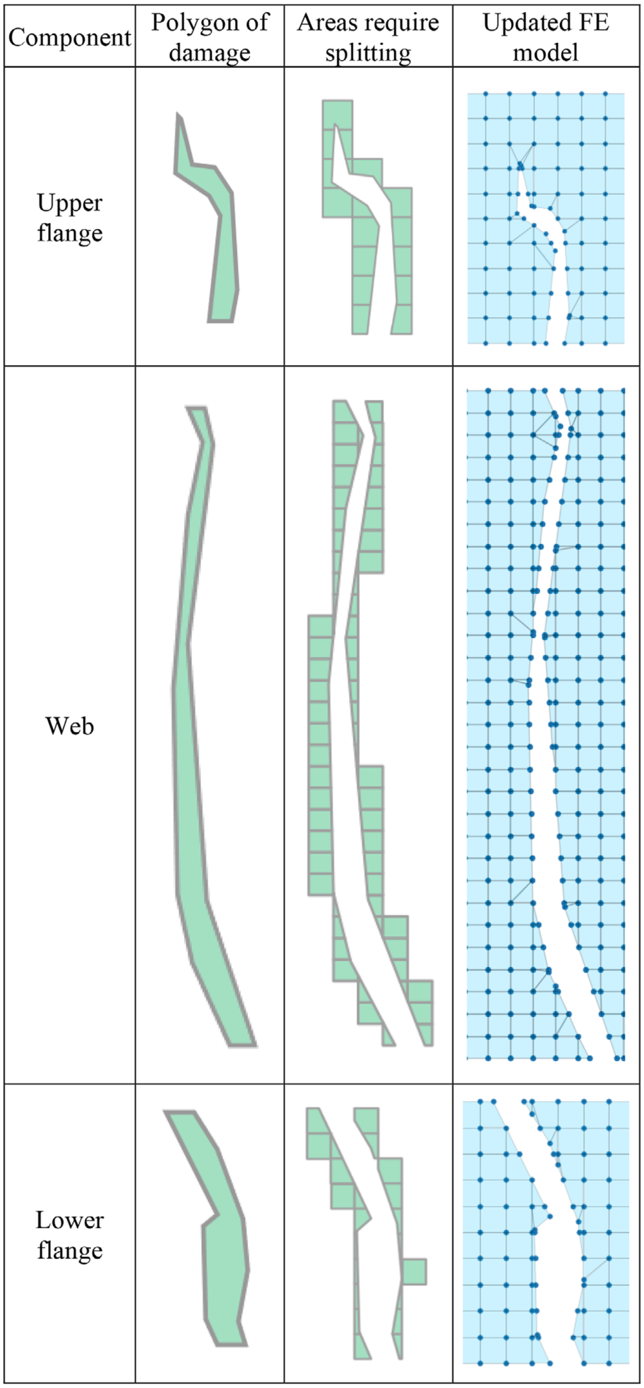


FIGURE 32 Details of the updated FE model

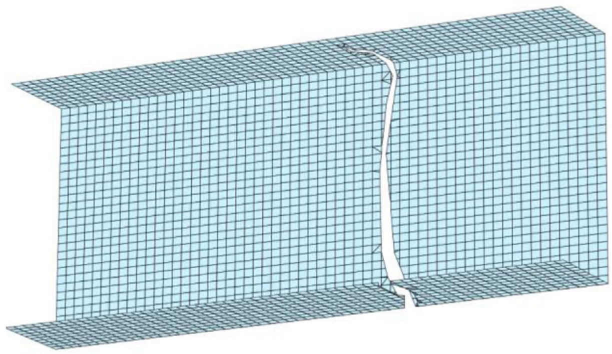


FIGURE 33 Bottom view of the updated FE model of the crossbeam

cannot be used to update the geometric changes of composite materials if the damage is depth relevant, such as the cases of microcracks and delamination on carbon fiber composite materials and concrete spalling. To overcome this problem, the proposed algorithms such as element deletion, generation, and splitting will be upgraded from 2-D to 3-D, using a similar approach in this article.

As mentioned before, this study is a step towards the final goal of using FE models as the digital twin to present the load capacity of the monitored structure in real time. Although only updating the visible damage of structures on FE models and performing numerical analyses cannot precisely answer the two aforementioned questions, the proposed method can automate the estimation of the rough upper bound of the loading capacity by using FE analyses. The proposed method requests low computational cost and it shows a very high potential to be utilized in the next generation of the SHM systems for real-time structural state assessment. Meanwhile, combining the proposed method and the referred damage prognostics methods, the basis of the computational engine of a structural digital twin for real-time damage diagnostics and prognostics might be developed.

ACKNOWLEDGMENT

The authors greatly appreciate the photos in case 3 provided by the Tokushima Office of River and National Highway, Shikoku Regional Development Bureau, Ministry of Land, Infrastructure, Transport and Tourism, Japan.

REFERENCES

Adeli, H., & Yu, G. (1995). An integrated computing environment for solution of complex engineering problems using the object-oriented programming paradigm and a blackboard architecture. *Computers & Structures*, *54*(2), 255–265.

Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., & Szeliski, R. (2011). Building Rome in a day. *Communications of the ACM*, *54*(10), 105–112.

American Society of Civil Engineers (ASCE). (2017). *Infrastructure report card: Bridges C+, 9.1% of bridges rated structurally deficient*. https://www.infrastructure-reportcard.org/wp-content/uploads/2017/01/Bridges-Final.pdf

Amezquita-Sanchez, J. P., & Adeli, H. (2015). Synchrosqueezed wavelet transform-fractality model for locating, detecting, and quantifying damage in smart highrise building structures. *Smart Materials and Structures*, *24*(6), 065034.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. *PAMI-8*(6), 679–698.

Castellazzi, G., D'altri, A. M., Bitelli, G., Selvaggi, I., & Lambertini, A. (2015). From laser scanning to finite element analysis of complex buildings by using a semi-automatic procedure. *Sensors*, *15*(8), 18360–18380.

Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *32*(5), 361–378.

Cha, Y. J., Choi, W., Suh, G., Mahmoudkhani, S., & Büyüköztürk, O. (2018). Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, *33*(9), 731–747.

Chen, J. G., Adams, T. M., Sun, H., Bell, E. S., & Büyüköztürk, O. (2018). Camera-based vibration measurement of the World War I memorial bridge in Portsmouth, New Hampshire. *Journal of Structural Engineering*, *144*(11), 04018207.

Chiachío, J., Chiachío, M., Prescott, D., & Andrews, J. (2019). A knowledge-based prognostics framework for railway track geometry degradation. *Reliability Engineering & System Safety*, *181*, 127–141.

Chiachío, J., Chiachío, M., Sankararaman, S., Saxena, A., & Goebel, K. (2015). Condition-based prediction of time-dependent reliability in composites. *Reliability Engineering & System Safety*, *142*, 134–147.

Clement, R., Schneider, J., Brambs, H. J., Wunderlich, A., Geiger, M., & Sander, F. G. (2004). Quasi-automatic 3D finite element model generation for individual single-rooted teeth and periodontal ligament. *Computer Methods and Programs in Biomedicine*, *73*(2), 135–144.

Cristiani, D., Sbarufatti, C., & Giglio, M. (2021). Damage diagnosis and prognosis in composite double cantilever beam coupons by particle filtering and surrogate modelling. *Structural Health Monitoring*, *20*(3), 1030–1050.

Dassault Systèmes. (2015). ABAQUS 2016, ABAQUS/CAE user's guide (pp. 17–27). http://130.149.89.49:2080/v2016/pdf_books/CAE.pdf

Dong, C. Z., Celik, O., & Catbas, F. N. (2019b). Marker-free monitoring of the grandstand structures and modal identification using computer vision methods. *Structural Health Monitoring*, *18*(5–6), 1491–1509.

Dong, C. Z., Celik, O., Catbas, F. N., OBrien, E., & Taylor, S. (2019a). A robust vision-based method for displacement measurement under adverse environmental factors using spatio-temporal context learning and Taylor approximation. *Sensors*, *19*(14), 3197.

Feng, D., & Feng, M. Q. (2017). Identification of structural stiffness and excitation forces in time domain using noncontact vision-based displacement measurement. *Journal of Sound and Vibration*, *406*, 15–28.

Fernandez, I., Bairán, J. M., & Marí, A. R. (2016). 3D FEM model development from 3D optical measurement technique applied to corroded steel bars. *Construction and Building Materials*, *124*, 519–532.

Finnish Transport Infrastructure Agency (VÄYLÄ). (2018). *Maintenance of highways*. https://vayla.fi/web/en/road-network/maintenance

Ghahremani, K., Khaloo, A., Mohamadi, S., & Lattanzi, D. (2018). Damage detection and finite-element model updating of structural components through point cloud analysis. *Journal of Aerospace Engineering*, *31*(5), 04018068.

Gopalakrishnan, K., Gholami, H., Vidyadharan, A., Choudhary, A., & Agrawal, A. (2018). Crack damage detection in unmanned aerial vehicle images of civil infrastructure using pre-trained deep learning model. *International Journal for Traffic and Transport Engineering*, *8*(1), 1–14.

Huang, H. W., Li, Q. T., & Zhang, D. M. (2018). Deep learning based image recognition for crack and leakage defects of metro shield tunnel. *Tunnelling and Underground Space Technology*, *77*, 166–176.

Jahanshahi, M. R., Chen, F. C., Joffe, C., & Masri, S. F. (2017). Vision-based quantitative assessment of microcracks on reactor internal components of nuclear power plants. *Structure and Infrastructure Engineering*, *13*(8), 1013–1026.

Jiang, X., & Adeli, H. (2007). Pseudospectra, MUSIC, and dynamic wavelet neural network for damage detection of highrise buildings. *International Journal for Numerical Methods in Engineering*, *71*(5), 606–629.

Khuc, T., & Catbas, F. N. (2017a). Computer vision-based displacement and vibration monitoring without using physical target on structures. *Structure and Infrastructure Engineering*, *13*(4), 505–516.

Khuc, T., & Catbas, F. N. (2017b). Completely contactless structural health monitoring of real-life structures using cameras and computer vision. *Structural Control and Health Monitoring*, *24*(1), e1852.

Korumaz, M., Betti, M., Conti, A., Tucci, G., Bartoli, G., Bonora, V., Korumaz, G. A., & Fiorini, L. (2017). An integrated terrestrial laser scanner (TLS), deviation analysis (DA) and finite element (FE) approach for health assessment of historical structures. A minaret case study. *Engineering Structures*, *153*, 224–238.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lee, J. H., Yoon, S. S., Kim, I. H., & Jung, H. J. (2018 March). Diagnosis of crack damage on structures based on image processing techniques and R-CNN using unmanned aerial vehicle (UAV). In H. Sohn, J. P. Lynch, & K. W. Wang (Eds.). *Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2018*, 10598(1059811-1–1059811-8). International Society for Optics and Photonics.

Li, S., Zhao, X., & Zhou, G. (2019). Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering*, *34*(7), 616–634.

Li, X., Chen, H., Qi, X., Dou, Q., Fu, C. W., & Heng, P. A. (2018). H-DenseUNet: Hybrid densely connected UNet for liver and tumor segmentation from CT volumes. *IEEE Transactions on Medical Imaging*, *37*(12), 2663–2674.

Li, X., & Kurata, M. (2019). Probabilistic updating of fishbone model for assessing seismic damage to beam–column connections in steel moment-resisting frames. *Computer-Aided Civil and Infrastructure Engineering*, *34*(9), 790–805.

Lin, W., Lam, H., Yoda, T., Ge, H., Xu, Y., Kasano, H., Nogami, K., & Murakoshi, J. (2017). After-fracture redundancy analysis of an aged truss bridge in Japan. *Structure and Infrastructure Engineering*, *13*(1), 107–117.

Liu, Y. F., Nie, X., Fan, J. S., & Liu, X. G. (2020). Image-based crack assessment of bridge piers using unmanned aerial vehicles and three-dimensional scene reconstruction. *Computer-Aided Civil and Infrastructure Engineering*, *35*(5), 511–529.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3431–3440), Boston, MA.

Lydon, D., Lydon, M., Taylor, S., Del Rincon, J. M., Hester, D., & Brownjohn, J. (2019). Development and field testing of a vision-based displacement system using a low cost wireless action camera. *Mechanical Systems and Signal Processing*, *121*, 343–358.

Ministry of Land, Infrastructure, Transport and Tourism (MLIT), Japan. (2018). White paper on land infrastructure transport and tourism in Japan, 2019. https://www.mlit.go.jp/common/001325161.pdf

Tokushima Office of River and National Highway (TORNH), Shikoku Regional Development Bureau, MLIT, Japan. (2003). Damage of steel girder bridge (damage of crossbeam). *Cases of damages of steel bridges (superstructure) (in Japanese)* (p. 10). https://www.mlit.go.jp/road/sisaku/yobohozen/yobo3_1_1.pdf

Narazaki, Y., Hoskere, V., Hoang, T. A., Fujino, Y., Sakurai, A., & Spencer, Jr. B. F. (2020). Vision-based automated bridge component recognition with high-level scene consistency. *Computer-Aided Civil and Infrastructure Engineering*, *35*(5), 465–482.

Pan, X., & Yang, T. Y. (2020). Postdisaster image-based damage detection and repair cost estimation of reinforced concrete buildings using dual convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *35*(5), 495–510.

Rafiei, M. H., & Adeli, H. (2017). A novel machine learning-based algorithm to detect damage in high-rise building structures. *The Structural Design of Tall and Special Buildings*, *26*(18), e1400.

Rafiei, M. H., & Adeli, H. (2018). A novel unsupervised deep learning model for global and local health condition assessment of structures. *Engineering Structures*, *156*, 598–607.

Ramer, U. (1972). An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, *1*(3), 244–256.

Salaan, C. J. O., Okada, Y., Mizutani, S., Ishii, T., Koura, K., Ohno, K., & Tadokoro, S. (2018). Close visual bridge inspection using a UAV with a passive rotating spherical shell. *Journal of Field Robotics*, *35*(6), 850–867.

Sajedi, S. O., & Liang, X. (2020). Vibration-based semantic damage segmentation for large-scale structural health monitoring. *Computer-Aided Civil and Infrastructure Engineering*, *35*(6), 579–596.

Schönberger, J. L., & Frahm, J. M. (2016). Structure-from-motion revisited. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4104–4113), Honolulu, Hawaii.

Smith, A. R. (1978). Color gamut transform pairs. *ACM Siggraph Computer Graphics*, *12*(3), 12–19.

Suzuki, S., & Abe, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, *30*(1), 32–46.

Tajima, K., Chen, K. K., Takahashi, N., Noda, N., Nagamatsu, Y., & Kakigawa, H. (2009). Three-dimensional finite element modeling from CT images of tooth and its validation. *Dental Materials Journal*, *28*(2), 219–226.

Xu, Y., Bao, Y., Chen, J., Zuo, W., & Li, H. (2019). Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images. *Structural Health Monitoring*, *18*(3), 653–674.

Yu, G., & Adeli, H. (1993). Object-oriented finite element analysis using EER model. *Journal of Structural Engineering*, *119*(9), 2763–2781.

Zhang, A., Wang, K. C., Fei, Y., Liu, Y., Chen, C., Yang, G., Yang, E., Li, J. Q., & Qiu, S. (2019). Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network. *Computer-Aided Civil and Infrastructure Engineering*, *34*(3), 213–229.

Zhang, A., Wang, K. C., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q., & Chen, C. (2017). Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. *Computer-Aided Civil and Infrastructure Engineering*, *32*(10), 805–819.

Zhang, C., Chang, C. C., & Jamshidi, M. (2020). Concrete bridge surface damage detection using a single-stage detector. *Computer-Aided Civil and Infrastructure Engineering*, *35*(4), 389–409.

How to cite this article: Zhang, Y., & Lin, W. Computer-vision-based differential remeshing for updating the geometry of finite element model. *Comput Aided Civ Inf*. 2022;*37*:185–203. https://doi.org/10.1111/mice.12708