Bauhaus University Weimar
Faculty of Civil Engineering
Faculty of Media
Chair of Intelligent Technical Design
Study Program Digital Engineering

Bauhaus-Universität Weimar

Master Thesis on

Automated Approach for Building Information Modelling of Crack Damages via Image Segmentation and Image-based 3D Reconstruction

Submitted by

Mohamed Said Helmy Alabassy
Born on 23.09.1992 in Rosetta, Beheira Governorate, Egypt
Immatriculation Number 119530
mohamed.said.helmy.alabassy@uni-weimar.de

Examiners

Chair of Intelligent Technical Design
Prof. Dr.-Ing. Christian Koch
c.koch@uni-weimar.de

MSc. Mathias Artus
mathias.artus@uni-weimar.de

Weimar, December 18, 2020

Acknowledgement

Many thanks to Mr. Mathias Artus from the Chair of Intelligent Technical Design for guidance and critical comments, Ms. Mariya Kaisheva from the Chair of Computer Vision for helpful references recommendations. Last but not least, my deepest gratitude goes to my family for their ceaseless support and encouragement.

Statutory Declaration

I hereby affirm that the master's thesis at hand is my own written work and that I have used no other sources and aids other than those indicated. All passages, which are quoted from publications or paraphrased from these sources, are indicated as such, i.e. mentioned, cited and attributed. This thesis was not submitted in the same or in a substantially similar version, not even partially, to another examination board and was not published elsewhere.

Place, Date

Mohamed Said Helmy Alabassy

Weimar, December 18, 2020

Abstract

As machine vision-based inspection methods in the field of Structural Health Monitoring (SHM) continue to advance, the need for integrating resulting inspection and maintenance data into a centralised building information model for structures notably grows. Consequently, the modelling of found damages based on those images in a streamlined automated manner becomes increasingly important, not just for saving time and money spent on updating the model to include the latest information gathered through each inspection, but also to easily visualise them, provide all stakeholders involved with a comprehensive digital representation containing all the necessary information to fully understand the structure's current condition, keep track of any progressing deterioration, estimate the reduced load bearing capacity of the damaged element in the model or simulate the propagation of cracks to make well-informed decisions interactively and facilitate maintenance actions that optimally extend the service life of the structure. Though significant progress has been recently made in information modelling of damages, the current devised methods for the geometrical modelling approach are cumbersome and time consuming to implement in a full-scale model. For crack damages, an approach for a feasible automated image-based modelling is proposed utilising neural networks, classical computer vision and computational geometry techniques with the aim of creating valid shapes to be introduced into the information model, including related semantic properties and attributes from inspection data (e.g., width, depth, length, date, etc.). The creation of such models opens the door for further possible uses ranging from more accurate structural analysis possibilities to simulation of damage propagation in model elements, estimating deterioration rates and allows for better documentation, data sharing, and realistic visualisation of damages in a 3D model.

Contents

List of Figures

List of Tables

Acronyms

ANN Artificial Neural Network. 29

API Application Programming Interface. 66

BIM Building Information Modeling. 19

BnB Branch-and-Bound. 28

CAD Computer Aided Design. 66

CNN Convolutional Neural Network. 21

CSG Constructive Solid Geometry. 20

DIC Digital Image Correlation. 29

DoG Difference-of-Gaussians. 34

DTM Distance Transform Method. 31

EDT Euclidean Distance Transform. 56

FDCT Fast Discrete Curvelet Transform. 29

FFT Fast Fourier Transform. 30

FHT Fast Haar Transform. 30

FLANN Fast Approximate Nearest Neighbour. 34

FoV Field of View. 82

GCPs Ground Control Points. 39

GLCM Grey Level Co-occurrence Matrix. 29

GNSS Global Navigation Satellite System. 66

GPR Ground Penetrating Radar. 33

GPS Global Positioning System. 39

GUI Graphical User Interface. 35

HMS Hit-and-Miss. 48

ICP Iterative Closest Point. 27

IFC Industry Foundation Classes. 19

IoU Intersection Over Union. 27

LoG Laplacian of Gaussian. 30

MAPE Mean Absolute Percent Error. 34

NDT Non Destructive Testing. 33

PL-SGDLR Piecewise Linear Stochastic Gradient Descent Logistic Regression. 32

RANSAC Random Sample Consensus. 34

RBM Restricted Boltzmann Machine. 30

RDP Ramer–Douglas–Peucker. 55

RPN Region Proposal Network. 31

RTK Real-Time Keinematics. 66

SFM Structure from Motion. 34

SHM Structural Health Monitoring. 7

SICN Signed Inverse Condition Number. 76

SIFT Scale-Invariant Feature Transform. 34

SIGE Signed Inverse Gradient Error. 76

SURF Speeded-Up Robust Features. 34

TuFF Tubularity Flow Field. 31

UAVs Unmanned Aerial Vehichles. 20

VO Visual Odometry. 34

VR Virtual Reality. 21

VSLAM Visual Simultaneous Localisation and Mapping. 34

XFEM Extended Finite Element Method. 82

1 Introduction

This chapter sets forth some introductory insight into the topic of the thesis, stating the problem that the conducted research is trying to solve, whilst reasoning the motivation behind carrying it out and formulates the objectives to be reached and its potential contribution for future work.

1.1 Problem Statement

Looking at the latest progress of information modelling of damages, there exist two approaches for modelling. The first of which is largely based on damage images textured on top of the model, as what has been successfully carried out in Industry Foundation Classes (IFC) data model [39]. While the authors undoubtedly demonstrated the capability of IFC models to handle images, most IFC viewers currently available are not yet capable of viewing them properly. The textures still serve an important purpose of documenting semantic attributes collected through the inspection process and visualising the damage textures within a centralized model, yet cannot provide detailed information about the actual damage geometry. The latter approach relies on modelling damage shapes' geometries manually and inserting them into the IFC model, as demonstrated by [7].

This second approach is most suited for damage types such as cracks and spalls, where the shapes could be subtracted from the original building element containing them in the model. Some early methods already exist for automated geometric construction of crack shapes in literature. However, they are neither applied within a Building Information Modeling (BIM) context [21, 65] nor reliable to produce valid shapes at high curvatures. For spalling shapes, the aforementioned proposal [42] requires almost entirely a high level of human interference through the whole modelling process and the alignment of the shapes to the model is vaguely described.

The prototype implementation based on images in the exemplary use cases of spalling and cracks published by [7, 8] proved the feasibility of such approach, albeit the way of implementation is extremely time consuming, relied entirely on human experience and interaction in every single step throughout the shapes reconstruction, and was not always guaranteed to produce valid shapes for the damages in the end, which rendered it inefficiently applicable in a full-scale central BIM project. It showed however mixed results when viewed in IFC viewers. That discrepancy could either be attributed to problems from geometric kernels of IFC viewers in modelling such free form voided shapes of damages or the validity of the shapes per se to be modelled appropriately as both of which are still open to further investigation.

A similar, yet more advanced, approach was proposed by Isailović et al. [42] for modelling spallings based on point clouds generated from LASER scanning to 3D reconstruct the constitutive part of the damaged building element in the information model. However, that proposal still falls short of the automation level proposed in this thesis, as it required a high level of human expertise intervening throughout the whole process for closing openings in the meshed point cloud, alignment of the meshed point cloud from the LASER scanning to the as is mesh, and solidifying the relevant damaged regions of interest from the scanned point cloud necessary to perform a Constructive Solid Geometry (CSG) boolean difference operation. The process of conversion from a meshed surface to a closed surface by scaling cannot possibly result in a solid shape, without an intermediary step, like thickening or extrusion, to make it possible; something that was never mentioned in said article. Moreover, the scaling up of shapes by 2-5% approach as described, is flawed, as it changes the dimensions of the constructed geometries, rendering them, apart from the visual enrichment of the model, useless for other more practical purposes (e.g., to calculate volumes and surface areas or compare them to previous states), since the scaled up shapes no longer match the real world dimensions of the structure.

1.2 Motivations

While cost-time analyses between traditional and Unmanned Aerial Vehichles (UAVs) assisted inspection or other methods vary in results depending on the individual structure inspected, its size, location, the level of operations' interruption, and required permits for drones operation, the general trend from several case studies [15, 96] suggests roughly cost savings of 40-60% and 40% on average respectively. For smaller structures that don't require traffic control or access equipment, the cost of non-traditional inspection methods may be slightly higher but with improved inspection deliverables. Though time-savings on site in the data acquisition phase of the presented inspections were achieved, the post-processing of acquired images and data still required a significant amount of time, lowering the overall total time-savings that could be achieved throughout the whole inspection process. With the increasing loads of imagery data acquired, there is a need for further development in automated methods for processing, analysis and visualisation of inspection data that could help reduce both time and cost. This study tries to find a balanced approach to reduce human intervention in time consuming modelling processes prone to human errors, saving time and costs of repetitive remodelling and manually documenting and updating ever changing inspection information to be included in a centralised as-built model on a regular basis by automating the redundant repetitive tasks of modelling damage shapes, yet it still maintains a Human-in-the-loop engagement necessary for operating the workflow optimally, setting and modifying parameters, and monitoring the results of each step to ensure reliable end-results.

1.2.1 Potential Benefits and Uses for Damages' Geometries

Among the most important reasons driving the interest towards geometric modelling of damages are:

- Acquiring the geometric representation of such damages should enrich the centralised information models for buildings with detailed semantics and related inspection information
- It could provide a reliable documentation format digitally accessible at any time, if needed, from the 3D models in proprietary formats, as well as in the IFC neutral data format as published in [12].
- Collecting the geometric data of damages and updating them regularly after each inspection in a centralised BIM could be used to compare the current state of structures to that of previously recorded inspections easily, that would be useful to probabilistically estimate deterioration rates and predict the future state of structures.
- It offers better enhanced visualisation possibilities of damaged building elements in 3D models and in a Virtual Reality (VR) environment.
- With the availability of realistic damage shapes modelled based on the building's actual condition and their location within the information model, they could be used to digitally simulate the currents state of structure to assess the structural safety like propagation analysis for cracks, or estimate reductions in load bearing capacities.
- It has the potential to facilitate further automation steps in condition assessment of the relevant damaged elements and the whole structure.

1.3 Research objectives and target

The 3D image-based reconstruction of valid shapes of damages (e.g., without non-manifold edges, self-intersecting, non-manifold and unreferenced vertices, etc.) is a primary goal for the topic of this thesis. A successful implementation should allow better interactive visualisation with the 3D model, easier access to the semantic information of modelled damages for retrieval, comparison, analysis and manipulation if need be, and open the door for further advancement in automating the process of determining the structural safety by providing full information about their geometric shapes and their location in the 3D model for simulating propagation and monitoring deterioration from previous state and better assess the condition of damaged structures.

1.4 Methodology

In order to facilitate the geometrical modelling of damages with minimal human involvement as possible, both well-established methods can be utilised along with the latest available models of Convolutional Neural Network (CNN) to construct damage shapes with a main focus on compatibility with the open and neutral data format IFC for BIM. Pixel-based crack detection and segmentation from images has been reliably predicted in the last few years using neural networks [23, 54, 56, 59, 113], like the available trained DeepCrack Models [59, 113] or the model architecture used for CrackNausNet [10]. Vectorising the resulting feature maps can be achieved by utilising classical image analysis

algorithms to get the centre-poly-line of the crack and photgrammetric computer vision algorithms to retrieve the relevant depth, location and orientation information. The shapes can be created given the known depth and profile shape to construct its geometry and realign it correctly to the 3D model. Using the XbimToolkit, an IfcBuildingElementProxy can be created utilising the shape representation of the constructed damage shape, then placed correctly in its specified location in the model then subtracted to generate voided shapes in the IFC model using the IfcVoidingFeature. Further enrichment of the model with inspection data and derived attributes from the modelled shapes of damages could be added as implemented in [7, 8, 12, 39, 42]

1.5 Roadmap

The thesis' structure is ordered according to the sequence of steps taken in the implementation of use cases and is organised as follows:
Chapter 2 lays out the required background on the topic of this research explaining the concepts of techniques and tools utilised, and elaborates on some necessary details needed, to give a clearer picture of the methodological approach towards the implementation.
Chapter 3 covers the latest related research published so far relevant to the topic of the thesis.
Chapter 4 offers a detailed view of the implementation on the use case for modelling crack damages.
Chapter 5 reviews the results of the proposed modelling approach for the purpose of this thesis objectively as a proof of concept and goes through the steps taken to verify and validate them.
Chapter 6 provides a summary of the whole research implemented for the purpose of this study, offers concluding remarks and delivers some remarks on difficulties encountered during the implementation of the use cases, suggests further steps that could be taken to improve on the used approach, as well as the potential work of relevant interest to be pursued in the future.

2 Background

In this chapter background information are introduced on topics implemented in this research explaining the concepts of techniques and tools utilised, and elaborates on some necessary mathematical details needed, to give a clearer picture throughout the rest of the thesis.

2.1 Projective Geometry

This term is mainly used to define the mathematical relationship between 2D images and their 3D scene. In contrast to classical euclidean geometry, it introduces an extra dimension for free scaling, often denoted as $\lambda or w$ in literature. Thus a 3D point in euclidean space $[X, Y, Z]^T$ would be defined by an extra parameter as $[X_1, X_2, X_3, X_4]^T$ such that $X = X_1/X_4,\ Y = X_2/X_4,\ Z = X_3/X_4$, where $X_4 \neq 0$. This homogeneous scaling factor solves the limitations faced in euclidean space with cases such as defining vanishing points at infinity or projecting 3D points onto image planes.

2.1.1 Calibration

It is the process of estimating the intrinsic parameters of a camera essential for 3D reconstruction by using a checkerboard pattern. Those parameters are:

- c: principal distance c is the perpendicular distance from the projection centre to the image plane. This term is sometimes confused with the focal length; which is the principal distance of the camera when focused at infinity.
- (c_x, c_y): the position of principal point in pixels.
- s is the skew factor
- k, p: radial and tangential distortion parameters respectively.

It is essential not to underestimate the effect of distortion on the quality of 3D reconstructed point clouds especial for the pixels closer to the borders of the image where the distortion effect is at greatest. Knowing the estimated distortion parameters from calibration, the images could be corrected using the undistortion formulae defined as in Equations (2.1) and (2.2) from radial and tangential distortions [93].

(2.1)
$$x_u = x_d + (x_d - x_c)(k_1 r^2 + k_2 r^4 + ...) + (p_1(r^2 + 2(x_d - x_c)^2) + 2p_2(x_d - x_c)(y_d - y_c))(1 + p_3 r^2 + p_4 r^4)$$

(2.2)
$$y_u = y_d+(y_d-y_c)(k_1r^2+k_2r^4+...)+(2p_1(x_d-x_c)(y_d-y_c)+p_2(r^2+2(y_d-y_c)^2))(1+p_3r^2+p_4r^4)$$

where:

- (x_d, y_d) is the undistorted image point as projected in image plane.
- (x_u, y_u) is the undistorted image point as projected by an ideal pinhole camera.
- (x_c, y_c) is the distortion centre (i.e., usually the principal point).
- k_n is the n_{th} radial distortion coefficient.
- p_n is the n_{th} tangential distortion coefficient.
- $r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$

And the formulae in Equations (2.3) and (2.4) are used for correcting radial distortion only [26].

(2.3) $$x_u = x_c + \frac{x_d - x_c}{1 + k_1r^2 + k_2r^4 + ...}$$

(2.4) $$y_u = y_c + \frac{y_d - y_c}{1 + k_1r^2 + k_2r^4 + ...}$$

2.1.2 Perspective Transformation

In order to establish the relationship between a pixel point in an image to its position in 3D world coordinates, a perspective transformation between all coordinate systems is required. In a perspective camera model, there exist four coordinate systems involved as shown in Figure 2.1, where:

- u, v are the pixel coordinates in an image.
- x, y, z are the coordinates in image plane.
- dx, dy are the translation components in x and y directions from the upper left corner to the the origin in imag
- X, Y, Z are the 3D point coordinates in camera coordinate system.
- U, V, W are the coordinates of a 3D point in the world coordinates.
- f_x, f_y are the components of focal length in x and y directions respectively in pixel units.
- s is the skew coefficient
- R is the $3x3$ rotation matrix.
- t is the translation vector.

- c_x, c_y is the principal point.
- d_x, d_y are the size of pixels.
- λ is the scaling factor.

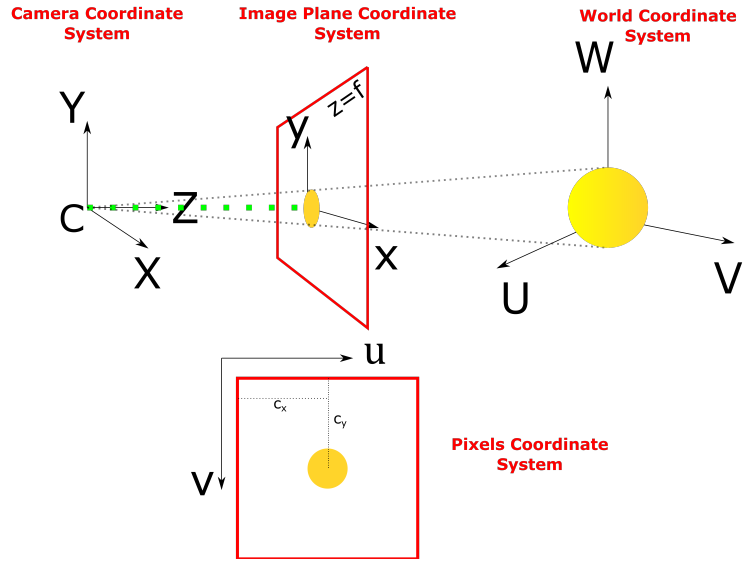

Figure 2.1: Various coordinate systems in camera perspective model.

1. The pixel coordinate system of the image where pixel intensity values are stored with the origin at the upper left corner.
2. The coordinate system of the image plane with the origin located at the intersection point between the image plane and the optical axis of the camera (i.e., principal point). The conversion equation between pixel coordinates and image plane coordinates is shown in Equation (2.6). where (c_x, c_y) is the origin of image plane defined in pixel coordinate system (i.e., principal point).

$$(2.5)\quad \begin{cases} u = \frac{x}{dx} + c_x \\ v = \frac{y}{dy} + c_y \end{cases} \Rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1/dx & 0 & c_x \\ 0 & 1/dy & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

3. The camera coordinate system where the optical centre is the origin the Z-axis pointing towards the image plane, and is related to the world coordinate system through a pose comprising of rotation and translation. The conversion between camera coordinates system and image plane coordinates is defined in Equation (2.6)

$$(2.6)\quad \begin{cases} x = f\frac{X_c}{Z_c} \\ y = f\frac{Y_c}{Z_c} \end{cases} \Rightarrow Z_c \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

4. The world coordinate system (i.e., metric system in our case) that defines the position of objects in our real life 3D metric system, where its relationship to the camera coordinate system is defined by Equation (2.7).

$$\begin{bmatrix} X \\ X \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & t^T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{2.7}$$

Hence, the transformation from world coordinates to pixel coordinates and vice versa requires a conversion between the aforementioned four coordinate systems that can be calculated as shown in Equation (2.8) when simplified.

$$\lambda \cdot Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T & t^T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{2.8}$$

Equation (2.8) constitutes the formula for projecting a 3D point into a pixel coordinate system of an image (i.e., forward-projection). However, to retrieve the 3D position of a pixel p using the inverse of the formula (i.e., backward-projection) only results in calculating the ray from the camera centre to the pixel along which p was projected formulated in Equation (2.9) because Z remains unknown[99]. Several methods exist to solve the problem of finding depth in images, either by using stereo photogrammetry or by taking multiple images with sufficient overlap to to retrieve the missing depth by triangulation. For the conducted research the latter approach was utilised to retrieve the missing depth and back-project specific points of interest for damages into 3D space.

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = C + \lambda R^{-1} K^{-1} p \tag{2.9}$$

2.2 Transfer Learning for TernausNet16

Image segmentation is a type of classification that predicts a class to each pixel in the image. With several CNN architectures for image segmentation available, including VGG, AlexNet, UNet, and GoogleNet for instance, the best segmentation metrics published so far are that of TernausNet, albeit the segmentation task was for medical robotic instruments. However the segmentation metrics published by Benz et al.[10] for segmenting cracks and planking patterns successfuly by retraining the CNN by applying transfer learning proves its viability for other segmentation tasks. The architecture of the TernausNet shown in Figure 2.2 is mainly based on UNet modified with the first 16 layers of VGG network as its encoder. The pre-trained encoder speeds up convergence even on datasets with different semantic features than that used for training it [40], as it doesn't need to be trained from scratch but rather through transfer learning. However, the decoder that upsamples the intermediate feature map does require full training.

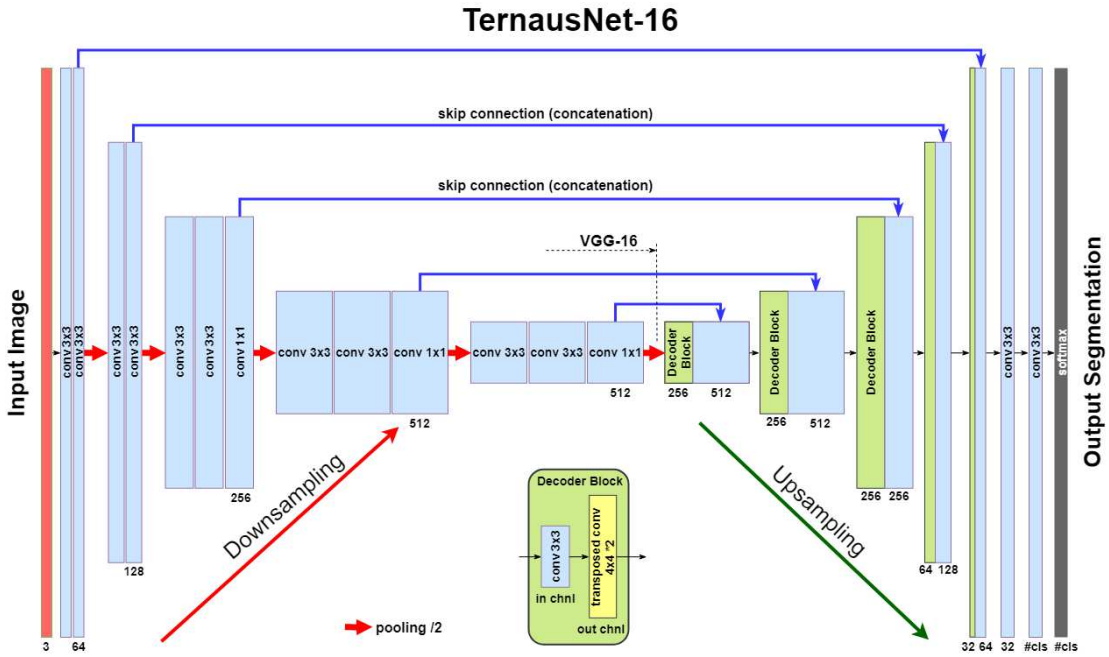

Figure 2.2: Architecture of the TernausNet16. Picture taken from the original paper [86].

In their implementation, the Jaccard index (i.e., Intersection Over Union (IoU)) was used as the evaluation metric. Equation (2.10) defines the calculation formula for two sets A and B, while Figure 2.3 provides a visual representation of its meaning on a Venn Diagram.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.10}$$

The aforementioned expression could be formulated for diecrete objects (i.e., pixels) as shown in Equation (2.11), where y_i and$\hat{y}_i$ are the binary labels and prediction probability of any pixel i.

$$J = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i \hat{y}_i}{y_i + \hat{y}_i - y_i \hat{y}_i}\right) \tag{2.11}$$

Additionally, as in Equation (2.12), the binary cross entropy H was defined as the common pixel classification loss function.

$$L = H - \log j \tag{2.12}$$

2.3 Global Registration using ICP

While most of the available Iterative Closest Point (ICP) algorithms rely on a good initial guess to to converge to a solution, the outcome is highly dependable on the initial alignment. Libraries like Open3D rely on a rough alignment for the initial guess and

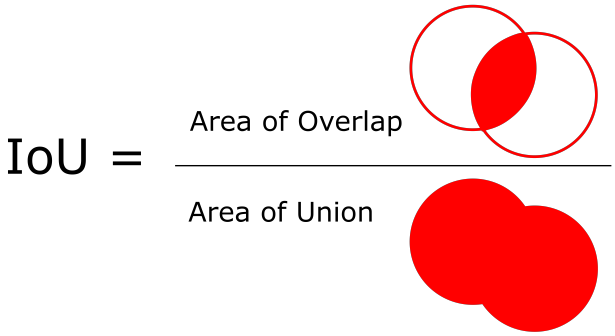

Figure 2.3: Meaning of Intersection over Union (i.e., Jaccard index) commonly used as a similarity measure.

then initialises a second step to refine the coarse initial alignment. Yet still it does not guarantee the algorithm will converge to a global minimum and often stop when getting stuck at local minima.

After reviewing the available algorithms, GoICP [101] was chosen for its robustness at global registration. The algorithm implements the conventional ICP algorithm as a second step within a Branch-and-Bound (BnB) method to search for a global optimum iteratively. When a better solution is found, the ICP is initialised to reduce the objective function defined in Equation (2.13), then the algorithm use the updated upper bound from the ICP to continue with the BnB till convergence [100]. However, GoICP does not include preprocessing methods to normalise, and downsample the source and target point clouds into voxel grids. Other external packages in python such as Open3D and Pyntcloud could be utilised for the downsample voxelisation.

$$\sum_{i=1}^{N} e_i(R,t)^2 = \sum_{i=1}^{N} \left\| Rx_i + t - y_{j*} \right\|^2, \qquad where\ j^* = \underset{j \in \{1,..,M\}}{\arg\min} \left\| Rx_i + t - y_{j*} \right\| \tag{2.13}$$

such that:

- R is the rotation
- t is the translation
- X, Y are the source and target point coordinates respectively, where $X = \{x_i\}, i = 1, ..., N$ and $Y = \{y_j\}, j = 1, ..., M, \forall x_i, y_j \in R^3$

3 Related Work

3.1 Cracks Detection, Segmentation and Properties Retrieval

The traditional approach that predates the latest methods currently used for crack detection and segmentation relied heavily on image processing techniques of which the review conducted by Zakeri et al. for image-based techniques for crack detection, classification and quantification in asphalt pavements [104], while Mohan et al. [67] compiled a collective review concisely explaining those various image processing techniques used in engineering structures mainly of concrete and to a lesser extent of steel.

3.1.1 Image Processing Methods

Nazaryan et al. relied on the centre of gravity (i.e., centroid) of the images' area to calculate the width and depth of cracks [70]. Meanwhile, anisotropic diffusion filtering at the pixel level was implemented in [6, 37] to smooth out noise and artefacts in the background whilst preserving the crack defect contours; thus, improving the segmentation of cracks from a system of images. The load differential method developed by Chen et al. [17] resorted to comparing ultrasonic guided wave signals under the same damage state independent of past recorded damage free data to avoid baseline subtraction under mismatched conditions. The Digital Image Correlation (DIC) method utilised in [3, 14, 35, 41] identified each pixel in a subsequent set of images by examining its neighbouring pixels to measure the full-field strains and displacement of loaded three dimensional objects in addition to visualising the resulting defects. Gunkel et al. [30] developed a package written in R that allowed cracks detection and statistical analysis of their quantities using a shortest path algorithm, where crack clusters were predicted from connected components of pixels given a minimum threshold value, then the cracks' paths were determined by Dijkstra's algorithm. Image based multi-directional crack detection approach utilising Gabor filter was proposed by Glud et al. [29] requiring 5 user-inputs that are dependent on human visual apprehension. The Fast Discrete Curvelet Transform (FDCT) was first utilised in low contrast and dark coloured images along with texture analysis using the Grey Level Co-occurrence Matrix (GLCM) [57] to automatically detect cracks. GLCM was also implemented in [45] in combination with an Artificial Neural Network (ANN) classifier to estimate the cracks' length and width.
Pereira et al. [75] have proposed using two image processing algorithms for crack detection from images captured by UAVs. Their first used edge detection algorithm was based on the discrete differentiation Sobel operator [87], where the corresponding gradient vector or the norm of this vector was calculated for each image pixel. Their second algorithm was a non-parametric filter based on Bayes algorithm, the Particulate

Filter [90] that seeks to relate the probability of an image segment to be characterised by a crack or not, based on pixel intensity and the number of pixels in its neighbourhood. Last but not least, Abdel-Qader et al. [1] compared the effectiveness of four edge detection algorithms on crack images of a bridge surface by using Fast Fourier Transform (FFT), Sobel filter, Fast Haar Transform (FHT) and Canny filter. The FHT is relatively new but it was shown to be significantly more reliable than the other three edge-detection techniques in identifying cracks. FHT transform decomposes the image into low-frequency and high-frequency components. This process is followed by isolating those high-frequency coefficients from which the edge features of an image are identified.

3.1.2 Machine Learning Methods

On the other hand, the rapidly advancing approach for crack detection using machine learning methods has gained more popularity in the last few years, where its outstanding results compared to some of the aforementioned methods [71] encourages the replacement of the preceding traditional methods [53]. Dorafshan et al. evaluated the performance of six edge detectors (i.e., Roberts, Prewitts, Sobel, Laplacian of Gaussian (LoG) in spatial domain, and both Butterworth and Gaussian filters in frequency domain) and compared them to the performance of a fully trained AlexNet CNN concluding superior results to the latter and further proposed a hybrid detector in which sub-images were first labelled by a trained CNN then LoG edge detector was only applied on the sub-images identified as C class to reduce the noise ratio [22].

Zhang et al. [105] proposed a CNN model for binary classification of image patches containing cracks leveraging deep learning based detectors that was successfully applied to images with complex background captured using a smartphone. Xu et al. [98] proposed another CNN structure based on a Restricted Boltzmann Machine (RBM) encoder for crack identification and extraction of comprising image windows containing cracks with a complex background. Fan et al. proposed another CNN structure for classifying crack images in addition to an adaptive thresholding method by searching for a threshold along the principal diagonal of the 2D histogram that stores the intensity of each pixel and the mean intensity of its neigbourhood using k-mean clustering [24].

The DeepCrack implementation by Zou et al. returned results for multiple scales of the input, which were eventually fused in a 1 x 1 convolutional layer, where both an encoder and a decoder contributed to scales corresponding to the pooling or deconvolutional stage respectively and the cross-entropy loss was adapted to incorporate losses on individual scale levels [113]. Comparably, the CNN model proposed by Liu et al. [59] for a CNN named DeepCrack computed the cross-entropy loss on the side-outputs of each scaled level then applied conditional random fields and guided filtering to reach a fused ground truth with no explicit decoder learned.

Benz et al. [10] presented an expanded approach by applying transfer learning for crack segmentation on the model of TernausNet that is per se based on the architecture of the UNet. The implemented CNN named CrackNausNet could learn separate representations of crack and planking by introducing a third class for planking patterns.

Kang et al. [46] proposed a crack detection, localisation and quantification method utilising a three step algorithm. First, a faster proposal region convolutional neural network (Faster R-CNN) was used to detect crack regions. The network contained two different networks: a Region Proposal Network (RPN) and a fast region-based convolutional network (Fast R-CNN) [28]. The RPN provided possible object locations using various bounding box sizes, and as a classifier, the Fast R-CNN proposed the probability of the object. Then, the determined windows were inputted into the modified Tubularity Flow Field (TuFF) [69] algorithm for crack segmentation at the pixel level. Finally, the segmented cracks were processed by the modified Distance Transform Method (DTM) to measure their thicknesses and lengths.

3.2 Spalls Detection, Segmentation and Properties Retrieval

Paal et al. [73] proposed a method to automatically detect spalled regions on the surface of reinforced concrete columns and measure their properties from image data. The region of spalling was first isolated using a local entropy-based thresholding algorithm, then the exposure of longitudinal reinforcement (i.e., depth of spalling into the column) and length of spalling along the column were measured using a global adaptive thresholding algorithm in conjunction with image processing methods for template matching and morphological operations. Their method was tested on a a set of images for damaged RC columns, indicating its validity against manual measurements.

They later improved on their work [73] by adapting the aforementioned algorithms for spalling detection and property retrieval to sufficiently detect the absence of spalled regions on concrete surfaces and detect transverse reinforcement and distinguish it from longitudinal reinforcement. Those enhancements enabled a better classification based on contextual information from depth retrieval pertaining to the amount and type of reinforcement, which is exposed into one of five respective categories: no spalling, spalling of concrete cover, no exposure of reinforcement, spalling which exposes transverse reinforcement, spalling which exposes longitudinal reinforcement, and spalling of concrete which exposes both transverse and longitudinal reinforcement.

Dawood et al. [20] developed an integrated model based on image processing techniques and machine learning to automate consistent spalling detection and numerical representation of distress in subway networks. It consisted of a hybrid algorithm, an interactive 3D presentation, and supported by regression analysis to predict spalling depth. Images were first preprocessed to denoise the image and enhance the crucial clues associated with spalling, then a spalling processor was used to detect distress attributes, providing 3D visualisation model of the defect. The depth and severity of spalling distress were later measured by means of regression analysis model in conjunction with image processing techniques in intensity curve projection. Their implementation was validated through 75 images in which the regression model was able to satisfactory quantify the spalling depth with an average validity of 93%.

Wu et al. [97] proposed spalling detection method by analysing surface roughness reconstructed from point clouds acquired by laser scanning. In the proposed method, After filtering out the the points on ancillary facilities via circular scan-line fitting and large residual error filtering. A roughness descriptor defined as the ratio of surface area to the projected area for a unit, was used to identify high rough patches, then high rough areas on the tunnel surface, such as bolt holes, and segment seams were filtered out as well after classifying them using Hough transformation and similarity analysis to verify its classification. The remaining patches were presumed to be concrete spalling.

Hoang et al [38] proposed another approach to identify image texture for feature extraction and a Piecewise Linear Stochastic Gradient Descent Logistic Regression (PL-SGDLR) for pattern recognition. Image texture obtained from statistical properties of colour channels, GLCM, and grey-level run lengths were used as features to characterize surface condition of a concrete wall. Based on these extracted features, PL-SGDLR was utilised to classify the features into spalling and nonspalling classes.

3.3 Vectorization and Retrieval of Crack Properties

3.3.1 Crack Width, Length and Orientation

Dare et al. [19] proposed 2 algorithms named Route Finder and Fly Fisher to vectorize a crack given the 2 endpoints are manually selected. Assuming the crack is a dark feature in the image, the first algorithm advances step by step on a baseline connecting the two selected points at predefined intervals searching for another point with the lowest intensity value along a profile perpendicular to that line taking into account bilinear interpolation to determine the intensity values off the regular pixel grid if any of the points is not at integer pixel coordinates. The second algorithm (i.e., Fly Fisher) is an adaptation of the former that starts instead from one of the selected points and steadily advances towards the other by casting 100 radial profiles extended radially from -90° to +90° from the baseline. Each profile consists of the sum of 9 intensity values spaced at one pixel intervals along the profile, and the profile with the lowest sum is chosen as the best route.
The width measurement algorithm starts with the polyline representation of the aforementioned delineation algorithms and casts a 21 pixels wide profile perpendicular to the polyline and subsampled 10 times to generate an array of 210 discrete pixel values. A threshold is then calculated to obtain the coordinates for the left and right edges of the crack at subpixel resolution.

Yu et al. [102] proposed a method to calculate the length, thickness, and orientation of concrete cracks through a graph search, given the 2 endpoints are manually selected. Zhu et al. [111] proposed measuring the crack length as the equivalent to the crack skeleton segment length, which is approximated by the height of a bounding box that circumscribes crack skeleton segment points. The crack orientation is the crack skeleton segment orientation, which is indicated by the direction of the bounding box. A distance field that determines the nearest distance for each crack pixel in the map to its boundaries is calculated using an Euclidean distance transform [13]. The average of the distance

values of all skeleton points is calculated, and the doubled result denotes the average crack width. Similarly, the double of the largest distance value that exists at skeleton points represents the crack's maximum width.

3.3.2 Crack Depth Estimation

Almost all Non Destructive Testing (NDT) methods utilised to detect surface cracks in concrete structures depend on one or more of the following physical phenomena (i.e., reflection, scatter, diffraction or wave conversion), where the crack depth might be evaluated from either the directional characteristics of probe position, echo pulse (shape, spectrum) or the time of flight [62].

The NDT methods could be further split into contact and non-contact types. Contact type methods like impact echo [58, 80, 81] and ultrasonic pulse echo [83] can estimate depths with relatively high accuracy. However, they are time consuming because of their static operability limitations and requirement for accurate wave speed values in the medium, which cannot be defined as an absolute constant for nonhomogeneous anisotropic materials like concrete.

Non-contact methods such as Ground Penetrating Radar (GPR) provide high-speed field inspection but with some limitations to the accuracy of location estimations to obtain detailed information and cost due to the weight and complexity of the system configuration to be operated dynamically while performing on-site and real-time data interpretations [34, 52, 79, 95, 103]. Meanwhile, laser systems are limited to the analysis of visible parts due to the laser short wavelength properties, but descriptive results of laser sensors are clearer than the implicit outputs of the GPR [16, 74, 94].

Zhang et al. [107] used the fast dense depth sampling of its target's surface provided by the Kinect V1 infrared depth sensor in addition to fusion technique to completely cover the cracking surface due to the small region coverage of each Kinect frame. However, the raw data contained significant amounts of high-frequency noises and missing captures (e.g., holes on the surface). Yet the implementation proves that even commercial depth sensors not primarily purposed for that goal could be relatively effective to detect the cracking regions of interest in comparison with the results of LiDAR scans regardless of the quality and resolution of both meshes at a fraction of the cost.

Even though image based techniques for crack detection has matured enough, till now it is nearly impossible to directly quantify cracks' depths through colour brightness changes in photos. However, some indirect methods have been proposed.

- The report published by Lu et al. [62] is one of the earliest attempts to prove the feasibility using an ANN to estimate the depth of cracks in pavements. The project developed a prototype automatic system to estimate pavement crack depth. Using high-accuracy laser sensors to measure the crack geometry including crack width, edge slopes and depth. The obtained data was used along with further related pavement information (e.g., type, age, material, function and traffic) to train the

ANN to estimate the depth of cracks by inference. Based on the evaluation results, the developed system was found to detect pavement crack depths with statistically reliable accuracy and practical applicability.

- Adhikari et al. [2] proposed a supervised learning CNN model that predicted the depth of cracks given the average width with acceptable performance of 1.66% for the Mean Absolute Percent Error (MAPE). A drawback to such a model is the presumption of specific width and predicted depth for an entire crack segment which might not match the actual measurements of the cracks.
- Shehata et al. [84] utilised the Make3D toolbox developed by Saxena et al. [82] to estimate cracks' depths based on single still images taken with a KEYENCE (VK-X100) laser scanning microscope of the tested steel specimens. Though the idea itself is worthy of consideration, the use of the Make3D CNN model for inference of crack images on a totally different scale and purpose than the original dataset used for training it, might be a main contributing factor that should be taken into account and further investigated for the inaccuracy and imprecision of their resulting depth estimations to be validated.

3.4 Conversion from Camera's Pixel Units to a Metric World Coordinate System and Alignment to a 3D Model

3.4.1 Features Detection and Matching

Several approaches exist for estimating camera motion based on image-to-image registration of a scene including Structure from Motion (SFM), Visual Odometry (VO), and Visual Simultaneous Localisation and Mapping (VSLAM). However, only the SFM approach will be the covered in this review as it would be solely adopted for the purpose of this research. This technique is used to infer 3D structures and motion of objects from 2D transformations of their projective images without foreknown information of depth and was formulated by Shimon Ullman [91, 92]. To find matching points between image sequences, features such as corner points and edges are tracked between them. Several feature detectors are available for that purpose such as the Speeded-Up Robust Features (SURF) [9], where a Hessian matrix-based blob detector is used to calculate the sums of gradient components and the sums of their absolute values are used. Another algorithm is the Scale-Invariant Feature Transform (SIFT), which utilises the maxima of a pyramid of Difference-of-Gaussians (DoG) as features to calculate a dominant gradient, then the descriptor is rotated to match that orientation making it rotation-invariant [61]. It has a slower performance for detecting features than the SURF, albeit with higher accuracy in feature positions. [47]

The features detected from all the images could then be matched by any of the matching algorithms that track features from one image to another such as the Kanade–Lucas–Tomasi tracker [63] or the Fast Approximate Nearest Neighbour (FLANN) feature matching based on the implementation published by Muja et al. [68]. To remove outliers of incorrectly matched features, the Random Sample Consensus (RANSAC) algorithm is

commonly used, which is a probabilistic optimisation algorithm that randomly chooses 7 matches (i.e., consensus set), fits a fundamental matrix F using the conditions of epipolar geometry and evaluates it on the rest. The algorithm terminates either after a predefined number of iterations or if the model explains a predetermined number of matches well [66].

3.4.2 Linear Triangulation

It is used to calculate the relative 3D positions of points from image pairs, knowing the detected features enables the determination of the relative orientation of each pair of images with the fundamental matrix F. Through the definition of corresponding normal and skewed projection matrices by means of the known F matrix, a function for the linear triangulation of projective object points to remove different scaling factors of both projections could be realized to fit the epipolar geometry. A spatial 3D Homography matrix H could then be determined to transform projective object points in the first projective reconstruction to the corresponding Euclidean object points in the 3D Model. Such transformation applies as well to all object points of the projective reconstruction [36].

3.4.3 Bundle Adjustment

It is considered almost always the last step of every feature-based 3D reconstruction by refining a visual reconstruction to produce a common optimal 3D structure and viewing parameters (i.e., intrinsic and extrinsic camera parameters). The approach of which varies slightly according to the software, computer vision library or package used to implement it. It aims generally at minimising the reprojection error between the image locations of observed and predicted image points using nonlinear least squares algorithms like Levenberg–Marquardt. Below are some of the most commonly used free software for SFM reconstruction:

- 123D Catch: an SFM tool from Autodesk, offered as a web service. This means that the computationally expensive pre-orientation and 3D point calculation are carried out on the server using optimised algorithms. The user receives a finished, meshed 3D model.
- ARC3D: another web service for 3D point calculation like 123D Catch.
- PhotoSynth: a web service from Microsoft to create panoramic images using a stitching algorithm, the relative orientation of the images and the 3D points of the matching points are also calculated, so that a 3D point cloud of the recorded object could also be obtained.
- Visual SFM: an offline SFM tool from Changchang Wu, which provides graphics-optimised algorithms, where the computing time is minimised by a significant factor. In addition, the various steps of the 3D point calculation are well documented and visualised with a very user-friendly Graphical User Interface (GUI) and 3D viewer.

- Meshroom: a free software from Alicevision and is a good alternative to the widely used commercial Agisoft Metashape.
- CMPMVS: a web service from CTU Prague with the option of downloading the precompiled binaries and using the SFM tool offline.
- Bundler + PMVS2: the first SFM tool implementing the complete process chain in one software. The console-based, open-source program, in conjunction with PMVS2, offers a flexible tool with many settings for optimized results.
- OSM bundler: uses precompiled binaries from Bundler, SIFT and PMVS2 in a Phyton script for reconstruction.
- Agisoft Metashape (formerly Photoscan): an SFM tool with a full GUI and 3D viewer. It enables the calculation of the pre-orientation and 3D point reconstruction with different parameters with some capabilities of optimising the result as required by the user.
- SF3M: a tool for the calculation of 3D meshes from images, where the resulting point clouds can be geo-referenced and filtered.
- RealityCapture: constructs textured meshes and virtual reality scenes based on multiple images and/or laser scans, UAVs or synchronised camera rigs.
- OpenSfM: a well-maintained SFM library from Mapillary, written in Python with bindings to C++ dependencies. It consists of basic modules for feature detection, feature matching, and bundle adjustment, with focus on building robust and scalable reconstruction pipeline. It also provides a JavaScript viewer to preview the reconstructed models.

3.4.4 Point Cloud Registration

Though several variants of the ICP algorithm exist, the main differences often lie in the definition of their objective function, that optimises the estimation of the transformation matrix. With an input of two point clouds and an initial transformation guess, the algorithm then finds correspondences from the target point cloud and a transformed source point cloud by iteratively refining the initial transformation through optimising an objective function defined over the correspondence set to a minimum, to determine an optimal transformation that best aligns the two point clouds.

Among several widely used implementations are the multi-resolution ICP algorithm [11, 49], the point-to-plane ICP [18] with a faster convergence speed than the first, and the extended ICP [33] as demonstrated by the implementation of Zhang et al. [107] to estimate a relative rotation and translation of relevant parts of the reconstruction to align them correctly to the 3D model.

3.5 Crack Shapes Construction

Martinet et al. followed later by Desbenoit et al. demonstrated in detail the shape construction of cracks based on crack pattern "P" to be swept along a set of profile curves "C" utilizing boolean operations for unions and intersections of individual segments [21, 65], yet not in a BIM context and with no detailed mention of the surface marching algorithm implemented to map the curves onto the triangulated surface nor the modelling and viewing software tool used to illustrate it. A drawback stated in their proposed marching algorithm is the generation of self-intersecting skeletal elements when large crack patterns are swept on regions of high curvature; however, the same approach is well suited for the purpose of modelling crack geometries to be inserted into building models.

The approach could be further improved by allowing blended sweeping for a set of varying patterns P to account for varying depths of cracks given the availability of those measurements, which is technically feasible using the Blender module for Python for instance to execute a script in the background from the command line and visualise the shapes construction if necessary before insertion into a BIM model and mitigate the possibility of creating self-intersecting faces by checking for their existence and remeshing the shape to eliminate them, need be. To that end, the trained CNN model could be used to infer the ground truth of crack images as binary images. The crack pixels will have to be skeletonized then vectorized to extract the critical points comprising the crack center-polyline then reoriented to fit their actual positions in the 3D model and joined to create the polyline required for sweeping.

3.5.1 Meshing of Damage Shapes

While the IFC schema is capable of creating swept area solids along a Directrix using a geometric representation IfcFixedReferenceSweptAreaSolid, there is no defined representation for blended sweeping that allows a a change in profile of the extrusion, which makes the construction of the crack shapes using external Python libraries more preferable. It's not clear however how nodes for tetrahedra within the modelled meshes could be represented in IFC without invalidating it, as the standard IfcTriangulatedFaceSet used for meshes and tessellated items is limited only to modelling the exterior of the shapes, nor any reference, to our knowledge, could be found proposing an adaptation of any of the IFC entities available within the latest IFC standard (i.e., IFC 4.0 - Addendum 2 - Technical Corrigendum 1) to model FEM compatible shapes.

4 Methods for Modelling Cracks' Geometries and Application on Use Case

This chapter explains the main idea of the thesis and delves into the concepts and methods used to realise it. To achieve that goal a workflow for an implementation pipeline was devised. A Chevron Process diagram in Figure 4.1 explains the general workflow for modelling volumetric geometries of damages in 8 main steps. The chapter is dedicated to modelling cracks geometrically. To further clarify the interconnected workflow, the main 8 steps are expanded in detail in Figure 4.2, following the same colour blocks of the general Chevron Process diagram.

It is worth mentioning that for unavoidable technical limitations of compressing single-pixel lines in large photos for the print version, a zoom-in red bordered window is added in some figures to try to provide the reader with better viewing. However, it is only possible to check the full details of the photos in the electronic version.

Figure 4.1: General workflow for modelling volumetric geometry of damages.

Regarding data acquisition, the workflow was designed based on the presumption of available quality images or video footage from carried out inspections that could be used to extract image frames. Geolocation data was not included in the workflow because of technical limitations of the camera used to capture the images for testing and implementing the use cases. An exemplary image with asphalt cracks of a bike road taken with a Sony Xperia XZ smartphone camera using a an IMX300 sensor was used to illustrate the methods used and capture the images needed for the use cases. For the extremely inaccurate geolocation positioning data of the camera shown in test samples, the possibility to include Global Positioning System (GPS) data was not considered as it seriously affected the quality of the point clouds generated through SFM. However, including geo-positioning data is possible to integrate within the proposed framework, whether by including GPS data included in the the EXIF metadata of images or by providing lists of Ground Control Points (GCPs) and their 2D positions in images, which should in principle improve the quality of reconstructed point cloud. A sequence diagram of the developed code to materialise the proposed workflow, as well as the methods of various classes in Python for vectorisation, backwards projection, estimating correct normals and shape construction are provided in the appendix.

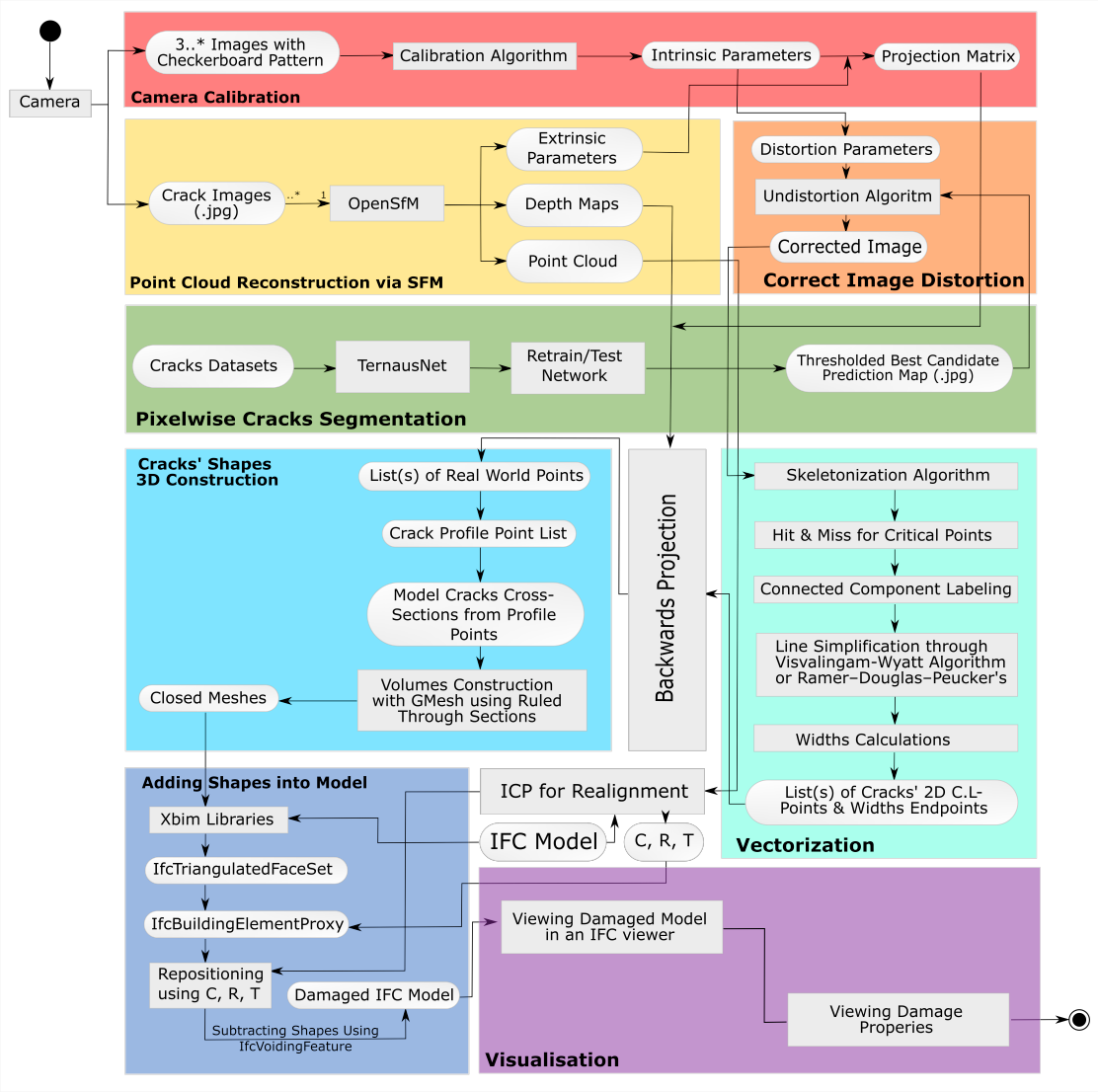

Figure 4.2: Detailed workflow for modelling volumetric geometry of cracks.

4.1 Camera Calibration and Distortion Correction

The calibration of the camera(s) used for shooting the images is essential to determine the intrinsic parameters needed for the 3D reconstruction through SFM as well as for undistorting the segmented masks generated through inference from a retrained TernausNet model. Both MATLAB and OpenCV module for python were utilised as the former produced lower reprojection errors for estimating a Brown-Conrady camera model with 3 radial distortion, and 2 tangential distortion coefficients. However, the parameters estimation of a perspective model with just 2 radial distortion coefficients using the latter (i.e, OpenCV) was found more accurate with lower reprojection errors than that of MATLAB on the same checkerboard pattern photos used for calibrating the

camera used in the use cases. A set of 30 photos was used for a typical 25 mm 10x7 checkerboard pattern. Figure 4.3 demonstrates the input and output results of this step on a calibration image.

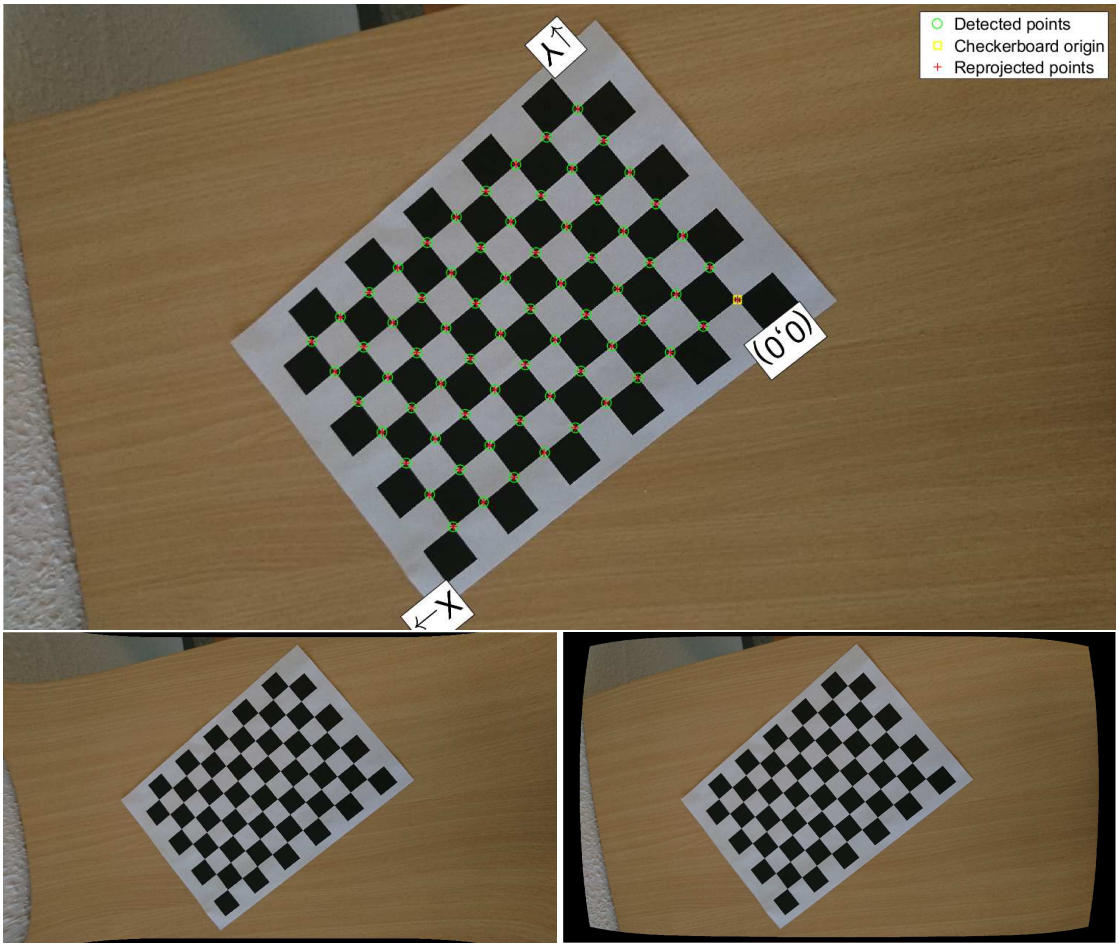

Figure 4.3: Detected corners of the checkerboard pattern used for calibration shown in the upper image for distortion parameters estimation in MATLAB, the same calibration image is undistorted based on 2 radial distortion parameters calibration model in the lower left image, and on 3 radial and 2 tangential distortion parameters in the lower right image respectively.

4.2 Point Cloud Reconstruction via SFM

In order to reconstruct a dense point cloud from the acquired images, several libraries and packages were considered. The OpenSfM library was chosen for the advantages it offers mainly for the superior quality of its resulting point clouds in comparison with other methods, such as: VisualSfM, ORB-SLAM2, the pipeline using Bundler, CMVS, and PMVS2. It requires the installation of other dependencies OpenCV, OpenGV, Ceres

Solver, Networkx, and PyYAML among others listed in the requirements file for the Pip environment. It is written in Python with bindings for C++ through pybind11, that exposes C++ types in Python and vice versa.

This makes it easier to run the commands for 3D reconstruction seamlessly in the background by parsing the necessary commands to the console without the need for manual interaction with a GUI and it offers more flexibility for using its default methods for backwards projection and debug files to retrieve the estimated pose of the camera for each image. It also provides the possibility to control the parameters of the 3D reconstruction, run on multiple threads for faster processing, determine specific values for intrinsic camera parameters, choose from several camera models (e.g., perspective, Brown-Conrady, and Fish-eye models). Furthermore, the availability of several feature detectors to choose from depending on the scene and images acquired and the fact that it integrates external sensor information (e.g., GPS, or accelerometer) for geographical alignment and includes a JavaScript viewer to preview the models and debug the pipeline on a web browser as shown in in Figure 4.4.

After installing all the dependencies required and building the OpenSfM library, the 3D reconstruction was automated by parsing the intrinsic parameters estimated from the calibration to a .json file that overrides default camera settings as well as another configuration file for the parameters controlling the feature detection, matching, bundle adjustment and the required resolution for the depth maps. The images acquired from an inspection could simply be copied to the library's data folder by requiring the user to locate the path to the images directory, then run a shell script that parses the following commands to be passed to a background process in Python.

- extract_metadata: it extracts metadata from images' EXIF tag including location, sensor model, nominal focal length, date and time, embedded geolocation.
- detect_features: it computes features for all images depending on the type specified in the configuration file.
- match_features: it matches features between image pairs.
- create_tracks: it links matches pair-wise into tracks.
- reconstruct: it computes the reconstruction.
- mesh: it adds delaunay meshes to the reconstruction.
- undistort: it saves the radially undistorted images.
- compute_depthmaps: it computes depthmaps as per the specified resolution in the configuration file.
- export_ply: it exports the reconstruction to a .ply file.

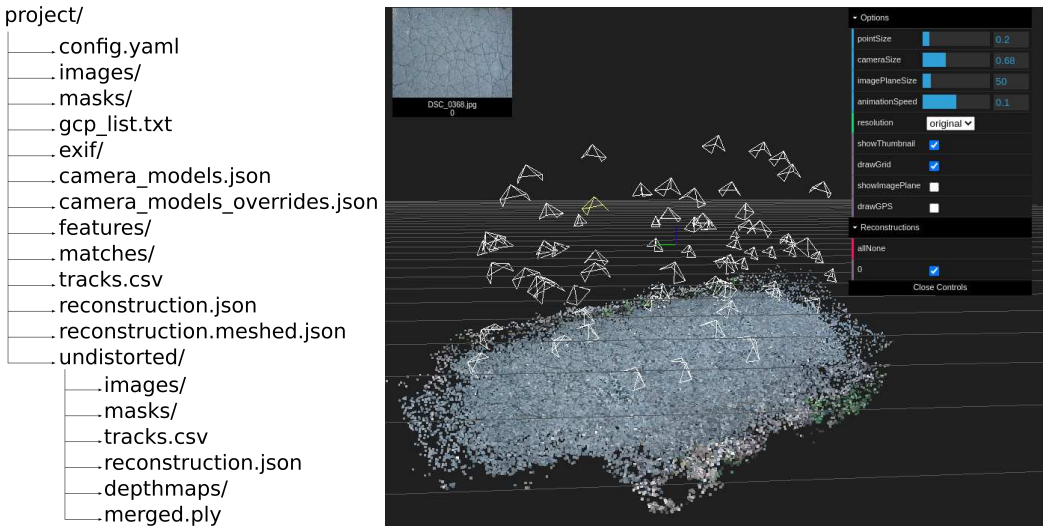

Figure 4.4: Folder structure for an OpenSfM project on the left and a screenshot from the JavaScript viewer for a 3D reconstructed point cloud of the cracks modelling use case on the right.

4.3 Pixelwise Segmentation

As the proposed workflow relies mainly on accurate pixelwise segmentation of cracks for modelling geometries, four published CNNs from Benz et al. [10], H. K. Ha [32], Zou et al.[113], and Liu et al. [59] were taken into consideration for potential use. For the first, only the training datasets were available online, not the retrained model. The second from H. K. Ha based on Unet architecture with a VGG16 encoder similar to that used for the CrackNausNet from Benz et al., albeit trained only for binary classification of cracks, rather than classifying planking patterns additionally, had average evaluation metrics, as the glsIoU (i.e., Jaccard Index) and the mean F1 score (i.e., Dice Coefficient) were estimated at 0.4687 and 0.6033 respectively, even though the model was trained on a very large dataset of 20,000 images.

Both the DeepCrcak models from Zou et al. and Liu et al. were available online with very good evaluation metrics published. However, the model from Zou et al. was found to be producing very thin prediction maps when tested on a never seen collection of images as it was trained exclusively on datasets with single pixels' masks for the cracks' ground truths that often did not fit the actual widths of the cracking patterns exactly when overlaid together as shown in Figure 4.5. This major limitation deemed it unsuitable for the purpose of this research that required a good accuracy for segmentation of the whole crack width. The last model from Liu et al. was observed to perform very poorly, contrary to the evaluation published in [10], when tested on a part of the challenging datasets from Zou et al. that was never included in its training set (i.e., CRKWH100 and CrackLS315). The entirety of tests including the prediction maps inferred from both the crack segmentation CNN of H. K. Ha and the generated fused prediction maps and the refined result with guided filtering from Liu et al. are provided in Appendix A.2.

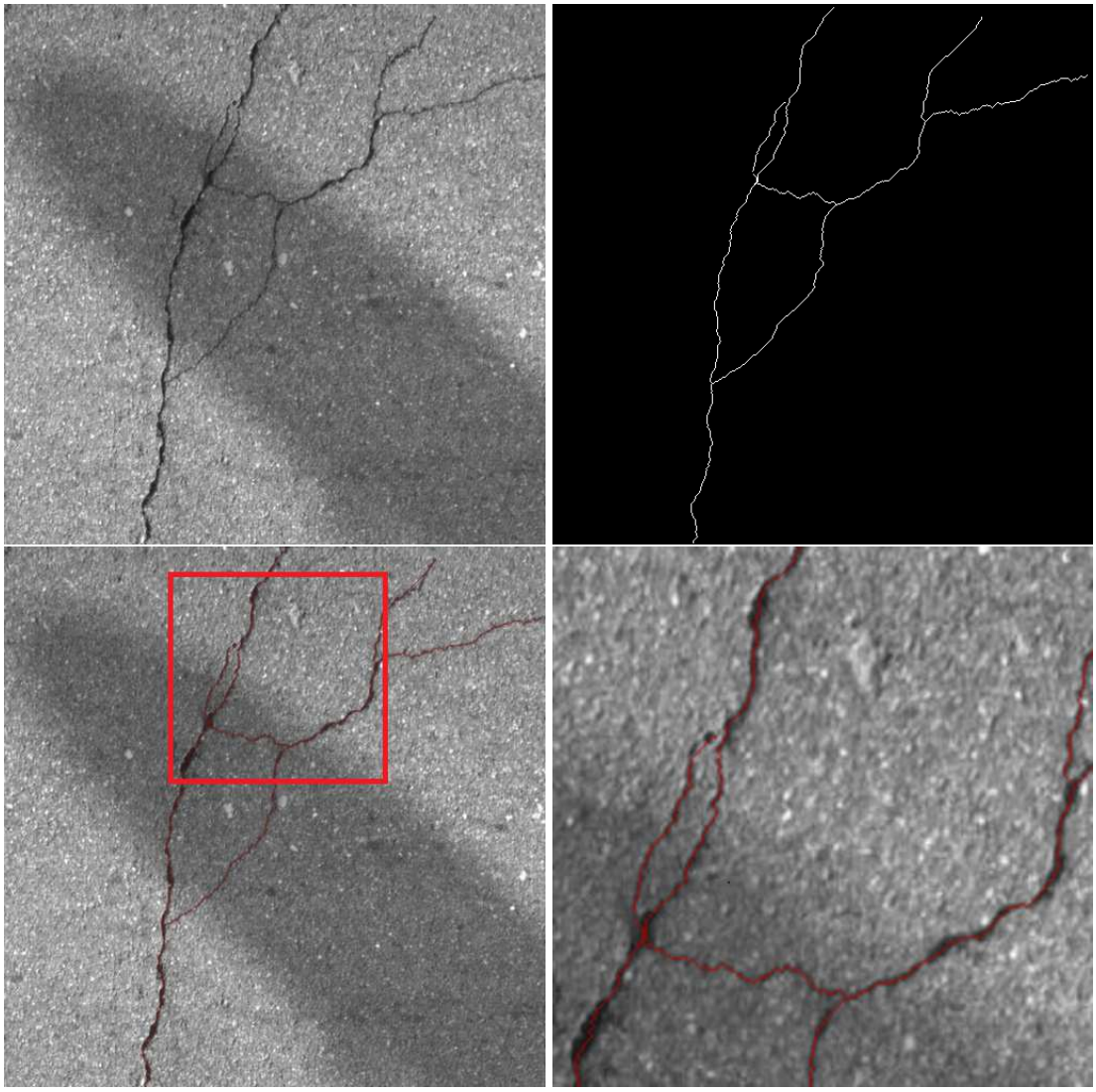

Figure 4.5: Exemplary image from the datasets used for training DeepCrack CNN [113] (i.e., dataset: CRKWH100, image: 1080.png) on the left, its segmentation mask on the upper right, the mask shown in red overlaid on top of the image in the lower left image, and a zoom in crop of the overlaid image at the lower right.

With the lack of suitable models to use, an attempt to retrain a more general and robust model to cracks' thickness was taken to try to avoid the limitations observed in the aforementioned networks. A TernausNet with VGG16 encoder architecture similar to that from Benz et al. was selected to retrain by applying transfer learning on a dataset of 28,800 images and binary masks compiled from the most suitable images of all currently available datasets listed below:

- AEL [4]
- CFD [85]

- CRACK500 [105]
- cracktree200 [112]
- DeepCrack [59]
- GAPS384 [25, 64]
- METU [72]
- noncrack [32]

4.3.1 Dataset Augmentation

Solt package in Python was used to crop the original crack images and masks to 448x448 pixels and augment the datasets to significantly increase their size and further diversify the cracking patterns by applying geometric transformations like random rotation, scaling, horizontal and vertical flipping 15 times for each image in addition to the inclusion of the non cracks dataset from [32] that was already augmented.

4.3.2 Training

Following a similar transfer learning approach for TernausNet16 as that published in [86], a 5-Fold cross validation was utilised, instead of a classical train-test split by holding each fold once for validation against the other folds and using it the remaining K-1 times for training [43]. The Scikit Learn package was used for splitting the dataset into folds after shuffling and the IoU was used as the evaluation metric. The training parameters used for training on Google Colab Pro were as follows:

- Folds Number: 5
- Learning Rate: 0.0001
- Classification Type: binary
- Jaccard Weight: starting from 1 and incrementally reduced
- Epochs: 10/fold
- Workers: 12
- Batch Size = 6
- Crop Size = 448x448

Whereas Figure 4.6 shows the learning curve by plotting the Jaccard score and the validation loss estimated at all epochs.

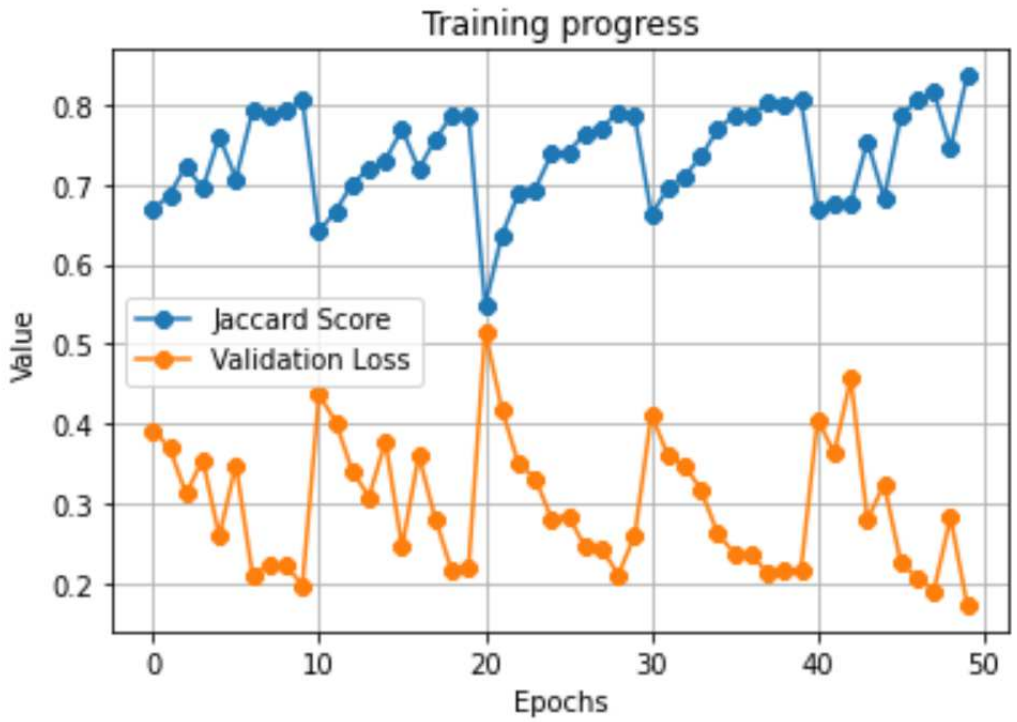

Figure 4.6: A plot showing the Jaccard index and validation loss values over all training epochs for the 5 folds used.

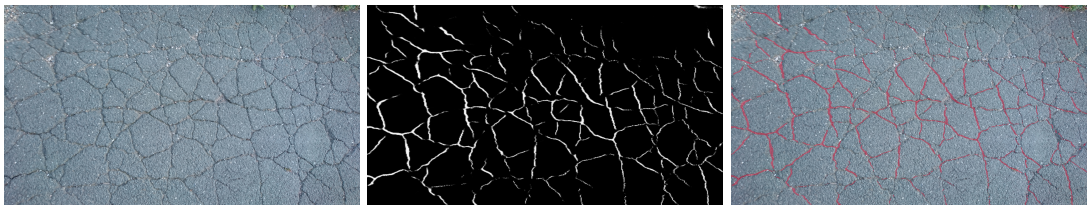

Figure 4.7: An example result from the retrained TernausNet model for pixel-wise segmentation. The original RGB image on the left, the prediction greyscale map in the middle, and an overlay of both images together on the right displayed respectively.

4.4 Vectorisation

In this section, the goal of the steps undertaken was to extract the critical points that comprise the centre-polylines forming the cracks from the segmented probability map delivered from the TernausNet, their widths and the endpoints of the width measurements at each critical point, and the lengths in pixel units.

4.4.1 Thresholding and Post-Processing

By getting the segmented probability maps for cracks images through inference from the retrained TernausNet model as shown in Figure 4.7, the images will need to be thresholded by specifying a limit below which pixel probability values are classified as no crack pixels. Some further post-processing steps like morphological operations on the resulting segmented maps (i.e., opening and closing) might be taken into consideration, depending on the quality of the taken images fed into the trained model, which varies depending on the scene pictured, lighting condition and the used camera sensor, to remove isolated salt and pepper noise that might hinder the consequent modelling process.

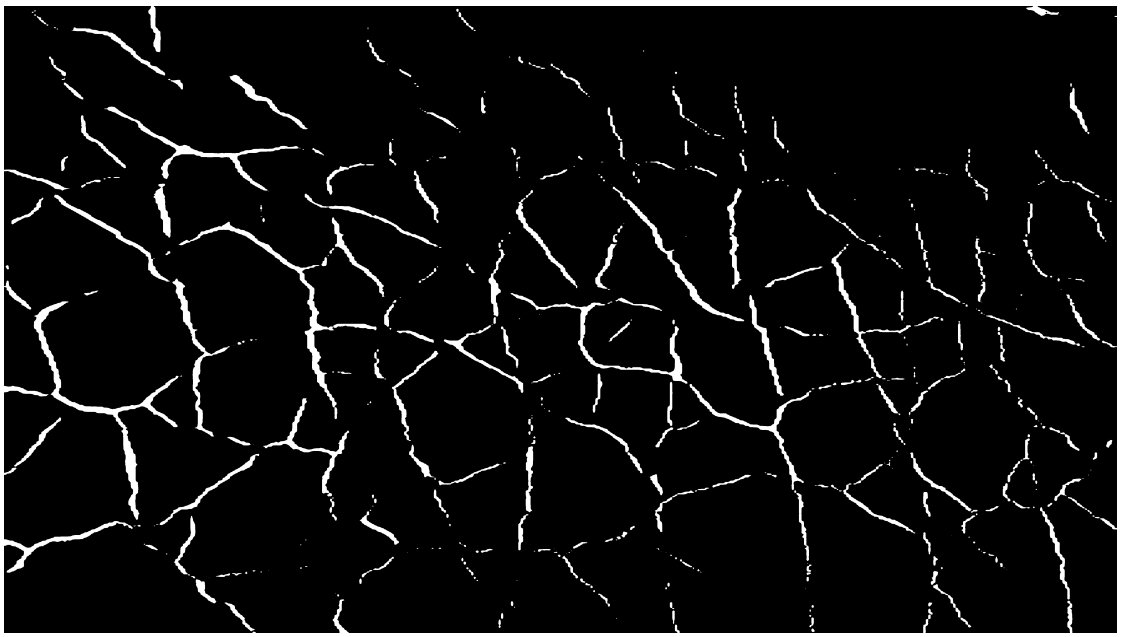

Figure 4.8: Thresholding the cracks foreground in the prediction mask to pixels only above the intensity value of 127 results in a binary segmentation map used.

4.4.2 Skeletonisation

To estimate an average centreline of the cracks in the segmented images, a thinning algorithm is used to reduce the thickness of the crack segments to a single pixel that acts as a centreline for the crack's contour. Several algorithms already exist with varying results for the same input according to the shapes to be skeletonised. For the cracks thinning to a skeleton task, three of the most widely used algorithms (i.e., Guo-Hall [31], Lee [55], and Zhang-Suen [106] respectively) were tested on a sample of 30 images and the resulting skeletonised patterns were visually evaluated for a suitable candidate. While skeletonisation using a Hit-and-Miss algorithm proposed in Section 4.4.3 is possible, it was disqualified from further consideration for its poor initial results on the sample in comparison with the other candidate algorithms. For most images, Guo-Hall algorithm

produced the best results in terms of shape preservation and lines connectivity followed by Lee. However, a common observation among the skeletons produced from all three algorithms, yet to a lesser extent in Guo-Hall, is the occurrence of some artefact shaped ticks at regions of sudden thickness changes or higher curvature, and along the borders of the image, if the cracks are continuing till its very edge. The former could be mitigated by removing these artefacts from the skeletonised image by setting a minimum value for the number of pixels in each skeleton segment to roughly half the average widths in pixel unit, after splitting the skeletonised network of cracks as in Section 4.4.4. To determine that value, the total surface area of the cracks' contours was calculated and divided by the length of the skeletons. The latter was eliminated by padding the segmentation map with a 10-pixels thick border before applying the skeletonisation, and then cropping the skeletonised image back to its original size.

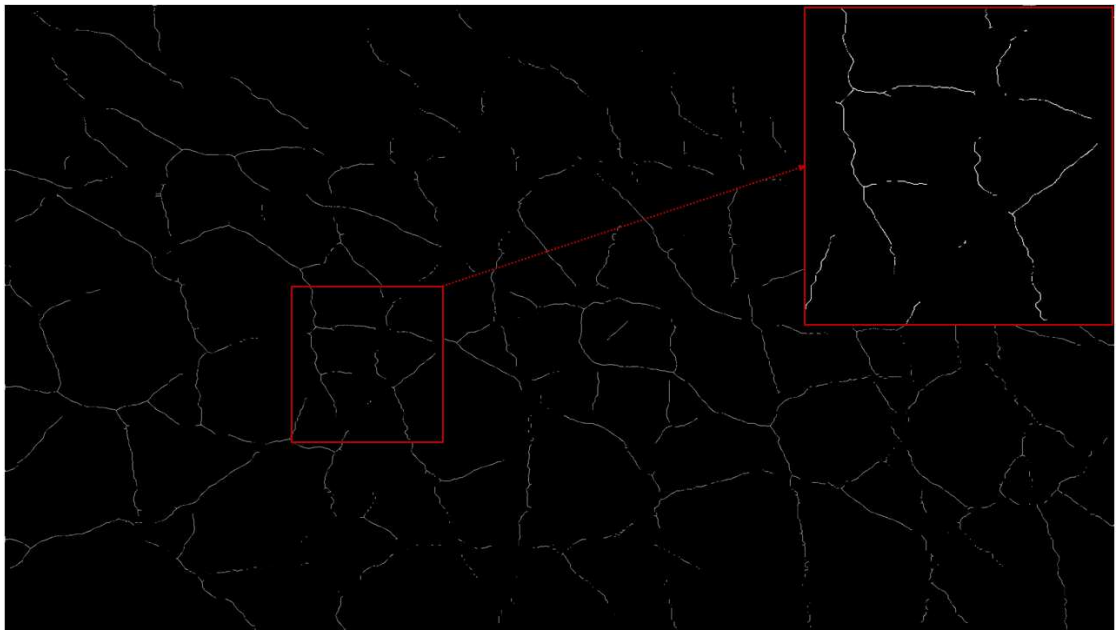

Figure 4.9: Skeletonisation algorithm of Guo-Hall on thresholded crack segmentation mask in Figure 4.7.

4.4.3 Extracting Critical Points

A general case for cracks in which a network pattern of branching and merging crack elements exists was the driving reason to opt for the following solution as the published proposals in literature mainly consider a single line crack or branching pattern but hardly any proposals could be found covering such a general case with interconnected patterns with merging branches as well. To avoid overcomplicating the solution for the task at hand, a simple approach utilising the Hit-and-Miss Hit-and-Miss (HMS) algorithm was adopted. It's a high-level morphological operation that identifies specific patterns in images by comparing each pixel along with its surrounding pixel neighbours as a patch

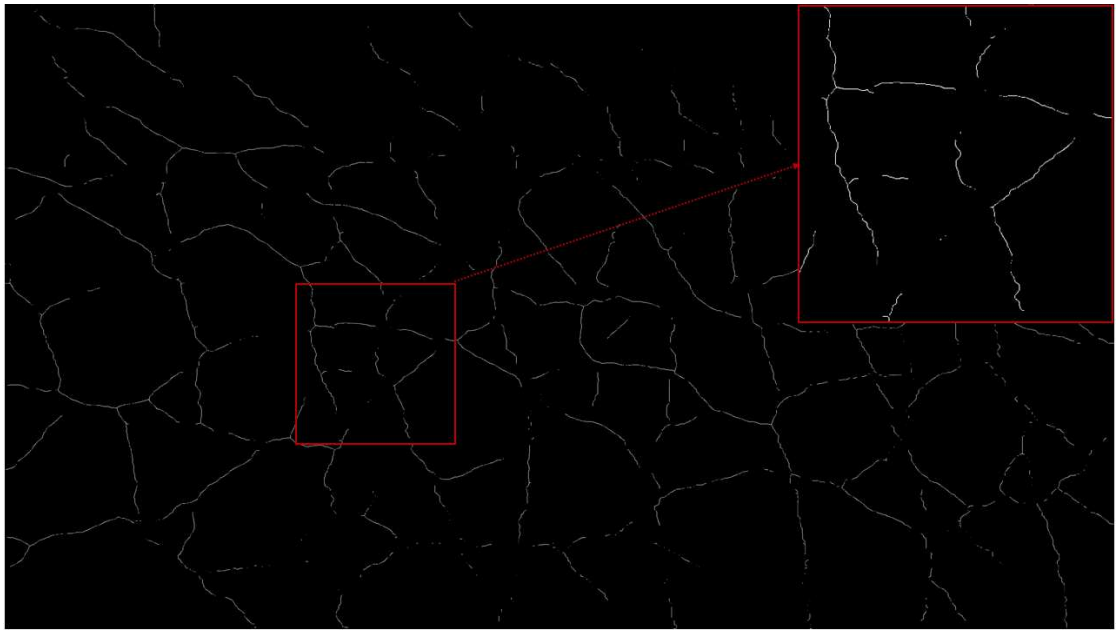

Figure 4.10: Skeletonisation algorithm of Lee on thresholded crack segmentation mask in Figure 4.7.

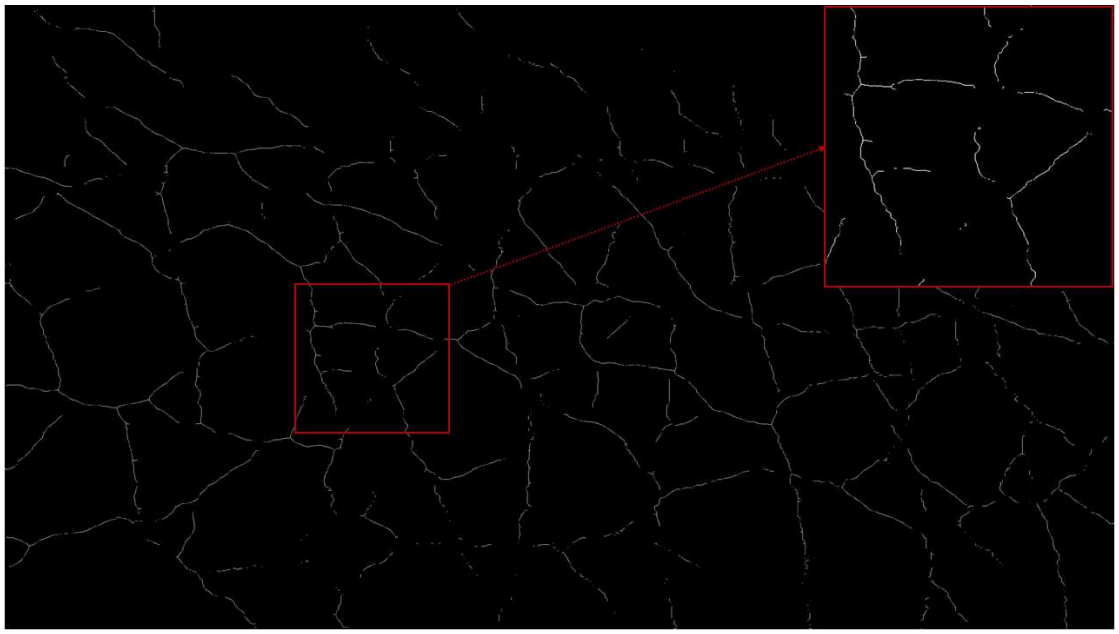

Figure 4.11: Skeletonisation algorithm of Zhang-Suen on thresholded crack segmentation mask in Figure 4.7.

of equal size to specifically configured structuring elements (i.e., kernels) for matches. For each pattern of interest in the image (e.g., intersection and ending points), a set of kernels could be applied to pinpoint only those pixels that match the patterns.

Line Endpoints

To extract the endpoints, the kernels in Figure 4.12 were used. Each 3x3 pixels pattern has to be rotated at 90° three more times to cover all possible configurations (i.e., 20 kernels in total were used). For a pixel to be determined as an endpoint, the black and white pixels in one of the kernels must match that in the skeletonised image when the centre of the kernel is positioned at a white pixel in the skeletonised image. The grey pixels could be either of black or white values.

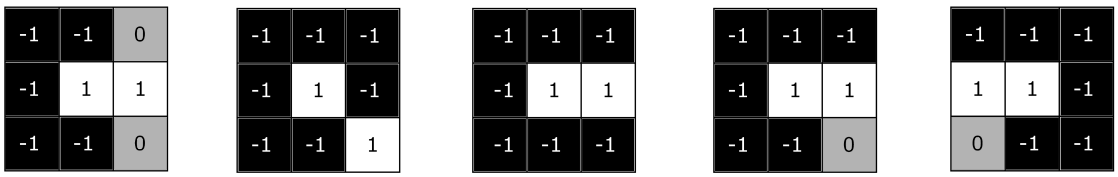

Figure 4.12: Various kernels used to detect endpoints using Hit-and-Miss algorithm.

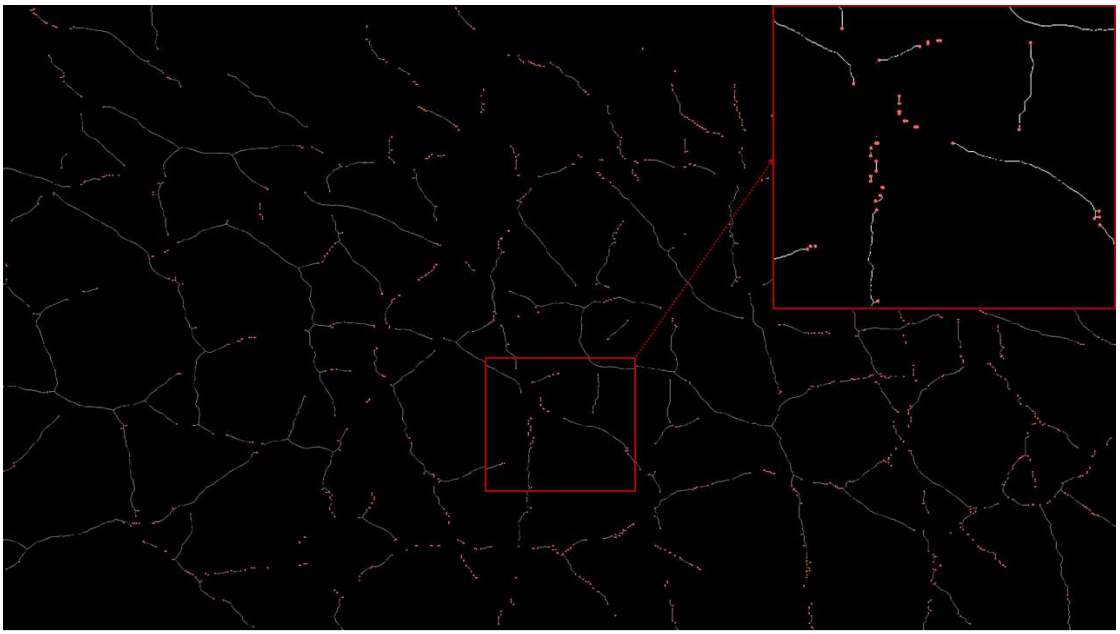

Figure 4.13: Extracted endpoints of skeletonised crack patterns using a Hit-and-Miss algorithm.

Line Junctions

Similar to the endpoints, a different set of kernels K_1 shown in Figure 4.15 were used with the same Hit-and-Miss algorithm to retrieve the set of junction points S_j in the cracks network. Each of the first four 3x3 pixels patterns has to be rotated at 90° three more times to cover all possible configurations (i.e., 18 kernels in total). For a pixel to be determined as a junction point, the white pixels in one of the kernels must match that in

the skeletonised image when the centre of the kernel is positioned at a white pixel in the skeletonised image. The grey pixels as shown in Figure 4.15 indicate it could be either of black or white value.

It was observed, however, that such a general configuration of kernels might result in neighbouring pixels being classified as junctions as shown in Figure 4.14, which in return would further complicate the following step for labelling each crack segment separately. Hence, another set of restrictive kernels K_2 was used by converting the zero values of kernels' pixels as shown in Figure 4.15 to -1, in order to retrieve a subset S_{Ja}, where $S_{Ja} \subset S_J$ for all possible junctions that satisfies the condition of not having any neighbouring junctions N_J within a 1 pixel neighbourhood. This subset defines the nominal position of intersection of all crack segments correctly, where each segment ends once split, yet it is still not sufficient to consider all the cracking segments as separate lines satisfying the condition for an end point as per the kernels in Figure 4.12, when removed from the skelotonised map. A workaround would be to use another subset $S_{Jb} : S_{Jb} \subset S_J \ \wedge \ Jb_i \notin S_{Ja}$ from the set of all possible junctions S_J using the XOR logical operand, such that:

$$S_{Jb} = S_J \oplus S_{Ja} \ \forall \ \{J_i \in S_J : \Longleftrightarrow \ J_i \cup N_{J_i} \in K_1 \wedge \exists (N_{J_{im}} \in S_{J_a}, \ m = [1, 8])\}$$

By modifying the pixel intensity in a copy of the skeletonised map for the entire 3x3 patches of S_{Jb} to the background value (i.e., zero) and retrieving the junction points with either of K_1 or K_2, another junctions set S_{Jc} could be retrieved that splits the rest of the crack patterns, where no neighbouring junctions exist within a 3x3 pixels interval. Both S_{Jb} and S_{Jc} effectively split all cracking patterns once set to the background value no matter where it occurs in the skelotonised map or how many segments intersect in any possible scenario.

4.4.4 Splitting Crack Networks

A first approach was pursued by using a 3x3 kernel starting from one end point to track the neighbouring crack pixels and store them in an NdArray for each segment till it reaches another or a junction. While this simple approach suffices to handle the vast majority of cracking patterns successfully, it was found incapable of dealing with special cases, where more than 4 segments meet at one junction or at the edges and corners of images. In such a case, it's possible to have neighbouring junctions; combined with the possibilty of having such a special case at the border or a corner of the image, made it cumbersome to configure properly. Thus, a second more generalised 3 steps approach was sought as explained in Figure 4.17 by first converting the pixel intensity value at junctions J_i, where $[J_i \in S_{jb} \cup S_{Jc}]$ detected in Section 4.4.3 to zero. Once removed from the foreground, a connected component algorithm can be used to label each element in the skeletonised image separately. The last step would be to utilise the 3x3 kernels to reassign labels to the junction pixels based on its neighbouring labels in a modified skeletonized map of depth 8 (i.e, the grey scale skeletonised map at the first layer followed by 7 background layers of equal shape), which covers all possible label assignments for all 8 neighbouring pixels.

	5541	5542	5543	**5544**	5545	5546	5547	5548
0	0	0	0	255	0	0	0	0
1	0	0	0	255	0	0	0	0
2	0	0	0	255	0	0	0	0
3	0	0	0	255	0	0	0	0
4	0	0	0	255	0	0	0	0
5	0	0	0	255	0	0	0	0
6	0	0	0	255	0	0	0	0
7	0	0	0	255	0	0	0	0
8	0	0	0	255	0	0	0	0
9	0	0	0	255	0	0	0	0
10	0	0	0	255	0	0	0	0
11	0	0	0	255	0	0	0	0
12	255	255	255	255	255	0	0	0
13	0	0	0	0	0	255	0	0
14	0	0	0	0	0	0	255	0
15	0	0	0	0	0	0	0	255
16	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0

Figure 4.14: Example of an intersection problem at the junction [12,5544], where $J_{[12,5544]} \in S_{Ja}$ is a nominal intersection point. However, setting the intensity value of the pixel at this index to zero doesn't split the crack pattern into separate segments, as the pixel at index [11,5544], where $J_{[11,5544]} \in S_{Jb}$, is considered another junction point that does split the cracking pattern when set to zero value.

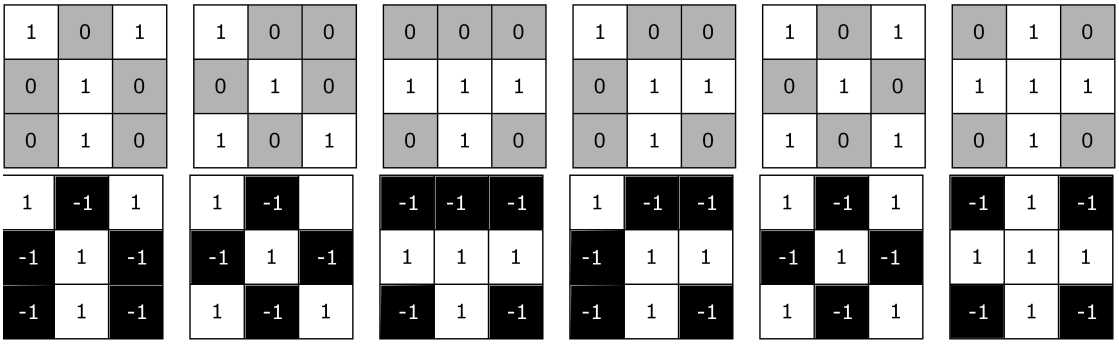

Figure 4.15: Sets of kernels K_1 and K_2 respectively used to detect junctions using Hit-and-Miss algorithm.

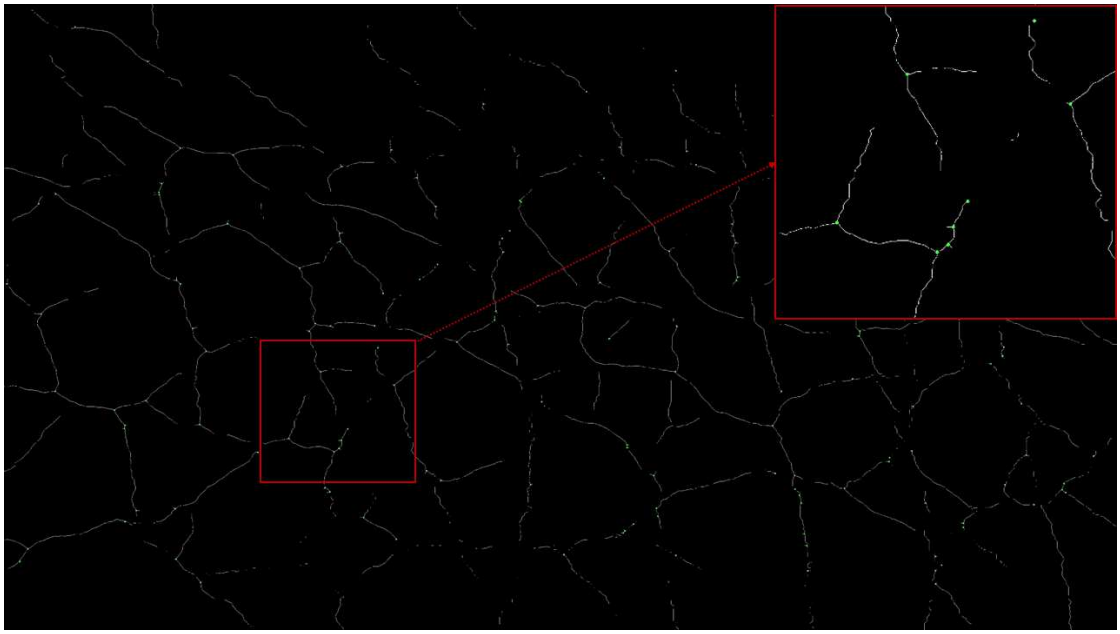

Figure 4.16: Extracted junctions of skeletonised crack patterns using a Hit-and-Miss algorithm.

4.4.5 Labelling of Pixels at Junctions

The conversion of pixel intensity values at the junctions to zero was necessary to split any interconnected pattern to allow the connected component algorithm to successfully label each crack segment separately, yet it resulted in a new problem of reassigning the correct label to the junction pixels. The following criteria were used to reassign a label to a junction pixel:

- No merging of crack segments is allowed, contrary to the approach adopted in [111], to avoid overcomplicating the following line simplification process, as it could be removed by the utilised algorithms, if no condition in place was put to handle them differently, which might result in a different simplification pattern, that does not match the reality insitu.
- Every junction Jc_i in S_{Jc} is assigned the labels of its neighbouring labelled segments in a 3x3 patch centred at Jc_i.
- In a 3x3 patch centred at each junction Jb_i in the second junctions set S_{Jb}, all neighbouring pixels N_{Jb_i} with $(N_{Jb_i} : N_{Jb_{im}}, m \in [1, 8])$ similarly labelled pixels are checked not to match any of the aforementioned kernel patterns in set K_2 in Figure 4.15 when set to a 256 value to filter the branching segments out of the labelling process.

It is worth noting that only four layers are actually sufficient to fill all possible labels at any junction in S_{Jb}, however the presence of extra layers is needed for further work already developed beyond the presented use-case in the thesis, to split junctions bordering other damage types, like spalling for instance, in a more complicated mix´ed damage patterns modelling scenario.

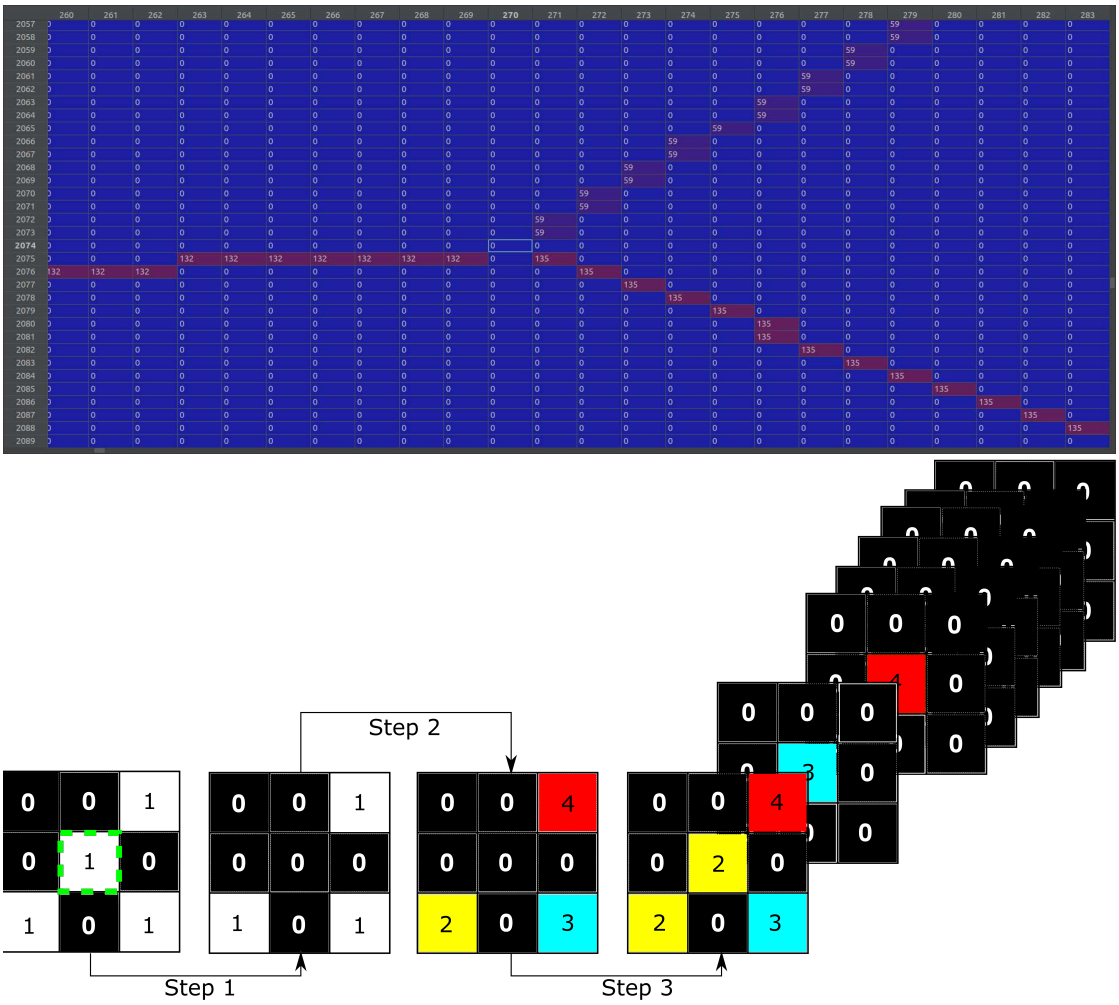

Figure 4.17: Explanation of algorithm to split cracks' networks at a junction in S_{Jc}.

4.4.6 Removing Skeletonisation Artefacts

To remove the artefacts by providing a minimum length for the crack segments in pixels, a rough estimation of the average width is calculated by dividing the surface area of the cracks contour estimated by dividing the surface area of the contours calculated with the Suzuki's contour tracing algorithm [89] over the total length of the cracks' skeletons, following the approach developed by Zhu et al. [111].

Figure 4.18: Labelled segments of a skeletonised segmentation map for cracks assigned a random colour per label.

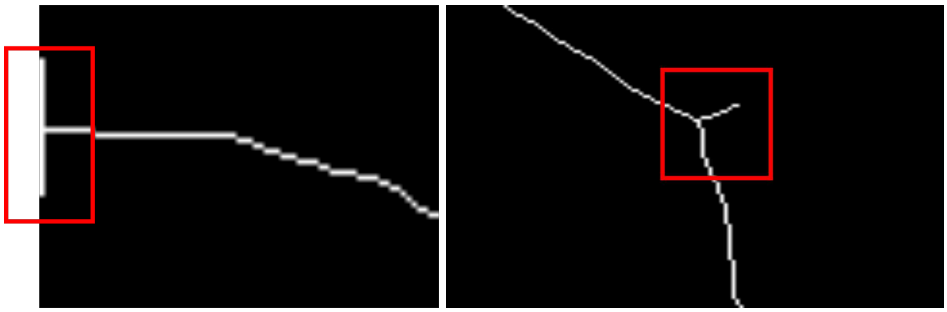

Figure 4.19: Example of typical artefacts at edges on the left and sudden higher curvature regions resulting from skeletonisation algorithms on the right.

4.4.7 Lines Simplification

After splitting the crack networks into separate entities, the position of pixels for each entity could be retrieved and stored by searching for the position of pixels per label. A line simplification algorithm can be used such as Ramer–Douglas–Peucker (RDP) or Visvalingam-Wyatt to decimate the lines to extract only the position of important pixels that preserve the shape of the polyline and help avoid the existence of self intersecting crack profiles along at such points. Both algorithms performed well at lower curvatures in sample tests, albeit the latter slightly better, with more equally distributed points at higher curvatures when a suitable threshold value (i.e., epsilon) is provided. For the RDP and Visvalingam-Wyatt algorithms, ϵ values in the intervals $[W, 1.5W]$ and $[W^2, 1.5W^2]$ respectively, where W is the average width in pixels, were observed to produce optimum results for short and medium length cracks.

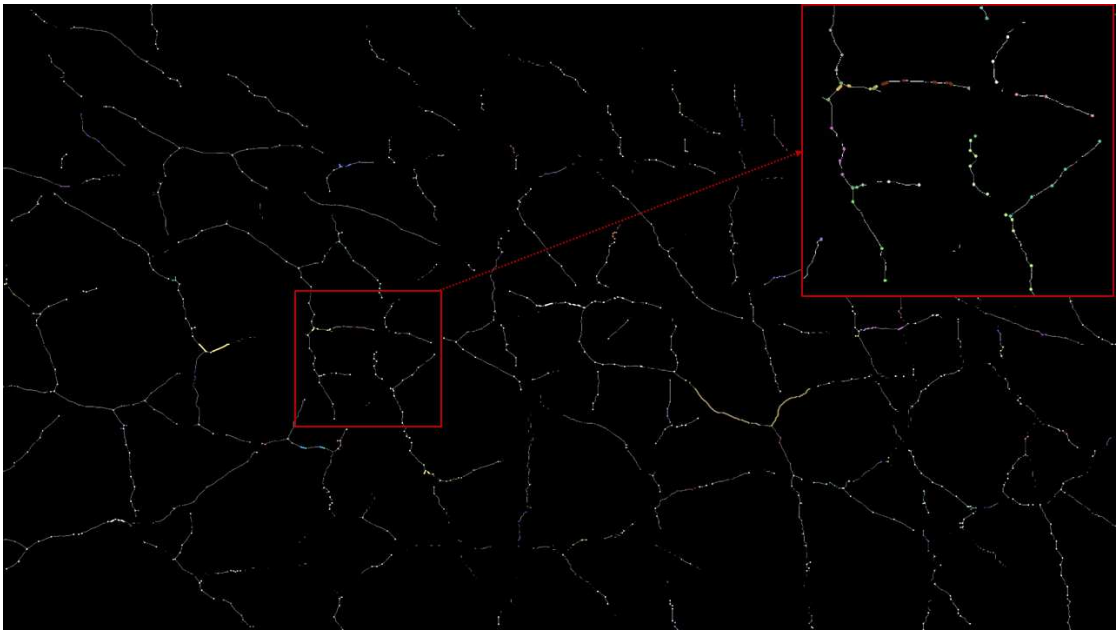

Figure 4.20: Resulting lines simplification from Ramer–Douglas–Peucker algorithm with ϵ=7 pixels.

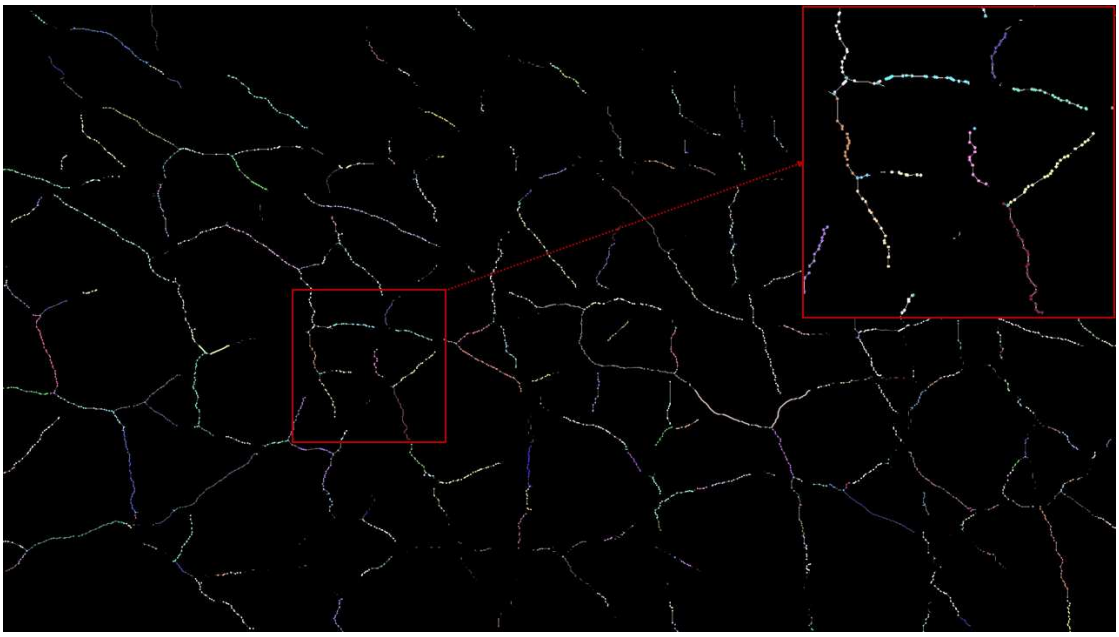

Figure 4.21: Resulting lines' simplification from Visvalingam-Wyatt Algorithm with ϵ=12 pixels.

4.4.8 Determining Width and its Endpoints of Measurement

To calculate the width at the simplified points resulting from the previous step, the Euclidean Distance Transform (EDT) was utilised following the implementation of Zhu et al. [111] for its efficiency. It estimates the euclidean distance between each pixel and

its closest background value (i.e., 0) in a map with distance values gradually increasing till the maximum at the centre-polylines of the cracks, as shown in Figure 4.22. For background pixels, the distance would be of value zero and for points along the skeleton the distance would be to the nearest contour point Pc_1.

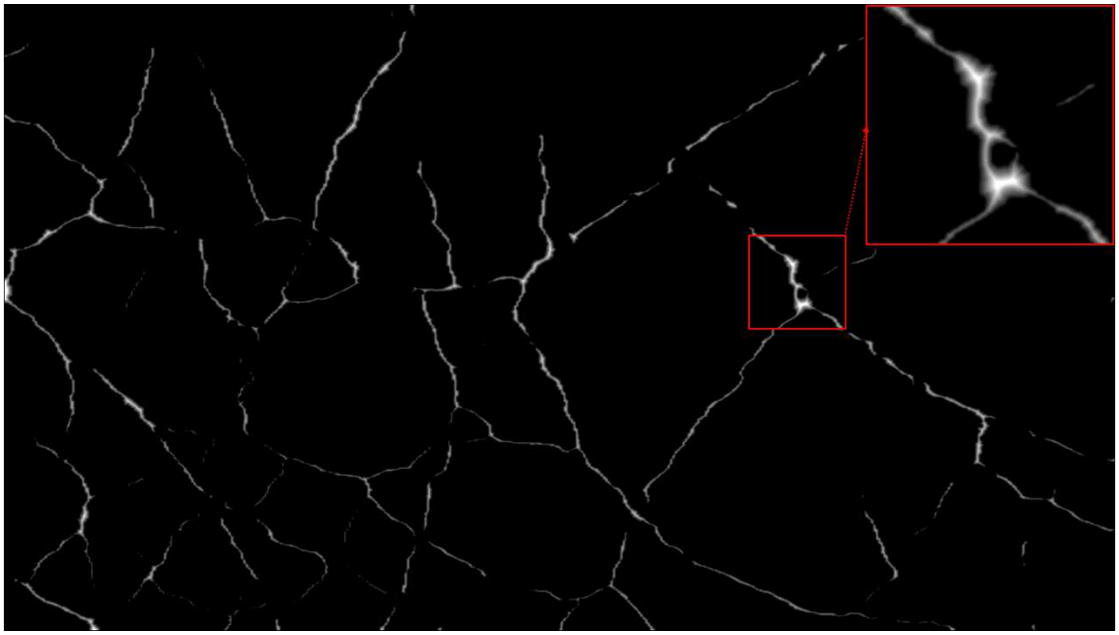

Figure 4.22: A greyscale image of the output map from the EDT operator

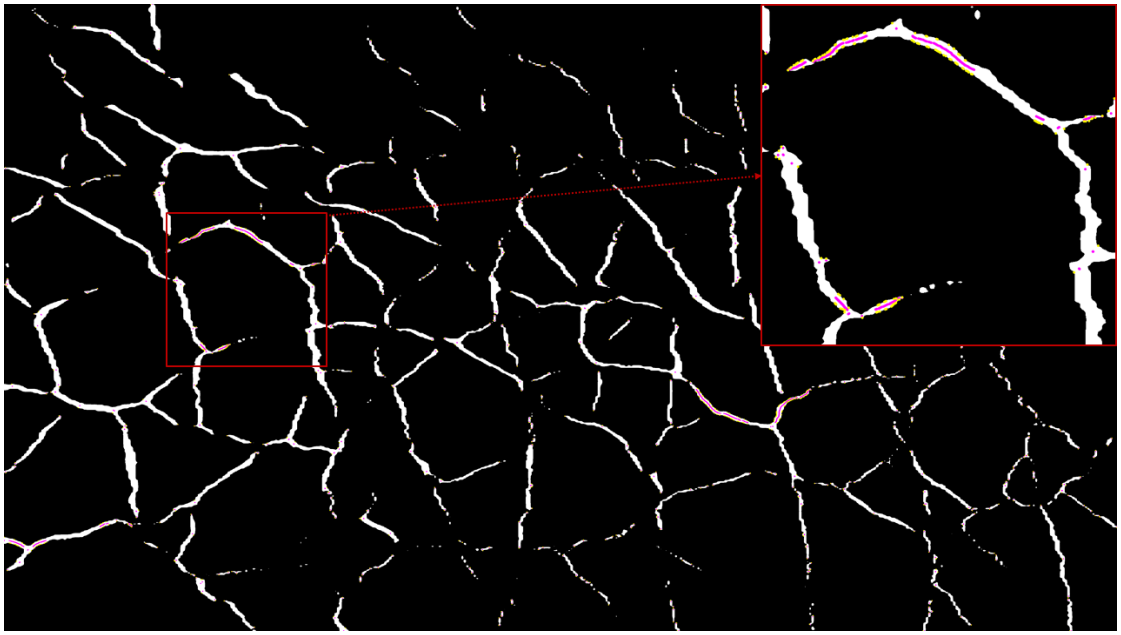

Figure 4.23: Based on euclidean distances, the first end points of the width measurements could be determined and plotted in yellow.

The second endpoints of the width measurement Pc_2 were possible to estimate correctly by determining the direction of a vector from the first endpoints to the centre-points of the crack (i.e, V_{Pc_{1i},Ps_i}), then extrapolating along the unit direction vector to the last non-zero pixel the vector intersects. Another alternative to determine the vector direction, is to utilise the Sobel operator to calculate the gradients of the EDT greyscale map that performs the task similarly as the implemented method, with a special attention to capping the values in both methods to the range of image shape and forcibly correcting the indices to the closest edge value when found to be out of borders.

However, it was observed that the assumption of width to be double the value of the euclidean distance underestimates the value of the actual width in pixel units at any point on the skeletonised cracks, except for the endpoints, where the cracking patterns generally tend to taper. As the skeletonised segmentation map is merely a rough estimation of where the centrelines should be, exacerbated by the limitation of calculating only at pixel level not subpixels.

Correcting the underestimation of widths problem by extracting the correct values through extrapolation resulted in the creation of self intersecting profiles, when the vector along the simplified point Ps on the crack skeleton and its estimated closest contour point Pc retrieved from the EDT are not perpendicular to the simplified polylines (i.e., $V_{Ps_i,Pc_{1i}} \not\perp V_{Ps_i,Ps_{i+1}}$), producing warped cracking profiles that self intersect when the simplified points are not spaced out evenly at higher curvatures, making this approach ill suited for the purpose of modelling valid crack shapes reliably. A more practical alternative considered is classical vector algebra calculations to estimate the distance based on the direction vector perpendicular to the simplfied skeleton polylines.

Another limitation encountered when utilising the EDT is its inherent property to calculate distances based on the closest proximity to the background, which does not necessarily meet the criterion of having an orthogonal width distance to the direction of the polyline required for modelling, resulted in retrieving points that are on either side of the simplified polyline depending on how close the contour is to the the vertices of the simplified polyline even within each crack segment. That irregularity was defined as the main reason for crashing the algorithm for modelling cracking shapes through ruled extrusion in Gmsh, due to its reliance on a sorted sequence of points for each profile. Trying to sort the points $[Pc_1, Ps, and Pc_2]$ in a clock-wise order based on 2 reference points in the image (i.e., the upper left and lower left corners of the map to mitigate the collinearity possibility) did not fully solve the problem, due to the warping effect in the estimated profiles, as well as the varying directions calculated along any two consecutive simplified points (i.e., $V_{Ps_i,Ps_{i+1}}$) within each individual cracking segment.

4.5 Conversion from Pixel Units to 3D World Coordinates

Given the point cloud of the scene reconstructed through OpenSfM, and all the information entailed from the process, it is possible to convert the cracks' points of interest from pixel units into 3D world coordinates in metric units up to scale given that no control point were included in the 3D reconstructed point.

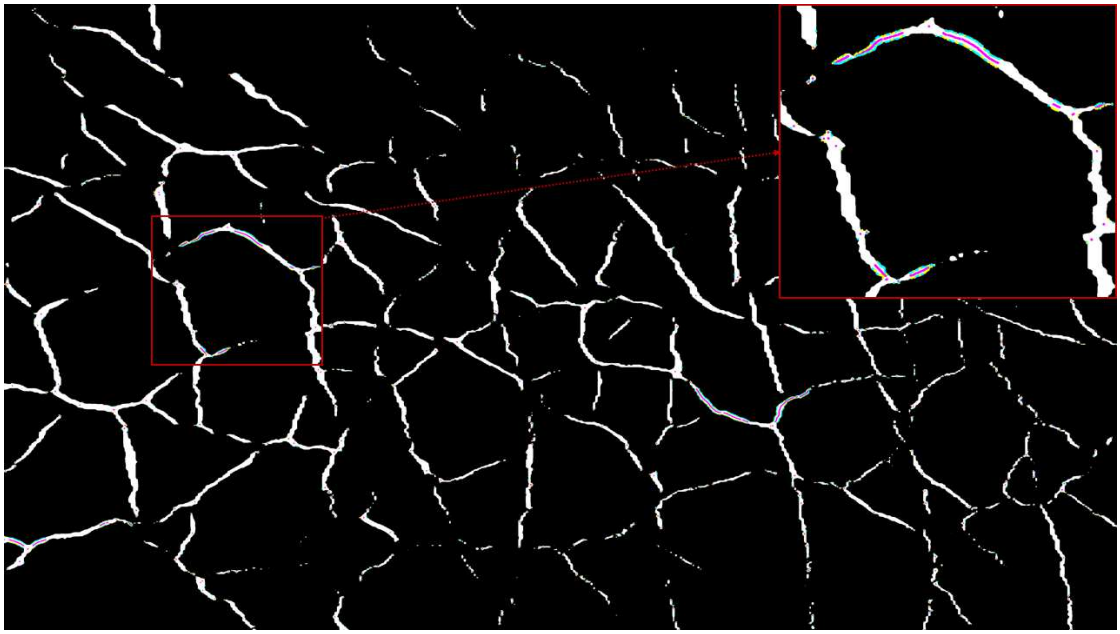

Figure 4.24: Based on the euclidean distance transform map and position of first end points of the width measurements, the second end points could be determined by extrapolation and plotted in cyan.

4.5.1 Selection of the Best candidate Image in the Scene

The selection criteria may vary depending on the quality of the images acquired, the purpose of the modelling, and the region of interest in the structure under inspection. The criterion selected for this workflow was to pick the skeletonised image with the largest number of white pixels as the best candidate for the ensuing modelling process. Other criteria could be quite easily included, as needed, in real world scenarios.

4.5.2 Backwards Projection

With the development of the vectorisation pipeline proposed in Section 4.4, the following sequence of steps could be easily performed:

1. Once the candidate image is identified, its thresholded prediction map from Section 4.4.1 will have to be undistorted according to the distortion parameters estimated from the calibration process.
2. After undistorting the said image, it has to be resized to its depth-map resolution specified from the configuration settings of the SFM process described in Section 4.2.
3. Afterwards, the whole vectorisation process will be performed on the undistorted resized image to extract the centreline points of cracking patterns from their simplified skeletons and both endpoints of their width measurements.

4. With the knowledge of the camera matrix based on the estimated intrinsic parameters determined through the calibration, the depth map, pose of the camera for the candidate image, as well as the scaling factor from the step in Section 4.2, all the unknown variables in the perspective transformation equation are now known to allow the backwards projection of pixel points to their 3D world coordinates (i.e., their correct location in the constructed point cloud in this case, as the geolocation data were purposefully disregarded). Equations (4.1) and (4.2) were used to perform the backwards projection of the specified points of interest, which follow the main equations detailed in Section 2.1, albeit reformulated according to OpenSfM backwards projection method as follows:

   (4.1) $Q_i = d_i.K^{-1}.q'_i$

   (4.2) $P_i = [R'.Q_i - t']'$

   where:

   - Q is the 3D point in camera coordinates.
   - d is the depth at any given pixel index.
   - K is the camera matrix.
   - q is the normalized weighted pixel position in the image.
   - P is the 3D point in world coordinates.
   - R is the 3x3 rotation matrix.
   - t is the translation vector.

5. The accurate estimation of normals is crucial to determining which direction the depth of cracks' profiles could be added in the following step (i.e., opposite direction of the normal to the mesh). For the newly determined 3D points, it could be retrieved from the planes estimation calculated during the step in Section 4.2, but it was found through testing that it requires a large number of neighbouring points to determine a general normal direction of the point to the meshed point cloud, which significantly slowed the point cloud reconstruction down and yielded unsatisfactory results. Hence, determining the normals for the back-projected points was estimated by merging the 3D point to a decimated mesh of their parent point cloud and re-estimating the normals to all points with a large number of neighbours in Open3D [108]; thus ensuring a more accurate estimation of normal direction at said points.

With the lack of accurate geolocaion data from images, or GNSS reciever to measure the location of some control points for the geo-rectification step to provide for an accurate estimation of scale and positioning during the point cloud reconstruction through OpenSfM, it had to be estimated by comparing measurements from the point cloud to that on site as shown in Figure 4.29.

Figure 4.25: The undistorted raw depth map in greyscale for the candidate cracks image

Figure 4.26: The vertices of the simplified polylines representing the cracks' skeletons back-projected into 3D world coordinate units with randomly assigned colour for each segment.

It was observed that along the edges of the main point cloud, the Open3D algorithm for normals estimation tends to flip their direction around the edges of the point cloud as shown in Figure 4.30. The default solution from Open3D is to force a consistent orientation of normals to the plane by propagating the normal orientation using a Riemannian graph, which does result in consistent normals yet they might be all oriented to the backside of the point cloud. This could lead to self intersecting faces in the meshing process in case

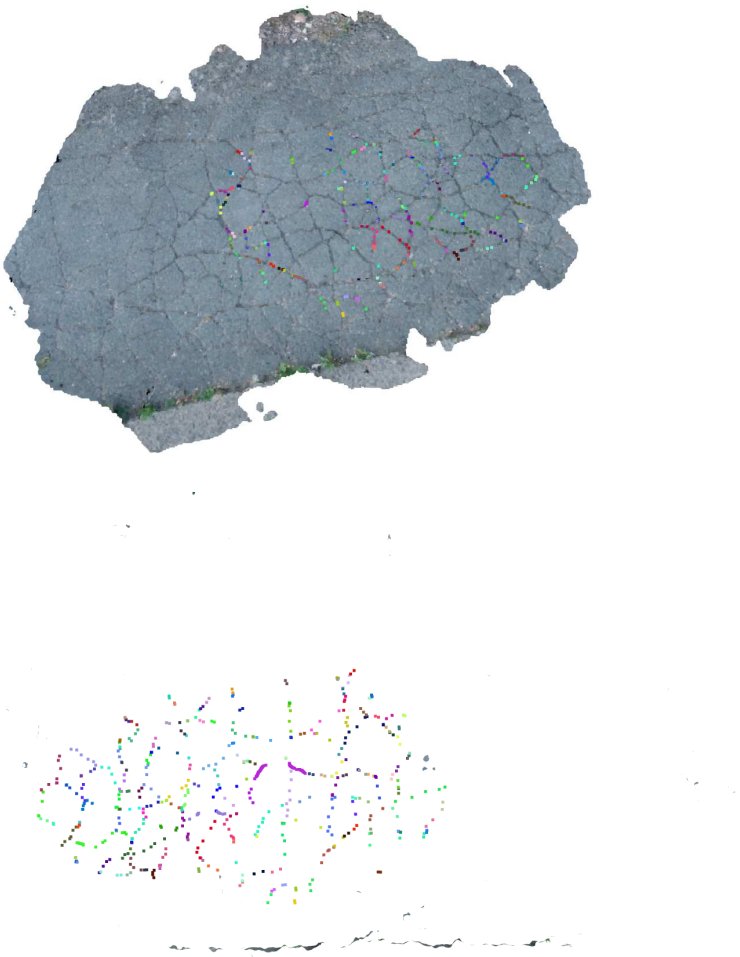

Figure 4.27: The backwards projected points of the simplified polylines correctly align with the parent point cloud when overlaid together from the front and back side.

any of the back-projected points were affected by an inconsistent normal direction, or the crack profiles that were created are not embedded in the model, but rather poking out of the face of the building element of which the point cloud is reconstructed. Hence, it might require human interference in cases of modelling flat regions as chosen for the shown use case to flip the direction when observing the visualisation window in the Python project.

Figure 4.28: Screenshots showing the normals of the vertices in the point cloud correctly re-estimated from the frontside and backside of the mesh respectively.

Figure 4.29: Measurements taken on site and from the point cloud to estimate the scaling factor.

Figure 4.30: Inconsistency in initial estimation of normals to the point cloud that has to be corrected before taking further steps into the modelling workflow.

4.5.3 Alignement of 3D Reconstructed Point Cloud to 3D Model

Having the crack shapes modelled based on the backwards projected points from the point cloud, there exist a misalignment if directly added to the model due to difference in the world coordinate systems of both the reconstrcted point cloud and the model. To solve the realignment problem, a Global ICP algorithm was utilised to estimate the rotation and translation required to transform the pose of the point cloud (i.e., source) to align with the coordinate system of the 3D model (i.e., target). For that purpose, a simple model of the pavement and curb on the side in the use case were modelled in

Autodesk Revit as shown in Figure 4.31 and exported to IFC. The IFC model was then imported into Blender for fine triangulation of the shapes to extract a dense point cloud of the generated meshes' vertices as shown in Figure 4.32.

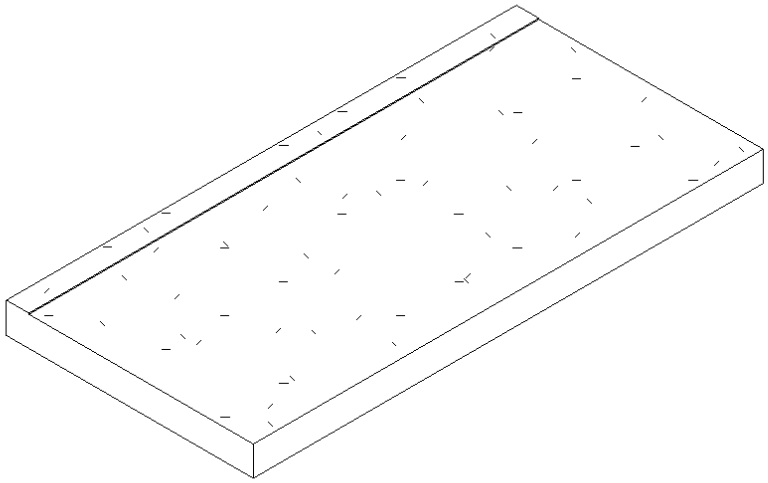

Figure 4.31: Modelling the pavement to actual measurements in mm in Autodesk Revit.

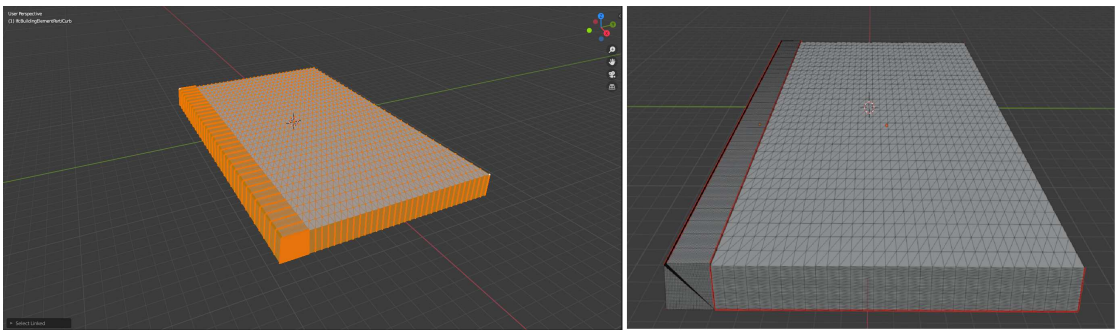

Figure 4.32: Meshing the shapes of the IFC model in Blender required for registration through ICP.

The point-to-point and point-to-plane algorithms provided in Open3D Library were first used to retrieve the transformation matrix. However, even with such a simple model the results were often unreliable and highly dependable on the initial guess derived from the initial RANSAC based global registration performed. Therefore, it was found more suitable to use another alternative algorithm that would provide for a more reliable alignment independent of the initial guess using GoICP [100, 101], that is more robust against noise in the source point cloud resulting from registering a 3D reconstructed point cloud of a rough-textured pavement to a very smooth-surfaced target. Figure 4.33

shows the result of the global registration of a point cloud generated from a decimated mesh of the 3D reconstructed dense point cloud via OpenSfM to the point cloud of the IFC model generated in Blender.

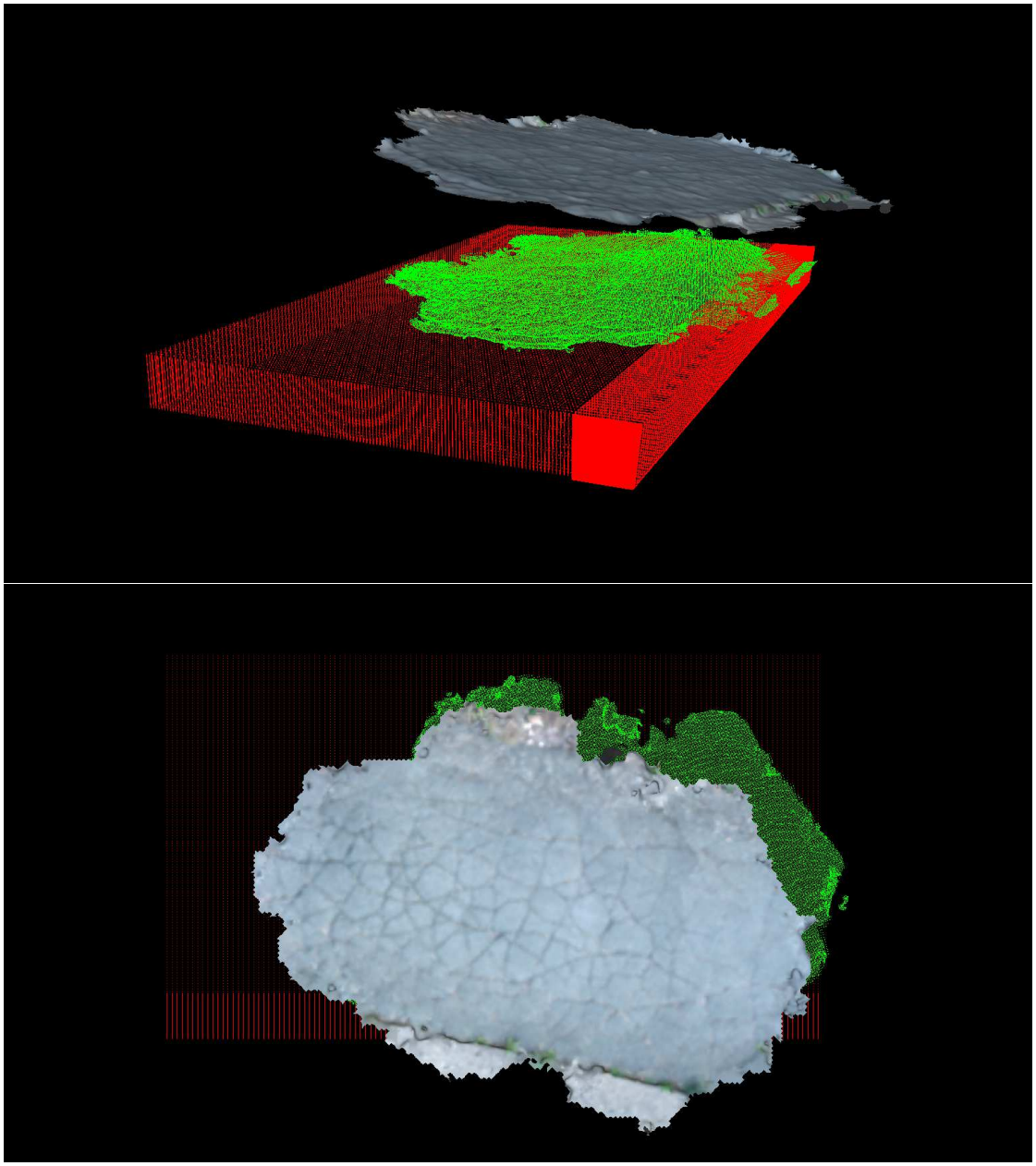

Figure 4.33: Resulting alignment of the source point cloud from the GoICP registration is shown in green, the original decimated source point cloud in real colours of the mesh and the point cloud of the 3D model in red.

GoICP requires the source and target point clouds to be normalised (i.e., all 3 components of the vertices within a $[-1, 1]$ values interval), which is not pre-included in the package and has to be done prior to the registration process and right after voxel downsampling

according to Equations (4.3) and (4.4). The 3D coordinates transformation requires a local translation for each point cloud and a common scaling to be applied on the source and target. After retrieving the transformation matrix from global registration (i.e., $\tilde{H}$), it has to be reverse conditioned following Equation (4.5) to use appropriately on the original coordinates of the source (i.e., H).

$$T_{4x4} = s \,.\, t = \begin{bmatrix} 1/s & 0 & 0 & 0 \\ 0 & 1/s & 0 & 0 \\ 0 & 0 & 1/s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -t_1 \\ 0 & 1 & 0 & -t_2 \\ 0 & 0 & 1 & -t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$\tilde{x_i} = T \,.\, x_i \tag{4.4}$$

$$H = T^{-1}\tilde{H}T \tag{4.5}$$

With the relatively small size of the model for the use case, it was obvious which building element the point cloud belonged to. However, the use of precise geolocation data (i.e., latitude, longitude within 1 cm accuracy, and altitude) for full scale models, which is normally available at large-scale scanning operations with surveying grade equipment using Real-Time Keinematics (RTK) capable Global Navigation Satellite System (GNSS) receivers, it should be possible to get an accurate alignment of the 3D point cloud reconstruction that could be relied on to roughly estimate to which building element the damage shapes should be aligned, by locating the building element closest to the point cloud in a correctly geolocated 3D model.

4.6 Cracks Shapes 3D Reconstruction

For the reconstruction of cracks' geometries, Gmsh library's Application Programming Interface (API) for Python was used. It is an open source 3D finite element mesh generator with a built-in CAD engine and post-processor. It also uses OpenCASCADE for constructive geometry features, and interfaces the optional external mesh and mesh adaptation librairies Netgen and Mmg3d respectively, in addition to a cross-platform GUI based on FLTK and OpenGL [27].

To model the cracking shapes for the use case, a simple triangular profile for the cracks' cross-section connecting both endpoints of the width measurements and a third point, calculated by translating the centre-polyline point a distance of magnitude equal to the depth in the opposite direction of its unit normal vector, was used to reduce the time and complexity required for meshing, but it is technically possible with Gmsh using its default Computer Aided Design (CAD) engine to create simple extrusions along smooth curves or the more advanced option utilising OpenCascade CAD geometry engine to create semi-elliptic profiles or any other closed profile shapes with lines, splines or higher degree Bsplines, necessary for constructing volumes.

As a proof of concept only one junction of the whole cracks network was modelled. Two modelling approaches were experimented to test the feasibility of the proposed workflow.

1. The first by considering a simple approach to extrude the triangular profile of the crack through a spline connecting the centres of mass of the profiles drawn at each back-projected vertex as shown in Figure 4.34. This profile is created based on calculating the average width for the crack segment to be modelled, a depth specified by the user and the inwards direction, retrieved by flipping the normal orientation already known at each vertex of all crack segments. A downside to this approach is the use of an average width at each profile instead of a varying width at each vertex of the spline and converting the cracking pattern from a sharp polyline-based shape to a smooth curve that doesn't exactly resemble the actual cracking pattern on site, which might be difficult to rely on as a reference to compare with in subsequent inspections.
2. The second more sophisticated approach developed is to force the creation of ruled surfaces between each two profiles, thus enabling the use of variant width measurements in each crack segment and following the actual cracking pattern as closely as possible to obtain shapes that could be used as a reliable reference. Figure 4.38 shows the difference between the two approaches for the same junction chosen to be modelled as a proof of concept.

4.7 Adding Shapes into Model

The last step in the workflow would be to add the shapes in the model that is achievable within 2 sub-steps detailed below.

4.7.1 Voids Subtraction via Boolean Difference

Using the approach to model damages geometrically proposed in [7, 8], The cracks shapes were exported from the Python project as .stl files, which were imported into Blender and exported into a new IFC project using Blender BIM Add-On as an IfcElement of class IfcBuildingElementProxy. A script written in C# using XBimToolkit [60] to add the crack elements into the main IFC model as follows:

1. The cracks building element (i.e., IfcBuildingElementProxy) is copied into the BIM project.
2. A new instance of IfcVoidingFeature is instantiated and assigned a name 'Void', a GUID and an object type labelled 'voiding feature' and given a type of label 'CUTOUT'.
3. The shape representation of the cracks proxy is assigned to the newly created voiding feature using IfcProductRepresentation.

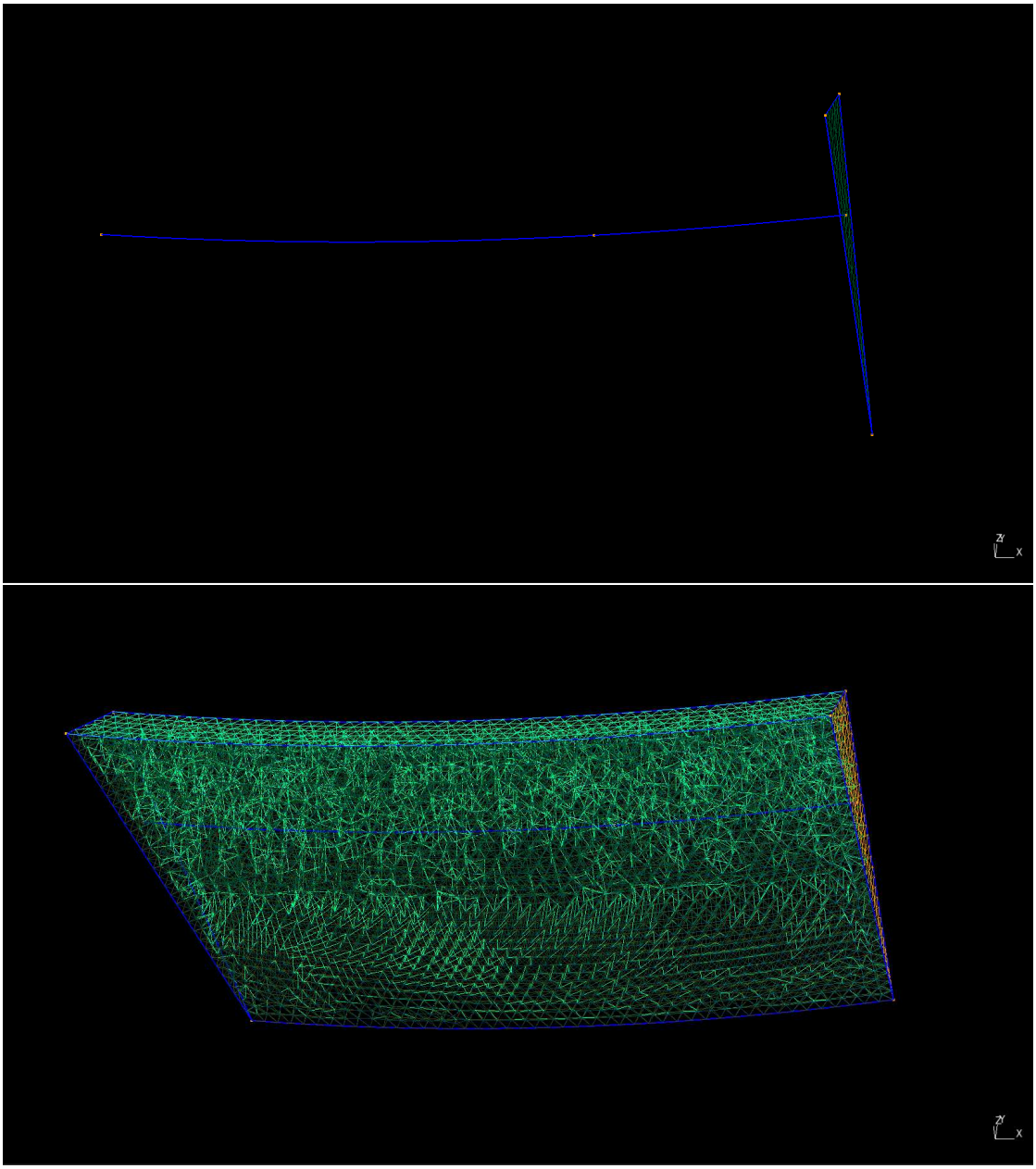

Figure 4.34: The process of modelling a crack shape by extruding a triangular profile along an exemplary 3-vertices spline passing through the centres of mass of each profile calculated at each vertex.

4. The voiding feature location is assigned based on that of the cracks IfcProduct using IfcObjectPlacement.
5. A decomposition relationship of type IfcRelAggregates is created to relate the new voiding feature to the IfcSlab of the pavement containing it.

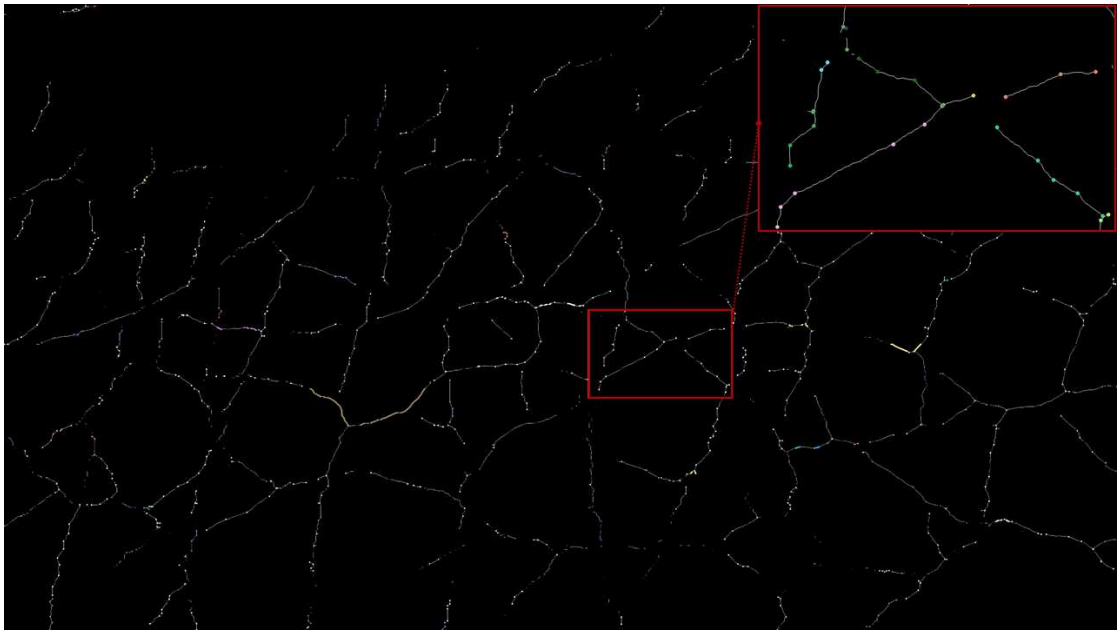

Figure 4.35: Location of the selected use case to be modelled in the simplified lines map created from Section 4.4.7.

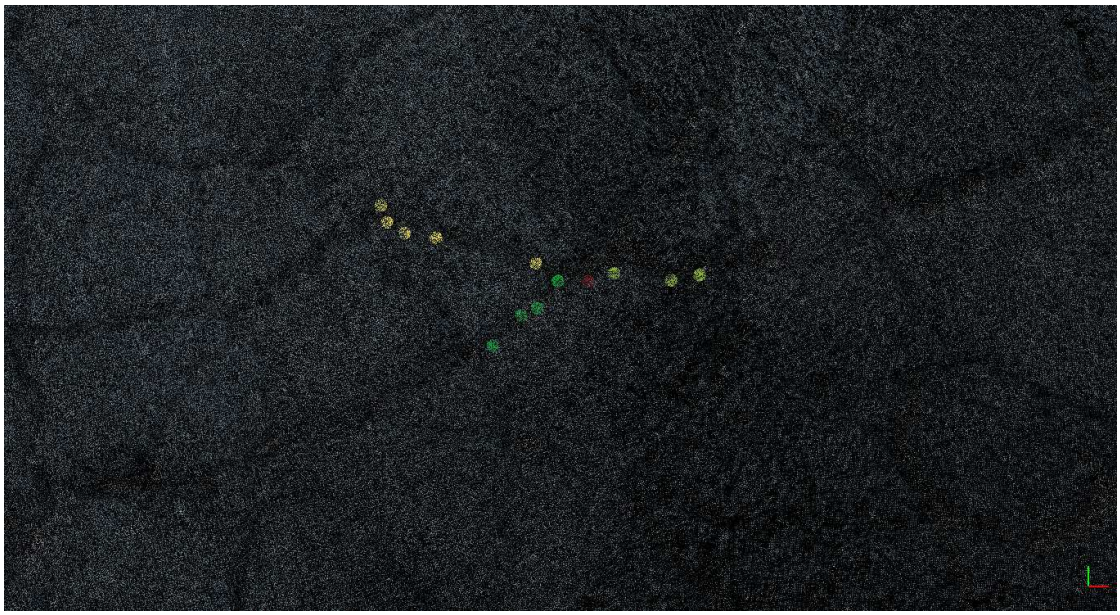

Figure 4.36: A screenshot showing the location of the polylines' vertices of the use case overlaid onto the point cloud.

6. The feature element subtraction is generated by creating another decomposition relation linking the related opening element (i.e., the voiding feature) to the relating building element (i.e., the pavement's slab) using IfcRelVoidsElement.
7. The IfcBuildingElementProxy entity for the cracks blocking the view of the voiding feature is deleted from the project.

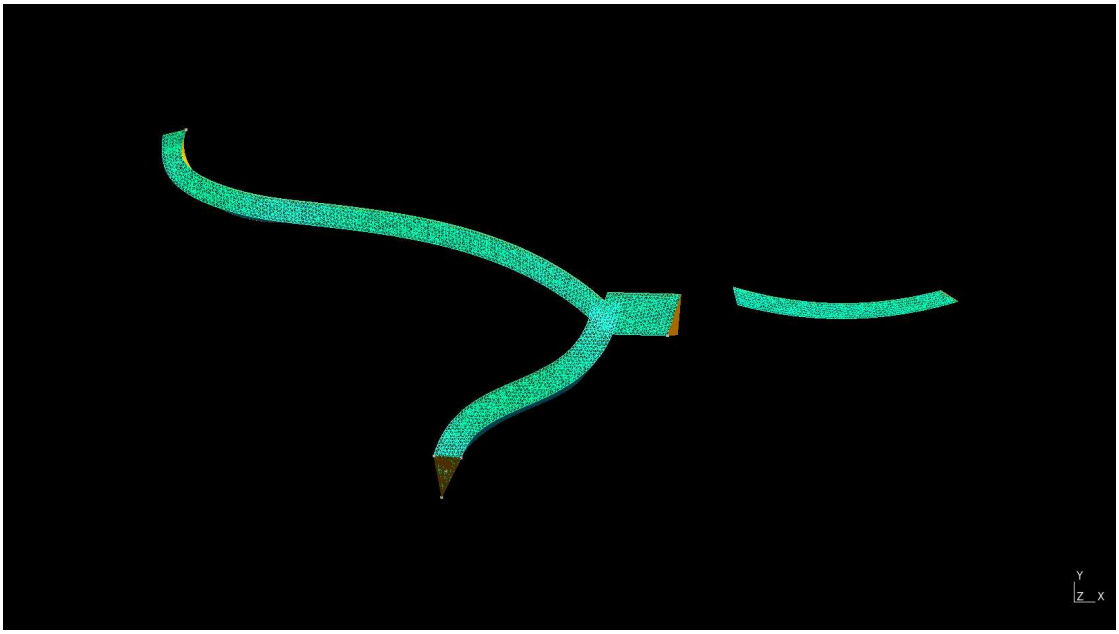

Figure 4.37: Constructing the shape of a cracking pattern at a junction with a profile extruded along a spline passing through the centres of mass of all profiles calculated at each vertex.

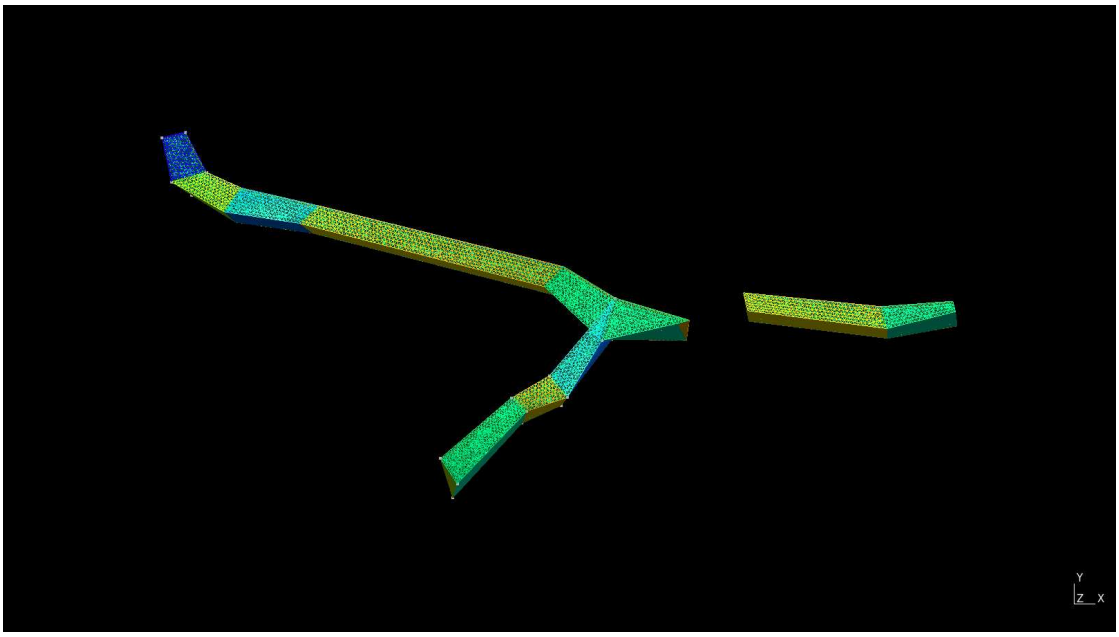

Figure 4.38: Constructing the shape of a cracking pattern at a junction with forced ruled surfaces through each two profiles.

Figure 4.39 shows the final results of the damage elements in the IFC model displayed using usBIM viewer where the cracks voiding features are placed properly in the model, however it was not possible to view shapes as voids due to the limitation of the IFC viewers in general to display complicated free-form geometries properly, not for invalidity

of the shapes created as reported with testing on various IFC viewers for use cases implementing IfcVoidingFeature published by [7]. Figures 4.40 and 4.41 show the the implementation in the IFC model and the assignment of relationship between the damage and the building element containing it (i.e., the slab).

The inspection related information and the calculated attributes from the workflow could be easily added to the properties of the damage shapes as the date and time of inspection of the damage from the EXIF metadata, geometry related attributes as the depth, width, length and orientation, etc. utilsing the IFC entities implemented in [7, 8, 39] as instances of IfcProprtySingleValue added to a attached to an IfcPropertySet related to the cracks damage using IfcRelDefinesByProperties that could be easily reviewed in IFC viewers.

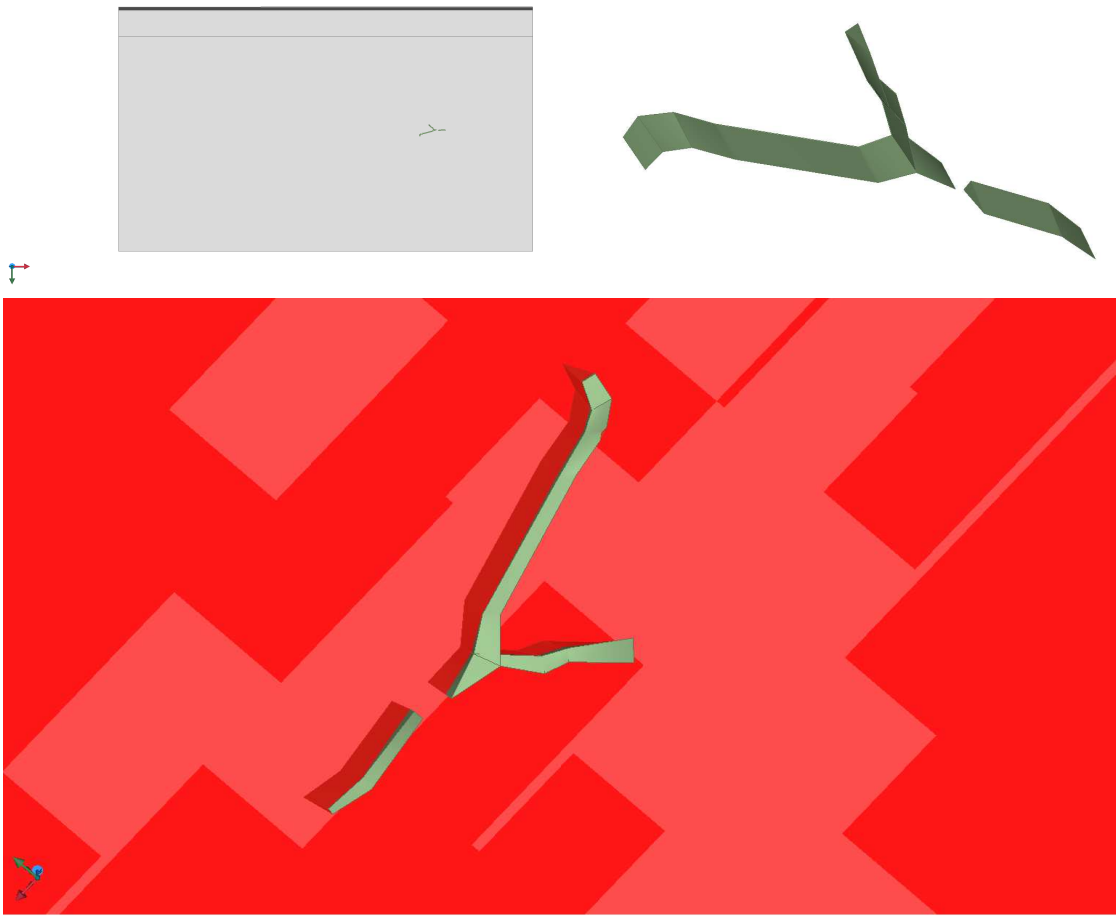

Figure 4.39: Final results of the automated modelling workflow for crack damages displayed in usBIM IFC viewer.

```
26871  #26864=IFCSHAPEREPRESENTATION(#11,'Box','MappedRepresentation',(#26863));
26872  #26865=IFCPRODUCTDEFINITIONSHAPE($,$,(#26857,#26864));
26873  #26866=IFCSLAB('0w$A9QP7L4oeLutI1d2uGq',$,'Pavement',$,$,#26850,#26865,$,.FLOOR.);
26874  #26867=IFCCARTESIANPOINT((1.69076180458069,-7.55711030960083,-0.436388164758682));

74719  #74718=IFCCARTESIANTRANSFORMATIONOPERATOR3D(#74719,#74720,#74721,1.,#74722);
74720  #74719=IFCDIRECTION((1.,0.,0.));
74721  #74720=IFCDIRECTION((0.,1.,0.));
74722  #74721=IFCCARTESIANPOINT((0.,0.,0.));
74723  #74722=IFCDIRECTION((0.,0.,1.));
74724  #74723=IFCSHAPEREPRESENTATION(#65940,'Body','MappedRepresentation',(#74717));
74725  #74724=IFCPRODUCTDEFINITIONSHAPE($,$,(#74723,#74725));
74726  #74725=IFCSHAPEREPRESENTATION(#74726,'Box','MappedRepresentation',(#74731));

74756  #74755=IFCVOIDINGFEATURE('b195417c-e221-4e9e-9938-2cbbc39b692c',#74756,'Void',$,'V
       oiding Element',#65917,#74724,$,.CUTOUT.);
74757  #74756=IFCOWNERHISTORY(#74758,#74759,$,.ADDED.,1609914675,$,$,0);
74758  #74757=IFCPERSON($,'user','',$,$,$,$,$);
74759  #74758=IFCPERSONANDORGANIZATION(#74757,#26897,$);
74760  #74759=IFCAPPLICATION(#26897,'Unspecified','Unspecified',$);
74761  #74760=IFCRELAGGREGATES('12MGLeJonCaOAvtx1YGX6C',#74756,$,$,#26866,(#74755));
74762  #74761=IFCRELVOIDSELEMENT('0nYPUYfCb5RgmMBPNiJ2AP',#74756,$,$,#26866,#74755);
```

Figure 4.40: Excerpt of the IFC model showing the attributes used to model the cracks damage.

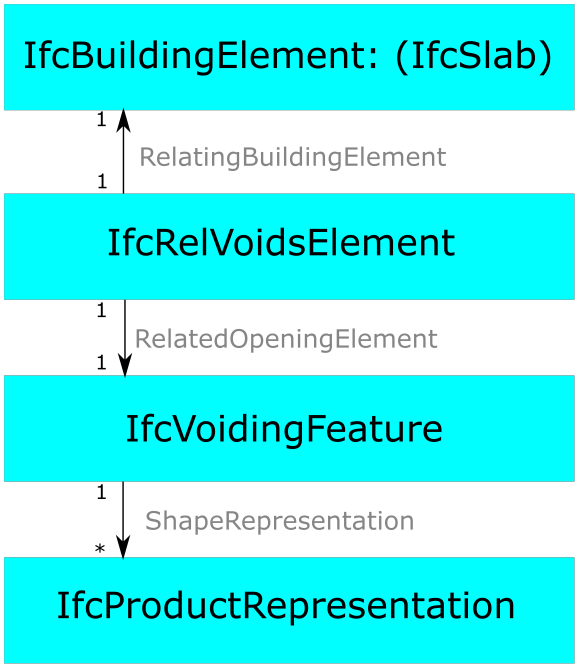

Figure 4.41: IfcVoidingFeature used to model the cracks damage geometrically and its relationship assignment based on the published use case by Artus and Koch [7].

5 Evaluation of Results

In this chapter, the steps taken to verify the output of the software, whenever possible, are presented, the execution time of some methods is shown, as well as the measures to validate the resulting model of the cracks' shapes.

5.1 Camera Calibration

For the use case presented, the intrinsic parameters of the camera were retrieved. This can be done either by a script or using the GUI of the camera calibrator app. The resulting errors for the estimated intrinsic parameters and the mean projection error of the calibration process are presented in Table 5.1.

5.2 3D Reconstruction via OpenSfM

Most of the default configuration settings provided in the package were used. However, the option to match images by distance retrieved from geolocation metadata was disabled for lack of accurate geolocation data that could cause the final reconstruction to split into several parts. The depth map downsize resolution was set to 1080 pixels, and the minimum number of views required to reconstruct a point for it to be valid was raised to 4. The execution time relatively increased when threading was set to the maximum number available (i.e., 8) for the computer used for running it as shown in Table 5.2. The JSON reports on the execution of the reconstruction are provided in Appendix A.2.

Parameter	Value
Mean Reprojection Error	1.8479
Focal Length Error	[14.4241, 13.4194]
Principal Point Error	[2.5008, 3.5934]
Radial Distortion Error	[0.0053, 0.0197]

Table 5.1: Estimated errors for the camera calibration of a perspective model with 2 radial distortion coefficients.

Command	Execution Time (seconds)
extract_metadata	12.4892
detect_features	48.5597
match_features	247.3549
create_tracks	10.3639
reconstruct	343.2193
mesh	27.0066
undistort	53.3742
compute_depthmaps	901.5655
Total	1643.9337

Table 5.2: Execution time taken for all commands of the OpenSfM 3D reconstruction.

5.3 The Retrained TernausNet16 Model

Though the retrained CNN model was capable of segmenting the images taken for the presented use case well, as well as on cracking patterns in soil that were never added into its training sets when tested, it was observed that the CNN was not robust enough in segmenting images with challenging high contrasting shadows as shown in Figure 5.1, as there were only very few training samples of such patterns. Something that is quite common to occur when taking images on site under normal circumstances. Thus, retraining the network on more images in different lighting conditions included in its training set should be considered for future work.

Figure 5.1: Resulting segmentation map by inference from the retrained TernausNet overlaid on test images.

The performance of the retrained TernausNet in comparison with the previously mentioned models in Section 4.3 is illustrated in Table 5.3, where evaluation is conducted on the challenging datasets CRKWH100 and CrackLS315 from Zou et al. [113]. The generated prediction maps from all three models and the evaluation log files are provided in Appendix A.2.

Model	Metric	Mean	Std.
Alabassy (fold_4)	Dice	0.4261	0.1640
	Jaccard	0.2837	0.1285
H. K. Ha	Dice	0.1430	0.0607
	Jaccard	0.0782	0.0356
DeepCrack (Liu et al.)	Dice	0.0410	0.0884
	Jaccard	0.0242	0.074

Table 5.3: The IoU (i.e., Jaccard index) and F1 score (i.e., Dice coefficient) values of the CNNs evaluated on the CRKWH100 and CrackLS315 datasets from Zou et al. [113].

Measured Widths	1	2	3	4	5	6	7	8	9	10
In Model	9.17	4.22	5.29	8.32	5.30	5.04	6.86	4.99	4.52	2.11
On Site	8.0	5.0	6.0	9.0	6.0	7.0	6.0	6.0	6.0	4.0
Rel. Error	11.43%									

Table 5.4: Estimated relative error in average width measurements from the modelled crack shapes to the actual width values on site.

5.4 Validating Estimated Width Measurements

In addition to the measurements taken for scaling the 3D reconstruction due to lack of known homologous 3D points and their related pixel location in images, some measurements along three different segments were taken to validate their real width values to the estimation based on the 2 backwards projected endpoints of the width measurement from the binary segmentation map. As shown in Figure 5.2, the accumulation of errors from the reconstructed point cloud, the backwards projection, its decimation and the global registration resulted in a slight protrusion from the surface of the pavement estimated from the highest protruded point at around 5.803 mm (i.e., approximately 25.2% of the average depth value used for the modelled cracks' profiles). A total of 10 measurements for the widths of this cracks' region were taken on site roughly in similar positions to those measured from the modelled shapes as shown in Figure 5.3 to estimate an average error for the width as detailed in Table 5.4.

5.5 Quality of Meshing

Using the AnalyseMeshQuality plugin in Gmsh, some quality metrics of the meshed model could be measured by calculating the following parameters:

- Gamma: is the elementary aspect ratio = inscribed radius / circumscribed radius.

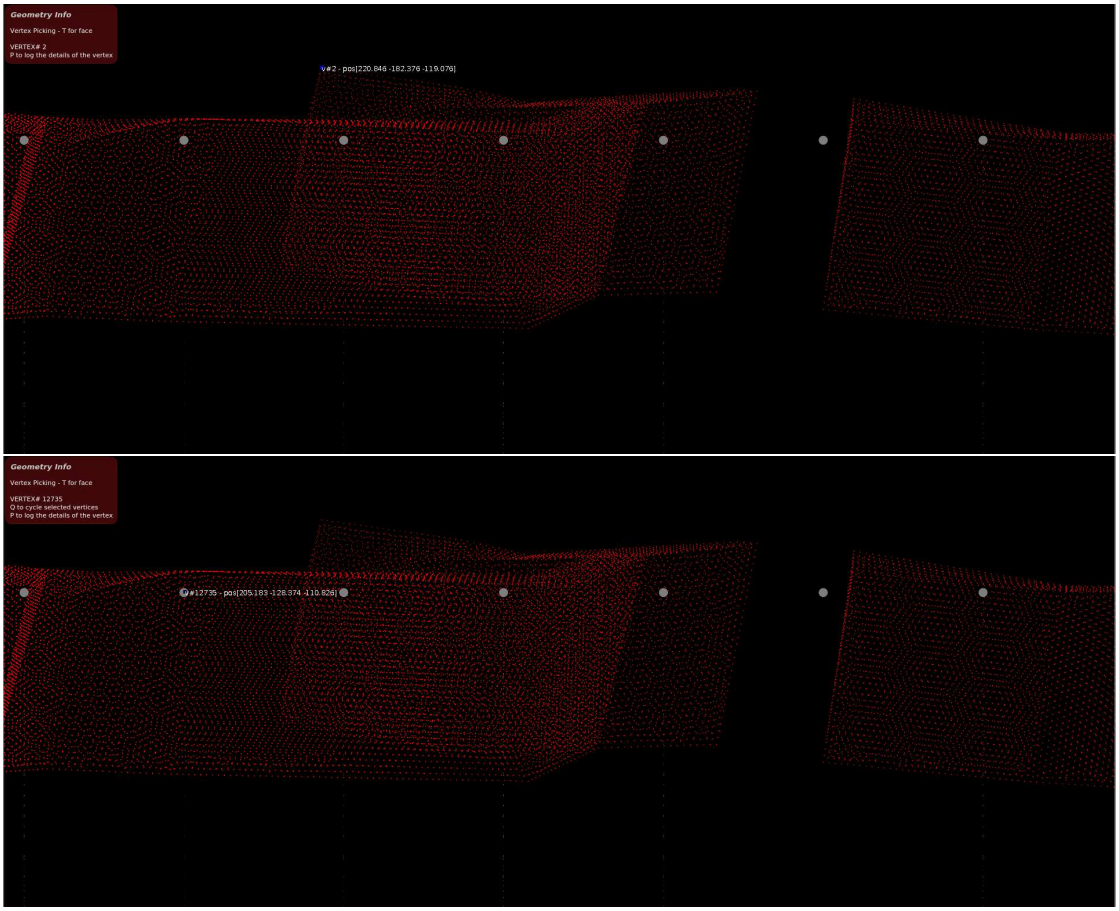

Figure 5.2: Measuring the misalignment of the modelled cracks' shapes between the highest protruded point in red and the pavement surface in grey that results in a slight protrusion from the surface of the pavement.

- Signed Inverse Gradient Error (SIGE): is the signed inverse error on the gradient of the finite element solution.
- Signed Inverse Condition Number (SICN): is related to the condition number of the stiffness matrix explained in [44, 48].

Figure 5.4 shows the parameters in the statistics window on the upper right of each sub-figure for each segment of the four segments modelled for the use case lying within the default range for validity.

5.6 Point Cloud Registration via GoICP

Generally, it was found that for N points in source point clouds, a higher trimming factor of around 0.003xN and a lower convergence threshold of 0.0001xN for the L2-error were sufficient as explained in Equation (2.13). However, this comes at the cost of a longer execution time to get a satisfactory transformation result. For 35,788 vertices in the

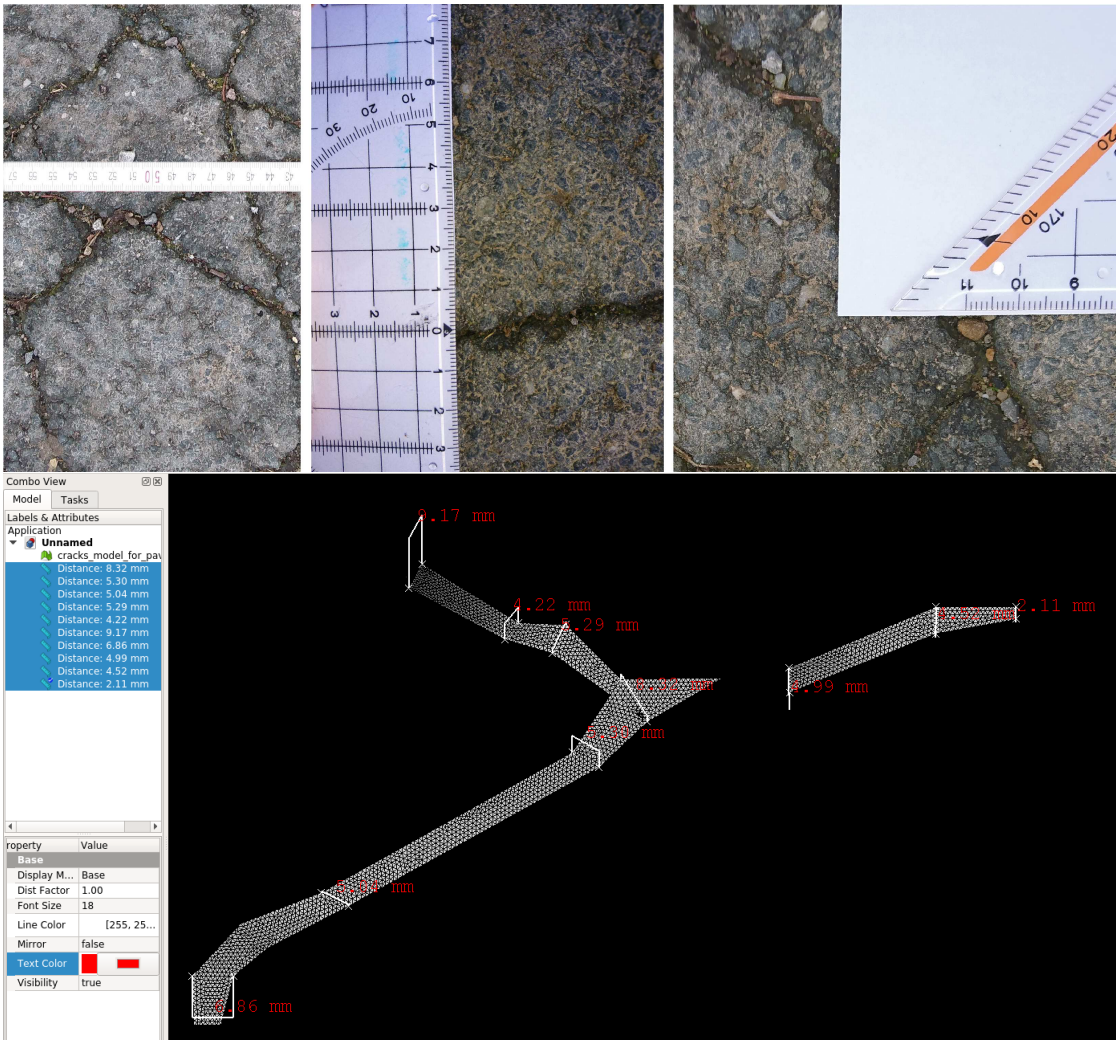

Figure 5.3: Taking measurements for cracks' widths on site and similarly from the modelled crack shapes.

source point cloud of the decimated mesh generated from the OpenSfM 3D reconstruction, and 360,452 vertices in the target point cloud (i.e., the model), the global registration algorithm calculated an initial L2-error of 22090.7 and a final error of 2.53039 with a total execution time of 813.0583 seconds (i.e., 14 minutes approximately). This time is calculated excluding the execution time taken for the preprocessing steps of converting the IFC finely meshed model into .obj (i.e., 54 minutes 46 seconds), voxelisation (i.e., 2 minutes approximately), and normalisation (i.e., 73.3549 seconds) respectively.

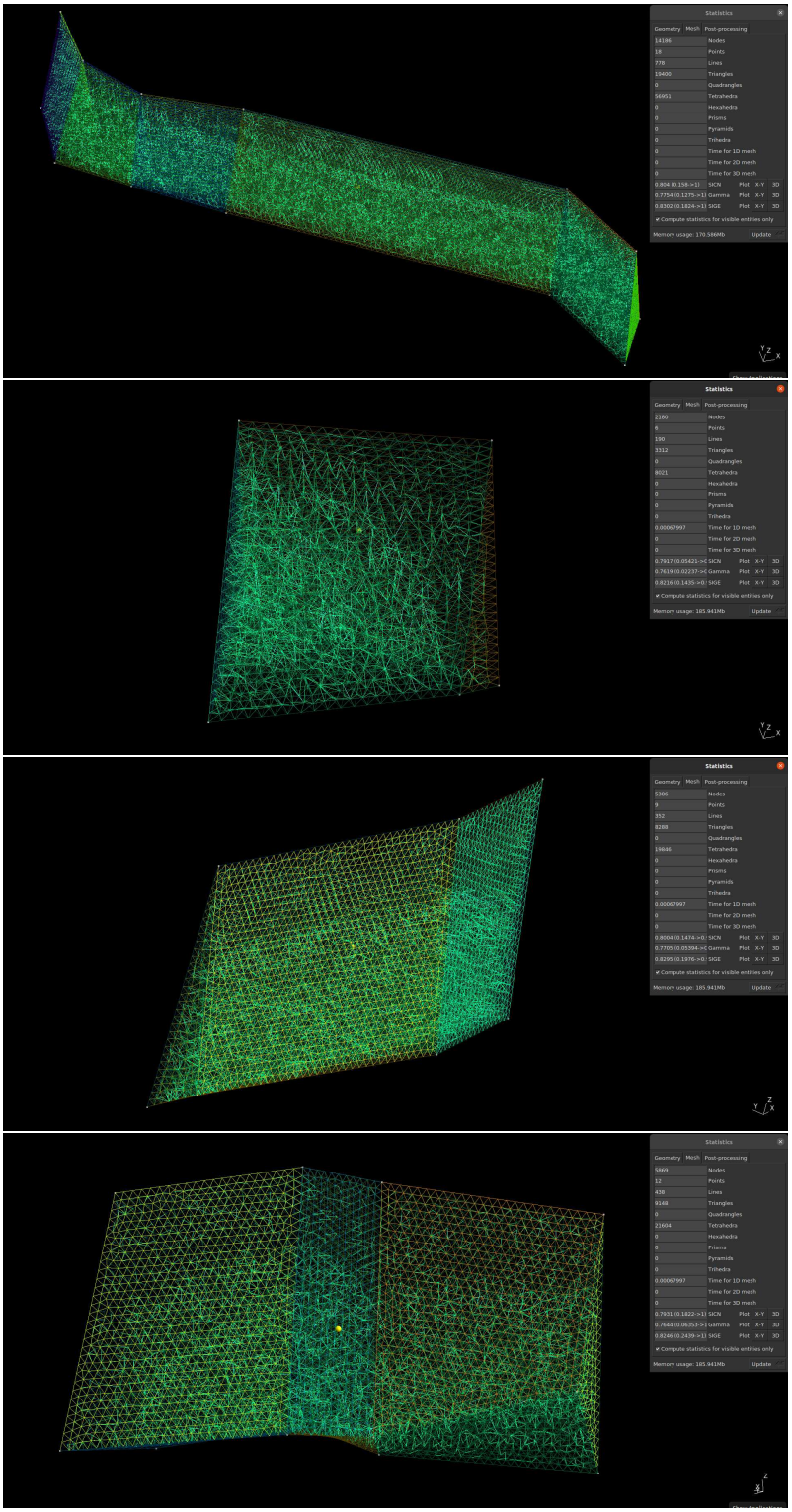

Figure 5.4: Gamma, SIGE, and SICN quality metrics of the meshed models retrieved from Gmsh are shown lying within their valid intervals respectively.

5.7 IfcVoidingFeature vs. Blender's Boolean Difference Modifier

While the the modelling attempt in the IFC model using IfcVoidingFeature could not be successfully viewed in any of the IFC viewers available, except for usBIM as presented in Figure 4.39, where only the shape was visible but not displayed properly as a cutout void in the IfcSlab. Another modelling attempt in Blender was made on the imported IFC model to prove the validity of the modelled crack shapes for boolean difference operation, which turned out to be successful as shown in Figure 5.5.

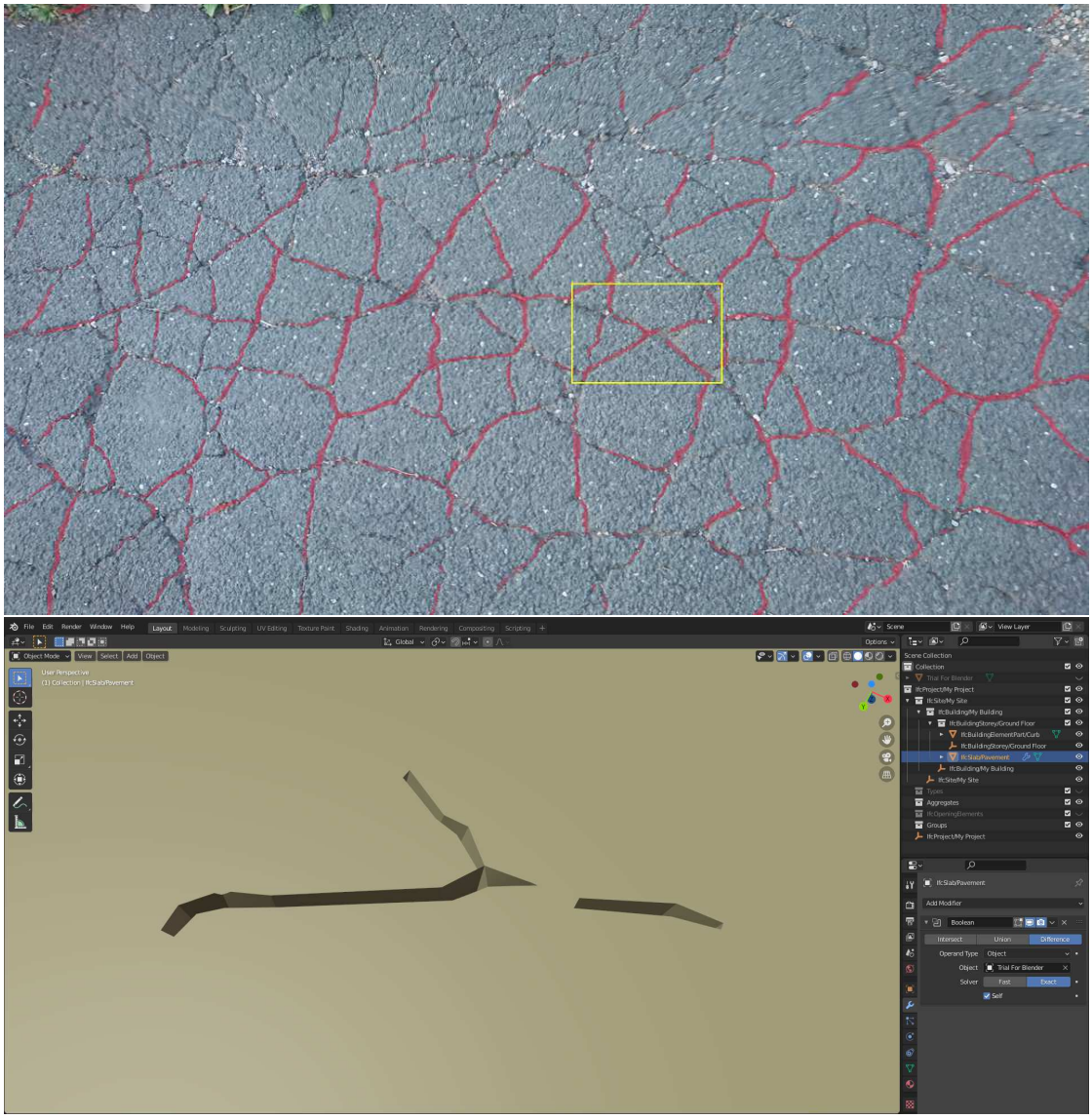

Figure 5.5: A window marking the location of the use case modelled in the candidate image of the pavement in the upper image, and the crack voids modelled in the imported IFC model with Blender using a boolean difference modifier.

6 Summary and Concluding Remarks

6.1 Summary

This thesis proposed a comprehensive approach to reliably model valid shapes for cracking patterns automatically based on images, which are nowadays usually acquired in standard inspection procedures, through a workflow comprising of 8 major steps starting from data acquisition to visualisation at the end. Classical methods for camera calibration are proven to be effective and are normally required once per each configuration of the camera and could be improved in future work with an autocalibration approach. The 3D reconstruction of the point cloud based on the taken images helped in using it as a reference to extract the position of only the main points of particular interest for the cracks' centre-polylines, which proved useful to model the crack shapes independent of any segmentation process applied on the point cloud as a prerequisite.

Though OpenSfM was found very robust, not all options available within the library could be successfully utilised. However, with the working functionalities it proved to be the best suited SFM library for the task. The use of a retrained TernausNet with a VGG16 encoder allowed for a reliable identification of the crack pixels with good accuracy in most conditions, which facilitated the application of several classical image analysis algorithms including, skeletonisation, morphological operations, edge detection, connected components labelling, line simplification and the euclidean distance transform to extract the main points comprising the centre-polylines of each segment in any interconnected network of cracking patterns that should work seamlessly on curved surfaces and on angular edges. With the knowledge of those points of interest and all parameters of the perspective projection model, the 2D pixels could be back-projected to the 3D space and used to model the crack shapes after transforming their coordinates to align with the parent building element containing the cracks (i.e., composition relation) via extrusion.

Such crack shapes could be then converted into various formats depending on the use required, whether be it only for visualisation in IFC format, or performing more advanced FEM based structural analysis procedures, as they preinclude the nodes for tetrahedrons, wires defining all trimmed surfaces, as well as the standard vertices, edges, and faces of a triangulated mesh, thanks to the use of Gmsh library. Both the quality metrics of the modelled shapes provided in Gmsh and the proper creation of voids in Blender prove the shapes are valid and the visualisation limitations in IFC viewers is mostly related to the geometry kernel's inability to properly model more complicated IFC products. The use of XbimToolkit allows for embedding the crack shapes into the IFC model as voiding features with the help of a simple C# script, and facilitates the inclusion of other related

inspection information in a structured and organised manner within IFC. Thus, fulfilling the main goal of the thesis to automate the repetitive slow procedure of recording and archiving damage states for structures and storing them digitally for future use.

6.2 Technical Challenges and Observed Shortcomings

Among some identified points of difficulties in the conducted research at hand, the depth estimation for open surface cracks was presumed as a constant of twice the width, yet it remains a topic of great interest. To our knowledge, there's no direct way to accurately determine the depth based solely on images in any of the trained neural network models published in literature or publicly available under any open source license, and even those published and available can not be reliably considered without due investigation as they lack thorough peer reviews.

Furthermore, there's no image based method that exactly determines the profile shape of the crack even though cracks caused by fatigue are presumed to be of elliptic or semi-elliptic shape in numerical models [76]. The triangular profile was chosen merely to simplify the complexity of meshing required when creating the shapes via Gmsh, but various profiles would be included in future improvements to the code.

The sole reliance on images for the whole workflow makes it susceptible to disruption and failure in inspections undertaken at bad lighting conditions, weather conditions, and highly dependable on the quality of the camera sensors used. A mix of RGBD cameras that often have a limited Field of View (FoV) and short range for depth detection and cameras capable of taking only RGB images of wider FoV might be helpful for such cases.

All trials to utilise a Brown-Conrady camera model in OpenSfM failed at the incremental reconstruction step with no apparent reason as the perspective model with just two radial distortion parameters performed well. The cause of the error could not be identified during the course of the research conducted and the issue would be later investigated and fixed if needed by debugging the OpenSfM package to improve the quality of the 3D reconstruction in future related work.

The use of the EDT proved to cause more problems than actually solving them; hence the switch to classical vector geometry calculations to determine the endpoints of width measurement at any polyline vertex in order to generate valid ruled surfaces in Gmsh as in the second modelled use case. Yet it remains an efficient option when only a rough estimate of widths is required.

It was not clear how crack segments should be modelled at junctions as each segment inevitably intersects with the neighbouring ones as shown in Figure 6.1, when extruded to the same profile orientation at the junction. No reference in literature could be found to rely on nor any guidance from numerical methods for simulating cracks' propagation to properly model this case, as such realistic models were so far not considered feasible to produce valid surfaces, nodes and tetrahadrons for Extended Finite Element Method (XFEM).

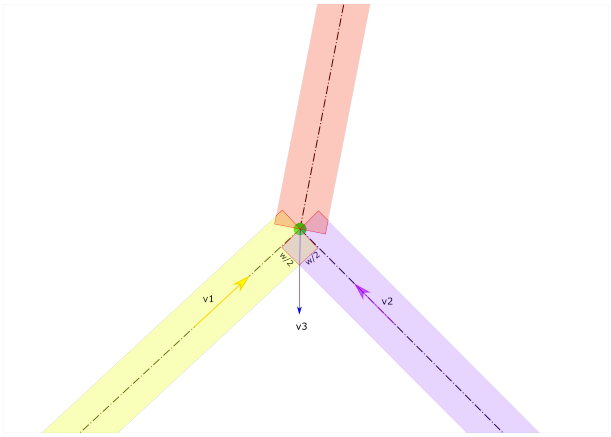

Figure 6.1: Displayed overlapping regions of the crack shapes at junctions bordered in red, which were left on purpose in the shapes construction algorithm and handled only with a boolean union before exporting the whole pattern, till a proper way of modelling that satisfies fracture mechanics conditions is identified.

The consistent direction of the normals was cumbersome to correctly determine, especially in cases of flat surfaces, which are still prone to failure in the implemented code when the source point cloud is very noisy and need to be further refined to make sure it always points in the direction of the camera capturing the scene in all modelling scenarios.

The requirement of high processing power and relatively long computing time for the whole workflow to execute from start to end are not missed from the author's side, yet they are definitely less demanding and faster than that required for a human to manually model such free form damage patterns and place them accurately in a 3D model. However, there is a considerable room for enhancing its efficiency and speed, as the main goal during the workflow's development was getting it to work successfully, with little regard for shortening modelling time, optimising memory usage, and reducing graphics and processing power requirements, which should be tackled in future work.

6.3 Suggestions for Further Improvements and Future Work

While the conducted research has proven it is possible to automate the process of modelling damage shapes geometrically based only on images, it is far from being reliable and faces some obvious technical limitations.

- The reconstruction of dense point clouds using OpenSfM library requires further improvements to allow the use of the more sophisticated Brown-Conrady camera model with 3 radial distortion parameters and 2 tangential distortion parameters, instead of just a perspective camera, could help improve the quality of the constructed point cloud. Yet during the time spent on the implementation and testing on

use cases, all trials to utilize it were unsuccessful probably due to some technical problems with the library itself, not the implementation, as both the fish-eye and perspective camera models were successfully operational.

- The integration of geolocated control points could be useful in providing a good initial alignment for the point cloud that could be used as a reasonable initial guess for the ICP to refine for a final transformation that optimally align with the building elements in the 3D model.
- The failure observed with the retrained TernausNet model in segmenting images with shadows would be mitigated in future work by including more images in the training set with high contrast shadows.
- modelling of other damage types of a voulme loss or open surface volumetric subtraction geometrically like spallings and potholes is a topic of great interest. Furthermore the use of the IfcDiff the IfcOpenShell module in Python to compare various states of such damages is of future consideration.
- The feature subtraction for cracks shapes was performed with the prior knowledge of the sole existence of one building element containing the damage in the IFC model. However, in real case scenarios with a full-size model, a possibility to detect the closest building element by means of nearest neighbour algorithm is worth considering and will be addressed in future improvements.

6.4 Conclusion

The goal of the thesis at hand is to automate the reconstruction of damage shapes based on image sequences, with primary focus on open surface cracks and spalls. Based on the results presented in the use cases for modelling cracks, it has proved it is feasible to automate the process of BIM of damages geometrically, but with some technical limitations that could be overcome with future improvements to the current workflow. Given the review of the current state of research, the conducted research further contributes to the previous work of modelling damages within IFC [7, 8, 39, 42], yet the vast majority of implementations and proposals so far suffice only with triangulating a dense point cloud for damaged elements in a Scan Vs. BIM or Scan to BIM approach. Recent advancements in Large point clouds segmentation [77] accompanied by semantic as-built modelling approaches on the segmentations might might provide a feasible solution to the BIM of as-built models derived from Laser scans [88]. Some previously published applications for instance are to superimpose a dense reconstruction on a 3D model to check for construction progress [50] or errors [78], identify certain building elements such as columns [109, 110] and create BIM models for heritage sites from laser scanning [5], or perform a mesh analysis for damage detection [107]. A drawback of the latter is low performance at regions of subtle surface roughness and global shape variation (e.g., on wide and smoothly curved surfaces). Such applications went no further with constructing the 3D shapes of damages automatically to be embedded into their original location in a

3D building information model, even though the same tools and techniques utilised are fit for automating the workflow to that end as well, with the sole exception of the proposed implementation published by [42] for modelling spall shapes.

Crack detection and segmentation is possible with the inference from a retrained CNN model so as to determine the critical points comprising the polylines of cracks. Using and image sequence enables the creation of a dense point cloud that is necessary for determining a fitting transformation to align to the 3D model. Yet only necessary points of relevance would be eventually inserted into the model (i.e., points comprising the spalling regions excluding rebars and the critical points comprising the polylines of cracks along with their related crack width and depth information).

For crack shapes construction the concept of Desbenoit et al. implementation [21] is well suited for that purpose. Such construction of damage shapes should facilitate attaching all semantic information gathered during inspection directly to its shape and further allow their enhanced visualisation in the 3D model or a VR environment for evaluation and comparison with previous recorded states for deterioration rate assessment. Having the geometric representation of the damages would also open the door for automating the evaluation of the structural safety of the damaged building element required for any condition assessment of structures by improving simulation of cracks propagation and estimate the stiffness reductions using FEM analysis software tools.

With all technical measures taken to minimise the errors of calculations in the presented implementation, the global registration step remains the main contribute to the observe protrusion from the building element surface. The damage shape positioning has to be verified against accumulative imprecise estimations with actual measurements, as false positive feature matches are not filtered out by the RANSAC algorithm during the bundle adjustment calculations or the the ICP algorithm.

To conclude, Laptev et al. [51] reasoned that it is practically not feasible to develop a generic algorithm to extract all properties from digital images, as there will always be human intervention at some point in the automation process and an acceptable level of such intervention should be defined based on accuracy, efficiency and repeatability [19].

Bibliography

[1] I. Abdel-Qader, O. Abudayyeh, M. E. Kelly. “Analysis of Edge-Detection Techniques for Crack Identification in Bridges”. In: *Journal of Computing in Civil Engineering* 17.4 (2003), pp. 255–263. ISSN: 0887-3801. DOI: 10.1061/(ASCE)0887-3801(2003)17:4(255) (cit. on p. 30).

[2] R. S. Adhikari, O. Moselhi, A. Bagchi. “Image-based retrieval of concrete crack properties for bridge inspection”. In: *Automation in Construction* 39 (2014), pp. 180–194. ISSN: 09265805. DOI: 10.1016/j.autcon.2013.06.011 (cit. on p. 34).

[3] S. Y. Alam, A. Loukili, F. Grondin, E. Rozière. “Use of the digital image correlation and acoustic emission technique to study the effect of structural size on cracking of reinforced concrete”. In: *Engineering Fracture Mechanics* 143 (2015), pp. 17–31. ISSN: 00137944. DOI: 10.1016/j.engfracmech.2015.06.038 (cit. on p. 29).

[4] R. Amhaz, S. Chambon, J. Idier, V. Baltazart. “Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.10 (2016), pp. 2718–2729. ISSN: 1524-9050. DOI: 10.1109/TITS.2015.2477675 (cit. on p. 44).

[5] M. Andriasyan, J. Moyano, J. E. Nieto-Julián, D. Antón. “From Point Cloud Data to Building Information Modelling: An Automatic Parametric Workflow for Heritage”. In: *Remote Sensing* 12.7 (2020), p. 1094. DOI: 10.3390/rs12071094 (cit. on p. 84).

[6] S. A. Anwar, M. Z. Abdullah. “Micro-crack detection of multicrystalline solar cells featuring an improved anisotropic diffusion filter and image segmentation technique”. In: *EURASIP Journal on Image and Video Processing* 2014.1 (2014). DOI: 10.1186/1687-5281-2014-15 (cit. on p. 29).

[7] M. Artus, C. Koch, eds. *Modeling Geometry and Semantics of Physical Damages using IFC*. 2020. ISBN: 978-3-7983-3156-3. DOI: 10.14279/depositonce-9977 (cit. on pp. 19, 22, 67, 71, 72, 84).

[8] M. Artus, C. Koch. “Modeling Physical Damages Using the Industry Foundation Classes – A Software Evaluation”. In: *Proceedings of the 18th International Conference on Computing in Civil and Building Engineering*. Ed. by E. Toledo Santos, S. Scheer. Vol. 98. Lecture Notes in Civil Engineering. Cham: Springer International Publishing, 2021, pp. 507–518. ISBN: 978-3-030-51294-1. DOI: 10.1007/978-3-030-51295-8{\textunderscore}36 (cit. on pp. 19, 22, 67, 71, 84).

[9] H. Bay, T. Tuytelaars, L. van Gool. “SURF: Speeded Up Robust Features”. In: *Computer Vision – ECCV 2006*. Ed. by A. Leonardis, H. Bischof, A. Pinz. Vol. 3951. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33832-1. DOI: 10.1007/11744023{\textunderscore}32 (cit. on p. 34).

[10] C. Benz, P. Debus, H. K. Ha, V. Rodehorst. “Crack Segmentation on UAS-based Imagery using Transfer Learning”. In: *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*. IEEE, 2.12.2019 - 04.12.2019, pp. 1–6. ISBN: 978-1-7281-4187-9. DOI: 10.1109/IVCNZ48456.2019.8960998 (cit. on pp. 21, 26, 30, 43).

[11] P. J. Besl, N. D. McKay. “A method for registration of 3-D shapes”. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 14.2 (1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791 (cit. on p. 36).

[12] A. Borrmann, S. Muhic, J. Hyvärinen, T. Chipman, S. Jaud, C. Castaing, C. Dumoulin, T. Liebich, L. Mol. “The IFC-Bridge project – Extending the IFC standard to enable high-quality exchange of bridge information models”. In: *Proceedings of the 2019 European Conference on Computing in Construction*. Computing in Construction. University College Dublin, 2019, pp. 377–386. DOI: 10.35490/EC3.2019.193 (cit. on pp. 21, 22).

[13] G. R. Bradski, A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library / by Gary Bradski and Adrian Kaehler*. Farnham and Cambridge: O’Reilly, 2008. ISBN: 978-0596516130 (cit. on p. 32).

[14] W. S. M. Brooks, D. A. Lamb, S. J. C. Irvine. “IR Reflectance Imaging for Crystalline Si Solar Cell Crack Detection”. In: *IEEE Journal of Photovoltaics* 5.5 (2015), pp. 1271–1275. ISSN: 2156-3381. DOI: 10.1109/JPHOTOV.2015.2438636 (cit. on p. 29).

[15] G. Buffi, P. Manciola, S. Grassi, M. Barberini, A. Gambi. “Survey of the Ridracoli Dam: UAV–based photogrammetry and traditional topographic techniques in the inspection of vertical structures”. In: *Geomatics, Natural Hazards and Risk* 8.2 (2017), pp. 1562–1579. ISSN: 1947-5705. DOI: 10.1080/19475705.2017.1362039 (cit. on p. 20).

[16] L. Bursanescu, F. Blais. “Automated pavement distress data collection and analysis: a 3-D approach”. In: *Proceedings. International Conference on Recent Advances in 3-D Digital Imaging and Modeling (Cat. No.97TB100134)*. IEEE Comput. Soc. Press, 12-15 May 1997, pp. 311–317. ISBN: 0-8186-7943-3. DOI: 10.1109/IM.1997.603881 (cit. on p. 33).

[17] X. Chen, J. E. Michaels, S. J. Lee, T. E. Michaels. “Load-differential imaging for detection and localization of fatigue cracks using Lamb waves”. In: *NDT & E International* 51 (2012), pp. 142–149. ISSN: 09638695. DOI: 10.1016/j.ndteint.2012.05.006 (cit. on p. 29).

[18] Y. Chen, G. Medioni. “Object modeling by registration of multiple range images”. In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 9-11 April 1991, pp. 2724–2729. ISBN: 0-8186-2163-X. DOI: 10.1109/ROBOT.1991.132043 (cit. on p. 36).

[19] P. Dare, H. Hanley, C. Fraser, B. Riedel, W. Niemeier. “An Operational Application of Automatic Feature Extraction: The Measurement of Cracks in Concrete Structures”. In: *The Photogrammetric Record* 17.99 (2002), pp. 453–464. ISSN: 0031-868X. DOI: 10.1111/0031-868X.00198 (cit. on pp. 32, 85).

[20] T. Dawood, Z. Zhu, T. Zayed. “Machine vision-based model for spalling detection and quantification in subway networks”. In: *Automation in Construction* 81 (2017), pp. 149–160. ISSN: 09265805. DOI: 10.1016/j.autcon.2017.06.008 (cit. on p. 31).

[21] B. Desbenoit, E. Galin, S. Akkouche. “Modeling cracks and fractures”. In: *The Visual Computer* 21.8-10 (2005), pp. 717–726. ISSN: 0178-2789. DOI: 10.1007/s00371-005-0317-z (cit. on pp. 19, 37, 85).

[22] S. Dorafshan, R. J. Thomas, M. Maguire. “Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete”. In: *Construction and Building Materials* 186 (2018), pp. 1031–1045. ISSN: 09500618. DOI: 10.1016/j.conbuildmat.2018.08.011 (cit. on p. 30).

[23] C. V. Dung, D. Le Anh. “Autonomous concrete crack detection using deep fully convolutional neural network”. In: *Automation in Construction* 99 (2019), pp. 52–58. ISSN: 09265805. DOI: 10.1016/j.autcon.2018.11.028 (cit. on p. 21).

[24] R. Fan, M. J. Bocus, Y. Zhu, J. Jiao, L. Wang, F. Ma, S. Cheng, M. Liu. “Road Crack Detection Using Deep Convolutional Neural Network and Adaptive Thresholding”. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 9.06.2019 - 12.06.2019, pp. 474–479. ISBN: 978-1-7281-0560-4. DOI: 10.1109/IVS.2019.8814000 (cit. on p. 30).

[25] Fan Yang, Lei Zhang, Sijia Yu, Danil Prokhorov, Xue Mei, Haibin Ling. *Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection*. 2019 (cit. on p. 45).

[26] A. W. Fitzgibbon. “Simultaneous linear estimation of multiple view geometry and lens distortion”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. IEEE Comput. Soc, 8-14 Dec. 2001, pp. I-125-I–132. ISBN: 0-7695-1272-0. DOI: 10.1109/CVPR.2001.990465 (cit. on p. 24).

[27] C. Geuzaine, J.-F. Remacle. “Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities”. In: *International Journal for Numerical Methods in Engineering* 79.11 (2009), pp. 1309–1331. ISSN: 00295981. DOI: 10.1002/nme.2579 (cit. on p. 66).

[28] R. Girshick. “Fast R-CNN”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 7.12.2015 - 13.12.2015, pp. 1440–1448. ISBN: 978-1-4673-8391-2. DOI: 10.1109/ICCV.2015.169 (cit. on p. 31).

[29] J. A. Glud, J. M. Dulieu-Barton, O. T. Thomsen, L. Overgaard. “Automated counting of off-axis tunnelling cracks using digital image processing”. In: *Composites Science and Technology* 125 (2016), pp. 80–89. ISSN: 02663538. DOI: 10.1016/j.compscitech.2016.01.019 (cit. on p. 29).

[30] C. Gunkel, A. Stepper, A. C. Müller, C. H. Müller. “Micro crack detection with Dijkstra’s shortest path algorithm”. In: *Machine Vision and Applications* 23.3 (2012), pp. 589–601. ISSN: 0932-8092. DOI: 10.1007/s00138-011-0324-1 (cit. on p. 29).

[31] Z. Guo, R. W. Hall. “Parallel thinning with two-subiteration algorithms”. In: *Communications of the ACM* 32.3 (1989), pp. 359–373. ISSN: 0001-0782. DOI: 10.1145/62065.62074 (cit. on p. 47).

[32] H. K. Ha. *Crack Segmentation*. 2018. URL: https://github.com/khanhha/crack_segmentation (cit. on pp. 43, 45).

[33] D. Hähnel, S. Thrun, W. Burgard. “An Extension of the ICP Algorithm for Modeling Nonrigid Objects with Mobile Robots”. In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Ed. by Georg Gottlob, Toby Walsh. Morgan Kaufmann, 2003, pp. 915–920. URL: http://ijcai.org/Proceedings/03/Papers/132.pdf (cit. on p. 36).

[34] U. B. Halabe, H.-L. Chen, V. Bhandarkar, Z. Sami. “Detection of Sub-Surface Anomalies in Concrete Bridge Decks using Ground Penetrating Radar”. In: *ACI Materials Journal* 94.5 (1997). ISSN: 0889-325X. DOI: 10.14359/324 (cit. on p. 33).

[35] M. Hamrat, B. Boulekbache, M. Chemrouk, S. Amziane. “Flexural cracking behavior of normal strength, high strength and high strength fiber concrete beams, using Digital Image Correlation technique”. In: *Construction and Building Materials* 106 (2016), pp. 678–692. ISSN: 09500618. DOI: 10.1016/j.conbuildmat.2015.12.166 (cit. on p. 29).

[36] R. I. Hartley. “Euclidean reconstruction from uncalibrated views”. In: *Applications of Invariance in Computer Vision*. Ed. by G. Goos, J. Hartmanis, J. L. Mundy, A. Zisserman, D. Forsyth. Vol. 825. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 235–256. ISBN: 978-3-540-58240-3. DOI: 10.1007/3-540-58240-1{\textunderscore}13 (cit. on p. 35).

[37] R. Heideklang, P. Shokouhi. “Multi-sensor image fusion at signal level for improved near-surface crack detection”. In: *NDT & E International* 71 (2015), pp. 16–22. ISSN: 09638695. DOI: 10.1016/j.ndteint.2014.12.008 (cit. on p. 29).

[38] N.-D. Hoang, Q.-L. Nguyen, X.-L. Tran. “Automatic Detection of Concrete Spalling Using Piecewise Linear Stochastic Gradient Descent Logistic Regression and Image Texture Analysis”. In: *Complexity* 2019 (2019), pp. 1–14. ISSN: 1076-2787. DOI: 10.1155/2019/5910625 (cit. on p. 32).

[39] P. Hüthwohl, I. Brilakis, A. Borrmann, R. Sacks. “Integrating RC Bridge Defect Information into BIM Models”. In: *Journal of Computing in Civil Engineering* 32.3 (2018), p. 04018013. ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000744 (cit. on pp. 19, 22, 71, 84).

[40] V. Iglovikov, A. Shvets. “TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation”. In: *ArXiv* abs/1801.05746 (2018) (cit. on p. 26).

[41] S. Iliopoulos, D. G. Aggelis, L. Pyl, J. Vantomme, P. van Marcke, E. Coppens, L. Areias. “Detection and evaluation of cracks in the concrete buffer of the Belgian Nuclear Waste container using combined NDT techniques”. In: *Construction and Building Materials* 78 (2015), pp. 369–378. ISSN: 09500618. DOI: 10.1016/j.conbuildmat.2014.12.036 (cit. on p. 29).

[42] D. Isailović, V. Stojanovic, M. Trapp, R. Richter, R. Hajdin, J. Döllner. “Bridge damage: Detection, IFC-based semantic enrichment and visualization”. In: *Automation in Construction* 112 (2020), p. 103088. ISSN: 09265805. DOI: 10.1016/j.autcon.2020.103088 (cit. on pp. 19, 20, 22, 84, 85).

[43] G. James, D. Witten, T. Hastie, R. Tibshirani. *An Introduction to Statistical Learning*. Vol. 103. New York, NY: Springer New York, 2013. ISBN: 978-1-4614-7137-0. DOI: 10.1007/978-1-4614-7138-7 (cit. on p. 45).

[44] A. Johnen, C. Geuzaine, T. Toulorge, J.-F. Remacle. “Efficient Computation of the Minimum of Shape Quality Measures on Curvilinear Finite Elements”. In: *Procedia Engineering* 163 (2016), pp. 328–339. ISSN: 18777058. DOI: 10.1016/j.proeng.2016.11.067 (cit. on p. 76).

[45] S. Kabir. “Imaging-based detection of AAR induced map-crack damage in concrete structure”. In: *NDT & E International* 43.6 (2010), pp. 461–469. ISSN: 09638695. DOI: 10.1016/j.ndteint.2010.04.007 (cit. on p. 29).

[46] D. Kang, S. S. Benipal, D. L. Gopal, Y.-J. Cha. “Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning”. In: *Automation in Construction* 118 (2020), p. 103291. ISSN: 09265805. DOI: 10.1016/j.autcon.2020.103291 (cit. on p. 31).

[47] M. Kashif, T. M. Deserno, D. Haak, S. Jonas. “Feature description with SIFT, SURF, BRIEF, BRISK, or FREAK? A general question answered for bone age assessment”. In: *Computers in biology and medicine* 68 (2016), pp. 67–75. DOI: 10.1016/j.compbiomed.2015.11.006 (cit. on p. 34).

[48] P. M. Knupp. “Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II?A framework for volume mesh optimization and the condition number of the Jacobian matrix”. In: *International Journal for Numerical Methods in Engineering* 48.8 (2000), pp. 1165–1185. ISSN: 00295981. DOI: 10.1002/(SICI)1097-0207(20000720)48:8{\textless}1165::AID-NME940{\textgreater}3.0.CO;2-Y (cit. on p. 76).

[49] Kok-Lim Low. “Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration”. In: *Department of Computer Science* University of North Carolina at Chapel Hill (2004) (cit. on p. 36).

[50] C. Kropp, C. Koch, M. König. “Interior construction state recognition with 4D BIM registered image sequences”. In: *Automation in Construction* 86 (2018), pp. 11–32. ISSN: 09265805. DOI: 10.1016/j.autcon.2017.10.027 (cit. on p. 84).

[51] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger, A. Baumgartner. “Automatic extraction of roads from aerial images based on scale space and snakes”. In: *Machine Vision and Applications* 12.1 (2000), pp. 23–31. ISSN: 0932-8092. DOI: 10.1007/s001380050121 (cit. on p. 85).

[52] C. L. Lau, T. Scullion, P. Chan. “Modeling of Ground-Penetrating Radar Wave Propagation in Pavement Systems”. In: *Transportation Research Record: Journal of the Transportation Research Board* 1355 (1992), pp. 99–107. ISSN: 0361-1981. URL: http://onlinepubs.trb.org/Onlinepubs/trr/1992/1355/1355-012.pdf (cit. on p. 33).

[53] Y. LeCun, Y. Bengio, G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539 (cit. on p. 30).

[54] J. Lee, H.-S. Kim, N. Kim, E.-M. Ryu, J.-W. Kang. “Learning to Detect Cracks on Damaged Concrete Surfaces Using Two-Branched Convolutional Neural Network”. In: *Sensors (Basel, Switzerland)* 19.21 (2019). DOI: 10.3390/s19214796 (cit. on p. 21).

[55] T. C. Lee, R. L. Kashyap, C. N. Chu. “Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms”. In: *CVGIP: Graphical Models and Image Processing* 56.6 (1994), pp. 462–478. ISSN: 10499652. DOI: 10.1006/cgip.1994.1042 (cit. on p. 47).

[56] S. Li, X. Zhao. “Image-Based Concrete Crack Detection Using Convolutional Neural Network and Exhaustive Search Technique”. In: *Advances in Civil Engineering* 2019 (2019), pp. 1–12. ISSN: 1687-8086. DOI: 10.1155/2019/6520620 (cit. on p. 21).

[57] X. Li, H. Jiang, G. Yin. “Detection of surface crack defects on ferrite magnetic tile”. In: *NDT & E International* 62 (2014), pp. 6–13. ISSN: 09638695. DOI: 10.1016/j.ndteint.2013.10.006 (cit. on p. 29).

[58] Y. Lin, W.-C. Su. “Use of Stress Waves for Determining the Depth of Surface-Opening Cracks in Concrete Structures”. In: *ACI Materials Journal* 93.5 (1996). ISSN: 0889-325X. DOI: 10.14359/9855 (cit. on p. 33).

[59] Y. Liu, J. Yao, X. Lu, R. Xie, L. Li. “DeepCrack: A deep hierarchical feature learning architecture for crack segmentation”. In: *Neurocomputing* 338 (2019), pp. 139–153. ISSN: 09252312. DOI: 10.1016/j.neucom.2019.01.036 (cit. on pp. 21, 30, 43, 45).

[60] S. Lockley, C. Benghi, M. Černý. “Xbim.Essentials: a library for interoperable building information applications”. In: *The Journal of Open Source Software* 2.20 (2017), p. 473. DOI: 10.21105/joss.00473 (cit. on p. 67).

[61] D. G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/b:visi.0000029664.99615.94 (cit. on p. 34).

[62] J. J. Lu, X. Mei, M. Gunaratne. “Development oF an Automatic Detection System for Measuring Pavement Crack Depth on Florida Roadways”. In: (2002). URL: https://fdotwww.blob.core.windows.net/sitefinity/docs/default-source/research/reports/fdot-bb884-rpt.pdf?sfvrsn=3ec85132_2 (cit. on p. 33).

[63] B. D. Lucas, T. Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI’81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 1981, pp. 674–679 (cit. on p. 34).

[64] M. Eisenbach, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, H. Gross. “How to get pavement distress detection ready for deep learning? A systematic approach”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 2039–2047. DOI: 10.1109/IJCNN.2017.7966101 (cit. on p. 45).

[65] A. Martinet, E. Galin, B. Desbenoit, S. Akkouche. “Procedural modeling of cracks and fractures”. In: *Proceedings Shape Modeling Applications, 2004*. IEEE, 7-9 June 2004, pp. 346–349. ISBN: 0-7695-2075-8. DOI: 10.1109/SMI.2004.1314524 (cit. on pp. 19, 37).

[66] N. Meierhold, M. Spehr, A. Schilling, S. Gumhold, H.-G. Maas. “Automatic Feature Matching between Digital Images and 2D Representations of a 3D Laser Scanner Point Cloud”. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Science* 38-5 (2010), pp. 446–451 (cit. on p. 35).

[67] A. Mohan, S. Poobal. “Crack detection using image processing: A critical review and analysis”. In: *Alexandria Engineering Journal* 57.2 (2018), pp. 787–798. ISSN: 11100168. DOI: 10.1016/j.aej.2017.01.020 (cit. on p. 29).

[68] M. Muja, D. G. Lowe. “Fast Approximate Nearest Neighbours with Automatic Algorithm Configuration”. In: (2009). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.1721&rep=rep1&type=pdf (cit. on p. 34).

[69] S. Mukherjee, B. Condron, S. T. Acton. “Tubularity flow field–a technique for automatic neuron segmentation”. In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 24.1 (2015), pp. 374–389. DOI: 10.1109/TIP.2014.2378052 (cit. on p. 31).

[70] N. Nazaryan, C. Campana, S. Moslehpour, D. Shetty. “Application of a He3Ne infrared laser source for detection of geometrical dimensions of cracks and scratches on finished surfaces of metals”. In: *Optics and Lasers in Engineering* 51.12 (2013), pp. 1360–1367. ISSN: 01438166. DOI: 10.1016/j.optlaseng.2013.05.002 (cit. on p. 29).

[71] F. Ni, J. Zhang, Z. Chen. “Pixel-level crack delineation in images with convolutional feature fusion”. In: *Structural Control and Health Monitoring* 26.1 (2019), e2286. ISSN: 15452255. DOI: 10.1002/stc.2286 (cit. on p. 30).

[72] Ç. F. Özgenel. *Concrete Crack Images for Classification*. 2019. DOI: 10.17632/5y9wdsg2zt.2 (cit. on p. 45).

[73] S. G. Paal, I. Brilakis, R. DesRoches. “Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments”. In: *Advanced Engineering Informatics* 26.4 (2012), pp. 846–858. ISSN: 14740346. DOI: 10.1016/j.aei.2012.06.005 (cit. on p. 31).

[74] L. D. Payne, R. S. Walker. *The Use of Lasers for Pavement Crack Detection*. URL: https://library.ctr.utexas.edu/digitized/texasarchive/phase2/1141-1.pdf (cit. on p. 33).

[75] F. C. Pereira, C. E. Pereira. “Embedded Image Processing Systems for Automatic Recognition of Cracks using UAVs”. In: *IFAC-PapersOnLine* 48.10 (2015), pp. 16–21. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.08.101 (cit. on p. 29).

[76] S. Pommier, A. Gravouil, A. Combescure, N. Moës. *Extended Finite Element Method for Crack Propagation*. Hoboken, NJ USA: John Wiley & Sons, Inc, 2013. ISBN: 9781118622650. DOI: 10.1002/9781118622650 (cit. on p. 82).

[77] F. Poux, C. Mattes, L. Kobbelt. “UNSUPERVISED SEGMENTATION OF INDOOR 3D POINT CLOUD: APPLICATION TO OBJECT-BASED CLASSIFICATION”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIV-4/W1-2020 (2020), pp. 111–118. DOI: 10.5194/isprs-archives-XLIV-4-W1-2020-111-2020 (cit. on p. 84).

[78] D. Rebolj, Z. Pučko, N. Č. Babič, M. Bizjak, D. Mongus. “Point cloud quality requirements for Scan-vs-BIM based automated construction progress monitoring”. In: *Automation in Construction* 84 (2017), pp. 323–334. ISSN: 09265805. DOI: 10.1016/j.autcon.2017.09.021 (cit. on p. 84).

[79] T. Saarenketo, T. Scullion. “Road evaluation with ground penetrating radar”. In: *Journal of Applied Geophysics* 43.2-4 (2000), pp. 119–138. ISSN: 09269851. DOI: 10.1016/S0926-9851(99)00052-X (cit. on p. 33).

[80] M. Sansalone, N. J. Carino. “Impact-Echo Method: Detecting Honeycombing, the Depth of Surface-opening Cracks and Ungrouted Ducts”. In: *Concrete International* 10.4 () (cit. on p. 33).

[81] M. Sansalone, J.-M. Lin, W. B. Streett. “Determining the Depth of Surface-Opening Cracks using Impact-Generated Stress Waves and Time-of-Flight Techniques”. In: *ACI Materials Journal* 95.2 (). ISSN: 0889-325X. DOI: 10.14359/362 (cit. on p. 33).

[82] A. Saxena, M. Sun, A. Y. Ng. “Make3D: learning 3D scene structure from a single still image”. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 31.5 (2009), pp. 824–840. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.132 (cit. on p. 34).

[83] U. Schlengermann. “Determination of Crack Depths using Ultrasonics - An Overlook”. In: *UTonline Application Workshop* 2.05 (May 1997). URL: https://www.ndt.net/article/wsho0597/schleng2/schleng2.htm (cit. on p. 33).

[84] H. M. Shehata, Y. S. Mohamed, M. Abdellatif, T. H. Awad. “Depth estimation of steel cracks using laser and image processing techniques”. In: *Alexandria Engineering Journal* 57.4 (2018), pp. 2713–2718. ISSN: 11100168. DOI: 10.1016/j.aej.2017.10.006 (cit. on p. 34).

[85] Y. Shi, L. Cui, Z. Qi, F. Meng, Z. Chen. “Automatic Road Crack Detection Using Random Structured Forests”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.12 (2016), pp. 3434–3445. ISSN: 1524-9050. DOI: 10.1109/TITS.2016.2552248 (cit. on p. 44).

[86] A. A. Shvets, A. Rakhlin, A. A. Kalinin, V. I. Iglovikov. “Automatic Instrument Segmentation in Robot-Assisted Surgery using Deep Learning”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 17.12.2018 - 20.12.2018, pp. 624–628. ISBN: 978-1-5386-6805-4. DOI: 10.1109/ICMLA.2018.00100 (cit. on pp. 27, 45).

[87] I. Sobel, G. Feldman. *An Isotropic 3x3 Image Gradient Operator*. 2015. DOI: 10.13140/rg.2.1.1912.4965 (cit. on p. 29).

[88] H. Son, C. Kim. “Semantic as-built 3D modeling of structural elements of buildings based on local concavity and convexity”. In: *Advanced Engineering Informatics* 34 (2017), pp. 114–124. ISSN: 14740346. DOI: 10.1016/j.aei.2017.10.001 (cit. on p. 84).

[89] S. Suzuki, K. be. “Topological structural analysis of digitized binary images by border following”. In: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), pp. 32–46. ISSN: 0734189X. DOI: 10.1016/0734-189X(85)90016-7 (cit. on p. 54).

[90] S. Thrun, W. Burgard, D. Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. Cambridge, Mass. and London: MIT, 2005. ISBN: 0262201623 (cit. on p. 30).

[91] S. Ullman. “The interpretation of structure from motion”. In: *Proceedings of the Royal Society of London. Series B, Biological sciences* 203.1153 (1979), pp. 405–426. ISSN: 0950-1193. DOI: 10.1098/rspb.1979.0006 (cit. on p. 34).

[92] S. Ullman. *The Interpretation of Visual Motion*. The MIT Press series in artificial intelligence. Cambridge, Mass.: MIT Press, 1979. ISBN: 9780262257121 (cit. on p. 34).

[93] J. P. de Villiers, F. W. Leuschner, R. Geldenhuys. “Centi-pixel accurate real-time inverse distortion correction”. In: *Optomechatronic Technologies 2008*. Ed. by Y. Otani, Y. Bellouard, J. T. Wen, D. Hodko, Y. Katagiri, S. K. Kassegne, J. Kofman, S. Kaneko, C. A. Perez, D. Coquin, O. Kaynak, Y. Cho, T. Fukuda, J. Yi, F. Janabi-Sharifi. SPIE Proceedings. SPIE, 2008, p. 726611. DOI: 10.1117/12.804771 (cit. on p. 23).

[94] R. S. Walker, R. L. Harris. *Noncontact Pavement Crack Detection System*. Washington D.C: 1991 (cit. on p. 33).

[95] G. A. Washer. “Developments for the non-destructive evaluation of highway bridges in the USA”. In: *NDT & E International* 31.4 (1998), pp. 245–249. ISSN: 09638695. DOI: 10.1016/S0963-8695(98)00009-7 (cit. on p. 33).

[96] J. Wells, B. Lovelace. *Improving the Quality of Bridge Inspections Using Unmanned Aircraft Systems (UAS)*. URL: https://www.dot.state.mn.us/research/reports/2018/201826.pdf (cit. on p. 20).

[97] H. Wu, X. Ao, Z. Chen, C. Liu, Z. Xu, P. Yu. “Concrete Spalling Detection for Metro Tunnel from Point Cloud Based on Roughness Descriptor”. In: *Journal of Sensors* 2019 (2019), pp. 1–12. ISSN: 1687-725X. DOI: 10.1155/2019/8574750 (cit. on p. 32).

[98] Y. Xu, S. Li, D. Zhang, Y. Jin, F. Zhang, N. Li, H. Li. “Identification framework for cracks on a steel structure surface by a restricted Boltzmann machines algorithm based on consumer-grade camera images”. In: *Structural Control and Health Monitoring* 25.2 (2018), e2075. ISSN: 15452255. DOI: 10.1002/stc.2075 (cit. on p. 30).

[99] Y. Morvan. “Acquisition, compression and rendering of depth and texture for multi-view video”. PhD thesis. Department of Electrical Engineering, 2009. DOI: 10.6100/IR641964 (cit. on p. 26).

[100] J. Yang, H. Li, D. Campbell, Y. Jia. “Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration”. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 38.11 (2016), pp. 2241–2254. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2015.2513405 (cit. on pp. 28, 64).

[101] J. Yang, H. Li, Y. Jia. “Go-ICP: Solving 3D Registration Efficiently and Globally Optimally”. In: *2013 IEEE International Conference on Computer Vision*. IEEE, 1.12.2013 - 08.12.2013, pp. 1457–1464. ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.184 (cit. on pp. 28, 64).

[102] S.-N. Yu, J.-H. Jang, C.-S. Han. “Auto inspection system using a mobile robot for detecting concrete cracks in a tunnel”. In: *Automation in Construction* 16.3 (2007), pp. 255–261. ISSN: 09265805. DOI: 10.1016/j.autcon.2006.05.003 (cit. on p. 32).

[103] T. Yu, S. Vinayaka. “Quantification of surface crack depth in concrete panels using 1.6 GHz GPR images”. In: *Nondestructive Characterization and Monitoring of Advanced Materials, Aerospace, Civil Infrastructure, and Transportation IX*. Ed. by P. J. Shull, T.-Y. Yu, A. L. Gyekenyesi, H. F. Wu. SPIE, 27.04.2020 - 08.05.2020, p. 6. ISBN: 9781510635371. DOI: 10.1117/12.2558952. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11380/2558952/Quantification-of-surface-crack-depth-in-concrete-panels-using-16/10.1117/12.2558952.full (cit. on p. 33).

[104] H. Zakeri, F. M. Nejad, A. Fahimifar. “Image Based Techniques for Crack Detection, Classification and Quantification in Asphalt Pavement: A Review”. In: *Archives of Computational Methods in Engineering* 24.4 (2017), pp. 935–977. ISSN: 1134-3060. DOI: 10.1007/s11831-016-9194-z (cit. on p. 29).

[105] L. Zhang, F. Yang, Y. Daniel Zhang, Y. J. Zhu. “Road crack detection using deep convolutional neural network”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 25.09.2016 - 28.09.2016, pp. 3708–3712. ISBN: 978-1-4673-9961-6. DOI: 10.1109/ICIP.2016.7533052 (cit. on pp. 30, 45).

[106] T. Y. Zhang, C. Y. Suen. “A fast parallel algorithm for thinning digital patterns”. In: *Communications of the ACM* 27.3 (1984), pp. 236–239. ISSN: 0001-0782. DOI: 10.1145/357994.358023 (cit. on p. 47).

[107] Y. Zhang, C. Chen, Q. Wu, Q. Lu, S. Zhang, G. Zhang, Y. Yang. “A Kinect-Based Approach for 3D Pavement Surface Reconstruction and Cracking Recognition”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.12 (2018), pp. 3935–3946. ISSN: 1524-9050. DOI: 10.1109/TITS.2018.2791476 (cit. on pp. 33, 36, 84).

[108] Q.-Y. Zhou, J. Park, V. Koltun. *Open3D: A Modern Library for 3D Data Processing*. URL: http://arxiv.org/pdf/1801.09847v1 (cit. on p. 60).

[109] Z. Zhu, I. Brilakis. "Automated Detection of Concrete Columns from Visual Data". In: *Computing in Civil Engineering (2009)*. Ed. by C. H. Caldas, W. J. O'Brien. Reston, VA: American Society of Civil Engineers, 6192009, pp. 135–145. ISBN: 9780784410523. DOI: 10.1061/41052(346)14 (cit. on p. 84).

[110] Z. Zhu, I. Brilakis. "Concrete Column Recognition in Images and Videos". In: *Journal of Computing in Civil Engineering* 24.6 (2010), pp. 478–487. ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000053 (cit. on p. 84).

[111] Z. Zhu, S. German, I. Brilakis. "Visual retrieval of concrete crack properties for automated post-earthquake structural safety evaluation". In: *Automation in Construction* 20.7 (2011), pp. 874–883. ISSN: 09265805. DOI: 10.1016/j.autcon.2011.03.004 (cit. on pp. 32, 53, 54, 56).

[112] Q. Zou, Y. Cao, Q. Li, Q. Mao, S. Wang. "CrackTree: Automatic crack detection from pavement images". In: *Pattern Recognition Letters* 33.3 (2012), pp. 227–238 (cit. on p. 45).

[113] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, S. Wang. "DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* (2018). DOI: 10.1109/TIP.2018.2878966 (cit. on pp. 21, 30, 43, 44, 74, 75).

All links were last followed on December 18, 2020.

A Appendix

A.1 Diagrams explaining the developed Python Projects for the Workflow

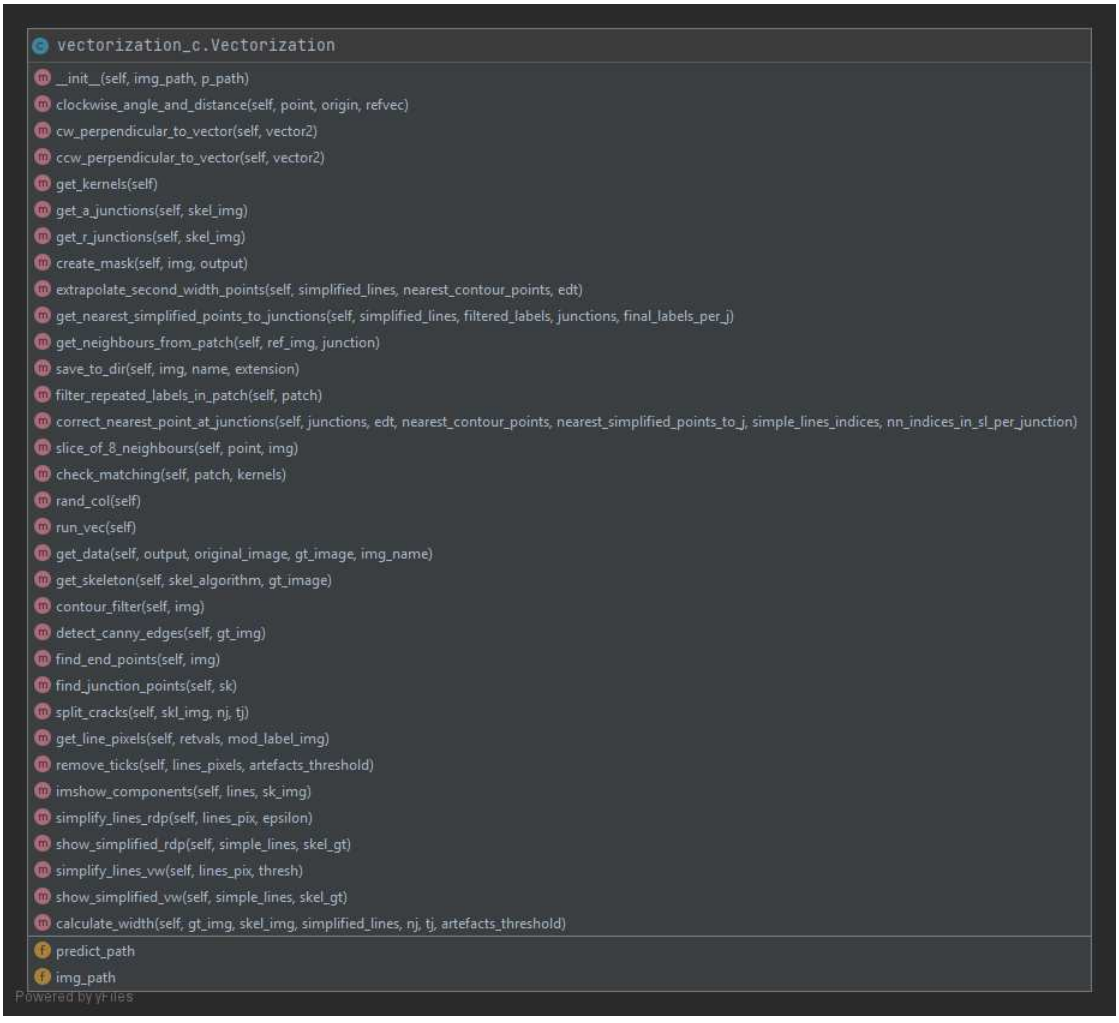

Figure A.1: A class UML diagram for vectorisation.

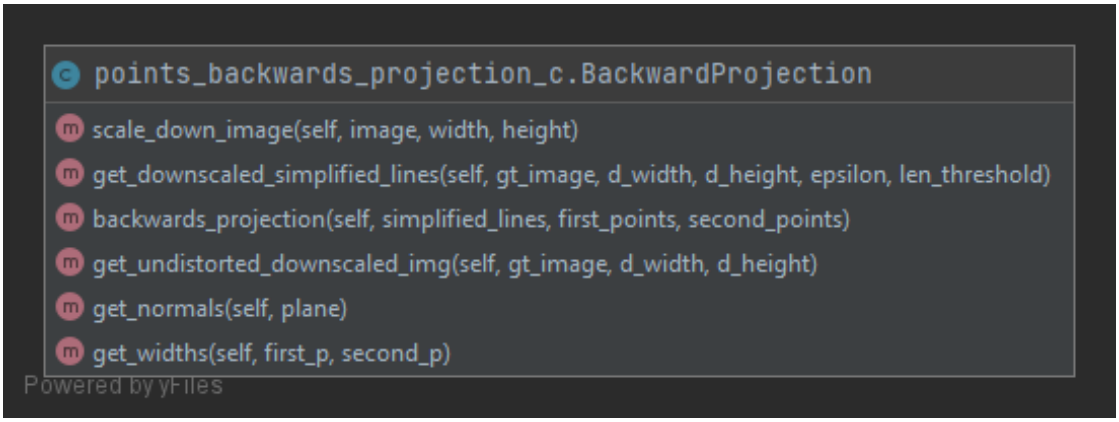

Figure A.2: A class UML diagram for the backwards projection project.

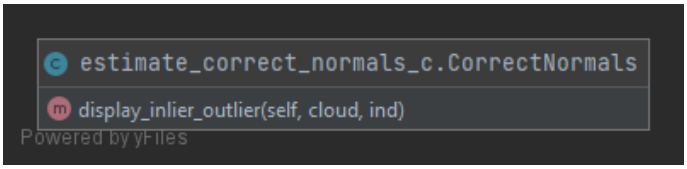

Figure A.3: A class UML diagram for estimating correct normals project.

Figure A.4: A class UML diagram for the ICP project.

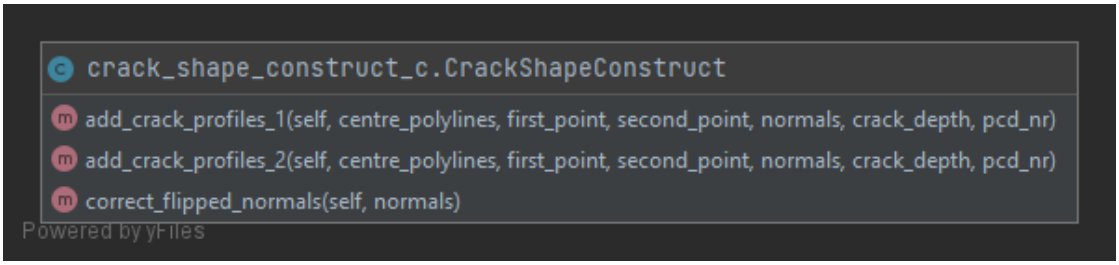

Figure A.5: A class UML diagram for shape construction project.

A.2 Additional Online Material

The relatively large sized content of the appendix that cannot fit for printing is provided in a digital format under the following URL: https://1drv.ms/u/s!AvzPFc2eYo6ThzYe_Q9dKJvBq-ZI?e=ZiqHyw

Figure A.6: A sequence diagram showing the interaction between different classes in Python.