TECHNICAL UNIVERSITY OF CRETE
SCHOOL OF ELECTRICAL & COMPUTER ENGINEERING

Capturing and analysis of facade images of historical buildings to detect defects in order to assess their damage risk

Λήψη και Ανάλυση Εικόνων Προσόψεων Ιστορικών Κτιρίων για την Ανίχνευση Ρωγμών προκειμένου να Εκτιμηθεί ο Κίνδυνος Κατάρρευσής τους

Author:

Koutmos Vasileios

Thesis Committee:

Professor Stavrakakis Georgios (supervisor)

Professor Zervakis Michalis

Professor Stavroulakis Georgios (School of Production Engineering and Management)

A Thesis

To be Submitted to the

School of Electrical and Computer Engineering (ECE) Technical University of Crete, Chania, Crete

ACKNOWLEDGEMENTS

Dr. Georgios Stavrakakis, my supervisor, was instrumental in making this thesis a reality. He was the one who gave me the opportunity to complete my studies, and I will be eternally grateful for his help and support. Furthermore, I am thankful to Dr. Georgios Stavroulakis for the hours he spent assisting my work and generously sharing his knowledge, expertise, and guidance.

I'd be remiss not to mention my entire family, particularly my parents, brother and sisters. Throughout this exhausting process, their faith and trust kept my spirits and motivation high, enabling me to reach the current point in my life. Additionally, special mention deserves my close friends and colleagues on and off the campus for their continues support, patience and companionship throughout this journey and more yet to come.

Ευχαριστίες

Ο Δρ. Γεώργιος Σταυρακάκης, ο επιβλέπων μου, συνέβαλε καθοριστικά στην πραγματοποίηση αυτής της διατριβής. Ήταν αυτός που μου έδωσε την ευκαιρία να ολοκληρώσω τις σπουδές μου και θα είμαι αιώνια ευγνώμων για τη βοήθεια και την υποστήριξή του. Επίσης, είμαι ευγνώμων στον Δρ. Γεώργιο Σταυρουλάκη για τις ώρες που αφιέρωσε βοηθώντας το έργο μου και τις γνώσεις, την τεχνογνωσία και την καθοδήγησή που μοιράστηκε γενναιόδωρα μαζί μου.

Θα ήταν σημαντική παράλειψη να μην αναφέρω ολόκληρη την οικογένειά μου, ιδιαίτερα τους γονείς, τον αδερφό και τις αδερφές μου. Σε όλη αυτή την εξαντλητική διαδικασία, η πίστη και η εμπιστοσύνη τους κράτησαν τη διάθεση και τη θέλησή μου ψηλά, επιτρέποντάς μου να φτάσω στο σημερινό σημείο της ζωής μου. Επιπλέον, αξίζει ιδιαίτερη μνεία στους στενούς μου φίλους και συναδέλφους εντός και εκτός της πανεπιστημιούπολης για τη συνεχή υποστήριξή τους, την υπομονή και τη στήριξή τους σε όλο αυτό το ταξίδι και πολλά άλλα ακόμη.

Abstract

The exterior appearance and structural stability of buildings are negatively impacted by defects in the façades of residential and historical structures. During maintenance, manual labor is usually used to address façade defects in buildings. This method takes a long time, produces arbitrary outcomes, and may end in mishaps or casualties. Ultimately, it is ideal to prevent all types of defects in the design or construction stages, but this is a very difficult goal to achieve. Thus, there is a need for a method to effectively monitor defects in the maintenance phase and actively respond to the occurrence of the defects. Therefore, it is necessary to develop a technology that can continually and automatically monitor defects in residential buildings that minimize the dependence on manpower. Furthermore, there are various types of defects in residential building, and each defect type in the real world appears in an irregular pattern. To consider the characteristics of these defects, automated defect monitoring technology should be able to simultaneously detect and effectively classify various types of defects in image data.

To address this proposal, plenty of methods have been implemented, utilizing different types of deep learning models. The current thesis emphasizes on two different methodologies, which are expanded upon and combined, in order to more efficiently manage defects by minimizing the involvement of manpower.

The dataset used for training a deep-learning-based network contains actual residential and historical building façade images. Faster regions with a convolutional neural network (Faster R-CNN) structure are employed for more accurate defect detection in such environments. As it is difficult to detect defects in a training environment, it is necessary to improve the performance of the network. However, the object detection network employed in this dissertation yields an excellent performance in complex real-world images, indicating the possibility of developing a system that would detect a great number of defects in more types of building façades.

Summarizing the contents of the present thesis, that combines two distinct methodologies, and presents the results from the implementation on real historical buildings. The ultimate purpose of the paper is to expand the use of non-traditional methods in defects in historical building's façades, that depend less on manpower. In the end of the dissertation, the results are going to be evaluated in terms of their accuracy and their efficiency, concluding in some key aspects the field will benefit from the transition to a more automated model.

Περίληψη

Τα ελαττώματα στις προσόψεις κατοικιών και ιστορικών κτιρίων επηρεάζουν τη δομική ακεραιότητα των κτιρίων και υποβαθμίζουν την εξωτερική εμφάνιση. Τα ελαττώματα στην πρόσοψη ενός κτιρίου αντιμετωπίζονται συνήθως με τη χρήση εργατικού δυναμικού κατά τη συντήρηση. Αυτή η προσέγγιση είναι χρονοβόρα, αποφέρει υποκειμενικά αποτελέσματα και μπορεί να οδηγήσει σε ατυχήματα ή ακόμη και θύματα. Έτσι, είναι ιδανική η πρόληψη όλων των τύπων ελαττωμάτων στα στάδια του σχεδιασμού ή της κατασκευής, αλλά αυτός είναι ένας πολύ δύσκολος στόχος να επιτευχθεί. Επομένως, υπάρχει ανάγκη για μια μέθοδο για την αποτελεσματική παρακολούθηση των ελαττωμάτων στη φάση της συντήρησης και την ενεργή δράση στην εμφάνιση των ελαττωμάτων. Ως εκ τούτου, είναι απαραίτητο να αναπτυχθεί μια τεχνολογία που να μπορεί να παρακολουθεί συνεχώς και αυτόματα ελαττώματα σε κτίρια κατοικιών που ελαχιστοποιούν την εξάρτηση από τον ανθρώπινο παράγοντα. Επιπλέον, υπάρχουν διάφοροι τύποι ελαττωμάτων σε κτίρια κατοικιών και κάθε τύπος ελαττώματος στον πραγματικό κόσμο εμφανίζεται με ακανόνιστο μοτίβο.

Για να ληφθούν υπόψη τα χαρακτηριστικά αυτών των ελαττωμάτων, η αυτοματοποιημένη τεχνολογία παρακολούθησης ελαττωμάτων θα πρέπει να είναι σε θέση να εντοπίζει και να ταξινομεί αποτελεσματικά διάφορους τύπους ελαττωμάτων στα δεδομένα εικόνας. Για την αντιμετώπιση αυτής της πρότασης, έχουν εφαρμοστεί πολλές μέθοδοι, χρησιμοποιώντας διαφορετικούς τύπους μοντέλων βαθιάς εκμάθησης. Η παρούσα διατριβή δίνει έμφαση σε δύο διαφορετικές μεθοδολογίες, οι οποίες επεκτείνονται και συνδυάζονται, προκειμένου να διαχειριστούν αποτελεσματικότερα τα ελαττώματα, ελαχιστοποιώντας τη συμμετοχή του ανθρώπινου δυναμικού.

Το σύνολο δεδομένων που χρησιμοποιείται για την εκπαίδευση ενός δικτύου που βασίζεται σε βαθιά εκμάθηση περιέχει πραγματικές εικόνες πρόσοψης κατοικιών και ιστορικών κτιρίων. Ταχύτερες περιοχές με δομή συνελικτικού νευρωνικού δικτύου (Faster R-CNN) χρησιμοποιούνται για πιο ακριβή εντοπισμό ελαττωμάτων σε τέτοια περιβάλλοντα. Καθώς είναι δύσκολο να εντοπιστούν ελαττώματα σε ένα εκπαιδευτικό περιβάλλον για το νευρωνικό δίκτυο, είναι απαραίτητο να βελτιωθεί η απόδοση του δικτύου. Ωστόσο, το δίκτυο ανίχνευσης αντικειμένων που χρησιμοποιείται σε αυτή τη διατριβή αποδίδει εξαιρετική απόδοση σε σύνθετες εικόνες πραγματικού κόσμου, υποδεικνύοντας τη δυνατότητα ανάπτυξης ενός συστήματος που θα ανίχνευε μεγάλο αριθμό ελαττωμάτων σε περισσότερους τύπους προσόψεων κτιρίων.

Συνοψίζοντας, η παρούσα διπλωματική εργασία συνδυάζει δύο διακριτές μεθοδολογίες, και παρουσιάζει τα αποτελέσματα από την εφαρμογή σε πραγματικά ιστορικά κτίρια. Ο απώτερος σκοπός της εργασίας είναι να επεκτείνει τη χρήση μη παραδοσιακών μεθόδων σε ελαττώματα στις προσόψεις των ιστορικών κτιρίων, που εξαρτώνται λιγότερο από το ανθρώπινο δυναμικό. Στο τέλος της διατριβής, τα αποτελέσματα πρόκειται να αξιολογηθούν ως προς την ακρίβεια και την αποτελεσματικότητά τους, καταλήγοντας σε ορισμένες βασικές πτυχές ότι το πεδίο θα επωφεληθεί από τη μετάβαση σε ένα πιο αυτοματοποιημένο μοντέλο.

TABLE OF CONTENTS

LIST OF ACRONYMS

CNN: Convolutional Neural network

DNN: Deep Neural Network

YOLO: You Only Look Once

NMS: Non-Maximum Suppression

DADT: Damage Augmented Digital Twins

MVS: Multiple-View Stereopsis

SFM: Structure from Motion

LOD: Level of Detail

ISO: International Organization for Standardization

RPN: Region Proposal Network

ROI: Region of Interest

IoU: Intersection over Union

FPN: Feature Pyramid Model

ReLU: Rectified linear unit activation function

DFL: Dual Focal Loss function

CIoU: Complete IoU loss

AEC: Architecture, Engineering and Construction industry

SAR: Synthetic Aperture Radar

CHAPTER 1 – INTRODUCTION

1.1 Problem Assessment and State of the Art

Defects in residential and especially historic buildings compromise the structural stability of the structures and diminish their exterior appeal [1]. Building façades, in particular are regarded as significant components of structures as they affect the buildings' beauty, structural safety, and insulation while also acting as an outside barrier against the elements and pollution. However, compared to other building components, aging is accelerated substantially more quickly by prolonged exposure to unfavorable climatic conditions. Eventually, this phenomenon shows up as a variety of defects on the building's structure [2,3].

Building façade defects are frequently repaired with manual labor during maintenance, a method that is associated with various issues, such as the subjectivity of the results arising from human-centered inspection, time consumption and an increase in labor costs. Therefore, a technique to efficiently detect defects in the maintenance phase and take prompt action when they appear is required. It is essential to create technology that can continuously and automatically detect defects in residential structures in order to reduce the reliance on human labor. This technology has to be versatile and accessible, in order to better be incorporated in the field of structural engineering [1, 4, 5].

A variety of trials of new technologies have been implemented by combining drones or robots with imaging devices (e.g., infrared or multispectral camera, laser scanner, and RGB camera), or a combination of these devices [2], to obtain the visual information of the façade.

For instance, a technique known as SAR interferometry, first developed for airborne or spaceborne applications [6], can be usefully exploited in ground-based radar. Typical applications are topographic mapping [7] and more recently monitoring of ground displacements [8]. Preliminary results have shown the method to be promising as a non-invasive and remote sensing technique for structural monitoring of buildings [9]. Additional non-destructive techniques, include infrared thermal imaging [10], ultrasound wave propagation velocity [11], acoustic emission [12], and electrical resistivity [13]. The definition of which test to use depends on the pathology type and the material to be inspected [14]. Some studies have proposed, that the use of UAVs for visual inspections can be an alternative to costly photogrammetric measurements and time-consuming field measurements that historical building inspection needs [15,16,17]. As stated, UAV data acquisition can be successfully used in the energy sector for inventories of power lines and inspecting the state of the energy infrastructure. Utilizing UAVs for inspections of hard-to-reach or hazardous locations plus increases the safety of work [5].

Among these new technologies, RGB image-based visual inspection is the most conveniently adopted choice in the industry and the most widely explored topic in the academia, because it can provide detailed visual information with a satisfactory resolution and high speed [18].

Another recent approach to gathering information about a building's façade and structure presents itself in the form of 3d reconstructions of the building. For instance, structure from motion (SfM) and multiple-view stereopsis (MVS) can generate detailed models as textured meshes to reconstruct 3D scenes after earthquake events [19]. Another more convenient approach due to the size of the resulting files in the SfM, is the approach of models such as level of detail (LOD) models, which contain simplified geometrical information about the asset. Usually, LOD models are generated by post-processing 3D raw in- formation output from laser scanner and/or photogrammetry pipelines [20] and may require some manual intervention [21]. This model can be combined with damage information in the form of a Digital Twin (DT). A DT is a precise digital representation of a physical object, in this case a building, that can contain varying information depending on the application and is linked to the physical entity via data or services. Especially useful have been the geometric digital twins (GDT), which are 3D models that contain detailed information about the geometry of a physical asset. To form a damage-augmented digital twin (DADT), these GDTs are combined with damage information detected using image data [22].

Several studies in recent years have had promising results utilizing DTs to monitor the structural health of buildings. For example, a study presented, to monitor and define structural maintenance, the use of DT in the form of building information models (BIM) [23] and another that used DTs to study the integrity of the structural system of historic masonry buildings and specifically analyze the system response [24]. Another team of authors, Shabani et al. [25], assessed different challenges and strategies for using DT to assess the building response under different load scenarios. For damage assessments after natural disasters and structural maintenance, a key application of DTs, another paper [26] presented ''regular'' post-earthquake assessment procedures and pioneered new technologies such as photogrammetry and laser scanning after the Zagreb and Petrinja earthquakes in 2022. Lastly a DT framework was proposed for post-earthquake building evaluation with unmanned aerial vehicle imagery, component identification, and dam- age evaluation [27],[22]. However, the process of extracting useful information from the collected images and models still remains quite challenging.

A promising approach to this problem presents itself in the form of deep learning techniques. Deep learning techniques are data-driven, rule-free processes. The sole steps in the process of creating a model are choosing an appropriate network structure, a function to evaluate the model output, and an efficient optimization algorithm. Deep learning techniques are driving advances in computer vision to tackle the drawbacks of classical defect detection models that allow the automatic capture of intricate structures of large-scale data with models comprising multiple processing layers. However, this alternative approach poses a new set of setbacks regarding its use in residential and historical building maintenance, and each one shows up in the actual world in an irregular pattern. Automated defect monitoring system should be able

to simultaneously detect and accurately categorize numerous types of faults in picture data in order to take into account the characteristics of these problems [1].

Current studies and have applied various machine and deep learning algorithms to classify, localize, and detect façade defects that will be analyzed in the segments to follow. However various building designs in raw images used, make it difficult to detect defects on their façades because of their various types and complex architecture and backgrounds [22]. Also, many developed models generate predictions that are overlapping with each other, due to the unpredictable nature of the defects, which could cause overestimation of the severity if using the number of bounding boxes to calculate the number of defects. On the other hand, despite the fact that certain research has suggested strategies for defining assessment areas for condition evaluation in accordance with ISO standards, the established evaluation areas are insufficiently effective to deliver appropriate and objective information for the evaluation of façade condition. Many techniques made the assumption that any random area may serve as an evaluation area. So, when a number of defects are concentrated in a restricted area of the evaluation area, this assumption could result in an underestimating of the severity level [3]. In addition to these problems, systems developed for defect detection, need to be able to detect and classify a number of distinct defects that may occur in a building façade, which is rarely a simple task.

The efforts of the present dissertation attend to approach the aforementioned problems utilizing the already developed technologies of deep learning, computer vision and automation. In the segments to follow, current studies will be analyzed, with the goal of understanding the contributions made in the field of automated façade defect detection. Each study is analyzed in each own chapter, with the original papers cited in the title of the individual chapter. Furthermore, their efforts will be combined and expanded upon, with the purpose of producing a complete model that is flexible enough to work with all kinds of infrastructure, in real world conditions.

1.2 Building Façade Defects

As stated above, visual assessments of building damage can be cumbersome, time-consuming, arbitrary, and often challenging to record. The goal of the present thesis is to develop a system that can generate pictorial guides for preliminary visual inspection of Historical buildings with multiple façades, concerning five (5) main defects [28]:

- cracks
- mold
- peeling paint
- stairstep cracks
- and water seepage

- Cracks account for the vast majority of all defects present in building façades, both residential and especially historical. They are easy to form especially with the passage of time weathering the façade materials and structure. Cracks can be classified according to their depth and the area that is afflicted by the defect, requiring different levels of urgency in the measures needed.
- Mold is most common in historical and heritage building façades, that are close to a body of water or in an area prone to frequent rainfalls. Another common reason, is the leakage from pipes or sewage that promotes the growth of fungi and other biological growths.
- Peeling paint occurs when moisture collects under a painted surface. Moisture enters the surface from the unpainted side, gets absorbed and then dries. This repeated swelling and shrinking of wood causes the paint to pull away from the surface, which results in cracking and peeling paint.
- Stairstep cracks are a special kind of crack that form usually on brick, concrete or stone block walls. This defect can be hazardous especially in the case of historical buildings as it compromises the structural integrity of the whole building.
- Lastly water seepage occurs when water is leaking and comes into contact with the wall. This can be cause by a variety of reasons and can also be the first step in the development of other defects, such as mold and peeling paint. Hence the reason it should be counteracted promptly.

These five defects are the defects the present dissertation will focus on detecting as they are some of the most common and also urgent in the maintenance process. It is important to note, that some defects on façade may have similar appearances and are less distinguishable than infrastructure defects, which makes it more difficult to categorize different defects to generate a high-quality dataset. In addition, the diverse occurrence rate of façade defects causes severer imbalance dataset problem than infrastructure defects. This is a potential problem in the dataset creation process that will be addressed in later segments.

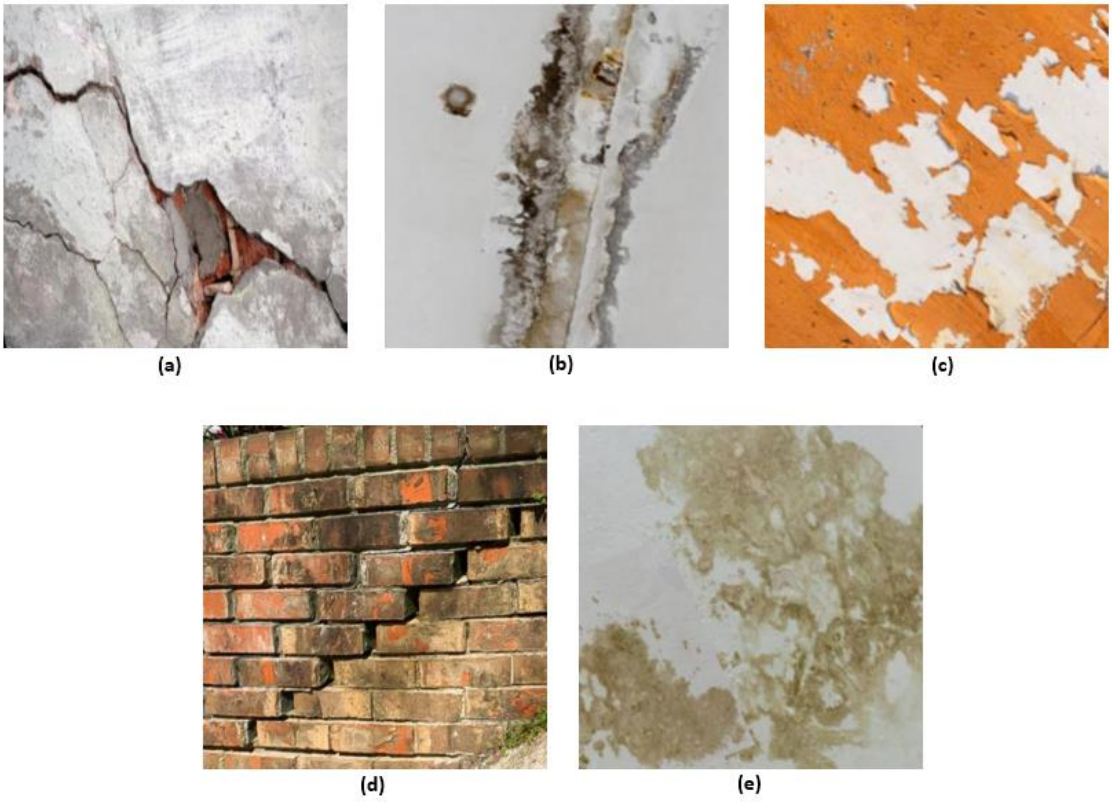


Figure 1. Example of Defects: (a) Crack, (b) Mold, (c) Peeling paint, (d) Staircase Crack, (e) Water seepage

1.3 Deep Machine Learning applied in Building Defect Detection and Characterization

The term "deep learning" has become prevalent in the computer industry. Many real-time applications utilize it, and it is a division of Machine Learning as a whole. To make decisions regarding new data, deep learning needs a large amount of data, which is essential. Data processing is done using neural networks that are categorized as Deep Neural Networks (DNN). The phrase "deep neural networks" has gained popularity since neural networks are frequently utilized in deep learning techniques. Convolutional-Neural Networks (CNNs) are among the most often utilized deep neural networks. It is not necessary for humans to extract features from CNN. The CNN directly extracts the characteristics from a dataset of raw images with related features not pre-learned when networks are train on a batch of images. The most accurate learning model for computer vision tasks such as object identification, categorization, and recognition are this automated feature extraction method. Machine Learning techniques that rely on human feature extraction and a different algorithm to categorize each object have been around for a long time. However, in Deep Learning techniques, the network itself extracts the features without involving the user and classifies the items using several hidden layers [29]. Figure 2 depicts this.

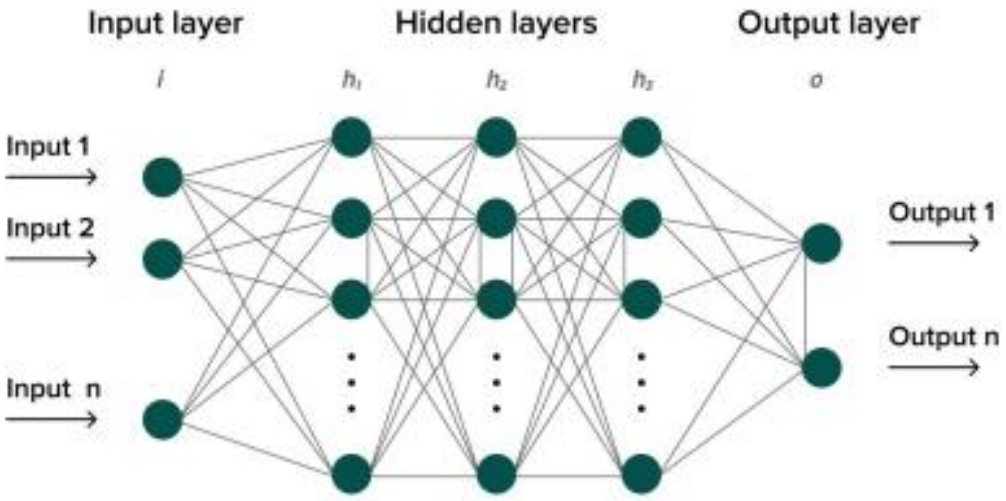


Figure 2: Deep Learning representation

Convolutional neural networks (CNNs) are characterized by their capacity to automatically learn spatial hierarchies of features from raw input data. This property allows CNNs to identify global structures, local patterns, and semantic representations. In most cases, convolutional, pooling, and fully connected layers structure a CNN. Kernels are convolved with the input data in the convolutional layers in order to extract the most important features and create feature maps. The network is capable of handling translations and changes more efficiently due to the pooling layers' assistance in reducing the spatial dimensions of the feature maps. Ultimately, the retrieved features are combined and the classification is carried out by the completely linked layers. To maximize performance, the output of the first layer is being provided as an input of the next layer, which in turn will extract other complex features of the input image like corners and combinations of edges [30].

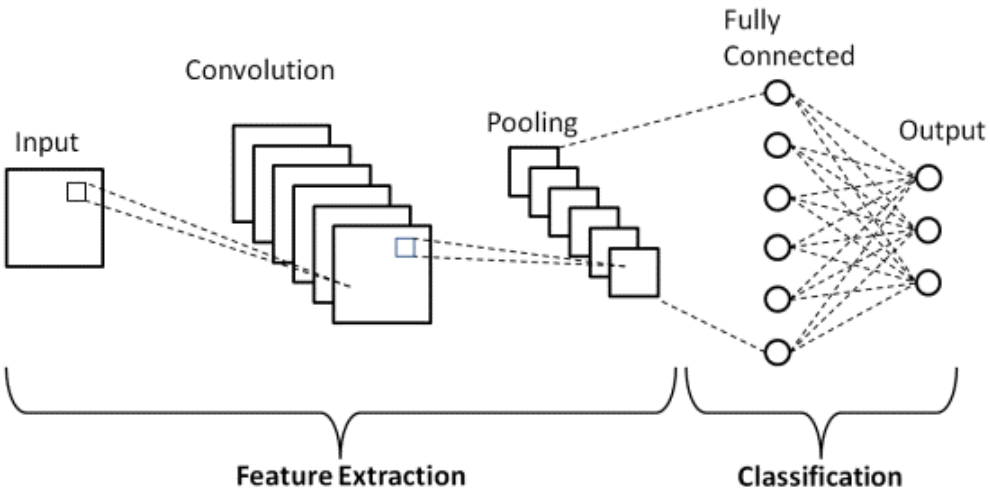


Figure 3. In depth CNN representation

The layers of CNNs are made up of several artificial neurons. Artificial neurons are mathematical functions that are used to calculate the sum of numerous inputs and give output in the form of an activation value. They are analogous to the neuron cells that are utilized by the human brain for conveying various sensory input signals and other responses. The value of each CNN neuron's weights determines how it will behave. The artificial neurons of a CNN can recognize a variety of visual traits and specifications when provided with pixel values. The deeper into the convolutional neural network, the more the layers start detecting various higher-level feature [30].

A great number of studies have incorporated Deep Neural Networks together with image computing techniques, for the purposes of defect detection in historical building façades. The most common deep learning approaches that can be used for detection of objects and defects

from the 2D/3D images in the architecture, engineering and constructing industry are the convolutional neural networks (CNNs)-based image classification and patch-wise segmentation [31] and fully convolutional networks (FCNs)-based pixelwise segmentation [32],[33]. Therefore, a great number of authors have made contributions such as an automated defect detection and classification method from closed-circuit television (CCTV) inspections based on a deep convolutional neural network (DCNN) that takes advantage of the large volume of inspection data [34], a condition-aware model of structures that incorporated a textured 3D building model with defects detected by deep learning models and mapped using UV mapping [35], a vision-based method for concrete crack detection and density evaluation using a deep fully convolutional network (FCN) [37], a transfer learning method based on multiple DCNN knowledge for crack detection [38] and an automated defect detection system with an object detector based on a convolutional neural network (CNN), commonly known as the YOLOV3 network [39]. Shen et al. [40] developed a Fourier-transform-based steel bridge coating defect-detection approach (FT-DEDA) that makes use of the fact that the differences between background pixels are not as large as the differences between defect pixels to detect their existence. However, as image data obtained in the real world are quite diversified, IPT using prior knowledge are limited in recognizing defects in image data [41]. [1]

In general, detectors used in academia are divided into three types: detectors to classify the image into a specific type of defect, detectors to localize the defect using a bounding box, and detectors to segment the defect areas in the image. First, the model for classification is mainly based on various architectures of CNN such as GoogleNet [42], VGG network [43, 44, 4] and ResNet [28] to extract features and achieve image classification directly. These classification models are still inefficient or inaccurate enough for defects localization even though they are commonly integrated with different tools and algorithms. Thus, many studies used more complex models to locate the defects with bounding boxes including one-stage detectors such as the single shot multibox detector (SSD) [45, 46] and two-stage detectors such as Faster R-CNN [3].

1.4 Performance Evaluation Metrics

Some metrics that are broadly applicable across different object detection models, consist of the following [47].

- **Precision and Recall:** Precision quantifies the proportion of true positives (TP) among all positive predictions, assessing the model's capability to avoid false positives (FP). On the other hand, Recall calculates the proportion of true positives (TP) among all actual positives, measuring the model's ability to detect all instances of a class.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

- **F1 Score:** The F1 Score is the harmonic mean of precision and recall, providing a balanced assessment of a model's performance while considering both false positives (FP) and false negatives (FN).

$$F1\ Score = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}$$

- **Confidence:** The confidence scores that YOLOv8 outputs is a combination of two confidence scores, box confidence and class confidence. This enables it to balance between how certain it is that a box contains an object and how certain it is about which class this object belongs to.

The most commonly used metric for evaluating the performance of object detection models is the Average Precision (AP), traditionally called Mean Average Precision (mAP). It measures the average precision across all categories, providing a single value to compare different models. The mAP metric is based on precision-recall metrics, handling multiple object categories, and defining a positive prediction using Intersection over Union (IoU) [48].

Multiple object categories in an image must be located and identified by object detection models. The mAP measure confronts this by computing the average precision (AP) for each category separately and then averaging these APs across all categories. This approach ensures that the model's performance is evaluated for each category individually, providing a more comprehensive assessment of the model's overall performance [48]. By predicting bounding boxes, object detection attempts to locate objects in images with high accuracy. To evaluate the accuracy of the predicted bounding boxes, the AP metric incorporates the Intersection over Union (IoU) measure. IoU is the ratio of the intersection area to the union area of the predicted bounding box and the ground truth bounding box. It measures the overlap between the ground truth and predicted bounding boxes (Fig. 4) [48].

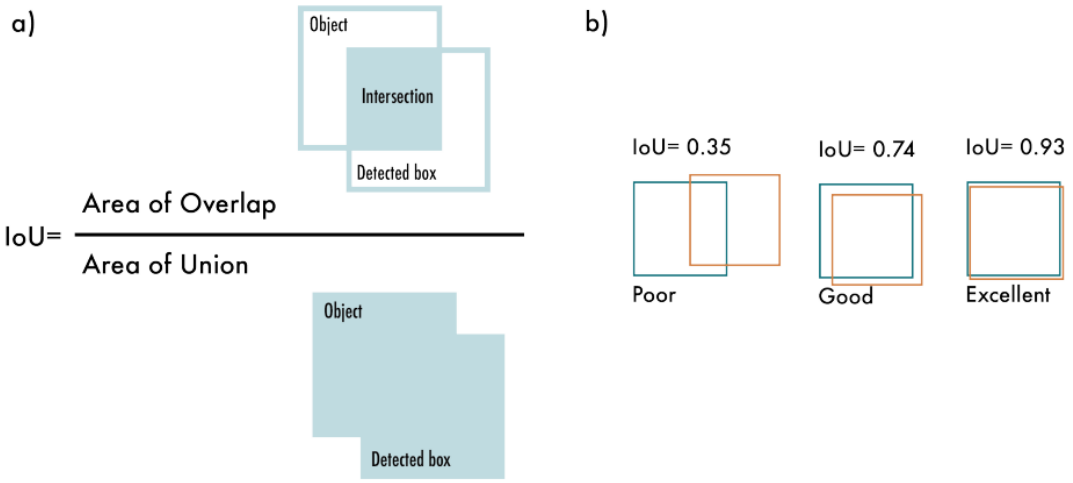


Figure 4: Intersection over Union (IoU). a) The IoU is calculated by dividing the intersection of the two boxes by the union of the boxes b) examples of three different IoU values for different box locations.

CHAPTER 2 – DAMAGE AUGMENTED DIGITAL TWINS (DADT) FOR INSPECTION OF BUILDINGS [22]

2.1 Purpose of the study

The first essential study used for the present thesis proposes an end-to-end pipeline that generates damage-augmented digital twins (DATs) for buildings at LOD3, including geometrical information as well as data pertaining to damage condition and its characterization. As the authors state, current procedures for the rapid inspection of buildings and infrastructure are subjective, time-consuming, and cumbersome to document, necessitating new technologies to automate the process. Though they have not yet been included in infrastructure applications, recent advancements in image technology and artificial intelligence, such as computer vision, provide the required tools for this automation to take place. In their framework, multiple-views images are used to classify damage, segment damage information, and build a level of detail model. The **structure from motion**, which is utilized to recreate the architectural scene, and machine learning models that segment and characterize damage form the basis of the technique.

In addition to this, the study focuses on crack detection and characterization, as cracks are the most commonly observed form of damage during inspection, particularly in brittle structures including concrete or masonry, making it crucial to segment cracks from images. According to the paper, although image-based methods have used digital image correlation methodologies for the characterization of cracks with fairly accurate results, they are restricted by the need for a particular configuration of image devices and sometimes the structure itself.

To overcome this, Pantoja-Rosero et al. proposed a novel method for describing crack kinematics based on 2D point registration, which employed as input binary masks that represent crack patterns obtained through techniques such as deep learning, a methodology valuable for automating damage assessments of real structures.

Computer vision techniques for damage assessment have also been used to automatically generate 3D models. For example, structure from motion (SfM) and multiple-view stereopsis (MVS) can generate detailed models as textured meshes to reconstruct 3D scenes after earthquake events with good accuracy but resulting in big files for a large number of assets. Because they provide streamlined geometrical information about the object, models like level of detail (LOD) models are more practical to employ. In this study, LOD models for buildings are built using CityGML 2.0, a standard for storing and transferring virtual 3D city models. Building models are categorized into five levels of detail in accordance with CityGML 2.0, ranging from LOD0 (the least detailed) to LOD4 (the most detailed), as shown below.

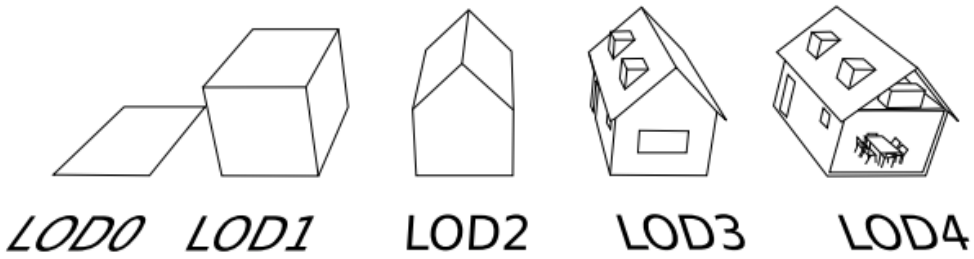


Figure 5: The five LODs defined by CityGML 2.0. The geometric detail and the semantic complexity increase, ending with LOD4, which contains indoor features. Source: Figure taken from Pantoja-Rosero et al.

The objective of the effort is an end-to-end pipeline that automatically creates a rich 3D model with damage information in the form of a digital twin (DT) at LOD3 rather than individually tackling the tasks necessary for a building damage assessment with the goal of partial automation. A DT is an exact digital representation of a real-world object that connects to the real thing through data and/or services and can hold a variety of information. The authors are interested in geometric digital twins (GDT), which are 3D models that contain detailed information about the geometry of a physical asset. To form a damage-augmented digital twin (DADT), these GDTs will be combined with damage information detected using image data, specifically relating to cracks and their characterization via kinematic algorithms to determine their propagation mode (mode I and mode II), useful for planning future interventions and decisions.

Several examples of recent studies had promising results using DTs are given, each indicating the importance of implementing frameworks for the data collected from new technologies to improve damage assessment in terms of objectivity, time, and documentation. Drawbacks of these studies are also listed, such as the generation of very large documentation outputs that are difficult to store, often focusing on a single asset, and rarely serve for rapid damage assessments. So, the proposition is an end-to-end automated pipeline for creating DADT for free-standing buildings using as input multiple-view RGB photographs of the building asset using computer vision and machine learning to address these problems. This approach produces a light-weight DADT model that is suitable for quick inspections, allows the addition of semantic data derived from image data, does not concentrate on a single asset, is flexible enough to be applied to other types of infrastructure.

The hereby proposed methodology requires as input multiple-view images of the building asset suitable for SfM. SfM processes the images, producing camera poses and a point cloud that are used to generate a LOD3 model. A CNN trained to detect cracks processes the images used to generate the 3D model. The cracks are then characterized by computing their kinematics with a least-squares-based 2D registration algorithm (crack displacements in mode I and model II). Finally, using the SfM information, the damage information is mapped to the LOD3 model to generate the desired DADT output, that includes a 3D reconstructed geometry of a building asset with cracks and their characterization. The results generated with this pipeline represent a significant step towards an automated infrastructure damage

assessment, quoting the authors.

2.2 Methodology of the study

In the first study cited in this thesis, the authors offered a complete system for automatically creating damage-augmented digital twins for automated building inspection. The objective is to encode geometrical information about buildings as well as segment and characterize cracks based on their kinematics using computer vision and machine learning algorithms including SfM, CNN, and least-squares. This leads to the integration of a 3D LOD3 simplified model of the building with damage data and its characterization, paving the way for automated damage assessments for tracking an asset throughout its service life.

The pipeline begins with the collection of multiple-view RGB images which are then processed by the SfM framework to reconstruct the structure (point clouds) and motion (camera poses) of the building scene. From this information, an LOD2 model of the building is created by clustering plane primitives in the structure, and an LOD3 model is created by projecting openings segmented by deep learning models using camera poses and epipolar geometry. Before mapping to the LOD3 model with the camera positions provided by the SfM technique, damage is segmented using a trained CNN to generate binary crack patterns that are described by a crack kinematics algorithm, providing the building's DADT. The pipeline proposed in the current study is depicted below.

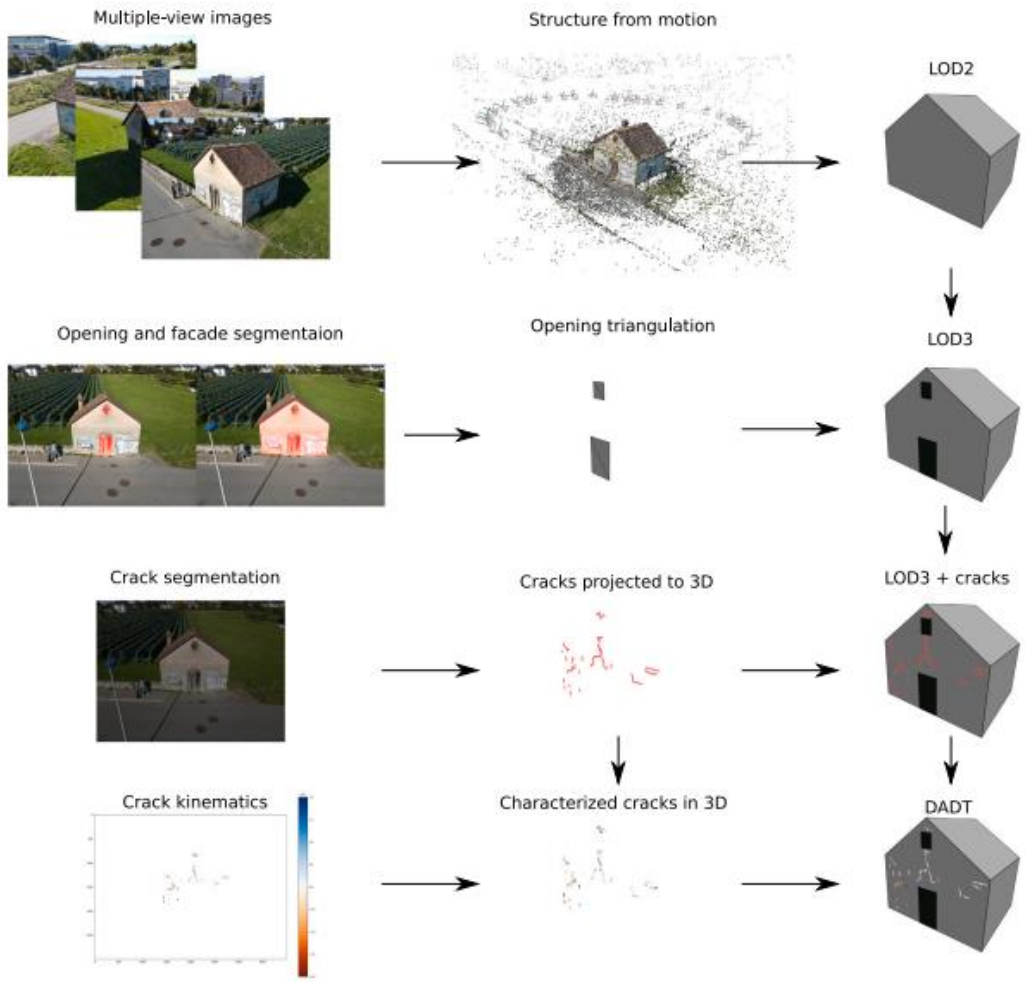


Figure 6: Pipeline for generating LOD3 models of freestanding buildings [31]. Top: LOD models are generated using the Polyfit framework. Bottom: LOD2 models are upgraded to LOD3 by segmenting and triangulating the 3D openings.

Firstly, the geometrical model (LOD3) has to be generated. To do this, the previously published work of Pantoja-Rosero et al. is utilized, in which the authors created the model using a combination of SfM and semantic segmentation. As shown in Fig. 6, their workflow starts by using SfM (Meshroom software) to create point clouds and camera projection matrices for each perspective of the building asset. The Polyfit algorithm processes the point cloud, grouping the points into planar primitives to produce candidate faces. These candidate faces are then selected using linear optimization to produce a LOD2 model.

To upgrade the LOD2 model to an LOD3 containing information about the building openings, the authors used images registered in the SfM pipeline and their corresponding camera projection matrix P to segment the openings and then triangulate them to 3D space using epipolar geometry. In their pipeline, the openings in 3D are represented by their corners $\mathbf{X}$, whose triangulation is possible by determining their corresponding 2D image coordinates in two views ($\mathbf{x}$, $\mathbf{x}'$) and the camera projection matrix of these two views ($\mathbf{P}$, $\mathbf{P}'$) and by using singular value decomposition to solve the equation below derived from epipolar geometry:

$$\begin{bmatrix} \mathbf{P} & -\mathbf{x} & 0 \\ \mathbf{P}' & 0 & -\mathbf{x}' \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \lambda \\ \lambda' \end{bmatrix} = \mathbf{0},$$

where λ and λ' are constants denoting the location of the point $\mathbf{X}$ over the rays connecting it to the respective camera centers. To extract information about the openings that is triangulated to 3D, facade segmentation is carried out using trained deep learning models in the image correspondent to the first view (detection of openings' corners $\mathbf{x}$). The deep learning model uses the TernausNet architecture and a dice loss function. After inference, the trained models generate three outputs as binary masks: opening corners, openings, and facade. These are combined to define the ($\mathbf{x}$, $\mathbf{x}'$) opening corner points to compute their 3D correspondences ($\mathbf{X}$) by solving the equation.

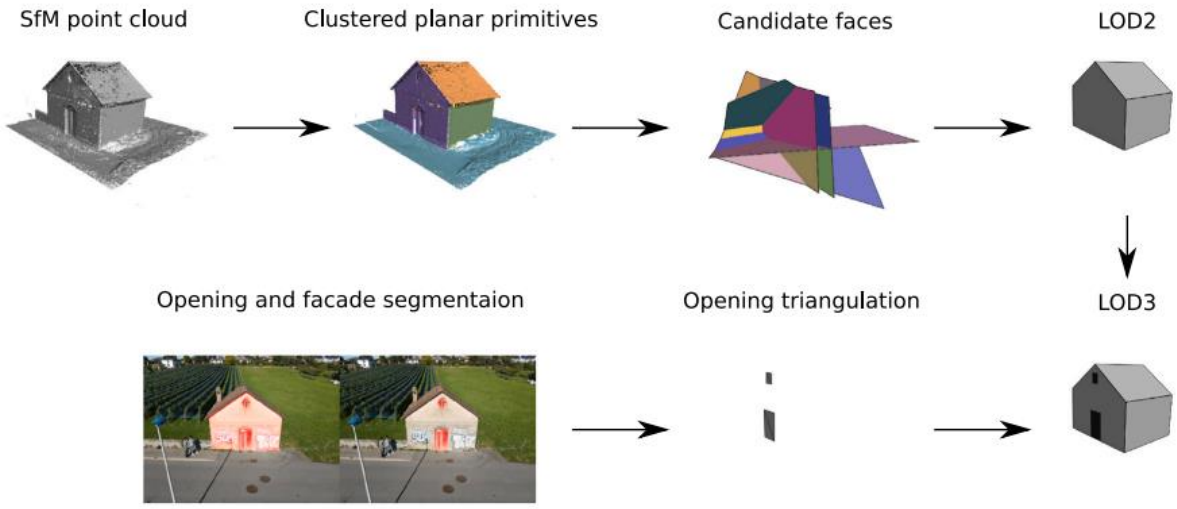


Figure 7: Pipeline for generating LOD3 models of freestanding buildings. Top: LOD models are generated using the Polyfit framework. Bottom: LOD2 models are upgraded to LOD3 by segmenting and triangulating the 3D openings.

The next essential step employed by the authors, is crack segmentation. The pipeline automatically projects detected cracks in images into three dimensions. The trained model developed by Pantoja-Rosero et al. is used for this purpose, which combines the FCNN

TernausNet architecture with a loss function that takes the topological information of the cracks into account. The loss function, TOPO-Loss, penalizes the loss value for pixels that result in discontinuous cracks in the forecasts while the ground truth crack is continuous by using the maximin connectivity technique. This method is utilized, in addition to its cutting-edge results for crack continuity, because the data set supplied and used for training in that study includes pictures of masonry buildings with the same typology as the examples here. The example shown in Fig. 8 illustrates how the deep-learning approach used in the pipeline segments cracks.

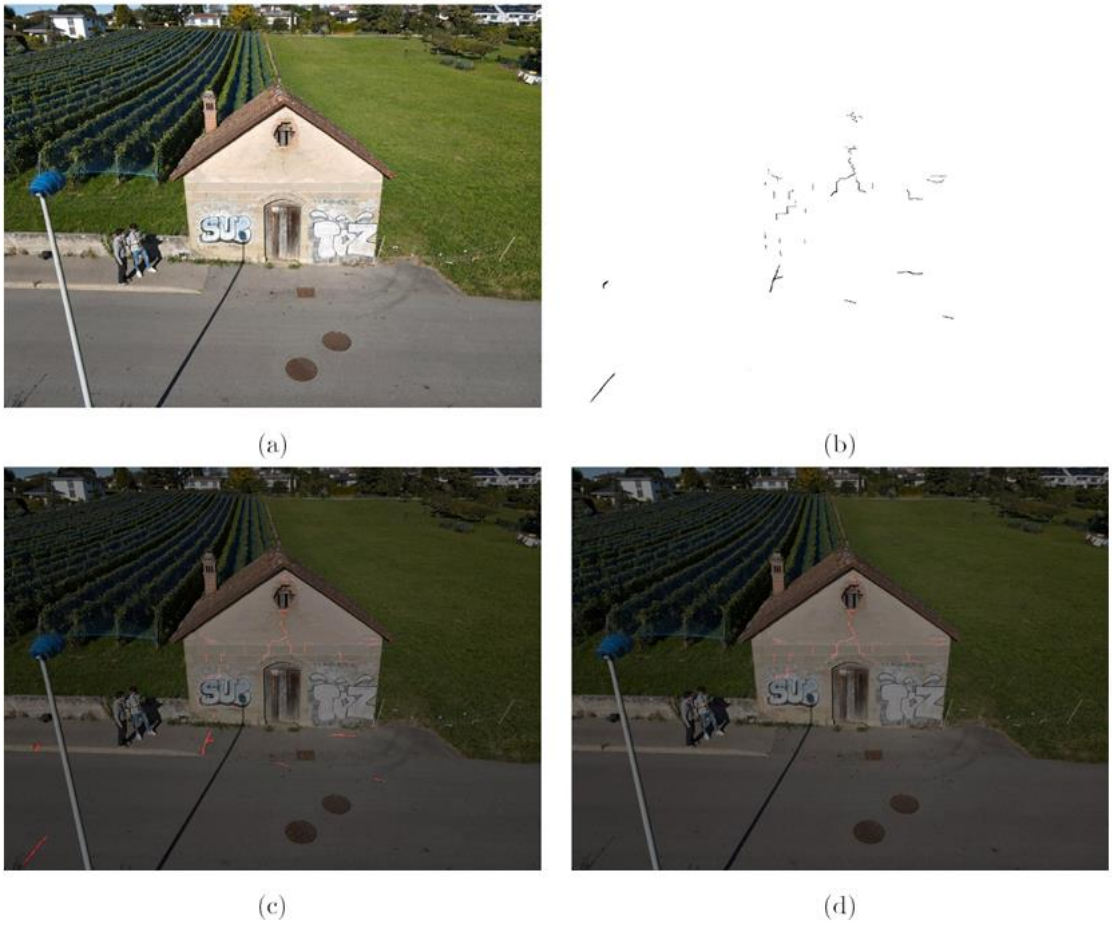


Figure 8: Crack segmentation using the deep-learning methodology developed by Pantoja-Rosero et al. [21]. (a) Input image of a damaged building. (b) Binary mask output from the deep-learning model. (c) Segmented cracks overlaid on the input image. (d) Cracks filtered using segmented facade and openings overlaid on the input image.

According to the study, it is crucial to accurately characterize the damage in order to ascertain how structural damage develops over time. By employing kinematics to characterize the cracks, the authors' research identifies the manner of crack propagation. In the framework, the problem is framed and treated as a 2D point-set registration problem, and the crack kinematics are calculated using a least-squares-based algorithm developed by Pantoja-Rosero et al. According to their research, crack propagation is characterized by displacements of the

crack pattern's edges throughout its length that are both normal and tangential. In order to calculate this, opposite edges along the crack were detected in a binary mask representing the segmentation of the crack pattern and then registered using non-linear least-squares to find an optimal transformation matrix that encodes the crack displacements during its propagation. In the current paper, the binary mask generated by the trained CNN is used as input.

To map these aforementioned cracks, the pipeline provides raycasting functionality that traces a ray beginning at the camera center of an image view with camera projection $\mathbf{P}$, passes by a point $\mathbf{x}$ in the image that belongs to the crack, and intersects $\mathbf{X}$ in its 3D correspondence with one of the planes of the LOD3 model $\Pi = AX + BY + CZ + D = 0$ (with unitary plane normal $\mathbf{p}n = [A, B, C]$). There is a lot more depth to the technique, which is beyond the scope of the present thesis.

Finally, this process is applied to each crack skeleton point to obtain their 3D correspondences, which are then merged with the LOD3 model to generate the DADT final output as shown in Fig. 9.

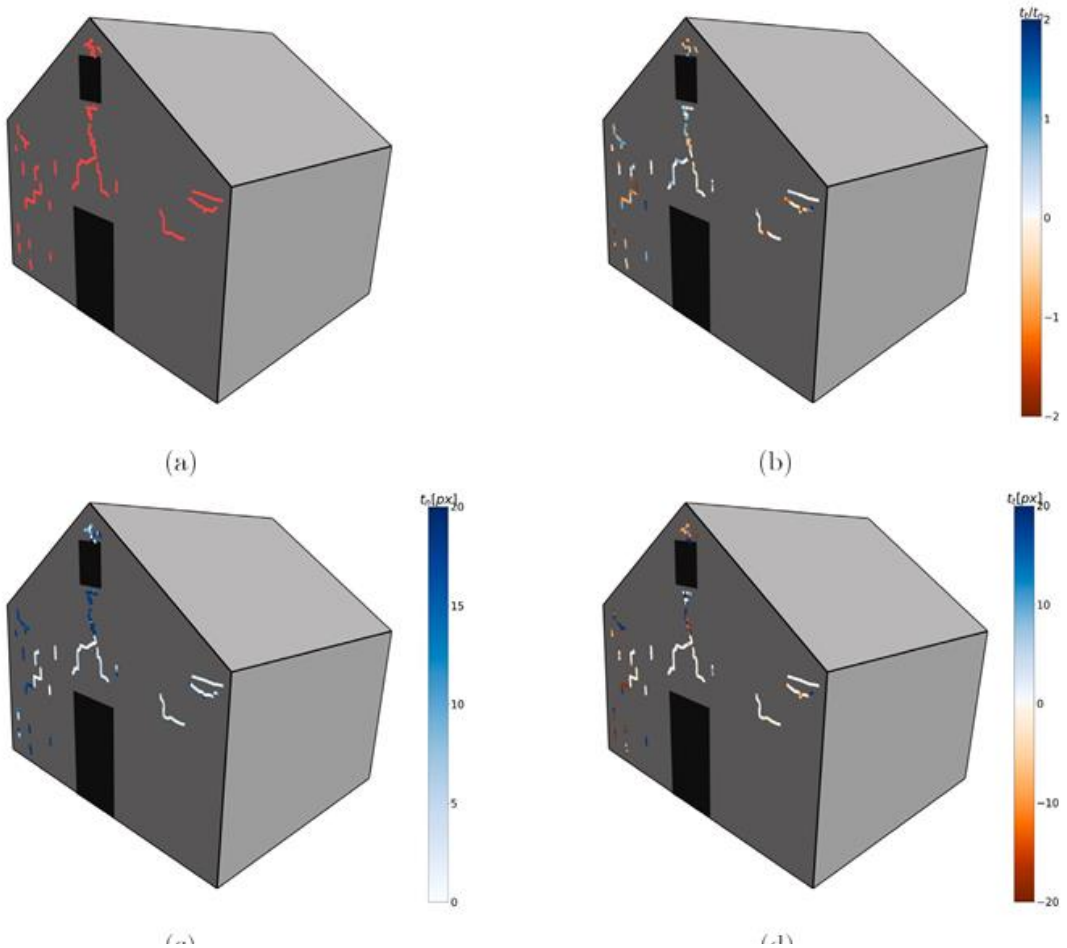


Figure 9: DADT—damage mapped to 3D and merged with an LOD3 model to generate the DADT. (a) DADT composed of LOD3 and cracks. (b) DADT composed of LOD3, cracks and tt/tn. (c) DADT composed of LOD3, cracks and tn. (d) DADT composed of LOD3, cracks and tt

The final deliverable of the process is a simplified geometry of the structure containing damage information. Nevertheless, the same procedure can be used to map the identified damage of more accurate models such as MVS, as shown in Fig. 10. Though these models require a considerable amount of storage space. Instead, the selected LOD3 models are compact and easy to store, making them ideal for use in situations in which numerous assets must be inspected, such as in post-earthquake damage assessments.

An essential feature of the proposed pipeline, is that similar to the process for cracks, the implementation can map any extracted semantic information from SfM-registered images, including linear shapes and polygons that can be defined by points ($\mathbf{x}$), as the DADT is generated by augmenting a geometric model with cracks and kinematics. As an example, Fig. 10 depicts a building in which painted graffiti on a facade surface are manually segmented from images by selecting polygonal vertices $\mathbf{x}$ and are then mapped to 3D to find $\mathbf{X}$ as the intersection of rays with geometric model planes. When other types of damage, such as out-of-plane deformations, spalling, leaching, rebar exposure, etc., are automatically segmented and need to be mapped to a 3D space to obtain a more comprehensive DADT representation for subsequent interventions, this feature becomes especially important. As for the accuracy of crack segmentation, the F1-score is employed, which is the most frequent metric for evaluating the results of binary segmentation methods.

Figure 10:. Augmenting models with extra semantic information. (a) Semantic information segmented from images. (b) MVS textured model, cracks and extra semantic information. (c) DADT composed of LOD3, cracks and extra semantic information.

2.3 Results and Conclusions

In the results segment, the authors present a variety of case studies that apply the proposed pipeline. The input data correspond to multiple-view images of free-standing masonry buildings in Croatia that were damaged by the 2020 Zagreb and Petrinja earthquakes, one of which is shown in Fig. 11. The results presented demonstrate the viability of this approach, which will allow for a more comprehensive representation of building damage that can be monitored over time.

Figure 11: DADT for free-standing masonry buildings whose damage is mainly cracks—View 2 (buildings A to E from top-down). (a) Image view of damaged building. (b) 3D textured models of the buildings merged with spatial information of the cracks. (c) DADT of LOD3 + cracks characterized via kinematics.

The values of the F1 score in regard to fracture detection are acceptable and showed that the majority of the cracks were found. The measure below indicates that on average 68% of the cracks are found. Although this number appears to be low when compared to other studies for crack segmentation, the goal of this study is to improve quick damage assessment practice, not crack detection accuracy, by providing a tool for documenting damage. However, as this methodology is independent of the crack segmentation technique, it is easily adaptable if the user has superior trained models.

Table 1
Quantitative performance evaluation for DADT models.

	LOD2		LOD3		Crack detection	File size DADT [MB]	
	FRDS	IMF	FRDS	IMF	F1	Textured	LOD
Building A	0.97	0.0252	0.94	0.0125	0.71	383.0	0.0424
Building B	0.94	0.0194	0.90	0.0146	0.69	405.9	0.1092
Building C	0.96	0.0659	0.94	0.0391	0.69	279.2	0.0546
Building D	0.91	0.0839	0.87	0.0782	0.69	222.2	0.0257
Building E	0.95	0.0477	0.94	0.0365	0.60	268.3	0.0326
Building F - fast	0.96	0.0471	0.94	0.0264	0.72	65.8	0.0190
Building G - extra	0.92	0.0280	0.90	0.0184	0.68	163.8	0.0173
Building H - extra	0.86	0.0520	0.84	0.0441	0.67	32.0	0.0222

The success of the methodology, the scientists observed in their conclusion, depends on the quality of the image data. They contend that uncertainty analyses for each pipeline component must be added in order to fully evaluate the pipeline as it is presented here; this work will be considered future. One significant aspect that is not evaluated in this work is damage characterization as the dataset at hand lacked information on crack measurements.

In conclusion the present study, proposes an end-to-end pipeline for automatically generating DADTs of freestanding buildings comprised of 3D simplified models and cracks. In the framework, multiple-view images of a building are processed via SfM, the output of which is used to generate a simplified polygonal surface model of the building. Cracks detected by a deep learning model are mapped to the geometrical model using SfM information to generate the DADT. This combines three cutting-edge methods to generate LOD3 models of buildings, semantically segment cracks, and characterize cracks with their kinematics. Contrary to existing techniques that use DTs of buildings for structural health monitoring, this particular pipeline does not call for manual user intervention, outputs a lightweight model that is perfect for storage and quick assessments, makes it easy to add information from image data, can be applied to multiple assets, and is flexible enough to work with other kinds of infrastructure.

CHAPTER 3 – RULE-BASED DEEP LEARNING METHOD FOR BUILDING FAÇADE DEFECT DETECTION [3]

3.1 Purpose of the study

Another very interesting study, that focuses on a different set of problems with the purpose of expanding the use of non-traditional methods in defects in historical building's façades, and improving the efficiency and accuracy of the existent techniques, outlines a rule-based deep learning approach for evaluation-oriented façade defects detection that can be utilized to generate useful evaluation areas using the data required for condition evaluation. As the paper suggests, the methods in current research studies mainly focus on accuracy improvement rather than providing effective evaluations of defects according to the requirement of industry standards. In addition, traditional machine learning algorithms are unable to achieve satisfying performance in detecting defects with higher complexity (e.g., spalling) or multi-class classification of defects. Therefore, this study proposes a rule-based deep learning method to achieve evaluation-oriented façade defects detection, which can be used to provide effective evaluation areas containing the necessary information (e.g., type, location, quantity, and size of façade defects) for condition evaluation.

As stated, in recent years, various standards have been published regarding various types of façades. According to the standards from the International Organization for Standardization (ISO), the type, location, quantity, and size of façade defects are the key indices for evaluating the condition of such facade. Rather than simply aggregating the information of each individual defect on the facade, ISO standards (i.e., the series of ISO 4628) require to define a certain evaluation area and evaluate its condition by rating the quantity and the average size of defects in the evaluation area. However, there is a disconnect between the demand for ISO-compliant condition evaluation and the state of defect detection research today. On the one hand, earlier research focused more on improving the classification, identification, or segmentation of problems than on offering helpful data for condition assessment.

For example, many developed models generated predictions that were overlapping with each other as shown in Fig. 12(a), which could cause overestimation of the severity if using the number of bounding boxes to calculate the number of defects. On the other hand, despite the fact that certain research has suggested strategies for defining assessment areas for condition evaluation in accordance with ISO standards, the established evaluation areas are insufficiently effective to deliver appropriate and objective information for the evaluation of façade condition. Many techniques made the assumption that any random area may serve as an evaluation area. When a number of defects are concentrated in a restricted area of the evaluation area, this assumption could, however, result in an underestimating of the severity level, such as Fig. 12(b).

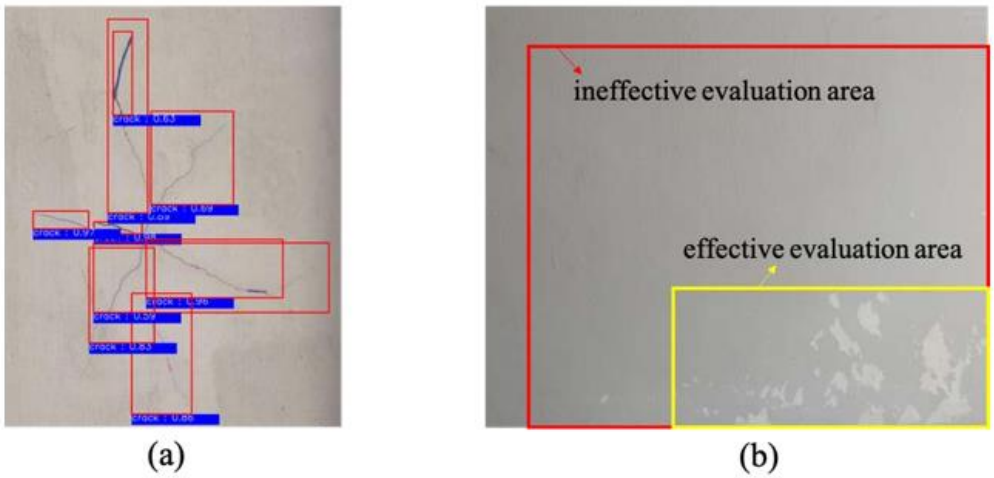


Figure 12: (a): Example of predictions that were overlapping with each other. (b): Example of number of defects concentrated in a restricted area of the evaluation area.

The purpose of the present study, is to propose a rule-based deep learning method for façade defects detection to automatically generate effective evaluation areas suitable for façade condition evaluation, according to ISO standards. Mask R-CNN is utilized as the foundational model to accomplish this goal, and it is further modified utilizing the created rules. The three stages of the development process—model training, model prediction, and dataset creation—all incorporate the designed rules. The designed rule is put into practice in each phase to alter the data generated to meet the needs of the evaluation region and increase detection accuracy.

As stated by the authors, building a training dataset is the first stage in achieving deep learning-based errors identification. To ensure that the detector performs as expected, the dataset must adhere to certain specifications. First, the dataset's fundamental requirement is the precise categorization of a variety of flaws. Some research used pre-existing guidelines, such as an official design handbook or a particular inspection operation code, to satisfy this criterion. Other studies developed their own categorization rule to produce the dataset. To

achieve the function of localization, the locations of defects should be annotated. The most convenient method used by a number of studies is to use bounding boxes to localize the defects, where the bounding boxes should be created close to the boundaries of defects. Confusion during the training stage resulted from the flaw covering the entire image, which was frequently labeled inconsistently. Furthermore, the bounding boxes employed in earlier research were simply used for localization and had no bearing on the assessment of the condition.

Related studies tried to tackle this problem, nonetheless, the proposed methods can be confusing and could cause inconsistency in the dataset for localization. Quoting the authors, although previous studies have explored the annotation rules for classification and segmentation, there are very few studies that proposed a rule to instruct the localization of the defects and took the requirement of evaluation into consideration when creating a dataset.

Using the created dataset, the developed deep learning model for defects detection can be trained to achieve the supposed functions. As the most widely adopted defects detector, the Faster R-CNN model contains a feature extractor to generate feature maps, a region proposal network (RPN) to provide proposals, and a region of interest (ROI) head to predict the location and class of each proposal.

To calculate the loss for backward computation and updating the parameters, the proposals generated by RPN should first be matched with ground truth bounding boxes with the largest intersection over union (IoU) to assign ground truth labels. Then, the proposals are randomly sampled for the next stage of prediction and loss calculation by comparing with the assigned ground truth labels. However, considering the irregular shapes of defects, the defects may not occupy the entire area of the ground truth bounding box. Instead, although having an IOU value above the threshold, the recommendations might only address a very tiny percentage of the defects. These suggestions should actually be classified as background defect items rather than foreground defect objects, which would turn into noise while determining the classification loss. However, this issue wasn't addressed in any of the earlier investigations.

Additionally, as the predictions made by these detectors frequently overlap, the number of bounding boxes used to calculate the number of defects might overestimate the severity of the problem. Instance segmentation approaches such Mask R-CNN, which can pinpoint the defect using a bounding box and segment the area of defect inside the bounding box, were used by some researchers to solve this issue, the authors note. Nonetheless, considering that the Mask R-CNN model is developed based on the Faster R-CNN model, the problems appeared in the Faster R-CNN model when dealing with defects detection still exist in Mask R-CNN.

In conclusion, as the authors note, directly using the Mask R-CNN model cannot provide an effective evaluation area as there are problems in all of the steps mentioned above. Firstly, during the dataset construction process, inconsistent annotations may result from a lack of specific and unambiguous annotation rules, particularly for the bounding boxes, which may cause confusion during the formation of evaluation areas. Then, at the model training step,

the quality of proposals is unstable due to special defects shapes, which can cause noises for the classification of the evaluation areas and lastly, at the model prediction step, several overlapped predictions may be generated with the same defect type which can cause repetitive computation on the evaluation area.

3.2 Methodology of the study

In the methodology segment of the study, the authors present a summary of their techniques used. The summary is explained in the Fig. 13 below and the methods are expanded upon in the following segments. To solve the aforementioned research problems for generating effective evaluation areas, this study optimizes the development procedure for achieving façade defects detection according to the requirement of condition evaluation standards. As shown in, the procedure consists of three steps: (1) dataset creation, (2) model training, and (3) model prediction.

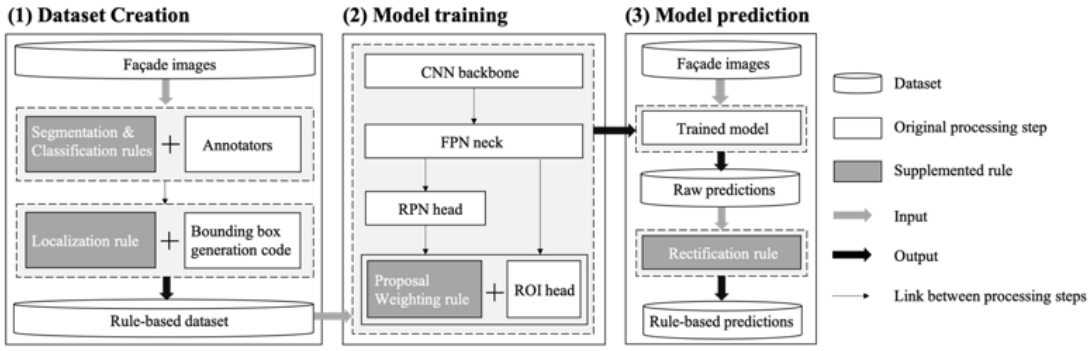


Figure 13: Development procedure for the evaluation-oriented façade defects detector.

First, at the dataset creation step, the annotators are required to manually annotate the boundary of each object using polygons and assign a defect category to each polygon. Then, a bounding box would be automatically generated for each polygon using an algorithm. Segmentation and classification guidelines are provided to the annotators in order to standardize the manual annotation labor for the evaluation of façade condition. For segmentation, the annotators must employ polygons to precisely identify all the regions afflicted by the defects as masks, and the annotation of the masks must be as thorough as feasible. For classification, the authors' previous study has proposed a classification rule for six defect categories. A bounding box is first constructed for each mask before localization.

Then, a localization rule is created to organize the annotations of individual defects into groups rather than directly output them to create rule-based annotations.

To create efficient evaluation zones for condition evaluation, defects are clustered. It makes more sense to combine two defects that are adjacent to one another and belong to the same defect category to create a wider bounding box for the evaluation area. In order to determine if two defects should be grouped together and placed in the same evaluation area, a distance threshold should be established. After interviewing several qualified façade inspectors for the purposes of the present study, it is found that inspectors usually define the grouping of defects based on their own experience and the evaluation precision required by the building owner. Hence, different distance thresholds are adopted in different scenarios.

As a result, the algorithm created in this work may categorize defects based on any given distance threshold d. According to the pseudo-code shown in Fig. 14, The algorithm will iterate until no annotations should be merged, ensuring that every annotation is correctly clustered in accordance with the distance threshold. Finally, all façade images have rule-based annotations created automatically for the model's training, including labels, bounding boxes, and masks.

Algorithm 1 Merge the annotations in an image based on the localization rule

Input: $L = \{l_n\}_{n=1}^{N}, B = \{b_n\}_{n=1}^{N}, M = \{m_n\}_{n=1}^{N}, d$ // L, B, M: lists of labels, bounding boxes, and masks
$U \leftarrow$ **unique** $(L) = \{c_u\}_{u=1}^{U}$ // obtain the unique class of labels
initialize merge ← 1
while merge > 0 **do**
    group {L, B, M} as $\{(L_u = \{l_i\}_{i=1}^{K_u}, B_u = \{b_i\}_{i=1}^{K_u}, M_u = \{m_i\}_{i=1}^{K_u})\}_{u=1}^{U}$, where $l_i \leftarrow c_u$ for l_i in L_u and $\sum_{u=1}^{U} K_u \leftarrow N$
    initialize merge ← 0, $L_{new} \leftarrow [\,]$, $B_{new} \leftarrow [\,]$, $M_{new} \leftarrow [\,]$
    for $u = 1$ to U **do** // for each unique class
        initialize $B_{merge} \leftarrow [\,]$, $M_{merge} \leftarrow [\,]$
        for $i = 1$ to K_u - 1 **do** // for each box corresponding to the unique class
            for $j = i + 1$ **to** K_u do // for the each of the rest boxes corresponding to the unique class
                if distance $(b_i, b_j) \leq$ d **or distance** $(b_j, b_i) \leq$ d **then** // decide whether b_i and b_j should be merged
                    merge ← merge +1
                    $B_{merge} \leftarrow B_{merge} \cup [b_i, b_j]$, $M_{merge} \leftarrow M_{merge} \cup [m_i, m_j]$
                end if
            end for
        end if
        end for
        $L_{new} \leftarrow L_{new} \cup \{l_h\}_{h=1}^{H}$, where $H \leftarrow$ **len (cluster** (B_{merge})**) and** $l_h \leftarrow c_u$ // cluster the boxes as groups
        $B_{new} \leftarrow B_{new} \cup$ **merge (cluster** (B_{merge})**)** // merge each group of boxes
        $M_{new} \leftarrow M_{new} \cup$ **merge (cluster** (M_{merge})**)** // merge each group of masks
        for $i = 1$ to K_u **do**
            if $b_i \cap B_{merge} == 0$ **then**
                $L_{new} \leftarrow L_{new} + l_i$, $B_{new} \leftarrow B_{new} + b_i$, $M_{new} \leftarrow M_{new} + m_i$
            end if
        end for
    end for
    $L \leftarrow L_{new}, B \leftarrow B_{new}, M \leftarrow M_{new}$ // update the annotations
end while // only when there are no boxes to be merged in the updated annotations can end the loop
Return L, B, M

Figure 14: Pseudo code for the algorithm to merge annotations in an image based on the localization rule.

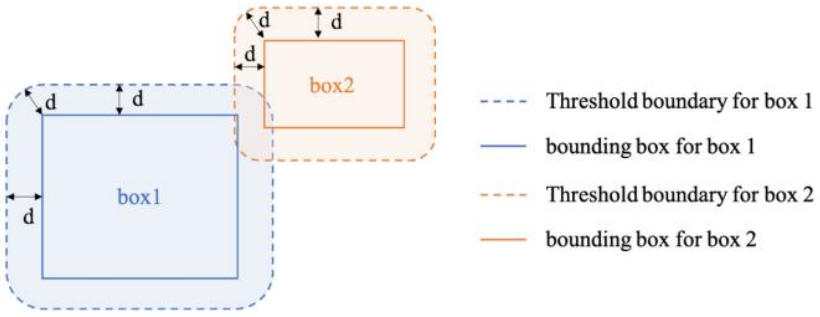


Figure 15: Deciding whether bounding boxes 1 and 2 should be merged based on distance threshold d.

Secondly, at the model training step, the model is optimized by combining a proposal weighting rule to selectively learn from the proposals. To accomplish the goals of delivering assessment information through instance segmentation, the model is built on Mask R-CNN design. The pixel-level instance of each object inside the image is identified by instance segmentation. The architecture of the suggested model is shown in Fig. 16. The proposal weighting rule is put between the RPN module and the ROI head in this study's proposed optimized model with a proposal weighting rule to enhance training quality.

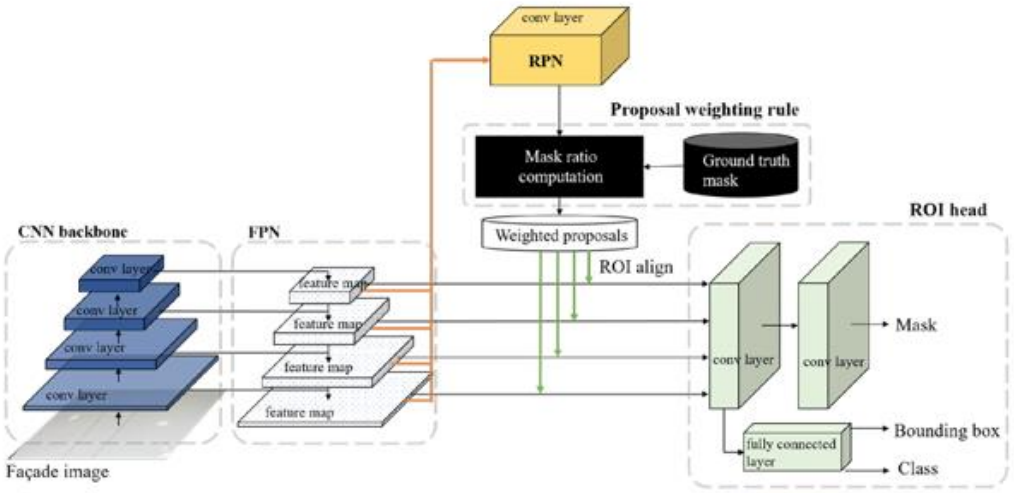


Figure 16: Architecture of the proposed façade defects detector based on Mask R-CNN model.

This model uses a standard ResNet-50 architecture for encoding the input façade images. At every convolutional layer, the sizes of feature maps are reduced by half and the number of feature maps is doubled through the CNN backbone. Through this feedforward calculation from a bottom-up pathway, four feature maps are generated from lower convolutional layers to upper convolutional layers. To generate final feature maps for processing in RPN, the original feature maps are processed via a top-down pathway by hallucinating higher resolution features to upsample the spatially coarser but semantically stronger feature maps from higher pyramid levels. Then, the features are combined using lateral connections from the bottom-up pathway and the top-down pathway to construct a feature pyramid that has rich semantics at all levels. The above processing steps on feature maps are formed as FPN. For each point in the feature maps, a set of anchors with various sizes and ratios are placed using the anchor box generator and sliding window. The RPN then uses each anchor's two output types of predictions—classification and bounding box regression—by combining them with the feature maps for each anchor. For categorization, two scores are produced: one for background (i.e., not a fault) and one for foreground (i.e., a defect). In bounding box regression, four scores are calculated for each of the four offset values (x, y, w, h), where (x,

y) corresponds to the center point's coordinates and (w, h) to the anchor's width and height, respectively.

After the anchors predicted as foreground have their offsets determined, the offsets are applied to those anchors to get the actual ROIs as proposals. The neighborhood anchors have somewhat comparable scores and are regarded as proposals for the same candidate region because the anchors are produced by the sliding window. Due to this, hundreds of thousands of redundant proposals are generated, and processing them is labor-intensive. The Non-Maximum Suppression (NMS) technique is used to reduce the overlapped proposals by filtering the foreground proposals depending on a threshold value of IoU.

These approaches nevertheless present a burden for subsampling calculations that require more computing. Furthermore, only a very small part of these region proposals—which are first categorized as foreground—are positive (i.e., defective), while the majority are negative (i.e., not defective), as the authors of the present study state.

A random sampler is used to select a predetermined number of proposals with predetermined ratios of positive and negative in order to solve these issues. An IoU matrix comprising the IoU between each proposal's bounding box and its ground truth is first calculated. The ground truth with the highest IoU value is determined for each proposal, and its bounding box and label are taken into consideration as the proposal's matching ground truths. Therefore, if the biggest U value already acquired above a specific IoU threshold, the proposition is said to be positive. If not, the proposition is deemed to be a negative one. Usually, the IoU threshold to classify the proposals is set as 0.5. However, the mask of the defect may only occupy a small portion of the area of the bounding box in comparison to the backdrop due to the particular forms of either the individual defect or defects group.

To solve the additional problem of the proposal only covering a very small part of the mask while the IOU is still larger than 0.5, which will make the proposal to be defined as a positive one and assigned a label of defect as ground truth during training, proposal weighting rule is designed to adjust the weights of proposals by preferring proposals with higher quality.

In this study, the proposal quality is determined by the value of mask ratio, which is calculated as the ratio between the area of ground truth mask inside the proposal and the total area of ground truth mask. A proposal with a higher mask ratio is supposed to have a higher quality on classification. Considering that some proposals may have high mask ratio but low IoU, directly using the mask ratio to substitute the IoU would influence the performance on regression. As a result, the IoU and mask ratio are combined in this study's proposals. The mask ratio is utilized to specify the weight of the proposal for the purpose of calculating classification loss, but the IOU threshold is still used as the index to give the ground truth label to the proposal. The weight increases as the mask ratio rises. The significance of the suggestion with a lower mask ratio would therefore be diminished in the overall loss.

Algorithm 2 Reweight the proposals

Input: $L = \{l_n\}_{n=1}^{N}, R = \{r_n\}_{n=1}^{N}$ // L, R: lists of ground truth labels and mask ratios of the sampled proposals
$U \leftarrow$ **unique** $(L) = \{c_u\}_{u=1}^{U}$ // obtain the unique class of ground truth labels
initialize $W \leftarrow \{w_n\}_{n=1}^{N}, w_n \leftarrow 1$ // initialize the weight of each proposal as 1
$w_{label} \leftarrow w_{sum}/U, w_{sum} \leftarrow \sum_{n=1}^{N} w_n,$
for u = 1 to U **do** // for each unique class
  initialize $idx \leftarrow [\,], R_{label} \leftarrow [\,]$
    for n = 1 to N **do** // for each proposal
      if $l_n == c_u$ **then**
        $idx \leftarrow idx \cup n, R_{label} \leftarrow R_{label} \cup r_n$
      end if
    end for
    $W_{label} \leftarrow w_{label} \times R_{label}/$**sum**$(R_{label})$ // calculate the weights according to the mask ratios
    for $i = 1$ to **len** (idx) **do**
      $W[idx[i]] \leftarrow W_{label}[i]$ // assign weight to the target proposal
    end for
end for
Return W

Figure 17: Illustration of the calculations of IoU and mask ratio.

In addition, this study also uses the proposal weighting rule to alleviate the imbalance problem of proposals. The imbalance problem of proposals is caused by the various sizes of objects in an image. Because the anchors are generated for each point of feature maps, there can be much more anchors matching the ground truth of larger objects than smaller objects. Therefore, the proposal weighting rule is further improved by assigning the same total weight to each group of proposals with the same label to alleviate the imbalance problem. Fig. 17 presented the pseudo-code to reweight the proposals according to the proposal weighting rule.

In summary, this proposal weighing rule added a weight to each proposal while maintaining the Mask R-CNN model's structural integrity. The mask ratio defines the proposals with greater quality, and the weight contributes to emphasize such offers. Because a proposal with a greater mask ratio implies that it contains a higher proportion of the object rather than the backdrop inside the bounding box, it should be emphasized that the weight would only be relevant for classification. Furthermore, by balancing the overall weights of suggestions with various labels, the weight also resolves the issue of imbalance on proposals between small defects and major defects.

Finally, the optimized model outputs a series of proposals with predicted coordinates for bounding box, a predicted label with a score, and a predicted mask of the defect. A rectification rule is used to adjust the raw predictions into rule-based predictions, as some proposals are still overlapped with each other or could not meet the defined distance for

evaluation. As the authors state, even those who created the model cannot fully comprehend how the variables are merged by the model to make predictions, despite the fact that the current training method has increased prediction accuracy. As a result, it's possible that the deep learning model won't create predictions that closely adhere to the dataset's predefined criteria.

To rectify the model predictions of defects to meet the requirement of condition evaluation, the raw predictions are further adjusted according to a rectification rule. This rectification rule is mainly based on the localization rule as shown in Figure 14.

So as explained before, to reduce invalid predictions with low confidence, predictions with scores lower than a threshold are filtered out. Predictions with low confidence would influence the adjustment quality based on the rectification rule and the prediction quality based on the detection accuracy. It should be noted, according to the authors, that because the score threshold is only applied to test images during application, it has no impact on the model's training performance. If two anticipated defects share the same defect category and are closer together than the threshold d, they will be combined. Additionally, the final score of a combined prediction result is the highest score among the merged forecasts. As a result, the rectification rule might be used to optimize the predictions for condition evaluation.

3.3 Experiments and Results

Two experiments were planned to test the performance improvement and to discuss the ability of the suggested strategy, with various distance threshold settings, in order to confirm the viability for constructing an evaluation-oriented façade faults detector.

For Experiment 1, four façade defects detector development settings were created:

1) Setting 1: original development procedure without rule,
2) Setting 2: development procedure with rule-based dataset creation,
3) Setting 3: development procedure with rule-based dataset creation and rule-based model training, and
4) Setting 4: development procedure with rule-based dataset, rule-based model training and rule- based model prediction.

The distance threshold for Experiment 1 is the 25th percentile, which is 246 pixels.

The goal of the second Experiment was to demonstrate that the suggested approach could be used with various distance threshold values to address various façade condition evaluation scenarios. In order to compare the three distances, which were chosen based on the distribution of distances between all manual annotations of defects with the same label, three distances were examined. As a result of the discovery that the majority of image defects could already be grouped using the distance value at the 50th percentile of all distances, the upper limit of the distance threshold for comparison in Experiment 2 is set at the 50th percentile, or

675 pixels. The other two distance thresholds for comparison are 96 pixels and 246 pixels, which are the 12.5th percentile and the 25th percentile, respectively.

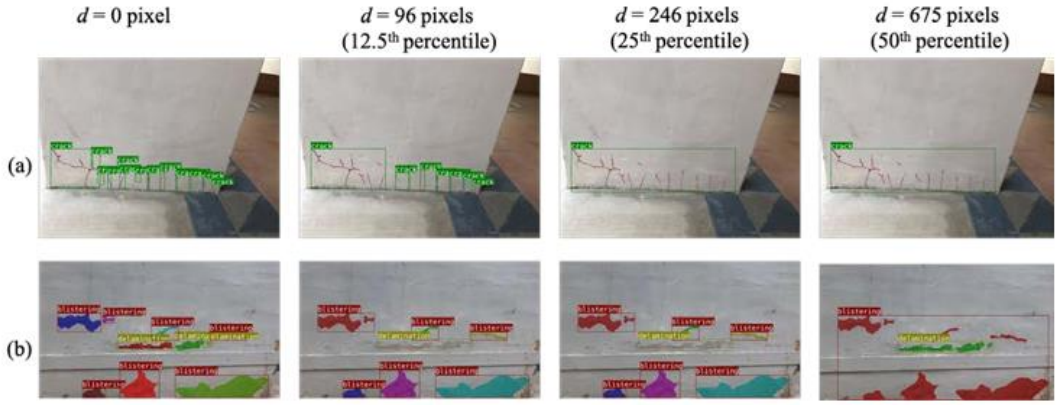


Figure 18: Examples of dataset under different settings of distance threshold: (a) example with crack and (b) example with blistering and delamination.

The visualization results in the following Table, show the comparison in the results of the first Experiment for the four Settings.

	mAP50 (localization)				mAP50 (segmentation)			
	Min	Max	Mean	Variance	Min	Max	Mean	Variance
Setting 1	0.5036	0.5143	0.5089	0.1369	0.4609	0.4683	0.4640	0.0906
Setting 2	0.6196	0.6345	0.6281	0.2329	0.5530	0.5612	0.5470	0.0921
Setting 3	0.6460	0.6496	0.6480	0.0148	0.5740	0.5774	0.5767	0.0103
Setting 4	0.6489	0.6537	0.6509	0.0217	0.5816	0.5855	0.5837	0.0157

Comparison of mAP50 (localization) and mAP50 (segmentation) for the four settings

Experiment 1's findings can be summed up by asserting that the comparison showed how the designed rules, which were implemented in three steps—dataset creation, model training, and model prediction—successfully enhanced the performance of the façade defects detector in both localization and segmentation. The deep learning model was able to offer bounding boxes that were better suited for condition evaluation and had a greater detection accuracy thanks to the rule-based dataset creating process. By producing more proposals that were correctly classified and enhancing the training quality to stabilize the training process, the rule-based model training stage further increased the detection accuracy. Furthermore, the rule-based model prediction step rectified the predictions into effective evaluation areas. Although the total running time for training and testing after applying the three designed rules had increased, according to the study, the detection accuracy (i.e., mAP50) in localization and segmentation had improved by 27.9% (from 0.5089 to 0.6509) and 25.8% (from 0.4640 to 0.5837), respectively. Besides, the results of Setting 4 were much more effective for condition evaluation compared to the results of Setting 1. Therefore, Experiment 1 demonstrated superior performance compared to the traditional model.

As for the second Experiment conducted in this study, as shown in Fig. 19, the value of mAP50 of Setting 2 did not have an obvious change as the distance increased. However, the values of mAP50 for both Settings 3 and 4 increased with the increase of distance. The predictions of Settings 3 and 4 had a similar trend because the model of Setting 4 was inherited from Setting 3. There are two potential reasons that caused the increase of mAP50 for Settings 3 and 4. On the one hand, the dataset itself became easier to be learned and verified because number of small objects was decreased with the increase of distance and the bounding boxes became coarser on localization. On the other hand, the rule-based model training in both Settings 3 and 4 kept the quality of training stable by always highlighting high-quality proposals. However, Setting 2 could not benefit from the dataset with a larger distance setting because the model of Setting 2 could generate more low- quality proposals with the increase of distance, considering that the proportion of background further increased with the increase of distance. This problem also caused the increase of variance for Setting 2 as the distance increased, whereas the predictions from Settings 3 and 4 remained stable.

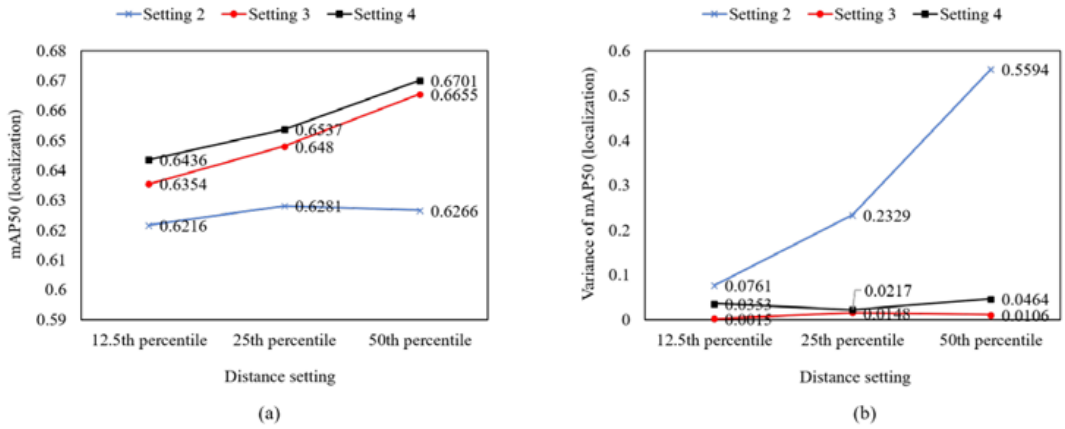


Figure 19: Comparison of the results for the three distance settings: (a) mAP50 of localization and (b) variance of mAP50 of localization.

Additionally, it became clear that predictions made using Setting 4 consistently had the highest mAP50, demonstrating the ability of the rule-based model prediction to rectify the predictions under different distance settings. The rectification rule for model prediction was, however, sensitive to the predictions due to the substantially higher variance of Setting 4 compared to Setting 3, even minute changes to the anticipated bounding boxes from Setting 3 might produce completely different predictions from Setting 4. Regardless of the adjustments to the distance setting, the three rule-based settings all performed better than Setting 1 overall. Furthermore, the model from Setting 4 consistently outperformed other models at various distance settings, and the change in distance settings had no significant impact on its performance.

3.4 Conclusions of the study

By enhancing the development of the existing Mask R-CNN model, this study aims to provide helpful information, especially effective evaluation regions, for achieving visual evaluation of façade condition in accordance with ISO standards. By confronting the research gaps in computer vision-based condition evaluation for building facades and offering a novel method for fusing the particular needs of civil engineering with the comprehensive model from the field of computer science, the proposed rule-based deep learning method makes a valuable contribution to the research community. On the other hand, this work also makes a contribution to the construction industry by offering a method for detecting evaluation-oriented façade defects that might produce effective evaluation areas for rating the condition in accordance with ISO standards. However, quoting the authors, there is still space for improvement in the detection precision of the final predictions. In this study, segmentation optimization work was also largely neglected. Future research needs to add more training data to significantly increase detection accuracy. Moreover, by optimizing the segmentation branch of prediction in subsequent studies, the size and quantity of defects included in an evaluation area can be more precisely calculated.

CHAPTER 4 – YOLO MODEL: USES AND ARCHITECTURE

4.1 Introduction

The YOLO (You Only Look Once) framework has become a central real-time object detection system for robotics, driverless cars, and video monitoring applications [48]. In this short segment, a comprehensive analysis of YOLO's architecture and use cases is presented, examining the innovations and contributions in each iteration from the original YOLO accentuating the Ultralytics developed versions, YOLOv5 and especially YOLOv8, which was used in the specific dissertation. There is a plethora of improvements made across YOLO's evolution encompassing various aspects such as network design, loss function modifications, anchor box adaptations, and input resolution scaling. This underscores the importance of considering the context and requirements of specific applications when selecting the most appropriate YOLO model, aiming to justify the use of the new YOLOv8 for the present thesis [48].

Among the various object detection algorithms, the YOLO framework has stood out for its remarkable balance of speed and accuracy, enabling the rapid and reliable identification of objects in images. YOLO's real-time object detection capabilities have been invaluable in autonomous object detection applications, enabling quick identification and tracking of various objects [48]. These capabilities have been applied in numerous fields, including action recognition in video sequences for surveillance, sports analysis, and human-computer interaction [48]. In order to support precision agriculture methods and automate farming procedures, YOLO models have been used in agriculture to identify and categorize crops, pests, and diseases. Additionally, they have been modified for use in biometrics, security, and facial recognition systems for face detection tasks. YOLO has been employed in the medical field for pill identification, skin segmentation, and cancer detection. This has improved diagnostic accuracy and streamlined treatment procedures. Land use mapping, urban planning, and environmental monitoring have all benefited from its application in remote sensing, which involves the detection and classification of objects in satellite and aerial imagery. For real-time video feed monitoring and analysis, security systems have incorporated YOLO models. This enables quick identification of suspicious activity, social distancing, and face mask detection. Additionally, for the purpose of managing ecosystems and conserving biodiversity, they have been used in the detection and monitoring of wildlife. Finally, YOLO has been applied extensively in robotics and drone object detection [48].

Traditionally, object detection algorithms can be divided into two main categories: single-shot detectors and two-stage detectors. R-CNN, Fast R-CNN, Faster R-CNN and Mask R-CNN are two-stage detectors that use regions to localize the objects with in image. The network looks

at some parts of the image which have the highest probabilities of containing the object. YOLO is an object detection algorithm which is different from the regions-based algorithms (R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN). In YOLO, a single Convolutional Neural Network predicts the bounding boxes and the class probabilities for these boxes [49].

In order to decrease the amount of overlapping bounding boxes and enhance the overall detection quality, object detection algorithms including the YOLO framework employ the post-processing technique known as non-maximum suppression (NMS). Multiple bounding boxes surrounding the same object are frequently created by object detection algorithms, each with a distinct confidence score. Only the most precise bounding boxes are retained after NMS filters out unnecessary and unimportant ones [48].

4.2 YOLOv1

Beginning from the original YOLO version, the YOLO model is made up of the three key components, found in most modern detector models: the head, neck, and backbone [49], as shown in Fig. 20.

- The backbone is where the convolution layers detect the key features of the image for processing. This is typically pretrained on a classification dataset.
- The neck takes the features from the backbone into fully connected layers to predict the bounding box coordinates and classification probabilities.
- Finally, the final output layer is the head. It can be interchanged with other layers with the same input shape for transfer learning.

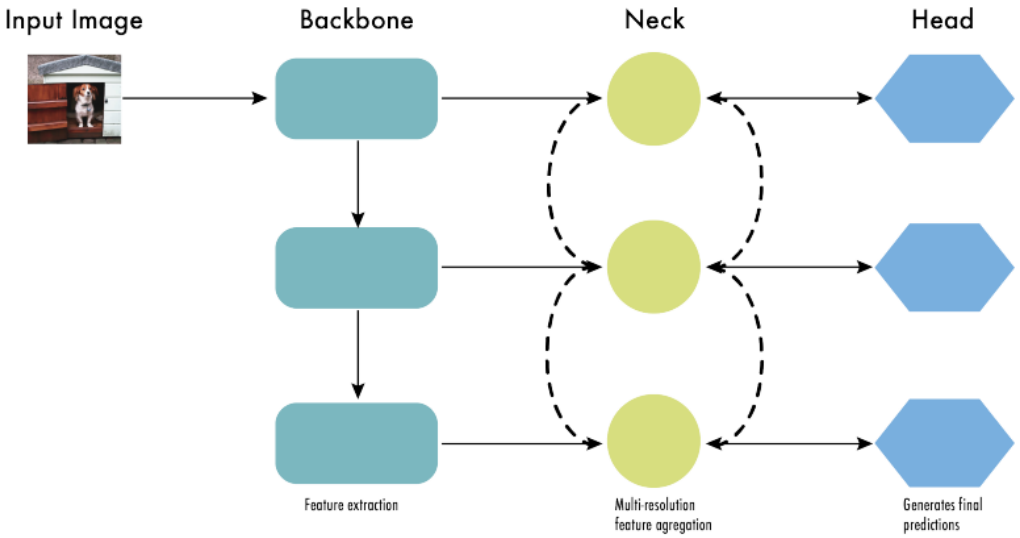


Figure 20: The three key components, comprising a CNN's architecture.

YOLOv1 architecture comprises of 24 convolutional layers followed by two fully-connected layers that predict the bounding box coordinates and probabilities. All layers use **leaky rectified linear unit activations** (ReLU) except for the last one that uses a **linear activation function**. YOLO uses 1 × 1 convolutional layer to reduce the number of feature maps and keep the number of parameters relatively low.

The YOLOv1 model unified the object detection steps by detecting all the bounding boxes simultaneously. To accomplish this, YOLO divides the input image into a S × S grid and predicts B bounding boxes of the same class, along with its confidence for C different classes per grid element.

Each bounding box prediction consists of five values: P c, bx, by, bh, bw where P c is the confidence score for the box that reflects how confident the model is that the box contains an object and how accurate the box is. The bx and by coordinates are the centers of the box relative to the grid cell, and bh and bw are the height and width of the box relative to the full image. The output of YOLO is a tensor of S × S × (B × 5 + C) optionally followed by non-maximum suppression (NMS) to remove duplicate detections [48].

Fig. 21 shows a simplified output vector considering a three-by-three grid, three classes, and a single class per grid for eight values. In this case, the output of YOLO would be a 3 × 3 × 8 tensor. YOLOv1 achieved an average precision (AP) of 63.4 on the PASCAL VOC2007 dataset.

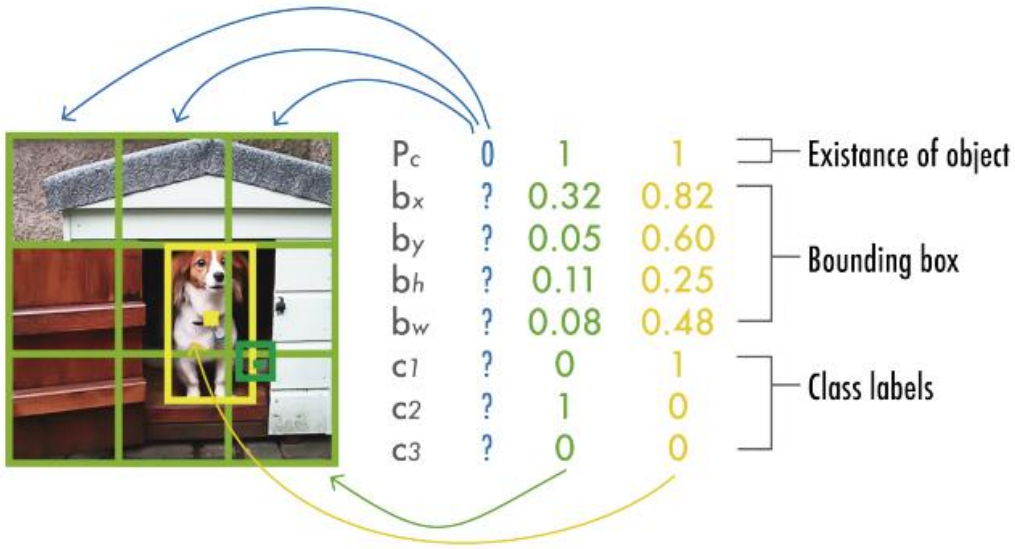
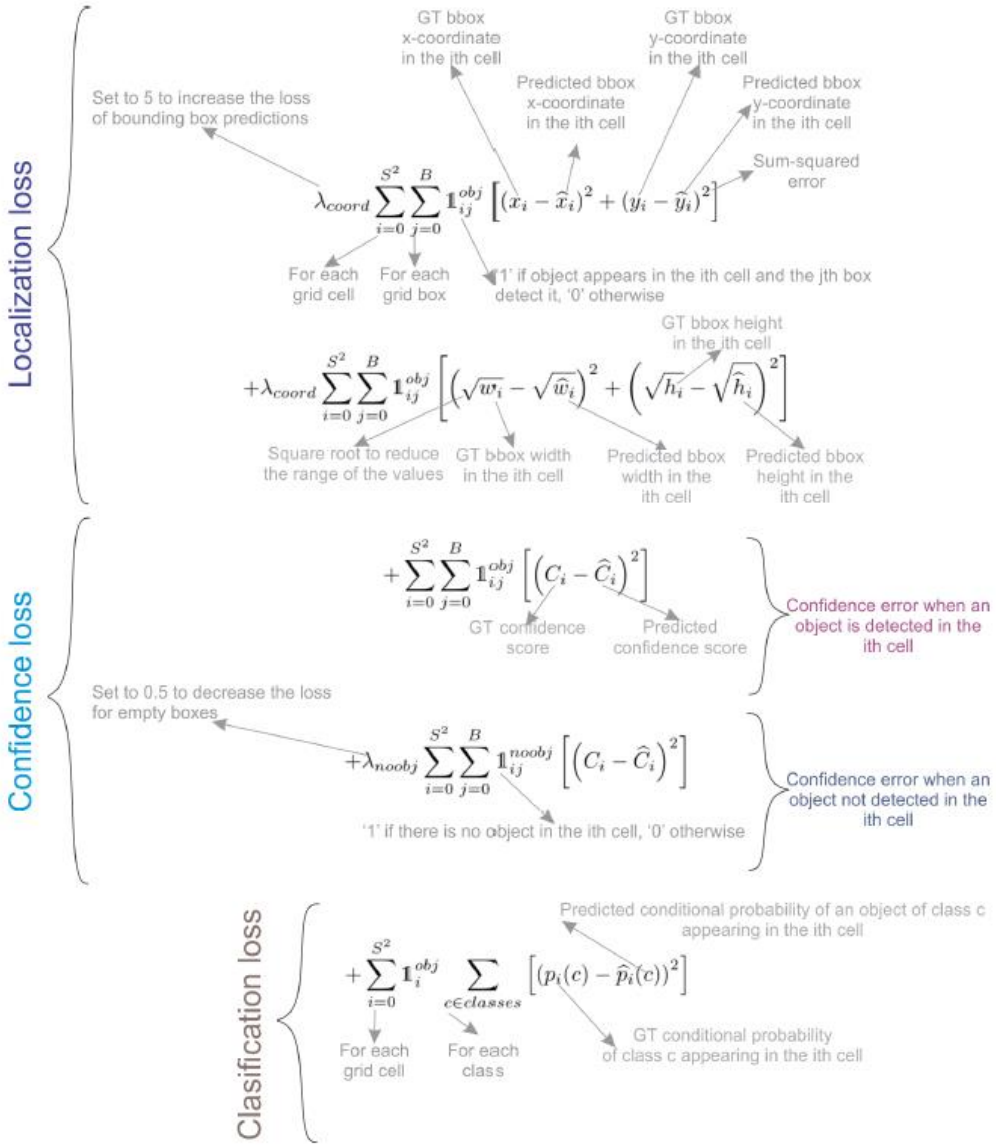


Figure 21: YOLO output prediction. The figure depicts a simplified YOLO model with a three-by-three grid, three classes, and a single class prediction per grid element to produce a vector of eight values.

YOLOv1 used a loss function composed of **multiple sum-squared errors**. In the loss function there are scale factor that give more importance to the bounding boxes predictions, and decrease the importance of the boxes that do not contain objects. The first two terms of the loss represent the localization loss. The third and fourth loss terms represent the confidence loss. The third term measures the confidence error when the object is detected in the box (1 obj ij) and the fourth term measures the confidence error when the object is not detected in the box (1 noobj ij). The final loss component is the classification loss that measures the squared error of the class conditional probabilities for each class only if the object appears in the cell (1 obj i) [48].



The simple architecture of YOLO, along with its novel full-image one-shot regression, made it much faster than the existing object detectors allowing real-time performance. However, while YOLO performed faster than any object detector, the localization error was larger compared with state-of-the-art methods such as Fast R-CNN. There were three major causes of this limitation [48]:

1. It could only detect at most two objects of the same class in the grid cell, limiting its ability to predict nearby objects.

2. It struggled to predict objects with aspect ratios not seen in the training data.

3. It learned from coarse object features due to the down-sampling layers.

4.3 YOLOv5

YOLOv5 was released in 2020 by Glen Jocher, founder and CEO of Ultralytics. It uses many features present in former releases but is developed in Pytorch instead of Darknet. YOLOv5 incorporates an Ultralytics algorithm called AutoAnchor. This pre-training tool checks and adjusts anchor boxes if they are ill-fitted for the dataset and training settings, such as image size. It first applies a k-means function to dataset labels to generate initial conditions for a Genetic Evolution (GE) algorithm. The GE algorithm then evolves these anchors over 1000 generations by default, using CIoU loss and Best Possible Recall as its fitness function. Fig. 22 shows the detailed architecture of YOLOv5 [48].

The backbone is a modified CSPDarknet53 that includes convolutional layers that extract relevant features from the input image after a Stem, a strided convolution layer with a large window size to reduce memory and computational costs. The upsample layers improve the resolution of the feature maps, while the SPPF (spatial pyramid pooling fast) layer and the subsequent convolution layers process the features at different scales. Through the pooling of features from various scales into a fixed-size feature map, the SPPF layer seeks to accelerate network computation. SiLU (sigmoid function) activation and batch normalisation (BN) follow each convolution [48]. The neck uses SPPF and a modified CSP-PAN, while the head resembles YOLOv3. YOLOv5 uses several augmentations such as Mosaic, copy paste, random affine, MixUp, HSV augmentation, random horizontal flip, as well as other augmentations from the albumentations package. It also improves the grid sensitivity to make it more stable to runaway gradients [48].

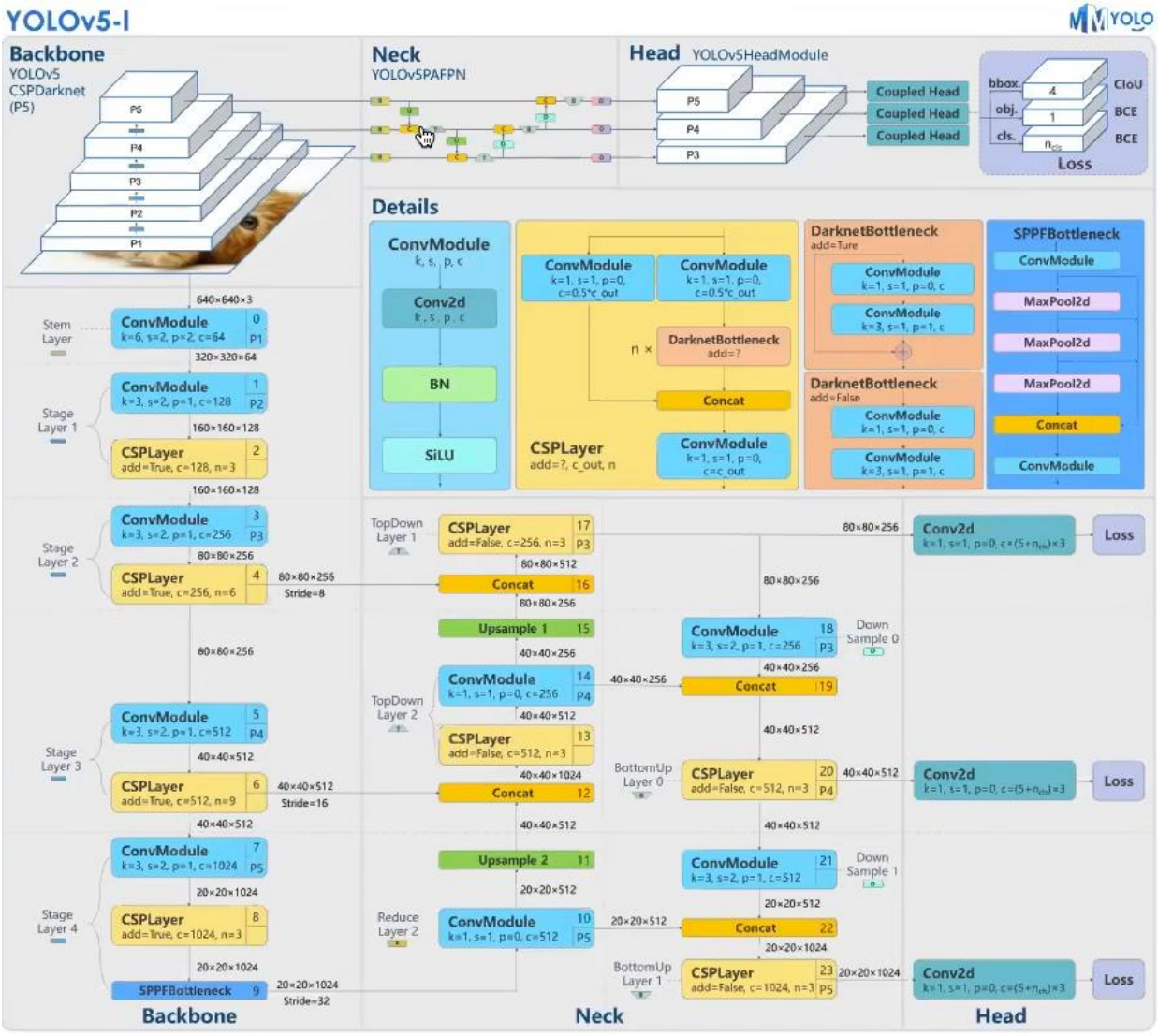


Figure 22: YOLOv5's architecture

4.4 YOLOv8

YOLOv8 was released in January 2023 by the same company that developed YOLOv5, Ultralytics. It provided five scaled versions: YOLOv8n (nano version), YOLOv8s (small version), YOLOv8m (medium version), YOLOv8l (large version) and YOLOv8x (extra-large version), with mAP values and speed average increasing the larger the version. YOLOv8 supports multiple vision tasks such as object detection, segmentation, pose estimation, tracking, and classification. Figure 23 shows the detailed architecture of YOLOv8 [9]. YOLOv8 uses a similar backbone as YOLOv5 with some changes on the CSPLayer, now called the C2f module. The C2f

module (cross-stage partial bottleneck with two convolutions) combines high-level features with contextual information to improve detection accuracy.

This specific version of YOLO uses **an anchor-free model** with a decoupled head to independently process classification, objectness and regression tasks. This means it is able to automatically detect the central point of an object rather than the object's offset from the center of the pre-defined anchor box for that region. This design is essentially allowing each branch to focus on its specific task improving the model's overall accuracy and speed. Anchor free detection reduces the number of box predictions, which speeds up Non-Maximum Suppression (NMS) [50]. In the output layer of YOLOv8, **the sigmoid function** is used as the activation function for the objectness score, representing the probability that the bounding box contains an object. It uses the **softmax function** for the class probabilities, representing the probability of the object belonging to each possible class. YOLOv8 uses **CIoU and DFL loss functions** for bounding box loss and binary cross-entropy for classification loss. The use of those loss functions has improved the performance of the object detection mode, especially when dealing with smaller objects, which have been proven particularly difficult to confront [51].

This version also provides a semantic segmentation model called YOLOv8-Seg model. The backbone is a CSPDarknet53 feature extractor, followed by a C2f module instead of the traditional YOLO neck architecture. The C2f module is followed by two segmentation heads, which learn to predict the semantic segmentation masks for the input image. The model has similar detection heads to YOLOv8, consisting of five detection modules and a prediction layer [51].

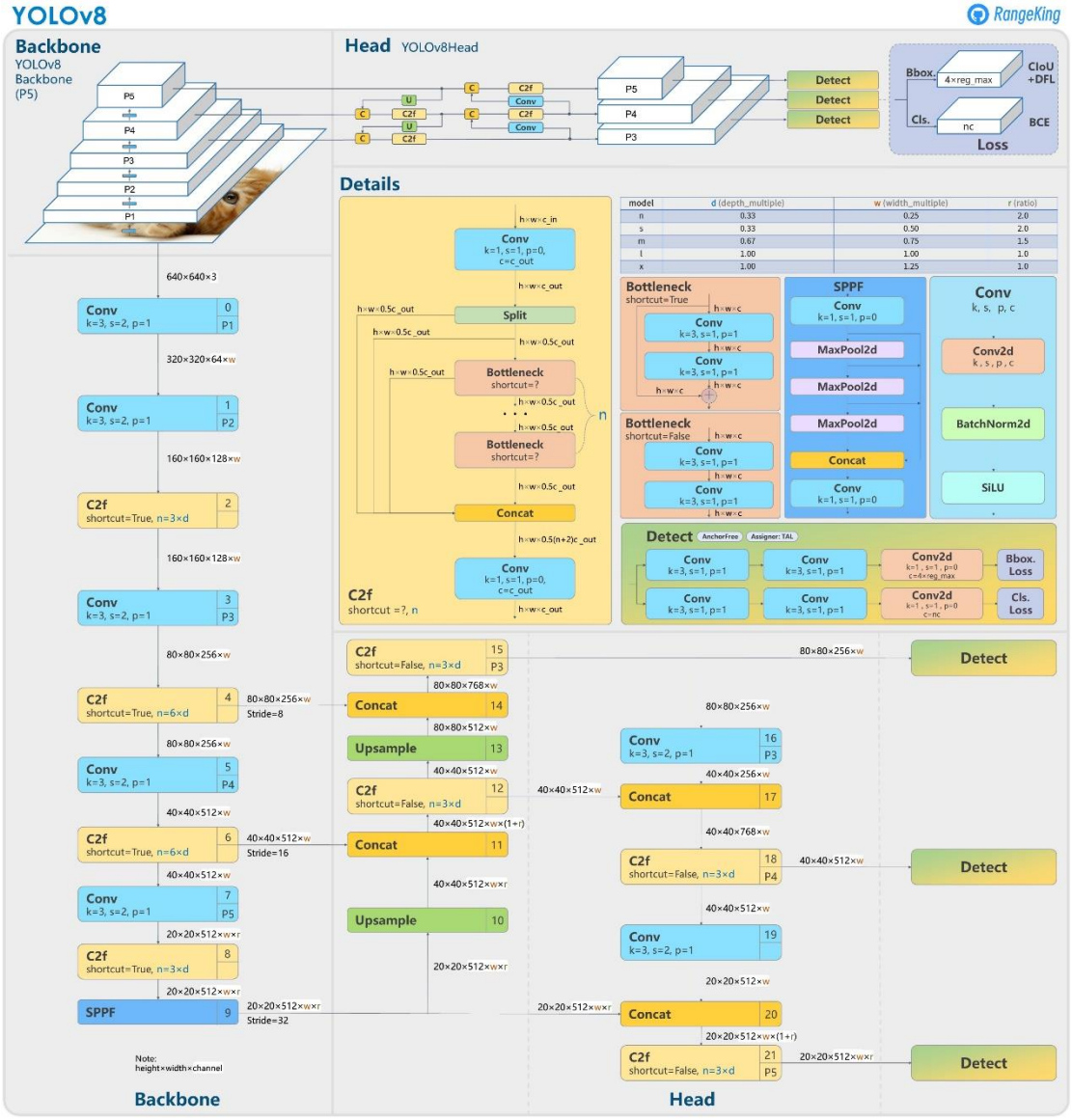


Figure 23: YOLOv8 Architecture

The YOLOv8-Seg model has achieved state-of-the-art results on various object detection and semantic segmentation benchmarks while maintaining high speed and efficiency. YOLOv8 can be run from the command line interface (CLI), or it can also be installed as a PIP package. In addition, it comes with multiple integrations for labeling, training, and deploying [51].

CHAPTER 5 – METHODOLOGY

5.1 The Combination of the two methods

As the subject of the present dissertation divulges, the goal is to capture façade images of historical buildings in order to analyze them, detecting defects and assessing any risks for further damage. This way the time-consuming and subjective inspections that have to be executed on a regular basis, in order to preserve these buildings, can be automated. The results will have to be objective and consistent with the current way defects are being characterized.

To automatically achieve visual evaluation of façade condition with high accuracy, studies have applied various machine learning and deep learning algorithms to classify, localize, and segment the defects. However, this approach poses a new set of setbacks regarding its use in historical building maintenance, and each one shows up in the real world in an irregular pattern [1]. Automated defect monitoring systems should be able to simultaneously detect and accurately categorize numerous types of faults in picture data in order to take into account the characteristics of these problems. The methodology this paper is proposing, is a combination of the ideas from the two distinct studies covered above, expanding on them to solve core issues with the current techniques used.

On the one hand, by generating DADTs of historical buildings, as proposed in Chapter 2 [22], a 3D reconstructed geometry of a building asset can be provided, without the interference of lighting conditions and even foreground and background foreign objects. This way the complex backgrounds and the subjectivity of the images provided are eliminated. Consequently, the automatically generating DADTs, are excellent to provide clear images of each façade in the building. This methodology grants complete and convenient access to every outer surface of the building, offering detailed images of its façades, rendering the complicated defects as clear as possible. Therefore, this technique is essential for the purposes of the present study, where real world images of historical buildings are employed.

On the other hand, the thorough study in Chapter 3 of the present thesis, made clear through the experiments it conducted, that the common approaches used to cater the need of automated visual evaluation of façade condition are not sufficient. These approaches, focused, almost exclusively on maximizing the accuracy of their predictions, not accounting for the use of their methods in real world scenarios, where the defects are of much higher complexity [3]. Therefore, the study proposed a method, implementing essential rules in different parts of the prediction process, to satisfy the evaluation requirement based on ISO standards.

The present dissertation, implements a CNN approach to the issue, modifying the aforementioned methods and combining them into one comprehensive and efficient automated model. The segments to follow, explain the methodology of the approach, initiating with the development of the rule-based CNN model and the DADT generation of

real-world historical buildings. Then the methods are combined and the results are evaluated in regards to the accuracy and efficiency of the resulting model.

5.2 Dataset and annotations

In order to achieve the aforementioned efficiency and accuracy in the results, the deep learning model had to be thoroughly trained for the specific application it will be used in: Façade Defects Detection. The term 'training' of a neural networks is defined as the process of adjusting the value of the weights of the model and is the step where the model is 'learning' [30].

Firstly, the CNN initiates with the random weights. During the training of CNN, the neural network is presented with a large dataset of images being labeled with their corresponding class labels (crack, mold, peeling paint etc.). Then the CNN processes each image with its values being assigned randomly and makes comparisons with the class label of the input image [30]. In the case of the output not matching the class label, which is most common at the beginning of the training process and therefore makes a respectively small adjustment to the weights of its CNN neurons, in order for the output to correctly match the class label image. The corrections to the value of the weights are being made employing a technique known as backpropagation. Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent. The approach determines the gradient of the error function with regard to the weights of the artificial neural network given an error function and an artificial neural network [52].

Each and every run of the training of the image dataset is being called an "epoch." The CNN model goes through several series of epochs during the process of training, adjusting its weights as per the required amounts. After each epoch step, the neural network becomes more accurate and confident at classifying and correctly predicting the class of the training images. As the CNN improves, the adjustments being made to the weights become proportionally smaller and smaller accordingly [30].

Subsequently a separate, validation set of data is used to evaluate and fine-tune the machine learning model, helping to assess the model's performance and make various adjustments. By evaluating a trained model on the validation portion of the dataset, its ability to generalize to unseen and unknown data is assessed. This assessment helps identify potential issues such as overfitting, which can have a significant impact on the model's performance in real-world scenarios [53]. A machine learning model can be tailored to the training set of data and becomes overfitted losing its ability to generalize and generate accurate predictions on new data. As such, Overfitting negatively impacts the model's performance in predicting test and validation sets, in contrast to the training set. The validation set is also essential for hyperparameter tuning. Hyperparameters are settings that control the behavior of the model, such as learning rate or regularization strength [53]. This process fine-tunes the model and

maximizes its performance, providing the user with invaluable data in regards to the efficiency of the network.

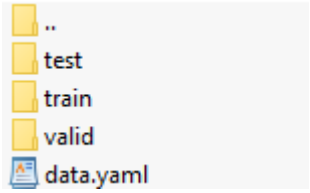
After the training of the CNN and the validation of its performance is performed, a test dataset is used to verify its accuracy. The test dataset is a set of labelled images that are not included in the training or the validation process so that they are foreign to the model. Each image is presented to the CNN, and the output is compared to the actual annotated class label of the test image. Essentially, this step serves as an unbiased measure of how well the model generalizes to unseen data, assessing its generalization capabilities in real-world scenarios. By keeping the test set separate throughout the development process, a reliable benchmark of the model's performance is obtained [53]. The test dataset represents unseen data that the model has never encountered before, evaluating the model fit on the test set provides an unbiased metric into its practical applicability [53]. We can ascertain from this evaluation whether the trained model has effectively picked up pertinent patterns and is capable of producing precise predictions outside of the training and validation contexts.

The CNN model employed in the present dissertation, was trained on two combined public datasets. It is critical to note that each CNN model needs the image labeling to conform to a specific annotating format. Therefore, online datasets, even if they are pre-annotated, need to be adjusted to the annotating format in order to be used for the training process. The first dataset included closeup images of both historical and residential buildings, with defects present and annotated in the COCO JSON Format, that had to be converted [54] and the second dataset used was not labeled [55].

For the purpose of annotating the datasets correctly and then combining them for this dissertation, a computer vision dedicated platform was employed, RoboFlow. This framework improved the speed and convenience of the dataset creating process exponentially. Using RoboFlow the final dataset employed, was generated:

- Firstly, the labels of the formerly annotated images, which were annotated in a foreign format for the model used, are converted to the format employed by it (YOLOv8 PyTorch TXT).
- Then the unannotated data of the second set, are labeled by hand, using the labeling tool already imbedded in the platform, utilizing the bounding boxes technique.
- Subsequently the separate datasets are uploaded in the platform and seamlessly combined into the final dataset used to train the model.

After the completion of the aforementioned steps, a complete dataset is generated. The resulting set is segregated into the three segments, as mentioned above, training set, validation set and test set. This functionality is also embedded in the platform. The generated dataset has the following layout:



With each set including the images and their corresponding labels.

- The segment 5.3, covers briefly the RoboFlow platform, its uses, as well as its features.

In the next step of the dataset generation, as the study in Chapter 3 states, for the creation of the final dataset used to train the model, the defects have to be clustered. According to the authors, it makes more sense to combine two defects that are adjacent to one another and belong to the same defect category to create a wider bounding box for the evaluation area [3]. In order to determine if two defects should be grouped together and placed in the same evaluation area, a distance threshold should be established. Also in their own experience, different distance thresholds are adopted in different scenarios.

The code shown below is designed according to the pseudo-code shown in Fig. 14, in the third chapter, to merge the bounding boxes of each image in the dataset generated based on the localization rule. The distance threshold used is 30 pixels as it produced great results in the aforementioned study and also great results in the present dissertation.

Additionally, an overlapping rule is designed, in order to manage some cases, where overlapping of the bounding boxes is occurring and the distance rule cannot be employed. The overlapping threshold, after experimentation, is set at 0.2. That essentially means that if two annotation bounding boxes of two defects belonging to the same category overlap 20% or more, the bounding boxes will be merged to create a wider bounding box for the evaluation area. The code merges overlapping bounding boxes iteratively, checking the distance threshold for each bounding box in an image with the previously merged bounding box and saves the merged annotations in the same format.

```python
def merge_boxes(boxes, localization_rule):
    merged_boxes = []

    for box in boxes:
        should_merge = False
        for merged_box in merged_boxes:
            if localization_rule(box, merged_box):
                merged_box[1] = min(merged_box[1], box[1])
                merged_box[2] = min(merged_box[2], box[2])
                merged_box[3] = max(merged_box[3], box[3])
                merged_box[4] = max(merged_box[4], box[4])
                should_merge = True
                break

        if not should_merge:
            merged_boxes.append(box)

    return merged_boxes
```

```python
input_folder = 'U:\Downloads\Dipl.v2i.yolov8/test/labels'
output_folder = 'U:\Downloads\Dipl.v2i.yolov8/test/merged2'
distance_threshold = 30
os.makedirs(output_folder, exist_ok=True)

annotation_files = [f for f in os.listdir(input_folder) if f.endswith('.txt')]

for annotation_file in annotation_files:
    with open(os.path.join(input_folder, annotation_file), 'r') as file:
        lines = file.readlines()
        boxes = []

        for line in lines:
            line = line.strip().split()
            label = line[0]
            x_center, y_center, width, height = map(float, line[1:])
            x_min = int((x_center - width / 2) * 640)
            y_min = int((y_center - height / 2) * 640)
            x_max = int((x_center + width / 2) * 640)
            y_max = int((y_center + height / 2) * 640)
            boxes.append([label, x_min, y_min, x_max, y_max])

        merged_boxes = merge_boxes(boxes, lambda box1, box2: distance_rule(box1, box2, distance_threshold))
        merged_boxes = merge_boxes(merged_boxes, proximity_rule)

        output_file = os.path.join(output_folder, annotation_file)
        with open(output_file, 'w') as output:
            for box in merged_boxes:
                label, x_min, y_min, x_max, y_max = box
                x_center = (x_min + x_max) / 2 / 640.0
                y_center = (y_min + y_max) / 2 / 640.0
                width = (x_max - x_min) / 640.0
                height = (y_max - y_min) / 640.0
                output.write(f"{label} {x_center} {y_center} {width} {height}\n")
```

```python
# Overlap Rule
def proximity_rule(box1, box2):
    overlap_x = min(box1[3], box2[3]) - max(box1[1], box2[1])
    overlap_y = min(box1[4], box2[4]) - max(box1[2], box2[2])
    area1 = (box1[3] - box1[1]) * (box1[4] - box1[2])
    area2 = (box2[3] - box2[1]) * (box2[4] - box2[2])
    return (overlap_x > 0 and overlap_y > 0) and (overlap_x * overlap_y >= 0.2 * min(area1, area2))


# Distance-based rule
def distance_rule(box1, box2, distance_threshold):
    horizontal_distance = min(abs(box1[3] - box2[1]), abs(box2[3] - box1[1]))
    vertical_distance = min(abs(box1[4] - box2[2]), abs(box2[4] - box1[2]))

    return max(horizontal_distance, 0) <= distance_threshold or max(vertical_distance, 0) <= distance_threshold
```

Distance Rule:

The distance rule evaluates the proximity of bounding boxes in terms of the distance between their edges. Specifically, it calculates the horizontal and vertical distances between box edges and checks if either of these distances is less than or equal to a predefined distance threshold of 30 pixels. If the boxes are within this threshold, they are merged. This rule prioritizes spatial proximity and ensures that close objects are represented by a single bounding box.

Overlap (Proximity) Rule:

The overlap rule measures the degree of overlap between two bounding boxes. It computes the overlap in both horizontal and vertical dimensions and checks if the overlap area exceeds a certain threshold, 20%. If the boxes overlap sufficiently, they are merged. This rule captures cases where objects are partially or completely embedded within each other and need to be merged to create a more accurate representation.

The code follows the following workflow:

- It reads a set of YOLOv8 annotation files from the specified input folder.
- For each annotation file, it parses the lines to extract class labels and bounding box coordinates. These coordinates are converted from YOLOv8 format to absolute image pixel values.
- The distance-based rule is applied first to merge bounding boxes that are within the defined distance threshold.
- The overlap rule (proximity rule) is subsequently applied to the merged bounding boxes. It further merges boxes that exhibit a significant degree of overlap, capturing cases where objects intersect or partially overlap.
- The final merged bounding boxes are saved to new annotation files in the specified output folder, maintaining the YOLOv8 format.

The proposed methods effectiveness in reducing redundant bounding boxes while preserving object information, is demonstrated in Fig. 24. By first applying the distance-based rule and then the overlap rule, the code ensures a systematic and comprehensive merging process. This results in a more concise and accurate representation of objects in the annotation files.

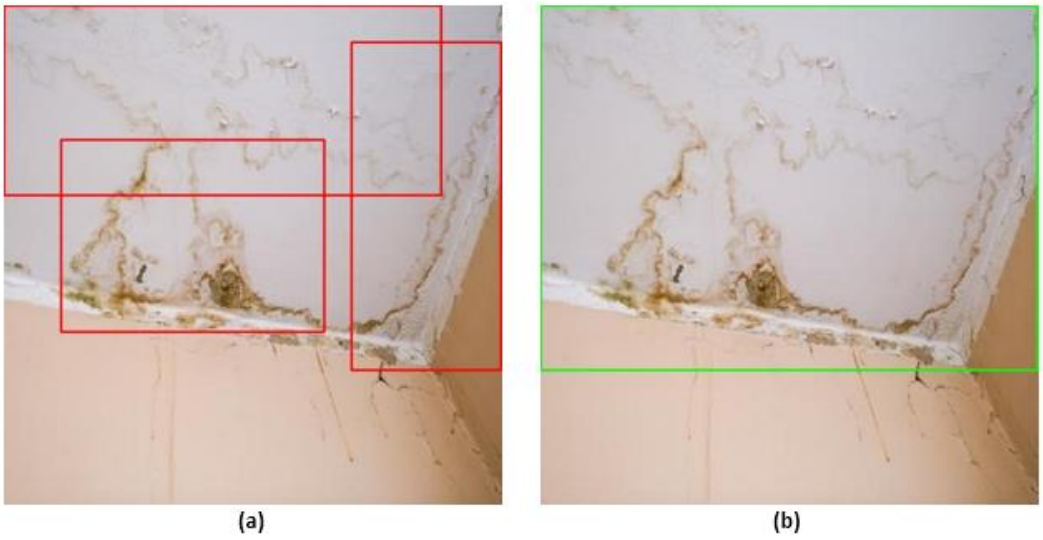


Figure 24: (a) Annotated Image with water seepage, (b) Annotated Image after applying clustering rules

In the final generated dataset, all images have rule-based annotations created automatically for the model's training, including labels and bounding boxes.

5.3 RoboFlow computer vision platform

RoboFlow is a computer vision platform that allows users to build computer vision models faster and more accurately through the provision of better data collection, preprocessing, and model training techniques. The computer vision developer framework allows users to upload custom datasets, draw annotations, modify image orientations, resize and modify image contrast and perform data augmentation. It encompasses a universal annotation conversion tool that allows users to upload and convert annotations from one format to another without having to write conversion scripts for custom object detection datasets. It supports a variety of popular annotation formats for the most widely used computer vision models, which it can be used to train [56].

Model libraries already supported by the framework include models such as EfficientNet, MobileNet, Yolo, TensorFlow, PyTorch, etc. Thereafter model deployment and visualization options are also available hence. Also, the entire workflow can be coordinated with teams within the framework.

Roboflow is used in a wide area of computer vision related applications, as it severely simplifies and streamlines the whole process needed to obtain a well-trained model specifically for the current need. In the present thesis it was primarily used as an annotator and it also provided an essential Google Colab notebook, partially used to train the complex model YOLOV8m on the large dataset needed to achieve the desirable level of accuracy and confidence.

5.4 LOD3 Generation

As a result of the previous segments, a prediction model is generated, with satisfying performance and accuracy. The next step of the process is to generate a 3D LOD model of a real-world historical building, utilizing the great efforts of the engineers in the study presented in Chapter 2 of the present thesis [22]. The data and code of the aforementioned study, are public, which is a major help in the efforts of the thesis.

Firstly, the project is installed, following the comprehensive and explanatory instructions, that the authors kindly provided. Next, the captured images, are inserted as instructed in the Meshroom Software, using SfM to create point clouds and camera projection matrices for each perspective of the building asset [57]. Consequently, the Polyfit algorithm processes the point cloud, grouping the points into planar primitives to produce candidate faces. These candidate faces are then selected using linear optimization to produce a LOD3 model of the building [58].

For the purposes of the present dissertation, the LOD (Level of Detail) representation of the building, offers a clear and objective image of the building's façade, not influenced by the complex background and foreground objects. The goal is to employ the generated façade images as a source for the model's prediction. In order to achieve this, the image is processed through masks, removing the doors and windows of the building, as well as the black borders generated by the given pipeline. Afterwards, the resulting clear façade image is divided into 320x320 pixel tiles, resulting in several close images of the façade, to be presented to the model for effective defect detection and classification.

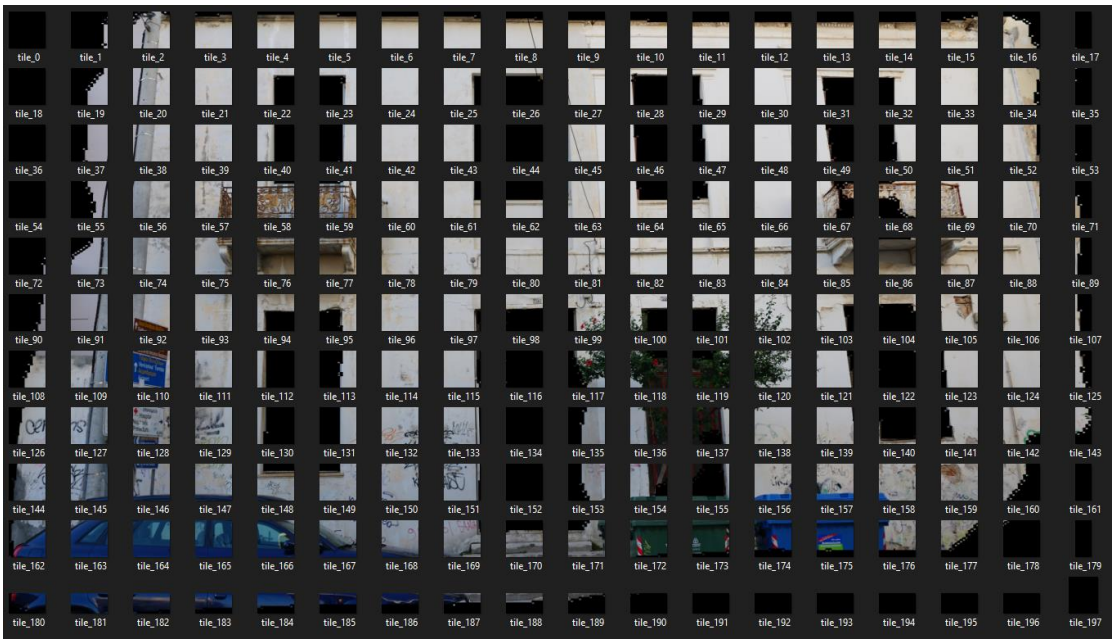
Additionally, a modification was implemented in the established pipeline, that can greatly assist the users of the present methodology. The pipeline was already detecting the openings of the building façade and identifying them utilizing a SDD (Small Data Driven) deep learning network, with the architecture U_Net16. The modification implemented, occurs after the opening detection process and counts the real openings found, displaying the number to the user. This information is invaluable, especially in real world civil engineering scenarios.

The following code segment, is used to implement the aforementioned functionality:

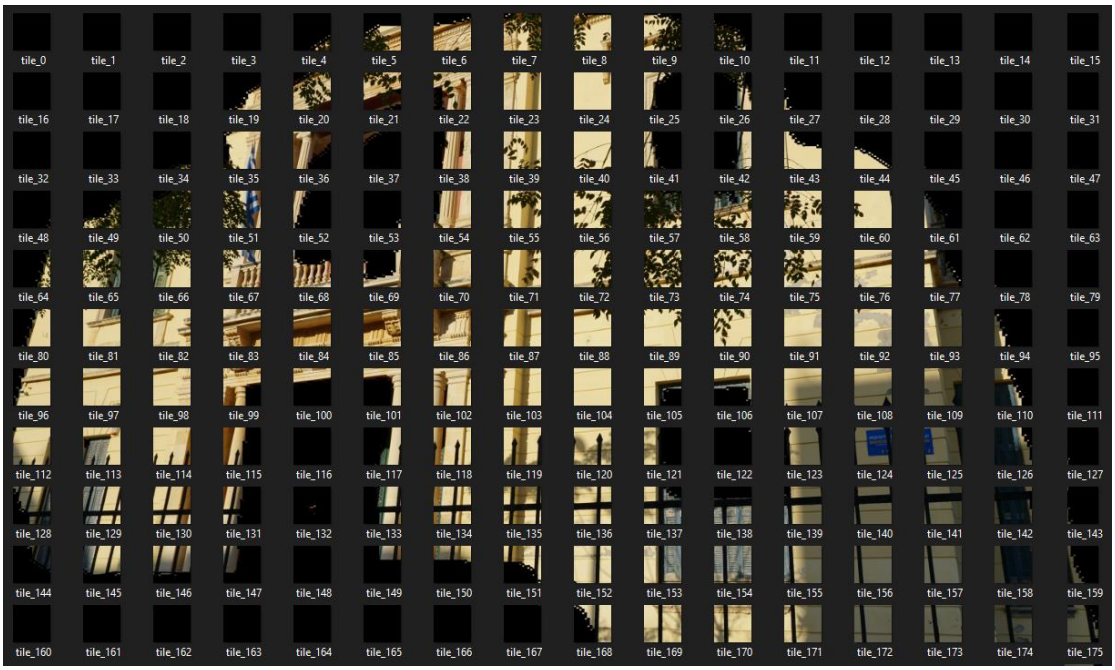
```
#Divide the croped image into exactly 320x320 pixel tiles in order to test the NN
tile_size = 320
tiles = []

for x in range(0, cropedImg.shape[0], tile_size):
    for y in range(0, cropedImg.shape[1], tile_size):
        tile = cropedImg[x:x + tile_size, y:y + tile_size]

        # Check if the tile size is less than 320x320
        if tile.shape[0] < tile_size or tile.shape[1] < tile_size:
            # Calculate padding
            pad_height = max(0, tile_size - tile.shape[0])
            pad_width = max(0, tile_size - tile.shape[1])

            # Pad the tile with zeros
            tile = np.pad(tile, ((0, pad_height), (0, pad_width), (0, 0)), mode='constant')
        tiles.append(tile)
```

- In summary, this code creates a directory to store the image tiles, divides the cropped image into 320x320 pixel tiles, handles padding for smaller tiles, and saves each tile as a separate image file, as presented in the following Figs.

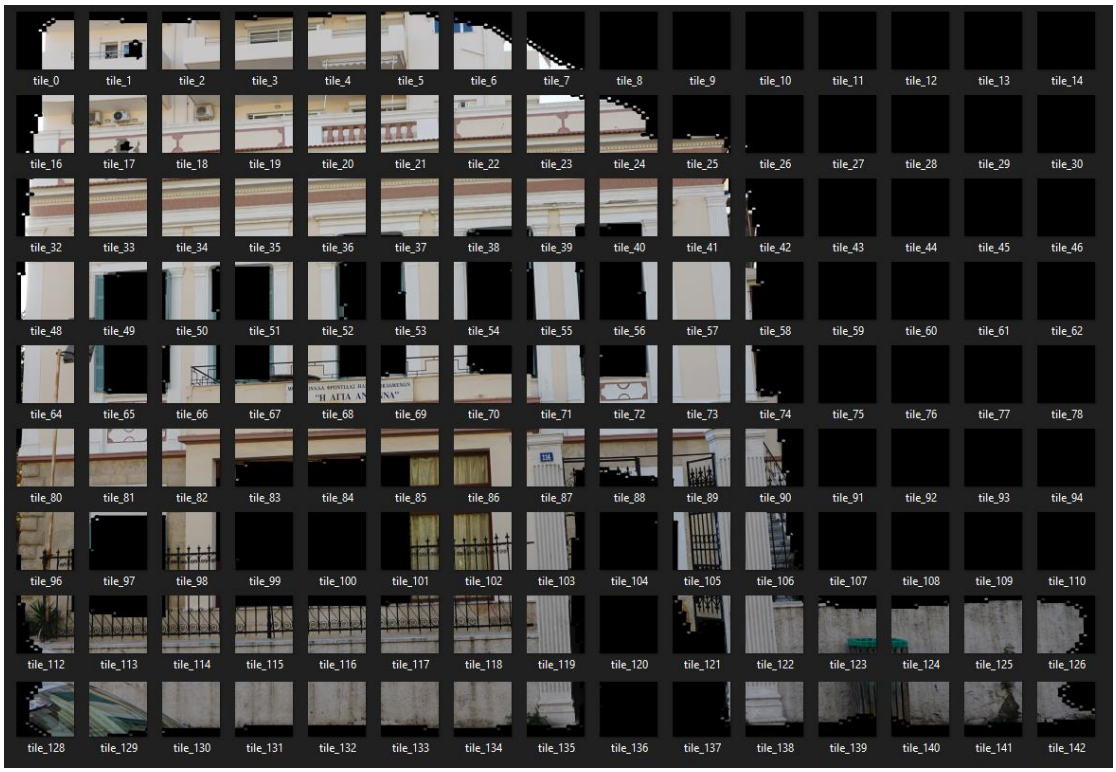
Building 1:



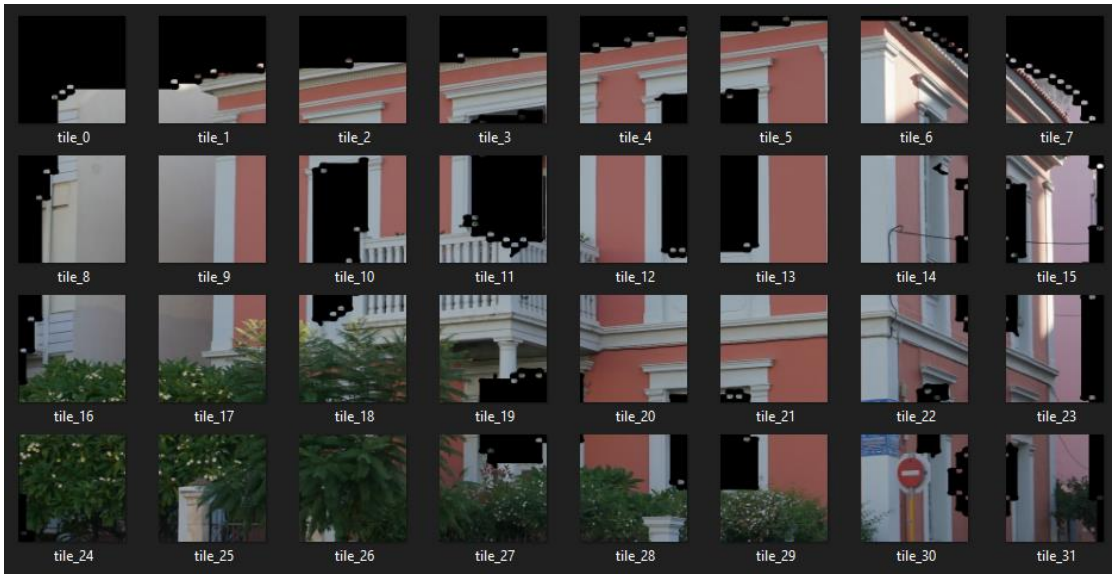
Building 2:



Building 3:



Building 4:



Building 5:

Building 6:


tile_14 tile_15 tile_16 tile_17 tile_18 tile_19 tile_20 tile_21 tile_22 tile_23 tile_24 tile_25 tile_26
tile_28 tile_29 tile_30 tile_31 tile_32 tile_33 tile_34 tile_35 tile_36 tile_37 tile_38 tile_39 tile_40
tile_42 tile_43 tile_44 tile_45 tile_46 tile_47 tile_48 tile_49 tile_50 tile_51 tile_52 tile_53 tile_54
tile_56 tile_57 tile_58 tile_59 tile_60 tile_61 tile_62 tile_63 tile_64 tile_65 tile_66 tile_67 tile_68
tile_70 tile_71 tile_72 tile_73 tile_74 tile_75 tile_76 tile_77 tile_78 tile_79 tile_80 tile_81 tile_82
tile_84 tile_85 tile_86 tile_87 tile_88 tile_89 tile_90 tile_91 tile_92 tile_93 tile_94 tile_95 tile_96
tile_98 tile_99 tile_100 tile_101 tile_102 tile_103 tile_104 tile_105 tile_106 tile_107 tile_108 tile_109 tile_110
tile_112 tile_113 tile_114 tile_115 tile_116 tile_117 tile_118 tile_119 tile_120 tile_121 tile_122 tile_123 tile_124

CHAPTER 6 – RESULTS AND DISCUSSION

6.1 Model Training and Validation

The ensuing step in employing the model, is training it using the specified dataset and hyperparameters. This way the model's parameters can be optimized so that it can accurately predict the classes and locations of objects in a real-world image. For the purposes of the present thesis, the model YOLOv8m is employed. Being the medium model in the YOLOv8 'family', it serves as a really accurate and precise neural network, while being relatively quick to train and predict.

The first very small portion of the training process, started in an environment with Python 3.10, Pytorch 2.0.1, CUDA 12.2 on a Windows 10 system, utilizing the CUDA cores of a GTX 1060 with 6GB of video memory and 16 GB of RAM. This part of the training took a significantly large amount of time to train the complex model YOLOV8m on the large dataset, to achieve the desirable level of accuracy and confidence. Therefore, for the purpose of improving the speed of the process, the training was resumed employing a Google Colab notebook, provided by the RoboFlow framework, on the CUDA cores of a much more powerful GPU, the T4 Tensor Core developed by Nvidia and 32 GB of RAM.

A total of 100 epochs were carried out training and at the end of them, the threshold of score for filtering out low-quality predictions was set at 0.5. After every training epoch was performed, a validation step was also employed in order to evaluate the performance of the model in unknown images and monitor its learning process. Finally, after the training process concluded, the YOLO model saved the following table of graphs (Fig. 25), that show the progress of the model during training.

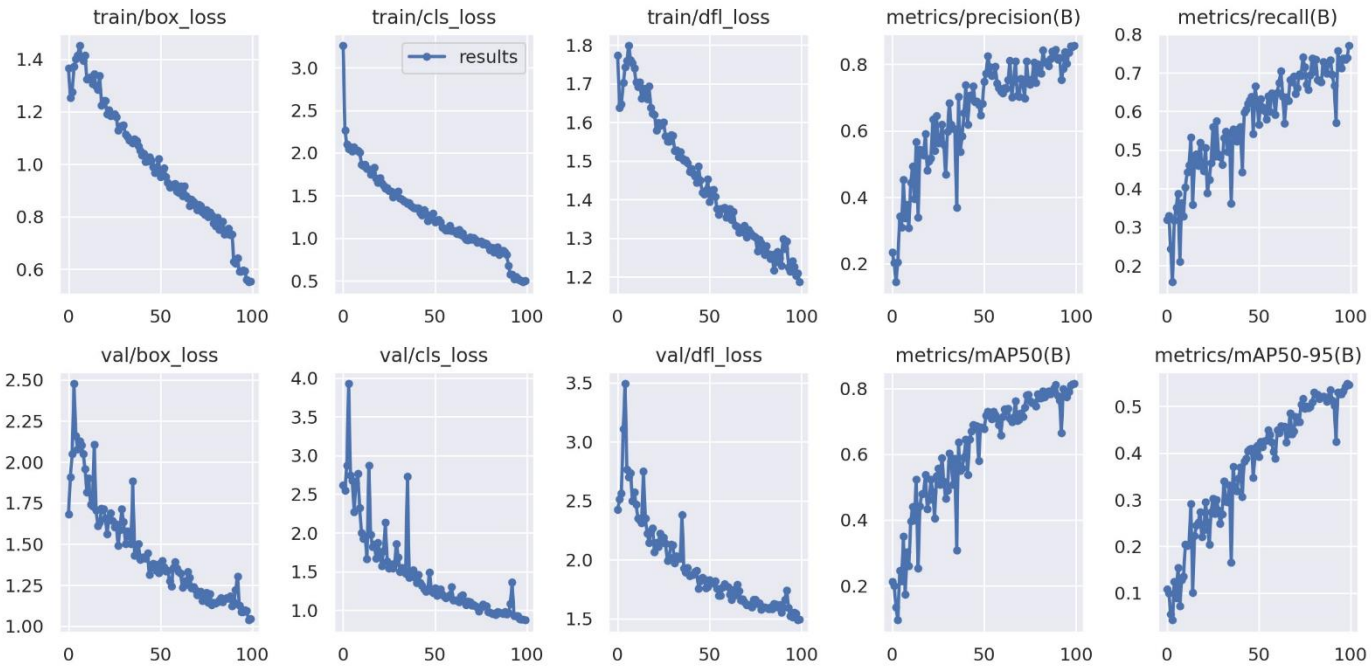


Figure 25: Performance Graphs of the model

As the graphs make apparent, the training process was successful in significantly improving the model's performance. The box, cls and dfl loss graphs show that the loss decreased with each epoch and the precision and recall graphs show an increase in the ability to identify and classify objects, as indicated by the mAP and mAP (50-95) metrics.

Next, the validation step was performed, presenting the model with unknown images in order to evaluate its overall performance through different metrics. The process generated scores for each class regarding the Precision and Recall of the model, as well as the mAP50 and mAP (50-95) mentioned in the introduction Chapter (Fig. 26).

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	446	724	0.708	0.651	0.704	0.449
crack	446	329	0.735	0.72	0.768	0.491
mold	446	78	0.686	0.692	0.699	0.418
peeling_paint	446	176	0.645	0.558	0.627	0.428
stairstep_crack	446	52	0.828	0.832	0.925	0.593
water_seepage	446	89	0.647	0.454	0.501	0.317

Figure 26: Model Validation scores

Additionally, graphs evaluating the model's performance are generated by the validation step and are presented in Fig. 27.

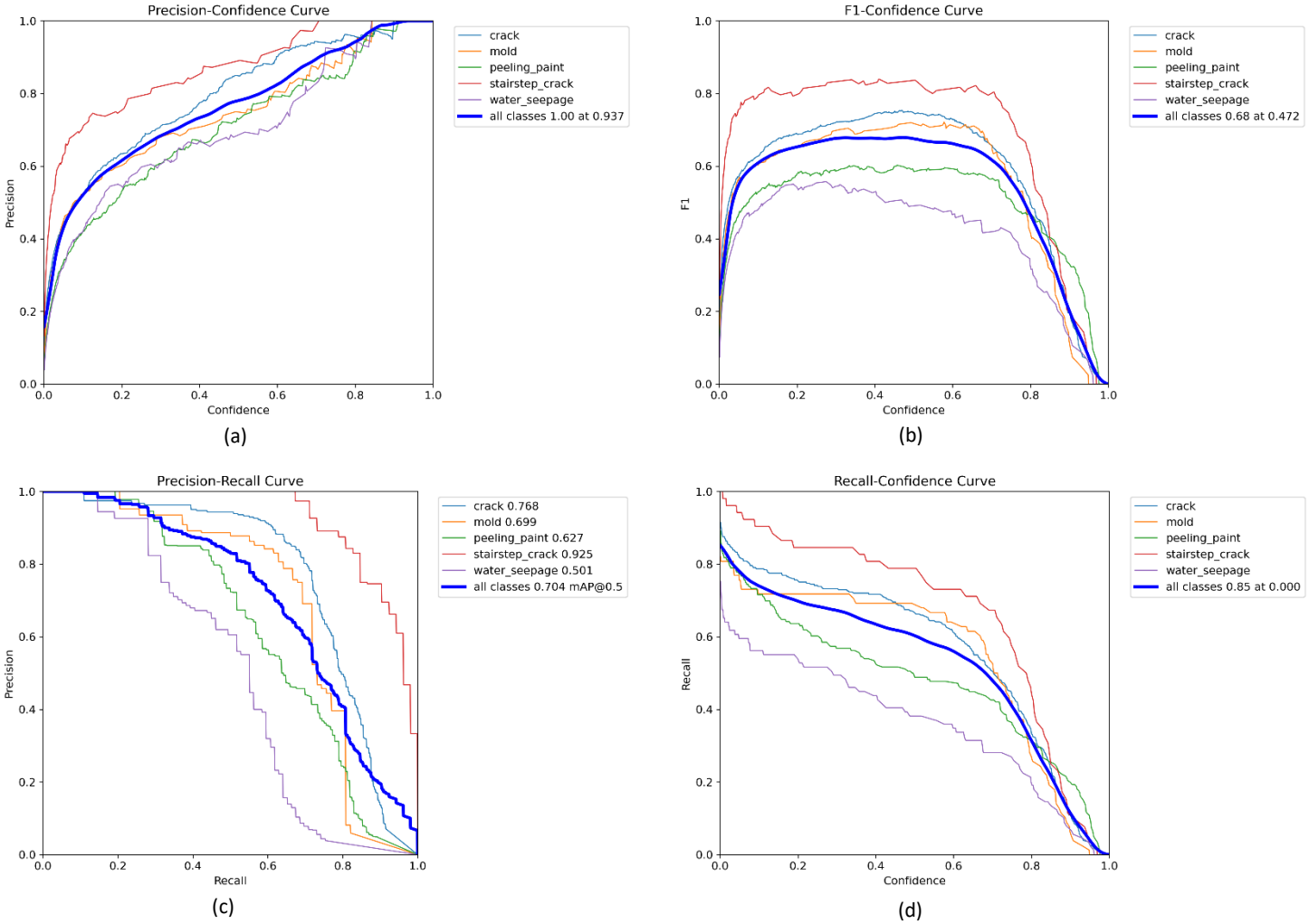


Figure 27: Validation Performance Graphs: (a): Precision-Confidence Graph. (b): F1-Confidence Graph. (c): Precision-Recall Graph. (d): Recall-Confidence Graph.

These graphs generated by the validation process, plot a different aforementioned metric on each of their axis and are essential in the evaluation of the model as presented below [47]:

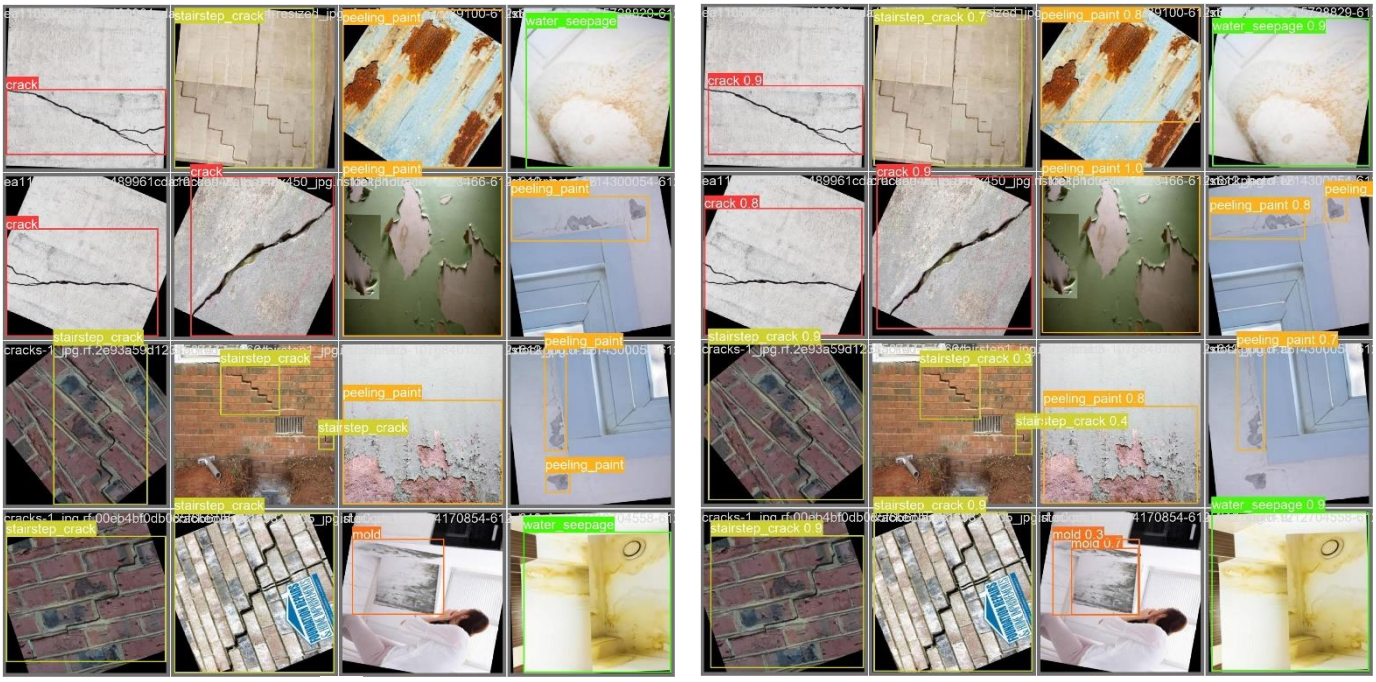
- **Precision-Confidence Curve**: A graphical representation of precision values at different thresholds. This curve helps in understanding how precision varies as the confidence increases.
- **F1 Score-Confidence Curve**: This curve represents the F1 score across various confidence thresholds. Interpreting this curve can offer insights into the model's balance between false positives and false negatives over different thresholds.

- **Precision-Recall Curve**: An integral visualization for any classification problem, this curve showcases the trade-offs between precision and recall at varied confidence thresholds. It becomes especially significant when dealing with imbalanced classes.
- **Recall-Confidence**: This graph illustrates how the recall values change across different thresholds.

The scores and graphs generated by the validation process show that the training of the model was successful in improving the accuracy and performance of the neural network. The mAP scores are especially satisfactory, when taking into account the limited nature of the datasets available for the training of Deep Neural Networks in Façade Defects Detection.

It is important to note that, as shown in the table, the defect classes in the dataset are not represented equally. This has to do with the frequency of each defect appearing in a building façade varying greatly from the another. This variance becomes more apparent in the stairstep crack class, having lower instance count than the others, resulting in improved scores but some possible overfitting, even though the results that follow show excellent performance in predicting this specific class. Another noteworthy observation, is that the model performed worse than expected in the water seepage defect class. The decline in performance can be attributed in the similarities between some defect classes as they appear in the real world, especially the mold, peeling paint and water seepage defects. These defects often appear alongside each other in building façades, or comprise an evolved state of each other. In the final Chapter of the present thesis, this problem is addressed in greater detail.

Lastly the test dataset is used to verify the model's accuracy. The test dataset is a set of labeled images that were not being included in the training or the validation process so that they are foreign to the model. Each image is presented to the CNN, and the output is compared to the actual annotated class label of the test image. The comparison is presented in Fig. 28 below.



(a) **(b)**

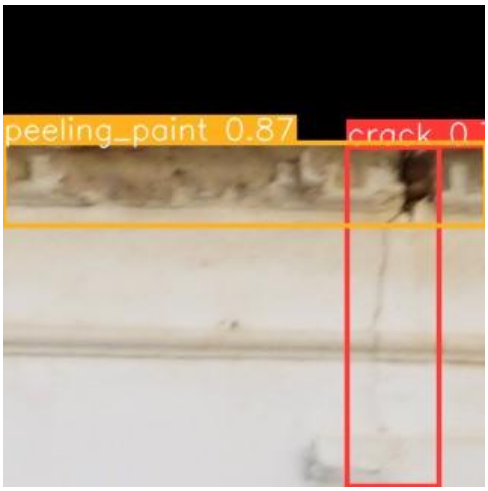
Figure 28: (a): Images as labeled in the dataset. (b): Images as predicted by the model

6.2 Implementation of the method in Historical Buildings in Chania

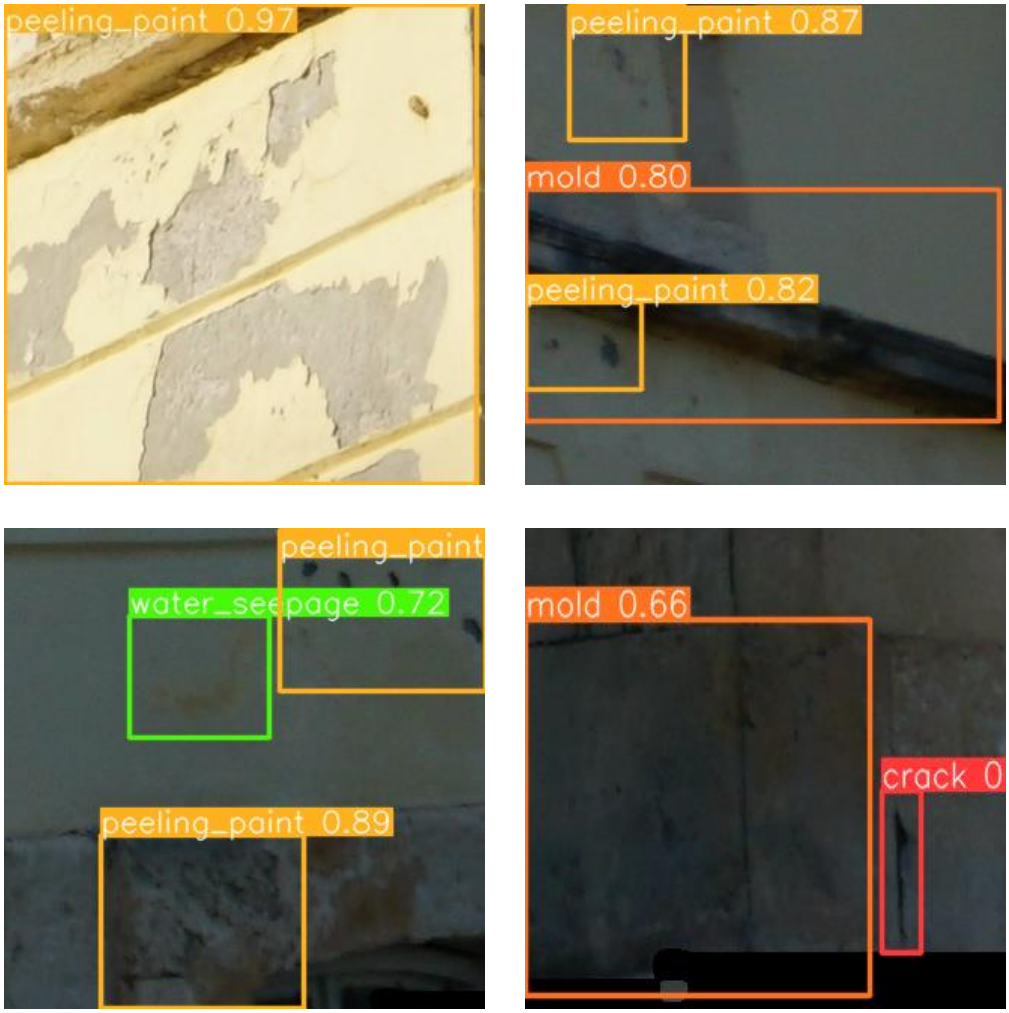
Case Studies:

After the building façades are divided into tiles, the resulting images are given as input to the Neural Network for façade defects prediction, with a confidence threshold of 0.5 for filtering out redundant predictions. The localization rule designed for the database creation, based on the pseudocode presented in Fig. 14 [3]. The method is effective in reducing redundant bounding boxes while preserving object information. A showcase of the most severe defect cases in conjunction with the predicted bounding boxes, for every building examined, is presented in the following figures.

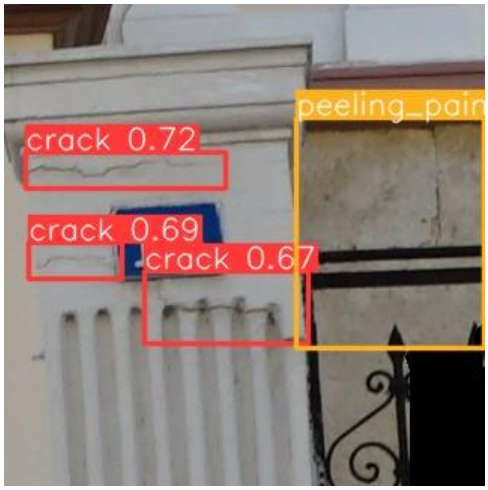
Case Study 1(Οδός Βενιζέλου):

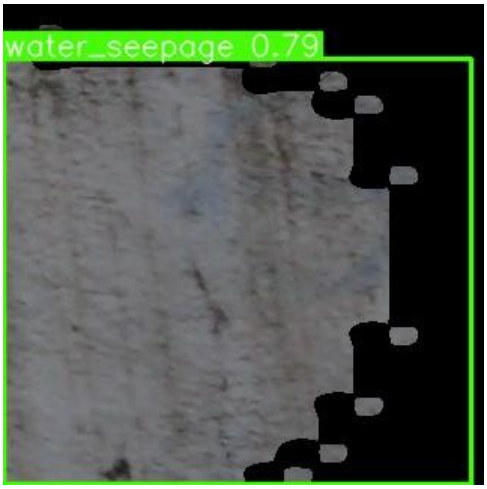


Case Study 2(Βίλα Κούνδουρου (Ηρ. Πολυτεχνείου)):

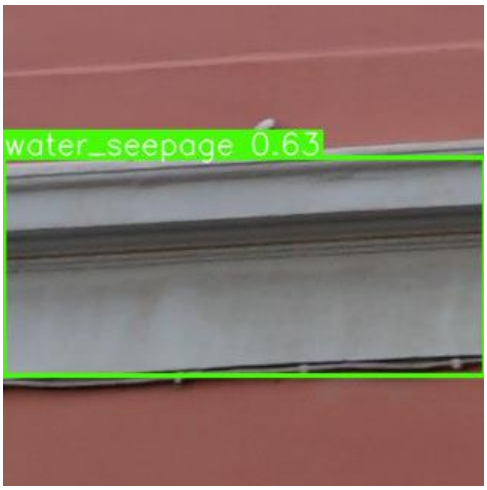


Case Study 3(Οίκος ευγηρίας Αγ. Άννα πρώην κλινική Μητσοτάκη (οδός Ακρωτηριου)):

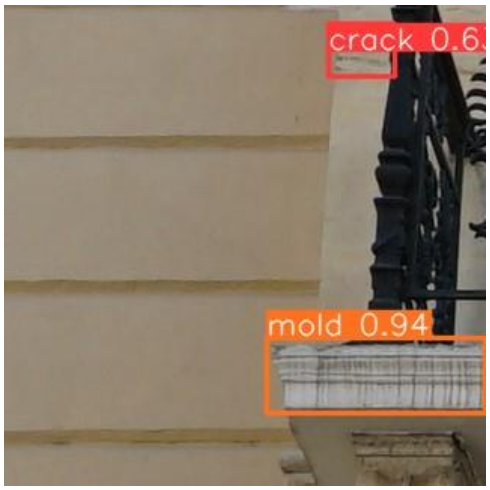




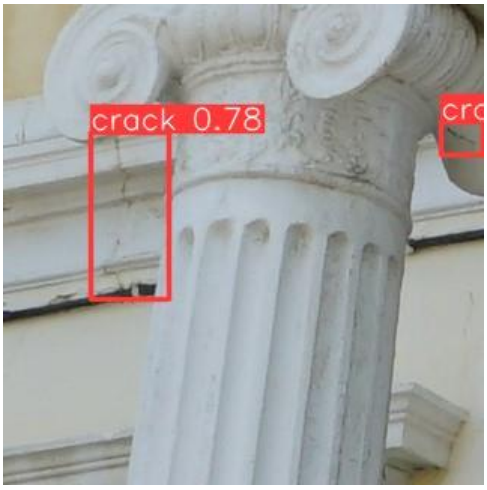
Case Study 4(Οδός Ηρ. Πολυτεχνείου διαστ. Κροκιδά):

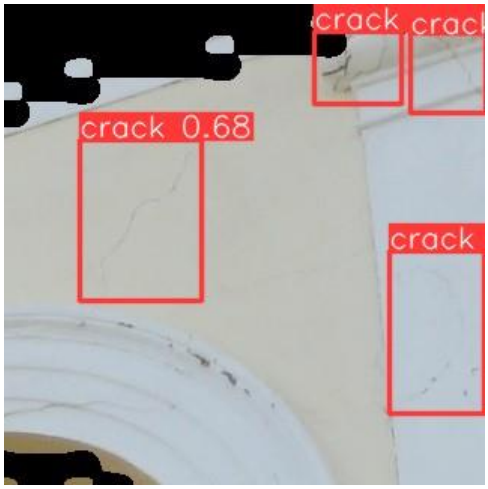


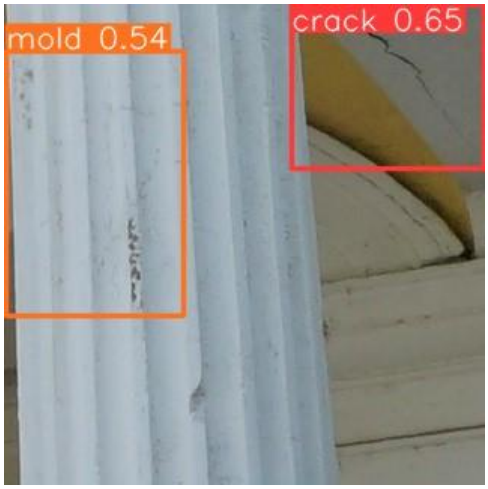
Case Study 5 Κτίριο Παπαδόπετρου (οδός Τζανακάκη):

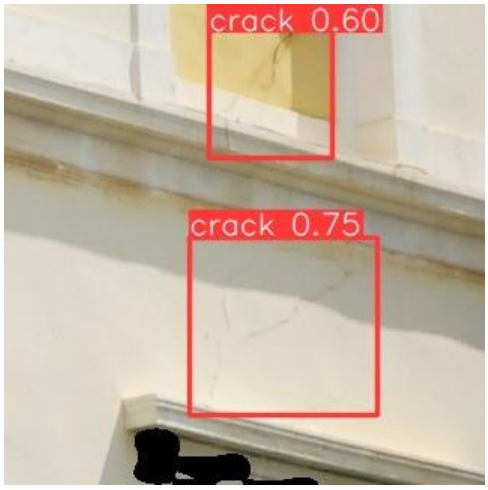


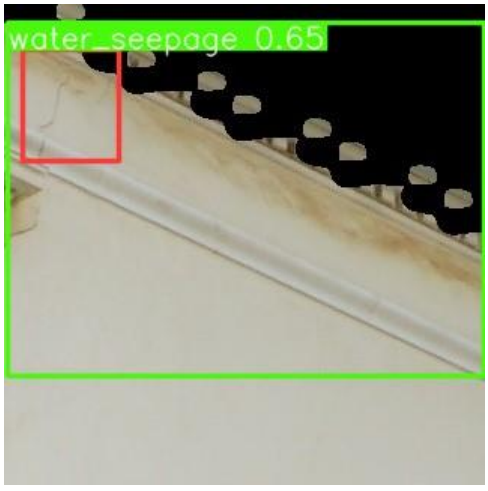
Case Study 6(Κτίριο ΤΕΕ/ΤΔΚ (οδός Νεάρχου)):











The buildings examined in the present thesis are real world historical buildings from the city of Chania Crete, Greece. The results of the prediction model make apparent that most of them, suffer a variety of defects in their façade, requiring further maintenance. It is important to note, that the selection of images presented in this paper, is curated, in order to best showcase the most severe defects present in each building, requiring urgent attention and maintenance. Additionally, for the purposes of the study, the variety of defects present in each image was taken into account, showcasing the methods performance and versatility in dealing with real world, complex, multiclass detections.

- Case Study 1(Οδός Βενιζέλου): The first building analyzed, shows several cases of paint peeling of the façade, as well as some water damage. Most importantly some severe, deep cracks are present requiring urgent attention. The model detected the defects accurately with great confidence, especially in the most severe cases.
- Case Study 2(Βίλα Κούνδουρου (Ηρ. Πολυτεχνείου)): As the images make apparent, the façade of the second building suffers from plenty of paint peeling off, as well as several cases of minor water seepage and mold, that the model detected.
- Case Study 3(Οίκος ευγηρίας Αγ. Άννα πρώην κλινική Μητσοτάκη (οδός Ακρωτηριου)): This building's façade was built utilizing a variety of architectural patterns, presenting a challenging case for the model. The neural network detected cracks, peeling paint and mold cases, even when they are present among these complex backgrounds with sufficient confidence and accuracy. As such, this case proves the ability of the model to perform in complex real-world environments.
- Case Study 4(Οδός Ηρ. Πολυτεχνείου διαστ. Κροκιδά): The fourth building examined showed little damage in its façade and was primarily void of defects, with the exemption of some water seepage in limited places.
- Case Study 5 Κτίριο Παπαδόπετρου (οδός Τζανακάκη): The neural network model performed with great accuracy in detecting peeling paint and slight mold in the façades of this building which were the most severe and common defects present.
- Case Study 6(Κτίριο ΤΕΕ/ΤΔΚ (οδός Νεάρχου)): In this particular building's façades, there is a great concentration of surface level cracks along with peeling paint. The model managed to detect most of the defects with satisfying accuracy, except for some cases where the defects are concentrated in a small area, or there are complex patterns present in the façade of the building.

An important result to note, is that, due to the complex patterns on several building façades, totally accurate detection and classification of each and every defect present, becomes difficult. Additionally, in images where there is a high concentration of different kinds of defects, the model often detects some of them. These problems become apparent from the resulting images and highlight that manual visual inspection of the buildings, may still be essential to effectively maintain the condition of their façades.

CHAPTER 7 – CONCLUSIONS AND FUTURE WORK

Multiple defects occur in various locations in actual buildings. To minimize the negative effects of these defects on the sustainability of buildings, there is a need for a technology that can efficiently monitor and classify those defects. Convolutional neural networks (CNN)-based image processing is a novel approach in the architecture heritage domain with proven high performance in detecting and classifying the defects.

The present thesis, aims to aid with the maintenance of real-world historical and heritage buildings, being evaluated in building from the city of Chania. These real-world cases usually exhibit building façades composed of various shapes, patterns and colors. This means that the background irregularities of image data are quite common, thus, reliably detecting defects becomes very challenging. The combination of the two methods analyzed, showed promising results regarding the accuracy and confidence of the predictions made by the model. Generating a LOD representation of the building provided complete and convenient access to every outer surface of the building, offering detailed images of its façades, rendering the complicated defects as clear as possible. Additionally, the essential rules that were implemented during the dataset creation and the prediction processes enhanced the model's performance, in accordance with ISO standards. The application performance of the YOLOv8 model used was verified through the collected data, and the average performance of each defect type was approximately 0.7 based on mAP50 metric. This is considered to be a meaningful result, justifying the possibility of multiple defect detection using a deep learning model. To achieve a satisfying performance in this application, the dataset used to train the model, was not built under refined conditions. Real-world data were used where irregularities always exist, and various image interferences occur.

In conclusion, this approach can allow for reducing failures and errors in judgment, rating, and evaluation in the architectural heritage field. Addressing several issues like the subjectivity of the results arising from human-centered inspection, time consumption and an increase in labor costs, defects in historical buildings façades, can be detected and classified effectively in the maintenance phase, and prompt action can be taken when required.

However, there are still some limitations in this method to consider. Firstly, obtaining high quality images from the façades of historical and heritage buildings can be difficult and resource intensive. Another approach can be utilized like the use of UAVs (unmanned aerial vehicle), or SAR interferometry (Synthetic Aperture Radar). Future work on the rules implemented in this study, may involve fine-tuning the distance and overlap thresholds to optimize the merging process further. Additionally, exploring machine learning techniques for

adaptive threshold selection could be a valuable extension. Also, it is necessary to expand the database for training and make it well-balanced across all of the defect categories, to increase the accuracy of the model proposed in this thesis. It is essential that the image data present in the dataset, are images of real-world buildings, to most effectively train the model for the application it is intended. Lastly, as the results make apparent, manual visual inspection of the building façades cannot be entirely replaced with deep learning techniques. Several defects can be layered on top of each other, and the lines distinguishing them can be blurred. There is a need for trained experts to access the results of the object detection models, utilizing them effectively and efficiently.

BIBLIOGRAPHY

[1] Kisu, L. Goopyo, H. Lee, S., Sanghyo, L. & Ha Young, K. (2019, August). *MultiDefectNet: Multi-Class Defect Detection of Building Façade Based on Deep Convolutional Neural Network.* Retrieved from MDPI: https://www.mdpi.com/2071-1050/12/22/9785

[2] González-Aguilera, D. (2016, January 21). *Multispectral Radiometric Analysis of Façades to Detect Pathologies from Active and Passive Remote Sensing.* Retrieved from ResearchGate: https://www.researchgate.net/publication/291530161_Multispectral_Radiometric_Analysis_of_Facades_to_Detect_Pathologies_from_Active_and_Passive_Remote_Sensing

[3] Guo, J. & Qian, W. (2021, November). *Evaluation-oriented façade defects detection using rule-based deep learning method.* Retrieved from ResearchGate: https://www.researchgate.net/publication/354045955_Evaluation-oriented_facade_defects_detection_using_rule-based_deep_learning_method

[4] Samhouri, M. Al-Arabiat, L. & Al-Atrash, F. (2022, June 12). *Prediction and measurement of damage to architectural heritages facades using convolutional neural networks.* Retrieved from SPRINGER LINK: https://doi.org/10.1007/s00521-022-07461-5

[5] H. Yusof, A. Ahmad Mustaffa, A.M.T. Abdullah, (2020). Historical Building Inspection using the Unmanned Aerial Vehicle (Uav), International Journal of Sustainable Construction Engineering and Technology, Universiti Tun Hussein Onn Malaysia, https://doi.org/10.30880/ijscet.2020.11.03.002

[6] Massonet, D. and Rabaute, T. 1993, Radar interferometry: limits and potential. IEEE Transactions on Geoscience and Remote Sensing, 31, 455–464

[7] Marechal, N., 1995, Tomographic formulation of interferometric SAR for terrain elevation mapping. IEEE Transactions on Geoscience and Remote Sensing, 33, 726–739.

[8] Moreira, J. Schwabisch,M. Fornaro,G. Lunari, R. Bamler, R. Just, D. Steinbrecher, U. Breit, H. Eineder, M. Franceschett, I. G. Geudtner, D. and Rinkel, H. 1995, X-SAR interferometry: first results. IEEE Transactions on Geoscience and Remote Sensing, 33, 950–956.

[9] Tarchi, D., Rudolf, H., Pieraccini, M., & Atzeni, C. (2010, November 25). *Remote monitoring of buildings using a ground-based SAR: Application to cultural heritage survey.* Retrieved from Taylor & Francis Online: https://www.tandfonline.com/doi/abs/10.1080/014311600750037561

[10] E.Z. Kordatos, D.A. Exarchos, C. Stavrakos, A. Moropoulou, T.E. Matikas, Infrared thermographic inspection of murals and characterization of degradation in historic monuments, Constr. Build. Mater. 48 (2013) 1261–1265, https://doi.org/10.1016/j.conbuildmat.2012.06.062.

[11] M. Molero, I. Segura, M.A.G. Izquierdo, J.V. Fuente, J.J. Anaya, Sand/cement ratio evaluation on mortar using neural networks and ultrasonic transmission inspection, Ultrasonics 49 (2009) 231–237, https://doi.org/10.1016/j.ultras.2008.08.006.

[12] A. Carpinteri, G. Lacidogna, Damage monitoring of a historical masonry building by the acoustic emission technique, Mater. Struct. 39 (2006) 161–167, https://doi.org/10.1617/s11527-005-9043-2.

[13] O. Sengul, Use of electrical resistivity as an indicator for durability, Constr. Build. Mater. 73 (2014) 434–441, https://doi.org/10.1016/j. conbuildmat.2014.09.077.

[14] Maurício, M. R., Enrico, B. G., Letícia, A. S., Yuri, d. S., Eneida, A., & Renan, P. S. (2022, June). *Infrared thermal imaging to inspect pathologies on façades of historical buildings: A case study on the Municipal Market of São Paulo, Brazil.* Retrieved from ScienceDirect: https://doi.org/10.1016/j.cscm.2022.e01122

[15] Al-Kaff, A., Moreno, F.M., San José, L.J., García, F., Martín, D., de la Escalera, A., Nieva, A. and Garcéa, J.L.M., 2017, April. Vbii-uav: vision-based infrastructure inspection-uav. In World Conference on Information Systems and Technologies

[16] Banaszek, A., Banaszek, S. and Cellmer, A., (2017). Possibilities of use of UAVs for technical inspection of buildings and constructions. In IOP Conference Series: Earth and Environmental Science (Vol. 95, No. 3, p. 032001). IOP Publishing

[17] Mader, D., Blaskow, R., Westfeld, P., & Weller, C. (2016). Potential of Uav-Based Laser Scanner and Multispectral Camera Data in Building Inspection. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, 41

[18] LERMA, J. L., RUIZ, L. Á., & BUCHÓN, F. (2000). *APPLICATION OF SPECTRAL AND TEXTURAL CLASSIFICATIONS TO RECOGNIZE.* Retrieved from isprs: https://www.isprs.org/proceedings/xxxiii/congress/part5/480_xxxiii-part5.pdf

[19] F. Yamazaki, W. Liu, Remote sensing technologies for post-earthquake dam- age assessment: A case study on the 2016 Kumamoto earthquake, in: ASIA Conference on Earthquake Engineering, (6ACEE) 2016, pp. 22–24, https://www.researchgate.net/publication/307351403.

[20] Dore, C., Murphy, M., (2012), Integration of Historic Building Information Modeling and 3D GIS for Recording and Managing Cultural Heritage Sites, 18th International Conference on Virtual Systems and Multimedia: "Virtual Systems in the Information Society", 2-5 September, 2012, Milan, Italy, pp. 1433–1444, doi:10.21427/e7sy-rt81

[21] M. Murphy, E. McGovern, S. Pavia, Historic building information modelling- adding intelligence to laser and image-based surveys, Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci. 38 (5) (2011) W16, XXXVIII (March) (2011) 2–4., doi: 10.1016/j.isprsjprs.2012.11.006

[22] Pantoja, R. B. Beyer, K. & Achanta, R. (2023, June). *Damage-augmented digital twins towards the automated inspection of buildings.* Retrieved from ResearchGate: https://www.researchgate.net/publication/369559295_Damage-augmented_digital_twins_towards_the_automated_inspection_of_buildings

[23] Rainieri, C., Rosati, I. Cieri, L. Fabbrocino, G. (2023). Development of the Digital Twin of a Historical Structure for SHM Purposes. In: Rizzo, P., Milazzo, A. (eds) European Workshop on Structural Health Monitoring. EWSHM 2022. Lecture Notes in Civil Engineering, vol 254. Springer, Cham., pp. 639–646, https://doi.org/10.1007/978-3-031-07258-1_64.

[24] G. Angjeliu, D. Coronelli, G. Cardani, Development of the simulation model for Digital Twin applications in historical masonry buildings: The integration between numerical and experimental reality, Comput. Struct. 238 (2020) 106282, http://dx.doi.org/10.1016/j.compstruc.2020.106282.

[25] A. Shabani, M. Skamantzari, S. Tapinaki, A. Georgopoulos, V. Plevris, M. Kioumarsi, 3D simulation models for developing digital twins of heritage structures: Challenges and strategies, Procedia Struct. Integr. 37 (C) (2021) 314–320, http://doi.org/10.1016/j.prostr.2022.01.090.

[26] Stepinac, M. Lulić, L. Ožić, K. (2022). The Role of UAV and Laser Scanners in the Post-earthquake Assessment of Heritage Buildings After the 2020 Earthquakes in Croatia. In: Osman, A., Moropoulou, A. (eds) Advanced Nondestructive and Structural Techniques for Diagnosis, Redesign and Health Monitoring for the Preservation of Cultural Heritage. Springer Proceedings in Materials, vol 16. Springer, Cham. https://doi.org/10.1007/978-3-031-03795-5_3

[27] N.M. Levine, B.F. Spencer, Post-earthquake building evaluation using UAVs: A BIM-based digital twin framework, Sensors 22 (3) (2022), https://doi.org/10.3390/s22030873.

[28] Guo, J. Wang, Q. Yiting, L. & Pengkun, L. (2020, June 10). *Façade defects classification from imbalanced dataset using meta learning-based convolutional neural network.* Retrieved from WILEY Online Library: https://doi.org/10.1111/mice.12578

[29] Gupta, J. Pathak, S. & Kumar, G. (2022). *Deep Learning (CNN) and Transfer Learning: A.* Retrieved from IOPScience: https://iopscience.iop.org/article/10.1088/1742-6596/2273/1/012029

[30] Sharma, P. (2022, March 1). *Basic Introduction to Convolutional Neural Network in Deep Learning.* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-convolutional-neural-network-in-deep-learning/

[31] Ali, L. N. K. Valappil, D. N. A. Kareem, M. J. John, and H. Al Jassmi. 2019. “Pavement crack detection and localization using convolutional neural networks (CNNs).” In Proc., 2019 Int. Conf. on Digitization (ICD), 217–221. New York: IEEE.

[32] Alipour, M., D. K. Harris, and G. R. Miller. 2019. “Robust pixel-level crack detection using deep fully convolutional neural networks.” J. Comput. Civ. Eng. 33 (6): 04019040. https://doi.org/10.1061/(ASCE)CP.1943 -5487.0000854

[33] Yuhan, J. Sisi, H. & Yong, B. (2021, September). *Building and Infrastructure Defect Detection and Visualization Using Drone and Deep Learning Technologies.* Retrieved from ResearchGate:

https://www.researchgate.net/publication/354837699_Building_and_Infrastructure_Defect_Detection_and_Visualization_Using_Drone_and_Deep_Learning_Technologies

[34] Li, D. Cong, A. Guo, S. Sewer damage detection from imbalanced CCTV inspection data using deep convolutional neural networks with hierarchical classification. Autom. Constr. 2019, 101, 199–208. https://doi.org/10.1016/j.autcon.2019.01.017

[35] V. Hoskere, Y. Narazaki, T.A. Hoang, B.F. Spencer, Towards automated post- earthquake inspections with deep learning-based condition-aware models, 2018, doi: arXiv:1809.09195

[36] Dung, C.V. Anh, L.D. Autonomous concrete crack detection using deep fully convolutional neural network. Autom. Constr. 2019, 99, 52–58. https://doi.org/10.1016/j.autcon.2018.11.028

[37] Yang, Q. Shi, W. Chen, J. Lin, W. Deep convolution neural network-based transfer learning method for civil infrastructure crack detection. Autom. Constr. 2020, 116, 103199. https://doi.org/10.1016/j.autcon.2020.103199

[38] Yin, X.; Chen, Y.; Bouferguene, A.; Zaman, H.; Al-Hussein, M.; Kurach, L. A deep learning-based framework for an automated defect detection system for sewer pipes. Autom. Constr. 2020, 109, 102967. https://doi.org/10.1016/j.autcon.2019.102967

[39] Shen, H.-K.; Chen, P.-H.; Chang, L.-M. Automated steel bridge coating rust defect recognition method based on color and texture feature. Autom. Constr. 2013, 31, 338–356. https://doi.org/10.1016/j.autcon.2012.11.003

[40] Cha, Y.-J.; Buyukozturk, O. Structural Damage Detection Using Modal Strain Energy and Hybrid Multiobjective Optimization. Comput. Aided Civ. Infrastruct. Eng. 2015, 30, 347–358. https://doi.org/10.1111/mice.12122

[41] Y. Xue and Y. Li. (2018). A Fast Detection Method via Region-Based Fully Convolutional Neural Networks for Shield Tunnel Lining Defects. Computer-Aided Civil and Infrastructure Engineering, 33(8), pp.638-654. https://doi.org/10.1111/mice.12367

[42] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal. (2017). Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. Construction and Building Materials, 157(C), pp.322-330. https://doi.org/10.1016/j.conbuildmat.2017.09.110

[43] Y. Gao and K. M. Mosalam. (2018). Deep Transfer Learning for Image-Based Structural Damage Recognition. Computer-Aided Civil and Infrastructure Engineering, 33(9), pp.748-768. https://doi.org/10.1111/mice.12363

[44] S. Jiang and J. Zhang. (2020). Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system. Computer-Aided Civil and Infrastructure Engineering, 35(6), pp.549-564. https://doi.org/10.1111/mice.1251

[45] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata. (2018). Road Damage Detection and Classification Using Deep Neural Networks with Smartphone

Images: Road damage detection and classification. Computer-Aided Civil and Infrastructure Engineering, 33(12), pp.1127-1141. https://doi.org/10.1111/mice.12387

[46] Shah D. (2023, May 4). Top Performance Metrics in Machine Learning: A Comprehensive Guide. retrieved from V7labs: https://www.v7labs.com/blog/performance-metrics-in-machine-learning

[47] Juan, T. & Diana, C.-E. (2023, 6 12). *A COMPREHENSIVE REVIEW OF YOLO: FROM YOLOV1 AND BEYOND.* Retrieved from CORNELL UNIVERSITY: arXiv:2304.00501

[48] Mukherjee S. YOLOv8: Pioneering Breakthroughs in Object Detection Technology. Retrieved from https://blog.paperspace.com/yolov8-a-revolutionary-advancement-in-object-detection-2/

[49] Solawetz, J. (2023, January 11) What is YOLOv8? The Ultimate Guide.Retrieved from Roboflow: https://blog.roboflow.com/whats-new-in-yolov8/

[50] Jocher, G. Chaurasia, A. & Qiu, J. (2023, January 10). *YOLO by Ultralytics.* Retrieved from GitHub: https://github.com/ultralytics/ultralytics

[51] McGonagle, J. Shaikouski, G. Williams, C., Hsu, A. Khlm, J. & Miller, A. (n.d.). *Backpropagation.* Retrieved from Brilliant: https://brilliant.org/wiki/backpropagation.

[52] Akruti, A. (2023, June 13). *Training, Validation, Test* Split *for Machine Learning Datasets.* Retrieved from Encord: https://encord.com/blog/train-val-test-split/

[53] Builddef2. (2023). *Building defect on walls Computer Vision Project*. Retrieved from RoboFlow: https://universe.roboflow.com/builddef2/building-defect-on-walls/dataset/4

[54] Elhariri, E. El-Bendary, N. & Taie, S. (2020). *Historical_Building_Crack_2019*. Retrieved from Mendeley Data: https://data.mendeley.com/datasets/xfk99kpmj9/1

[55] Theophilus S. (2021, September 3). Roboflow: Converting Annotations for Object Detection. retrieved from Medium: https://medium.com/analytics-vidhya/converting-annotations-for-object-detection-using-roboflow-5d0760bd5871

[56] Griwodz, C. Gasparini, S. Calvet, L. Gurdjos, P. Castan, F. Maujean, B. . . . De Lillo, G. (2021, September 21). *AliceVision Meshroom: An open-source 3D reconstruction pipeline.* Retrieved from ACM Digital Library: https://dl.acm.org/doi/10.1145/3458305.3478443

[57] L. Nan, P. Wonka, PolyFit: Polygonal surface reconstruction from point clouds, in: Proceedings of the IEEE International Conference on Computer Vision, 2017-Octob, 2017, pp. 2372-2380, http://dx.doi.org/10.1109/ICCV.2017.258.