UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN INGEGNERIA CIVILE E ARCHITETTURA
XXXVI CICLO

A training dataset photogrammetric simulation framework for neural networks rockfall identification

Sviluppo di un simulatore fotogrammetrico per l'addestramento di reti neurali per l'identificazione di eventi di caduta massi

Coordinatore: PROF. ING. ANDREA SPAGNOLI
Tutor: PROF. ING. RICCARDO RONCELLA
Co-Tutor: PROF. ING. GIANFRANCO FORLANI

Dottorando: ING. PIETRO GARIERI

ANNI ACCADEMICI: 2020/2021 – 2022/2023

Acknowledgements

I would like to express my deepest gratitude to Prof. Roncella and Prof. Forlani for their priceless and invaluable guidance and fundamental advice during the course of these three years of doctoral studies.
A special thanks to Prof. Bruno for supporting me with her friendship and fortitude during the most difficult moments.
Thanks to my colleagues and friends who have accompanied me during these three years.
To my family for supporting me from the beginning. Without you nothing would have been possible. You are my inner strength.
To Emanuela, for being my life partner and my everything.

SUMMARY

List of figures

1 Chapter 1 - Introduction

1.1 Background and motivations

In the last three decades photogrammetry has greatly benefited from advancements in computer vision, to the extent that the boundary between the two disciplines has become increasingly blurred. During the early 1990s, computer vision developed Structure from Motion (SfM) techniques [1] [2]. SfM refers to the automated reconstruction of a three-dimensional object from images taken from multiple perspectives, without prior knowledge of the interior or exterior orientation parameters of the cameras [3]. SfM brought a great improvement to automatic photogrammetric block orientation and was soon adopted as standard in photogrammetric software packages, though the final estimation of the interior and exterior orientation parameters is still performed by the bundle block adjustment. The high degree of automation offered by computer vision algorithms, aided by the increasing processing power of modern computers, brought photogrammetry to a prominent position in various applications in civil and environmental engineering that require extensive use of digital metric data. Close-range photogrammetry has emerged as a powerful and efficient approach for 3D topographic modeling [4], as the reliability of data obtained from overlapping stereo pairs or larger image blocks have significantly improved [5] [6] also thanks to methodologies for the analytical calibration of non-metric cameras [7] [8].

The versatility of photogrammetry has long been widely recognized in the engineering field, and the geo-technical field was no exception. [9].Within the framework of geo-technical engineering, a particularly challenging environment are quarries. Since the 2000, with the introduction of rock stability analysis based on high resolution DTM [10] [11], photogrammetry and laser scanning have competed to become the most efficient technology to provide such DTMs [12]. In quarries, ensuring the safety and stability of walls and ceilings is a critical task; though the above-mentioned stability analysis aims to a global assessment of the rock face, this is not the only relevant information needed to ensure safety. In particular, rock collapses [13] [14] [15], [16] of various dimensions may pose a significant threat to both workers in the quarry and the surrounding environment.

In dynamic workplace environments such as quarries and civil construction sites, workers are exposed to a wide variety of risks. Surveying and monitoring activities are necessary to provide up-to-date data to design appropriate safety measures. Specifically, rockfalls are regarded as primary factors of human injury, fatalities, and damage to both personnel and equipment in quarries, as excavation walls often exhibit limited geo-mechanical stability as they are continuously evolving due to excavation processes. Moreover, rock excavation techniques employ highly intrusive methods such as blasting, which can potentially result in uncontrolled debris detachments, though providing cost reduction and improved efficiency [17]. Monitoring rock mass stability allows developing predictive models that can facilitate comprehensive risk assessments, supplying critical information [18].

In this regard, photogrammetry and LiDAR have demonstrated to be a powerful investigation tools [19] [20] .

Fixed stereo monitoring systems have been increasingly used for stability analysis in quarry environments [21]. They consist of two or more cameras installed at fixed locations in the quarry to capture stereo images of the area of interest. Compared to other methodologies (TLS for instance), they have the advantage of being a lower-cost solution but still entail other difficulties. The main issues (which is nonetheless shared with TLS) regards the reliability of autonomous operation [22]. Another is the careful geometric network design necessary to obtain reliable data, as the image-based 3D reconstruction cannot enjoy the same level of redundancy of observations of aerial blocks. In particular, ensuring an optimal base-to-distance ratio and the convergence of the optical axes (to ensure good overlap between images) is one of the most relevant points in designing a photogrammetric fixed monitoring system [22], [23]. Though to maintain stability of optical parameters (camera interior orientation parameters) has proven to be challenging, fixed systems have emerged as a viable option in the geoscience monitoring field, with a growing number of examples provided in different publications in the recent years [24] [25]. By comparing images and 3D Digital Surface Models (DSM) over time, changes in the quarry environment, such as rock collapses, can be detected and quantified [26].

However, detecting rockfalls from images or 3D models involves substantial effort, typically by skilled operators. Some degree of subjective evaluation of situations can be expected, given the constantly shifting and frequently unpredictable characteristics of collapses, which complicates their identification. Another concern arises from measurement noise, which can further hinder the accurate recognition of such phenomena in surveys spanning multiple time periods.

On the purpose of automating such tasks, neural networks, a type of machine learning algorithm, are considered to have the adequate potential to address this issue. They are capable of processing large amounts of data and identifying patterns that can be used to predict future events [27].

In the context of rockfall hazard assessment, neural networks can help process and analyze data from various sources, to predict potential rock collapses [28]. This predictive capability can aid in the development of management strategies to mitigate the risks associated with rockfall hazards.

However, the performance of neural networks in rockfall identification largely depends on the availability of high-quality, diverse, and representative datasets. One of the major challenges associated with neural networks is the (sometime) limited availability of data for the training and evaluation phases. Collecting training data is often a difficult and time-consuming task, as all data of interest must be accurately labeled. Labeling data is a time-consuming and costly process that requires experts and is prone to errors, especially in complex identification tasks. A recent study illustrated the development of a dataset for rock collapse classification in point clouds, which leverages five years of LiDAR acquired data for a single rock face in Canada [29]. This exemplifies the considerable time and effort necessary for manual classification and labeling of training datasets in specialized domains, where human error may even notably impact the accuracy of the expected results.

To overcome these challenges, researchers have explored alternative methods for generating synthetic training data that are able to partially or fully replicate the phenomenon of interest [30]. The main advantage of simulated datasets is the ability to create different scenarios within a controlled environment, which is essential for validating and optimizing neural network models [31]. In addition, simulated datasets can be generated in large quantities, overcoming the limitations of real data, which can be expensive or difficult to obtain in certain circumstances [32] [33].

The process of simulating data sets to detect changes, nonetheless, presents numerous challenges. A major one is to ensure that the synthetic data are very similar to the real data in terms of complexity and diversity to ensure a proper generalization of the models [34] [35] [36] [37]. This requires the use of appropriate simulation methods and techniques able to mimic the characteristics of real-world phenomena as well as a deep analysis of the environment to be simulated. On the other hand, it is indeed important to balance the diversity of the training datasets in order to properly allow the network to generalize the given problem.

Various techniques have been proposed for simulating neural network datasets. As an example, Variational Autoencoders and Generative Networks have been used to generate realistic and diverse synthetic data [38] [39]: CycleGAN, a type of generative model, has been used to generate synthetic thermal images for remote sensing to train neural networks [40].

The simulation of data in the photogrammetric domain needs to consider the different sources of noise which may affect the accuracy and the reliability of the surveys, such as image acquisition, camera calibration, and image matching [41]. Noise may arise from different factors such as atmospheric conditions, sensor noise, and the presence of operating machines such as diggers and trucks, which may produce unexpected vibrations of the camera stations [42] in fixed photogrammetric systems operating in open pits. Furthermore, camera calibration errors, including inaccurate lens distortion models and interior orientation parameters estimation, can contribute to errors in the 3D reconstruction. Repetitive textures, occlusions, and illumination changes are other factors that can introduce noise during the image matching and point cloud generation stages [43].

This dissertation is dedicated to the development and implementation of a photogrammetric simulation framework for generating training data for neural networks applied to detect and localize rockfall events at multiple scales on excavation walls.

The collapse identification and classification is founded upon the analysis of raster difference maps depicting the temporal variations in the 3D model of the wall derived from photogrammetric surveys. The simulated data for the neural network, therefore, must accurately represent real-world samples and its generation is a critical aspect from both a theoretical and practical standpoint.

To ensure the effectiveness of the training dataset, the simulation must consider the actual conditions under which the photogrammetric monitoring system operates on the field. Consequently, it is essential to accurately represent the noise sources of the photogrammetric surveys.

This project is carried out in collaboration with the Centre for Geotechnical Science and Engineering at the University of Newcastle (NSW, Australia), which is testing a fixed stereo photogrammetric system [22] for monitoring purposes in a quarry environment in several sites (NSW, Australia). However, the methodology presented in this thesis may have wider implications in civil structural monitoring, for instance in the detection of cracks and spalling in various types of structures.

1.2 STRUCTURE OF THE DISSERTATION

This dissertation consists of five chapters. These chapter are structured as follows:

1. **Chapter 1** explains the motivation for the research together with an overview of the topics covered.
2. **Chapter 2** will provide the theoretical aspects and the state of the art of neural networks, with a deeper insight on convolutional neural networks and on their applicability to monitoring.
3. **Chapter 3** will present the framework for the simulation of rock collapses and will illustrate the detailed structure of the application.
4. **Chapter 4** will provide an overview on the methodologies carried for the calibration and validation of the simulation software, which will be compared with real photogrammetric surveys data carried in collaboration with the University of Newcastle (NSW, Australia).
5. **Chapter 5** will be dedicated to the training of the neural network on simulated datasets and the evaluation of the results obtained on different test datasets.
6. **Chapter 6** will conclude the manuscript with a general insight of the work carried out in this thesis and the scientific outcomes produced. Moreover, some suggestions on the future perspectives of this work will be presented.

The appendix will feature the code for the neural network model used in this thesis as well as the Python code routines used to carry out the research work.

2 CHAPTER 2 - THEORETICAL ASPECTS AND STATE OF THE ART

2.1 INTRODUCTION

In recent years, advancements in computer technologies have led to the development of highly sophisticated neural networks. These advancements have contributed to the diffusion of neural networks, which now spans a wide range of scientific fields, including civil and environmental engineering (particularly in environmental monitoring) [44] [45] [46], medicine [47], finance [48]. Neural algorithms aim to emulate human cognitive capabilities to perform complex tasks and highlight patterns in data that would be hardly recognizable with other mathematical tools.

Today, neural networks are primarily used to perform three fundamental tasks: prediction, classification, and regression [49].

1. **Prediction**: Constructing a model that employs past data patterns to precisely anticipate the distribution of new equivalent data [50] .
2. **Classification**: Identifying whether a particular observation pertains to a distinct group of observations [51].
3. **Regression**: Determining a continuous pattern that closely approximates a given distribution of discrete data [52].

Modern neural network algorithms possess the ability to efficiently perform one or more tasks simultaneously, making them highly adept at addressing various real-world challenges. Their capability to generalize non-linear problems has paved the way for their widespread application across a multitude of fields.

One of the prime beneficiaries of the evolution of neural networks has been computer vision. This domain has witnessed remarkable growth in recent years, largely attributable to the advancements in convolutional neural networks (CNNs). These networks have demonstrated their efficacy in tasks ranging from facial recognition to autonomous vehicle navigation, the analysis of satellite imagery, and so on [53] [54] [55]. Although CNNs have been used to address various scientific problems since the late 1980s, their potential is only now being fully realized with the increasing accessibility of advanced computing resources, such as GPUs and cloud computing.

The advent of neural networks has also revolutionized the field of 3D reconstruction, enabling the creation of detailed three-dimensional models from 2D images with unprecedented accuracy and efficiency. One of the most notable tools in this area is the Neural Radiance Fields (NeRF) [56], [57], which utilizes a fully connected deep network to model a continuous volumetric scene function from a sparse set of 2D images. The strength of this approach lies in its ability to capture fine details and produce high-quality models from complex scenes, outperforming traditional methods and other neural network-based approaches. On the other hand, it's worth noting that while NeRF is a powerful tool for 3D reconstruction, it does have its limitations. For instance, it requires a large number of input images for training and can be computationally intensive. Moreover, it struggles with dynamic scenes where objects or lighting conditions change over time.

The main interest in neural networks for this thesis lies however in object detection and instance segmentation procedures for the identification of rockfall events on photogrammetry-derived raster difference maps. Object detection aims to identify distinct objects within digital images by highlighting their positions with bounding boxes; instance segmentation is the pixel-by-pixel classification of image areas covered by specific objects. This research concentrates on models capable of performing both operations simultaneously through a single training cycle.

2.2 LITERATURE REVIEW

2.2.1 STATE OF THE ART OF NEURAL NETWORKS AND CNNS

The evolution of neural networks lies in early attempts to understand the computational properties of the brain and the development of mathematical models that could imitate the information processing capabilities of biological nervous systems. The early days of neural network research were characterized by the development of simple models and their application to various problems such as binary classification (the task of categorizing input patterns into two classes) [58], linear function approximation, or regression [59] and simple pattern recognition (the task of identifying basic shapes such as squares and triangles, or

handwritten characters). These early problems provided a steppingstone for the research in the field of neural networks, as most of the concepts introduced to counter these problems have become the base for the following works in this field.
Originally, the perceptron, a pioneering neural network model, was introduced by Rosenblatt [58]. This simple yet powerful model, inspired by human neurons, was capable of learning linearly separable patterns using an algorithm that adjusted the weights of the connections between input and output neurons. The perceptron's success in solving simple classification problems was considered an adequate solution to counter more complex tasks. However, in the following years, researchers proved this solution to be unable to solve non-linearly separable problems, such as the XOR problem [60]. The XOR problem (or Exclusive Or) is a binary logical operator that takes in Boolean inputs under binary format (0 or 1) and provides True values (1 in this case) if and only the two inputs are different.
The XOR issue was a major hit to neural networks development, as research in this field experienced a long period of stagnation, often referred to as "AI winter", during 1970s and 80s. The perceptron limitation highlighted by the XOR problem, jointly with the competition from other techniques (such as decision trees) and computational constraints, greatly reduced interest (and therefore fundings) in the field.
Some researchers continued however to explore the potential of neural networks, leading to the development of new learning algorithms and architectures.
The early 1970s saw the emergence of a new class of neural networks known as radial basis function (RBF) networks, which countered the limitations of perceptrons by using nonlinear activation function [61] [62]. Radial basis function networks were able to approximate any continuous function to arbitrary precision, making them a powerful tool for various applications, such as function approximation, pattern recognition, and time series prediction. However, it was the development of the backpropagation algorithm [63] that truly revolutionized the field. This learning algorithm, based on the chain rule of derivative calculus, allowed for the efficient training of perceptrons enabling them to learn complex, non-linearly separable functions. In this context, the concept of Multi Layered Perceptrons (MLP) was introduced. MLPs are full arrays of one or more perceptron, arranged on independent layers. The backpropagation algorithm was a significant milestone in the evolution of neural networks, as it facilitated the development of more sophisticated architectures and learning techniques, starting a resurgence of interest in the field.

One of the key developments in the late 1980s and early 1990s was the introduction of recurrent neural networks (RNNs) [64]. In contrast to classical neural networks, RNNs have connections that go back on themselves, allowing them to process temporal sequences and maintain a form of internal memory. RNNs have been successfully applied to various tasks, such as time series prediction and natural language processing (NLP) [65].

In addition to the development of new architectures, researchers also focused on improving the learning capabilities of neural networks. For instance, the vanishing gradient problem, which occurs when the gradients of the error with respect to the weights become too small to effectively update the weights during backpropagation, was identified as a significant challenge in training deep neural networks [66]. To address this issue, various techniques, such as unsupervised pre-training and the development of activation functions with more interesting properties, such as rectified linear units (ReLU) [67], were proposed.

Another critical development during this period was the exploration of regularization techniques to prevent overfitting and improve the generalization capabilities of neural networks. Techniques such as weight decay [68] were introduced, allowing neural networks to learn more robust representations of the data and perform better on unseen (i.e., completely new) examples.

The late 1980s and early 1990s also saw significant advancements in the application of neural networks to real-world problems. One notable example is the development of convolutional neural networks (CNN) for image recognition tasks. In [69] the authors proposed a novel methodology for image classification that leverages a backpropagation algorithm to train a convolutional network. CNNs use shared weights to exploit the spatial structure of images, reducing the number of parameters to be learned and improving the network's ability to recognize patterns in image data.

The reference CNN model introduced by LeCun was LeNet, a deep structure model characterized by the presence of MLPs placed in series and having mutual connections. As it can be seen, the MLP concept previously introduced by Rumelhart [63] remained the basic structure for the subsequent evolutions of neural networks for processing digital images. In his study, a completely innovative network training model was proposed that exploited a backpropagation algorithm which leveraged gradient descent algorithms. The results of this study, which first involved the recognition and classification of handwritten characters on digital images of the MNIST dataset (Modified National Institute of Standards and

Technology, which consists in 70.000 grayscale images of handwritten digits), started a very successful strand for neural network training methodologies.

Meanwhile, between 1998 and 2010, several advancements were made in the field of machine learning. Support Vector Machines (SVM) [70] emerged as popular machine learning technique for classification and regression tasks. The core concept of SVMs is developed as input vectors undergo a non-linear transformation, projecting them into a high-dimensional feature space. Within this feature space, a linear decision threshold is established. The distinctive attributes of this decision boundary guarantee an exceptional capacity for generalization in the learning process. SVMs demonstrated strong generalization performances in these tasks. Some key applications of SVMs include image and text classification [71] and time series forecasting in the financial field [72].

In 2001, Random Forests (RF) were introduced that provided strong performances in classification and regression tasks compared to the neural network models available at the time [73]. RF is a learning method that constructs multiple decision trees for the same task and combines their prediction to improve accuracy. This methodology allows to increase the redundancy within the model's forecasts, facilitating the minimization of potential inaccuracies in the given task. Some notable applications regarded the medical field, with particular attention to bioinformatics and medical diagnosis [74] [75]. In geomatics Random Forests have been used for remote sensing, in particular for land cover classification, identifying natural resources and monitoring environmental changes [76] [77] [78].

After 2010, shifting the computational burden from the CPU to the GPU made it possible to greatly speed up processing times, especially in the training of millions of parameters (or weights) that characterize the most complex neural network models. In [79] a simple implementation of a neural network for small images is presented, that based the convolution operations no longer on the CPU, but rather on CUDA technology [80], which was charged to the GPU. Experimental results showed an enormous performance advantage in this respect, with processing times drastically improved with respect to the CPU counterparts.

In the same vein, in 2012 [81] illustrated a new convolutional network model for image classification, AlexNet, designed for the classification of the ImageNet dataset. This work represented a watershed for research activities on neural networks.

The novelty of this network lays in the structure of the convolutional layers, interspersed with new type of intermediate layer [82]. These intermediate layers leveraged a max pooling algorithm, which is a technique that allows a feature to be subsampled by reducing its size accounting for its maximum value only. In other words, the max pooling technique makes possible to significantly reduce the computational cost of operations on numerical matrices (and thus on digital images) and, at the same time, provide translation invariance in the network. A conceptually similar solution could be provided by the global average pooling technique [83], which implied the averaging of the elements within the search kernel. After each convolution-max pooling step, a dropout layer was chosen. Dropout [84] is a technique for regularizing the weights of the neural network that is applied exclusively during the training phases. This solution allows to avoid overfitting problems, which are common in neural networks characterized by millions of parameters. One of the main interests of this study was the evaluation of the network's performance as a function of the depth of the network itself. Compared to LeNet, AlexNet could boast more than twice the number of convolutional layers. Thanks to the increased network depth, an overall improvement in performance was observed. The reason for this improvement lies mainly in the possibility of extracting meaningful features through convolution operations on the image.

Various architectures derived from AlexNet were subsequently proposed. The main interest of the researchers concerned the optimization of convolution operations rather than the brutal increase in model depth. In this respect, one of the most successful solutions is ZF-Net, presented by [85] in 2013. This study was preliminarily oriented towards presenting a visualization model of the convolution operations that are processed within each neural layer. The motivations for this research were mainly to understand what happened at the transition between one neural layer and the next. In fact, until then, neural networks were considered black boxes, where there was no real knowledge of what might happen between input and output. In [86] the authors proposed a visualization method using a neural network with inverse behavior, based on deconvolution operations (DeconvNet). A DeconvNet is a network modelled with the same filtering and pooling parameters as the network on which it originally relies, but which develops its course in an inverse manner. If a convolutional network could map pixels into features while maintaining spatial correlations between them, the DeconvNet could perform the same task backwards. The solution presented requires that each layer of the convolutional network is connected to a layer of the DeconvNet, to have a reverse and direct tracking towards the pixels of the image.

The structure of ZF-Net roughly follows the one in AlexNet, the main changes concerning the size of the convolution window and the displacements associated with it.
In 2015, [87] proposed a new CNN model characterized by an even greater depth than the neural networks on the market. The network, which was named VGG-16, was characterized by the presence of as many as 16 neural layers. The possibility of significantly increasing the depth of the network (i.e., the number of layers) is primarily due to the technological progress of calculation units, increasingly powerful and more affordable.
The experiments on this new CNN model showed a considerable leap in feature extraction performance due to the presence of more neural layers. A further variant of VGG-16 was presented in the same work. The modification involved the integration of three additional neural layers, with the aim of assessing any performance gain achieved. In this respect, the authors concluded the study by pointing out the modest performance improvement of VGG-19 over VGG-16, while confirming the importance of the depth of the neural networks in feature extraction processes.
The possibility of increasing the depth of neural networks has revived an issue of considerable interest to researchers: the vanishing and exploding gradient.
In[66] the concept of the vanishing gradient was introduced for the first time. The author defined this issue on a theoretical level by analyzing the error flow for learning methods based on gradient descent (GD) algorithms.
In [88], a novel Google LeNet convolutional structure was presented that increased the depth of the network while simultaneously managing to reduce the computational burden of convolutional operations (reducing the number of parameters, or weights, involved in training operations). With the substantial increase in convolutional layers involved in the new neural models, the vanishing gradient problem illustrated above presented researchers with a new challenge.
In [89], a research group belonging to Microsoft presented a new residual characteristic convolutional neural network (R-CNN) named ResNet.
The innovative idea behind R-CNNs is to connect each convolutional layer not only with the next layer but also with the previous one, creating a shortcut connection.
The structure of the convolutional blocks enriched by these mutual connections between layers has made it possible to exponentially increase the depth of the networks, bringing

great benefit to feature extraction operations for object detection while avoiding the problems of vanishing\exploding gradient.
In this sense, the architecture proposed by the researchers in [89] brought about a considerable increase in the numerosity of the convolutional layers because of this solution. The first version of ResNet was characterized by the presence of 152 neural layers and was successful among researchers in different fields. The main reason for this success lies precisely in the concrete improvement of feature extraction tasks with its natural spin-off on object detection operations. Nowadays, residual feature convolutional networks occupy a place of primary importance in the field of machine learning aimed at object detection (and consequently instance segmentation).
CNN Object detection is a direct extension of the feature extraction methodologies illustrated above. The main methodologies for object detection reported in the literature are two: two-level object detection and Single Shot Detectors (SSD).
In this literature review, we chose to focus on two-level object detection solutions. This methodology has indicated a preferred route for subsequent architectures for semantic segmentation of digital images.
Two-step object detection solutions extend previous feature extraction architectures with neural layers responsible for searching for regions of interest within the images.
A first two-step solution was presented in [90]. Fast R-CNN was the very first architecture to use a Selective Search to generate regions of interest on the frame. Selective Search [91] is a region proposal method for object detection tasks. The main idea behind selective search is to efficiently generate a set of potential bounding boxes, called regions of interest (RoIs), which have high probability to contain objects within an image. These RoIs are then used as input for a classification algorithm to determine the presence of objects and to classify them in the image.
Selective search is based on the observation that objects in an image can be distinguished by their color, texture, and size. The algorithm combines these features in a hierarchical manner to generate candidate RoIs.
Experimental results conducted on different datasets, such as VOC2012 and COCO, showed excellent network performance in terms of both prediction accuracy and timing involved in training the network. The real bottleneck of Fast R-CNN, however, remained the performance in the image prediction phase. The promising object detection capabilities of

Fast R-CNN directed the researchers towards an optimization of the architecture in order to streamline the number of parameters involved and thus reduce the computational burden. In this sense [92], they proposed an architecture update that uses a Region Proposal Network (RPN) to perform the object detection tasks.
In Faster R-CNN, the feature map obtained from the R-CNN network is directly shared with the RPN to determine its position within the frame. Thanks to this solution, the procedure for identifying regions of interest is almost completely computationally irrelevant. An inspection kernel of size 3x3 pixels is moved within the image and allows the regions of interest to be identified by means of anchor boxes. The anchor boxes are regions of interest with dimensions and proportions set a priori. This method assumes that each region of interest has a unique score in order to identify the best fit for detection.
Advances in the field of object detection, mainly with the introduction of Faster R-CNN, have made it possible to explore the possibilities of integrating semantic segmentation algorithms into object detection procedures with a modest increase in computational demands.
The problem of semantic segmentation in digital images can be traced back to the problem of pixel-to-pixel classification of a frame.
In [93], a segmentation algorithm based on a fully convolutional network (FCN) was presented. This type of convolutional network makes it possible to circumvent certain limitations of classical FCNs, such as the fixed size of the input image. The architecture proposed in Long's work had the ability to process inputs of arbitrary dimensions and in turn generate segmentation maps with the same dimensionality. The advantage of this solution lays mainly in the speed of image processing due to the reduction in training parameters resulting from the use of an FCN network.
In [94], an innovative algorithm for performing semantic image segmentation operations called U-Net was presented. U-Net exploits an FCN architecture with an encoder-decoder structure in which the encoding and decoding branch are completely mirrored. In the contraction branch, several convolutional layers are applied, interspersed with pooling layers that are used to subsample the features, whose maps double in number at each step. In the expansion branch, the same procedure is carried out in the opposite direction. The oversampling of the features finally aims to produce segmentation maps of the same size as the input data.

The simultaneous resolution of object detection and semantic image segmentation problems introduced the concept of instance segmentation. Where a specialized object detection system can approximate the location of an object within the image by means of a bounding box, and semantic segmentation produces a unique classification for each pixel belonging to the same class, the instance segmentation procedure reconciles the two methods, producing a segmentation map for each category and for each instance of a particular class.
One of the first approaches to the problem was presented in [95] with the introduction of the DeepMask architecture. This framework aimed to generate possible segmentation masks for objects starting directly from image pixels. The algorithm operated at different scales on the image and generated segmentation proposals of different ranks, subsequently classified by a Fast R-CNN model. The authors illustrate promising results on both the COCO dataset and PASCAL VOC2007, both of which are characterized by the massive presence of features at different scales.
Subsequently, the method presented in [96], named Mask R-CNN expanded the concept of Faster R-CNN [92] by adding a branch for the prediction of masks associated with regions of interest (RoIs). Segmentation is performed on each region pixel by pixel using an FCN (Fully Connected) neural network. The results on instance segmentation procedures were processed using several datasets, including the Cityscapes dataset [97]. This dataset comprises a series of cityscape sequences recorded in approximately 50 different cities and contains 5000 frames at high resolution with high quality annotations to evaluate the performance of instance segmentation algorithms.
The motivations that led the authors to choose this dataset lie mainly in the variability of the features present in the frames, both in terms of morphology and scale of representation. As of today, Cityscapes dataset is involved in major CV projects regarding autonomous driving [98].
The results obtained by the authors in [97]using a ResNet-FPN-50 convolutional backbone (characterized by the presence of 50 convolutional layers for feature extraction) are superior to (or at least in line with) other solutions found in the literature.
Today, Mask R-CNN and its evolutions still hold a prominent place in the state of the art of instance segmentation procedures thanks to the simplicity of application of the model to various fields of technical and scientific interest.

2.3 Theoretical aspects for Neural Networks

2.3.1 Neural Networks

Neural networks are tools characterized by a repetitive structure consisting of an input layer (or input), one or more neural layers, commonly called hidden layers, and an output layer [99].

Each layer consists of nodes, called neurons. Each node in the n-th layer is directly connected to all nodes in the next n+1-th layer.

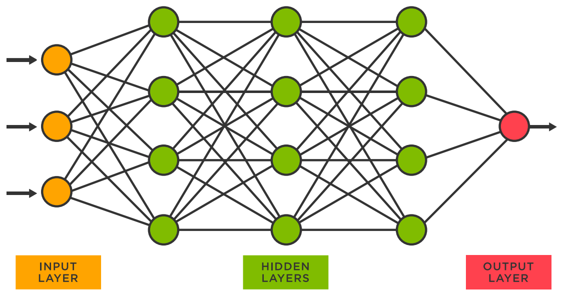


Figure 2-1: NNs general structure

The place where the mathematical operations are performed is the **neural node**, characterized by a function of the following type:

$$f(x, W, b) = \sigma(W\,x + b)$$

Where **W** denotes a weight value, **b** denotes a bias value and **σ** refers to an activation function. Activation functions play a key role in the representation of non-linear problems and are the reason why deep learning algorithms excel at processing extremely complex data sources such as digital images.

Using inverse propagation (backpropagation) algorithms for training, a neural network can adapt its weights and biases to generalize a given problem through the minimization of a loss function [100].

A loss function, which is also known as cost function, quantifies the disparity between the predicted output generated by a neural network and the actual target output provided in the training data. The loss function is essential during the training phases of a neural network, as it provides the measure of how well (or not) the network produces the correct prediction to match the ground truth training data, therefore providing the right direction to the learning process.

A loss function is mathematically expressed as a scalar value that expresses the dissimilarity between the ground truth (y) and the prediction ($\hat{y}$):

$$L\,(y, \hat{y}) = y - \hat{y}$$

There are different types of loss function which are adopted to tackle different tasks. For example, MSE (Mean Squared Error) loss functions are used in regression and prediction tasks [101], while for classification purposes the most common loss function is cross-entropy [102].

Backpropagation algorithms are widely used for training neural networks. Backpropagation involves computing the gradients of the loss function with respect to the network's parameters using the chain rule of derivative calculus. Activation functions are responsible for giving reasons for non-linear relationships. Each node is capable of being trained to learn to recognize a specific feature and, consequently, the set of nodes in a neural network enables the identification of the set of features belonging to a specific class.

The initialization of the weights is a procedure of crucial importance for neural algorithms, since excessively small weights (tending to 0 already in the first neural layers) or excessively large weights can lead the network to fail in training due to the problems of vanishing or exploding gradients.

Vanishing gradient is a phenomenon where the gradients of the loss function with respect to the network's parameters become very small, causing the weights of earlier layers to be updated very slowly or not at all. This can lead to the degradation of the network's performance, as the early layers fail to learn meaningful representations of the input data. The vanishing gradient problem was first identified in the early days of neural networks and has been studied extensively ever since. In [63] the authors introduced the backpropagation algorithm and highlighted this potential issue. Different approaches have been proposed to avoid vanishing gradient issues in training, including the use of batch normalization [103] which is a procedure that helps to stabilize the distribution of activations and gradients throughout the network by normalizing the activations of each layer across the mini batch of data, and then rescaling and shifting the normalized values using learnable parameters. A similar issue is the exploding gradient problem, in which the gradients of the loss function become very large, causing the weights to be updated in a way that overshoots the optimal solution. This can lead to the divergence of the training process, resulting in poor performance overall. Exploding gradient is less common than the vanishing gradient problem but it can still occur in certain types of networks characterized by recurrent connections. It has been studied extensively, and several techniques have been developed to address it. One approach is to use gradient clipping, which involves scaling down the gradients if their norm exceeds a certain threshold, to prevent the gradients from becoming too large. Other methods include weight regularization [104], which can help to constrain the magnitudes of the weights, and careful initialization of the network's parameters [105], which can help to prevent the gradients from growing too large.

2.3.2 ACTIVATION FUNCTIONS

Activation functions are fundamental in the modelling of a neural network. There are several functions to rely on, each with peculiarities that make them suitable for different needs. The main activation functions [106] reported in the literature are:

Step function: this is the simplest activation function, where a neuron is activated depending on a threshold value. The equation that defines its behavior is:

$$f(x) = 1, x > 0$$
$$f(x) = 0, x < 0$$

It is a function closely related to binary classification problems. It is not usually used in transition layers in a neural network as it hinder the back-propagation process on node weights as its derivative as a function of x is always zero.

Sigmoid function: it is a non-linear function that returns a value between 0 and 1 from a coefficient in the real domain. It is widely used in the field of neural networks as it is derivable in the entire domain. It is defined by the equation:

$$f(x) = \frac{1}{1 - e^{-x}}$$

A common problem related to this function concerns the vanishing gradient. The main reason for this lies in its asymmetry around zero, which makes it difficult to unambiguously identify the most effective direction for the gradient descent. Sigmoid functions are largely used in classification layers (i.e., the last layer of a neural network used for classification purposes) as it defines a continuous function in which classification intervals are defined.
In tasks of machine learning, different classification problems pose distinct challenges and methodologies. For binary classification, where the objective is to differentiate between two classes, the sigmoid function is often employed. This function maps any real-valued number into a value between 0 and 1, which can be interpreted as the probability of the instance belonging to the positive class. Given a threshold, typically 0.5, outputs above this value indicate class 1, and below indicate class 0.
Rectified Linear Unit (ReLU): is the most used function in the interleaving layers of convolutional neural networks. The ReLU function returns the maximum observed value between 0 and x, so all values greater than 0 retain their value while values less than 0 take on a null value. This methodology reduces the complexity of processing time. The function is defined by the equation:

$$f(x) = \max(0, x)$$

The ReLU (Rectified Linear Unit) activation function is widely used in convolutional neural networks (CNNs) because of its efficiency and simplicity. This function, which replaces all negative values with zero and leaves positive values unchanged, has been shown to speed up convergence during training compared with other traditional activation functions such as sigmoid or hyperbolic tangent. In addition, ReLU helps mitigate the problem of gradient disappearance, which can occur with activation functions that compress the output into limited intervals. As CNNs increase in depth and complexity, the use of ReLU can help maintain stable and effective training while reducing training time. This function is able to switch off non-relevant neurons and therefore speed up the computational workflow for training and inferencing processes. As negative values are not employed in the activation of the neuron by ReLU function, it is critical to refer to normalized inputs (i.e., changing the pixel intensity values in a range between 0 and 1) for convolutional networks.

Leaky ReLU: is a variation of the classic ReLU function that allows the number of inactive nodes in the backpropagation process to be minimized. Instead of evaluating the contribution of the ReLU function to be zero, a small linear component is applied. The equation which defines the function is:

$$f(x) = 0{,}01x, x < 0$$
$$f(x) = x, x > 0$$

This function is the evolution of the previous ReLU function. Leaky ReLU allows for additional control in the activation of neurons across neural layers, possibly preventing vanishing gradient problems when the neural architecture is very deep. On the other hand, leaky ReLU usage implies longer processing time because most of the neurons are fully functional at each step (while possibly not heavily contributing to tasks). This function is very useful when nodes weights rapidly descend to null values in the training process.

SoftMax function: it is the most exploited function in classification problems with rank greater than two (i.e., multi-class classification problems with three or more classes). It is a combination of several sigmoid functions and calculates a probability that the input belongs to an i-th class between 0 and 1. In this case, the network attempts to give classification intervals between 0 and 1. In this way, the result of the network's processing can be classified according to the output value calculated using the hidden layers. This function is usually used to process the output of the last neural layer using the following formulation:

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{i=1}^{K} e^{x_i}}$$

Tanh function: it is a function that transforms the input x into a value between -1 and 1 with symmetry around 0. It is rarely used in convolutional neural networks as it presents vanishing gradient problems in deep-structure architectures. The reference equation is:

$$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

The choice of activation functions in a neural network is of crucial importance to maximize its performance [107] [108]. One could say that activation functions are the most important parameter in the research of peak performance of neural networks. The choice of activation functions is most of the time an experimental task, meaning that the user cannot know a priori if the solution adopted is suitable for a specific problem or not. Most modern convolutional neural networks use ReLU activation function for convolutional layers instead of traditional sigmoid functions. The motivation is that ReLU (and all of its variants like Leaky ReLU) promote sparsity in the neuron's usage, meaning that it reduces considerably the activation of certain neurons. The empirical proofs show that training a neural network is much faster with ReLU [27] and its variants, which makes them particularly suitable for deep neural networks involving millions of neurons (or weights) to be computed simultaneously, like for example deep convolutional neural networks.

2.3.3 CONVOLUTIONAL NEURAL NETWORKS (CNNS)

Convolutional Neural Networks (CNNs) are a class of deep learning models that have gained significant popularity in recent years due to their exceptional performance in various computer vision tasks, such as image classification, object detection, and semantic segmentation [27]. These networks are designed to mimic the processing of visual information in the human visual system, where different neurons in the visual cortex are responsible for detecting specific features and patterns in the visual field [109].

The fundamental building block of CNNs is the convolutional layer, which leverages convolution operations. A convolution is a mathematical operation that combines two functions to produce a third, resulting in a measure of how one function is influenced by another. Each convolutional layer consists of multiple filters, also known as kernels (Figure 2-2), that slide over the input data and perform element-wise multiplication followed by a sum of the results [38]. This operation can be interpreted as a measure of similarity between the filter and the local region of the input data, effectively capturing local patterns in the

data. Filters are learned by the network during the training process, allowing CNNs to automatically discover relevant features for the task at hand.
Another essential component of CNNs is the pooling layer (which is nonetheless used in different neural network architectures as, for example, Recurrent Neural Networks), which is introduced to reduce the spatial dimensions of the feature maps produced by the convolutional layers. By performing a down sampling operation, the pooling layer helps the network achieve invariance to translations, making the model robust to variations in the input data [110]. Furthermore, reducing the spatial dimensions also reduces the computational complexity and memory requirements of the network, allowing for more efficient training of deeper models. Pooling layers are typically located between two consecutive convolutional layers.
The three predominant types of pooling operations that are commonly employed are max pooling, average pooling, and min pooling.
Max pooling is the most frequently used pooling technique, and its operation can be described as selecting the maximum value from each non-overlapping sub-region in the input feature map. For example, given a 2x2 region in the feature map **A**, the max pooling operation **P** is defined as:

$$P(A) = \max\left(a_{i,j}\right)$$

Where $a_{i,j}$ is the feature map element in the i, j-th position. The core advantage of max pooling lies in its ability to preserve the most salient features in the input, effectively serving as a form of non-linear down-sampling. This robustness aids the network in achieving translation invariance, as the maximum value remains unchanged irrespective of minor translations within the sub-region.
Average pooling computes the average value for each non-overlapping sub-region in the input feature map. As for the previous example, the average pooling operation is defined as:

$$P(A) = \frac{1}{n}\sum_{i=1}^{n} a_{i,j}$$

Unlike max pooling, average pooling maintains a representative value that accounts for all the elements within the sub-region, which can sometimes provide a smoother representation of the feature map. This operation is particularly useful when it is desirable to conserve the general context information in the feature map.

Min pooling is less commonly used but operates by selecting the minimum value from each non-overlapping sub-region. The mathematical formulation is analogous to that of max pooling. Min pooling can be instrumental in scenarios where the smallest features in the data need to be emphasized, potentially serving as a form of outlier detection within the convolutional layers.

A typical CNN architecture consists of multiple convolutional and pooling layers, followed by one or more fully connected layers. The fully connected layers, also known as dense layers, allow to integrate the features learned by the convolutional layers and map them to the desired output, such as class probabilities in the case of classification tasks [69]. To train the network, a loss function is employed to measure the discrepancy between the model's predictions and the ground truth labels, and an optimization algorithm, such as stochastic gradient descent, is used to update the model's parameters iteratively [111].

CNNs have also been extended and adapted for various other applications beyond computer vision, such as natural language processing and speech recognition. For instance, in natural language processing, one-dimensional CNNs can be used to capture local patterns in text data by performing convolutions over word embeddings[112].

On the other hand, CNNs also have some limitations and challenges. One of the main challenges is the requirement for large amounts of labeled data to train these models effectively. This limitation has led to the development of various data augmentation techniques, such as random transformations and cropping, to artificially increase the size of the training dataset and improve model generalization [113]. Furthermore, transfer learning techniques have been proposed to leverage pre-trained CNNs on large datasets as feature extractors or initializations for training on smaller datasets, significantly reducing the need for extensive labeled data [114]. Transfer learning is a machine learning technique that involves utilizing a model trained on a specific task or dataset to enhance the performance of a related task or dataset. Notably, researchers can access various open-source pre-trained weights (for example, COCO or ImageNet pre-trained weights), which offer the advantage of achieving faster and more reliable results in common object detection tasks.

2.3.4 TRAINING

The training of neural networks is carried out in a supervised manner (generally called supervised learning) using the input information provided by a reference dataset. By means of back-propagation algorithms, the network proceeds to optimize the parameters to minimize the prediction error. The focus of operations concerns the minimization of a function that identifies the differences between the predicted (output) data $\hat{Y}$ and the ground truth Y, this function is called the **cost function** (or loss function, or objective function). The relationship between a calculation node and the node in the next layer is mathematically identified by means of the "*chain rule*" (Leibnitz). Given a (loss) function:

$$h(x) = f(g(x))$$

where x represents all the parameters (weights and biases) of the neural network, the variation of h(x) as a function of the variation of h deriving from a variation of the parameters x can be calculated as:

$$\partial \mathrm{h(x)} = \frac{\partial \mathrm{f}\ \partial \mathrm{g}}{\partial \mathrm{g}\ \partial \mathrm{x}}\, \partial \mathrm{x}$$

The gradient descent procedure allows the operating parameters of the network to be updated by following the negative gradient of the objective function. To facilitate the calculations, an SGD (Stochastic Gradient Descent) procedure is generally used [115] in the modern neural network models. The Stochastic Gradient Descent (SGD) algorithm is a variant of the traditional Gradient Descent (GD) optimized to reduce computational time. While the standard GD computes the gradient using the entire training dataset, the SGD, as the name suggests, operates stochastically, randomly selecting a subset or minibatch of the training dataset. The process begins with the random selection of a minibatch from the dataset. Next, the gradient of the loss function with respect to the model parameters is calculated based on that minibatch. Once the gradient is obtained, it is used to update the model parameters. This updating is done by multiplying the gradient by a factor called the learning rate and subtracting the result from the current parameters. This step is crucial because it steers the model in the direction in which the loss decreases most rapidly. It is important to note that due to the stochastic nature of the approach, the trajectory of parameter updates with SGD can be more erratic than with traditional GD. However, this can actually help the model avoid local minima and converge more quickly to a solution.

This procedure is repeated for each minibatch in the dataset. Once all minibatches have been processed, an epoch is said to have been completed. The process is then repeated for a predefined number of epochs or until the loss converges to a minimum value. To reduce the model complexity, and to counter overfitting problems, various regularization techniques of the network are used in the training phase. The issue of overfitting was first addressed in [116] as the problem of a neural network that tends to fit exactly the data distribution on which it is trained rather than finding an appropriate generalization rule. There are two main regularization techniques: L2 regularization and dropout. L2 regularization [117], often simply referred to as "weight decay", introduces a penalty term to the objective (or loss) function to deter the magnitudes of the model parameters from becoming too large. Specifically, the penalty term is proportional to the sum of the squared values of the model parameters. This is achieved by implementing a λ hyperparameter as a penalty term on larger weights. The inclusion of this term serves multiple purposes. By penalizing large weights, the model is steered towards simpler, more generalized solutions that are less likely to overfit to the training data. The parameter λ acts as a tuning knob: when λ is set to zero, no regularization is applied, whereas a larger λ increases the regularization strength, pushing weights more aggressively towards smaller values.

This method essentially adds a constraint to the optimization problem, where not only is the model trying to minimize the loss with respect to the training data, but it is also trying to keep its weights small. As a result, when weights are regularized with L2, they tend to be driven to values close to zero, especially if λ is set to a high value. This does not mean the weights become exactly zero but rather they are kept small, ensuring a certain level of simplicity in the model's learned representation. The second technique regards Dropout [84], which allows for the complete inhibition of a random portion of nodes in the network. This methodology enables significant simplification of calculations during neural network training by allowing for the deactivation of multiple nodes. This, in turn, reduces the number of equations required from one hidden layer to the next. Deactivating a node through dropout can lead to a reduction in the number of computations by a factor of "n" where "n" corresponds to the ratio of the number of deactivated connections and the number of the total nodes. For instance, it is worth considering a scenario where a layer comprising 100 nodes is fully connected to a second layer, also containing 100 nodes. In this configuration, the total connections amount to 100 x 100 = 10000. Upon applying a dropout rate of 0.5, approximately half of these connections become inactive, resulting in the random deactivation of n = 5000 connections.

2.4 MASK R-CNN – TOOL FOR INSTANCE SEGMENTATION

2.4.1 INTRODUCTION TO THE NEURAL ARCHITECTURE

One of the most popular solutions for performing instance segmentation tasks is Mask R-CNN [96], developed within the FAIR (Facebook AI Research) research group. Mask R-CNN belongs to the class of R-CNNs (Regional Convolutional Neural Networks) and is a direct extension of Faster R-CNN. Unlike traditional convolutional neural networks, which involve the progressive scanning of all areas of the image, regional neural networks allow the identification of areas of interest (called RoIs) where the presence of an object of interest is assumed. This procedure streamlines the network's computational burden, reducing training and inference times without affecting the model's achievable accuracies. However, an in-depth analysis of this neural network model is not the subject of this thesis, therefore only a general overview of the structure and special features of Mask R-CNN will be provided.

2.4.2 MASK R-CNN STRUCTURE

Mask R-CNN consists of three main components:

1. Convolutional backbone to perform feature extraction operations.
2. RPN (Region Proposal Network) to identify potentially relevant regions within the image depending on the presence of features.
3. Mask Branch a derived convolutional neural network to generate the masks for the identified objects.

Figure 2-3 represents the basic structure of Mask R-CNN.

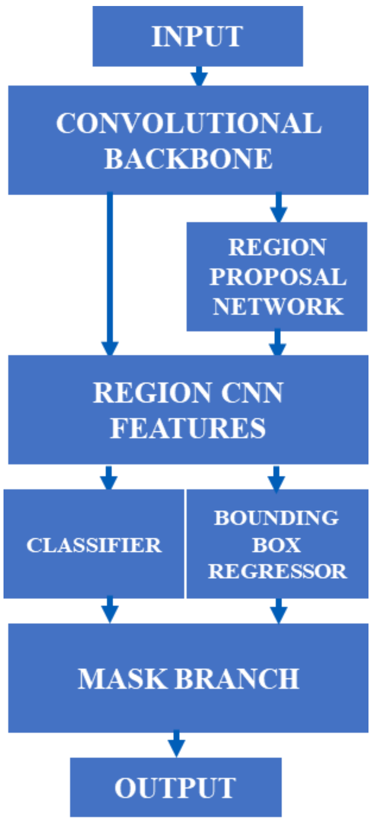


Figure 2-2: Mask R-CNN structure as illustrated by authors in [96].

1. The backbone network in Mask R-CNN is responsible for extracting features from the input image that can be used for both object detection and instance segmentation, serving as a deep convolutional neural network (CNN) that extracts features from images. Typically, it is pre-trained on a large-scale image classification dataset, allowing the use of learnt general features useful for various subsequent tasks (e.g., object detection and/or instance segmentation). The choice of the backbone network plays a significant role in the model's overall performance. In the original Mask R-CNN research, the ResNet-101 backbone was used, which achieved state-of-the-art results on the COCO dataset for object detection and instance segmentation. This deep residual network has 101 layers and is effective for various computer vision tasks. Other backbone networks, such as VGG and Inception, have also been utilized. VGG is a well-known CNN architecture initially intended for image

classification, while Inception is a CNN family designed to be efficient and scalable. In recent years, there has been an increasing trend of using even more complex and more extensive backbone networks for object detection and instance segmentation. For instance, the recent DETR [118] model utilizes a transformer-based backbone network pre-trained on a vast dataset of natural captions, resulting in state-of-the-art performance on the COCO dataset. Although ResNet-101 is an excellent choice, other architectures might be more effective. Researchers continue to explore and develop new and more powerful backbone networks for object detection and instance segmentation, aiming to achieve even better results on challenging datasets.

2. The Region Proposal Network (RPN) is responsible for generating potential object regions or proposals using the feature maps from the backbone network. The RPN is a small neural network that takes the feature maps as input and outputs a set of rectangular regions likely to contain objects. The RPN uses a sliding window approach to generate candidate regions. For each spatial location in the feature maps, the RPN predicts a set of k anchor boxes with different scales and aspect ratios. The anchor boxes define a set of potential regions, which are then filtered based on their score to produce the final proposals. The score reflects how probable is that an object exists in the proposed region and is calculated as the product of two terms: the probability that the anchor box contains an object (foreground score) and the probability that the anchor box is correctly aligned with an object in the image (background score). The background score is determined based on the intersection-over-union (IoU) between the anchor box and the ground truth objects in the image. The RPN is trained end-to-end with the rest of the network using a multi-task loss that combines the object loss and the bounding box regression loss. The object loss trains the RPN to differentiate between foreground and background regions, while the bounding box regression loss trains the RPN to predict accurate bounding box coordinates for the object proposals. In the original Mask R-CNN paper, the RPN was constructed using a 3x3 convolutional layer with 256 channels, followed by two 1x1 convolutional layers for predicting the object score and the bounding box coordinates, respectively.

3. The Mask Network, also known as the Mask Branch, is responsible for generating binary masks for each object proposal generated by the RPN. It takes each proposed region as input and generates a binary mask that highlights the pixels belonging to the object. The mask network is typically a fully convolutional neural network that takes the feature maps

produced by the backbone network and the proposed regions as input and outputs a binary mask of the same size as the input region. The mask network is trained to predict the binary mask of each object proposal. Its architecture is similar to the one of the backbone network, but with additional layers for generating the binary mask. In the original Mask R-CNN paper, the mask head architecture was straightforward, comprising four 3x3 convolutional layers followed by a single 1x1 convolutional layer with a sigmoid activation. The mask network is trained using a binary cross-entropy loss, which measures the difference between the predicted mask and the ground-truth mask for each object proposal. The mask loss is added to the overall loss function of the model, which also includes the object detection loss and the RPN loss.

The loss function in Mask R-CNN is a multi-task loss that combines losses corresponding to the different components of the model. It is defined by the formula:

$$Loss = L_{cls} + L_{bbox} + L_{masks}$$

where i) L_{cls} represents the classification loss of the prediction box, ii) L_{bbox} represents the regression loss for the predicted bounding box and iii) L_{masks} represents the loss for the predicted masks.

i) The classification loss aims to train the network to correctly identify the category of the object enclosed by the proposed bounding box. It is implemented using softmax loss for multiple categories. Given n categories, including the background, and assuming p as the predicted probability distribution and g as the ground truth distribution, the classification loss can be defined as:

$$L_{cls} = \sum_{i=1}^{n} g_i * \log(p_i)$$

Where g_i and p_i correspond to the ground truth label and predicted probability of the i-th category.

ii) The bounding box regression loss is meant to refine the coordinates of the initial object proposal generated by the Region Proposal Network (RPN). The used bounding box regression loss is the Smooth L1 loss (which is shared with the first implementation of Faster R-CNN. Let t* = (Tx*, Ty*, Tw*, Th*) and t = (Tx, Ty, Tw, Th) be the ground truth and predicted parameterized coordinates of the bounding box, respectively. Tx and Ty are representative of the lower left corner of the bounding box, while Tw and Th are respectively the width and the height of the bounding box. The Smooth L1 loss can be defined as:

$$L_{bbox} = \sum_{i \in (x,y,w,h)} Smooth_{L1}\,(t_i^* - t_i)$$

where SmoothL1 is defined as:

$$Smooth_{L1}(x) = \begin{cases} 0.5x^2 if\ |x| < 1 \\ |x| - 0.5\ otherwise \end{cases}$$

iii) The mask loss is crucial for the pixel-level segmentation task. It employs per-pixel sigmoid activation followed by binary cross-entropy loss. Given M as the ground truth mask and $\widehat{M}$ as the predicted mask, the mask loss L$_{mask}$ is computed as:

$$L_{mask} = \sum_{w=1}^{W}\sum_{h=1}^{H}[M(w,h)\log\left(\widehat{M}(w,h)\right) + \left(1 - M(w,h)\right)\log\left(1 - \widehat{M}(w,h)\right)$$

Where W and H are the width and height of the given mask. Such loss function is the per pixel adaptation of the sigmoid function expressed in section 2.3.4. The composite nature of this loss function allows Mask R-CNN to perform the tasks of object detection and instance segmentation jointly, thus enabling a unified framework for these different but related computer vision tasks. In other words, each term of this composite loss function helps in the end-to-end training of the network, ensuring that the model is proficient not only in categorizing objects but also in localizing and segmenting them at a pixel-level.
Mask R-CNN produces three main types of outputs for each object in an image:

1. **Class label**, a n-dimensional vector containing the class of each predicted element.

2. **Bounding box coordinates (X, Y, B, H)**, a *4n* dimensional vector containing the image coordinates of each prediction. X, Y are the coordinates of the upper left corner of the bounding box and B is the width and H is the height of the bounding box.
3. **Segmentation masks**, each pixel of the object is classified through a binary mask that indicates the class in which the object belongs, the output of the model is a list of *n* elements, one for each mask.

2.4.3 MASK R-CNN CODE ADAPTATIONS

Mask R-CNN is an open-source project designed for general use in a variety of domains. A specific implementation can be found on a GitHub virtual repository (https://github.com/matterport/Mask_RCNN). This implementation is designed to work with various image formats, including JPEG and PNG. Different research papers proposed modification to Mask R-CNN original source code in order to fit to the specific purposes of the research [119]. This section discusses the changes made to the model to meet the needs of the research in this thesis.

Input format adaptations

The implementation available from the project repository is designed to handle compressed RGB images with 8-bit color depth. This format is very popular for image management on the Internet, since it provides a good balance between quality and file size. However, the purposes of this thesis require the processing of data that allows to exploit the full potential of the monitoring system involved in change detection processes. The goal was to adapt the model to handle single-channel data recorded in 32-bit TIFF (Tagged Image File Format), where each pixel represents a 32-bit floating point value. To overcome this problem, it was necessary to intervene in the input layers of the convolutional kernel. In the original implementation, the latter accepts input in uint8 (unsigned 8-bit integer) format, and then converts it to a floating-point number only in subsequent processing layers.
A second action regarding the input format concerns the image loading function in the neural model. The original implementation of Mask R-CNN involves receiving a three-channel image (usually in RGB or BGR order) that is subsequently treated as a vector of dimensions

[H, W, C] where H is the height of the original image, W is its width, and C is the number of channels associated with the image. For grayscale images (1 channel) or RGB-A images (where an alpha channel is added to the 3 RGB channels), the model is designed to still return an image with three channels by duplicating the grayscale channel or eliminating the alpha channel, respectively. However, this import routine is not compatible with the TIFF data format that was decided to be exploited in this thesis work and requires the definition of a new image loading function. Python libraries for reading images, such as Skimage or OpenCV-Python, produce output in vector form of the type [H, W] that is not dimensionally homogeneous with the input layers of Mask R-CNN. In fact, the latter are encoded to receive an input whose number of channels is uniquely and strictly defined during configuration. In this thesis work, we chose to work with single-channel images, consequently the final format of the input image is [H, W, 1].

Normalization algorithm

In order to allow for faster convergence of neural models, it is common practice to normalize input data. Normalization is essential in machine learning because it ensures that inputs are scaled to a standard range. If inputs aren't normalized, they can land in a region of the activation function that doesn't change much, making learning difficult or even stagnant. Furthermore, without normalization, numerical problems can arise, further complicating the training process. Mask R-CNN's algorithm for image normalization is designed to process 3 channel images. The original process involves extracting the mean pixel value for each channel of the image, and then generating the normalized image by subtracting the mean value to the original value of each pixel associated to the channel and then dividing for the standard deviation. As the difference maps for the training sample only count one channel (which is recorded in 32-bit floating points values for better precision) an alternative normalization algorithm has been integrated that allows the model to operate with values in a range between 0 and 1. For each image, the maximum and minimum values are calculated, and then the contribution of each pixel is defined according to the following notation:

$$normalized[i,j] = \frac{pixel\ [i,j] - \min_{image}}{\max_{\text{image}} - \min_{\text{image}}}$$

This process maintains the same spatial correlations between pixels with respect to the original image. Similarly, the function for returning the original image from the normalized one was defined:

$$denormalized[i,j] = normalized[i,j] * (\max_{\text{image}} - \min_{\text{image}}) + \min_{\text{image}}$$

This function is only used in inference mode as it allows the user to visually review the output of the neural network.

Mask resolution output

A second issue encountered with the current implementation of Mask R-CNN concerns the size (in pixels) of the output masks of the Mask branch. Specifically, the branch used for estimating segmentation masks is programmed by default to produce an output of 28 x 28 pixels within the detection box. This low-resolution mask is then dynamically upscaled according to the size of the bounding box in which it is enclosed. In this way, the training procedure of the classifier is less onerous (i.e., fewer parameters are involved in the process) and the estimation at the prediction stage is found to be faster. The accurate definition of the collapse areas needs the resolution of the output masks to be as high as possible while maintaining computational feasibility. It was decided to add two additional convolutional layers in the mask prediction branch to increase the output resolution by two times the original resolution. This gives the model the ability to produce masks of size 56 x 56 pixels, which are useful for better approximation of the contours of the detachment niche. It is important to point out that while using a machine with enough memory the user can increase the resolution of the output beyond this threshold by adding several additional convolutional layers to the mask branch network.

2.4.4 NEURAL NETWORK DATA FORMATTING

The format of both input and output data plays an important role in the training and the evaluation of the neural model. Ensuring that data is structured consistently across the pipeline can significantly streamline model training, inference, and subsequent tasks.

For training purposes, the input data is formatted to align with the style of the output data. Specifically, it employs the COCO (Common Objects in Context) [120] annotation framework, an industry standard in machine learning and computer vision tasks. The reason for adopting the COCO annotation style is its ability to robustly define object locations through bounding boxes and to facilitate semantic segmentation via object instance masks. The input data comprises high-resolution images along with corresponding annotations summarized in a JSON file. This file includes a structured collection of dictionaries and arrays, encompassing metadata such as image identifiers, object categories, and bounding box coordinates, similar to the output data.

The output data from the Mask R-CNN model is also generated in a structured JSON file which strictly adheres to the COCO annotation style. Such a format encompasses key elements like image identifiers, object categories, and more critically, the segmentation masks which are encoded using Run-Length Encoding (RLE). The selection of RLE for mask encoding is motivated by the necessity for computational efficiency; it allows the model to compress the binary mask information into a more manageable size, particularly useful when dealing with high-resolution imagery.

The symmetrical formatting between input and output data not only augments computational efficiency but also eliminates any need for data transformation or conversion steps in between the training and evaluation phases. This unified approach thus enables seamless integration with analytical pipelines and facilitates compatibility with a variety of evaluation metrics such as Intersection over Union (IoU), Precision, Recall, and F1 score which will later be introduced and explained for the model evaluation.

3 CHAPTER 3 – ROCKFALL SIMULATOR

3.1 INTRODUCTION

The simulation software was structured with the primary aim of replicating the steps of the photogrammetric process for a fixed monitoring system. In this thesis the application of the software focuses on the simulation of collapses on mine walls, however it could be utilized for various other purposes, as depicted in section 1.1, and therefore may provide a useful time-saving task when employing neural networks to detect changes in other different fields. The software package was developed in C# language.

The simulation software produces (simulated) difference maps in raster format between the models of a reference epoch and those of subsequent epochs where rockfall events occur, knowing exactly all the event characteristics. Such difference maps are the input data for training the neural network used to automate the detection and evaluation of the rockfalls over time.

This chapter describes the generation of a 3D model featuring different collapse events. The simulation starts with the generation of a set of tridimensional rock blocks, which are then "subtracted" to a copy of the original (reference) mesh producing detachment niches on the new (final) mesh.

Once the wall collapse simulation is complete, the simulation of the photogrammetric pipeline is performed with the aim of reproducing all the noise sources that affect the 3D reconstruction and therefore to produce a realistic dataset.

Rockfall Simulator photogrammetric pipeline, depending on the user's needs, can employ the three-dimensional reconstruction algorithms of Agisoft Metashape, or rely on purposedly-implemented original algorithms. Regardless of the method, the output is a simulated mesh.

The first option uses the reconstruction routines from Agisoft Metashape, one of the most popular commercial software in the engineering photogrammetry sector. Leveraging Metashape's algorithms, in this stage the simulation software produces an intermediate product of the 3D reconstruction pipeline, specifically the depth maps, as they were computed using dense matching algorithms. Depth maps encode three-dimensional information in a raster format, where each pixel represents the distance from the camera's acquisition center to the object. After depth map simulation the remaining reconstruction stages are performed using Metashape. Although the user is not completely aware of the operations carried by the reconstruction algorithms in Metashape, as the developer clearly does not share algorithmic implementation details adopted in the software, in this way the processing pipeline is very much similar to the one adopted in a normal photogrammetric application relying on commercial software.

The second option, to the contrary, allows for a complete simulation of the photogrammetric process (dense matching and tridimensional reconstruction), with some simplifications (e.g., in the depth map fusion) w.r.t. Metashape's workflow. However, in this case, the user can keep track of every single detail on each phase of the reconstruction pipeline.

The primary reason for using three-dimensional data to generate difference maps for collapse identification, instead of trying simulating new images of the detached blocks and run all the photogrammetric pipeline from the very beginning, is the actual difficulty of simulating realistically the images of the collapse. The identification of collapse phenomena on three-channel (RGB) digital images may present considerable problems due to different factors such as illumination (and consequently on the presence of bulky shadows), reflections, and so on. The presence of areas that are totally in shadow (or, conversely, highly reflective) [121] could sometimes totally preclude the extraction of significant features for the identification of detachments. Starting the process from 3D data can overcome these limitations and potentially improve the accuracy and reliability of automatic identification procedures.

Though for sake of simplicity it will be referred to as measurement noise, the uncertainties that affect all steps of the photogrammetric process - specifically not just the identification and position measurement of corresponding points but also the determination of interior and exterior orientation parameters - must be considered in the simulation process to produce realistic data. Uncertainties in the orientation parameters cause systematic effects that affect all tie points object coordinates [122]. Though in principle the effect of each parameter can be computed and studied separately, the reverse is hardly possible. One may expect the effect of image matching uncertainty to be specific to each tie point pair, being in principle equivalent to a measurement by the (human) operator. However, this depends on the type of image matching algorithm and is today safer to assume this stage to cause both random and pseudo-systematic effects as methods like the Semi-Global Matching (SGM) [123] [124]

don’t work with a local/individual matching strategy as, for instance, the least squares image matching [125] [126].
The following sections will provide an overview of the different routines implemented in the Rockfall Simulation software.

3.2 INPUT DATA

The input data required to start the simulation process is the reference epoch image block (for instance the user can provide a Metashape project with images already oriented and calibrated. The objective of this input phase is to obtain information on the project reference system and the calibration of the optics of the cameras of the monitoring system. Rockfall Simulator interacts with Metashape's API (Application Programming Interface) using Python language. Using a Python script, the following characteristics of the image block are exported:

- Sensor characteristics (including camera model distortion coefficients).
- EO parameters of the cameras of the fixed monitoring system.

This information is used to generate a new Metashape project (i.e., a simulation image block). Opting to create a new project from scratch allows to define a new reference system better suited for the spatial localization of blocks within the three-dimensional model. The mesh obtained in this initial step is treated as ground truth data, which will be the basis of the subsequent collapse simulation and photogrammetric process simulation.
The initial mesh (i.e., the ground truth mesh) is given in the Metashape project reference system (usually a local topographic reference system). To make the subsequent processing stages simpler a geometric transformation is computed such that the mean plane of the wall will lie in the XY plane of the new reference system. An example of such procedure is shown in the following images. Please note the directions of reference axes in the lower-right side of the image.

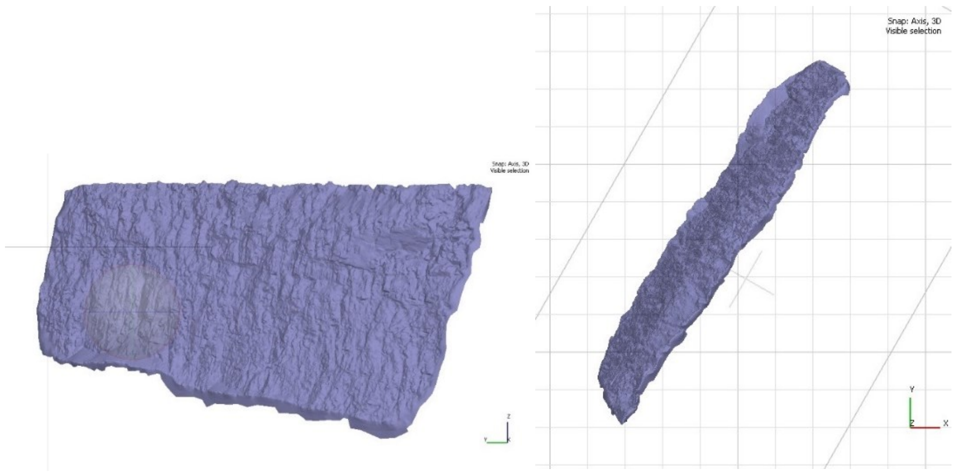


Figure 3-1: Front view of the original mesh (left), orthographic view of the original mesh in the original reference system (right).

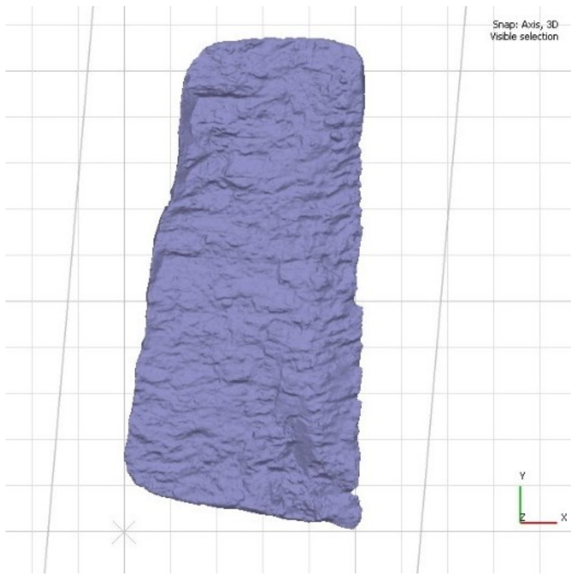


Figure 3-2: Orthographic view of the original mesh in the new reference system.

The origin $V_0 = [0\ 0\ 0]^T$ of the new reference system is placed in the lower right vertex of the mesh in the original reference. Images 3-1 and 3-2 below illustrate the orthographic view of the mesh in the original and transformed reference system:

The transformation matrix is expressed as:

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where r[i, j] are the rotation matrix ***R*** elements [139]and t[i] the translation vector ***T*** components.

The software first estimates the best-fitting plane using all the reference mesh vertices by the Singular Value Decomposition (SVD) method [127].

At the end of this step, the mesh is conveniently oriented for the rasterization process and for extracting the coordinates of the simulated collapses.

3.3 ROCK COLLAPSES SIMULATION

The input data for the block simulation procedure is the reference mesh, i.e., the ground truth mesh oriented in the new reference system (i.e., approximately parallel to the XY plane). All difference maps at different simulated epochs will be produced from the reference one. Providing a safe zone for the simulation of collapses along the mesh border is desirable, in order to avoid placing simulated detachments on the edges of the mesh, often critical areas in 3D reconstruction processes. Moreover, photogrammetric software, such as Metashape, uses interpolation techniques to fill empty areas of the mesh [128].

Figure 3-4 shows the hole-filling technique implemented in Metashape. Neighboring data are used to interpolate missing information. The software first identifies the edges of the holes and then uses geometry and texture information from the surrounding area to fill the gaps. Obviously, hole-filling techniques may not produce accurate results with large data gaps. On the edges of the 3D model such interpolation can oversimplify the actual shape of the model.

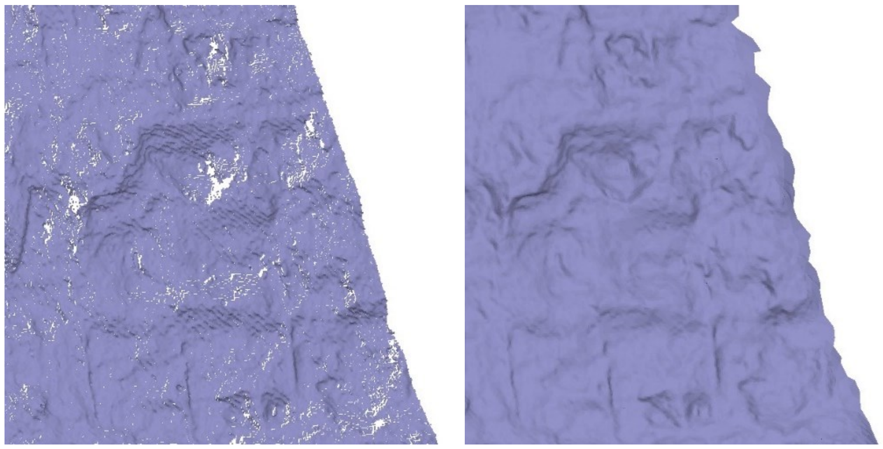


Figure 3-3: lower-right border of the original mesh left: no hole-filling applied; right: hole-filling enabled.

Figure 3-3 shows the lower-right border detail in the original mesh reconstruction quality. The mesh on the left has been produced by disabling any interpolation technique during the 3D reconstruction while the image on the right has been processed by enabling the interpolation on the mesh vertices. The completeness of the model is crucial at this point of the work, but the border information retrieved mainly by interpolation is unreliable. To overcome this issue, it was decided to exclude these areas from the simulation of the rock collapses on the wall. To this aim, a safe zone on the ground truth mesh is introduced through binary rasterization of the original mesh. Each pixel of the rasterized mask will assume an intensity value of 0 (black) if the pixel does not belong to the tridimensional mesh area while the safe area will be represented by pixels of intensity of 1 (white). The mask generated by this procedure spans the entire area of the mesh, including the interpolated edges. To effectively erode the edges of the mask (and thus shrink the effective area over which the simulation process takes place) it is necessary to impose an offset on the edge of the original mesh.

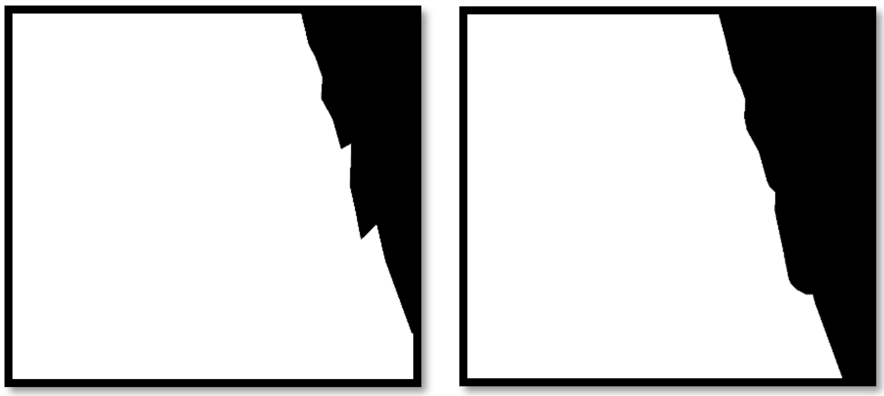


Figure 3-4: original model mask (left), eroded safe zone (right)

Figure 3-4 graphically illustrates the result of a portion of the mesh edge. The white portion of the image represents the binary mask associated with the mesh surface. As it can be observed from the right image, the edge is eroded according to a predefined offset value.
The simulation of the collapses on the original mesh (i.e., the ground truth mesh) is performed generating a number of blocks chosen by the user. First the blocks meshes to be overlapped to the rock wall mesh are generated. The method generates the vertices and triangles that create a rock block tridimensional mesh (currently two different shapes can be selected: cubic or spherical). The coarse block resulting from this procedure must be appropriately transformed to have irregular shapes and sizes. For this reason, it is subjected

to a shape transformer that operates according to three different criteria, respectively applied in cascade:

1. **Scaling factor**: the original block can be scaled in the three dimensions (length, height, and depth) according to random values extracted from a probability distribution (e.g., Gaussian). In this way, blocks of different sizes and proportions can be produced.
2. **Rotation factor**: the original block is rotated randomly around each axis (X,Y,Z) by an angle randomly extracted from a probability distribution to simulate different orientations in space.
3. **Surface irregularity factor**: this module adds random offsets to each vertex of the block, according to a probability distribution. The block therefore is not any more a simple parallelepiped or a sphere.

To produce a simulated rockfall event each block must then be placed on the rock wall, in such a way that when it is removed a cavity will form in the wall itself. The block center of gravity is initially placed in a randomly generated position X and Y, bounded within the original mesh extent.

The triangles of the wall mesh falling in the horizontal contour of the block projected over the wall mesh are searched for their maximum and minimum Z coordinate. The block altimetric offset is then calculated as:

$$offset_Z = -\frac{Block\ max_Z + Block\ min_Z}{2} - Mesh\ \mathrm{max}_Z$$

where Block maxZ and Block minZ are the maximum and the minimum Z coordinate of the block vertices respectively and Mesh maxZ is the maximum Z coordinate of the mesh vertices in the overlap area between the block and the mesh. With this height offset, the block center of gravity is placed at the point of maximum wall relief of the intersection area.

The block location is validated by checking whether the calculated offsets are compatible with the safe zone or not. The coordinates of the blocks bounding box (rather than the detachment niche contours) are inspected. If at least one pixel inside the bounding box region

has value 0 (and thus is not contained within the safe zone), the block is eliminated, and the process is repeated.
Once this first placement check has been completed, the block removal operations on the 3D mesh begin by computing the intersection between the two meshes (the rock wall and the detachment mesh).

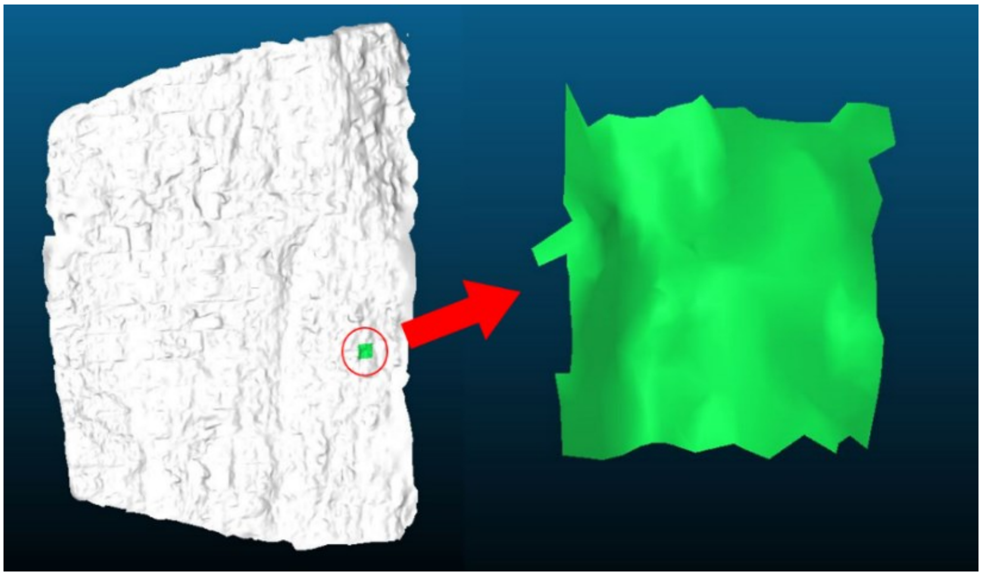


Figure 3-5: Rockfall bounding box region detail on the original mesh.

The operations performed result in four new meshes:

1. **Mesh difference**: is the portion of the rock wall with the hole generated by the intersection with the simulated block.
2. **Intersection mesh**: is the portion of the mesh that is removed by the intersection with the simulation block. The union of the intersection mesh and the difference mesh is the initial rock wall area.
3. **Block difference**: is the portion of the detachment block inside the wall, i.e., the detachment niche. Figure 3-6a shows the location of the lower portion of the block, which fills the gap obtained by removing the intersection mesh.

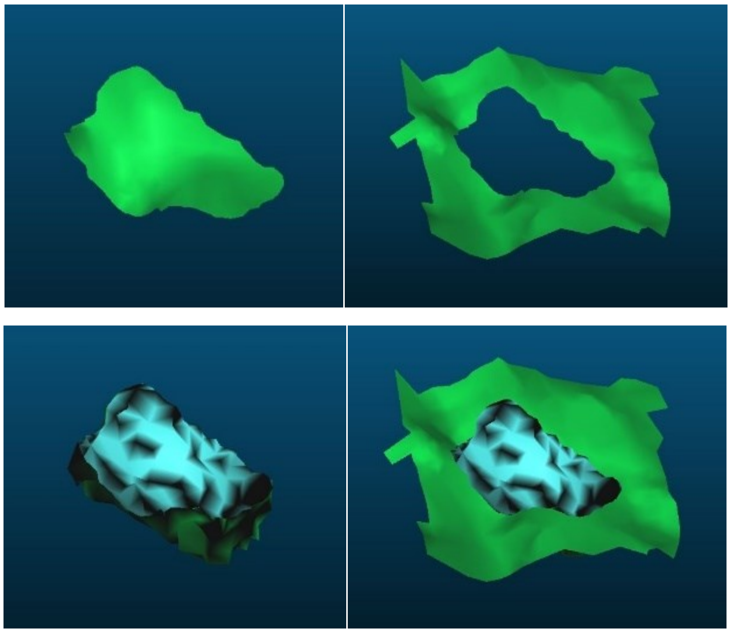


Figure 3-6: a) Mesh difference, b) Intersection mesh, c) Block difference, d) Final result

4. **Removed block**: this is the portion of the simulated block protruding from the wall. As it does not contribute directly to the creation of the final detachment niche, it plays no role in the subsequent interaction steps and can therefore be discarded.

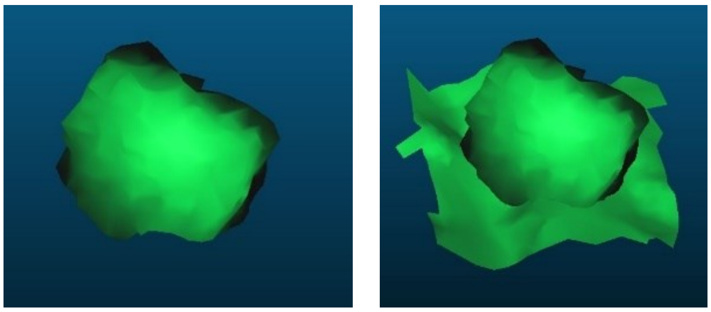


Figure 3-7: Removed block.

The process is repeated until the number of blocks defined by the user is reached. An additional check on the position of blocks is performed to prevent two or more blocks from intersecting, by comparing the horizontal coordinates of the bounding boxes, to which an offset value is added: in this way it is also verified that the blocks are not too close to each other. This procedure results in a list of simulated blocks that are defined by their spatial position on the rock wall.

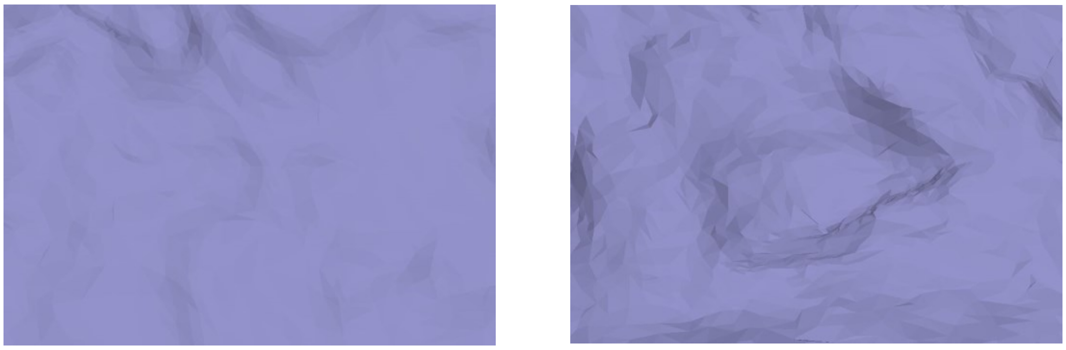


Figure 3-8: left) Original mesh, right) example of a simulated rockfall event on the original mesh.

The green textured space on the figure is the original mesh surface, while the blue/black area is the simulated rockfall event on the original mesh. The result of this procedure is the ground truth mesh with the simulated detachment niche for each collapse. This mesh will be subsequently processed by the photogrammetric simulation algorithm.

3.4 PHOTOGRAMMETRIC PIPELINE SIMULATION

The objective of this step is to simulate the dense matching pipeline in the photogrammetric workflow. This output a photogrammetric mesh (both for the reference epoch and the following epochs), which is affected by noise sources in its generation process. The first stages of the dense matching simulation are the same, while the final tridimensional reconstruction is processed by Agisoft Metashape algorithms (Method 1) or, alternatively, by Rockfall Simulator (Method 2). In the former case a black-box tool is used, since Metashape performs 3D reconstruction leveraging depth information stored in depth maps. In the latter one the user can interact and inspect the dense matching simulation in each major step.

The following diagram summarizes the key steps of each of the two alternatives implemented in Rockfall Simulator.

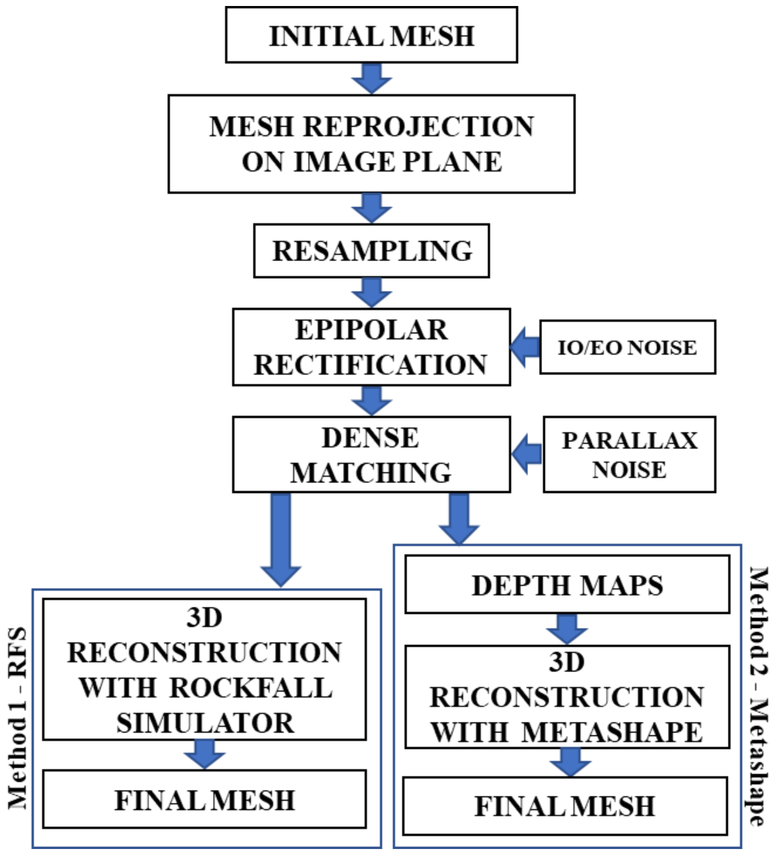


Figure 3-9: Rockfall Simulation photogrammetry simulation pipeline.

The simulation is carried both for the reference epoch and for the following ones. Simulating photogrammetric noise in each epoch assures consistency of the data (as the comparison products are both sourced by the same processing pipeline) and repeatability in the comparisons of the different photogrammetric meshes.

Initially, the software retrieves the original image planes from the dual-camera photogrammetric system to allow for mesh reprojection. The goal here is to generate two images (mimicking a new photo acquisition) showcasing the rock wall after the rockfall event, to replicate the dense matching pipeline.

In other words, each vertex of the initial mesh gets reprojected onto the image plane. This is done using the collinearity equations, using Interior (IO) and Exterior Orientation (EO) parameters. Using distortion parameters, distorted image coordinates are determined. This mimics the real-world scenario where images taken by cameras undergo lens distortion. In this stage the IO, EO and distortion parameters noise is applied. Noise simulation is executed separately for IO and EO, although the methodology remains consistent. An external function manages the probabilistic model, allowing the user to generate random measurement noise values from a gaussian distribution. The user may specify the mean and standard deviation of the probability distribution for each parameter included in the noise simulation (6 parameters for exterior orientation and 11 parameters for interior orientation and Brown-Conrady's camera distortion model). The selection of a Gaussian distribution is motivated by empirical observations which highlight that non-systematic measurement errors frequently manifest as a bell-curved pattern, suggesting a Gaussian or near-Gaussian behavior.

The two meshes reprojected on the image planes of the stereo-pair, represent a sort of image of the rock wall. However, it is important to understand that only the vertices coordinates are projected on image plane: no texture or RGB information are used in these stages. The stereo image pair is then epipolar rectified.

This process modifies the images from both camera perspectives, ensuring that corresponding points on both the new images align on the same row, thereby simplifying parallax computation. These operations have been performed with the OpenCV library routines, that need the intrinsic matrix K for each camera as well as the distortion parameters vectors and the relative rotation and translation matrices. The intrinsic matrix contains information about the camera's internal characteristics such as focal length and principal point position.

$$K_i = \begin{bmatrix} f_i & 0 & c_{x_i} \\ 0 & f_i & c_{y_i} \\ 0 & 0 & 1 \end{bmatrix}$$

where f is the focal length of the i-th camera of the stereo pair and c_x and c_y are the coordinates of the principal point of the respective camera.

Similarly, the distortion coefficients of both cameras are stored in a distortion vector.

$$dist_i = \begin{bmatrix} K1 \\ K2 \\ P1 \\ P2 \\ K3 \end{bmatrix}$$

where K_i are the radial distortion coefficients and P_i are the tangential distortion coefficients of each camera of the stereo pair. To apply the rectification, the relative rotation R and the baseline vector T between the two camera centers are necessary.
The rectification transform is processed for each camera and is identified by a 3x3 rotation matrix R that brings points given in the unrectified coordinate system to points in the rectified camera coordinate system. The new (rectified) camera matrices (P) are as follows:

$$P1 = \begin{bmatrix} f & 0 & c_{x_1} & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P2 = \begin{bmatrix} f & 0 & c_{x_2} & T_x * f \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Where T_x is the horizontal shift between cameras. The disparity-to-depth matrix is calculated as follows:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_{x_1} \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & \dfrac{c_{x_1} - c_{x_2}}{T_x} \end{bmatrix}$$

At the end of this stage, a new epipolar (rectified) image pair is obtained.
Given the estimated positions of the corresponding points (i.e., the projections of the mesh vertices) on the new epipolar image pair, it is possible to calculate the parallax values.

At this stage in the dense matching simulation pipeline, a second photogrammetric noise source is introduced: the parallax matching noise. The simulation of matching noise is carried out with two different contributions:

1. **Pixel noise:** which directly simulates a parallax error [151].
2. **Regularization noise**: which simulates the errors characteristics of Semi Global Matching algorithms.

Both noise contributions are governed by a distribution (gaussian) with zero mean and standard deviation to be specified by the user.
The parallax noise is calculated pixelwise for each parallax value.
The regularization term in the SGM can introduce local systematic effects that affect a certain group of adjacent pixels. To simulate this, a field of random values across the entire parallax map is considered: a grid over the entire parallax map with a step that can be set by the user is first set up. The noise value at each grid node is randomly extracted from a Gaussian distribution. The values to be assigned to each point on the parallax map are then calculated through interpolation as SGM algorithms handle image regions with low informational content, such as those caused by occlusion or significant contrast variations, by utilizing the values from neighboring depth map elements to estimate the undefined regions.
Finally, the function computes a parallax value for each of the vertices of the original mesh. At this stage, the pipeline splits into Method 1 and Method 2 previously described. The former requires the computation of the depth maps that are subsequently loaded in the original Metashape project (i.e., the project with the original EO and IO parameters). The output resolution of such depth maps produced with Rockfall Simulator can be set to any of three levels (which are proposed based on the Metashape proprietary settings), namely:

- **High quality**: the output depth map has half the resolution as the original frame.
- **Medium quality**: the resolution of the depth map is 1/4 of the original frame.
- **Low quality**: the depth map resolution is reduced to 1/8 of the original frame.

The final 3D reconstruction is performed using Metashape. The latter method reprojects directly the estimated position of the corresponding points in object space generating the reconstructed final mesh. Regardless of the chosen method, the result is a photogrammetric mesh that is then rasterized to compute the final raster difference maps between the first epoch and the i-th epoch.

3.5 LASER SCANNER PIPELINE SIMULATION

As alternative to photogrammetry, Rockfall Simulator can simulate a mesh from laser scanning. The starting point is the same original mesh of wall (reference) from which a simulated point cloud is extracted, which is later processed into a tridimensional mesh. Within this framework, the simulation is carried both for the first epoch tridimensional mesh (which is the reference epoch when producing raster difference maps) and for each later epoch mesh. The following flowchart resume the steps for the laser scanning simulation pipeline.

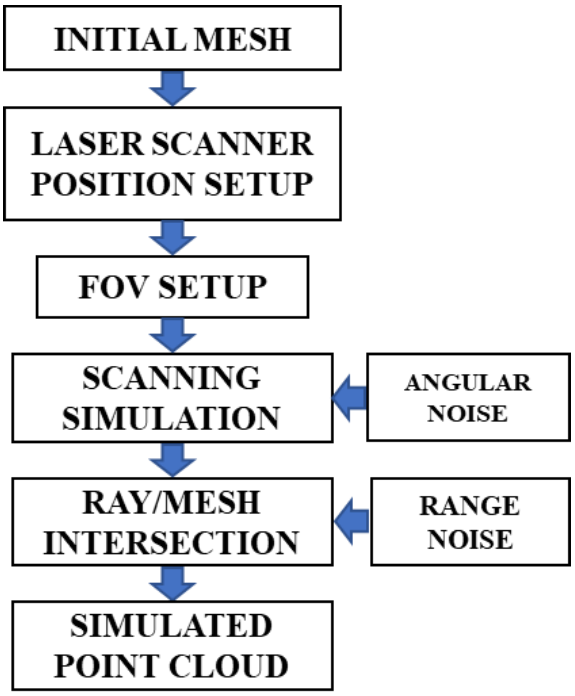


Figure 3-10: Rockfall Simulator laser scanner simulation pipeline.

Initially, a virtual laser scanning station position and orientation are set according to positional parameters that can be defined by the user. At this point the 3D mesh is spatially oriented as described in section 3.2.
The determination of the scanning window is based on the laser's Field of View (FoV) and the dimensions of the target 3D mesh. By defining the scanning window, the software sets the angular limits the laser scanner will operate within and the spatial and angular resolution parameters, which directly affect the quality of the data collected.
Scanning initiates at a specified angular step, effectively scanning the continuous space of possible laser ray directions. During this process, the noise intrinsic to the angular measurement of the device (the azimuth and the zenith angles of the laser rays) is introduced into each scanning direction. In this step, angular noise is simulated with the same criteria illustrated for the photogrammetric pipeline simulation, which involves the random extraction of simulated noise values from a known zero-mean gaussian distribution. For each emitted laser ray, the intersection with the 3D mesh is computed to measure the range, i.e., the distance between the scanner and the ray-mesh points of intersection.
The measured range is then corrupted based on a specific noise model for distance measurements in laser scanning.
Finally, using the nominal scanning direction (without angular noise) and the noisy range (inclusive of distance noise), the three-dimensional coordinates of the intersection point are calculated. This iterative procedure results in the generation of a point cloud, which is subsequently processed to create a simulated mesh. This combination of methods ensures that the synthetically generated data is as representative as possible of actual operational conditions.
At the end of this simulation step, a simulated mesh is produced, both for the reference mesh and for the following epochs meshes. For this work of thesis, it was decided to only rely on photogrammetry-derived data.

3.6 MESH CO-REGISTRATION

To obtain the best co-registration between reference and current epoch, an ICP alignment [130] can be executed prior to mesh rasterization and computation of the raster of differences. An Iterative Closest Point (ICP) algorithm has been implemented in Rockfall Simulator. For this purpose, the ground truth mesh acts as the reference for the co-registration procedure executed on the rockfall mesh. Rockfall Simulator provides several settings, thereby a tuning procedure is useful to get the best results. The ICP parameters implemented in Rockfall Simulator are the following:

1. **MaxIteration**: sets the maximum number of iterations for the ICP algorithm, whether or not an optimal solution has been found.
2. **OutlierDistanceThreshold**: sets the distance threshold for labelling outliers. Points in the moving mesh (i.e., the mesh which is being registered to the reference data) that are farther away than this threshold from their nearest neighbors in the reference mesh will be considered outliers. By adjusting this parameter, the user should take in consideration the noise levels expected in the photogrammetric block.
3. **OutlierThresholdDecayIteration**: indicates how many iterations the algorithm should wait before reducing the outlier threshold. If set to 1, the threshold is used from the first iteration.
4. **RemoveOutlier**: indicates if outliers' rejection is used or not.
5. **Threshold:** threshold for determining when the algorithm has converged, i.e., when the improvement (reduction in error) between iterations is less than the setup threshold.
6. **DeltaThreshold**: minimum change in residuals between iterations. If the change is smaller than this value, the algorithm assumes it has converged and stops.
7. **SubsamplePoints**: indicates whether to use a subset of points from the moving mesh for the computation. Typically, this approach is employed to expedite computational processes.
8. **SubsampleRatio**: percentage of points to be excluded from computation of the transformation when subsampling is enabled.

During the co-registration vertices having Not a Number (NaN) coordinate values where found, apparently due to the failure of the rock collapses simulation algorithm, in some cases, at computing the triangles vertices when subtracting the blocks from the original mesh surface. This is attributed to the libraries employed for manipulating the mesh products. Indeed, the procedures for mesh intersection have been found to generate the issues shown in Figure 3-11 along the perimeter of the block/mesh intersection:

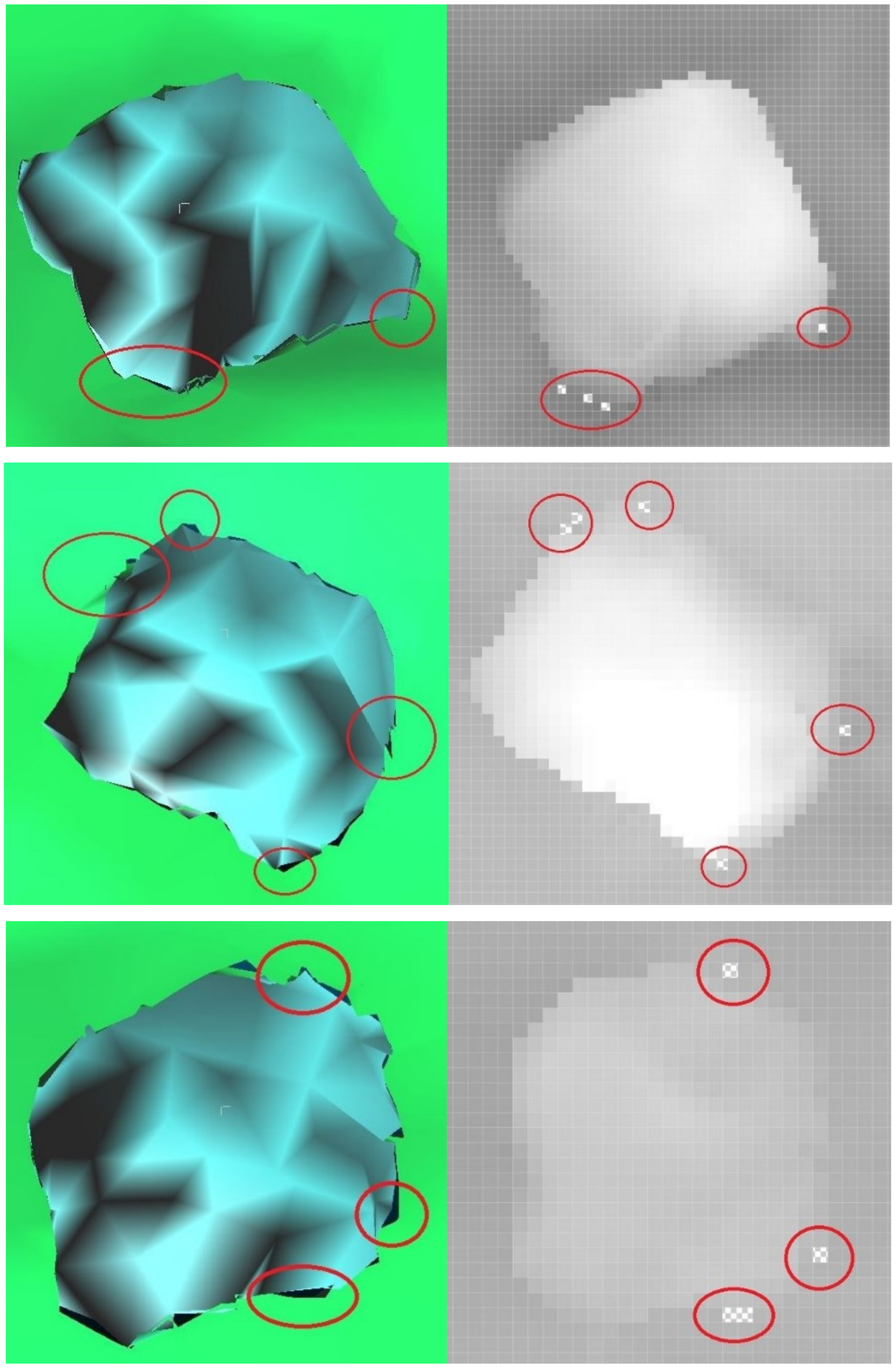


Figure 3-11: Representation of the NaN values observed in the rasterized difference maps obtained from the errors in the intersection of the meshes.

The red circles in Figure 3-11 highlight the incorrect mesh vertices on the left, while the images on the right highlight the resulting issue on the rasterized difference map. Such vertices often create distinct discontinuities within the mesh, which can result in NaN values on the rasterized difference map. To address this problem, a routine has been incorporated into the simulation code to weld erroneous edges, eliminating NaN and duplicate vertices within the mesh.

When exploiting the ICP algorithm incorporated with Rockfall Simulator, particular attention was given to the OutlierDistanceThreshold (2) parameter and the Subsample Ratio (8) parameter, as these are anticipated to yield the most significant improvements in the parameter set. Specifically, the Subsample Ratio is considered critical due to its direct impact on the selection of points for calculating the transformation with the ICP algorithm. By initially manipulating the ratio, the algorithm reduces the numerosity of the sample. Figure 4-38 represents the Root Mean Square Error (RMSE) of the fit for the variation of the Subsample Ratio parameter in the interval from 0 to 0.99 in the application of the Iterative Closest Point (ICP) algorithm.

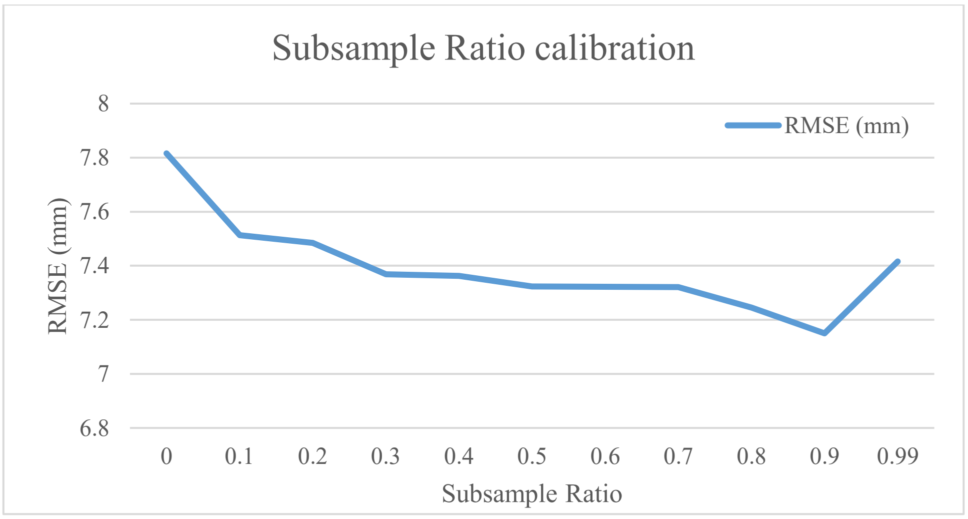


Figure 3-12; Subsample ratio parameter variation results chart.

Examination of Figure 3-12 shows that as the subsample ratio decreases (i.e., less points are being used), the RMS error generally increases. This result is as expected and underlines the benefits of using a larger number of points from the dataset in achieving more accurate results. However, for intermediate values of the ratio, the RMSE is constant, an indication that savings in computing times using fewer points are possible. Of course, these findings might be specific to this dataset, and similar studies should be conducted for other datasets or different use cases to determine the most effective subsample ratio for each specific situation.

A second assessment has been carried for the OutlierDistanceThreshold, which defines the distance of the nearest neighbors on the moving mesh that are labelled as outliers during the ICP computation. This means that the ICP algorithm discards all the points which fall above this threshold when computing the transformation. When calibrating this parameter, it is crucial to verify the expected levels of measurement noise. In order to search for the best threshold, it was decided to initially compute the transformation with an outlier threshold with the same value of the photogrammetric noise calculated (i.e., sigma = 3 cm) during the noise assessment carried in section 4.1.3.

The following chart represents the results of the calibration of the outlier rejection threshold.

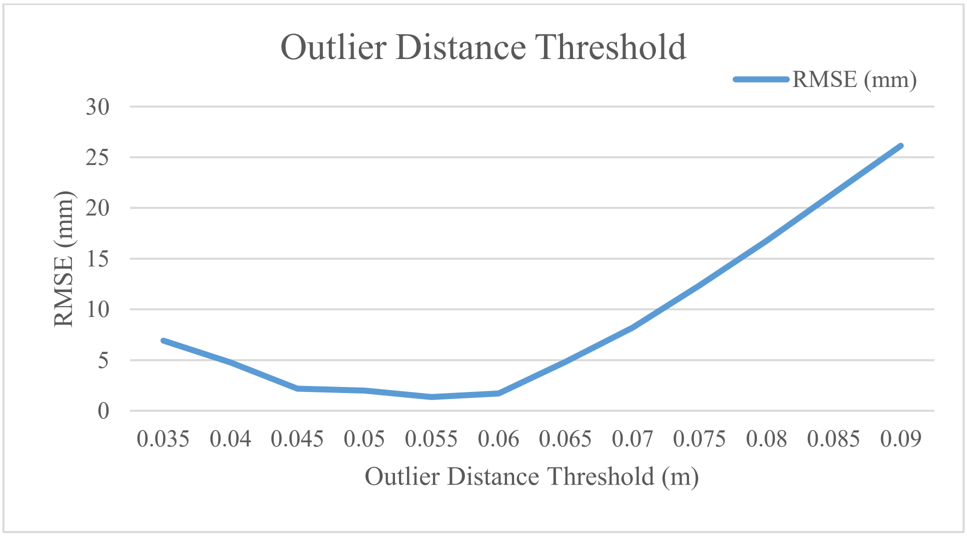


Figure 3-13: Outlier Distance Threshold results chart.

The above chart provides the Root Mean Square (RMS) error in millimeters for various OutlierDistanceThreshold settings of the Iterative Closest Point (ICP) algorithm (expressed in m).

Initially, the RMS starts at a relatively higher value of roughly 7 mm at an OutlierDistanceThreshold of 0.035 m. As the distance threshold increases to 0.05 m, the RMSE exhibits a noticeable decline, reaching its lowest point at 0.055 m with a RMSE value of around 1.5 mm. This suggests an optimal point for the threshold parameter where in the algorithm performs best. It would be reasonable to assess that at this particular threshold, the model is able to reject noise and erroneous correspondences effectively, thereby providing a transformation that yields the smallest discrepancy between the aligned point clouds.
Beyond this minimum, the RMS begins to increase steadily, exhibiting a convex behavior until it reaches 25 mm at a threshold of 0.09m. This ascent indicates a progressive deterioration in the accuracy of the ICP algorithm, likely caused by including outliers as the threshold value increases.
The non-linear behavior of RMS across the range of OutlierDistanceThreshold values implies that there exists a trade-off between robustness against outliers and the retention of true correspondences. Setting the threshold too low might make the algorithm too selective (i.e., discarding valid information and reducing too much the observation dataset) while setting it too high might result in including too many outliers, subsequently affecting the alignment's accuracy.

3.7 RASTERIZED DIFFERENCE MAPS

The final step of the simulation involves the generation of the difference maps between the reference epoch and the next ones, produced at a user-specified GSD. The method starts by extracting the bounding box of the reference mesh in order to create a new bidimensional bounding box that covers the same area. The reference and the simulated photogrammetric meshes are then rasterized within the limits of the bounding box. This is done to remove areas near the edges of the meshes, where the mesh data may be incomplete or noisy. The difference map is finally obtained subtracting (pixel by pixel) the reference and the simulated rasters:

$$diff_{final}[i,j] = Reference[i,j] - Epoch_k[i,j]$$

where Reference[i, j] is the pixel value for the reference raster and Epoch $_K$[i,j] is the pixel value for the k-th epoch raster. This process can be iterated to a large number of samples (each one referred to a specific simulation epoch), which can be defined by the user based on the needs, in order to produce realistic and numerous datasets for training neural network models.

The resolution of difference maps is a crucial parameter, especially when training deep neural networks. Such networks, given their potential to have millions of weights, demand vast system memory during training. Working with smaller images is a common strategy to tackle the problem. However, the loss in resolution can be detrimental. To retain information without blowing up computational demands, a solution is to further subdivide the difference raster resulting from mesh comparisons. An algorithm has been devised for this purpose, which breaks down the full raster map into smaller, more manageable tiles.

3.8 Sample annotation

In order to complete the dataset information for neural network training (and validation), the raw image data (i.e., the difference map) needs to be labelled and categorized. This procedure is generally carried out by a human operator and requires expertise and, most of all, time. Annotations are essential for neural networks as they allow to associate labels and information to input data in order to produce the desired output.

The coordinates of each simulated rockfall block are stored in memory in order to be logged in an annotation file. One of the most popular annotation models for object detection and instance segmentation is Microsoft COCO annotations [120] which is based on the style of annotations presented for the Microsoft COCO dataset. In a COCO annotation file each object is represented as a JSON object. The JSON file for COCO-style annotations has the following structure:

1. **Image_id**: the unique image ID in which the object is recognized.
2. **Category_id**: the ID of the category the object belongs to (in this case only two categories are defined: a background class and a block class. All blocks share the same category.
3. **Bbox**: the image coordinates of the bounding box [X, Y, B, H] surrounding the object.

4. **Area**: the number of the pixels belonging to the mask of the object.
5. **Iscrowd**: A Boolean value which points out whether the object is a single object (0) or a group of objects (1). As a verification check in the niche-generation controls that no simulated blocks mutually intersect, , Iscrowd will always be 0 in this work.

To compress the representation of the binary masks used for segmentation tasks, the **Area** contribution of the COCO structure is expressed as **Segmentation** instance. The encoding is carried out using RLE algorithms (Run Length Encoding) [131] which is a more compact and efficient methodology to provide and store binary mask data using a string of characters). RLE encoding describes for how long a certain color appears in a sequence of rows (or columns) in an image. The segmentation instance is expressed as:
1. **Size**: which defines the height and width of the binary mask image.
2. **Counts**: which is a list of integers that represents the encoding of the mask in the RLE format.

An automatic annotation routine has been developed to extract the fundamental information from the rock collapse simulation process. This method is iterated on each element belonging to the simulated blocks list.
First, the boundaries of the intersection mesh for each element (reported in section 3.3) are extracted and stored as the bounding box coordinates in the COCO JSON file. Subsequently, a binary mask for the intersection mesh is created and stored as RLE mask in the JSON file. Notice that the annotations need to account for the simulated noise introduced during the process and that change the position of the reconstructed rockfall. Alongside with the effect of noise on the rockfall coordinates, also the ICP algorithm introduces a roto-translation applied to the rockfall meshes. Its effect is accounted as well for annotation accurate localization.

4 CHAPTER 4 – SOFTWARE CALIBRATION AND EVALUATION

The need to monitor and predict geological events has been a driving force for the evolution of photogrammetry, especially in close-range applications. The evaluation of small-scale phenomena of engineering interest has driven the development of different solutions in photogrammetry. More specifically, it has catalyzed the development of increasingly efficient monitoring systems, which are paramount for risk assessment for phenomena such as rockfalls. One key example of such solutions is fixed photogrammetric systems, which allow automated and high-frequency surveys. The efficiency of photogrammetric systems in geo-sciences has been pointed out considering the strengths and weaknesses of such methods [132] [133] [133] [19]. A specific topic regarding geomorphological analysis is the identification of rock deformations and rockfall events [134] [135] [136]. These geological phenomena pose a significant threat to human safety, and the techniques used for their detection and analysis have undergone significant changes in recent years, especially considering the constant growth of interest toward neural networks, which have been applied to a wide variety of scientific fields in the last decade. Currently, various lines of research have been proposed that leverage the power of neural networks to process different products of topographic nature, such as point clouds. Among these approaches, the neural network PointNet certainly stands out when dealing with point cloud classification tasks [137] [29].

The approach chosen for this thesis explores the possibility of identifying collapses on digital raster data using the acquisition capabilities of fixed dual-camera systems of photogrammetric nature. Raster differencing provides significant operational advantages in the processing of multi-temporal surveys of the same object of interest. Developing a raster of differences between two (or more) successive epochs allows for appreciating the variations that the excavation walls may undergo during the survey period.

It is necessary, however, to consider that the stability (both geometric and optical) of the acquisition system is of vital importance in producing high-quality raster data. This is due to the possible variations in the camera configurations during the acquisition process, which could lead to distortions that may affect the procedures for recognizing rockfalls.

In this context, a primary objective of the thesis is to find an ideal configuration that allows for a reliable simulation of training data. This data is then used by the neural network to identify and classify collapses on the rock wall.
In this chapter, the methodologies for testing the simulation software will be initially discussed, which should provide an insight whether the simulation software is able to produce results comparable to those obtained through other photogrammetry commercial software. For the purpose of comparison, Agisoft Metashape was chosen as the reference software. This is one of the most widely used solutions in the field of photogrammetric restitution and is also the primary processing software employed by the Geomatics group at the University of Parma.
Similarly, the focus is on determining the working parameters for calibrating the simulation software, using available data collected from the monitoring system during the testing phase at quarries located at different sites. In this sense, the main characteristics of the fixed monitoring system deployed for the monitoring of these sites [138] will be illustrated.
The main aspects to consider is the determination of operational thresholds concerning measurement noise, and the measurement of the size of collapses on the rock surface. The calibration procedure aims to provide a working configuration to the software to be used in later analysis on different rock walls. The necessity of calibrating the software relies on the available data gathered during the different survey campaigns carried out on different periods of time. For last, the methodology to evaluate the performances of the neural network will be briefly discussed.

4.1 SOFTWARE TESTING

4.1.1 SIMULATOR VS METASHAPE DEPTH MAPS COMPARISON

Rockfall Simulator allows the user to directly exploit third party software (Metashape) reconstruction algorithms, as reported in section 3.4. The intermediate product of the photogrammetric pipeline involved in this process is the depth map. In order to exploit Metashape's capabilities, the simulated depth maps produced with Rockfall Simulator should be as much as possible coherent to the ones produced by the SGM algorithms implemented in Metashape.

For this purpose, a comparison of the depth maps produced by Agisoft Metashape and Rockfall Simulator has been carried out.
The original orientation parameters (EO and IO) from the Glendale block were maintained to further ensure the uniformity of conditions between the two software environments, therefore the comparisons are carried without any measurement noise on EO and IO parameters. The depth maps were processed without the presence of any external noise factors, thereby eliminating potential variables that might compromise the validity of the comparisons. Additionally, both depth maps also shared identical resolutions.
The comparisons have been carried in a GIS environment after exporting the depth maps produced by Metashape and by Rockfall Simulator. At this stage, minor differences are expected due to the different approaches used to compute depth maps in Agisoft Metashape and Rockfall Simulator.
To quantify these differences, both the mean and the standard deviation along with the maximum and minimal values of the difference map were calculated.

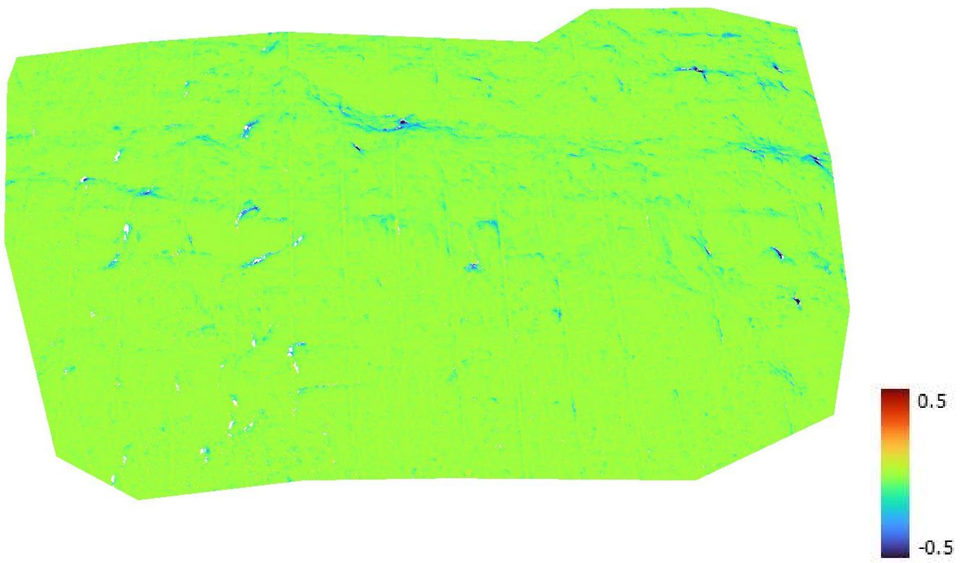


Figure 4-1: Difference map of depth values (m)

As can be observed, the predominant color on the difference map is green, indicative of values nearing zero. Differences of larger magnitude, that in a few pixels reach 70 cm (and are highlighted on the image by the blue pixels) usually occur at rock wall shape discontinuities and seem suggesting that the Metashape SGM matching routine tends to smooth too much the disparity/depth map in those areas.

Table 4-1: Statistical indices of the depth difference map.

Min (m)	**Max (m)**	**Mean (m)**	**StD (m)**	**RMS (m)**
-0.69	0.292	-0.0164	0.0264	0.0318

Figure 4-2 shows the histogram of the difference map.

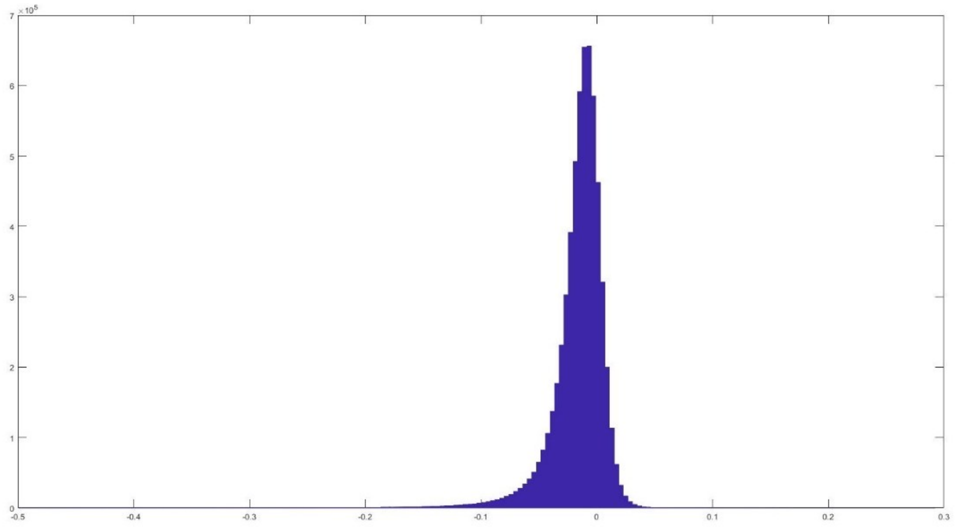


Figure 4-2: Histogram of the difference values obtained from the depth maps.

Overall, the agreement between Rockfall Simulator and Metashape's algorithms is good, with a mean difference around 1.5 cm and a StD of 2.5 cm. At this point, it must be noticed that Metashape deploys a proprietary algorithm for depth maps fusion, utilized during 3D reconstruction, which may (or may not) produce slight differences when comparing the depth maps.

4.1.2 SIMULATOR VS METASHAPE MESH RECONSTRUCTION COMPARISON

As far as 3D reconstruction of the photogrammetric mesh, a comparison between the Metashape's mesh, and the mesh created by the simulation process in Rockfall Simulator has been considered. Both meshes have been produced with the same set of EO and IO parameters, without any contribution of photogrammetric noise, therefore both meshes should be identical. Both meshes have been imported in CloudCompare and the Rockfall Simulator mesh has been finely registered to the reference mesh (the one produced with Metashape) in order to minimize any possible displacement. To compute the difference with the tool mesh-to-mesh difference, the Metashape mesh has been kept as the reference for this purpose. The following image illustrates the difference map obtained in CloudCompare.

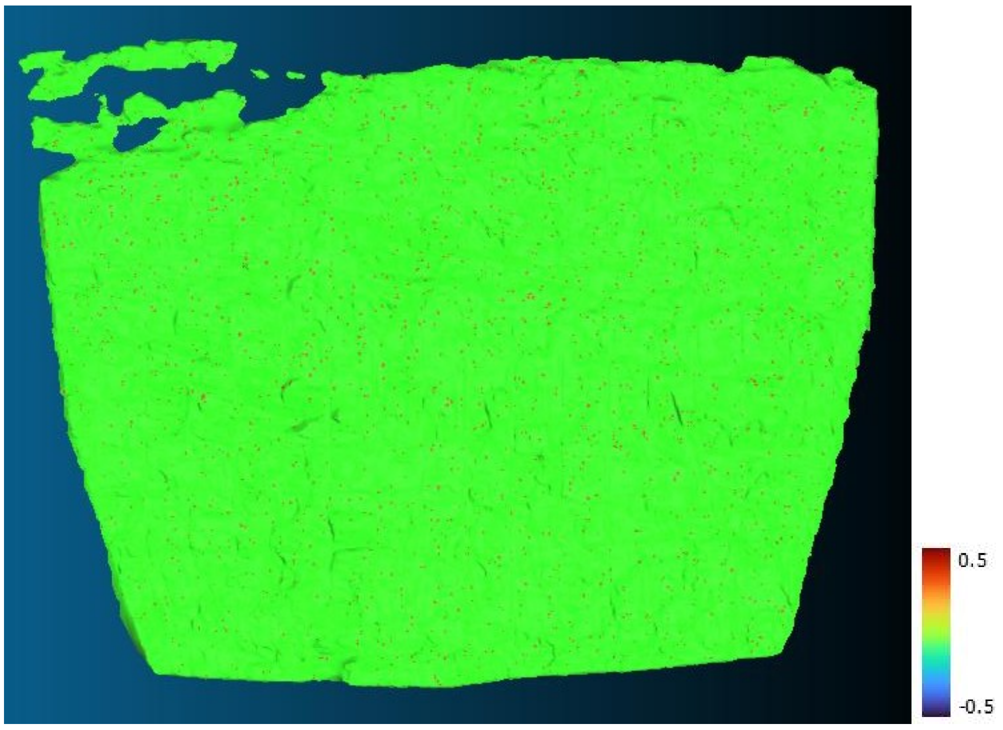


Figure 4-3: Representation of the mesh-to-mesh difference between the Metashape and the Rockfall Simulator meshes.

The prevailing color in the difference mesh is green, which in the color scale used indicate mesh-to-mesh distances close to zero. Table 4-2 illustrates the mean and the standard deviation of the differences.

Table 4-2: Mesh-to-mesh difference basic statistics

Mean (mm)	StD (mm)
0.12	1.091

The statistics of Table 4-2 show that the deviation between the reference mesh from Metashape and the reconstructed mesh from Rockfall Simulator is on average negligible. However, when visually inspecting the meshes difference in CloudCompare, it is possible to observe red spots, that highlight presumably higher differences. Figure 4-4 shows a zoom over different parts of the mesh (from left to right) where these spots are highlighted in CloudCompare.

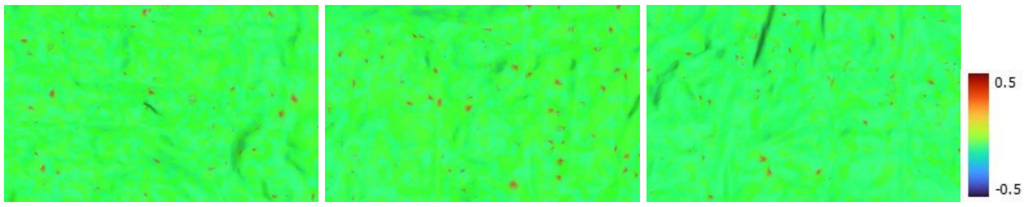


Figure 4-4: a) Western portion of the difference mesh, b) central portion of the difference mesh, c) eastern portion of the difference mesh

To delve into this issue more comprehensively, it was deemed necessary to conduct an examination of the vertices coordinates of the meshes.

Table 4-3: Mean and standard deviation of the differences between the reference mesh and the reconstructed mesh

Mean (mm)	0.00	0.00	0.00
StD (mm)	0.004	0.0017	0.0062

The situation depicted in Figure 4-4 could plausibly be associated with the rounding of the vertices' coordinates enacted by Metashape.

4.2 Software noise calibration

4.1.3 Fixed monitoring systems used in the experiments

The stereo monitoring system designed by Roncella et al. [138] has been deployed in different sites. The system consists of two units designed to guarantee autonomy and continuity of operation in the environmental working conditions and adequate stability of the installation over time, to preserve the calibration conditions as far as possible and thus ensures good repeatability of the acquisitions [139]. Each unit comprises a photogrammetric camera, a remote-control system, and a power supply system, housed in a IP66 protective case, which protects the internal parts from rain, moisture, and dust.

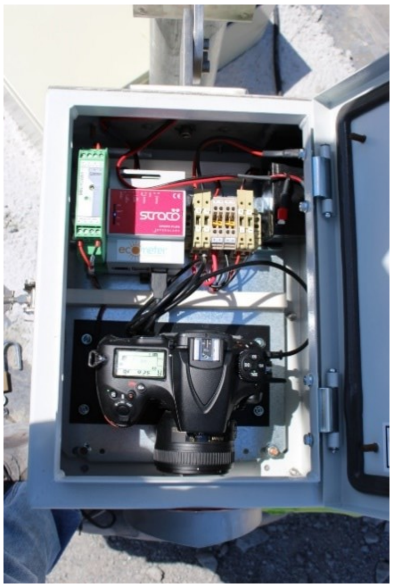
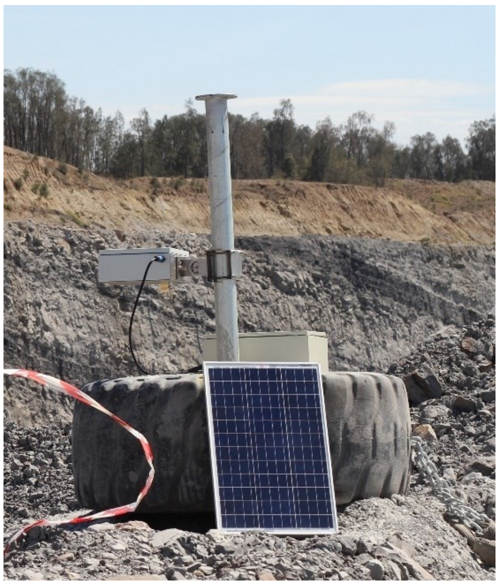



Figure 4-5:a) IP66 box of the fixed monitoring system b) Monitoring system installation in front of the rock wall.

The camera is a Nikon D810, with a sensor resolution of 36.3 MP (7360 x 4912 pixels) equipped with an f/1.8 lens with a focal length of 50 mm. The control system consists of a Raspberry Pi 3 microcontroller, which allows the camera's parameters to be set remotely, plan and control the status of the acquisitions and system conditions (temperature, power supply, etc.), and finally transmits the images via the Internet. The main power supply unit consists of two 26 Ah batteries connected to a photovoltaic system capable of producing

60W of energy. A further power supply module is located inside the box and is connected to both the camera and the logic unit, to protect the system from sudden power failures.
The system is equipped with a retro-reflective topographic prism mounted above each camera unit. In this setup, the monitoring system is fastened to a steel pole, firmly anchored to the ground, thus any translational displacements of the cameras due to shocks, wind, or unforeseen events should be sufficiently safeguarded during the system's operational cycles. However, potential minor rotational movements might be expected due to severe weather conditions or unexpected vibrations from heavy-vehicle movements.

Table 4-4 Nikon D810 specifications

Nikon D810	
Sensor	36.3 MP
Sensor size	35.9 x 24 mm
Focal length (mm)	50
Resolution	7360 x 4912
ISO	64-12800

The processing of images transferred via the Internet is carried out on a remote server using a software package developed at the University of Parma, called Slope Monitor [22]. This software allows for the automatic processing of the system's stereo pairs, exploiting Metashape's scripting capabilities to perform dense matching and 3D model reconstruction. The estimation of IO and EO parameters can be carried out by means of a BBA procedure [140] using ground control points (GCPs).
In the case of an image block with limited amount of photographic data (in this case, only an image pair) and with minimal depth variations, it is very likely to witness strong correlations among the IO parameters. For these reasons, the two-steps calibration methodology presented in [22] has been employed: firstly, the internal orientation parameters of the two cameras were estimated and, secondly, the external orientation parameters of the camera centers were determined by means of topographic surveys.

<u>*Site 1 – Glendale (Australia)*</u>

The first one is a mining quarry located near Glendale in Australia's Hunter Valley (NSW, Australia). The system acquires hourly information on the condition of the framed portion of the sub-vertical wall (Figure 4-6), which has an extension of 70 x 45 m^2 and is oriented North/North-West (dip direction of 255°).

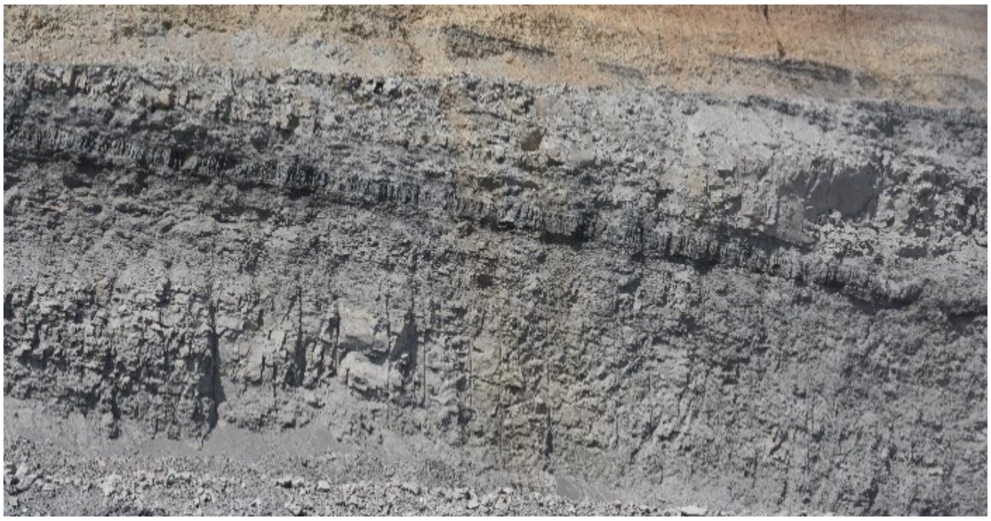


Figure 4-6: Glendale quarry rock wall

The wall is geologically divided into three different horizontal strata. The upper layer (Figure 4-7.a) consists of sedimentary rock (sandstone), with discontinuities mainly oriented on the vertical and sub-vertical face. Such condition makes this layer particularly prone to the detachment of large portions of rock. The medial layer of the wall (Figure 4-7.b), approximately 2.3 m thick, is characterized by the presence of coal. This layer is particularly fragmented, with discontinuities characterized by roughly perpendicular layering with respect to the excavation face. The lower layer (Figure 4-7.c)) is similar in morphology to the upper one, with strong vertical discontinuities parallel to the excavation face.

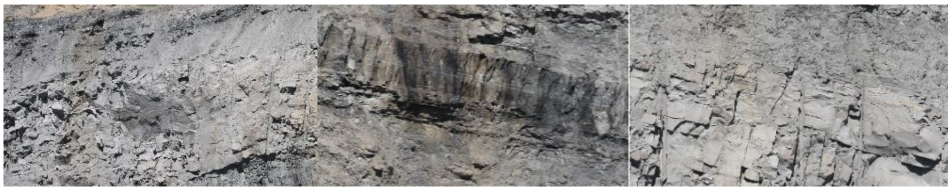


Figure 4-7 a) upper portion of the rock wall b) medial portion c) lower portion of the rock wall

The face of the wall has an average slope of approximately 70° with respect to the horizontal plane. The combination of the slope and the geological characteristics of the wall (with frequent intersections of discontinuities in the direction of the face) makes it particularly vulnerable to major collapse events. Detachments from the outer sandstone layers (particularly the upper one) can be a major risk factor for mine personnel and excavation equipment. Within this environment, it was decided to deploy a fixed dual-camera photogrammetric system to allow for precise monitoring of the rock wall.

To achieve the highest possible overlap between frames, the two cameras were aligned with a slight convergence of their optical axes [23]. The wall being imaged is situated at a distance of ca. 105 meters, while the distance between the centers of the two cameras (base-length) is approximately 28 meters. The a-priori precision of the stereo-pair system can be computed as:

$$\sigma_Z = \frac{Z^2}{c\,B}\,\sigma_{p_\xi} = 2.7\ cm$$

where σ_Z is the theoretical precision, Z is the distance to the object, c is the principal distance, B is the base-length (i.e., the distance between the two camera centers) and finally $\sigma p\xi$ is the theoretical precision of the parallax measurement between two homologous points. Assuming an image coordinate precision of 0.5 pixel the theoretical precision of stereo-restitution in depth direction is therefore 2.7 cm.

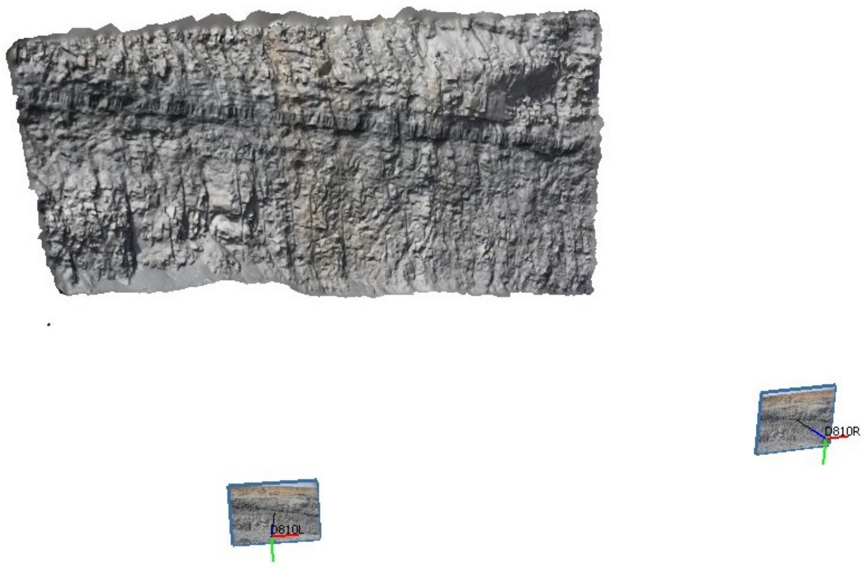


Figure 4-8: Glendale image block geometry.

Site 2 – Bulga (Australia)

A second site is the Bulga rock wall (Figure 4-10), which exhibits a pronounced horizontal stratification. The wall is approximately 70 x 45 m^2. The rock's texture appears predominantly fine-grained, although there are some patches that seem coarser, possibly indicating inclusions or varying sediment types within the layers. Additionally, there are a series of vertical fracture lines or joints, which may have resulted from tectonic activity, weathering, or other geological processes. The following images illustrate the stereo pair of the image block, with a GSD coherent to the one from Glendale (e.g., 2 cm per pixel).

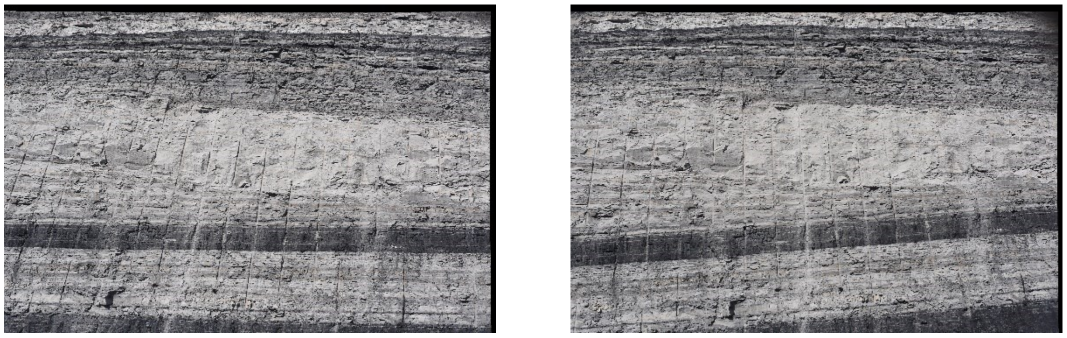


Figure 4-9: Bulga rock wall stereo pair acquired with the stereo fixed photogrammetric system illustrated in section 4.1.3.

As the image block geometry is structured as follows:

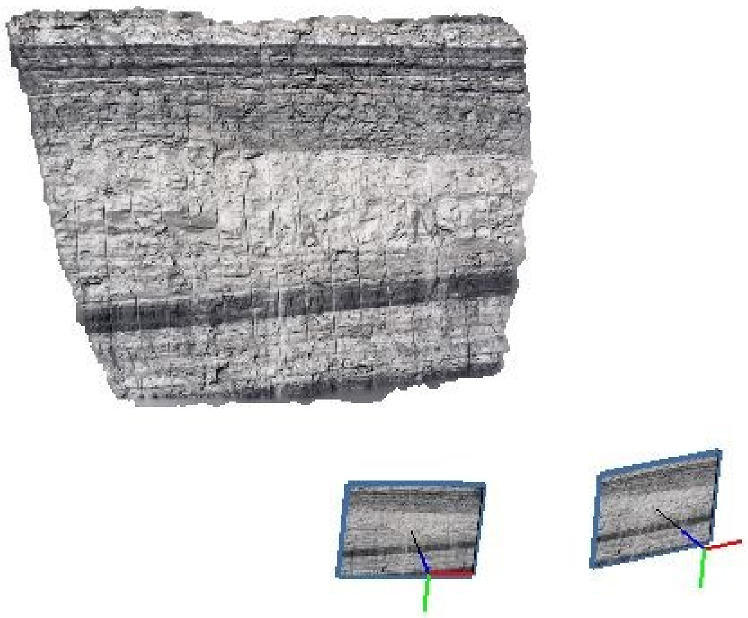


Figure 4-10: Bulga image block geometry.

<u>*Site 3 – Pietra di Bismantova (Italy)*</u>

A third one (Pietra di Bismantova, Figure 4-12) has been acquired on 05/09/2020 with low light conditions and refers to an approximately 70 x 70 m^2 wall. The Pietra di Bismantova is a striking rock formation located in the Emilia-Romagna region of Italy. The rock has undergone a variety of geological processes, including tectonic uplift, and erosion. In terms of its stratigraphy, the oldest layers are predominantly composed of calcareous rock [141]. The rock wall exhibits unique geomorphological features. It is defined by a series of discontinuity lines that predominantly orient vertically.

In addition to these lines of discontinuity, the rock wall is further complicated by the presence of numerous erosional indentations or notches. These concave formations are the result of extended erosive processes that have acted upon the slope, likely attributable to a combination of atmospheric elements, water flow, and freeze-thaw dynamics. These indentations provide a detailed portrait of erosion rates and may signify structural weaknesses that warrant further investigation for potential landslide or collapse risks. The following image pair has been exploited to generate the photogrammetric simulation block.

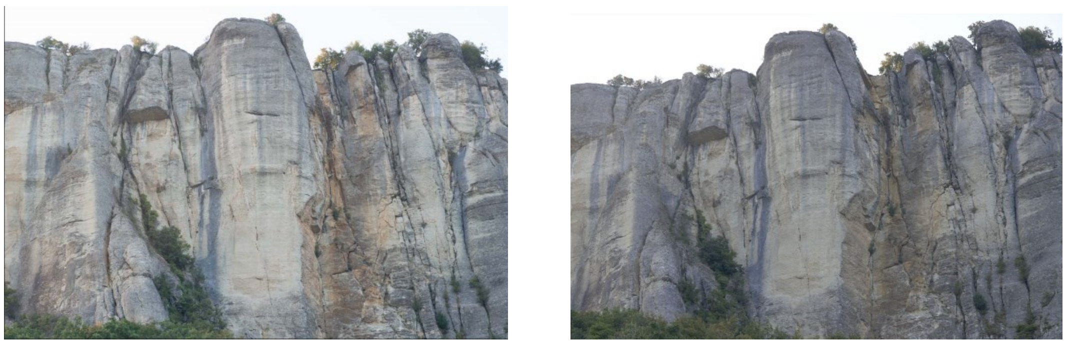


Figure 4-11: Pietra di Bismantova rock wall image pair acquired with the stereo fixed photogrammetric system illustrated in section 4.1.3.

As the geometry of the image block is reported in Figure 4-12.

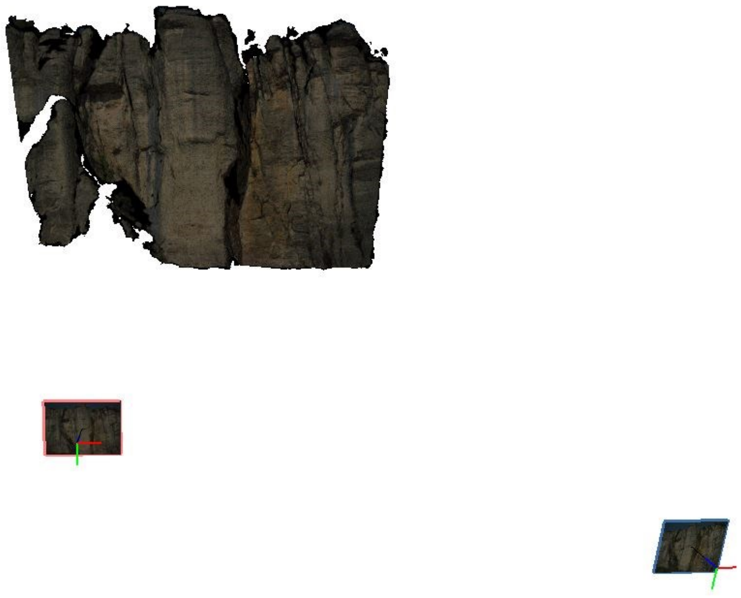


Figure 4-12: Pietra di Bismantova image block geometry,

Site 4 – Pilkington (Australia)

The last image block has been acquired in Pilkington (Australia) (Figure 4-14). The rock face appears to have distinct layers of horizontal and sub-vertical strata, with approximately dimensions of 15 x 8 m^2. These planes often represent zones of mechanical weakness and are characteristics of sedimentary rocks. There are zones on the rock wall where the rock strata exhibit offsets. Such offsets may indicate zones interest by rockfall events.

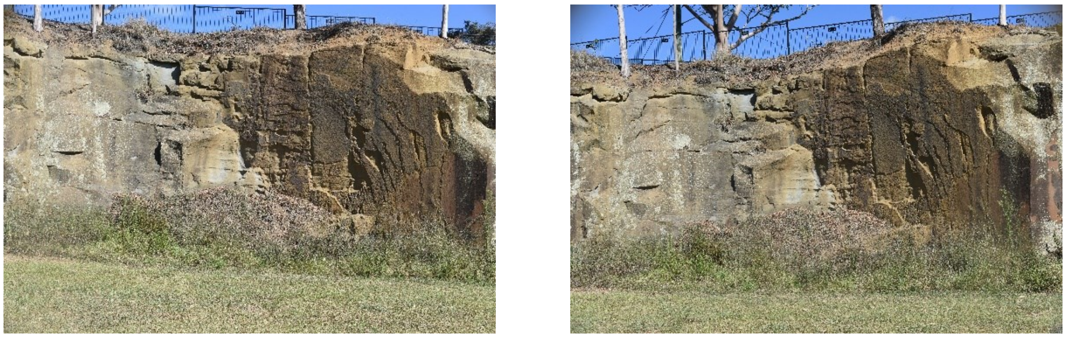


Figure 4-13: Pilkington rock wall image pair acquired with the stereo fixed photogrammetric system illustrated in section 4.1.3.

As the geometry of the image block is reported in Figure 5-18.

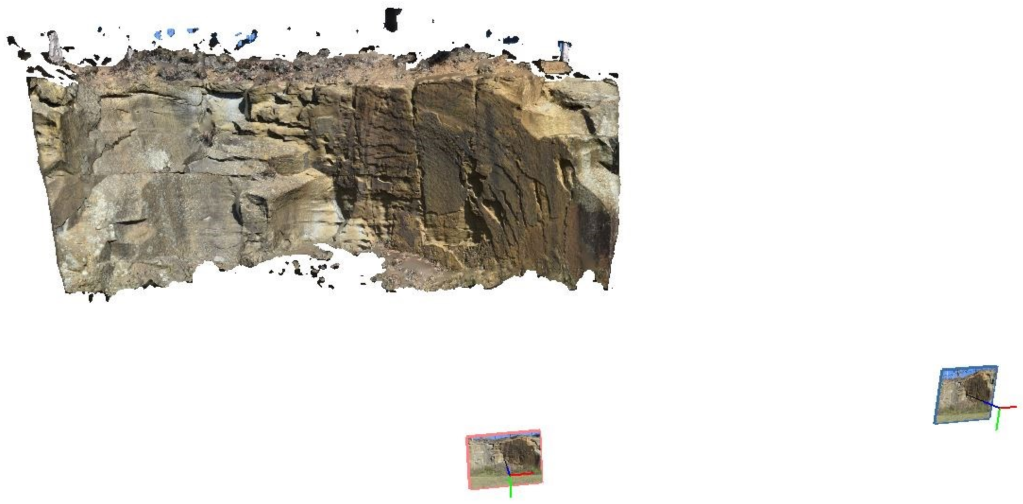


Figure 4-14: Pilkington image block geometry.

4.1.4 QUANTIFICATION OF PHOTOGRAMMETRIC NOISE FROM REAL DATA

To produce reference data for the evaluation of the photogrammetric noise and rockfall dimensions, the stereo pairs of eight acquisitions in the Glendale quarry over approximately two months (see Table 3) have been processed. The time window in which the acquisitions were carried out covers February and March 2018, as the system was temporarily installed for testing purposes. It was chosen to use the images taken around midday (12:00 A.M.) as, in a study over the same site [142], the best lighting conditions for the monitoring system were recognized to be between 12 A.M. and 2 P.M. when shadows are minimal or at least of little impact. Shadows may cause a strong alteration of the grey tones in the shaded area and, consequently, can affect the image matching procedures [143] and ultimately the 3D reconstruction accuracy.

Table 4-5: List of surveys carried at Glendale quarry facility during the testing campaign.

Survey	Date
Reference	09/02/2018
Acquisition 1	11/02/2018
Acquisition 2	22/02/2018
Acquisition 3	06/03/2018
Acquisition 4	11/03/2018
Acquisition 5	16/03/2018
Acquisition 6	24/03/2018
Acquisition 7	27/03/2018

All photogrammetric blocks were processed with Metashape Pro 1.8.5 [128] to obtain a 3D mesh model of the rock wall. The camera orientation parameters used in the processing are those obtained through the calibration procedures of the photogrammetric system described in the previous section. In particular, the internal orientation parameters used were obtained from a camera calibration carried out just after the reference campaign. Due to a blasting event that occurred in the middle of February 2018, the calibration of the stereo monitoring system had to be repeated in early March of 2018, on the 06/03/2018. It was observed that unexpected variations in the camera assets occurred, particularly a 2-degree rotation around the horizontal axis of the left camera.

The workflow employed to produce data for realistic noise estimation was as follow.
Key points are selected from images. Based on similarity (matching) of descriptors, tie points are found, and, after the BBA, a sparse point cloud made by all tie points found is generated. To obtain a large number of tie points and to achieve the best accuracy, the images have been employed at their native resolution (Quality = "High" in Metashape terminology). Next, the dense matching procedure is carried out. To speed up the process, the images were subsampled by a factor 4 (Metashape's 'Medium' value). Finally, the 3D mesh of the wall is produced and imported in CloudCompare^{TM} (in .obj format) to optimize the co-registration on the reference epoch using the Iterative Closest Point (ICP) algorithm [130], [144].
Starting from the co-registered meshes, a DEM was produced for each processed epoch to be compared with the reference one. To best represent the detachments process, the DEM XY plane should coincide with the mean plane of the wall: to this aim, a plane has been fitted to the reference epoch point cloud. The projection plane origin has been placed in the lower left corner of the wall, with the y-axis oriented towards the upper left vertex of the wall and the x-axis oriented towards the lower right vertex of the wall. The DEMs of the processed epochs have been generated with a ground resolution of 0.02 m/px (ground sampling distance), which coincides with the GSD of the image blocks acquired by the monitoring system.

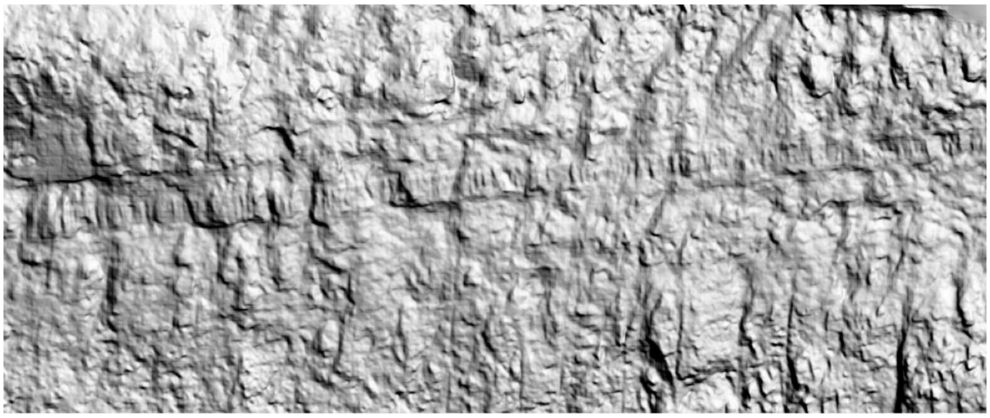


Figure 4-15 Hillshaded DEM of the wall at the reference epoch

The digital models are compared by means of a pixel-by-pixel subtraction of the reconstructed elevation values between the reference epoch and the following ones.

$$z_{difference\ map}[i,j] = z_{reference}[i,j] - z_{k-th\ epoch}[i,j]$$

where [i, j] is the position of the pixel on the raster and k denotes the k-the reconstructed epoch. Figure 4-16 shows a difference map obtained between the reference epoch (Reference in Table 4-5) and the next one.

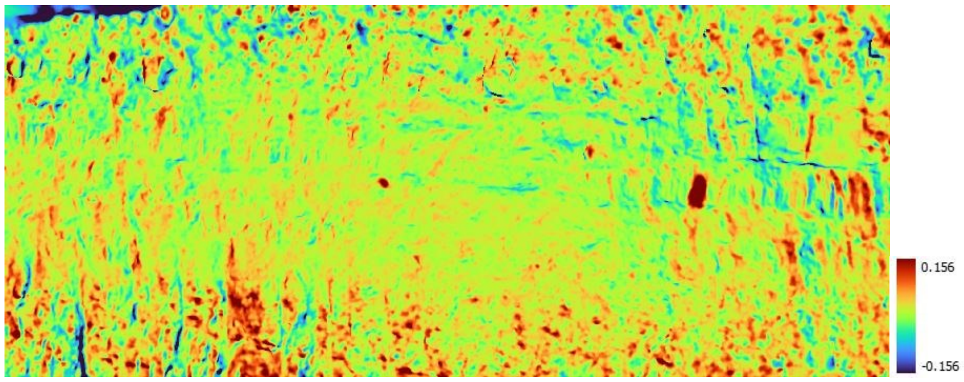


Figure 4-16: Difference map of the rock wall (Reference to Epoch01)

The color scale shown on the right side of Figure 4-16 encompasses the range [-6 σ_Z ; 6 σ_Z] where $\sigma_Z = 0.027$ m is the theoretical precision of the monitoring system in depth direction. In the figure above, pixels whose value falls outside the a priori established acceptance range are highlighted in red (holes) and blue (outcrops). Errors, and particularly outliers, can be expected in any automated photogrammetric process, which can cause gross or systematic errors in the coordinates of the object points. This condition occurs randomly due to errors in matching or due to too small intersection angles of the homologous rays (which are therefore almost parallel) and can lead to a forward (or backward) shift in the position of the point in object space [145]. Systematic errors might also be caused by the variation of IO parameters over time during the survey campaign. As the measurement noise estimation is carried out by a statistical analysis of the differences between the DTM of pairs of epochs, care must be taken to exclude areas affected by detachments between the various epochs, which could otherwise bias the estimates of noise.

At this stage, it is important to account for a both qualitative and quantitative evaluation of measurement noise, as it serves as a benchmark to the following simulation software calibration procedure.

Variations in the IO parameters, particularly the focal length and radial distortion parameters in the Brown-Conrady camera model may cause a global deformation of the model and therefore in the difference map. To provide a clearer understanding of the global deformations observed by processing image blocks from actual data, out of the diverse outcomes Figure 4-17 is provided.

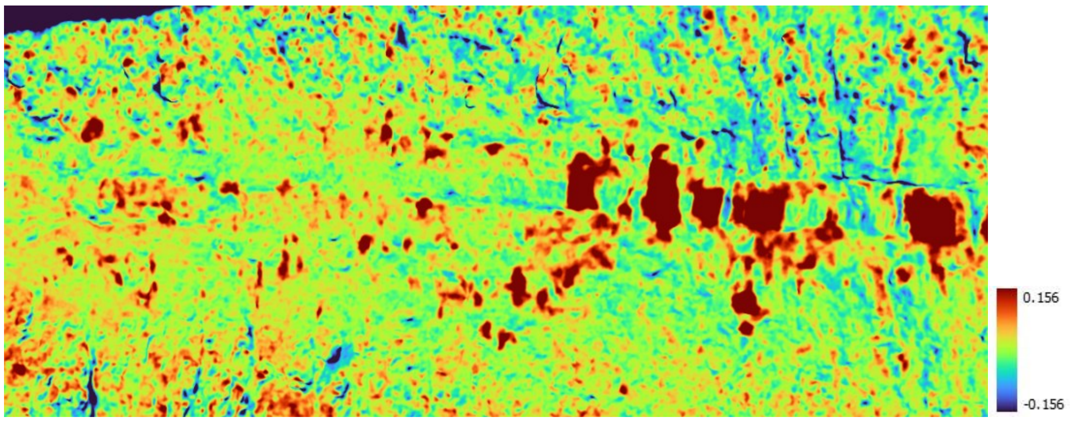


Figure 4-17: Difference map produced for the real study case of Glendale.

As depicted in Figure 4-17, excluding the deep red values, that potentially represent rock collapses, the central part of the difference map exhibits a consistent pattern that closely aligns with the reference model, as indicated by the green-colored areas representing differences near 0 m. In these areas, the model is accurately co-registered, demonstrating only minor discrepancies on the surface.

Conversely, the left and right sides of the model are represented by blue-colored areas, indicating negative difference values (i.e., the compared model is positioned "above" the reference model). This discrepancy highlights a noticeable curvature on the sides of the compared model, likely attributable to variations in the IO parameters, which can introduce such effects. Analogously, a similar deformation with the opposite sign is observable, indicating that the sides of the model are positioned "below" the reference model, resulting in positive difference values in the same regions.

Two sections of the previously attached difference map are shown in Figure 4-18 to illustrate the concept more clearly:

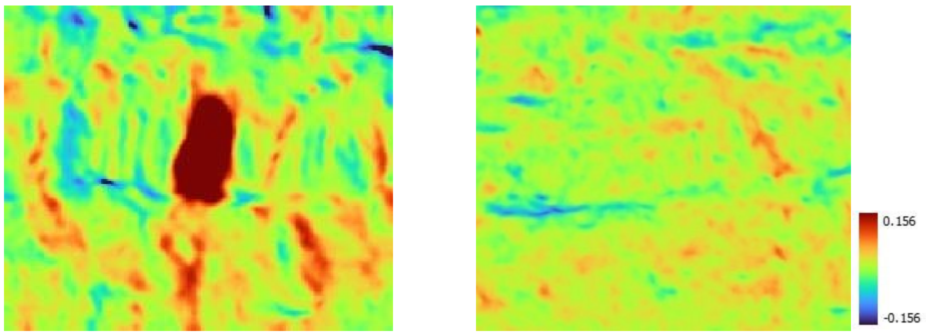


Figure 4-18: a) control window 1 b) control window 2

In case a), a non-random spatial distribution of differences is observed in the central zone, quite different from the one theoretically expected from the distribution of model reconstruction errors obtained from the photogrammetric process. This pattern indicates morphological variations with respect to the reference and, consequently, plausibly a collapse event. Section b), on the other hand, shows a morphology that is entirely comparable between the two models (prevailing green/orange color), except some areas of discontinuity (light blue, all almost horizontal) that could be due to a slight change of the geometric and optical arrangements of the cameras between the different epochs that affect mostly horizontal discontinuities. The following images provide a more detailed qualitative analysis of these effects on the difference maps.

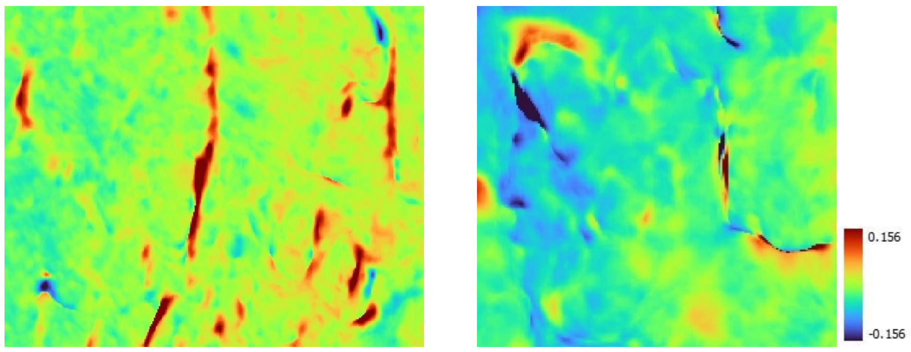


Figure 4-19: Examples of difference maps along discontinuities

In the top left image of Figure 4-19 (left), the described effect appears as missing material along the discontinuities of the rock wall. In fact, the red areas highlight positive difference values, as the compared tridimensional model does not properly align with the reference model. Positive differences would on the other hand be wrongly recognized by the identification process as rock collapse. Actually, discontinuities are weak rock structures, so detachment of material may well be expected there.

On the other hand, Figure 4-19 (right) shows that the difference maps may provide negative values along the discontinuity, meaning that there could be some sort of material accumulation in such areas. Such cases may originate for the same reasons, i.e., small variations of the orientation parameters in the following surveying epochs. As the sign of the difference values is opposite to what is observed in a real rock collapse, the identification system would make no commission errors. It may happen, however, that such areas hid or underestimate the amount of a true collapse nearby. In this case it is considered close to impossible to overcome such obstacle, as the automatic identification tools provide a compromise in such situations.

Additionally, it is important to evaluate the presence of measurement errors on flat and untextured rock surfaces (see Figure 4-24), that are far from uncommon. In such areas, which also include shadows and reflections, the reliability of matching algorithms is often compromised due to missing information and the attempt to fill the gaps resorting to various interpolation techniques.

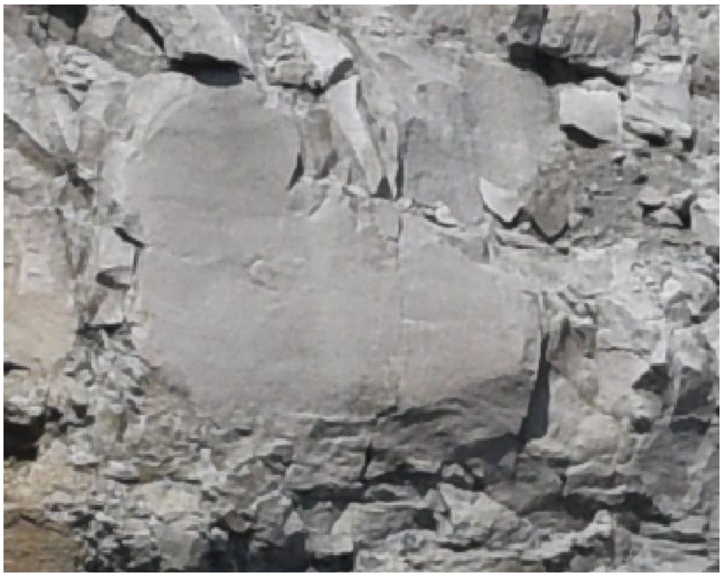


Figure 4-20: Flat untextured surface on the Glendale rock wall

As depicted in Figure 4-20, flat and untextured surfaces present a challenge for matching algorithms due to the lack of well-defined features. The rocky surface is largely flat and lacks distinct characteristics, thereby contributing to measurement errors during the reconstruction phase.

To illustrate the desired outcomes of the noise simulation, the ensuing figure showcases the difference map derived from manual processing of real survey data obtained in Glendale. The disparity map delineates the differences observed within this particular area, highlighting the challenges encountered when processing such data.

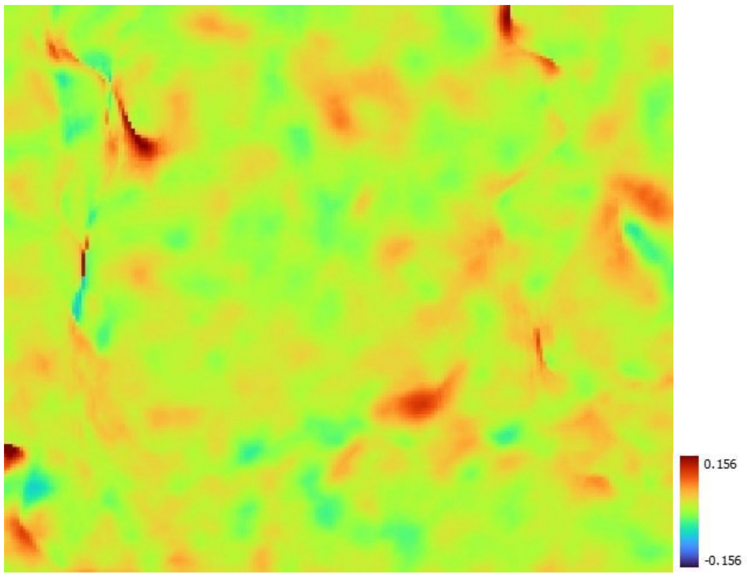


Figure 4-21: Difference map of the area in Figure 3-36

As depicted in Figure 4-21, the difference map reveals a notable discrepancy between the smooth surface observed in the original photo and its absence in the mapped representation. This discrepancy manifests as a distinct pattern, where areas with green hues (differences close to 0 m) are interspersed with orange and red areas probably due to errors during image matching.

When addressing quantitatively the real noise distribution, it must be considered the presence of collapses on the difference map, therefore it was necessary to restrict the noise analyses to areas possibly not affected by such phenomena. Three areas of approximately 5 x 5 m^2 (approximately 250 x 250 pixel each) were identified as stable across the whole-time span, by visually comparing the difference maps of all the epochs used for noise calibration. The three zones are shown in the blue boxes in the difference map (Figure 4-10) while Figure 4-23 shows their images.

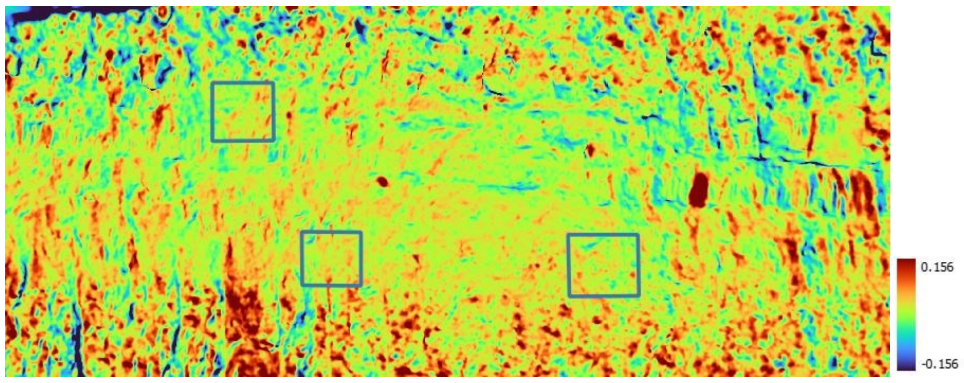


Figure 4-22: Inspection areas on difference map

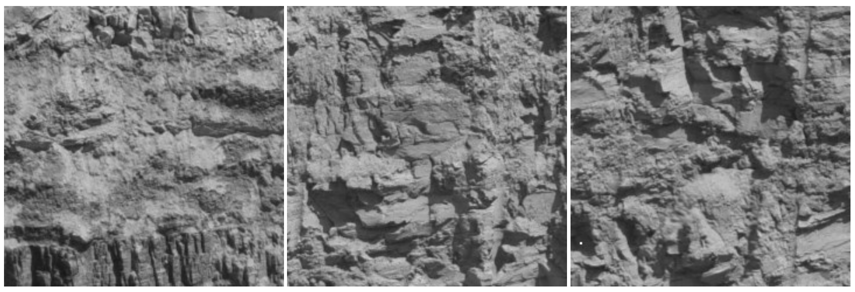


Figure 4-23 Area 1 (left), Area 2 (center), Area 3 (right) patches on the rock wall.

The left portion of the rock wall, referred to as Area 1, is located on the western side and is situated in the upper region. The morphology of this area is relatively smooth and lacks texture, with discontinuities running parallel to the excavation face. The lower region of the imaging window encompasses the medial coal stratus, which is more uniformly textured and exhibits greater depth variations. Areas 2 and 3, located respectively in the center and eastern portions of the rock wall's lower region, are highly textured and often exhibit consistent shadows across multiple epochs [154].

These areas were considered stable during the acquisition period since no major movement could be clearly identified on them. This notwithstanding, it must be noted that the morphology (by means of possible occlusions) and the lighting conditions (by means of extended shadows) may provide an obstacle to 3D reconstruction. In fact, it can be challenging to achieve accurate and detailed results [41] as there may be several sources of disturbance for matching algorithms. Another issue that needs to be considered is that the stability of the rock wall cannot be assumed to be perfect from the outset. Consequently, there may be some minor movements that are not clearly distinguishable as proper collapses, even in these supposedly stable areas. This may partially deviate the estimation of the real distribution of the noise.

For each difference map, the three patches have been exported singularly and then aggregated in a 3-layered matrix. To summarize the magnitude and distribution of noise in these patches, basic statistical indices such as median, standard deviation and RMS of the differences have been computed together with the difference map histogram. In order to conveniently exclude data belonging to the tails of the distribution (and therefore attributable to potential rockfall events), it was decided to calculate the median value to exclude larger differences from the statistical analysis as they may refer to potential movements on the rock wall. The expected median value from such analysis should be relatively close to 0 when comparing the different models, suggesting that there are no systematic errors observed over time.

In Table 4-6 the statistics are reported to enable the analysis of noise over time in case some trend is present.

Table 4-6: Statistical indices for the different confrontation epochs

	Date	Median (m)	StD (m)	RMSE (m)
Acquisition 01	09/02/2018	-0.000139	0.020154	0.0208
Acquisition 02	11/02/2018	0.00110	0.01951	0.0200
Acquisition 03	22/02/2018	0.000906	0.019631	0.0205
Acquisition 04	06/03/2018	0.001908	0.021768	0.0231
Acquisition 05	11/03/2018	0.004397	0.021027	0.0214
Acquisition 06	16/03/2018	0.003496	0.023944	0.0239
Acquisition 07	24/03/2018	0.005387	0.026707	0.0267

The following plots represent the data of Table 4-6. In these charts, the horizontal yellow line represents the mean value across all epochs, while the blue bars illustrate the median value (across the 3 different patches) calculated for each epoch.

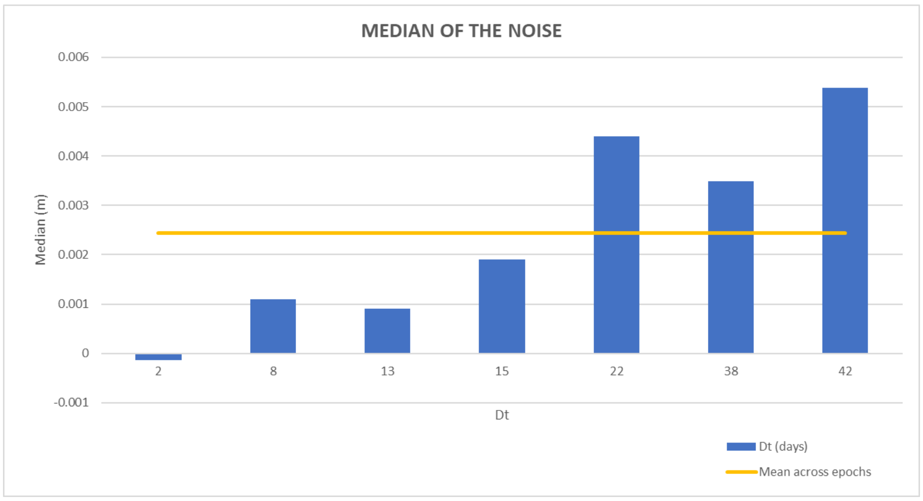


Figure 4-24: the chart represents the median value of the differences w.r.t. the reference epoch calculated across the three inspection areas at each epoch (blue bar) and the mean value calculated over all the epochs (yellow line).

The chart above highlights the median error across different epochs to detect any minor detachments and possible model deformations of the rock wall over time. The median error represents the vertical displacement (Z-axis) between the reference epoch and subsequent epochs. Notice that the Iterative Closest Point (ICP) algorithm is applied across the entire model, so a fraction of the median error may depend on the optimization not being restricted to the three boxes only. The median error is low in the initial three epochs, and the reconstructed models show no significant differences. However, as the time goes, small variations in the internal and external camera orientation parameters may affect the 3D model, which cannot completely be accounted for by the ICP algorithm in the registration. Consequently, this leads to an increase in median error values in subsequent epochs.

Figure 4-25 shows the standard deviation of the differences at each epoch, a measure of the noise values. The horizontal yellow value is the mean value of the standard deviation across all epochs.

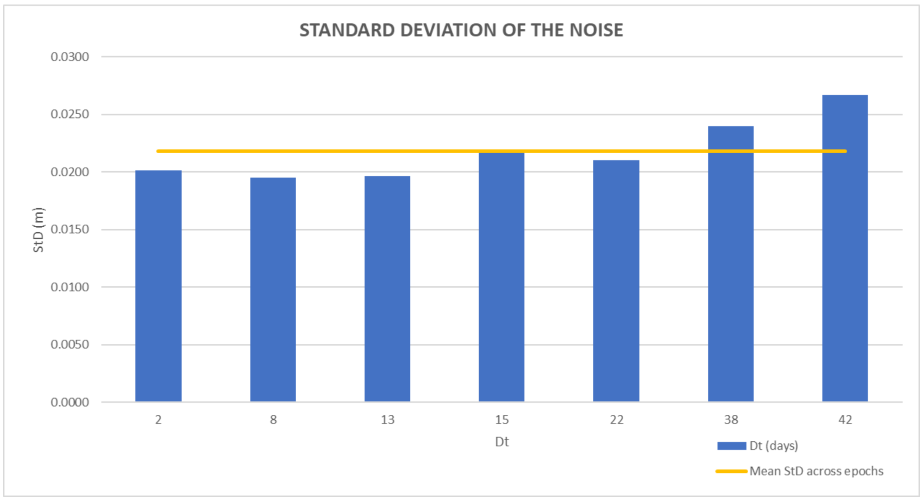


Figure 4-25: The chart represents the standard deviation of the noise for each epoch (blue chart) and the mean value across the different epochs (yellow line)

The standard deviation of the differences exhibits a consistent pattern over time, with an average value of approximately 2.4 cm. The first five epochs are all just above or just below 2 cm. An increase of the standard deviation is observed in Epoch 06 and 07; the latter, at 2.6 cm, is about 30% higher than in Epoch 01. Overall, however, the standard deviations remain relatively stable over the epochs, averaging around 2 cm. Moreover, the average standard deviation is consistent with the theoretical block precision calculated in section 4.1.3.

Table 4-7: Noise simulation thresholds for Glendale mining site rock wall

Median (mm)	***Mean StD (mm)***	***Mean RMS (mm)***
2.6	21.8	2.23

To faithfully simulate photogrammetric noise, the basic statistics alone are not enough, as it is also important to consider the distribution of errors, that should in principle be normal since theoretically the system is not subject of systematic errors. The following chart represent the relative frequency histograms of Epoch 07, as it is recognized as the most unfavorable case of the set. Epoch 07 shows the higher residuals in terms of mean error (meaning that there could be a possible systematism in the error distribution -for example a translation in the points coordinates or a curvature/variation in the model geometry due to the variation of the distortion parameters over time). It must be noticed that Epoch 07's acquisitions are gathered after 42 days from the deployment of the fixed system.

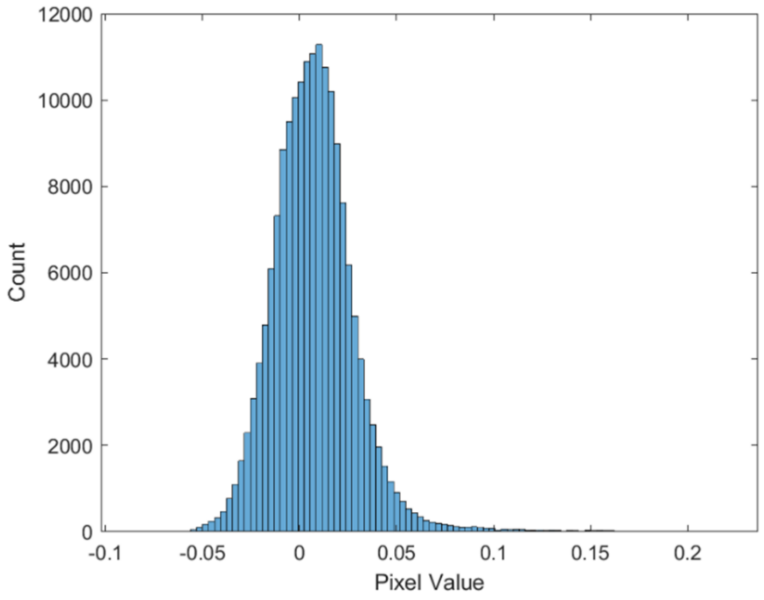


Figure 4-26: Histogram of Epoch 07 differences.

The relative frequency graphs demonstrate a pattern that roughly describes a bell-shaped distribution, typically associated with a Gaussian probability distribution.

4.1.5 SIMULATED NOISE CALIBRATION

To ensure accurate replication of the conditions observed during the analysis of a real case, it is crucial to calibrate the noise parameters to obtain similar results in the simulated difference maps generated by Rockfall Simulator. Calibrating the noise parameters in Rockfall Simulator involves specifying the parameters of the probability distribution for the random noise generation of the following parameters:

1. **Exterior orientation**: The variations in the orientation parameters of each camera, aim to reproduce the effect on the stability of the camera support structure of external disturbances (wind, vibrations, temperature changes, etc.).
2. **Interior orientation**: The Brown-Conrady's camera model parameters, can vary over time in response to temperature, lens deformations, and other factors affecting the optics.
3. **Depth maps**: This pertains to errors encountered during depth map generation, including inaccuracies in parallax estimation and errors introduced during the SGM matching procedures.

Although the goal of calibration is clear, i.e., to achieve error statistics and patterns similar to those found in 4.1.3, it is likely that many different combinations may be acceptable. Each parameter error has a specific effect that can be studied separately. However, determining the combination of these effects producing error maps similar to the empirical ones, when so many noise sources are involved, is not simple. An efficient approach for addressing the calibration procedure related to the simulation software might be the establishment of a cost function that accommodates for the empirical characteristics of errors. This cost function would serve as a mathematical representation of the divergence between observed empirical data and the corresponding simulated outputs. It should provide a quantifiable metric by which the optimization algorithms can iteratively adjust the orientation parameters to minimize the observed discrepancies. By effectively capturing the empirical characteristics of errors, the cost function would offer a more precise understanding of how each orientation parameter influences the overall system performance, thereby allowing for a more targeted calibration strategy.

A first assessment has been carried out on the stability of the exterior orientation parameters, in terms of translations on the position of each perspective center in the object space and the relative rotation of the optical axis of each camera.

In such systems, a minute deviation in the relative or individual positions and orientations of the camera stations can result in apparent discrepancies when producing Digital Terrain Models (DTMs). Even a slight unaccounted rotation of the camera can lead to significant changes due to the distance from the object (around 100 m from the rock wall). While the likelihood of a change in the base-length (i.e., movement along the line connecting the two acquisition centers) is low due to the robust and securely anchored support structure of the protection box, other factors such as vibrations from the transit of operating quarry vehicles, unpredicted windstorms or rock wall blasting operations can induce rotational movements in the cameras' orientation.

To assess the impact of rotations, it is preferable to apply the rotation to only one of the cameras. Figure 4-27 shows the effect of a Phi rotation of +0.01 degrees along the Y direction, with the Y-axis being vertically oriented in the image. In this sense, a rotation along Y will result in the camera to be inclined towards left (positive rotation) or right (negative rotation).

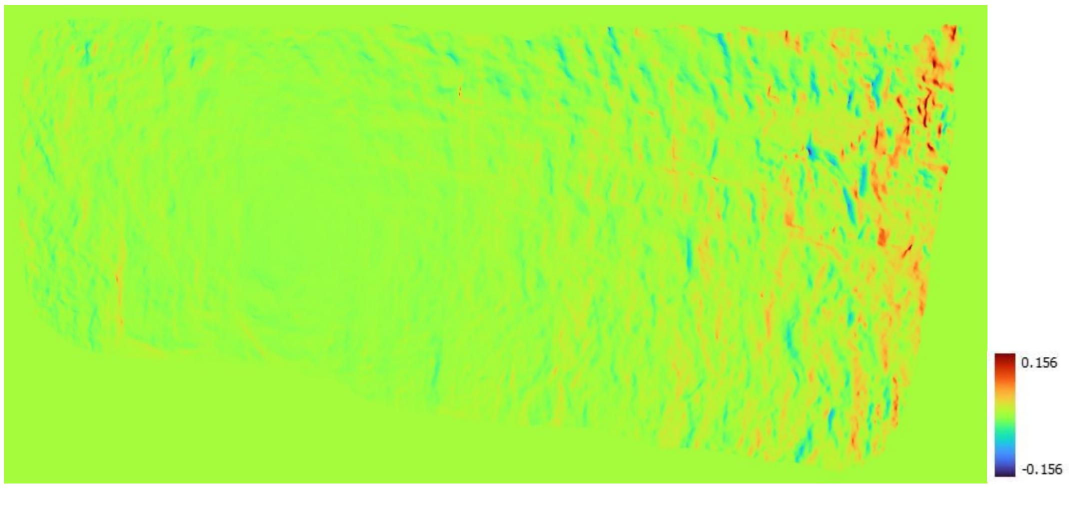


Figure 4-27: Difference map with a Phi rotation of +0.01 degrees

As it can be observed, the rotation slightly contributes to recreating the gaps along the vertical discontinuities, with a rather inhomogeneous pattern. In particular, the red areas are highlighted on the outer side of the difference map.
A similar rotation for the Omega angle, a rotation along the X-axis, is shown in Figure 4-28.

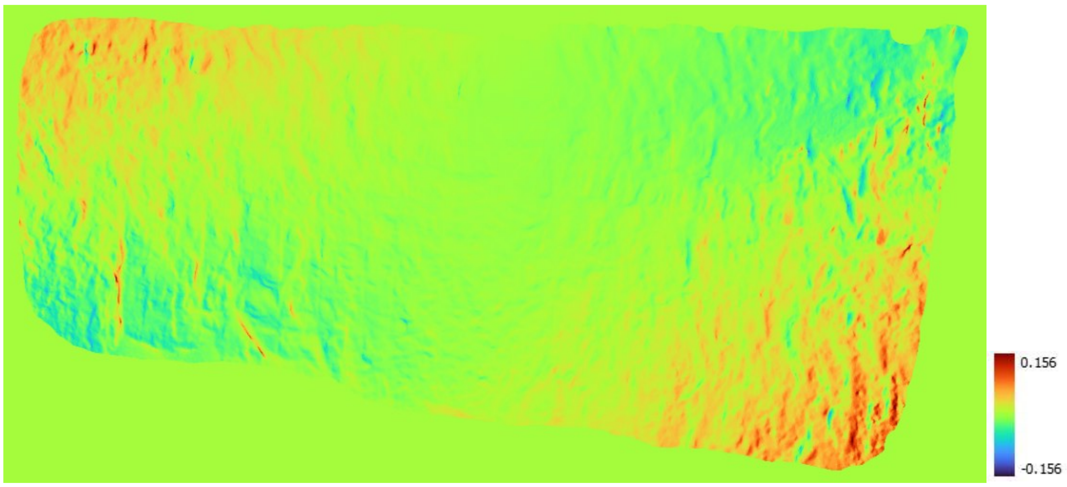


Figure 4-28: Difference map with an Omega rotation of +0.01 degrees

The deformation pattern exhibits a strong correlation with both sub-vertical and vertical discontinuities. Notably, the model demonstrates significant variations towards the sides of the model, which is a shared feature with the Omega rotation previously illustrated.
Testing the effects of changes in the EO parameters has been restricted to modest entity rotations only, as the system characteristics should ensure enough stability of the camera projection centers. Therefore, any variation in the base-length has been ruled out.
Assessing the amount and effects of changes in the IO parameters has been performed with reference to the acquisition campaigns of the monitoring system illustrated section 4.1.3.
First each parameter has been changed independently of the others, starting with the focal length, a major parameter in the photogrammetric process as it is tied to the scale of representation of the reconstructed object.
Additionally, there exists the possibility of stochastic correlations among these parameters, further increasing the complexity of the task.
As the image block consists of only a pair of images, self-calibration can result in some interior orientation parameters to be highly correlated (see Table 4-8).

Table 4-8: Correlation matrix (Cxx) for Glendale image block when performing self-calibration task.

	F	Cx	Cy	B1	B2	K1	K2	K3	K4	P1	P2
F	1	-0.11	-0.96	-0.74	0.85	0.37	-0.25	0.1	0.03	-0.16	0.85
Cx		1	0.09	-0.56	0.41	-0.53	0.07	0.28	-0.59	1	-0.24
Cy			1	0.72	-0.85	-0.34	0.13	0.02	-0.14	0.15	-0.73
B1				1	-0.96	0.03	0.15	-0.26	0.35	-0.52	-0.53
B2					1	0.11	-0.2	0.24	-0.27	0.36	0.65
K1						1	-0.78	0.49	-0.16	-0.56	0.57
K2							1	-0.92	0.71	0.1	-0.48
K3								1	-0.93	0.27	0.3
K4									1	-0.59	-0.1
P1										1	-0.28
P2											1

From Table 4-8 the parameter Cy is highly correlated with the focal length (F) and with P2. The same strong correlation can be observed between Cx and P1, and among the affinity parameters and the radial distortion parameters respectively. Such correlations are favored by the weakness of the block, and the low density of the ground natural points. Methods to improve the calibration accuracy in such scenarios have been presented in 4.1.1.

As a result, it becomes increasingly unfeasible to exhaustively explore all possible parameter combinations in the search for the optimal calibration setup.

To find a satisfactory solution that adequately depicts the real cases, both in a qualitative and in a quantitative manner, a heuristic approach has been preferred. Evaluation of the outcomes will be undertaken both quantitatively, in terms of reconstruction error metrics, and qualitatively, by examining whether the simulated data manifest patterns analogous to those observed in the empirical setting.

In order to provide a starting point for the heuristic approach, it was decided to assess three distinct calibration sets for the Nikon D810 cameras estimated during the maintenance operations of the fixed monitoring system situated in Glendale. This strategy aims to provide preliminary insights that could guide the search for an optimal combination of parameters that most faithfully represent real-world scenarios. The deviations observed between the three calibration sets could offer a coarse evaluation of the parameter's uncertainty. Table 9 resumes the average IO difference values observed among the three calibration blocks previously cited. These results are clearly not expected to provide an optimal solution to the IO noise simulation but may be useful as a starting point for the research of realistic Rockfall Simulator noise parameters.

Table 5: Mean and StD from of the IO parameters from the different calibration campaigns of the fixed system.

	Mean	**StD**
F (px)	0.008	7.53
Cx (px)	-0.001	87.44
Cy (px)	-0.006	34.91
b1	0.0023	4.29
b2	-0.46	10.91
k1	-0.0001	0.076
k2	0.0047	0.68
k3	-0.0537	1.46
k4	0.1847	0.59
p1	-0.000001	0.002
p2	-0.000023	0.0004

The standard deviations of each IO parameter presented in Table 4-8 have been imported in Rockfall Simulator to provide a first attempt solution to develop the calibration process on. A heuristic procedure has been devised that, like a greedy algorithm, minimizing in an iterative cycle the standard deviation of the original difference map, by changing a parameter value at a time, until the reference standard deviation (3 cm) is reached.
Measurement has been generated by a normal distribution with zero mean and a tentative standard deviation, whose value is established based on the tests on single parameter changes and the precision estimates of the parameters from the system calibration.
Figure 4-34 represents the results of the tests executed in search of the parameters error combination that better represent the difference maps in the real case study. The horizontal axis lists the iteration number while the vertical one reports the standard deviation (expressed in meters) of the Z error from the difference map. The blue line represents the simulated

tests, while the orange one the reference noise levels obtained from the survey image blocks in section 4.1.3.

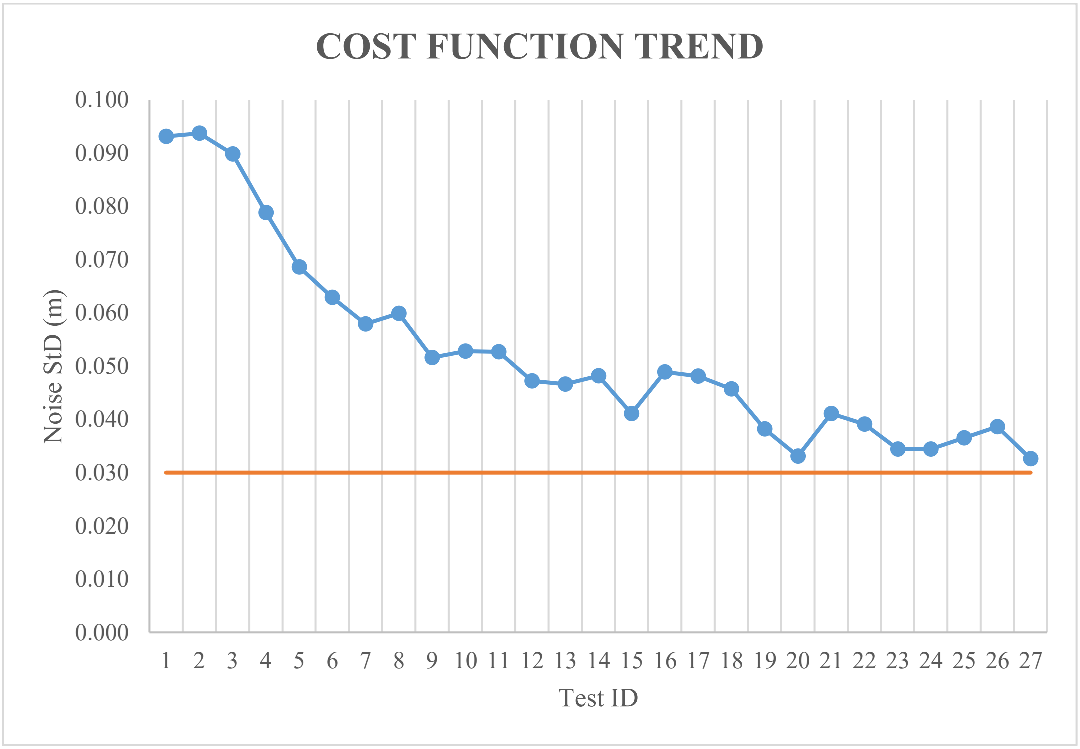


Figure 4-29: Noise Testing on Rockfall Simulator.

All the tests are identified with an ID number which refers to a specific parameters' dataset. The initial combination of parameters (Test #1) is the dataset obtained from the evaluation of the differences over time of the calibration campaigns parameters provided in Table 9. Such solution, which has been graphically represented in Figure 4-30, provides noise levels which are out of the margins of the real case, with standard deviation values of around 10 cm (around 4 times exceeding the real cases). Additionally, the solution does not provide a qualitative outcome (spatial distribution of the Z error) comparable to the real study case. The principal point coordinates (Cx and Cy) have been varied first, obtaining a decrease of the noise levels to around 6 cm from Test #2 to Test#7.

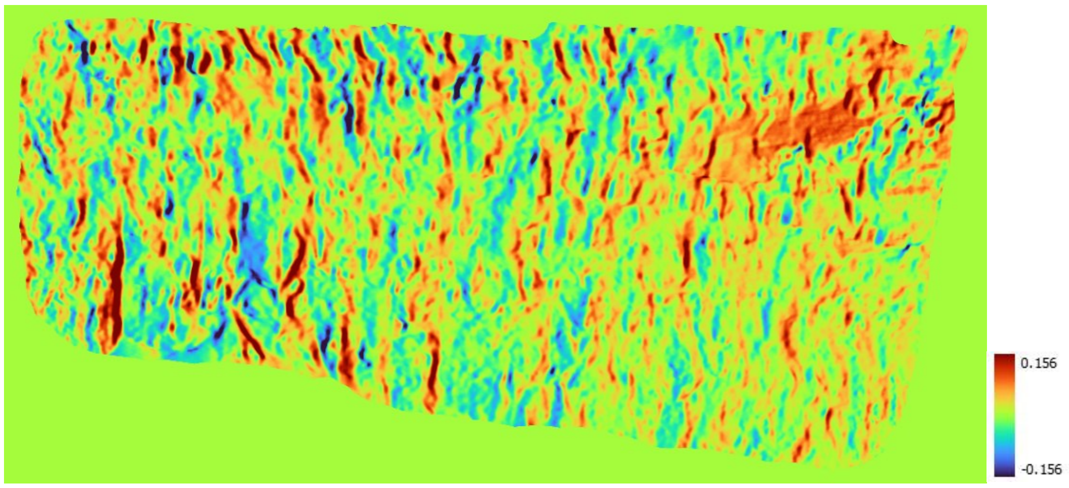


Figure 4-30: Noise effects obtained by the variation of component Cx.

More specifically, Cx has been varied from a value of around 80 pixels (estimated in the earlier calibration campaigns) to around 15 pixels in Test #7 by decrements of around 8 pixels per iteration (roughly 10% of the initial value). The steep descent of the standard deviation (reduced by around 40%) confirms the importance of using an accurate value of Cx, as shown in the testing of single parameters.

Introducing noise on the x component of the principal point results in a systematic vertical shift of the object points. Unaccounted changes in the position of the principal points could lead to misleading results if they are correlated to a translation of the acquisition center. For instance, if a translation vector Y_0 is defined under such conditions, and a negative variation in Cx (the principal point coordinate) is applied, they would counterbalance each other's effects. This observation is important because the positions of the camera stations remain relatively fixed, as they are not subjected to significant variations due to the monitoring system's structure. Generating such scenario would results in the following difference map:

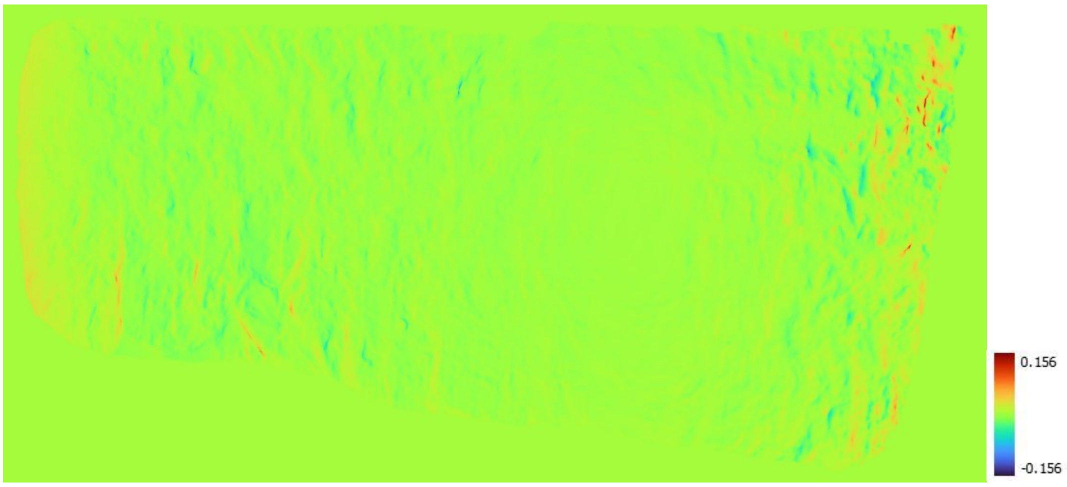


Figure 4-31: Difference map with negative Cx and positive Y_0 changes

The same Cx variation is directly applied while simulating a negative offset with an equivalent magnitude for the translation vector along the horizontal axis.
The same pattern has been repeated for Cy from Test #8 to Test #15. The initial solution was characterized by a deviation of around 30 pixels in Cy, a value that, as for Cx, was estimated in the calibration campaigns. At each consecutive steps a variation of Cy by around 3 pixels has been applied, which resulted in a decrease of around 30% of the standard deviation of the difference maps.

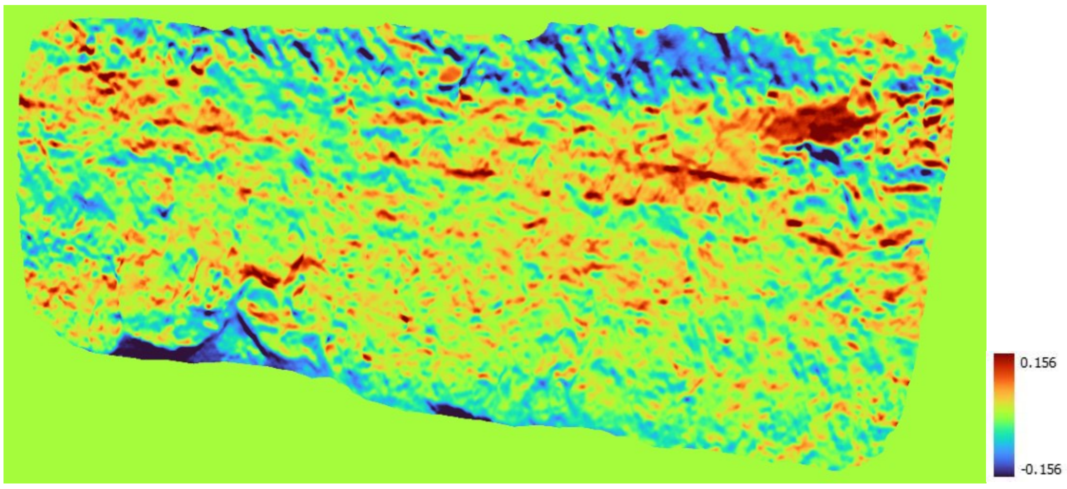


Figure 4-32: Noise effects obtained by the variation of component Cy.

From Test #16 to Test #24, an analysis was conducted to examine the influence of other internal orientation parameters, specifically radial distortion, tangential distortion, and the skew and affinity parameters. Test #17, which incorporated radial distortion parameters from calibration campaign assessments, initially demonstrated an increase in noise levels. However, Test #18, which reduced the intensity of radial distortion by 20%, indicated a minor decrease in noise levels.
The introduction of tangential distortion parameters in this test phase resulted in a rise in noise levels by approximately 15%. This suggests a significant contribution from tangential distortion parameters to the simulated noise levels. However, it is important to note that the quantitative noise representation at this stage did not align with the actual distribution observed in the case study.
In Test #21, the tangential contribution was incorporated into the noise simulation, using thresholds derived from the calibration. The swift rise in standard deviation highlighted the significant contribution of tangential distortion parameters to the noise levels, that resulted in a noise level increase of approximately 15%, signifying a secondary order increase relative to the radial distortion parameters.
By Test #23, incorporating all parameters (principal point, radial and tangential distortion parameters, and rotations) yielded a standard deviation of around 3 cm in noise levels, consistent with the real case study results. The difference map generated from this parameter combination is depicted in Figure 4-35.

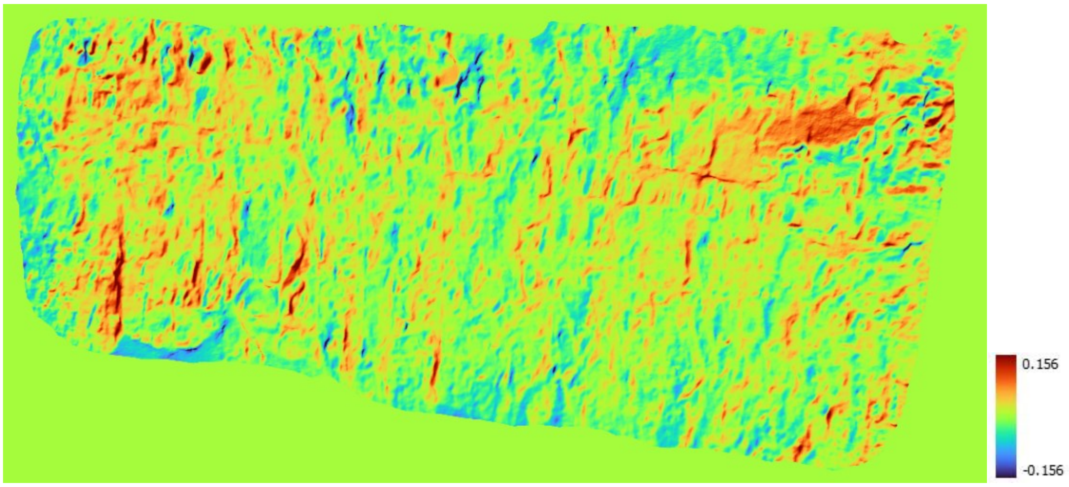


Figure 4-33: Difference map produced with Test #25 parameter dataset.

The difference map resulting from the Test #23 parameter dataset showcases a pattern similar to those derived from the real-world case study. The most characteristic features, such as patterns along vertical and sub-vertical discontinuities and minor deformation on the raster's outer sides, are present. However, it's also crucial to simulate patterns observed on smooth textured areas, a characteristic outcome of imprecise matches via the Semi-Global Matching (SGM) algorithms.

To simulate this component, Rockfall Simulator is equipped with a parallax noise and a regularization noise parameter, as discussed in section 3.4.

Tests #24 to #27 were designed to provide moderate contributions to these parameters to further understand their impact on noise levels. As most image matching methods consistently achieve sub-pixel accuracy [146], the parallax and regularization noise levels were kept below 1 pixel.

Initially, Test #24 introduced a 0.1-pixel parallax error. This marginally increased the noise levels by approximately 0.2 cm. To enhance this contribution, Test #25 assessed a 0.5-pixel error, which resulted in a further 0.2 cm increase in noise levels. This revealed a non-linear pattern for this noise parameter, achieving levels around 3.5 cm. In Tests #26 and #27, regularization noise of 0.5 and 0.25 pixels respectively were added to the process. The former amount resulted in noise levels around 4 cm, exceeding the desired range. However, halving the regularization noise contribution in the latter test drastically reduced the noise levels to approximately 3.4 cm, aligning well with the observed real noise values. The qualitative result derived from Test #27 is demonstrated in the subsequent figure.

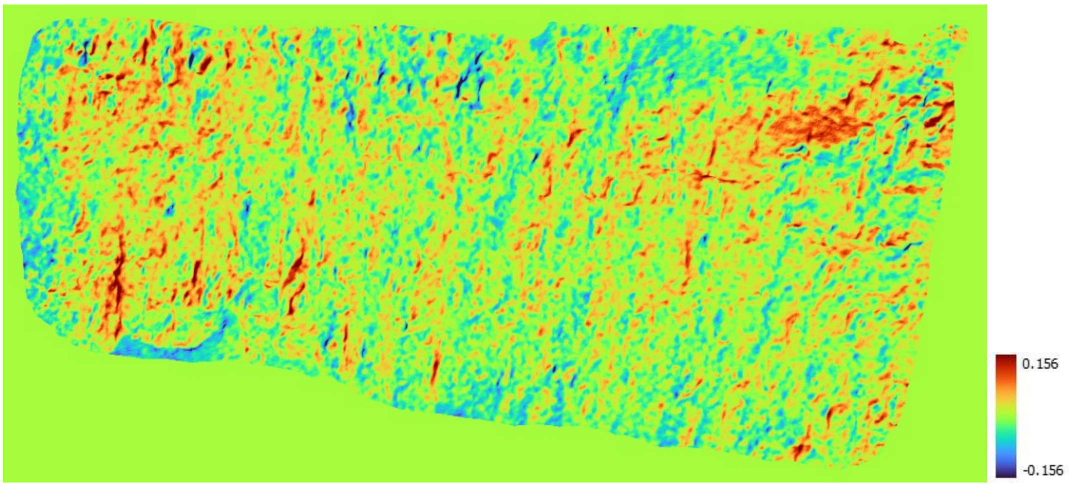


Figure 4-34: Final test difference map.

In Test #27 all error sources considered are present and produced a difference map similar to the one coming from real data. In this setting, the resulting thresholds for the simulation process may be summarized in the subsequent table.

Table 4-9: Noise simulation thresholds.

Noise thresholds			
Omega	0.04	**K1**	0.014
Phi	0.01	**K2**	0.136
Kappa	0	**K3**	0.28
F	7.5	**K4**	0.12
Cx	13.6	**P1**	0.00144
Cy	5.44	**P2**	0.00024
B1	2.52	**Reg**	0.25
B2	6.54	**Pix**	0.6

The specified thresholds are used to simulate the most severe case, identified processing different epochs collected by the fixed monitoring system during the 2018 survey campaign at the Glendale quarry environment. These thresholds provide a solution that best aligns with the qualitative and quantitative study case.

The worst-case scenario (Acquisition 07) was selected, as it provides the highest StD value. In this framework, it was assumed that this worst-case standard deviation covers almost all the noise (up to 99%). Then, it was decided to divide the standard deviation to three to get a 'normal' level of noise, which it is considered to cover about 68% of the cases.

This allows the software to comprehensively generate any random combination within this interval. The updated Rockfall Simulator parameters are reported in the following Table, defining the mean and standard deviation values used for the simulation processes.

Table 4-10: Rockfall Simulator simulation thresholds

Rockfall Simulator thresholds

	Mean	StD		Mean	StD
Omega	0	0.01	**K1**	0	0.005
Phi	0	0.003	**K2**	0	0.04
Kappa	0	0	**K3**	0	0.09
F	0	2.5	**K4**	0	0.04
Cx	0	4.5	**P1**	0	0.0005
Cy	0	1.8	**P2**	0	0.00008
B1	0	0.84	**Reg**	0.25	
B2	0	2.18	**Pix**	0.6	

Conventionally, the mean value of the gaussian probability distribution is consistently set to zero, thereby eliminating any systematic influence on the production of comparative meshes. However, the extent of variation or standard deviation values plays a significant role in regulating the noise simulation distributions. It must be noticed that regularization and parallax (pixel) noise are expressed only in standard deviation values, since the probability distribution is set to a zero-mean value by default.

4.2 NEURAL NETWORK EVALUATION

4.2.1 FRAMEWORK

The growing interest in deep learning has prompted the introduction of development environments with dedicated libraries for the implementation and application of neural models to various scientific problems. The use of these tools has also expanded among users who are not necessarily experts, so that flexible and easy-to-set-up development environments can be used effectively.

The main characteristics to evaluate when choosing the ideal development environment for one's needs are:

1. **Extensibility**: The ability to implement different types of neural architectures (convolutional, FCN, recurrent neural networks (RNN) (Giles et al., 1998)), and to carry out training according to different methodologies, e.g., supervised, or unsupervised.
2. **Hardware utilization**: the ability to support computing procedures in a distributed manner on CPU and GPU.
3. **Adaptability**: The versatility of the development tools provided by the environment to address the problems of interest.

In the current research, TensorFlow (see the next sections) was selected as the primary development environment due to its extensibility, adaptability and the possibility of using high-level libraries for NN modelling, training and testing as Keras [147].

TensorFlow [148], developed by Google in 2015, is the most popular development environment in the community of researchers working with neural networks. It supports major programming languages such as Python and C++. The peculiarity of TensorFlow concerns the methodologies with which neural network calculations are approached. Each operation carried out by the model is broken down into several tensors, i.e., matrices developed on several levels of depth.

The integrated libraries allow the required operations to be performed by relying on the GPU or the CPU. The parallelization of calculations enabled using the GPU results in considerably shorter training times.

The most widely used high-level library in TensorFlow by the research community is Keras [147]. Keras makes it possible to greatly simplify the construction of customized neural

models due to the presence of several independent modules that can be appropriately combined.
In this thesis work, the latter framework for the application of the neural models was involved in the research activity. The motivations behind this choice mainly concern the flexibility and simplicity of the application of the models combined with the possibility of customizing the model, which is superior to other solutions found in the literature.

4.2.2 EVALUATION METRICS

To evaluate the performances of the neural network, the predictions of the model are stored in a JSON file with the same formatting of the ground truth JSON file. To better understand the metrics used to evaluate the performances of the model, it is necessary to briefly introduce the following concepts involved in the computation of the model performance indices. When comparing the ground truth instances to the predicted instances it is possible to obtain different outcomes:
True Positives (TP): True Positives represent instances where the model correctly predicts the positive class in alignment with the ground truth data. In this work it means that the model correctly labelled a rockfall event on the provided difference map.
False Positives (FP): False Positives occur when the model predicts the positive class incorrectly or without matching the ground truth. In this work it means that the model erroneously detects non-existent objects or misclassified features.
False Negatives (FN): False Negatives happen when the model fails to predict the positive class. This could occur if the model overlooks actual objects or features of interest.
Such factors have direct contribution on the calculation of the performance metrics of the neural network. In particular, the following metrics have been considered:

1. The **mean average precision (mAP)**, which describes how well the model can predict true positives out of all positive predictions. The mAP is computed as:

$$mAP = \frac{TP}{TP + FP}$$

A high mAP score means that the model is capable of correctly identifying and localizing objects with high reliability and across the classes. A low mAP, on the other hand, means that the network has lower reliability in the prediction (i.e., increased presence of false positives).

2. The **mean average recall (mAR)**, which describes how well the model can predict true positives out of all predictions. mAR is calculated as:

$$mAR = \frac{TP}{TP + FN}$$

A high mAR value signifies that the model is adept at minimizing false negatives across the classes, thereby ensuring that most of the relevant instances are correctly identified. Conversely, a low mAR value can be indicative of a significant number of false negatives which implies that the model is failing to identify a substantial proportion of relevant features.

3. The **F1-Score** serves as a harmonic mean of precision and recall, aiming to strike a balance between these two metrics.

$$F1 = 2 * \frac{(mAP * mAR)}{(mAP + mAR)}$$

In practice, a high F1-Score suggests that the model not only identifies most of the positive instances (high recall) but also minimizes the inclusion of negative instances as positives (high precision). This harmonic balance ensures that the model's predictions are both exhaustive and exclusive, thereby maximizing its utility in applied contexts. On the other hand, a F1-Score close to 0 indicates that the model performs poorly in terms of either precision, recall, or both. This would mean that the model is producing a substantial number of false positives and/or false negatives, thereby rendering it unreliable for practical applications.

4. The **IntersArea** ratio of overlapping pixels of the predicted and ground truth segmentation masks. This index is a value ranging from 0 (no overlapping between the ground truth and the predicted masks) and 1 (complete overlapping).

$$InterSA = \frac{A_{prediction}}{A_{ground\ truth}}$$

A low value of the area intersection parameter could result in substantial inaccuracies in the estimation of the volumes involved in such collapse events. Although the current dissertation does not aim to assess the volumes implicated, maximizing this parameter has considerable relevance in practical applications.

To decide if a predicted object is actually a True or False Positive or a False Negative, an additional parameter should be computed. Intersection over Union (IoU), also known as the Jaccard index, has been selected as the metric of choice. This metric is mainly employed in computer vision tasks, such as object detection and semantic segmentation, to assess the degree of overlap between the predicted and the ground-truth instance. Mathematically, IoU is defined as the area of the intersection between the predicted and ground-truth entities, divided by the area of their union. The formula for IoU is thus:

$$IoU = \frac{Area\ of\ intersection}{Area\ of\ union}$$

IoU is a normalized parameter that lies in the interval [0, 1], thereby facilitating quantitative assessments of model performance. A value close to 1 indicates a high degree of overlap, suggesting a highly accurate detection or segmentation. A value close to 0 suggests minimal or no overlap, indicating a poor detection or segmentation.
In the operational framework, the Intersection over Union (IoU) metric has been deployed to label the output data from the neural network. Practically speaking, the decision was made to analyze the vectors corresponding to the ground-truth annotations and compare them with the vectors of output annotations generated by the network. Selecting a specific threshold for the IoU score allows to label a predicted instance as a True or False Positive and, similarly, a ground truth instance to be associated with a True Positive or a False Negative. In the scientific literature, metrics corresponding to IoU levels of 0.25 and 0.75 are frequently reported. In the former scenario, a more relaxed overlap criterion is applied,

allowing instances that are not perfectly aligned with the ground truth to still be considered as true positives. Conversely, as the IoU value increases, the evaluation becomes more stringent and sensitive to deviations from the ground truth.

In this framework, it is crucial to underscore the importance of uniqueness in the association between predicted and ground-truth bounding boxes. In this study, it was adopted a bidirectional approach for IoU analysis with the aim of mitigating any asymmetry or bias in the evaluation process. Specifically, each predicted object is matched, at most, with a single ground-truth object based on the highest IoU value, and vice versa. This methodology ensures a rigorous and unambiguous interpretation of performance indicators such as True Positives (TP), False Positives (FP), and False Negatives (FN), thereby contributing to a more robust and reliable evaluation of the model.

5 CHAPTER 5 - RESULTS

5.1 INTRODUCTION

This chapter illustrates the performance metrics of the Mask R-CNN model in rockfall event detection on synthetic data generated by the simulation software presented in Chapter 3. The testing methodology aims to address two main research questions (RQ):

1. **RQ1**: What impact does the size of rockfall events impact on the neural network's detection capabilities?
2. **RQ2**: What role does measurement noise play in the identification of rockfall events?

Both research questions share a fundamental aspect: the detectable size of rockfall events is directly correlated to the entity of measurement noise as it becomes more and more difficult to identify rockfall events when they are substantially hidden by measurement noise. Hence, the goal is to assess the neural network's performance in detection across varying levels of measurement noise and different dimensional classes of rockfall events.

The influence of collapse size on recognition capabilities (RQ1) has been a topic of interest in literature, as different methods may perform better for different size ranges [29]. This study seeks to determine how the size of collapses affects the neural network's ability to recognize them, in order to highlight the possible limitations of such tools. This investigation builds upon the work of previous researchers who have examined the impact of object size on the performance of object recognition algorithms in various domains, such as remote sensing and medical imaging [149] [150].

The influence of photogrammetric noise (RQ2) on the performance of the neural network is an essential consideration in the context of rockfall events recognition. Measurement noise comes from various sources, such as variations in internal and external orientation parameters, matching errors (due to shadows, differences in illumination and smooth texturing [146] [145].

5.2 NEURAL NETWORK TRAINING

Addressing the learning process with neural networks can be tricky, with several challenges to overcome. Two key challenges are overfitting and underfitting, which are the two sides of the same coin.

Overfitting occurs when a neural network learns too much from its training data. This means it becomes too tuned to that specific data and struggles to perform well on new, similar data. Essentially, it can't adapt well to new problems because it's too locked into what it already knows. In contrast, underfitting arises when the neural network doesn't have enough data to learn properly. This means it can't form a good understanding of the task at hand. Interestingly, both overfitting and underfitting end up with the same problem: the network isn't great at handling new tasks or problems due to the lack of generalization capabilities.

Given the time and computational resources required to train deep neural networks on large datasets, transfer learning [151] emerged as a popular and efficient alternative. Transfer learning leverages the knowledge gained while training one task and applies it to a different, yet related, task (i.e., the transfusion of information from one domain to a related one) . Essentially, it involves taking a pre-trained model (a neural network trained on a large dataset, typically for a benchmark task like image classification on ImageNet [152]) and fine-tuning it on a smaller, task-specific dataset. This method boasts several advantages, including reduced training times and lower data requirements, making it especially appealing for tasks where data is scarce or computational resources are limited.

However, the characteristics of transfer learning has also led to questions about its universality and applicability. While it can be an invaluable tool when the source and target tasks are closely related, problems arise when there's a significant divergence between the source and target domains [153], [154] , which may often result in negative transfer [155].

[172] and [173] focus on the application of transfer learning in the context of medical imaging diagnostics. The authors found out that transfer learning in this specific domain often does not provide significant benefits compared to training from scratch, especially when the correlation between the source domain (generally natural images) and the target domain (medical images) is low. Based on such empirical outcomes, it was decided to perform training from scratch for the rockfall identification tasks.

Another issue is the limited dimensionality of the classes involved in the training process. The case of study, indeed, is only related to two classes, i.e., a binary classification task, which involves the presence of a "rockfall" class and a "background" class (which refers to the absence of any event). When one of the two classes is significantly overrepresented compared to the other, results might be skewed, as the model tends to favor the more represented class at the expense of the minority class (class imbalance [156]).

If the model is trained on a dataset where the vast majority of images are normal and only a small fraction show rockfall, a problem could arise.

Thankfully, there are several techniques developed to mitigate the adverse effects of class imbalance [157]. In this work, the authors provide an efficient tool to overcome class imbalance issues by leveraging an innovative methodology for oversampling, where the number of samples from the minority class is increased by replicating them or generating synthetic examples. This work of thesis followed this approach, as the density and number of rockfall samples in the simulated data can be set easily.

The following sections will be dedicated to presenting the datasets used to train the network from scratch. Generally, when performing training with random weights initialization on deep architectures like Mask R-CNN, it is advisable to provide a wide variety (and numerosity) of samples, to allow the network to proficiently learn to recognize important features [27]. In this sense the capabilities of Rockfall Simulator comes useful when generating numerous and diverse samples for the synthetic datasets used for training the neural network.

As levels of noise and rock size in the collapses are expected to affect the network performance, a systematic generation of samples with different noise levels and rock sizes have been generated in order to provide a wide range of examples to be learned.

5.2.1 DATASETS

A total of 9 different datasets have been produced with different noise levels and block dimensions. Each dataset has been provided with approximately 1700 images (with an 80/20 split, leading to 1300 training images and 400 validation images, for a total of roughly 12000 training images and 3500 validation images). For the purpose of simulating the training datasets, providing a single rock wall was considered a limiting choice, as the difference in

stratigraphy matters when detecting this kind of phenomena, due to interplay with measurement noise. For this purpose, the datasets have been simulated using two rock walls described in section 4, Glendale (Site 1) and Bulga (Site 2). Each rock wall has respectively contributed to 50% of the total sample numerosity.
Each image in the datasets is a tile with a resolution of 512x512 pixels, extracted from an original high-resolution image of the entire rock wall measuring approximately 3500x1500 pixels (which represent an approximately a 70 m x 30 m rock wall (GSD is equal to 2 cm)). For each simulated rock wall, approximately 1500 rockfall events have been modeled across the entire surface, resulting in an average of 40 to 50 instances of rockfall within each training tile. Figure 5-1 resumes the grouping of datasets as a function of the measurement noise and the dimensions of the blocks:

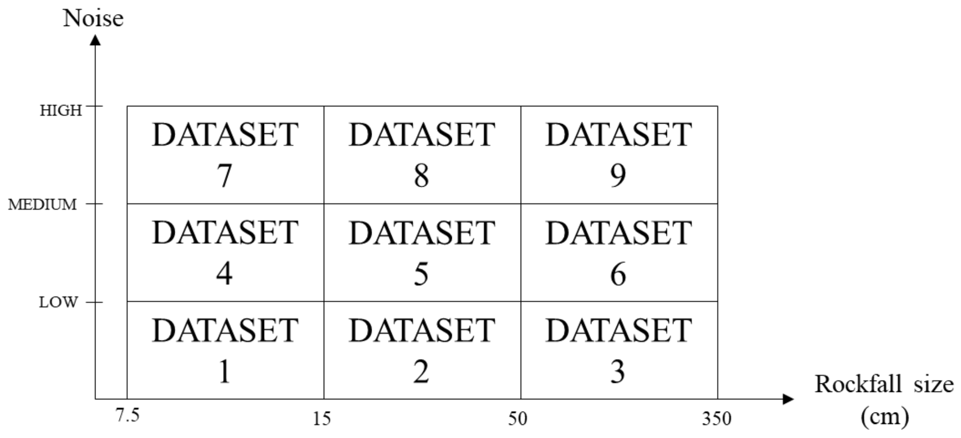


Figure 5-1: Training datasets scheme for measurement noise (Y axis) and rockfall sizes (X axis)

On the vertical axis of the chart three measurement noise thresholds are ranked:

1. **High noise**: refers to the noise levels in Table 5-1, representing the maximum measurement noise levels observed during the software calibration procedure.
2. **Medium noise**: 66% of the high noise simulation thresholds.
3. **Low noise**: 33% of the high noise simulation thresholds.

To check the noise level introduced by the simulation software, the standard deviation and the RMSE, observed on 45 raster samples of the simulated datasets per noise class are shown.

Table 5-1: Average noise values for the high noise dataset.

Mean (mm)	Mean StD (mm)	Mean RMSE (mm)
-0.1	23.8	24.4

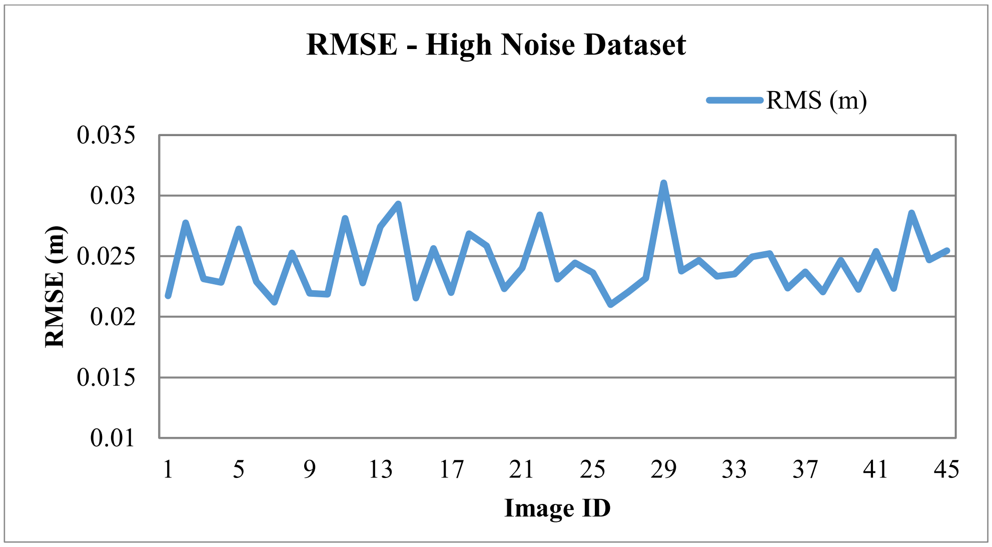


Figure 5-2: RMSE across difference maps for the high noise dataset.

Figure 5-2 shows the distribution of RMSE, which spans from 2.0 cm to 3.1 cm. The observed RMSE on the subset is approximately 2.4 cm.

Table 5-2: Average noise values for the medium noise dataset.

Mean (mm)	Mean StD (mm)	Mean RMSE (mm)
0.94	14.0	14.6

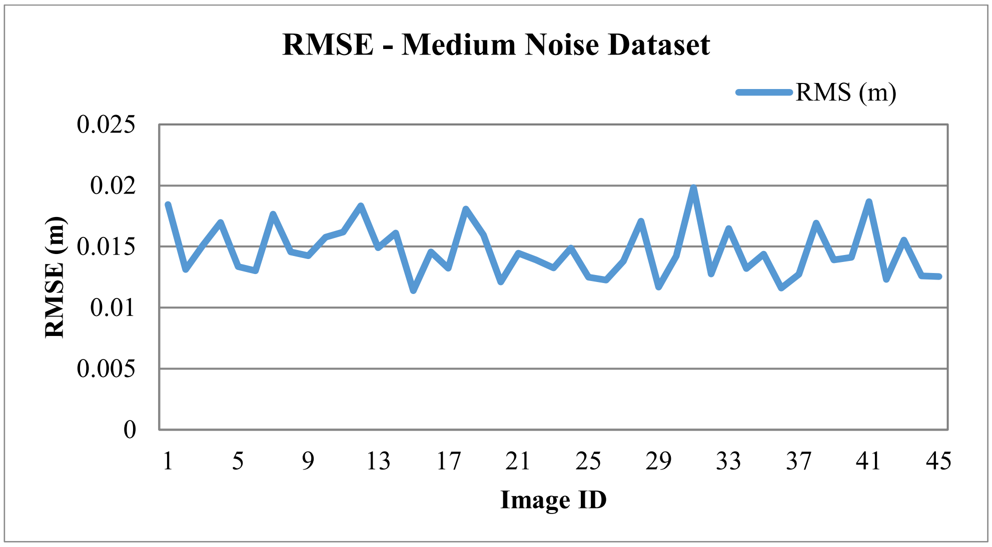


Figure 5-3: RMSE across difference maps for the medium noise dataset.

Figure 5-3 reports the statistics for a 45-sample dataset characterized by moderate noise levels. The RMSE is 1.4 cm, slightly below the 66% of the 2.5 cm benchmark established for higher noise class scenarios.

Table 5-3: Average noise values for the low noise dataset.

Mean (mm)	Mean StD (mm)	Mean RMSE (mm)
0.6	7.8	8.5

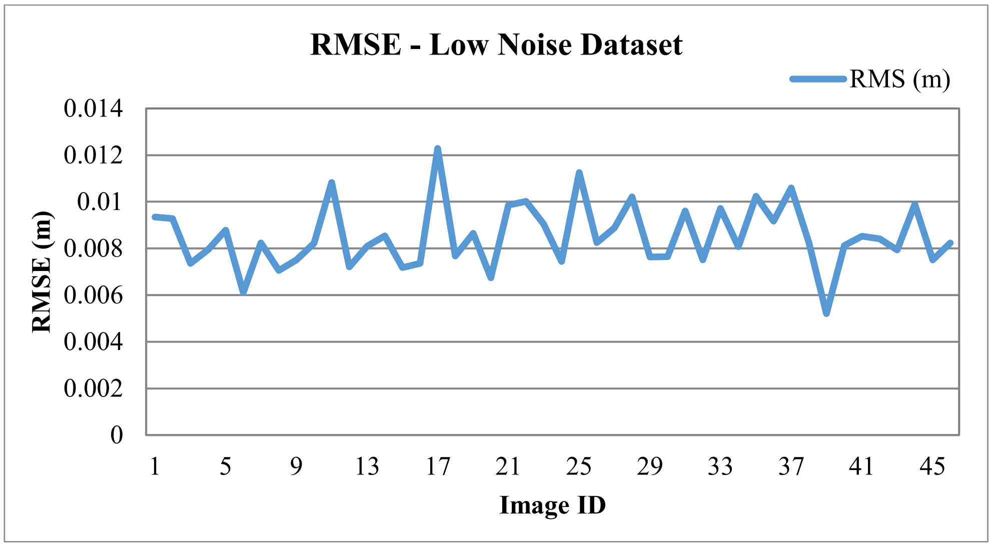


Figure 5-4: RMSE across difference maps for the low noise dataset.

Figure 5-4 reports the statistics of the low noise subset. The observed RMSE is 0.78 cm, well in agreement with the 33% of the noise threshold for the simulations.

It is an empirical fact that smaller rockfall events are more frequent [22] but less easily detected, thus posing a challenge for effective monitoring. Training the network to recognize these smaller events may possibly increase the chances of the neural network being able to detect them.

In previous research, a comprehensive evaluation of the frequency distribution across various size classes of rockfall events observed on the Glendale (NSW, Australia) [22] quarry rock wall was presented. Therefore, such empirical distribution of the block sizes has been taken as target for the simulation. As Rockfall Simulator utilizes randomly generated parameters extracted from a Gaussian distribution, the actual simulated distribution, composed by the sum of several Gaussian distributions, will only approximate the target one. However, an exact match is unnecessary, being the target histogram just an example of rockfall in an open pit mine.

Figure 5-5 shows the histogram of the relative frequencies of the rockfall dimensions for the simulated case.

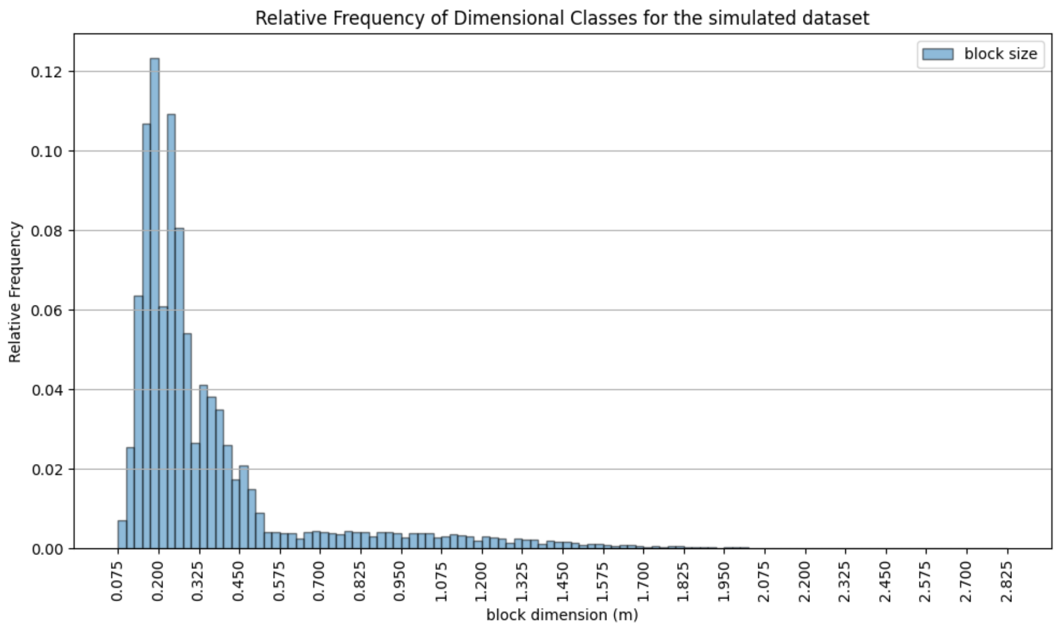


Figure 5-5: Histogram of the frequencies of classes referred to the dimensions of blocks.

Conversely, the simulation of larger rockfall blocks yields a sparser distribution of information within each image, thereby reducing the volume of training samples that can be generated. Given this, any practice to augment the number of images for datasets corresponding to larger rockfall blocks would be counterproductive, particularly when the primary aim of the research is devoted to the identification and classification of smaller rockfall blocks.

5.3 MODEL TRAINING

The training phase has been started with a random initialization of weights implemented in the Tensorflow library, drawing from a known distribution. A normal (Gaussian) distribution with mean value of 0 and standard deviation value of 1 have been used. The same initialization needs to be carried out for biases (as explained in section 2.3.1).
The training has been carried for several epochs, involving all the layers in the neural network structure. This is crucial when performing a training routine from scratch, as every branch of the network needs to adjust its weights.

The training was facilitated by a CUDA accelerated GPU equipped with 8 GB of RAM. Starting from a fresh state, without pre-existing knowledge from other datasets, the Mask R-CNN model was trained from scratch for ca. 350 epochs. A learning rate of 0.001 was selected for the training. Such a moderate learning rate strikes a balance - it's not too high to make the model converge chaotically, nor too low to drastically slow down training. It ensures stable convergence while allowing the model to learn meaningful patterns in the data over time.

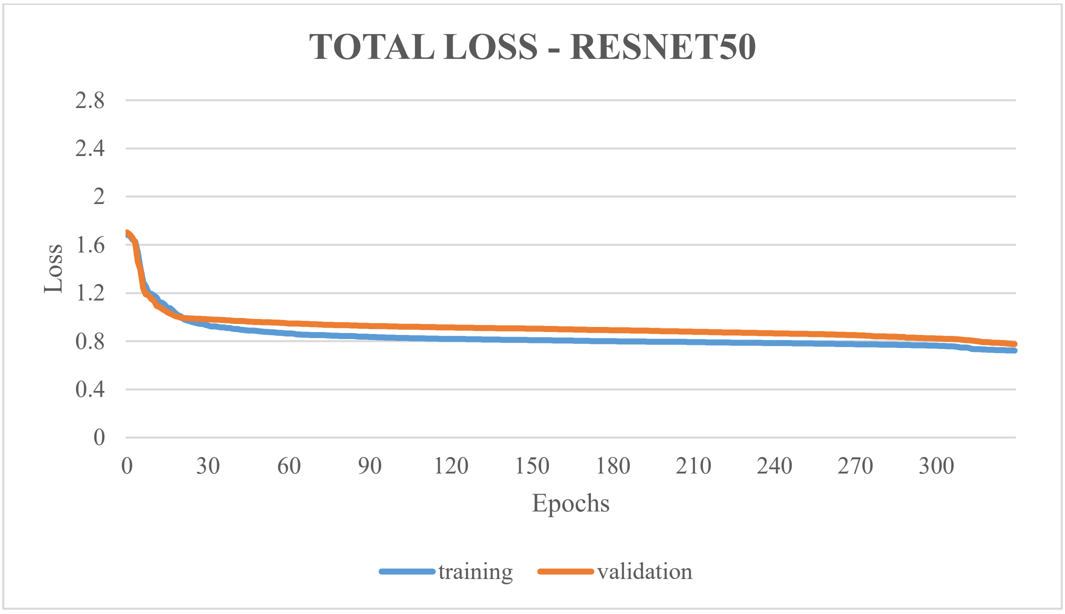
In the Mask R-CNN implementation used for the experiments two different feature extractors, namely ResNet101 and ResNet50, can be used. Theoretically, with its additional depth, ResNet101 can model more complex features and representations. This capability helps when the network must discern complex patterns improving feature extraction accuracy despite measurement noise. On the other hand, this network may be more sensitive to overfitting during training. For this purpose, the same training phase, with the same general settings, has been carried for both the feature extracts, to better evaluate their performances and possibly avoid overfitting.

The training sample image size (i.e., 512x512 pixels) allows to accommodate multiple images in a single minibatch during training. Compared to using a single image, minibatch training offers several advantages: processing multiple samples allows the GPU to parallelize operations, leading to faster training. Gradients calculated from minibatches tend to be less noisy than those from single samples, making the training process smoother. Training with minibatches can introduce a slight regularization effect, potentially leading to a model with better generalization to unseen data.

In the process of training deep learning models, especially those as intricate and specialized as Mask R-CNN, monitoring, and analyzing loss metrics is paramount. This section underscores the significance of observing loss and validation charts, particularly the total loss, bounding box (bbox) loss, classification loss and mask loss during the Mask R-CNN training process.

At the most fundamental level, the decline of loss over epochs or iterations indicates that the model is learning – i.e., it's adjusting its weights based on the given data and its corresponding targets. A stabilized loss suggests convergence, telling us the model might not significantly improve with further training on the current dataset. In order to optimize the computational efficiency and prevent overfitting during model training, an early stopping procedure has been implemented. Specifically, at the end of each epoch, the average gradient of the validation loss function over the most recent 10 epochs is computed. Should this average gradient descend below a user-specified threshold, the training process is terminated.

By comparing the training loss with the validation loss, it is possible to gauge the model's generalization capabilities. A model that performs well on training data (low training loss) but poorly on validation data (high validation loss) may be overfitting, suggesting it might not perform well on unseen data. The following charts will report the training results for both Resnet-50 and ResNet-101 as features extractors in Mask R-CNN.



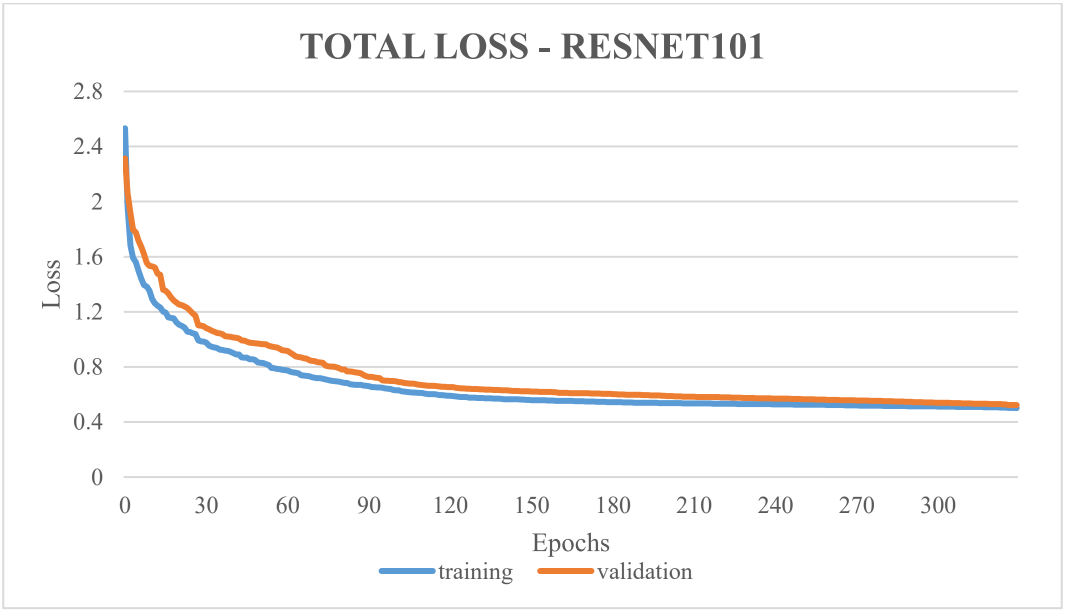


Figure 5-6: Total loss chart (training and validation) for Mask R-CNN model: above) ResNet-50, below) ResNet-101.

The blue line represents the training loss while the orange line represents the validation loss in Figure 5-6. Within this picture, the training loss shows a consistently descending pattern, meaning that the network is efficiently learning from the simulated data.

At first glance, the major difference between the two charts is certainly the slope of the curves. In the first case (ResNet-50), the training curve tends to stabilize at around 90 epochs, as a general flattening of the training loss is observed. The same pattern is not coherently replicated on the second case, as the training loss effectively stabilizes after around 150 epochs. To get a better insight on the validation performances of the network, Figure 5-7 illustrates the total loss comparison between the two ResNets.

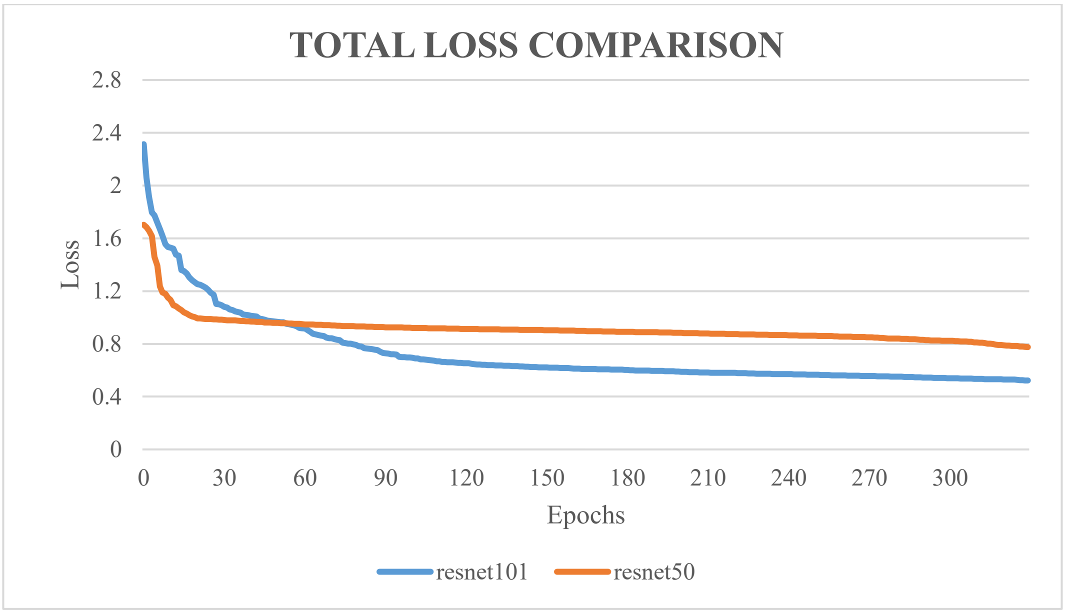


Figure 5-7: Total validation comparison for ResNet-50 (orange line) and ResNet-101 (blue line).

In Figure 5-7, the orange curve represents ResNet-50 validation loss while the blue one represents its 101-layer variant.

In this regard, ResNet-50's validation loss diminishes at a more accelerated rate during the initial epochs, implying a quicker adaptation to the training data compared to its 101-layer counterpart. This observation aligns with theoretical expectations. However, ResNet-50 being a less complex model, has a total validation loss that is consistently 20% to 25% higher than that of ResNet-101.

Notably, the validation loss curves intersect at approximately the 60-epoch mark, with a shared total loss value of around 0.9. Beyond this point, ResNet-101 maintains a more pronounced downward trajectory compared to the slower rate of decrease exhibited by

ResNet-50. In summation, ResNet-101, despite its initial higher loss, achieves a better performance (lower loss).

To get a better understanding of the validation error behavior, the loss contributions for Mask R-CNN have been disaggregated.

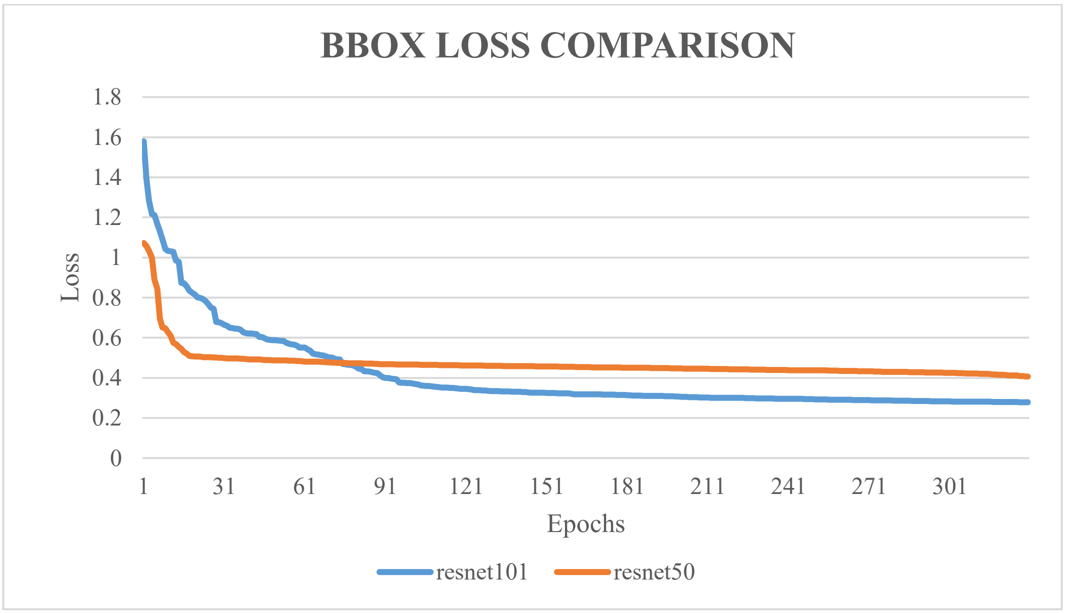


Figure 5-8: Bounding box regressor chart (validation) for Mask R-CNN model.

Bounding box loss function calculates an overall displacement over the bounding box pixel coordinates (namely the image coordinates of the lower left corner of the bounding box and its height and width). The cumulative loss is the sum of all the contribution of the L1 Smooth Loss reported in section 2.4.2 which represent the difference from the ground truth RoIs coordinates to the predicted instances bounding boxes.

Analyzing figure 5-8, both models start their learning curve with a high loss. ResNet-101 initiates with a slightly more elevated loss in comparison to ResNet-50. The initial phase of the training (up to approximately epoch 30) showcases a steep decline in the loss for both architectures. ResNet-50's trajectory in this phase is slightly steeper, suggesting a faster convergence rate during these initial epochs compared to ResNet-101. This is an expected behavior due to the different complexity of the two architectures. After the sharp descent in the initial epochs, both the architectures experience a phase where the loss starts to reach a plateau. ResNet-101 stabilizes at a lower value than ResNet-50, thereby suggesting an advantage in terms of bounding box accuracy. In the end, the trend remains unaltered as the 101-layer variant experiences lower loss values for the bounding box regressor. At this stage, the behavior heavily recalls the total loss function.

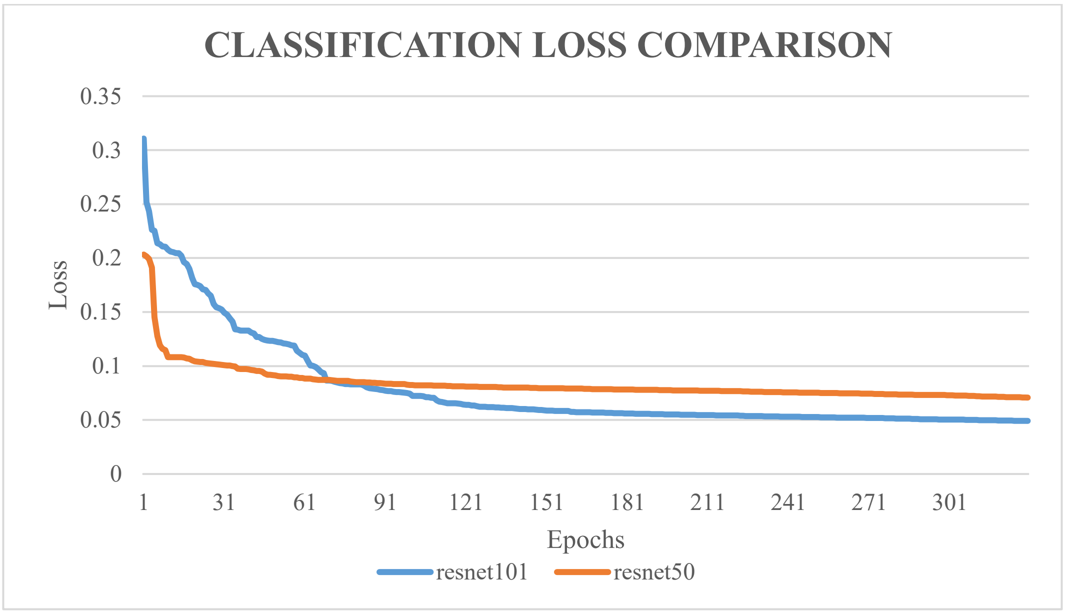


Figure 5-9: Loss for the classifier (validation) in Mask R-CNN model.

Figure 5-9 presents the analysis of the classification loss occurred during the training phase of two convolutional networks. Classification loss is calculated from the probability attributed to a positive anchor box being located on a ground truth instance.

From the beginning, both architectures demonstrate a rapid decline in loss values, which is indicative of the model's adeptness in adjusting its weights efficiently in response to the presented training data. However, certain distinctions between the two models become palpable. ResNet101 manifests a slower descent in its validation loss during the initial epochs compared to ResNet50. This can be attributed to the more profound depth and complexity of ResNet101, which entails a more extensive set of parameters, possibly requiring higher quantity of data in order to achieve the same goal.

On the other hand, around the 60th epoch, a noteworthy convergence of the loss curves is observable. Both models stabilize, showcasing minute fluctuations as the epochs advance.

This plateau suggests that both models, after substantial iterations, have potentially reached a point where no further improvements are experienced. It must be noticed that classification loss exhibits lower values with respect to the previous bounding box loss. The explanation lies in the nature of the given problem, as the network has been exploited for a simpler binary classification problem while its complexity may overcome classification problems of higher complexity.

However, while both models maintain a similar trend, ResNet50's loss appears marginally higher compared to ResNet101. This persistent difference in loss values could be indicative of ResNet50's inherent limitations in capacity when compared to its deeper counterpart, ResNet101.

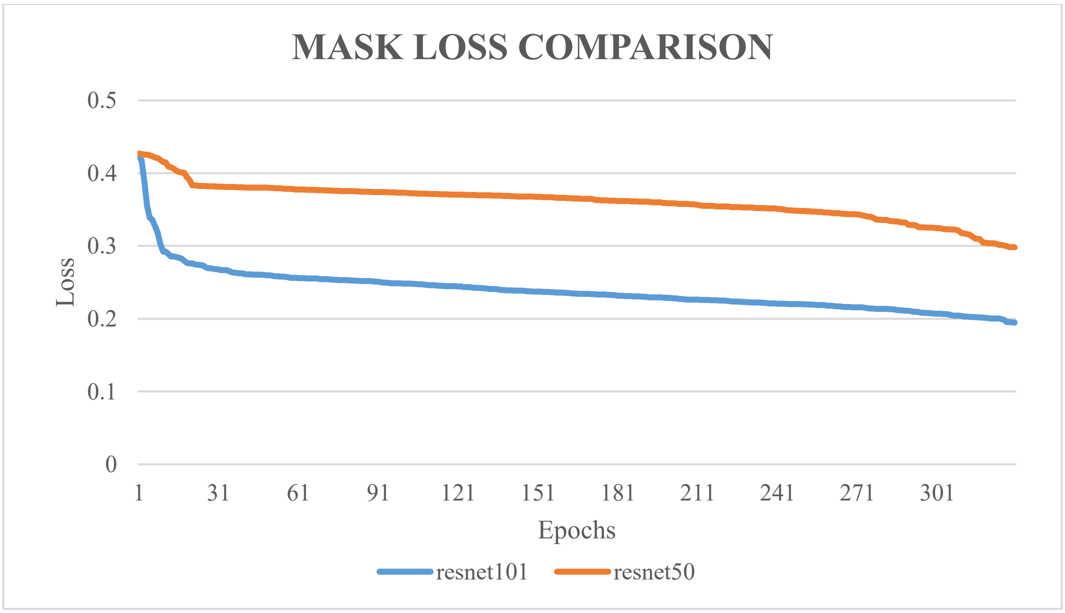


Figure 5-10: Loss for the pixel classifier (validation) in Mask R-CNN model.

The mask loss metric essentially quantifies the incongruences between predicted segmentation masks and their true annotations at a per-pixel level.

Initially, both architectures exhibit a quick decline in mask loss. However, ResNet101's loss trajectory is characterized by a more pronounced initial decrease as opposed to ResNet50.

Approaching the 60th epoch, the loss trajectories for both architectures begin to manifest a trend of deceleration. The gradual attenuation in the rate of decline suggests that the models have reached a saturation point.
While both trajectories taper off, ResNet50's mask loss remains consistently higher relative to ResNet101 throughout the latter epochs. This enduring disparity underscores ResNet101's superior ability in refining its per-pixel classification tasks over prolonged training, which may be resultant from its deeper architecture and greater parameter count, facilitating a better classification of each pixel.
In summation, the mask loss comparison delineates the intrinsic advantages of ResNet101, particularly in the domain of image segmentation tasks. Nevertheless, it's pivotal to weigh these benefits against computational demands and the specific objectives of the segmentation task when electing an appropriate architecture for deployment.

5.3.1 VALIDATION RESULTS

To provide a more comprehensive understanding of the network's performance it was decided to further process the full validation dataset. Loss curves can be incomplete indicators of a model's generalization capacity as they fail to reveal the model's efficacy in an operational context.
Consequently, the validation dataset was processed further through the performance evaluation code (section 2.4.4) to yield confusion matrices for each classification category. Confusion matrices are a fundamental diagnostic tool in supervised learning, providing a granular account of a classifier's performance across different classes [158]. Each confusion matrix is structured as a table, reporting, in this case, four essential elements: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These elements allow for the computation of other metrics such as Precision, Recall, the F1-score, as well as aggregate metrics like mean Average Precision (mAP) and mean Average Recall (mAR), which are detailed in section 2.4.4. Computing the True Negatives in this context makes no sense since ground truth annotated data only highlights the actual rockfall events (Positives) as it does the output of the neural network. The statistical indices do not require TN in their computation.

Results have been considered on the base of different IoU (section 2.4.4) thresholds, namely 0.25 and 0.75, and for different prediction confidence levels, namely 0.9, 0.95 and 0.99.
In the context of object detection, the confidence threshold is a hyperparameter that controls the selection of bounding boxes and their associated object classes. Only predictions with a confidence score above the threshold are considered to be valid detections, therefore filtering out low-confidence predictions that are more likely to produce false positives.
For this reason, a lower confidence threshold would increase the sensitivity of the model, allowing it to detect more objects, possibly including those that are harder to recognize. However, this comes at the cost of specificity, as the model would also produce more false positives. Conversely, a higher threshold improves specificity but may result in more false negatives.
For the same reason, the confidence threshold has a direct impact on the Precision-Recall curve. Lowering the confidence threshold tends to increase recall but reduce precision, and vice-versa. Depending on the application, one may prioritize precision (minimizing false positives) over recall (minimizing false negatives), or vice versa.
The following tables (Table 5-4 to Table 5-6) resume the comprehensive results on the validation datasets. Results have been reported for each of the 9 datasets, with different confidence levels and IoU thresholds.

Table 5-4: Validation results for the low noise datasets. Please note that each dataset is composed of a single dimensional class of blocks, as explained the previous section.

LOW NOISE (STD = 1 CM)

	DATASET 1					
CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	1696	1676	1607	1569	1427	1375
FP	129	189	124	162	111	145
FN	79	99	168	206	348	400
AP	92.93%	89.87%	92.84%	90.64%	92.78%	90.46%
AR	95.55%	94.42%	90.54%	88.39%	80.39%	77.46%
F1	94.22%	92.09%	91.67%	89.50%	86.15%	83.46%

	DATASET 2					
CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	3081	2912	3045	2888	2935	2810
FP	90	221	36	195	32	171
FN	77	246	113	270	223	348
AP	97.16%	92.95%	98.83%	93.67%	98.92%	94.26%
AR	97.56%	92.21%	96.42%	91.45%	92.94%	88.98%
F1	97.36%	92.58%	97.61%	92.55%	95.84%	91.55%

	DATASET 3					
CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	2262	2259	2251	2248	2249	2245
FP	2	3	2	3	1	9
FN	4	7	15	18	17	21
AP	99.91%	99.87%	99.91%	99.87%	99.96%	99.60%
AR	99.82%	99.69%	99.34%	99.21%	99.25%	99.07%
F1	99.87%	99.78%	99.62%	99.54%	99.60%	99.34%

Table 5-5: Validation results for the medium noise datasets.

MEDIUM NOISE (STD = 1.5 CM)

	DATASET 4					
CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	6811	6479	6514	6214	5585	5287
FP	610	942	360	660	75	273
FN	477	809	774	1074	1703	1901
AP	91.78%	87.31%	94.76%	90.40%	98.67%	95.09%
AR	93.45%	88.90%	89.38%	85.26%	76.63%	73.55%
F1	92.61%	88.10%	91.99%	87.76%	86.27%	82.95%

DATASET 5

CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	8318	8047	8418	7980	8027	7698
FP	367	838	205	643	51	380
FN	303	774	403	841	694	1123
AP	95.77%	90.57%	97.62%	92.54%	99.37%	95.30%
AR	96.49%	91.23%	95.43%	90.47%	92.04%	87.27%
F1	96.13%	90.90%	96.51%	91.49%	95.57%	91.11%

DATASET 6

CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	2138	2136	2138	2137	2130	2129
FP	7	11	5	7	5	4
FN	12	14	12	13	20	21
AP	99.67%	99.49%	99.77%	99.67%	99.77%	99.81%
AR	99.44%	99.35%	99.44%	99.40%	99.07%	99.02%
F1	99.56%	99.42%	99.60%	99.53%	99.42%	99.42%

Table 5-6: Validation results for the high noise datasets.

HIGH NOISE (STD = 2.5 CM)

	DATASET 7					
CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	7047	6422	6674	6109	5620	5234
FP	1561	2363	880	1439	208	590
FN	479	1104	852	1417	1906	2292
AP	81.87%	73.10%	88.35%	80.94%	96.43%	89.87%
AR	93.64%	85.33%	88.68%	81.17%	74.67%	69.55%
F1	87.36%	78.74%	88.51%	81.05%	84.17%	78.41%

DATASET 8

CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	7172	6701	7072	6634	6681	6352
FP	889	1179	346	784	192	521
FN	152	623	252	690	643	972
AP	88.97%	85.04%	95.34%	89.43%	97.21%	92.42%
AR	97.92%	91.49%	96.56%	90.58%	91.22%	86.73%
F1	93.23%	88.15%	95.94%	90.00%	94.12%	89.48%

DATASET 9

CONFIDENCE	**0.9**		**0.95**		**0.99**	
IOU	**0.25**	**0.75**	**0.25**	**0.75**	**0.25**	**0.75**
TP	2089	2088	2087	2085	2089	2088
FP	11	18	10	16	11	18
FN	27	28	29	31	27	28
AP	99.48%	99.15%	99.52%	99.24%	99.48%	99.15%
AR	98.72%	98.68%	98.63%	98.53%	98.72%	98.68%
F1	99.10%	98.91%	99.07%	98.89%	99.10%	98.91%

Table 5-7: Aggregate results for the validation datasets.

	CONFIDENCE = 0.9		**CONFIDENCE = 0.95**		**CONFIDENCE = 0.99**	
	IoU = 0.25	**IoU = 0.75**	**IoU = 0.25**	**IoU = 0.75**	**IoU = 0.25**	**IoU = 0.75**
MAP	91.42%	86.38%	95.02%	90.04%	98.09%	93.93%
MAR	97.16%	91.81%	94.74%	89.77%	87.42%	85.25%
F1-SCORE	94.20%	89.02%	94.88%	89.91%	92.44%	89.38%

Table 5-7 provides the aggregate results for the nine different datasets. If the higher confidence level (0.99) is used along with the less restrictive IoU threshold (0.25) the model seems exhibiting very good performances. With an Average Precision (AP) peaking at 98%, it is evident that, in this configuration, the model consistently produces precise object detections over a varied range of recall. The Average Recall (AR) of 87%, also underlines the model's does not produce too many false negatives regardless of the high confidence level. The F1-Score, a metric denoting the balance between precision and recall, corroborates this observation with a score of 93%.
Increasing the IoU threshold (0.75) makes the labelling of True vs False Positive more restrictive. Here, the AP recedes to 95%. The associated AR value of 85%, is not significantly different from the previous results. This is further emphasized by the F1-Score of 89%, which suggests a slight trade-off between precision and recall under these parameters.
While a lower IoU, such as 0.25, is more forgiving, allowing for moderate deviations between the predicted and ground truth bounding boxes, an IoU of 0.75 demands a substantial degree of accuracy in object localization. Furthermore, as the confidence threshold escalates, the model becomes more discerning, which, while amplifying precision, could also escalate the risk of false negatives, as observed in the AR values at the highest confidence level.

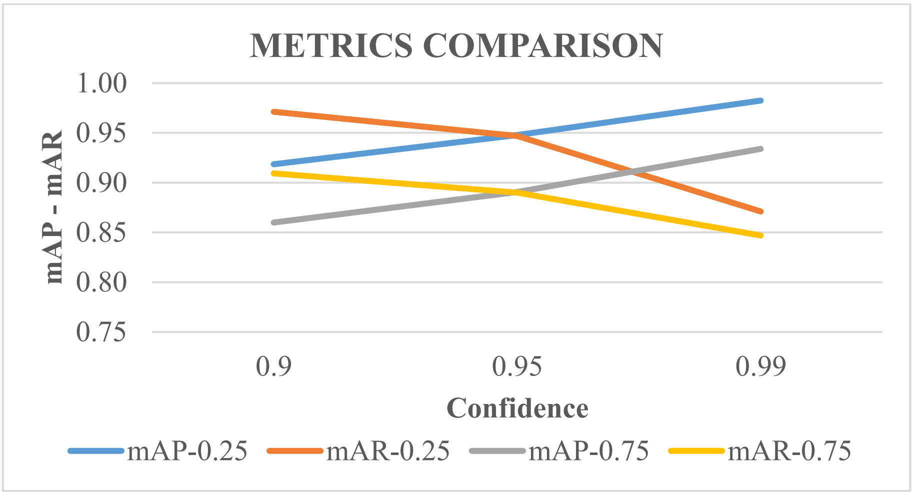


Figure 5-11: Comparison chart for the evaluation metrics on different confidence and IoU thresholds.

Figure 5-11 offers a highlight on the relationship between mean Average Precision (mAP), mean Average Recall (mAR), Intersection over Union (IoU) thresholds, and confidence thresholds.

As obviously the mean Average Precision (mAP) and mean Average Recall (mAR) curves intersect at a confidence level of 0.95, this can be taken as an equilibrium point between false positives and false negatives.

Precision, which is significantly impacted by the rate of false positives, and recall, influenced by the count of false negatives, are mutually dependent metrics. The optimal operational point for the neural network is therefore reasonably chosen midway between a too strict and a too relaxed confidence level.

The degree of overlap between prediction and ground truth can be employed to measure the goodness of the rockfall identification. This is encapsulated in the Intersection over Union (IoU) metric, which measures the extent of overlap between the predicted and ground truth bounding boxes.

An IoU threshold of 0.75 was elected for subsequent analyses, ensuring a satisfactory overlap between the predicted and actual areas, serving as a robust, yet not overly stringent, criterion to account for noise-induced deviations.

Table 5-8: Cumulative results for the validation datasets with confidence level = 0.95 and IoU = 0.75.

CONFIDENCE = 0.95 / IOU = 0.75			
	DATASET 7	**DATASET 8**	**DATASET 9**
MAP	80.94%	89.43%	85.03%
MAR	81.17%	90.58%	89.65%
F1	81.05%	90.00%	87.28%i
	DATASET 4	**DATASET 5**	**DATASET 6**
MAP	90.67%	92.71%	86.58%
MAR	88.01%	92.73%	91.01%
F1	89.32%	92.72%	88.74%
	DATASET 1	**DATASET 2**	**DATASET 3**
MAP	96.20%	93.74%	90.47%
MAR	88.39%	91.45%	92.59%
F1	92.13%	92.58%	91.52%

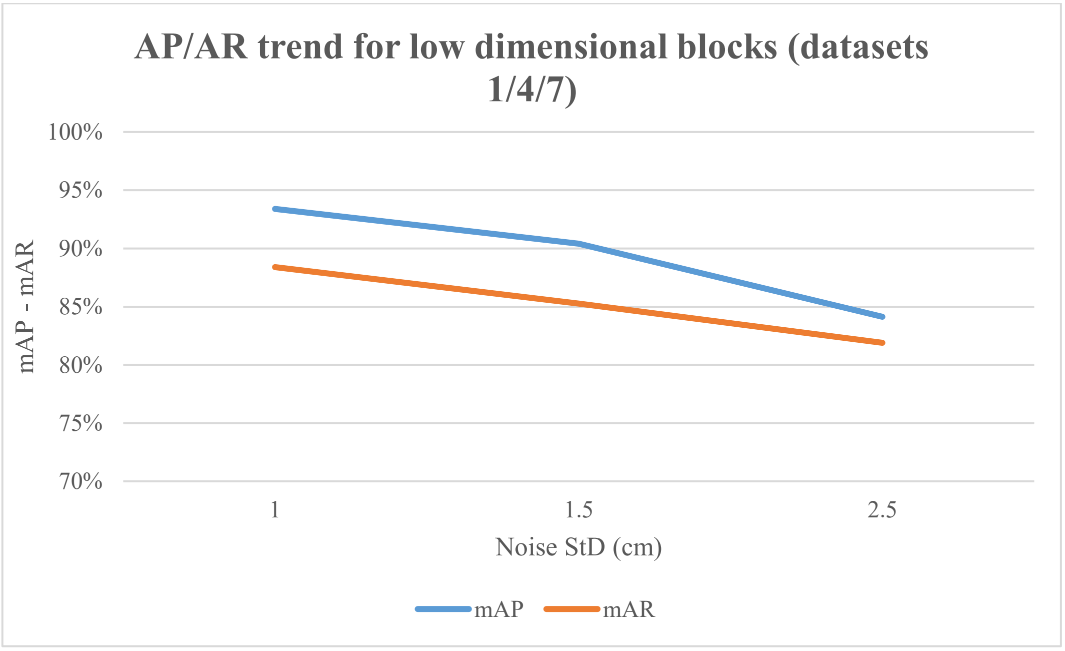


Figure 5-12: Precision and recall trend for the small size datasets as a function of increased measurement noise.

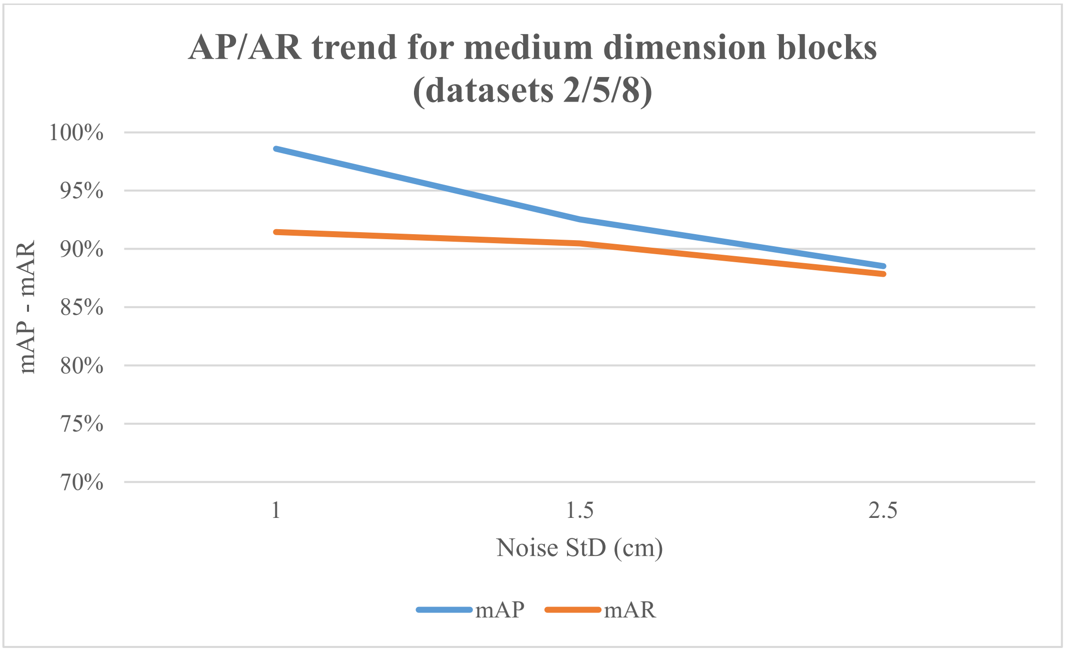


Figure 5-13: Precision and recall trend for medium dimensions blocks.

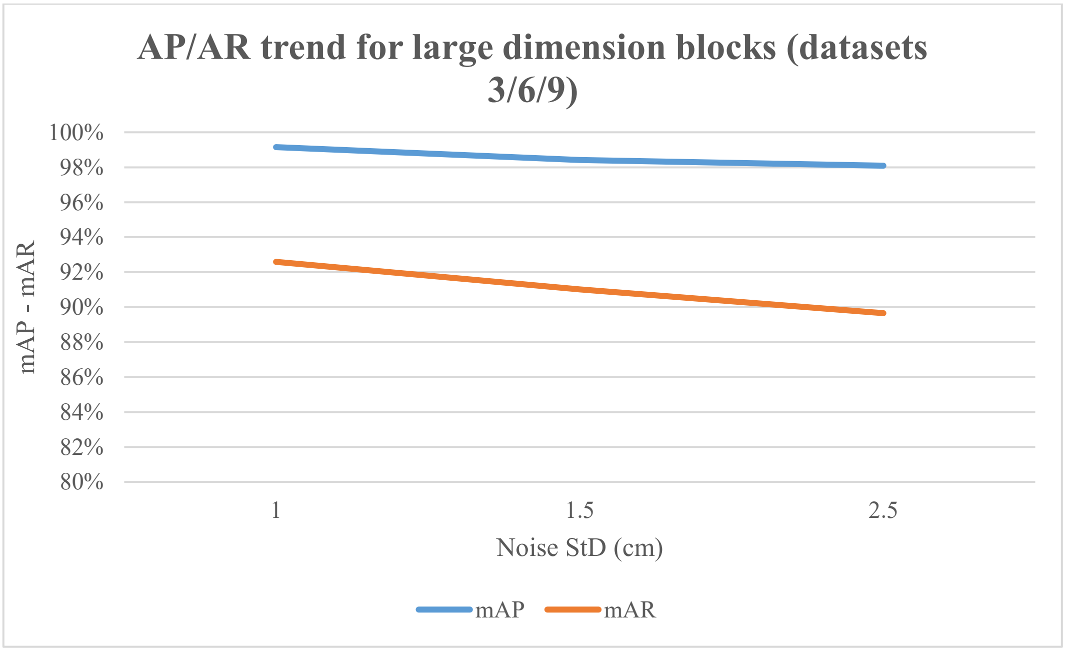


Figure 5-14: Precision and recall trends for larger dimension blocks class.

The above charts represent the performance of the neural network on the validation datasets in terms of block sizes and noise levels. he results are divided into three distinct categories based on the size of the blocks: small (7.5 to 15 cm), medium (15 to 50 cm), and large (50 cm to 3.5 m). Each graph represents the mean Average Precision (mAP) and mean Average Recall (mAR) trends across increasing levels of noise, ranging from 1 cm to 2.5 cm in noise standard deviation. Please note the difference in terms of scale on the vertical axis.

In the case of blocks with smaller dimensions, specifically those ranging from 7.5 cm to 15 cm, a steep decrement is evident in both mAP and mAR as the noise level escalates. This tendency highlights the network's susceptibility to noise factors for this size class. The decline in mAP is indicative of an increasing rate of false positives, which occurs concomitantly with a reduction in true positives.

Turning the attention to medium-sized blocks, which fall between 15 cm and 50 cm, a more resilient performance is observed. The mAR curve remains relatively invariant across an array of noise levels, suggesting that the rate of false negatives—directly influencing recall metrics—does not substantially increase. Nonetheless, a modest decline in precision is discernible at elevated noise levels. This can be attributed to a surge in false positives, presumably instigated by the perturbations of measurement noise. Importantly, the observed decrement in precision is less pronounced when compared to the data depicted in Figure 5-12, owing to a less drastic reduction in the number of true positives.

Finally, for blocks of larger dimensions, as shown in Figure 5-16, the AR curve manifests remarkable stability, with a minute 3% variation in mAR on the highest noise level, implying that the network's capability for block identification is largely impervious to noise interference at this size range. Conversely, the AP curve shows a less pronounced deviation, which can be described in the lower false positive count on higher noise levels.

5.4 TEST RESULTS ON SYNTHETIC DATASETS

Testing is a critical phase in every neural network training process. While the training is dedicated to "teaching" the network how to perform its task and, through validation, simultaneously verifying the adjustments on an unseen dataset with similar structure, testing aims to test the network under "unknown" conditions, to evaluate its generalization capability.

For this reason, in addition to the two previous rock walls used for training and validation, two additional test sites have been included, namely the Pilkington rock wall (NSW, Australia) and Pietra di Bismantova rock wall (Emilia-Romagna, Italy). The testing was structured to take in consideration higher levels of measurement noise, compared to the training phase.

Feeding the network data with a different range of measurement noise levels compared to training phase facilitates a more comprehensive interpretation of the neural network's performance metrics under diverse and less-controlled conditions.

Three levels of noise were defined:

1. The first level reproduces the original conditions of the datasets as described in section 4.2.3, with a standard deviation of around 2.5 cm. This condition serves as a baseline scenario and provides a point of union between validation and test of the neural network's performance.
2. In the second level, the noise standard deviation is doubled to 5 cm.
3. The third level is the most challenging condition and is defined by a standard deviation of 7.5 cm. The results have been processed with a prediction confidence of 0.95, which has been described in the previous section as the best compromise between precision (which is function of the TPs and FPs) and recall (which is function of TPs and FNs). Moreover, an IoU of 0.75 has been chosen for the purpose, as it provides a more stringent constraint to what is labelled as a TP. The following sections illustrates the results for the four different test sites. Please note that the tables are coherently formatted to Figure 5.1, which illustrates the distribution of the datasets with regards to measurement noise levels and rockfall events dimensions.

Datasets 1, 2 and 3 refer to the first noise level reported in Table 5-4. Datasets 4, 5 and 6 refer to the second noise level. Lastly, Datasets 7, 8 and 9 refer to the third noise level reported in table 5-6.

5.4.1 TEST SITE 1 - GLENDALE

The following section illustrates the results for the Glendale test site. Table 5-9 sums up the different indices for each dimensional block class and for each level of noise.

Table 5-9: Synthetic results for Glendale (Site 1) test site.

	DATASET 7	**DATASET 8**	**DATASET 9**
AP	37.27%	92.47%	96.08%
AR	41.49%	76.96%	90.74%
F1	39.27%	84.00%	93.33%
	DATASET 4	**DATASET 5**	**DATASET 6**
AP	72.00%	96.97%	99.02%
AR	73.58%	82.18%	90.18%
F1	72.78%	88.97%	94.39%
	DATASET 1	**DATASET 2**	**DATASET 3**
AP	82.07%	98.08%	98.97%
AR	82.20%	89.47%	91.43%
F1	82.13%	93.58%	95.05%

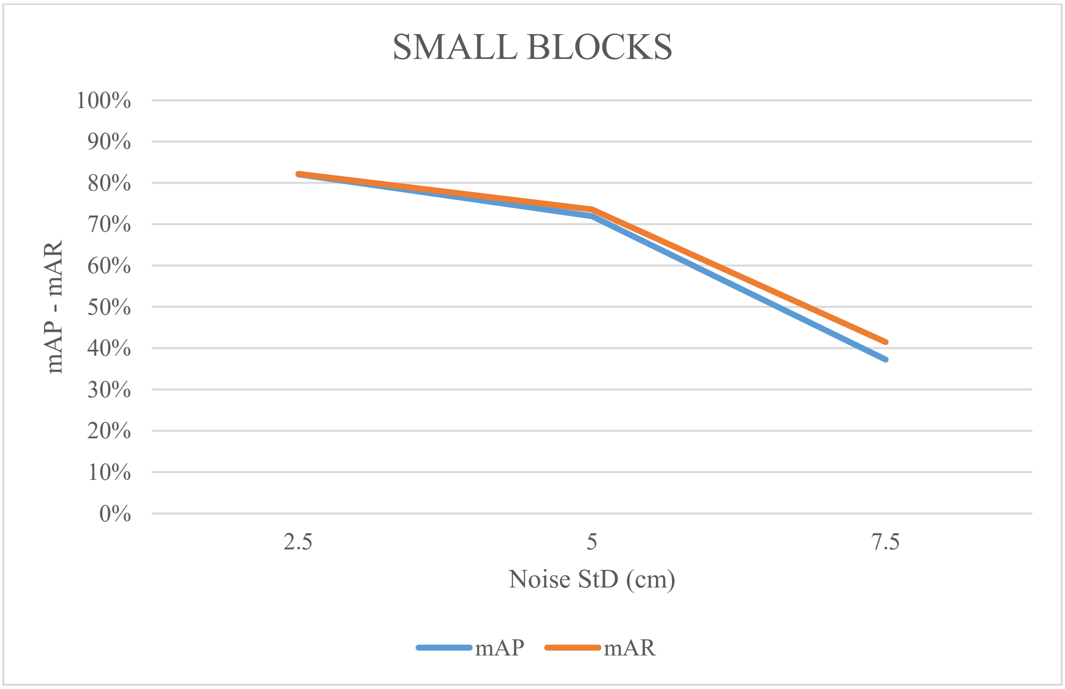


Figure 5-15: mAP – mAR trend curves for the small blocks in Glendale test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

The chart in Figure 5-15 provides the trends for mAP (blue line) and mAR (orange line) for the smaller dimensions of blocks. Starting with the lowest value of measurement noise (StD = 2.5 cm), it can be observed that the indices provide coherent values to the ones provided by the validation dataset. At this stage, doubling the measurement noise entity does not dramatically decrease the performances of the network, as only a 10% decrease in precision and recall is observed. On the other hand, with the highest level of noise (StD = 7.5 cm), the performance is highly impacted. The precision loss is larger, meaning that the increase in the false positive number is larger than the increase in false negatives. This is attributable to the network being misled by measurement noise spatial distribution, which can be often confused as a possible small rockfall event. Within this picture, the decrease in precision from level 2 to level 3 measurement noise is almost triple (10% from level 1 to level 2, 30% from level 2 to level 4).

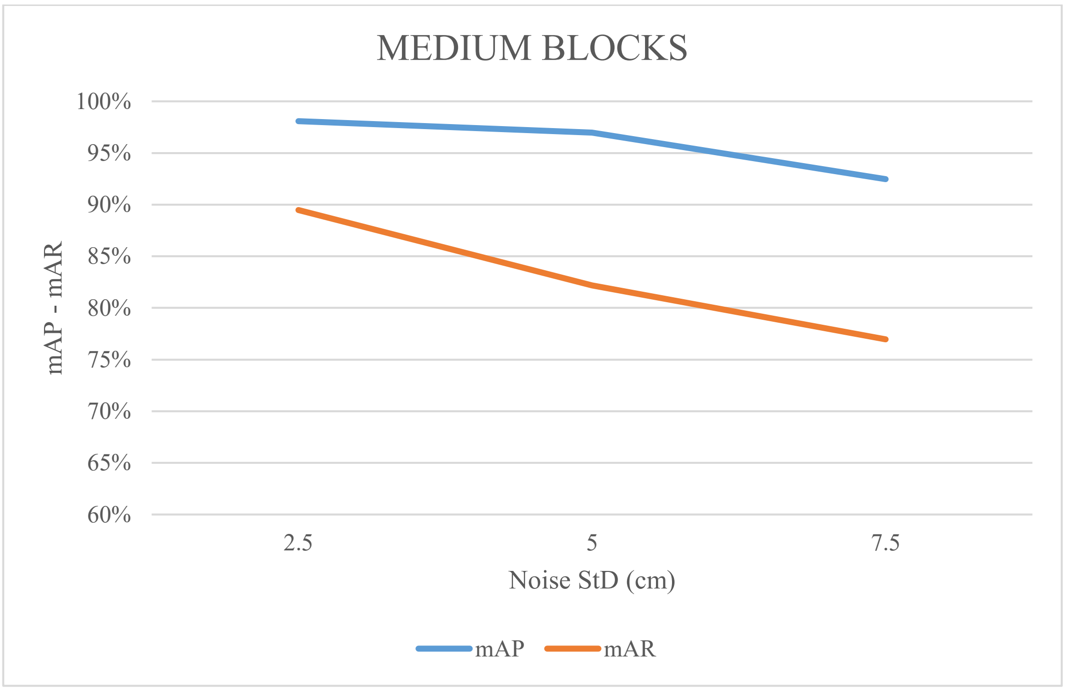


Figure 5-16: mAP – mAR trend curves for the medium blocks in Glendale test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

A different trend can be observed for medium sized blocks. For this dimensional class, roughly a 10% decrease in precision is observed from level 1 to level 3 noise, meaning that the network is less sensitive to false positives in this class. A slightly more pronounced decline can be observed for the recall curve, which highlights a lower influence of the measurement noise in the identification of true positives.

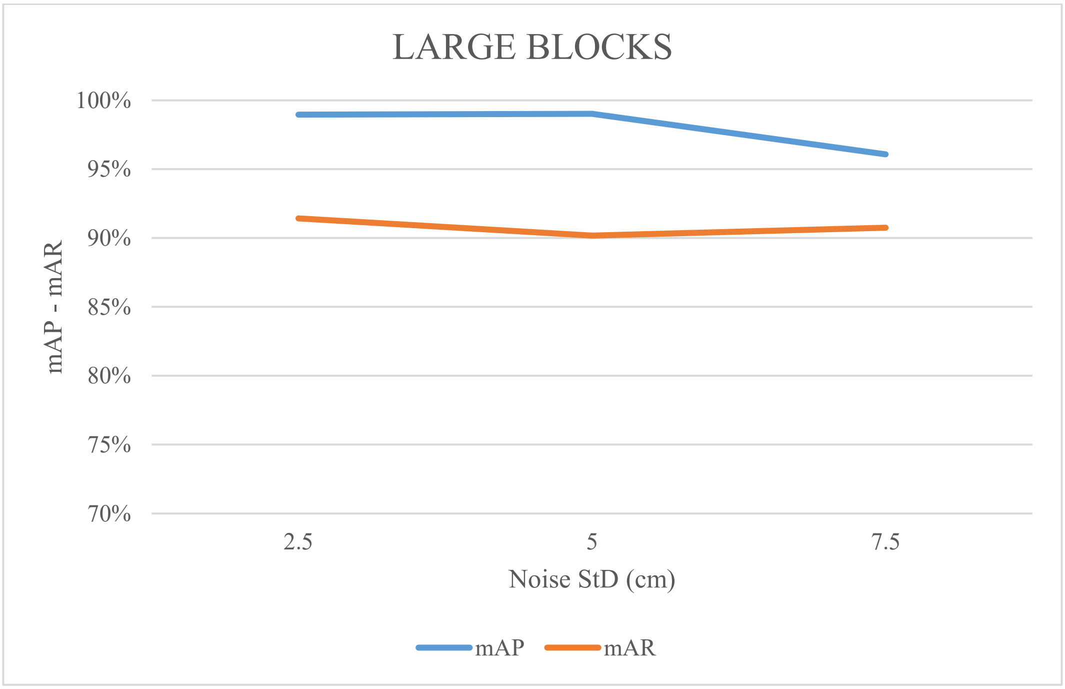


Figure 5-17: mAP – mAR trend curves for the large blocks in Glendale test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

Lastly, for the larger blocks, a somewhat constant trend is observed across the different noise levels, meaning that the influence for such class is relatively low. Comparing the mAP curves for the three-dimensional classes, it can be observed that the increase in the numerosity of false positives is higher on the smaller blocks class, which implies that the measurement noise is more easily mistaken as rockfall in such block size class.

5.4.2 Test Site 2 - Bulga

The following section illustrates the results for the Bulga test site.

Table 5-10: Synthetic results for Bulga (Site 2) test site.

	DATASET 7	DATASET 8	DATASET 9
AP	43.78%	93.12%	95.65%
AR	49.10%	77.42%	90.72%
F1	46.28%	84.55%	93.12%
	DATASET 4	**DATASET 5**	**DATASET 6**
AP	70.58%	96.34%	97.06%
AR	79.45%	85.92%	92.52%
F1	74.75%	90.83%	94.74%
	DATASET 1	**DATASET 2**	**DATASET 3**
AP	85.39%	98.33%	99.04%
AR	88.11%	94.90%	89.57%
F1	86.73%	96.58%	94.06%

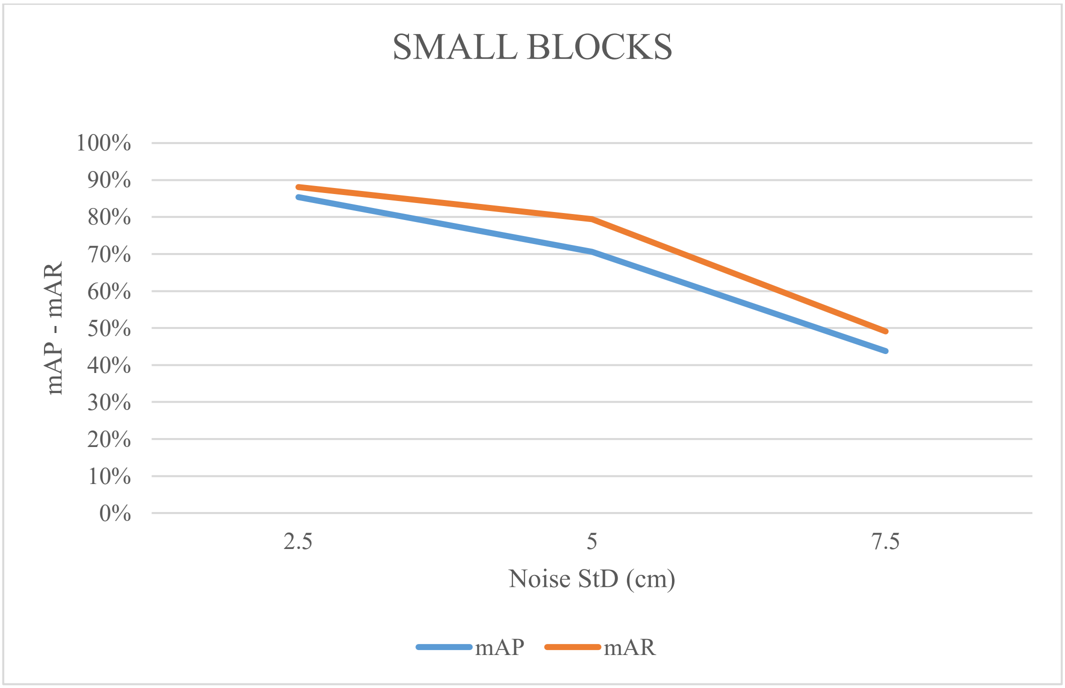


Figure 5-18: mAP – mAR trend curves for the small blocks in Bulga test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

A roughly same pattern can be observed on the Bulga test site. With respect to test site 1 (Glendale), there is a more pronounced divergence between the precision and recall curves. In this case, the mAP (blue curve) shows slightly lower values with respect to the test site 1. This implies that there is a higher numerosity of false positives in this case, which may be due to a different interaction of the measurement noise with the morphology of the rock wall.

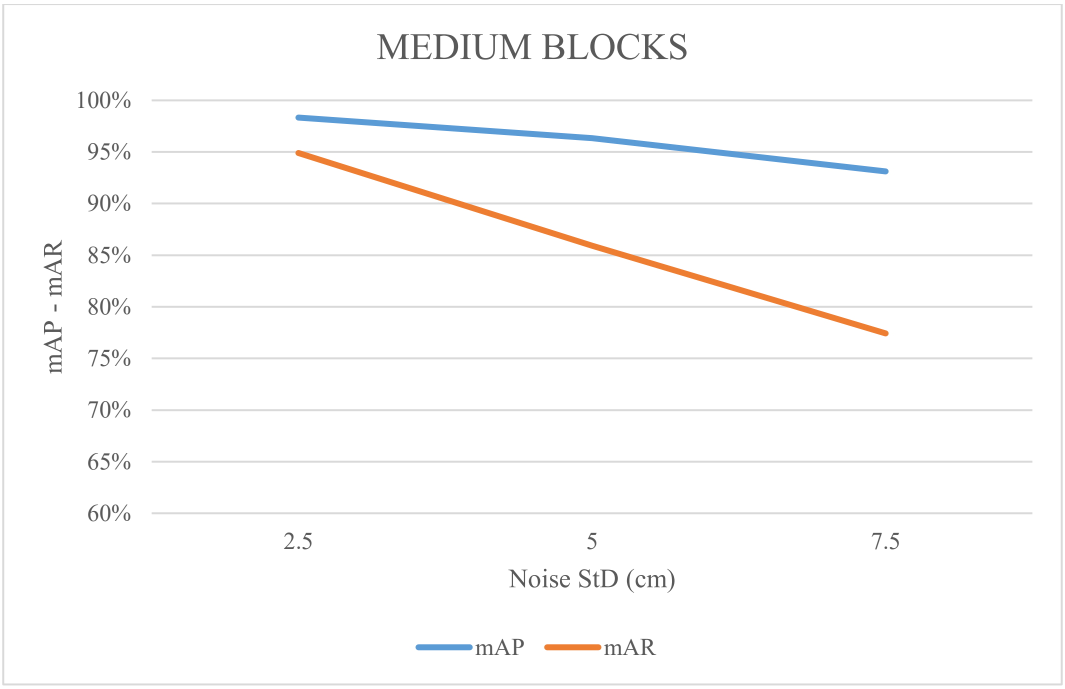


Figure 5-19: mAP – mAR trend curves for the medium blocks in Bulga test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

Figure 5-19 highlights the trend for the mAP and mAR curves for the medium sized blocks. In this case, the delta between the two curves tends to increase, meaning that the numerosity of false negatives (the blocks that have not been correctly identified by the network) is proportionally larger to the increase in false positives.

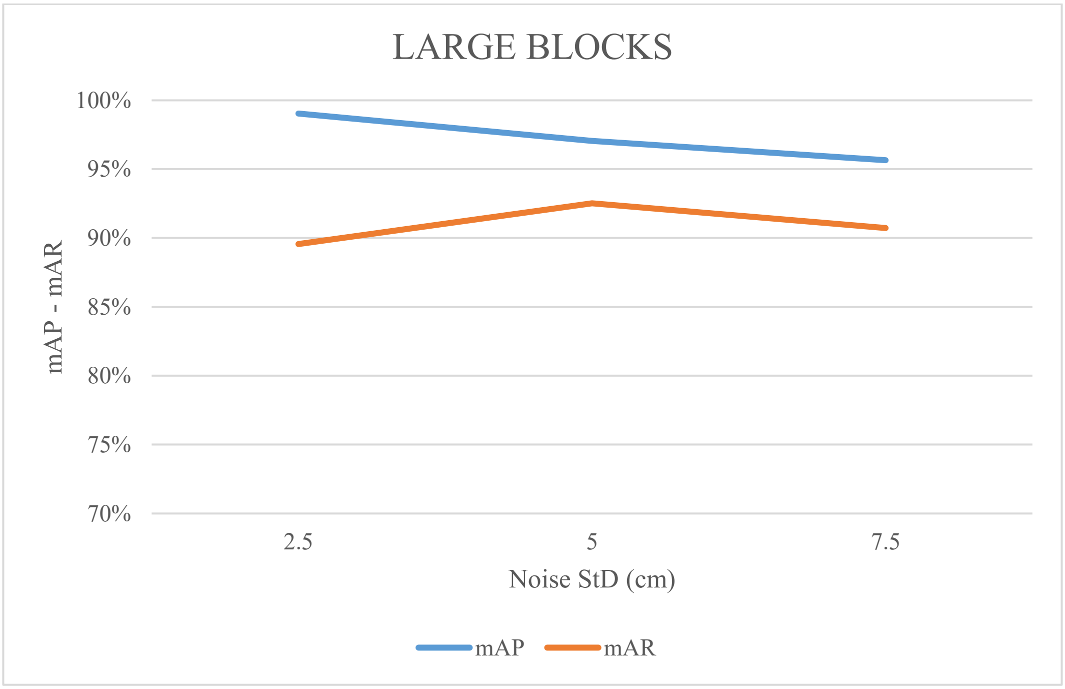


Figure 5-20: mAP – mAR trend curves for the large blocks in Bulga test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

In Figure 5-20, there is not a noticeably different pattern from the one observed in Figure 5-17. The higher dimensional class of blocks is therefore not influenced by measurement noise, both in terms of false positives and false negatives.

5.4.3 Test Site 3 – Pietra di Bismantova

Table 5-11 summarizes the results for the Pietra di Bismantova rock wall.

Table 5-11: Synthetic results for Pietra di Bismantova (Site 3) test site.

	DATASET 7	DATASET 8	DATASET 9
AP	26.79%	88.41%	99.23%
AR	32.90%	71.16%	90.53%
F1	29.53%	78.85%	94.68%
	DATASET 4	**DATASET 5**	**DATASET 6**
AP	62.10%	93.91%	99.62%
AR	60.03%	80.96%	93.62%
F1	61.05%	86.96%	96.53%
	DATASET 1	**DATASET 2**	**DATASET 3**
AP	90.17%	98.04%	99.81%
AR	78.75%	90.63%	94.89%
F1	84.07%	94.19%	97.29%

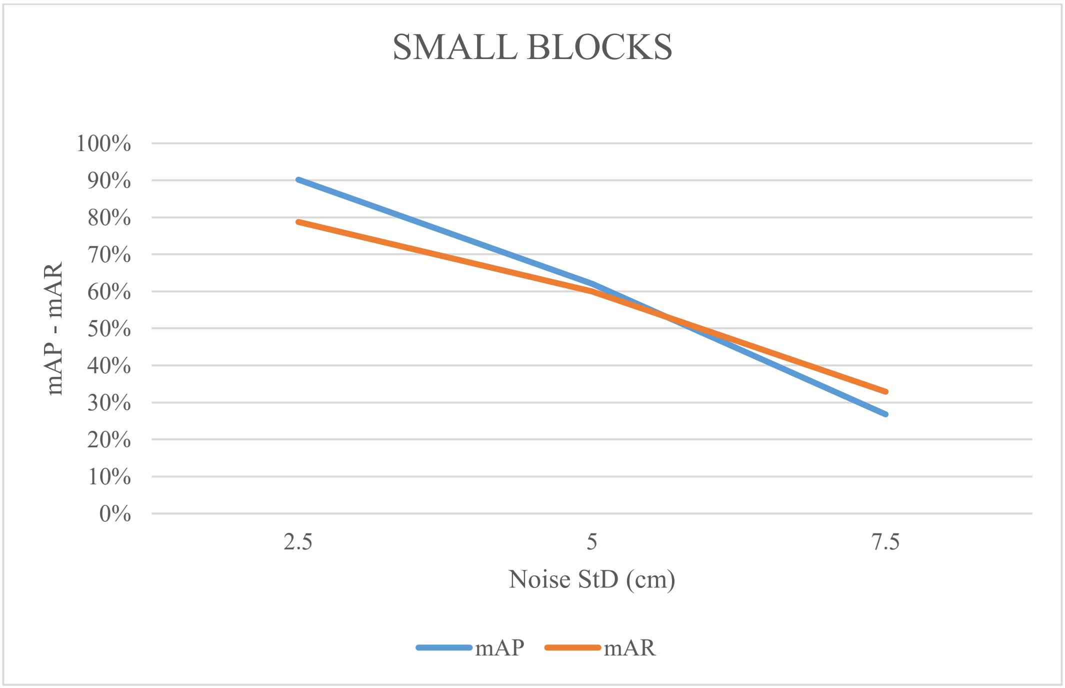


Figure 5-21: mAP – mAR trend curves for the small blocks in Pietra di Bismantova test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

In this wall, the most drastic decrease is observed for both the precision and recall curves. The precision decreases by more than 60% between noise level 1 and noise level 3 with a roughly linear trend. The recall curve, on the other hand, shows a less pronounced decrease in proportion, with a steeper decrease from level 2 to level 3 noise thresholds. In the simplest case, more than 20% of the simulated blocks are recognized by the network, while more than 60% of the blocks are not detected under large noise conditions. Thus, a significant influence of noise on the recognition of smaller blocks can be observed, possibly due to a more articulated initial wall conformation that, in combination with measurement noise, produces results that are not obviously aligned with previous cases. As the Pietra di Bismantova test site is an unknow dataset to the network, it can be observed that the precision is less resilient to the increase of measurement noise, in particular, when analyzing the same conditions in the Glendale and Bulga datasets, it can be observed that the decrease is largely more pronounced toward the 5 cm noise case. Variations in neural network performance in the context of the Pietra di Bismantova rock wall could be attributed to the complex morphology of the geological structure that might have resulted in noise spatial configurations with different characteristics compared to those of the model training phase, which may result in the increase of false positives.

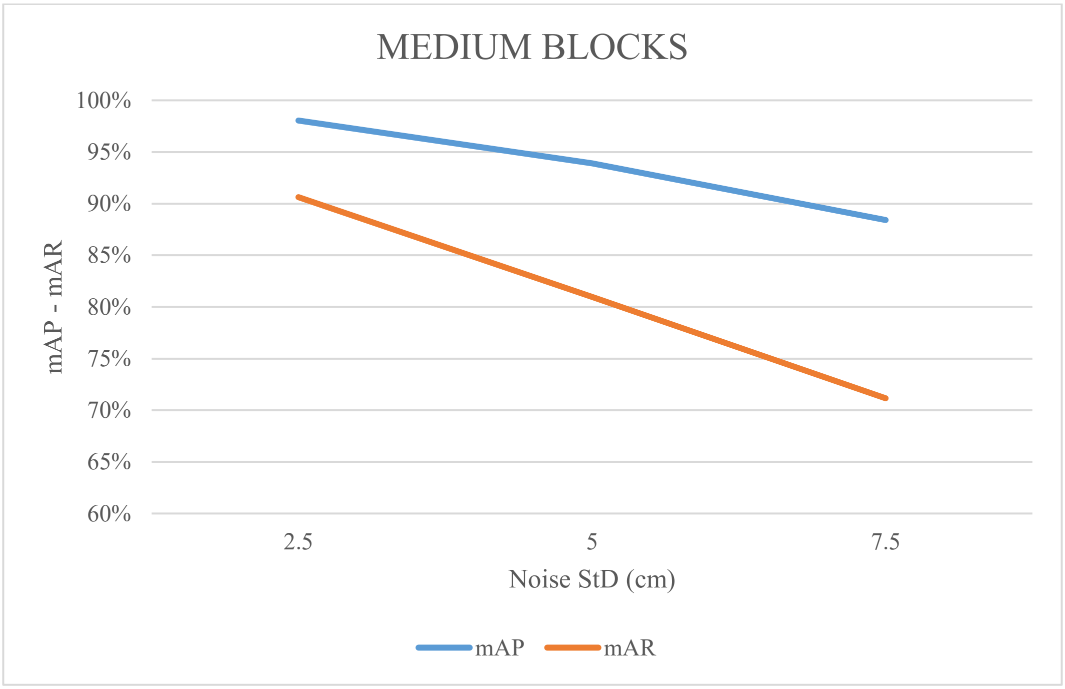


Figure 5-22: mAP – mAR trend curves for the medium blocks in Pietra di Bismantova test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

As for the medium-sized blocks, the decrease between noise level 1 and noise level 2 settles in about 10% on the precision curve and 20% on the recall curve between noise level 1 and noise level 3. In contrast to the small-sized datasets, the trend of the two curves follows a more concordant pattern to the other two tests, while still producing less accurate output overall.

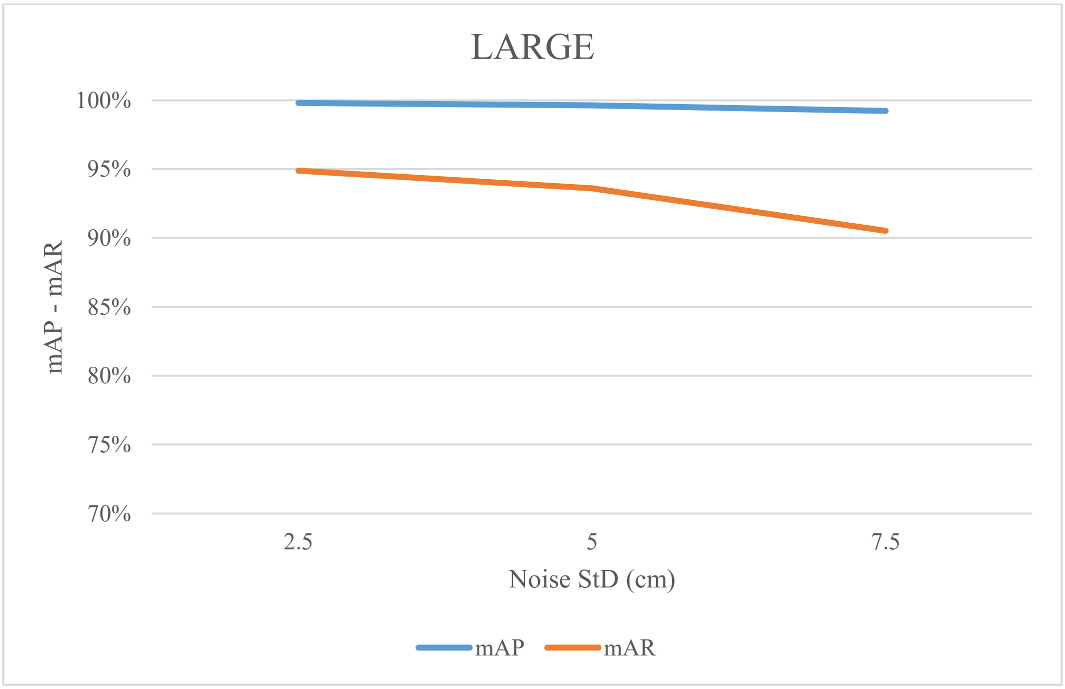


Figure 5-23: mAP – mAR trend curves for the large blocks in Pietra di Bismantova test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

For larger block sizes, as in the other previously observed cases, no significant influence of measurement noise on the identification of true positives and false positives is witnessed.

5.4.4 Test Site 4 - Pilkington

The last test was conducted on Pilkington (Site 4) rock wall.

Table 5-12: Synthetic results for Pilkington (Site 4) test site.

	DATASET 7	**DATASET 8**	**DATASET 9**
AP	35.32%	83.88%	98.85%
AR	46.41%	68.70%	89.12%
F1	40.11%	75.54%	93.73%
	DATASET 4	**DATASET 5**	**DATASET 6**
AP	58.31%	91.77%	99.44%
AR	59.71%	77.78%	90.72%
F1	59.00%	84.20%	94.88%
	DATASET 1	**DATASET 2**	**DATASET 3**
AP	86.71%	97.61%	99.72%
AR	79.76%	90.64%	90.86%
F1	83.09%	94.00%	95.09%

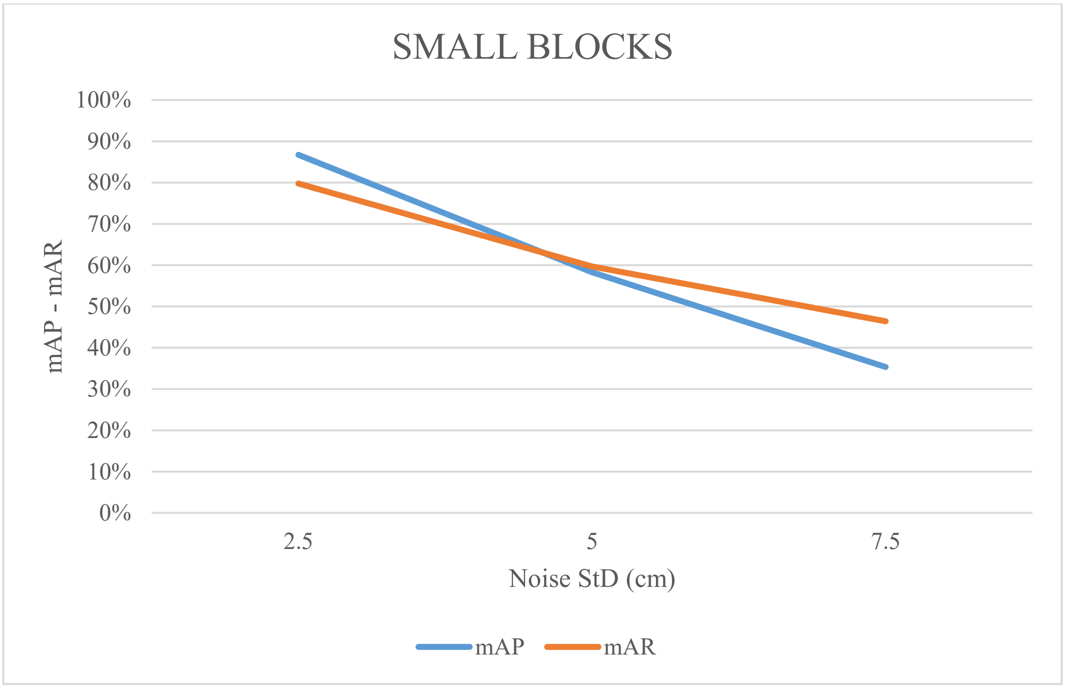


Figure 5-24: mAP – mAR trend curves for the small blocks in Pilkington test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

In Figure 5-24 there is evidence of a significant reduction in true positives straddling the different levels of measurement noise, with a total decrease between level 1 and level 3 of around 50% on the precision curve (indicating that tendentially half of the identified instances belong is a false positive) and about 45% on the recall curve, indicating that more than half of the simulated blocks are not identified. While producing comparable outputs to the Pietra di Bismantova, it must be noticed that the recall indices roughly decrease by 10% in this case, with comparable results to the test data produced with the same rock walls used in the training phase.

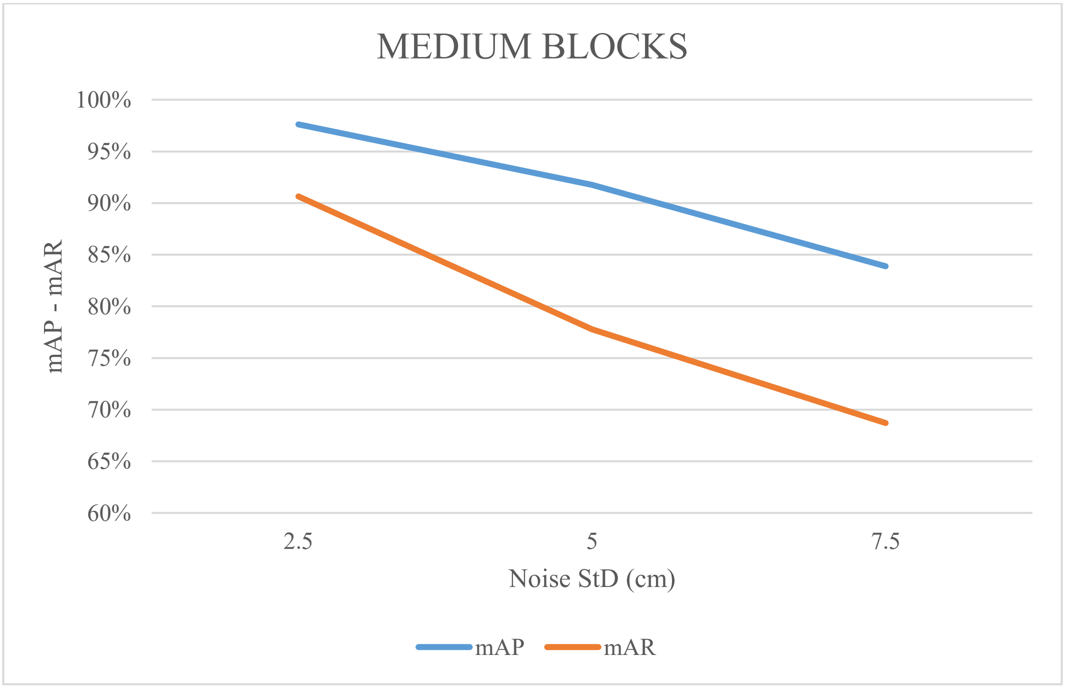


Figure 5-25: mAP – mAR trend curves for the medium blocks in Pilkington test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

The curve of medium-sized blocks tended to follow the pattern observed in test site 3, with a downward trend for both the precision and recall curves.

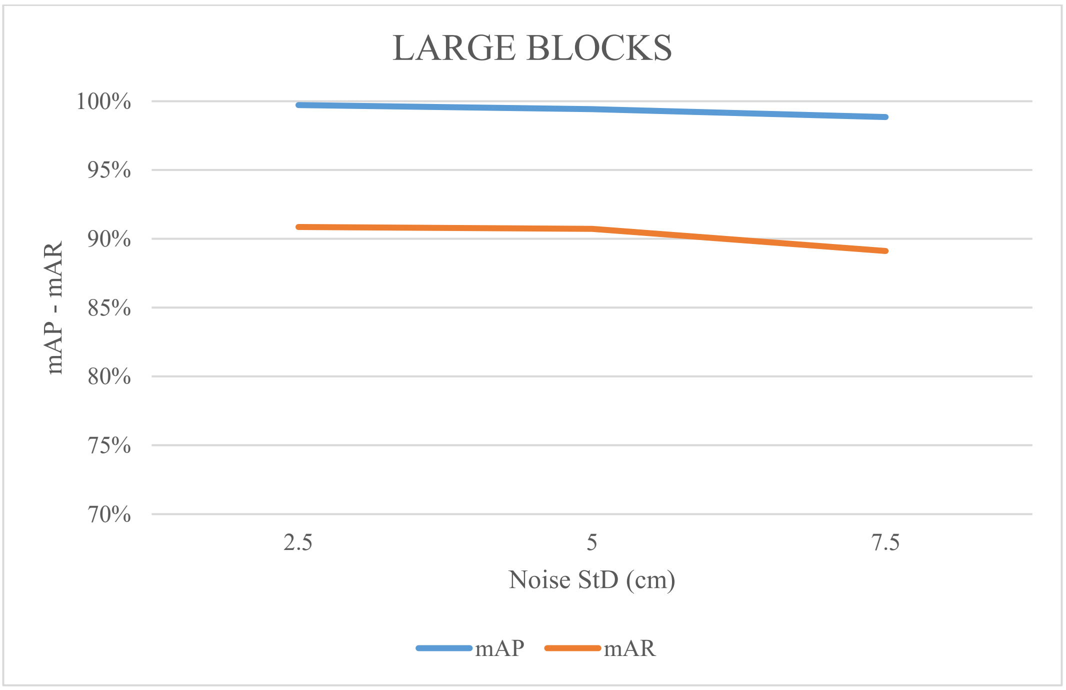


Figure 5-26: mAP – mAR trend curves for the large blocks in Pilkington test site, the horizontal axis represents the increasing measurement noise levels, the vertical axis summarizes the mAP (blue curve) and mAR (orange curve) in percentage.

Lastly, the largest size class is less tangibly affected by the effects of measurement noise. As with the other tests, the recall curve has a substantially constant delta from the precision curve, an indication that the network does not tend to produce false positives belonging to this class.

5.5 Concluding remarks

This chapter has been dedicated to the evaluation of the performance of the Mask R-CNN neural network in automatically detecting rockfall events in the context of open-pit mines. The study was structurally designed to address two main research questions:

1. The impact of the size of rockfall events on the neural network's detection capabilities.
2. The role of measurement noise in the identification of rockfall events.

The neural network's performance was evaluated using precision and recall curves. Three distinct classes of rock sizes—small, medium, and large—were analyzed under multiple noise conditions. A comparative analysis was also carried out using different reference models, namely the Pietra di Bismantova, Pilkington, Glendale, and Bulga datasets.
The results yielded critical insights into the limitations and capabilities of the Mask R-CNN model in this application. For small-sized rockfall events, the impact of measurement noise was found to be significant, leading to a substantial decrease in both precision and recall metrics. This was exacerbated by the complex geological structure and wall conformation of the Pietra di Bismantova site, which resulted in the worst performing scenario for the neural network. Medium-sized blocks displayed a somewhat different pattern. While still affected by noise, they exhibited a more concordant trend between precision and recall curves, albeit with less accurate output overall compared to other sizes.
Larger rockfall events demonstrated a resilience to measurement noise, with no significant impact on false positives or false negatives. As can be easily expected, the neural network is inherently more reliable in identifying larger block rockfall, irrespective of noise conditions.

6 CHAPTER 6 – GENERAL CONCLUSIONS AND RESEARCH PERSPECTIVES

This concluding section provides a summary of the outcomes and implications of the research, with a particular focus on the simulation methodology of the rockfalls.

The core of the thesis was divided into two primary strands: the structuring of photogrammetric process simulation software for generating synthetic rockfall datasets on rock walls, and the analysis of the Mask R-CNN neural network model's performance in the automatic detection of rockfall events.

One of the most significant methodological innovations of this thesis has been the use of synthetic data generated by simulation software, as presented in Chapter 3. This approach is novel in the field of rock wall monitoring and finds no similar examples in existing literature. The use of simulation software allowed for a controlled assessment of the neural network model's performance, considering various levels of measurement noise and different size classes of rockfall events.

The use of neural networks for the rockfall classification only finds one example in literature, as the research in [29] was carried using LiDAR derived data instead of photogrammetric derived data. The research focused on the definition of an extensive and representative rockfall detection database for point cloud classification purposes, which necessitated several years to be completed, and on the application of a deep neural model to proceed to automatic identification of such events. In this sense, such research appears to yield superior results in terms of classification accuracy; however, it is crucial to contextualize this observation. Notably, their work does not offer an extensive breakdown of results across different rockfall size classes, while this work of thesis focuses on such aspects. Given the importance of size classification in rockfall event assessment for risk assessment in prevention modeling ,this thesis adds a substantive layer of specificity in the analysis that is absent in the existing literature. For this purpose, it would be desirable to conduct a comparison using a common benchmark dataset, applying both LiDAR and photogrammetric techniques, to tease apart the intricacies and inherent biases of each approach.

Regarding the results obtained, a significant variation was observed in the neural network's performance depending on the size of the rockfall events and the level of measurement noise. While larger events showed resilience to noise, smaller events were significantly influenced, especially when subjected to higher levels of measurement noise.
Moreover, the testing over different rockfaces that those used for training and validations showed significant differences in performance, which should be further investigated. This finding underlines the importance of diversifying training datasets to include a range of geological formations and noise conditions for more robust generalizability.
It must be noted that this research has been primarily tested in a simulated environment. While the application of synthetic data does offer a controlled setting for performance evaluation, the real-world applicability of the neural network model has yet to be empirically tested. Future research should incorporate the evaluation of your model against manually classified, real-world datasets. This would offer insights into the model's generalizability across various geological formations and under diverse environmental conditions, thus contributing to a more comprehensive validation of this research.
In conclusion, the limits in performance of the neural network when noise and block size become comparable were expected. Improving this performance might require adding new information sources to the network. This could be by integration of digital images with geometric data in the neural network training process. The task will not be easy for the very same reasons that motivated the geometric approach of this work. Though high-resolution digital images can provide additional information such as texture, color, and variations in luminosity, they are often insufficient for accurate classification of these events and the training phase can be anticipated to be quite complex. These attributes could be used to improve the classification phase of the model, making it more sensitive to subtle variations that could be indicative of rockfall events. Most of all, adding image content to the network would render the simulation tool in its current form useless, resorting to building the training dataset from real data.
The fusion of geometric data with digital images may necessitate an architectural modification of the neural network to effectively accommodate and process the heterogeneous data.

This integrated approach could have broad applicability, extending the robustness and accuracy of the neural network not just for rockfall event detection but also for other geospatial and environmental applications.

7 BIBLIOGRAPHY

[1] S. Ullman, “The Interpretation of Structure from Motion,” *Proceedings of the Royal Society of London*, vol. 203, no. 1153, pp. 405–426, 1979, [Online]. Available: http://www.jstor.org/stable/77505

[2] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second. Cambridge University Press, 2004.

[3] J. J. Koenderink and A. J. van Doorn, “Affine structure from motion,” *Journal of the Optical Society of America A*, vol. 8, no. 2, p. 377, 1991, doi: 10.1364/josaa.8.000377.

[4] M. J. Westoby, J. Brasington, N. F. Glasser, M. J. Hambrey, and J. M. Reynolds, “‘Structure-from-Motion’ photogrammetry: A low-cost, effective tool for geoscience applications,” *Geomorphology*, vol. 179, pp. 300–314, 2012, doi: 10.1016/j.geomorph.2012.08.021.

[5] C. S. Fraser and S. Cronk, “A hybrid measurement approach for close-range photogrammetry,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 3, pp. 328–333, 2009, doi: 10.1016/j.isprsjprs.2008.09.009.

[6] S. N. Lane, T. D. James, and M. D. Crowell, “Application of digital photogrammetry to complex topography for geomorphological research,” *Photogrammetric Record*, vol. 16, no. 95, pp. 793–821, 2000, doi: 10.1111/0031-868X.00152.

[7] F. Remondino and C. Fraser, “Digital Camera Calibration Methods: Considerations and Assumptions,” *Symposium “Image Engineering and Vision Metrology,”* vol. 36, no. 5, pp. 266–272, 2006

[8] N. Micheletti, J. H. Chandler, and S. N. Lane, “Investigating the geomorphological potential of freely available and accessible structure-from-motion photogrammetry using a smartphone,” vol. 486, no. October 2014, pp. 473–486, 2015, doi: 10.1002/esp.3648.

[9] M. Scaioni, L. Longoni, V. Melillo, and M. Papini, “Remote Sensing for Landslide Investigations: An Overview of Recent Achievements and Perspectives,” *Remote Sens (Basel)*, pp. 1–53, 2014, doi: 10.3390/rs60x000x.

[10] T. R. Reid and J. P. Harrison, "A semi-automated methodology for discontinuity trace detection in digital images of rock mass exposures," vol. 37, pp. 1073–1089, 2000.

[11] F. Agliardi and G. B. Crosta, "High resolution three-dimensional numerical modelling of rockfalls," vol. 40, pp. 455–471, 2003, doi: 10.1016/S1365-1609(03)00021-2.

[12] G. Costantini, C. Comina, A. M. Ferrero, G. Umili, and S. Bonetto, "Applicazione in sotterraneo di tecniche fotografiche e GPR per il rilievo di discontinuità," no. April, 2023.

[13] A. M. Ritchie, "Evaluation of Rockfall and Its Control," *In Highway Research Record 17, Stability of Rock Slopes, Highway Research Board, National Research Council, Washington, D.C.*, pp. 13–28, 1963.

[14] J. Briones-Bitar, P. Carrión-Mero, N. Montalván-Burbano, and F. Morante-Carballo, "Rockfall research: A bibliometric analysis and future trends," *Geosciences (Basel)*, vol. 10, no. 10, pp. 1–25, 2020, doi: 10.3390/geosciences10100403.

[15] P. Yang, Y. Shang, Y. Li, H. Wang, and K. Li, "Analysis of potential rockfalls on a highway at high slopes in cold-arid areas (Northwest Xinjiang, China)," *Sustainability (Switzerland)*, vol. 9, no. 3, 2017, doi: 10.3390/su9030414.

[16] J. Briones-Bitar, P. Carrión-Mero, N. Montalván-Burbano, and F. Morante-Carballo, "Rockfall Research: A Bibliometric Analysis and Future Trends," *Geosciences (Basel)*, vol. 10, no. 10, p. 403, Oct. 2020, doi: 10.3390/geosciences10100403.

[17] S. Bhandari, "Engineering Rock Blasting Operations." Wordpress, 2009. [Online]. Available: https://miningandblasting.files.wordpress.com/2009/09/engineering-rock-blasting-operations_bhandari.pdf

[18] D. J. Armaghani, A. Mahdiyar, M. Hasanipanah, R. S. Faradonbeh, M. Khandelwal, and H. B. Amnieh, "Risk Assessment and Prediction of Flyrock Distance by Combined Multiple Regression Analysis and Monte Carlo Simulation of Quarry Blasting," *Rock Mech Rock Eng*, vol. 49, no. 9, pp. 3631–3641, 2016, doi: 10.1007/s00603-016-1015-z.

[19] A. M. Ferrero, G. Forlani, R. Roncella, and H. I. Voyat, “Advanced geostructural survey methods applied to rock mass characterization,” *Rock Mech Rock Eng*, vol. 42, no. 4, pp. 631–665, 2009, doi: 10.1007/s00603-008-0010-4.

[20] C. McAnuff, C. Samson, D. Melanson, C. Polowick, and E. Bethell, “Structural mapping of rock walls imaged with a lidar mounted on an unmanned aircraft system,” *J Unmanned Veh Syst*, vol. 7, no. 1, pp. 21–38, 2019, doi: 10.1139/juvs-2018-0015.

[21] F. Buill, M. A. Núñez-Andrés, N. Lantada, and A. Prades, “Comparison of Photogrammetric Techniques for Rockfalls Monitoring,” *IOP Conf Ser Earth Environ Sci*, vol. 44, no. 4, 2016, doi: 10.1088/1755-1315/44/4/042023.

[22] A. Giacomini *et al.*, “Temporal-spatial frequency rockfall data from open-pit highwalls using a low-cost monitoring system,” *Remote Sens (Basel)*, vol. 12, no. 15, 2020, doi: 10.3390/RS12152459.

[23] Ryan, Cooper, and Tauer, “Geometrical Calibration of Stereo Images in Convergent Camera Arrangement,” *Paper Knowledge . Toward a Media History of Documents*, pp. 12–26, 2013.

[24] R. Kromer, B. Walton, B. Gray, M. Lato, and R. Group, “Development and Optimization of an Automated Fixed-Location Time Lapse Photogrammetric Rock Slope Monitoring System,” *Remote Sens (Basel)*, 2019, doi: doi:10.3390/rs11161890.

[25] M. Santise, K. Thoeni, R. Roncella, S. W. Sloan, and A. Giacomini, “Preliminary tests of a new low-cost photogrammetric system,” *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 42, no. 2W8. pp. 229–236, 2017. doi: 10.5194/isprs-archives-XLII-2-W8-229-2017.

[26] M. Scaioni, R. Roncella, and M. I. Alba, “Change detection and deformation analysis in point clouds: Application to rock face monitoring,” *Photogramm Eng Remote Sensing*, vol. 79, no. 5, pp. 441–455, 2013, doi: 10.14358/PERS.79.5.441.

[27] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[28] H. Chen, C. Wu, B. Du, L. Zhang, and L. Wang, “Change Detection in Multisource VHR Images via Deep Siamese Convolutional Multiple-Layers Recurrent Neural

Network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 4, pp. 2848–2864, 2020, doi: 10.1109/TGRS.2019.2956756.

[29] I. Farmakis, P. M. DiFrancesco, D. J. Hutchinson, and N. Vlachopoulos, "Rockfall detection using LiDAR and deep learning," *Eng Geol*, vol. 309, no. March, p. 106836, 2022, doi: 10.1016/j.enggeo.2022.106836.

[30] M. Duquesnoy, C. Liu, D. Z. Dominguez, V. Kumar, E. Ayerbe, and A. A. Franco, "Machine learning-assisted multi-objective optimization of battery manufacturing from synthetic data generated by physics-based simulations," *Energy Storage Mater*, vol. 56, no. September 2022, pp. 50–61, 2023, doi: 10.1016/j.ensm.2022.12.040.

[31] J. Yeomans, S. Thwaites, W. S. P. Robertson, D. Booth, B. Ng, and D. Thewlis, "Simulating Time-Series Data for Improved Deep Neural Network Performance," *IEEE Access*, vol. 7, pp. 131248–131255, 2019, doi: 10.1109/ACCESS.2019.2940701.

[32] L. Khelifi and M. Mignotte, "Deep Learning for Change Detection in Remote Sensing Images: Comprehensive Review and Meta-Analysis," *IEEE Access*, vol. 8, no. Cd, pp. 126385–126400, 2020, doi: 10.1109/ACCESS.2020.3008036.

[33] S. Niu and V. Srivastava, "Simulation trained CNN for accurate embedded crack length, location, and orientation prediction from ultrasound measurements," *Int J Solids Struct*, vol. 242, no. February, p. 111521, 2022, doi: 10.1016/j.ijsolstr.2022.111521.

[34] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazırbas, and V. Golkov, "FlowNet : Learning Optical Flow with Convolutional Networks," *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

[35] M. Danielczuk *et al.*, "Segmenting Unknown 3D Objects from Real Depth Images using Mask R-CNN Trained on Synthetic Data," *2019 International Conference on Robotics and Automation*, pp. 7283–7290, Sep. 2018, [Online]. Available: https://bit.ly/2letCuE.

[36] S. Gavazzi *et al.*, "Deep learning-based reconstruction of in vivo pelvis conductivity with a 3D patch-based convolutional neural network trained on simulated MR data," *Magn Reson Med*, vol. 84, no. 5, pp. 2772–2787, Nov. 2020, doi: 10.1002/mrm.28285.

[37] B. Žabota and M. Kobal, “A new methodology for mapping past rockfall events: From mobile crowdsourcing to rockfall simulation validation,” *ISPRS Int J Geoinf*, vol. 9, no. 9, 2020, doi: 10.3390/ijgi9090514.

[38] I. Goodfellow, “Generative Adversarial Networks,” pp. 1–9, Dec. 2016, [Online]. Available: http://arxiv.org/abs/1701.00160

[39] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” no. Ml, pp. 1–14, Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.6114

[40] V. A. Mizginov, V. V. Kniaz, and N. A. Fomin, “a Method for Synthesizing Thermal Images Using Gan Multi-Layered Approach,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIV-2/W1-, no. April, pp. 155–162, 2021, doi: 10.5194/isprs-archives-xliv-2-w1-2021-155-2021.

[41] F. Remondino and S. El-Hakim, “Image-based 3D modelling: A review,” *The Photogrammetric Record*, vol. 21, no. September, pp. 269–291, 2006.

[42] J. Baqersad, P. Poozesh, C. Niezrecki, and P. Avitabile, “Photogrammetry and optical methods in structural dynamics – A review,” *Mech Syst Signal Process*, vol. 86, pp. 17–34, 2017, doi: 10.1016/j.ymssp.2016.02.011.

[43] B. P. W. Wolfgang Förstner, *Photogrammetric Computer Vision*. Springer Cham, 2016. doi: https://doi.org/10.1007/978-3-319-11550-4.

[44] H. Adeli and S.-L. Hung, *Machine Learning: Neural Networks, Genetic Algorithms, and Fuzzy Systems*, 1st ed. John Wiley & Sons, Inc., 1995.

[45] B. T. Ong, K. Sugiura, and K. Zettsu, “Dynamically pre-trained deep recurrent neural networks using environmental monitoring data for predicting PM2.5,” *Neural Comput Appl*, vol. 27, no. 6, pp. 1553–1566, 2016, doi: 10.1007/s00521-015-1955-3.

[46] C. Chen and H. Mcnairn, “A neural network integrated approach for rice crop monitoring,” vol. 1161, pp. 1366–1393, 2007, doi: 10.1080/01431160500421507.

[47] G. Baxt, “Application of artificial neural networks to clinical medicine,” pp. 1135–1138, 1995.

[48] A. S. Chen, M. T. Leung, and H. Daouk, “Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index,”

Comput Oper Res, vol. 30, no. 6, pp. 901–923, 2003, doi: 10.1016/S0305-0548(02)00037-0.

[49] L. Alzubaidi *et al.*, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, vol. 8, no. 1. Springer International Publishing, 2021. doi: 10.1186/s40537-021-00444-8.

[50] R. J. Hyndman and G. Athanasopoulos, *Forecasting : Principles and Practice*. OTexts, 2018.

[51] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, NY, 2006.

[52] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. pringer Science+Business Media, LLC, part of Springer Nature 2009, 2009. doi: https://doi.org/10.1007/978-0-387-84858-7.

[53] C. Pramerdorfer and M. Kampel, "Facial Expression Recognition using Convolutional Neural Networks: State of the Art," *Lecture Notes in Networks and Systems*, vol. 339, pp. 605–617, Dec. 2016, [Online]. Available: https://link.springer.com/10.1007/978-981-16-7018-3_45

[54] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Sci Robot*, vol. 4, no. 28, Mar. 2019, doi: 10.1126/scirobotics.aaw1975.

[55] S. Saito, T. Yamashita, and Y. Aoki, "Multiple Object Extraction from Aerial Imagery with Convolutional Neural Networks," *Electronic Imaging*, vol. 28, no. 10, pp. 1–9, Feb. 2016, doi: 10.2352/ISSN.2470-1173.2016.10.ROBVIS-392.

[56] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12346 LNCS, pp. 405–421, 2020, doi: 10.1007/978-3-030-58452-8_24.

[57] A. Karami, S. Rigon, G. Mazzacca, Z. Yan, and F. Remondino, "NERFBK: A High-Quality Benchmark for NERF-Based 3D Reconstruction," Jun. 2023, [Online]. Available: http://arxiv.org/abs/2306.06300

[58] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol Rev*, vol. 65, no. 6, pp. 386–408, 1958.

[59] B. Widrow and M. E. Hoff, “Adaptive switching circuits,” 1960.

[60] M. L. Minsky and S. A. Papert, *Perceptrons*. 1969.

[61] D. S. Broomhead and D. Lowe, “Radial basis functions, multi-variable functional interpolation and adaptive networks,” *Royal Signals and Radar Establishment*, vol. Memorandum, no. 4148, 1988.

[62] M. J. D. Powell, “Radial Basis Functions Approximations to Polynomials,” *Proceedings 12th Biennial Numerical Analysis Conference*, 1987.

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986, doi: 10.1038/323533a0.

[64] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities.,” *Proc Natl Acad Sci U S A*, vol. 79, no. 8, pp. 2554–2558, 1982, doi: 10.1073/pnas.79.8.2554.

[65] A. Graves, “Generating Sequences With Recurrent Neural Networks,” pp. 1–43, 2013, [Online]. Available: http://arxiv.org/abs/1308.0850

[66] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems*, vol. 6, no. 2, pp. 107–116, 1998, doi: 10.1142/S0218488598000094.

[67] G. E. Hinton and V. Nair, “Rectified Linear Units Improve Restricted Boltzmann Machines,” *Proceedings of the 27th International Confer- ence on Machine Learning*, no. 3, 2010.

[68] A. Krogh, “A Simple Weight Decay Can Improve,” pp. 950–957, 1991.

[69] Y. LeCun *et al.*, “Backpropagation Applied to Handwritten Zip Code Recognition,” *Neural Comput*, vol. 1, no. 4, pp. 541–551, Dec. 1989, doi: 10.1162/neco.1989.1.4.541.

[70] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[71] T. Joachims, “Making Large-Scale SVM Learning Practical,” no. October 1999, 1998, doi: 10.17877/DE290R-5097.

[72] K. J. Kim, “Financial time series forecasting using support vector machines,” *Neurocomputing*, vol. 55, no. 1–2, pp. 307–319, 2003, doi: 10.1016/S0925-2312(03)00372-2.

[73] L. Breiman, “Random forests,” *Mach Learn*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[74] R. Díaz-Uriarte and S. Alvarez de Andrés, “Gene selection and classification of microarray data using random forest,” *BMC Bioinformatics*, vol. 7, pp. 1–13, 2006, doi: 10.1186/1471-2105-7-3.

[75] X. Chen and H. Ishwaran, “Random forests for genomic data analysis,” *Genomics*, vol. 99, no. 6, pp. 323–329, 2012, doi: 10.1016/j.ygeno.2012.04.003.

[76] V. F. Rodriguez-galiano, B. Ghimire, J. Rogan, M. Chica-olmo, and J. P. Rigol-sanchez, “An assessment of the effectiveness of a random forest classifier for land-cover classification,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 67, pp. 93–104, 2012, doi: 10.1016/j.isprsjprs.2011.11.002.

[77] M. Belgiu and L. Dra, “Random forest in remote sensing : A review of applications and future directions,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, 2016, doi: 10.1016/j.isprsjprs.2016.01.011.

[78] P. Du, A. Samat, B. Waske, S. Liu, and Z. Li, “Random Forest and Rotation Forest for fully polarized SAR image classification using polarimetric and spatial features,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 38–53, 2015, doi: 10.1016/j.isprsjprs.2015.03.002.

[79] A. Krizhevsky and G. Hinton, “Convolutional deep belief networks on cifar-10,” *Unpublished manuscript*, pp. 1–9, 2010, [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Convolutional+Deep+Belief+Networks+on+CIFAR-10#0

[80] I. Buck, “GPU computing with NVIDIA CUDA,” *ACM SIGGRAPH 2007 Papers - International Conference on Computer Graphics and Interactive Techniques*, 2007, doi: 10.1145/1281500.1281647.

[81] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun ACM*, 2012, doi: 10.1145/3065386.

[82] N. Murray and F. Perronnin, “Generalized max pooling,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2473–2480, 2014, doi: 10.1109/CVPR.2014.317.

[83] M. Lin, Q. Chen, and S. Yan, “Network In Network,” *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pp. 1–10, Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.4400

[84] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[85] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.

[86] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, “Visualizing higher-layer features of a deep network,” *Bernoulli*, no. 1341, pp. 1–13, 2009, [Online]. Available: http://igva2012.wikispaces.asu.edu/file/view/Erhan+2009+Visualizing+higher+layer+features+of+a+deep+network.pdf

[87] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.1556

[88] C. Szegedy *et al.*, “Going deeper with convolutions,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.

[89] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[90] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Region-Based Convolutional Networks for Accurate Object Detection and Segmentation,” *IEEE Trans Pattern*

Anal Mach Intell, vol. 38, no. 1, pp. 142–158, 2016, doi: 10.1109/TPAMI.2015.2437384.

[91] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for object recognition,” *Int J Comput Vis*, vol. 104, no. 2, pp. 154–171, 2013, doi: 10.1007/s11263-013-0620-5.

[92] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[93] E. Shelhamer, J. Long, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 4, pp. 640–651, 2017, doi: 10.1109/TPAMI.2016.2572683.

[94] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” May 2015, [Online]. Available: http://arxiv.org/abs/1505.04597

[95] P. O. Pinheiro, R. Collobert, and P. Dollar, “Learning to segment object candidates,” *Adv Neural Inf Process Syst*, vol. 2015-Janua, pp. 1990–1998, 2015.

[96] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Trans Pattern Anal Mach Intell*, vol. 42, no. 2, pp. 386–397, 2020, doi: 10.1109/TPAMI.2018.2844175.

[97] M. Cordts *et al.*, “The Cityscapes Dataset,” *CVPR Workshop on The Future of Datasets in Vision*, 2015.

[98] A. Savkin, T. Lapotre, K. Strauss, U. Akbar, and F. Tombari, “Adversarial Appearance Learning in Augmented Cityscapes for Pedestrian Recognition in Autonomous Driving,” *Proc IEEE Int Conf Robot Autom*, pp. 3305–3311, 2020, doi: 10.1109/ICRA40945.2020.9197024.

[99] M. T. Hagan, H. Demut, M. Beale, and O. De Jesus, *Neural Network Design*, 2nd ed. 1996. [Online]. Available: hagan.okstate.edu/nnd.html

[100] P. J. Werbos, “Backpropagation Through Time : What It Does and How to Do It,” vol. 78, no. October, pp. 1550–1560, 1990.

[101] A. Jadon, A. Patil, and S. Jadon, “A Comprehensive Survey of Regression Based Loss Functions for Time Series Forecasting,” 2022, [Online]. Available: http://arxiv.org/abs/2211.02989

[102] A. Mao, M. Mohri, and Y. Zhong, “Cross-Entropy Loss Functions: Theoretical Analysis and Applications,” 2023, [Online]. Available: http://arxiv.org/abs/2304.07288

[103] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2015.

[104] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training Recurrent Neural Networks,” *Phylogenetic Diversity: Applications and Challenges in Biodiversity Science*, no. 2, pp. 41–71, Nov. 2012, doi: 10.1007/978-3-319-93145-6_3.

[105] L. N. Smith, “Cyclical learning rates for training neural networks,” *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, no. April, pp. 464–472, 2017, doi: 10.1109/WACV.2017.58.

[106] S. Sharma, S. Sharma, and A. Anidhya, “Activation functions in neural networks,” *International Journal of Engineering Applied Sciences and Technology*, vol. 4, no. 12, pp. 310–316, 2020.

[107] T. Szandała, “Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks,” 2018.

[108] S. Hayou, A. Doucet, and J. Rousseau, “On the impact of the activation function on deep neural networks training,” *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 4746–4754, 2019.

[109] D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *J Physiol*, vol. 160, no. 1, pp. 106–154, Jan. 1962, doi: 10.1113/jphysiol.1962.sp006837.

[110] D. Scherer, A. Müller, and S. Behnke, “Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6354, no. PART 3, K. Diamantaras, W. Duch, and L.

S. Iliadis, Eds., in Lecture Notes in Computer Science, vol. 6354. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–101. doi: 10.1007/978-3-642-15825-4_10.

[111] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980

[112] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” Aug. 2014, [Online]. Available: http://arxiv.org/abs/1408.5882

[113] F. Perez, C. Vasconcelos, S. Avila, and E. Valle, “Data Augmentation for Skin Lesion Analysis,” Sep. 2018, doi: 10.1007/978-3-030-01201-4_33.

[114] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” Nov. 2014, doi: 10.1016/j.cpc.2014.11.006.

[115] Shun-ichi Amari, “Backpropagation and stochastic gradient descent method,” *Neurocomputing*, vol. 5, no. 4–5. pp. 185–196, 1993.

[116] T. Dietterich, “Overfitting and Undercomputing in Machine Learning,” *ACM Computing Surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995, doi: 10.1145/212094.212114.

[117] C. Cortes, G. Research, and N. York, “L 2 Regularization for Learning Kernels,” *Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 109–116, 2004.

[118] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12346 LNCS, pp. 213–229, 2020, doi: 10.1007/978-3-030-58452-8_13.

[119] A. Bonhage, M. Eltaher, T. Raab, M. Breuß, A. Raab, and A. Schneider, “A modified Mask region-based convolutional neural network approach for the automated detection of archaeological sites on high-resolution light detection and ranging-derived digital elevation models in the North German Lowland,” *Archaeol Prospect*, vol. 28, no. 2, pp. 177–186, Apr. 2021, doi: 10.1002/arp.1806.

[120] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and*

Lecture Notes in Bioinformatics), vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014, doi: 10.1007/978-3-319-10602-1_48.

[121] W. C. Kao, M. C. Hsu, and Y. Y. Yang, "Local contrast enhancement and adaptive feature extraction for illumination-invariant face recognition," *Pattern Recognit*, vol. 43, no. 5, pp. 1736–1747, 2010, doi: 10.1016/j.patcog.2009.11.016.

[122] K. Kraus, *Photogrammetry: Geometry from Images and Laser Scans*, 1st ed. 2007. doi: https://doi.org/10.1515/9783110892871.

[123] I. Ernst and H. Hirschmü, "Mutual information based semi-global stereo matching on the GPU," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5358 LNCS, no. PART 1, pp. 228–239, 2008, doi: 10.1007/978-3-540-89639-5_22.

[124] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans Pattern Anal Mach Intell*, vol. 30, no. 2, pp. 328–341, 2008, doi: 10.1109/TPAMI.2007.1166.

[125] A. W. Gruen, "Adaptive least squares correlation: a powerful image matching technique," *South African Journal of Photogrammetry, Remote Sensing and Cartography*, vol. 14, no. 3, pp. 175–187, 1985, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary;jsessionid=8478C7133B070BA354E5DED2EFF60262?doi=10.1.1.93.6891%5Cnhttp://www.idb.arch.ethz.ch/files/alsm_awgruen.pdf

[126] D. Rosenholm, "LEAST SQUARES MATCHING METHOD: SOME EXPERIMENTAL RESULTS," *The Photogrammetric Record*, vol. 12, no. 70, pp. 493–512, Aug. 2006, doi: 10.1111/j.1477-9730.1987.tb00598.x.

[127] V. C. Klema and A. J. Laub, "The Singular Value Decomposition: Its Computation and Some Applications," *IEEE Trans Automat Contr*, vol. 25, no. 2, pp. 164–176, 1980, doi: 10.1109/TAC.1980.1102314.

[128] Agisoft, "Agisoft Metashape." 2023. [Online]. Available: http://www.agisoft.com/downloads/installer/

[129] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Proceedings - IEEE Workshop on Stereo*

and Multi-Baseline Vision, SMBV 2001, no. 1, pp. 131–140, 2001, doi: 10.1109/SMBV.2001.988771.

[130] P. J. Besl and N. D. McKay, “A method for registration of 3-D shapes,” *IEEE Trans Pattern Anal Mach Intell*, vol. 14, no. 2, pp. 239–256, Feb. 1992, doi: 10.1109/34.121791.

[131] D. Salomon, *Data Compression The Complete Reference FourthEdition*, 4th ed., vol. 53, no. 9. Springer, 2007.

[132] J. M. Monte, “Rock mass characterization using laser scanning and digital imaging data collection techniques,” *IEEE Micro*, vol. 18, no. 6, pp. 12–13, 2004.

[133] P. R. Wolf, B. A. Dewitt, and B. E. Wilkinson, *Elements of photogrammetry with application in GIS*, vol. Fourth Edi. McGraw Hill Education, 2014.

[134] R. A. Kromer *et al.*, “Automated terrestrial laser scanning with near-real-time change detection - Monitoring of the Séchilienne landslide,” *Earth Surface Dynamics*, vol. 5, no. 2, pp. 293–310, 2017, doi: 10.5194/esurf-5-293-2017.

[135] M. J. Royán, A. Abellán, M. Jaboyedoff, J. M. Vilaplana, and J. Calvet, “Spatio-temporal analysis of rockfall pre-failure deformation using Terrestrial LiDAR,” *Landslides*, vol. 11, no. 4, pp. 697–709, 2014, doi: 10.1007/s10346-013-0442-0.

[136] M. Scaioni, R. Roncella, and M. Alba, “Change Detection and Deformation Analysis in Point Clouds: Application to Rock Face Monitoring,” *Photogramm Eng Remote Sensing*, vol. 79, no. 5, pp. 441–455, 2013.

[137] Y. Chen, G. Liu, Y. Xu, P. Pan, and Y. Xing, “PointNet++ network architecture with individual point level and global features on centroid for als point cloud classification,” *Remote Sens (Basel)*, vol. 13, no. 3, pp. 1–17, 2021, doi: 10.3390/rs13030472.

[138] R. Roncella and G. Forlani, “A Fixed Terrestrial Photogrammetric System for Landslide Monitoring,” in *Modern Technologies for Landslide Monitoring and Prediction*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 43–67. doi: 10.1007/978-3-662-45931-7_3.

[139] R. Roncella, G. Forlani, M. Fornari, and F. Diotri, “Landslide monitoring by fixed-base terrestrial stereo-photogrammetry,” *ISPRS Annals of the Photogrammetry,*

Remote Sensing and Spatial Information Sciences, vol. II–5, no. 5, pp. 297–304, May 2014, doi: 10.5194/isprsannals-II-5-297-2014.

[140] S. I. Granshaw, "Bundle Adjustment Methods in Engineering Photogrammetry," *The Photogrammetric Record*, vol. 10, no. 56, pp. 181–207, 1980, doi: 10.1111/j.1477-9730.1980.tb00020.x.

[141] L. Borgatti and G. Tosatti, "Slope Instability Processes Affecting the Pietra Di Bismantova Geosite (Northern Apennines, Italy)," *Geoheritage*, vol. 2, no. 3, pp. 155–168, 2010, doi: 10.1007/s12371-010-0023-8.

[142] N. Bruno, A. Giacomini, R. Roncella, and K. Thoeni, "Influence of illumination changes on image-based 3D surface reconstruction," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 43, no. B2-2021, pp. 701–708, 2021, doi: 10.5194/isprs-archives-XLIII-B2-2021-701-2021.

[143] I. Nikolov and C. Madsen, "Benchmarking close-range structure from motion 3D reconstruction software under varying capturing conditions," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10058 LNCS, pp. 15–26, 2016, doi: 10.1007/978-3-319-48496-9_2.

[144] Y. D. Rajendra *et al.*, "Evaluation of partially overlapping 3D point cloud's registration by using ICP variant and cloudcompare," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. XL–8, no. 1, pp. 891–897, 2014, doi: 10.5194/isprsarchives-XL-8-891-2014.

[145] L. Barazzetti, G. Forlani, F. Remondino, R. Roncella, and M. Scaioni, "Experiences and achievements in automated image sequence orientation for close-range photogrammetric projects," *Videometrics, Range Imaging, and Applications XI*, vol. 8085, p. 80850F, 2011, doi: 10.1117/12.890116.

[146] T. Luhmann, S. Robson, S. Kyle, and I. Harley, *Close Range Photogrammetry: Principles, Techniques and Applications*. Whittles Publishing, 2006.

[147] F. Chollet, "Keras." 2015. [Online]. Available: https://github.com/fchollet/keras

[148] M. Abadi *et al.*, “TensorFlow: A system for large-scale machine learning,” *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016*, pp. 265–283, May 2016, [Online]. Available: http://arxiv.org/abs/1605.08695

[149] T. Blaschke *et al.*, “Geographic Object-Based Image Analysis – Towards a new paradigm,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 180–191, 2014, doi: 10.1016/j.isprsjprs.2013.09.014.

[150] G. Litjens *et al.*, “A Survey on Deep Learning in Medical Image Analysis,” no. 1995, 1998.

[151] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, *A survey of transfer learning*, vol. 3, no. 1. Springer International Publishing, 2016. doi: 10.1186/s40537-016-0043-6.

[152] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2010, doi: 10.1109/cvpr.2009.5206848.

[153] M. Raghu, “Transfusion : Understanding Transfer Learning for Medical Imaging,” no. NeurIPS, 2019.

[154] J. R. Z. Id *et al.*, “Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs : A cross-sectional study,” pp. 1–17, 2018, doi: https://doi.org/10.1371/journal.pmed.1002683.

[155] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans Knowl Data Eng*, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.

[156] M. Buda, A. Maki, and M. A. Mazurowski, “A systematic study of the class imbalance problem in convolutional neural networks,” pp. 249–259, 2018.

[157] Chawla, Nitesh V., Bowyer, Kevin W., Hall, Lawrence O., and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique Nitesh,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002, doi: 10.1002/eap.2043.

[158] D. M. W. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,” pp. 37–63, 2020, [Online]. Available: http://arxiv.org/abs/2010.16061

8 APPENDIX

The specific description and the motivations of the core code modifications of the neural algorithm are introduced in Chapter 2. The code pack for training and saving model predicted outputs are provided below, as the results presented in Chapter 5 are produced using these reference codes.

8.1 TRAINING_WITH_COCO_STYLE_DATASET.PY

```
# load the libraries
import os
import json
import string
import warnings
import tifffile
import datetime
import imgaug
import numpy as np
import pandas as pd
import tensorflow as tf
from mrcnn import utils
from datetime import datetime
from imgaug import augmenters as iaa
import mrcnn.model as modellib
from mrcnn.config import Config
from tensorflow.keras.callbacks import LearningRateScheduler


##################################################################
####                   TRAIN MASK RCNN MODEL                  ####
##################################################################

os.environ['TF_CPP_MIN_LOG_LEVEL'] = "3"
print("TensorFlow version:", tf.__version__)
warnings.filterwarnings('ignore')
tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)

ROOT_DIR = os.path.abspath(r'C:\Mask-RCNN-TF2')
DEFAULT_LOGS_DIR = os.path.abspath(r'D:\MRCNN_Logs')
DATASETS_DIR = os.path.abspath(r'C:\Datasets')
DATASETS_PATH = [f"{DATASETS_DIR}\\Dataset{i}" for i in range(1, 8)]


class BlocksConfig(Config):
    NAME = 'block'
    GPU_COUNT = 1
    FPN_CLASSIF_FC_LAYERS_SIZE = 1024
    EPOCHS = 1
    IMAGES_PER_GPU = 2
    IMAGE_CHANNEL_COUNT = 1
    BACKBONE = 'resnet101'
    NUM_CLASSES = 1 + 1
    USE_MINI_MASK = True
    IMAGE_RESIZE_MODE = 'none'
    IMAGE_MIN_DIM = 512
    IMAGE_MAX_DIM = 512
    LEARNING_RATE = 0.0001
    MASK_SHAPE = [28, 28]
    MAX_GT_INSTANCES = 50
    RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128)
    RPN_ANCHOR_RATIOS = [0.5, 1, 2]
    RPN_ANCHOR_STRIDE = 1
    RPN_TRAIN_ANCHORS_PER_IMAGE = 256
    ROI_POSITIVE_RATIO = 0.33
    TRAIN_ROIS_PER_IMAGE = 150
    STEPS_PER_EPOCH = 6000
    TOP_DOWN_PYRAMID_SIZE = 256
    VALIDATION_STEPS = 4500
    WEIGHT_DECAY = 0.0035
    DETECTION_MIN_CONFIDENCE = 0.65
    LOSS_WEIGHTS = {
        "rpn_class_loss": 1.,
        "rpn_bbox_loss": 1.,
        "mrcnn_class_loss": 1.,
        "mrcnn_bbox_loss": 1.,
        "mrcnn_mask_loss": 1.
    }


class BlocksDataset(utils.Dataset):
    def __init__(self, class_map=None):
        super().__init__(class_map)

    def load_dataset(self, images_dir):
        json_file_path = os.path.join(images_dir, "annotations.json")
        with open(json_file_path, 'r') as json_file:
            coco_json = json.load(json_file)

        dataset_name = "object"
        for category in coco_json['categories']:
            class_id = category['id']
            class_name = category['name']
            if class_id < 1:
```

```
            print('Error: Class id for "{}" cannot be less than one. (0 is reserved for the background)'.format(
                class_name))
            return

        self.add_class(dataset_name, class_id, class_name)

    annotations = {}
    for annotation in coco_json['annotations']:
        image_id = annotation['image_id']
        annotations.setdefault(image_id, []).append(annotation)

    seen_images = set()
    for image in coco_json['images']:
        image_id = image['id']
        if image_id in seen_images:
            print("Warning: Skipping duplicate image id: {}".format(image_id))
            continue
        seen_images.add(image_id)

        try:
            image_file_name = image['file_name']
            image_width = image['width']
            image_height = image['height']
            image_annotations = annotations.get(image_id, [])
        except KeyError as key:
            print("Warning: Skipping image (id: {}) with missing key: {}".format(image_id, key))
            continue

        if not image_annotations:
            continue

        image_path = os.path.abspath(os.path.join(images_dir, image_file_name))

        self.add_image(
            source=dataset_name,
            image_id=image_id,
            path=image_path,
            width=image_width,
            height=image_height,
            annotations=image_annotations
        )

  @staticmethod
  def rle_decode(mask_rle, size):
    mask = np.zeros(size[0] * size[1], dtype=np.uint8)
    starts = mask_rle[::2]
    lengths = mask_rle[1::2]
    current_position = 0
    for start, length in zip(starts, lengths):
            current_position += start
            mask[current_position:current_position + length] = 1
            current_position += length
        return mask.reshape(size[::-1]).T

    def load_image(self, image_id):
        info = self.image_info[image_id]
        image_file = tifffile.imread(info['path'])
        image = image_file[..., np.newaxis]
        return image

    def image_reference(self, image_id):
        info = self.image_info[image_id]
        if info["source"] == "shapes":
            return info["shapes"]
        else:
            super(self.__class__).image_reference(self)

    def load_mask(self, image_id):
        global mask
        instance_masks = []
        class_ids = []
        annotations = self.image_info[image_id]["annotations"]

        for annotation in annotations:
            class_id = annotation['category_id']

            if class_id:
                mask_rle = annotation["segmentation"]["counts"]
                size = annotation["segmentation"]["size"]
                mask = self.rle_decode(mask_rle, size)
                instance_masks.append(mask)
                class_ids.append(class_id)

        if class_ids:
            mask = np.stack(instance_masks, axis=2).astype(np.bool)
            class_ids = np.array(class_ids, dtype=np.int32)

        return mask, class_ids


# Training function
def train(model, config):
    # Load and prepare training dataset
    dataset_train = BlocksDataset()
    dataset_train.load_dataset(images_dir=os.path.join(DATASETS_PATH, 'train'))
    dataset_train.prepare()

    # Load and prepare validation dataset
    dataset_val = BlocksDataset()
    dataset_val.load_dataset(images_dir=os.path.join(DATASETS_PATH, 'val'))
    dataset_val.prepare()

    augmentation = iaa.Sequential([
        iaa.Fliplr(0.5),
        iaa.Flipud(0.5)
    ])

    # Fine-tune the model
    print("Fine-tuning all layers")
    model.train(dataset_train, dataset_val,
                learning_rate=config.LEARNING_RATE,
                epochs=config.EPOCHS * 500,
                layers='all',
                augmentation= augmentation)

    # Save trained weights
    output_model = os.path.join(DEFAULT_LOGS_DIR, "mask_rcnn_block_resnet101.h5")
    model.keras_model.save_weights(output_model)


# Main function
def main():
    config = BlocksConfig()
    config.display()
    model = modellib.MaskRCNN(mode="training", config=config,
model_dir=DEFAULT_LOGS_DIR)
    train(model, config)


if __name__ == '__main__':
    main()
```

8.2 SAVE RESULT TO JSON

```
#load the libraries
import os
import re
import json
import warnings
import numpy as np
import tifffile
from itertools import groupby
from mrcnn.config import Config
from mrcnn import model as modellib
import tensorflow as tf

##############################################################
####      SAVE RESULTS FROM MASK-RCNN TO JSON FILE      ####
##############################################################

MODEL_PATH = os.path.abspath(r'D:\MRCNN_Logs\MRCNN_Logs_2')
DATASETS_PATH = r"C:\Dati\Pilkington"

class InferenceConfig(Config):
   NAME = 'block'
   GPU_COUNT = 1
   FPN_CLASSIF_FC_LAYERS_SIZE = 1024
   IMAGES_PER_GPU = 1
   IMAGE_CHANNEL_COUNT = 1
   BACKBONE = 'resnet101'
   NUM_CLASSES = 1 + 1
   USE_MINI_MASK = False
   IMAGE_RESIZE_MODE = 'none'
   IMAGE_MIN_DIM = 512
   IMAGE_MAX_DIM = 512
   MASK_SHAPE = [28, 28]
   RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128)
   RPN_ANCHOR_RATIOS = [0.5, 1, 2]
   RPN_ANCHOR_STRIDE = 1
   TOP_DOWN_PYRAMID_SIZE = 256

def load_image(image_path):
   image_file = tifffile.imread(image_path)
   image = image_file[..., np.newaxis]
   return image


def binary_mask_to_rle(binary_mask):
   rle = {'counts': [], 'size': list(binary_mask.shape)}
   counts = rle.get('counts')
   for i, (value, elements) in enumerate(groupby(binary_mask.ravel(order='F'))):
      if i == 0 and value == 1:
            counts.append(0)
        counts.append(len(list(elements)))
    return rle


def process_images_with_confidence_level(confidence_level, folder):
    class DynamicInferenceConfig(InferenceConfig):
        DETECTION_MIN_CONFIDENCE = confidence_level

    dynamic_config = DynamicInferenceConfig()
    model = modellib.MaskRCNN(mode="inference", config=dynamic_config, model_dir="logs")
    model.load_weights(model_path, by_name=True)

    annotations = []
    for image_path in image_paths:
        image_filename = os.path.basename(image_path)
        image_id = int(re.search(r'\d+', image_filename).group())
        print("Processing", image_filename)

        image = load_image(image_path)
        result = model.detect([image])[0]

        masks = result['masks']
        num_instances = masks.shape[2]

        for i in range(num_instances):
            binary_mask = masks[:, :, i]
            rle_mask = binary_mask_to_rle(binary_mask)
            annotations.append({
                "image_id": int(image_id),
                "segmentation": rle_mask,
                "area": int(np.sum(binary_mask)),
                "iscrowd": 1,
                "bbox": result['rois'][i].tolist(),
                "category_id": int(result['class_ids'][i]),
                "id": int(i + 1),
                "scores": float(result['scores'][i])
            })

    confidence_str = str(confidence_level).replace(".", "_")
    output_json_path = os.path.join(folder, f"annotations_pred_{confidence_str}.json")
    with open(output_json_path, "w") as json_file:
        json.dump({"annotations": annotations}, json_file, indent=4)


model_path = os.path.join(MODEL_PATH, "mask_rcnn_block_resnet101.h5")
dataset_paths = [f"{DATASETS_PATH}\\Dataset{i}" for i in range(1, 10)]

for dataset_path in dataset_paths:
    folder = os.path.join(dataset_path, "val")
```

```
image_paths = [os.path.join(folder, fname) for fname in os.listdir(folder) if fname.endswith(".tif")]

print(f"Processing dataset: {dataset_path}")
print(f"  -> Using folder: {folder}")

for confidence_level in [0.9, 0.95, 0.99]:
    print(f"    -> Processing with confidence level: {confidence_level}")
    process_images_with_confidence_level(confidence_level, folder)
```