CONSTRUCTION DEFECT DETECTION USING AI WITH AUGMENTED REALITY

MSc Research Project
MSCAI1

Deep Kantilal Patel
Student ID: X23107260

School of Computing
National College of Ireland

Supervisor: SHERESH ZAHOOR

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	…… Deep Kantilal Patel ………………………………………………………………
Student ID:	……X23107260…………………………………………………………………….……
Programme:	……MSCAI……………………………………… **Year:** …2023-24..
Module:	… MSc Research Project …………………………………………………….………
Supervisor:	…… SHERESH ZAHOOR ……………………………………………….………
Submission Due Date:	……16 September 2024……………………………………………………….………
Project Title:	… Construction Defect Detection using AI with Augmented Reality…
Word Count:	…6240……………………… **Page Count**…20……………………….……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	……Deep Kantilal Patel………………………………………………………………
Date:	……16 September 2024…………………….………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Construction Defect Detection using AI with Augmented Reality

Deep Kantilal Patel
Student ID: x23107260

Abstract

This report explores how a systematic detection of building using deep learning has been developed, specifically targeting defects such as Blister, Cracks, Peels, Seepage, Mold. The data set initially exhibited class imbalance, with oversampling and undersampling techniques used to ensure equal distribution of each defect type to improve the model's ability to generalize, data enhancement was implemented by techniques such as on rotation, flipping, zooming and light adjustment.

A convolutional neural network (CNN) was used as the primary model for fault detection. This model is integrated with OpenCV to facilitate pre-imaging and fault ranking, ensuring accurate detection and real-time classification of defects. The CNN model was trained on a compressed dataset and it showed high accuracy and robustness in classifying defects.

The results of this study highlight the effectiveness of deep learning models using traditional imaging techniques for automatic fault detection in building construction. The successful implementation of this system highlights its applicability to real-world production quality, providing scalable and effective solutions to improve product reliability and safety.

KeyWords- convolutional neural network (CNN), Machine Learning, Deep Learning, Augmented Reality(AR), Artificial Intelligence (AI), RNN, LSTM

Chapter 1: Introduction

1.1 Research Background

Based on the AI implications, analysing the structural integrity of the business is far better compared to traditional matters. The implication of augmented reality is effective for analysing the efficiency of defects and inspection of management procedures for construction work (Tan *et al.* 2024). Traditionally, the detection of defects in building relies on manual inspection methods, which are often time-consuming, labor-intensive, and prone to human error. These traditional methods may not detect the subtle deficiencies or hidden flaws at all times, whereas the AR technology allows for important potential issues to be identified. As the demand for efficient and accurate quality control methods increases, there is interest in using technology to automate defect detection is increasing. The below images provide information on the development of working performance in the construction project to detect any defects in the building for mitigating the risk of working sectors.

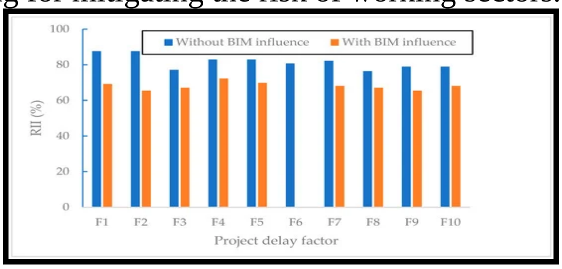

Figure 1.1: Influencing of working factors in a construction project with the support of BIM by mitigating delays

(Source: Alnase *et al.* 2023)

Augmented reality is effective in providing advanced building inspection for developing the working area by reducing the effects of real damages. Generally, AR is used for the detection of defects in construction areas, camera calibration, estimating orientation, analysing camera position, and implication of innovating ideas (Sahu *et al.* 2021). The working performance of AR is capable of implicating various cost-effective working conditions to the management team of construction projects to complete the work in an efficient possible manner. The working performance of augmented reality is effective in enhancing the efficiency and accuracy of project work. The innovative interface of any project is directly related to augmented reality (Lalik and Wątorek, 2021).

1.2 Importance of Research

The implications of this study are far-reaching. Using a defect detection system can significantly reduce the time and labor required for inspection, leading to faster completion and lower costs. Furthermore, improving the accuracy of defect detection and accuracy contributes to the long-term sustainability and safety of the construction industry. Based on the technological tools' engineers are capable of enhancing the quality of their work, which is effective in enhancing the satisfaction of the purchaser of the building. The importance of this research is its potential to revolutionize quality management practices in the construction industry. Construction defects, such as cracks, peel, blisters, seepage, and mold are not only common but if not detected and dealt with early can also cause serious damage to construction, which increases the cost of maintenance to increase; and premature failure of structures (Sabeti *et al.* 2021). The below provides information on the use of AI technology throughout the world. AI tool is used in various sectors to enhance the quality of project work.

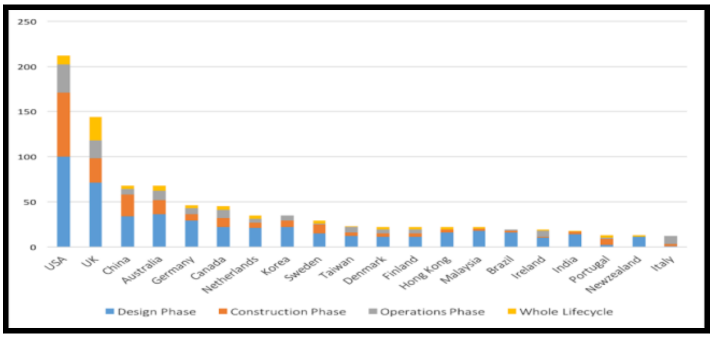

Figure 1.2: Use of AI for developing the working performance in the project work in the world

(Source: bimthinkspace.com, 2024)

Ultimately, this research addresses a fundamental need in the construction industry, providing practical solutions to increase the reliability and effectiveness of defect detection and thus contribute to broader objectives namely, improving the quality of construction, ensuring construction integrity, and advancing the technological applications of the built products. Both AI and AR are effective in enhancing the real-time multimodal for ensuring safety issues. The management team of construction projects is allowed to make perfect decisions for mitigating any types of issues, which is created during the detection of defects, or working procedures. The main importance of the AI implication with AR in the construction defects detection procedure is that it is applicable for enabling the real-time monitoring procedures of construction to progress the working performance of engineers.

1.3 Research problem

AI is most useful for the detection of any defects, which are created in the construction work. The construction industry relies heavily on the quality and integrity of the materials used in the building construction. Defects in construction materials such as blisters, cracks, peels, seepage, mold, etc. can compromise the durability, safety and overall performance of buildings, so it is crucial to detect these defects early to prevent possible structural failure edge, reduce maintenance costs and ensure long term reliability of buildings and infrastructure (Abd Al Rahman and Mousavi, 2020). Based on the implementation of AI technology in the construction working procedure allows for mitigating all of these issues to reduce the impact of manual errors.

1.4 Research Hypothesis

H_1: Based on the use of AI technology with augmented reality it is capable of detecting defects in construction projects.
H_0: Using AI technology with Augmented Reality is not effective in detecting defects in construction work.

1.5 Research Question

The research question is posed to enhance the quality of the investigation, those questions are,

1. Why the use of augmented reality is important to detect any defects in the construction project's work?
2. How AI technology is effective in understanding the working performance of construction workers to detect any issues in buildings?
3. What are the main features of augmented reality for mitigating all challenges in construction project work to detect defects?

1.6 Research Objective

Based on the addressing of the research question, the research objective is derived those are,

- To analyses the importance of the use of augmented reality for detecting any defects in the construction project's work
- To determine the procedures of use of AI technology for understanding the working performance of construction workers to detect any issues in buildings
- To evaluate the main features of AI technology based on Augmented reality for mitigating all challenges in construction project work to detect defects

1.7 Research limitation

Although the model was developed for real-time error detection, combining OpenCV with the deep learning algorithms can still encounter latency issues, especially when creating images with dimensions of high-quality processing or other optimizations that may be required to ensure continuous real-time operation.

The performance of the model is highly dependent on the quality of the input image. Differences in image resolution, noise, and lighting conditions can significantly affect error detection accuracy, which can result in false positives or negatives.

1.8 Outline of Reports structures

Figure 1.3: Outline the structure of the research
(Source: Self-Created)

Chapter 2: Literature Review

Introduction

The use of artificial intelligence (AI) and augmented reality (AR) in construction defect detection represents a significant advance in the accuracy and efficiency of building inspections. This review examines recent developments in AI and AR technologies to identify industry deficiencies, focusing on techniques such as image processing, machine learning and data enhancement

AI and imaging in construction defect detection

AI-powered methods have changed how defects are detected in construction. Computer-aided systems that use computer vision and deep learning techniques are used to supplement or replace traditional methods that typically require more manual and labor. Recent various research have shown that this technology is effective in detecting various product defects.

Deep learning for error classification

An important study by Zhang et al. (2019) used deep convolutional neural networks (CNNs) to detect and classify defects in construction images. Their model achieved high accuracy in separating cracks, spalls, and other surface defects by training a dataset enriched with synthetic images to improve the robustness and overall quality of the model (Zhang, Y., 2010). & Zhao , X. (2019).

Integrating AI and AR

The combination of AI and AR provides a revolutionary approach to fault detection. Zoo and more. (2020) explored how AR can be used in conjunction with machine learning models to provide real-time visualization of defects and assist in on-site decision making. Their study highlighted how AR can mask fault information into real-world processes, increasing the potential of AI models in practical applications (Xu, Y., Zhang, J., & Wang, H. (2020).).

Data Augmentation Techniques

Data augmentation is crucial for enhancing the performance of AI models. A study by Chen et al. (2021) investigated various data augmentation techniques, including rotation, scaling, and color adjustment, to improve defect detection accuracy. Their findings emphasized that synthetic data generation can significantly boost model performance, particularly in scenarios with limited training data (Chen, Y., Liu, X., & Liu, J. (2021).

Image based detection

Image-based methods use visual information to identify and classify defects. Traditional methods include simple imaging techniques such as thresholding and edge detection. However, recent research has emphasized the use of advanced machine learning algorithms. For example, Zhang et al. (2020) used convolutional neural networks (CNNs) for concrete crack detection, showing that CNNs outperform traditional methods in terms of accuracy and robustness (Zhang et al., 2020).

Machine learning and deep learning methods

Deep learning techniques, especially CNN, have become mainstream in fault detection, due to their ability to identify sequences from data. A study by Yang et al. (2021) and the author. The same is true of Wang. (2022) examined the use of transfer learning and data enhancement to enhance error detection performance in a limited data environment (Wang et al., 2022).

Advanced technology

Recent research has investigated the use of combined models and cluster learning to improve search performance. For example, Chen et al. (2021) proposed a hybrid approach that combines CNNs with support vector machines (SVMs) for improved fault classification (Chen et al., 2021). This approach shows promise in addressing the limitations of individual models and in improving overall accuracy.

Case studies and applications

Facilities and construction

Defects found in projects and construction have important safety and maintenance implications. The study of Liu et al. (2020) used deep learning techniques to detect cracks and defects in bridge inspection images, resulting in improved automation and reduced inspection time (Liu et al., 2020) as well as Kumar et al. (2022) investigated the use of deep learning for mold and leakage in building inspections, and highlighted the possibility of automated systems replacing manual inspections (Kumar et al., 2022).

Dataset limitations

One of the most important challenges in fault detection is the limitation of labeled data, especially for rare fault types. Data augmentation and synthetic data generation are possible solutions, but they need careful measurement to ensure data accuracy and usefulness (Kim et al., 2021).

General and adaptable controls

Another challenge is to ensure that models are sufficiently generalizable across locations and contexts. The study of Zhao et al. (2023) suggests that domain optimization techniques, which modify models trained on one data set to perform better on another, are important for increasing model adaptability (Zhao et al., 2023).

Chapter 3: Research Methodology

3.1 Introduction

The research methodology determines the implementation of the approach which is usable for the investigation. It specifically justifies that to gather necessary information for the research secondary data collection can be used. The methodology which will be used also outlines the process for data analysis which also helps in assessing key data factors. To make sure that the execution process is both efficient and effective methodology suggests to define the functional parameter. This structured approach is crucial in research investigation as it allows a thorough evaluation of the critical aspects required for accurate analysis and results.

3.2 Research method

The method involves the implementation of the secondary analysis approach which highlights the application of the constructional approach. This method emphasizes the use of

python programming for data processing and data analysis. Python is utilized mainly for the determination of the executable parameters which facilitates the logical analysis of the data. The process of execution involves defining and applying the specific parameters that guide the evaluation of the key data sections (El Jazzar *et al.*, 2021). The overall section also provides information about the details of the executable parameters.
Along with this approach machine learning and deep learning techniques are integrated to enhance the accuracy and effectiveness of data analysis (Zhou, and Güven, 2020). The AI implementation approach is implemented to determine and identify the key factors crucial for the research.

3.3 Data collection

The evaluation of secondary data that is collected from various secondary resources, focused on construction defects such as cracks, blisters, peel, seepage and molds. Each defect type was represented by a diverse set of images, capturing variations in size, shape and severity to ensure comprehensive coverage. To enhance the model robustness and generalizability of the defect detection model, the dataset also involves variations in background and lighting conditions, which allows the model to perform reliably across different contexts. Consistency and accuracy in data presentation were maintained with the help of standard labeling protocols. To address the imbalances in the dataset, two methods were employed: Undersampling was used to reduce the overrepresentation of the majority classes, whereas in oversampling along with techniques like SMOTE and artificial image generation was applied to bolster the underrepresented classes. The dataset was strategically split with 80% allocation for the training of the model, and 20 % for validation, to fine-tune the model performance.

3.4 Data Preprocessing

In data preprocessing firstly the image dataset is being effectively prepared for convolutional neural network. To do this I used a 'ImageDataGenerator' class from the keras library, which is instrumental in augmenting the dataset which enhance the model performance and generalization. This involves various augmentation techniques such as rescaling the pixel values, random rotation of up to 40 degrees, width and height shifts of 20%, and horizontal flips. These is done so that there are variability in the training data which will help to mitigate the overfitting.

3.5 Deep learning algorithms

The deep learning technique is applicable to determine the key executional elements of the determination. This provides the information about the handling of the data factors. The constructional evaluation supports the determination of the overall execution process by using OpenCV and deep learning approaches. This defines the implementation of deep learning models such as RNN, CNN, ANN, LSTM, and so on. CNN is determined as a "convolutional neural network", which is basically applicable to the evaluation of the image data. This introduces the application of image-processing approach (Dallasega *et al.*, 2021). Data collection requires careful planning and implementation to ensure a high-quality dataset for error detection. By overcoming issues such as data imbalance and ensuring high specification quality, the dataset is well suited for training and analyzing complex error detection models using OpenCV.

3.6 Tools and Technologies

In this research I have leveraged the artificial intelligence (AI) to enhance the defect detection accuracy in the images. This involves integrating machine learning (ML) and deep learning (DL) algorithms. Also I have utilized OpenCV in this research for its robust image processing capabilities, facilitating the practical application of these AI models. OpenCV is also used for real-time image processing and is extensively used for its powerful image manipulation capabilities, enabling tasks such as image transformation, feature extraction and object detection. I have selected python as the programming language for this project as it has extensive libraries that support machine learning and data analysis. The main libraries that I have used for building and training the neural network models are TensorFlow and Keras. TensorFlow gives a comprehensive computational framework and whereas keras simplifies the setting up of the networks. Along with this the I have used NumPy for data manipulation and Matplotlib for visualization of the outcomes.

3.7 Analytical Methods

The image preprocessing is crucial a step where images are resized to standard dimensions and subjected to augmentation techniques such as rotation, scaling, and flipping to expand the dataset artificially, enhancing the model's ability to generalize the new, unseen data. The convolutional neural network (CNN) model which has employed includes several layers like convolutional layers that extract relevant features from the images, pooling layers that reduce spatial dimensions to decrease the computational complexity, and dropout layers that mitigate the overfitting. The architecture concludes with fully connected layers topped with a softmax activation functions designed to classify the images into their respective defect categories accurately. Performance of the model is meticulously evaluated using accuracy as the primary metric, along with precision, recall and F1-Score which provided a critical insights into the model's efficacy across the individual classes.

3.8 Ethical Considerations

The most important section of the ethical evaluation defines the elimination of the copyright issue. This issue may rise when the author does not give created to the other's work. So, it is necessary to provide the necessary information regarding the source of the collected data and information. The proper authenticated data source is needed for the evaluation of the data. This defines the determination of the executional factors (Makhataeva, and Varol, 2020). Ethical determination also defines the issues related to data loss. This introduces a proper data security approach to remove those issues.

3.9 Research Limitation

Class imbalance

Class imbalance remains a significant challenge even after the application of undersampling and oversampling techniques, which leads to some classes being under or over sampled. This persistent imbalance causes the model to prioritize more frequent classes, which will reduce its accuracy in detecting the rare defects. In order to mitigate this issue, it is essential to balance the dataset using the advanced techniques such as class-aware loss functions to enhance the performance.

OpenCV Limitations

In OpenCV, although it offers the powerful tools for image processing, it may be limited in advanced error detection tasks when compared to more specialized deep learning frameworks. These limitations can restrict the use of sophisticated detection algorithms and complicate the handling of complex fault defect pattern. To address these limitations, detection capabilities along with OpenCV should involve the additional integration with complementary frameworks such as TensorFlow.

Chapter 4: Design and Implementation Specifications

4.1 Introduction

The dataset contains images divided into five different structural defect types: blister, cracks, peel, blisters, and molds. Each section consists of photographs, leaving the first section unbalanced.

Image size and normalization: All images were resized to the same resolution suitable for inclusion in the deep learning model. In addition, the pixel values were normalized to scale, increasing the performance of the model.

Sample parameters:

Layers: Multiple convolutional and pooling layers followed by fully connected layers

Function: ReLU function for the hidden layer and Softmax for the output layer.

Optimizer: The Adam optimizer was chosen for efficient gradient-based optimization.

Learning rate: A learning rate of 0.001 was used to balance convergence speed and accuracy.

Loss function: Since the problem is multiclasss, hierarchical cross-entropy was used.

4.2 Design approach

The design approach evaluates the involvement of the image data which are usable for training of the DL prototypes. The models/prototypes are designed by using the DL algorithms. This defines the application of the AI techniques. The functional section also illustrated the evaluation of the model/prototype parameters for the execution.

4.3 Implementation Specification

The implementation approach defines the initialization of the training approach for the DL models/prototypes. This determines the evaluation of the executable parameters which are usable for the evaluation. The process of execution supports to evaluation of the DL model/prototype construction and fitting approach (Kamari, and Ham, 2022). This way of execution provides information about the problem-solving approach in the detection process.

4.4 Key findings

The key findings provide information about the handling of the data parameters. The main finding is the construction of the DL models/prototypes. This determines the executable parameters which are usable for the determination. This provides information about the training and testing of the models/prototypes by using image data. The major finding also defines the defective section in the construction by using those DL algorithms.

5. Implementation

5.1 Preparation of data

The preparation process of data defines the construction of the data section. This introduces the implementation of the data functional section. This also provides the details of the executable parameters which are needed for the preparation. The preparation highlights the checking of the image data. It also highlights the training of the model/prototype by using the image data. The final testing is determined by using the evaluated outcome of the data section (Taheri *et al.*, 2022). The models/prototypes are fitted, and evaluated by those training data. The accuracy execution approach is implemented to understand the most suitable models/prototypes for the execution. This provides the information about the handling of the data elements.

5.2 Data splitting and Validation

First the model learns from the dataset which was preprocessed and augmentation. It trains over the 50 cycles, also I have included the dropout layer in it to mitigate the overfitting.

The dataset was splitted into two sets that are training data set and validation dataset using train-test split of 80-20. Where 80% is allocated for training the data and 20% of the data reserved for validation. To measure the performance of model metrics such as g accuracy, precision, recall, and F1-score were calculated.

OpenCV and trained CNN were used to detect defects on new images. The process included the following steps.

Preprocessing: Resized and normalized the input image using the OpenCV function.

Defect localization: Contour detection and other image processing techniques were used to identify areas of interest where defects may exist

Classification: The preprocessed image or the marked area was passed through the CNN model, which classified the errors.

5.3 Data Analysis

Imprort libraries

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
import matplotlib.pyplot as plt
import os
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
import numpy as np
import os
import shutil
```

```
from imblearn.over_sampling import RandomOverSampler
```

Figure 5.1: Import of the libraries

(Source: Own-Evaluated)

The libraries are implemented to determine the functional evaluation approach that is needed in the data execution process.
The data read section is implemented to read multiple image data which assists in the analysis process.
The training approach is implemented to train the data models using the image data. This provides 16334 images for 5 classes and other 4081 images for the other 5 classes.
The model setting approach is implemented to derive the executable factors which are needed for the evaluation. The accuracy determination approach is implemented to determine the accuracy values of the major section.

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

history = model.fit(
    train_generator,
    epochs=100,
    validation_data=validation_generator
)

4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 405ms/step - accuracy: 0.4150 - loss: 1.3576 - val_accuracy: 0.4000 - val_loss: 1.6721
Epoch 8/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 388ms/step - accuracy: 0.5800 - loss: 1.3023 - val_accuracy: 0.3000 - val_loss: 1.7640
Epoch 9/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 391ms/step - accuracy: 0.4350 - loss: 1.2467 - val_accuracy: 0.3500 - val_loss: 1.7094
Epoch 10/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 406ms/step - accuracy: 0.5167 - loss: 1.2226 - val_accuracy: 0.3500 - val_loss: 1.6402
Epoch 11/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 395ms/step - accuracy: 0.4683 - loss: 1.2695 - val_accuracy: 0.2000 - val_loss: 1.8866
Epoch 12/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 408ms/step - accuracy: 0.4417 - loss: 1.1410 - val_accuracy: 0.4500 - val_loss: 1.7083
Epoch 13/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 414ms/step - accuracy: 0.6483 - loss: 1.1133 - val_accuracy: 0.4000 - val_loss: 1.6535
Epoch 14/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 389ms/step - accuracy: 0.5017 - loss: 1.0965 - val_accuracy: 0.4000 - val_loss: 1.9287
Epoch 15/100
```

Figure 5.5: Evaluation of the model

(Source: Own-Evaluated)

The evaluated model data parameter defines the evaluated value points which highlights the step-wise execution using the epoch values. The determination highlights the accuracy as well as the losses of the model.
The model parameter determines the loss, value accuracy, accuracy, and value loss which describes the details of the model/prototype parameters.

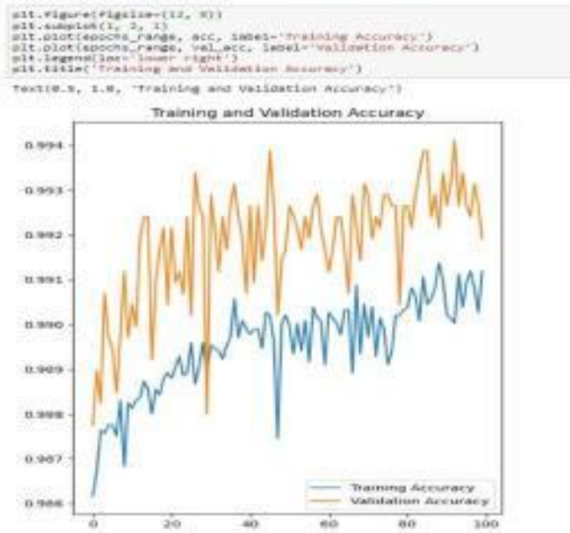

Figure 5.7: Training and validation accuracy plot

(Source: Own-Evaluated)

The training and assurance accuracy determines the functional portion of the evaluation. This provides the information about the usability of data context.

```
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

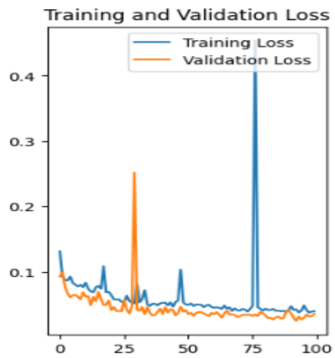

Figure 5.8: Training and validation accuracy plot

(Source: Own-Evaluated)

This also provides information about the training and assurance accuracy which assist in the investigation.

The test image approach is implemented to test the images of 5 different classes which highlights the accuracy of 0.9910, and loss of 0.0367.

The under sampling approach is implemented to evaluate the under sample data values.

```
def count_images(directory):
    return len(glob(os.path.join(directory, '*')))

blister_count = count_images(blister_dir)
cracks_count = count_images(cracks_dir)
mold_count = count_images(mold_dir)
peel_count = count_images(peel_dir)
seepage_count = count_images(seepage_dir)
```

```
print(f'Blister: {blister_count}')
print(f'Cracks: {cracks_count}')
print(f'Mold: {mold_count}')
print(f'Peel: {peel_count}')
print(f'Seepage: {seepage_count}')
```

```
Blister: 46
Cracks: 20155
Mold: 144
Peel: 33
Seepage: 37
```

Figure 5.11: Count of the image function

(Source: Own-Evaluated)

The count of the sample data determines the count of those parameters which are present in the data section.

```
min_count = min(blister_count, cracks_count, mold_count, peel_count, seepage_count)
print(f'Minimum count: {min_count}')
```

```
Minimum count: 33
```

Figure 5.12: Minimum image count

(Source: Own-Evaluated)

The minimum count of the determining image value is 33 which is applicable for the execution.

```
def undersample_and_copy(source_dir, target_dir, sample_size):
    all_images = glob(os.path.join(source_dir, '*'))
    sampled_images = random.sample(all_images, sample_size)

    for img_path in sampled_images:
        shutil.copy(img_path, target_dir)
```

```
undersample_and_copy(blister_dir, os.path.join(undersample_dir, 'blister'), min_count)
undersample_and_copy(cracks_dir, os.path.join(undersample_dir, 'cracks'), min_count)
undersample_and_copy(mold_dir, os.path.join(undersample_dir, 'mold'), min_count)
undersample_and_copy(peel_dir, os.path.join(undersample_dir, 'peel'), min_count)
undersample_and_copy(seepage_dir, os.path.join(undersample_dir, 'seepage'), min_count)
```

Figure 5.13: Under sampling data

(Source: Own-Evaluated)

The under-sampling data functionality defines the evaluated parameters that are applicable for the execution.

```
def count_images_in_classes(path):
    class_counts = {}
    for class_name in classes:
        class_path = os.path.join(path, class_name)
        num_images = len(os.listdir(class_path))
        class_counts[class_name] = num_images
    return class_counts


print("Original Dataset:")
original_counts = count_images_in_classes(dataset_path)
print(original_counts)


print("\nUndersampled Dataset:")
undersampled_counts = count_images_in_classes(undersampled_path)
print(undersampled_counts)
```

```
Original Dataset:
{'Blister': 46, 'Cracks': 20155, 'Mold': 144, 'Peel': 33, 'Seepage': 37}

Undersampled Dataset:
{'Blister': 33, 'Cracks': 33, 'Mold': 33, 'Peel': 33, 'Seepage': 33}
```

Figure 5.14: Count of data

(Source: Own-Evaluated)

The count of the data also details the overall count of various parameters that are usable for the evaluation.

```
import os
import cv2
import numpy as np
import shutil
from tqdm import tqdm
```

```
dataset_path = "Dataset/Assest"

undersampled_path = "Dataset/undersample"

classes = ["Blister", "Cracks", "Mold", "Peel", "Seepage"]

undersample_to = None
```

Figure 5.15: Classes execution for under-sampling

(Source: Own-Evaluated)

The classes determine the five determining elements such as Mold, Blister, Cracks, Seepage, and Peel. The path of the under-sampled data is also highlighted in this point.

```
def undersample_class(class_name, undersample_to=None):
    class_path = os.path.join(dataset_path, class_name)
    images = os.listdir(class_path)

    if undersample_to is None:
        undersample_to = min(len(images), 1000)  # Change 1000 to another number if needed

    undersampled_images = np.random.choice(images, undersample_to, replace=False)


    target_path = os.path.join(undersampled_path, class_name)
    os.makedirs(target_path, exist_ok=True)


    for img_name in undersampled_images:
        src = os.path.join(class_path, img_name)
        dst = os.path.join(target_path, img_name)
        shutil.copy(src, dst)
```

```
for class_name in classes:
    print(f"Undersampling class: {class_name}")
    undersample_class(class_name, undersample_to)
```

```
Undersampling class: Blister
Undersampling class: Cracks
Undersampling class: Mold
Undersampling class: Peel
Undersampling class: Seepage
```

Figure 5.16: Determination of the classes

(Source: Own-Evaluated)

The determination of various classes provides information about the functional sections of the data.

```
def count_images_in_classes(path):
    class_counts = {}
    for class_name in classes:
        class_path = os.path.join(path, class_name)
        num_images = len(os.listdir(class_path))
        class_counts[class_name] = num_images
    return class_counts


print("Original Dataset:")
original_counts = count_images_in_classes(dataset_path)
print(original_counts)


print("\nUndersampled Dataset:")
undersampled_counts = count_images_in_classes(undersampled_path)
print(undersampled_counts)
```

```
Original Dataset:
{'Blister': 46, 'Cracks': 20155, 'Mold': 144, 'Peel': 33, 'Seepage': 37}

Undersampled Dataset:
{'Blister': 33, 'Cracks': 33, 'Mold': 33, 'Peel': 33, 'Seepage': 33}
```

Figure 5.17: Original and under sampled data

(Source: Own-Evaluated)

This provides the evaluation of the original and also the under sampled dataset which provides information about the data values.

```
Original Dataset:
{'Blister': 46, 'Cracks': 20155, 'Mold': 144, 'Peel': 33, 'Seepage': 37}
```

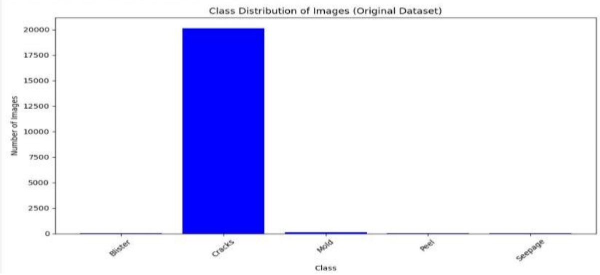

Figure 5.18: Class distribution of original images

(Source: Own-Evaluated)

The class distribution of the original images is highlighted in this plot of execution.

```
Undersampled Dataset:
{'Blister': 20, 'Cracks': 20, 'Mold': 20, 'Peel': 20, 'Seepage': 20}
```

Figure 5.19: Class distribution of under-sampled images

(Source: Own-Evaluated)

The class distribution of the under-sampled images is highlighted in this plot of execution.

Oversampling

```
from imblearn.over_sampling import RandomOverSampler
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
import numpy as np
import os
import shutil

import cv2

import matplotlib.pyplot as plt
from collections import Counter
from imblearn.over_sampling import RandomOverSampler
```

Figure 5.20: Oversampling approach

(Source: Own-Evaluated)

The oversampling data approach is implemented to evaluate the oversampled data factors.

```python
# Path to the dataset folder
dataset_path = "C:/Users/Deep/OneDrive/Desktop/MSC AI/Sem 2/Practicum/Final/project/Assest"

# Output path for oversampled dataset
oversampled_path = "C:/Users/Deep/OneDrive/Desktop/MSC AI/Sem 2/Practicum/Final/project/oversample"

# Define the classes (folders) in your dataset
classes = ["Blister", "Cracks", "Mold", "Peel", "Seepage"]
```

Figure 5.21: Classes for over-sampling

(Source: Own-Evaluated)

The classes of the over-sampling data are determined in this case of execution.

```python
def oversample_class(class_name, oversample_to=None):
    class_path = os.path.join(dataset_path, class_name)
    images = os.listdir(class_path)
    num_images = len(images)

    if oversample_to is None:
        oversample_to = max(num_images, 1000)

    # Create target directory if it doesn't exist
    target_path = os.path.join(oversampled_path, class_name)
    os.makedirs(target_path, exist_ok=True)

    # Calculate oversampling ratio
    ratio = oversample_to // num_images

    # Perform oversampling by replicating images
    for img_name in images:
        img_path = os.path.join(class_path, img_name)
        img = cv2.imread(img_path)

        for i in range(ratio):
            new_img_name = f"{os.path.splitext(img_name)[0]}_{i}.{os.path.splitext(img_name)[1]}"
            new_img_path = os.path.join(target_path, new_img_name)
            cv2.imwrite(new_img_path, img)
```

Figure 5.22: Function for over-sampling

(Source: Own-Evaluated)

The functional section of the class determination approach is defined in this point of evaluation for over-sampling.

```python
# Iterate over each class and perform oversampling
for class_name in classes:
    print(f"Oversampling class: {class_name}")
    oversample_class(class_name)
```

```
Oversampling class: Blister
Oversampling class: Cracks
Oversampling class: Mold
Oversampling class: Peel
Oversampling class: Seepage
```

Figure 5.23: Class determination

(Source: Own-Evaluated)

The class determination defines the evaluated value portion that is used in the case of execution.

```
Original Dataset:
{'Blister': 46, 'Cracks': 20155, 'Mold': 144, 'Peel': 33, 'Seepage': 37}
```

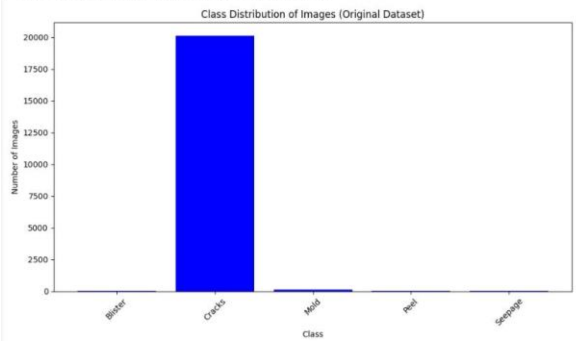

Figure 5.24: Class distribution of original images

(Source: Own-Evaluated)

The class distribution of the original images is highlighted in this plot of execution.

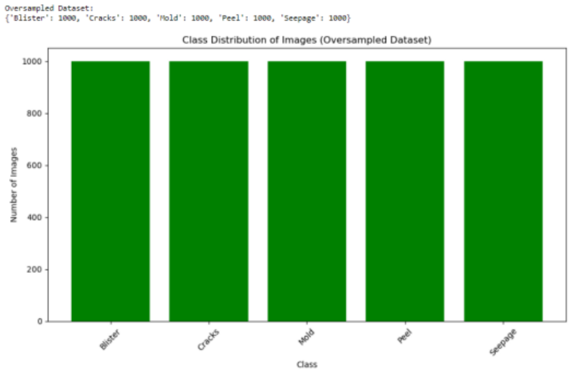

Figure 5.25: Class distribution of over-sampled images

(Source: Own-Evaluated)

The class distribution of the over-sampled images is highlighted in this plot of execution.

The model parameter provides information about the size of the batch, image width, and height.

The CNN model setting, execution, and evaluation are implemented in this point of determination.

The model fit approach is implemented for the execution of the model/prototype parameters.

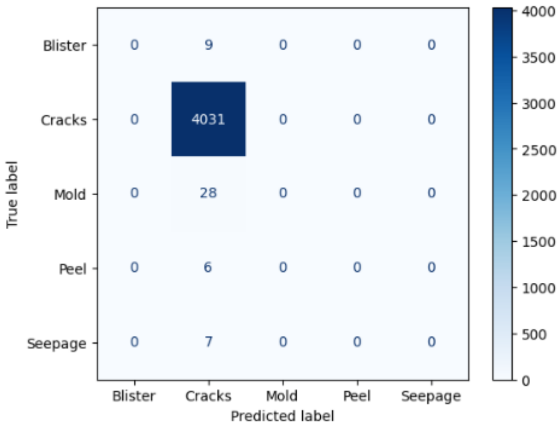

Figure 5.29: Confusion (Matrix)

(Source: Own-Evaluated)

The matrix determines the relational evaluation between true, and also the predicted labels. This provides the relational execution between five classes.

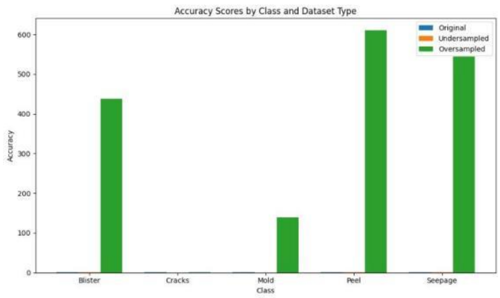

Figure 5.30: Accuracy score by different classes and also the dataset types

(Source: Own-Evaluated)

The accuracy evaluation is plotted in this section for the classes of original, over, and under sampled. This highlights the maximum oversampled data value for the execution of the section.

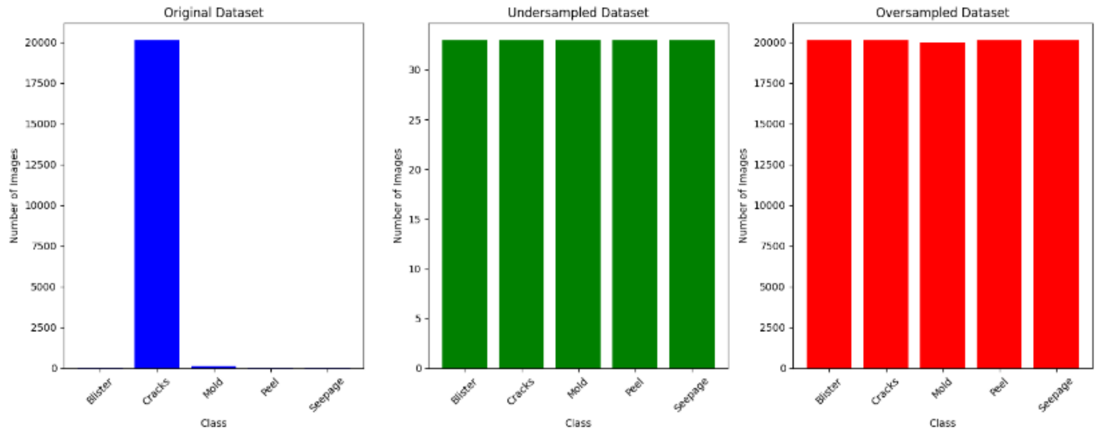

Figure 5.31: Number of images with respect to classes and dataset types

(Source: Own-Evaluated)

The number of images evaluated with respect to the classes and also the dataset types such as original, over, and under sampled.

The loading of the data is applicable to prepare the data and the split approach is needed for the execution.

Chapter 6: Evaluation

6.1 Introduction

This work successfully used a deep learning-based algorithm to identify manufacturing defects such as blisters, cracks, peel, seepage, and mold. The use of data enhancement and class balance methods significantly improved the performance of the model. The system shows promise for use in real-world scenarios, with the potential for further improvement through additional training data and model development.

6.2 Performance measure

CNN model construction approach is implemented for the determination.

The evaluated CNN model/prototype determines the functional sections of the executional parameters.

```
629/629 ━━━━━━━━━━━━━━━━━━━━ 13s 21ms/step
Oversampled Dataset Metrics:
Precision: 0.0414
Recall: 0.2035
F1 Score: 0.0688
              precision    recall  f1-score   support

     Blister       0.00      0.00      0.00      4035
      Cracks       0.00      0.00      0.00      4067
        Mold       0.00      0.00      0.00      4012
        Peel       0.20      1.00      0.34      4094
     Seepage       0.00      0.00      0.00      3908

    accuracy                           0.20     20116
   macro avg       0.04      0.20      0.07     20116
weighted avg       0.04      0.20      0.07     20116
```

Figure 5.35: Under sampled data evaluation

(Source: Own-Evaluated)

This describes the parameters and key elements of the Under sampled data.

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step
Oversampled Dataset Metrics:
Precision: 0.2287
Recall: 0.2070
F1 Score: 0.0928
              precision    recall  f1-score   support

     Blister       0.00      0.00      0.00       216
      Cracks       0.19      1.00      0.32       190
        Mold       1.00      0.09      0.16       192
        Peel       0.00      0.00      0.00       210
     Seepage       0.00      0.00      0.00       192

    accuracy                           0.21      1000
   macro avg       0.24      0.22      0.10      1000
weighted avg       0.23      0.21      0.09      1000
```

Figure 5.36: Oversampled data evaluation

(Source: Own-Evaluated)

This describes the parameters and key elements of the Oversampled data.

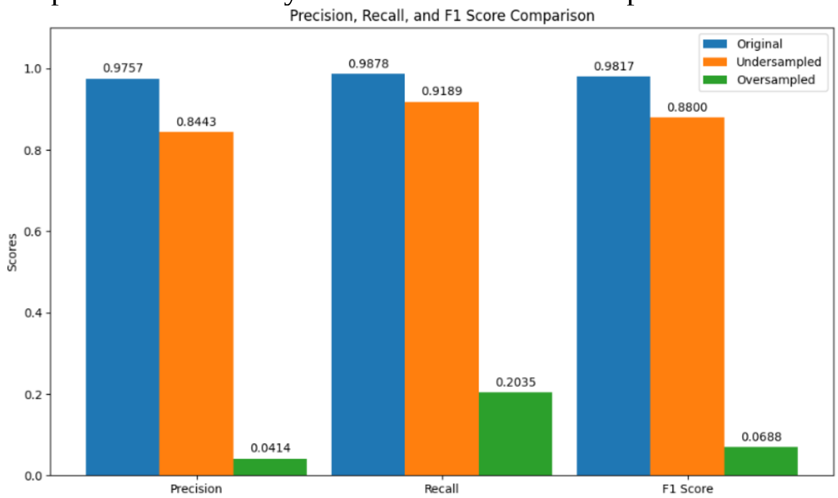

Figure 5.37: Comparison of F1, Recall, and Precision with respect to dataset types

(Source: Own-Evaluated)

The comparison plot is determined for the execution of the F1, Recall, and Precision. This defines the evaluation of those parameters for original, under, and oversampled data. In this case, the maximum precision is 0.2287 which is determined in the case of oversampling. The oversampled also defines the maximum recall and F1-scoring.

The training image is applicable to the process of the image.

Figure 5.39: Original and defective data Determination 1

(Source: Own-Evaluated)

This highlights the original and the defective determination for one type of defect.

The testing image data is applicable for the execution of the supportable model/prototype.

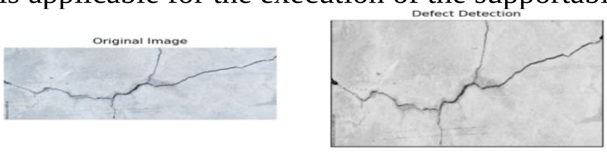

Figure 5.40: Original and defective test data Determination 1

(Source: Own-Evaluated)

This highlights the original and the defective determination test image data.

The all-over execution provides information about the application of Python. The coding process determines the evaluated parameters such as deep learning models to detect and predict the cracks in the construction. The defect evaluation is evaluated by using the collected images. The image processing approach is implemented to process the image for execution. The final evaluation defines the parameters that are needed for the detection of the fault in the section. The process of evaluation supports understanding the under, and over-sampling approaches.

Chapter 7: Conclusions and Discussion

7.1 Conclusion

The detection process implements deep learning techniques to evaluate the images for execution. The image data execution approach is applicable to determine the images in an informatic way. This assists in training the model with the sample image data. The testing provides information about the handling of the detection approach. This highlights the application of the executable functional factors which are needed for the evaluation. The final determination supports understanding the major evaluated parameters that are necessary for the execution. The process of finding also provides information about the evaluated parameters that are needed to determine the defects in the construction.

One of the most important of the fault detection systems is the Convolutional Neural Network (CNN), which was used to classify and detect defects in images. For easy preprocessing and fault localization, CNN is used with OpenCV for real-time detection and accurate classification. The model was trained on the augmented data set and exhibited robust performance with high accuracy across all error classes.

7.2 Discussion

The all-over execution defines the implementation of the CNN deep learning approach for the image data execution. This defines the implementation of the image recognition, and image processing approach to detect the defects in the image. In this case original colored image is applicable to detect the defects. This provides information about the presence of the defects by using the RBG color concept. On the other hand, the model evaluation approach is implemented to evaluate all those parameters that are applicable for the execution. The accuracy determination approach is implemented to evaluate the performance of each model/prototype. The evaluation determines the original, over, and under-sample elements. This provides information about the handling of each data factor. The executional pot determines that oversample has the maximum parameter values such as recall, F1, and precision.

7.3 Future Work

The modification in the detection approach is needed to make it more interactive, and also informatic. This defines the usability of the more advanced technique to reduce the error rate in the detection process. This defines the implementation of the executable factors which are usable for the evaluation. The construction approach is needed to evaluate those parameters that are usable for the determination. The future work also highlights the need for 3D evaluation, and determination to detect the defect more preciously. This view highlights the more accurate viewpoint which assists in understanding the present position of the defect. This defines the virtual construction of the overall construction area with the details of the evaluated value points. The future work relates the upgradation in the technology to resolve the issues related to the technological upgrade process (Taha *et al.*, 2021). The functional evaluation supports finding the best suitable approach for the designing of the 3D architecture of the building to detect the fault easily.

7.4 Recommendation

The primary recommendation for this design approach is the implementation of more reliable and user-friendly techniques. DL algorithms implement a complex problem-solving approach which needs to be replaced by a more sustainable and suitable construction data model/prototype. This needs to implement the analytical approach to upgrade the quality of the determination process. The major changes required in the AR section need to bring a new approach to the building construction and defect detection process (Martins *et al.*, 2022). The advancement in AI technology needs to be implemented to increase the accuracy of the model/prototype and reduce the time of execution.

References

Abd Al Rahman, M. and Mousavi, A., 2020. A review and analysis of automatic optical inspection and quality monitoring methods in electronics industry. Ieee Access, 8, pp.183192-183271.

Alnaser, A.A., Alsanabani, N.M. and Al-Gahtani, K.S., 2023. BIM Impact on Construction Project Time Using System Dynamics in Saudi Arabia's Construction. Buildings, 13(9), p.2267.

Al-Sabbag, Z.A., Yeum, C.M. and Narasimhan, S., 2022. Interactive defect quantification through extended reality. Advanced Engineering Informatics, 51, p.101473.

bimthinkspace.com (2024) Bim thinkspace: Global trends in BIM Research. Available at: https://www.bimthinkspace.com/2016/09/global-trends-in-bim-research.html (Accessed: 06 August 2024).

Dallasega, P., Revolti, A., Schulze, F. and Martinelli, M., 2021. Augmented reality to increase efficiency of MEP construction: a case study. In ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction (Vol. 38, pp. 459-466). IAARC Publications.

digitaldefynd.com, T. (2024) 20 pros and cons of augmented reality [2024], DigitalDefynd. Available at: https://digitaldefynd.com/IQ/augmented-reality-pros-cons/#:~:text=The%20long%2Dterm%20use%20of,need%20to%20process%203D%20visuals. (Accessed: 06 August 2024).

El Jazzar, M., Schranz, C., Urban, H. and Nassereddine, H., 2021. Integrating construction 4.0 technologies: A four-layer implementation plan. Frontiers in Built Environment, 7, p.671408.

Ho, P.T., Albajez, J.A., Santolaria, J. and Yagüe-Fabra, J.A., 2022. Study of augmented reality based manufacturing for further integration of quality control 4.0: A systematic literature review. Applied Sciences, 12(4), p.1961.

Júnior, G.G.S., Satyro, W.C., Bonilla, S.H., Contador, J.C., Barbosa, A.P., de Paula Monken, S.F., Martens, M.L. and Fragomeni, M.A., 2021. Construction 4.0: Industry 4.0 enabling technologies applied to improve workplace safety in construction. Research, Society and Development, 10(12), pp.e280101220280-e280101220280.

Kamari, M. and Ham, Y., 2022. AI-based risk assessment for construction site disaster preparedness through deep learning-based digital twinning. Automation in Construction, 134, p.104091.

Lalik, K. and Wątorek, F., 2021. Predictive maintenance neural control algorithm for defect detection of the power plants rotating machines using augmented reality goggles. Energies, 14(22), p.7632.

Lee, Y.S., Rashidi, A., Talei, A., Arashpour, M. and Pour Rahimian, F., 2022. Integration of deep learning and extended reality technologies in construction engineering and management: a mixed review method. Construction Innovation, 22(3), pp.671-701.

Makhataeva, Z. and Varol, H.A., 2020. Augmented reality for robotics: A review. Robotics, 9(2), p.21.

Martins, N.C., Marques, B., Alves, J., Araújo, T., Dias, P. and Santos, B.S., 2022. Augmented reality situated visualization in decision-making. Multimedia Tools and Applications, 81(11), pp.14749-14772.

Masood, T. and Egger, J., 2021. Augmented reality: focusing on photonics in industry 4.0. IEEE Journal of Selected Topics in Quantum Electronics, 27(6), pp.1-11.

Murugan, S., Sampathkumar, A., Kanaga Suba Raja, S., Ramesh, S., Manikandan, R. and Gupta, D., 2022. Autonomous vehicle assisted by heads up display (HUD) with augmented reality based on machine learning techniques. In Virtual and augmented reality for

automobile industry: Innovation vision and applications (pp. 45-64). Cham: Springer International Publishing.
Na, S., Heo, S., Han, S., Shin, Y. and Lee, M., 2022. Development of an artificial intelligence model to recognise construction waste by applying image data augmentation and transfer learning. Buildings, 12(2), p.175.
Nassereddine, H., Schranz, C., Bou Hatoum, M. and Urban, H., 2020. A comprehensive map for integrating augmented reality during the construction phase. In Creative Construction e-Conference 2020 (pp. 56-64). Budapest, Hungary: Budapest University of Technology and Economics.
Park, J., Chang, S., Lee, H. and Cho, Y.K., 2022. Inspection data exchange and visualization for building maintenance using AR-enabled BIM. In ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction (Vol. 39, pp. 483-490). IAARC Publications.
Sabeti, S., Shoghli, O., Baharani, M. and Tabkhi, H., 2021. Toward AI-enabled augmented reality to enhance the safety of highway work zones: Feasibility, requirements, and challenges. Advanced Engineering Informatics, 50, p.101429.
Sahu, C.K., Young, C. and Rai, R., 2021. Artificial intelligence (AI) in augmented reality (AR)-assisted manufacturing applications: a review. International Journal of Production Research, 59(16), pp.4903-4959.
Schranz, C., Urban, H. and Gerger, A., 2021. Potentials of Augmented Reality in a BIM based building submission process. Journal of Information Technology in Construction, 26.
Sharma, A., Mehtab, R., Mohan, S. and Mohd Shah, M.K., 2022. Augmented reality–an important aspect of Industry 4.0. Industrial Robot: the international journal of robotics research and application, 49(3), pp.428-441.
Taha, A., Qu, Q., Alex, S., Wang, P., Abbott, W.L. and Alkhateeb, A., 2021. Millimeter wave MIMO-based depth maps for wireless virtual and augmented reality. IEEE Access, 9, pp.48341-48363.
Taheri, H., Gonzalez Bocanegra, M. and Taheri, M., 2022. Artificial intelligence, machine learning and smart technologies for nondestructive evaluation. Sensors, 22(11), p.4055.
Tan, Y., Xu, W., Chen, P. and Zhang, S., 2024. Building defect inspection and data management using computer vision, augmented reality, and BIM technology. Automation in Construction, 160, p.105318.
Woo, J., Shin, S., Asutosh, A.T., Li, J. and Kibert, C.J., 2021. An overview of state-of-the-art technologies for data-driven construction. In Proceedings of the 18th International Conference on Computing in Civil and Building Engineering: ICCCBE 2020 (pp. 1323-1334). Springer International Publishing.
Zhang, Z., Wen, F., Sun, Z., Guo, X., He, T. and Lee, C., 2022. Artificial intelligence-enabled sensing technologies in the 5G/internet of things era: from virtual reality/augmented reality to the digital twin. Advanced Intelligent Systems, 4(7), p.2100228.
Zhou, B. and Güven, S., 2020. Fine-grained visual recognition in mobile augmented reality for technical support. IEEE Transactions on Visualization and Computer Graphics, 26(12), pp.3514-3523.