Machine-Learning Framework for Efficient Multi-Asset Rehabilitation Planning

by

Kareem Mostafa

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Civil Engineering

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner	NAME: Saiedeh Razavi Title: Associate Professor, Civil Engineering McMaster University
Supervisor	NAME: Tarek Hegazy Title: Professor, Civil and Environmental Engineering University of Waterloo
Internal Member	NAME: Chul Min Yeum Title: Assistant Professor, Civil and Environmental Engineering University of Waterloo
Internal Member	NAME: Mahesh Pandey Title: Professor, Civil and Environmental Engineering University of Waterloo
Internal-External Member	NAME: Mehrdad Pirnia Title: Graduate Attributes Lecturer, Management Sciences University of Waterloo

AUTHOR'S DECLARATION

Abstract

While smart cities are viewed as the way of the future, the infrastructure assets expected to support the different smart services are currently managed using frameworks that are outdated, subjective, and inefficient. Such inefficiencies have led to huge maintenance and rehabilitation backlogs that are far beyond the financial capabilities of cities, municipalities and large asset owners like school boards. For example, the cost to bring Ontario schools facilities to an acceptable level of service is estimated to be as high as $16 billion. Currently, most “smart asset initiatives” are geared towards building new assets and using sensors to get periodic info about their condition, with little thought given regarding the condition of existing assets. As such, there is a need to introduce a “smart rehabilitation” framework that answers the question “how to bring the current infrastructure assets up to speed to satisfy the needs of current and future generations?”.

To contribute to the overall vision of smart cities (*data-driven interconnected services*), the introduced framework uses machine learning and smart analytics to tackle three main functions of smart asset rehabilitation frameworks: (1) it automates the inspection and condition assessment processes by using convolutional neural networks (CNNs) to develop a machine learning system where defects can be automatically detected, classified, and quantified from images; (2) it uses data mining and clustering techniques to classify the assets according to their condition and need for repairs, and then uses optimization to select which assets are most worthy of immediate repairs subject to the existing funding constraints, thus enhancing the fund allocation phase by reducing its subjectivity; and (3) it uses novel computations, visualizations, and algorithms to facilitate cost-effective and fast-tracked delivery of the required rehabilitation works by considering them as units of a large repetitive project.

To verify the strengths and versatility of the model, the proposed framework is applied to built-up roofs of educational buildings such as schools and university campuses. First, images were collected from the University of Waterloo campus buildings to develop the image-based analysis module; a two-step CNN framework that can detect damages and classify them according to their type. Information from the image-based analysis were then combined with textual information related to building age and description and unsupervised learning was applied to develop the prioritization and fund allocation module. Results from this module are used as the inputs to an optimization procedure where the overall performance of the entire asset portfolio is maximized by selecting which buildings should undergo immediate repairs, given strict budgetary constraints. Finally, the selected rehabilitation works were scheduled as units in a large repetitive project for delivery planning. Accordingly, novel computations and algorithms were developed to create compact schedules with minimal gaps that comply with deadline constraints, and novel visualizations were introduced to showcase the crews movements and the timing of all tasks required in each unit.

The proposed framework offers powerful decision support features for a proposed smart rehabilitation layer to be included into the overall smart city vision. This framework deals with existing assets and provides objective assessments, cost-effective prioritization, and time-effective delivery plans. While this study used the case of built-up roofs as an example application, the framework is scalable towards other asset components as well as other assets in general. For example, components such as parking lots and concrete elements would rely heavily on the image-based inspection module, while other components such as HVAC systems would place more emphasis on the data analytics component, including more parameters related to different performance metrics as part of the analysis. Overall, this framework has the potential to revolutionize the multi-billion-dollar business of infrastructure renewal and provide cost effective decisions that save taxpayers' money on the long run.

Acknowledgements

First and foremost, I thank ALLAH (SW) our Lord, the most Gracious and the most Merciful, for giving me the strength, the ability, and the knowledge to complete this work.

I would like to express my appreciation and my gratitude to my supervisor, Dr. Tarek Hegazy, for his guidance, inspiration, and encouragement over the past several years. His efforts helped me grow not only as a better scholar but as a better person in general. It has been a great honor to work with him and to learn from his experience.

I would like to extend my sincere appreciation and gratitude to my committee members Prof. Saiedeh Razavi, Prof. Chul Min Yeum, Prof. Mehrdad Pirnia, and Prof. Mahesh Pandey for their insightful comments and their dedicated efforts to this study.

Additional thanks to the facility management team at the Toronto District School Board as well as the office of Plant operations in the University of Waterloo for their cooperation during the progress of this research.

Last but not least, I would like to thank my family, my friends, and my fellow colleagues in the construction research group for their support over the last few years. I would not have done it without you.

Table of Contents

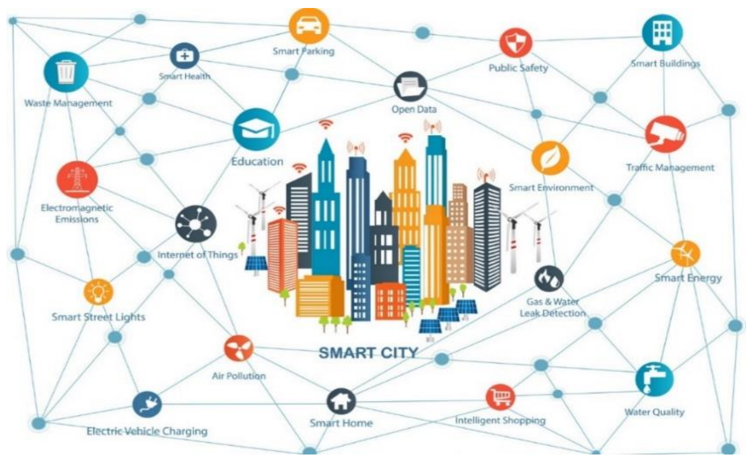
List of Figures

List of Tables

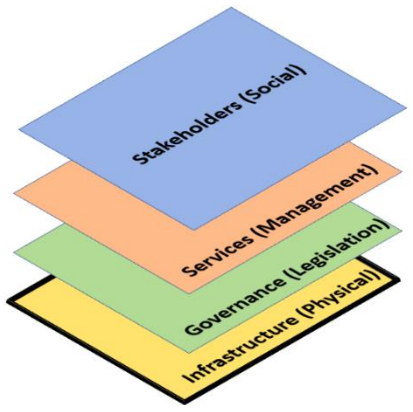
Chapter 1: Introduction

1.1 General

Smart cities are on the rise as an answer to challenges of limited resources, increasing population, and the need for technology-driven efficiency. It is estimated that the market size for smart cities will grow up to $820 billion by 2025 (Vuppuluri 2020) and generate up to $20 trillion in economic benefit worldwide (Challawalla et al. 2020). As shown in the typical illustration in Fig. 1.1, a smart city links a variety of individual smart services, including smart education, smart buildings, smart waste management, smart energy, and smart environment, (SmartCitiesWorld 2017; Patel 2019). Bawany and Shamsi (2015) provide a layered description of the various categories of services such as smart physical infrastructure; and smart governance (Fig. 1.1b). These services are interconnected through a core communication technology based on the Internet of Things (IoT).



(a) Individual smart components (Smart cities world 2017)



(b) Layered smart city services (Adapted from Bawany and Shamsi 2015)

Fig. 1.1: Components of a Smart City

As shown in Fig. 1.1b, the physical infrastructure (roads, bridges, pipelines, schools, hospitals, etc.) is the core foundation layer. As such, maintaining the city's physical assets is the key to sustain the city’s smart services (Smart Brantford 2019). To achieve that, Governance should be able to develop the necessary strategic and operational tools and procedures to automate and decide cost-effective intervention methods of rehabilitation.

1.2 Research Motivation

This research aims to develop a comprehensive smart rehabilitation framework, integrating modern technologies for building inspection, smart analytics of inspection data, and efficient scheduling of rehabilitation works to be delivered most efficiently. The research has been motivated by the following:

1.2.1 Smart City Requires Smart Rehabilitation for Existing Assets

The typical vision of a smart city (e.g., Fig. 1.1) focuses on building new infrastructure assets as smart assets (e.g., equipping them with sensors to periodically communicate their conditions). However, it overlooks the requirements of the large portion of existing assets that are old and require extensive rehabilitation work. Fig. 1.1a, for example, shows the physical assets (smart buildings, smart transportation, water quality, etc.) as isolated islands within the typical vision. As the world is becoming more and more digitally connected due to the increasing rate of technological advances and breakthroughs, municipal governments strive to accommodate those innovations and the accompanied clients' demands of higher levels of services from their cities. As mentioned earlier, current research work related to smart cities tends to overlook the maintenance and rehabilitation aspects. As such, there is a need to define the term "smart rehabilitation" as part of the greater term "smart asset management". Smart asset rehabilitation is aligned with the current vision of municipalities and governments in terms of creating a digital service strategy through data analytics, responsive operations, and intelligent infrastructure (City of Ottawa 2017).

1.2.2 Challenges in Condition Assessment of Existing Assets

Assessing the condition of existing assets serves as the background data repository based on which all asset rehabilitation decisions are made. Currently, however, most inspections to determine the conditions of assets are done manually. Such methods suffer from low productivity, subjectivity, and inconsistency as two inspectors might provide different reports for the same asset (Hoang et al. 2018). Inspection quality is also limited by the training and experience of the inspector. For each hour spent in the field for inspection, additional three hours are spent in the office to generate the reports (Abou Shaar 2012). As an example of how time-consuming inspection is (using current methods), in 2010 Ontario ministry of education issued a bid seeking a company to inspect a total of 4800 schools over a five-year period (MERX 2011). Because inspection is time-consuming, inspections tend to take place less frequently than desired. For example, schools in Ontario are typically inspected once every 3-5 years (Abou Shaar 2012). As such, the data based on which rehabilitation decisions are made are often outdated. Furthermore, smarter data collection allows for smarter analysis. For example, Ahluwalia and Hegazy (2010) used surveys to analyze roof defects and reported that less frequent and subjective inspection data puts many assets in the same condition category, which makes the prioritization and fund allocation processes harder than it would have been if such categorizations were based on timely and unbiased data. Hence, there is a need for automating the inspection process to save time and cost, be able to have more timely data, and make the process less subjective.

1.2.3 Potential of Machine Learning for Analytics and Asset Prioritization

A smart inspection and asset management system should be able to automatically analyze the data collected and come up with its own conclusion regarding the asset condition and the amount of rehabilitation needed. Such analysis can use methods such as deterioration modeling, deep learning networks, or others. Such smart analytics and data-driven based methodology aimed to digitize current inspection and performance reports may reveal more information that would change the way assets are valued. In addition to the economic benefits, this has social benefits as well in terms of bridging the social and political divide since automating the damage assessment provides for a “fairer” decision-making process because it eliminates the human bias when it comes to selecting which assets to receive the rehabilitation funds.

One way to make asset management and rehabilitation more automated and “smarter” is through the use of computer vision and image analysis techniques, as a cost-effective approach, to extract information from recorded images and videos. The most evident applications of computer vision can be seen in drones and autonomous vehicles. In essence, the same way unmanned vehicles use computer vision technologies to automatically detect pedestrians and other objects to avoid collisions, the proposed inspection framework should be able to use the same technology to automatically detect cracks and other defects and use this information for damage assessment in terms of severity and size.

1.2.4 Challenges in Developing Efficient Delivery Plans for Rehabilitation Works

Municipalities and governments are faced with various constraints when it comes to authorizing rehabilitation work. For example, there is an annual shortfall of $1 billion in order to keep schools of Ontario in good repair (Sachgau 2016). Furthermore, these repairs have to take place only during the summer vacation because of operational and weather constraints. Currently, Facility Management (FM) professionals treat rehabilitation work on a case-by-case basis assuming they happen in isolation.

Efficient delivery should be able to capitalize on similarities between tasks and aggregating them in a bigger work package with repetitive tasks to minimize costs. Thus, using repetitive scheduling techniques (Hegazy 2002) for the delivery of rehabilitation works provides opportunities for increased efficiency in terms of allowing crews to move uninterrupted to save time and cost. Hence, the need for utilizing repetitive scheduling techniques for the delivery of rehabilitation work becomes eminent, which is one of the components of the proposed research.

1.3 Research Objectives and Scope

The primary goal of this research is to establish "smart rehabilitation" as a major component of the smart asset management layer of smart cities. Specifically, this research utilizes machine learning tools, such as computer vision and data mining, and repetitive scheduling techniques to develop an automated framework for smart city rehabilitation. The framework includes different functions that perform efficient condition assessment, prioritization and fund allocation, and delivery planning of time-critical and cost-critical rehabilitation works. With a focus on rehabilitation management of built-up roofs as a case study, detailed objectives are as follows:

1. Clarify the challenges of asset rehabilitation by investigating the current practice of the different asset management phases (Inspection, prioritization, and delivery);
2. Based on data collected from the University of Waterloo buildings, use deep learning and computer vision techniques to detect damages, classify them according to their type, and quantify their sizes directly from roofing images, and develop an automated system to perform this function;
3. Combine the image assessment data with text-mining information to classify assets into categories according to their condition and need for rehabilitation, and develop an optimization model that assists asset management professionals in prioritizing the required rehabilitation events while satisfying budgetary constraints; and
4. Develop a scheduling framework that uses the defined rehabilitation work packages generated by the system in 3 above, treat them as a case of scattered repetitive works, and accordingly develop an efficient schedule that considers available resources and meets delivery constraints with least cost;

This research supports cities, municipalities, and other facility management departments such as school boards that are required to monitor the conditions of multiple buildings and perform the necessary rehabilitation works using limited budgets and/or within a limited timeframe. While the current research relied on data from TDSB, the proposed framework can be adapted to suit the needs of other infrastructure assets.

1.4 Research Methodology

The proposed research methodology to achieve the above research objectives are as follows:

1. Literature Review: Conduct an extensive review of existing asset management literature as well as current systems used in the industry. Special care will be given to areas related to inspection and damage assessment. In addition, conduct a review of computer vision and data mining techniques to identify which one is most suitable for developing the inspection and prioritization modules;
2. Data Collection and Analysis: Study past inspection reports from the Toronto District School Board (TDSB) to define key textual information that best describes building conditions (e.g., building age, description, etc.) and collect roofing images from the University of Waterloo buildings;
3. Image-Based Deep-Learning System: based on the results obtained from analyzing the pictorial data, a model will be built using python, a programming language, to automate the damage detection, classification, and quantification;
4. Automated Work Packaging: Working in tandem with the quantification submodule in item 3 above, RS means data, a database for cost estimation of construction and rehabilitation works, is used to create automated rehabilitation work packages that address all the poor and critical assets;
5. Text-mining system: Based on the results obtained from analyzing the textual data, as well as the results of the image-based system in item 3 above, a model is built to categorize the assets and rehabilitation events according to their criticality
6. Asset Portfolio Rehabilitation Optimization: Based on the criticality and work packaging information obtained from items 3-5, a simple optimization model is developed to prioritize the requested rehabilitation works to maximize the overall performance improvement of the asset portfolio while abiding by budgetary constraints;
7. Scattered Repetitive Scheduling: A system for efficient delivery planning of scattered repetitive work packages will be developed with streamlined computations and visualization;

8. Prototype Testing, and Validation: A prototype decision support system will then be developed. The prototype shall be tested using data from TDSB and University of Waterloo; and
9. Discuss the integration of the developed smart rehabilitation framework within the overall vision of smart cities that is adopted by municipalities.

1.5 Thesis Organization

The remainder of the thesis is organized as follows:

Chapter 2 presents a detailed literature review to highlight the drawbacks of existing inspection and condition assessment frameworks and the need for automation, as well as recent trends in machine learning and the different applications in analyzing textual and pictorial data. Unique characteristics of repetitive projects and drawbacks of current scheduling techniques are also discussed.

Chapter 3 holistically analyzes the key building component information existing in TDSB inspection reports and accordingly selects built-up roofs as the main building component to focus on for detailed study and analysis, before introducing the proposed roofing rehabilitation framework. Also, the data collection processes for the different data types (Textual: TDSB, Pictorial: University of Waterloo) are briefly introduced.

Chapter 4 studies the first component of the proposed framework, the image-based analysis model, in more detail. It goes through the different CNN model frameworks and architectures and discusses their capabilities in detecting, classifying, and quantifying the different roofing defects. The chapter also presents the real-life application and validation results of the image-based analysis module on University of Waterloo data.

Chapter 5 studies the second component of the proposed framework, the text mining framework, in more detail. It goes through the different algorithms adopted and the final model where the different algorithms are aggregated. The chapter also presents the real-life application and validation results of the text mining model using TDSB data

Chapter 6 presents the novel contributions towards scheduling scattered repetitive projects, which constitutes the final component of the proposed rehabilitation framework. The chapter describes the novel computations (designed interruptions and preventing schedule delays), heuristic algorithms (First-Come-First-Serve), and visualizations (duration-distance chart). The chapter highlights the advantages of the novel scheduling contribution in terms of robustness, explainability, and time and cost savings.

Chapter 7 summarizes the presented research works, highlights its contributions, and provides recommendations for future research

Chapter 2: Literature Review

2.1 General

This chapter first addresses the current state of building infrastructure as well as the introduction of smart cities, then provides a comprehensive overview of current inspection methodologies and asset management software packages. This is followed by an introduction of repetitive scheduling techniques and how they are more suitable for scheduling rehabilitation work than conventional scheduling methods such as the critical path method (CPM). Finally, a review of computer vision and deep learning techniques is presented, along with their current applications and research potential. The chapter concludes with a summary of the research gaps in the fields aforementioned, to be addressed in the proposed research (Chapter 3).

2.2 Smart and Sustainable cities

The concept of smart cities first emerged in the 1990s calling for utilizing new technologies to solve urban problems that are unsolvable using traditional planning approaches (Alawadhi et al. 2012). It then received a major boost when IBM proposed in 2009 their vision of smarter cities as a gateway for a sustainable future (Dirks and Keeling 2009). A smart city is a city that functions in an intelligent and sustainable way by integrating all its elements using modern technology to serve as one cohesive unit and effectively monitor its integrity.

Various researchers have taken interest in the term "smart cities" pursuing various objectives. In fact, the number of publications and projects related to smart cities has grown exponentially over the past decade since the term has been established (Camboim et al. 2019, Anand and Navio-Marco 2018). Literature focusing on smart cities concepts and frameworks depart from the notion that the people are the main driver of development rather than the mere technological advances (Albino et al., 2015; Hollands, 2008). Giffinger and Gudrun (2010) identified six main components for a smart city: smart governance, smart economy, smart mobility, smart people, smart environment, and smart living. Nam and Pardo (2011) categorized the main elements that comprise a smart city into three main categories: technology (hardware and software infrastructure), human (education, creativity, and diversity), and institution (policy and governance). Mohanty et al. (2016) provided an overview of the main components of a smart city highlighting that the city can still be considered "smart" even if it chose not to adopt all components and the choice regarding which components to adopt depends on multiple factors such as costs and available technology.

Literature highlighting technological advances focuses on developing new technology to enhance the urban environment (Meijer and Bolivar 2015). Allam and Dhuny (2019) analyzed the popularity of Artificial Intelligence (AI) and Big Data using Google trends and concluded that the interest in Big Data has grown significantly since 2011. Silva et al. (2018) proposed a layered framework of smart cities: A sensing layer for data collection, a data transmission layer (e.g. 3G, Wi-Fi, etc.), a data management layer that is concerned with data analysis and decision support, and an application layer that directly interacts with citizens. Examples include the "Big Data-enabled Smart Healthcare System Framework (BDHSF)" developed by Pramanik et al. (2017) where big data analytics, logistic support, and smart service-based architecture were utilized to achieve better quality and less costs of healthcare services. Oralhan et al. (2017) designed a waste container that can measure its capacity, temperature, and levels of carbon dioxide accumulated inside and utilize the Internet of Things (IoT) technologies to help calculate an effective waste collection route. Raja and Pang (2016) have developed robots that are capable of performing indoor inspections. While their works were aimed at proving that robots can perform this level of fine-grained inspection rather than performing a specific type of inspection, this can be considered a step towards fully automating indoor inspections in general, and maintenance inspections in specific.

In typical smart governance, smart rehabilitation could be viewed as inherently embedded within each asset type, e.g., within smart buildings, smart roads, etc. However, because rehabilitation cutting across mixed assets (e.g., roads and underlying pipes), it is mandated that smart rehabilitation is treated as a separate layer of governance. To emphasize this point of view, the vision for smart buildings and smart facility management (Fig. 2.1), for example, ignores (or at best hides) rehabilitation services and focuses on security, energy efficiency, etc. As such, smart rehabilitation should become an extra layer that connects the various city components: management of the performance level and longevity of the physical assets through legislation and governance. Yet this is also not expressed in any of the discussions related to smart city layers, neither introduced as a new layer nor incorporated into existing ones.

Examples of government-led examples of high-profile smart cities include Masdar in Abudhabi, Cyberjaya in Malaysia, and PlanIT Valley in Portugal. More exhaustive lists of smart city initiatives (especially within the European Union) can be found in the report issued by the European Parliament (Manville et al. 2014) as well as Collins et al. (2017). In Canada, Ottawa (2017) is taking the lead in spearheading detailed action plans for Smart Governance to include planning and maintaining physical infrastructure elements such as roads and bridges or underground systems. Such vision, however, lacks explicit or dedicated care for the largest portion of the assets which is old and requires special care for their continuous rehabilitation, capital renewal, and asset management. Similarly, Infrastructure Canada has organized a competition inviting all Canadian municipalities, governments, and indigenous communities to submit proposals that would leverage data and smart cities components to address current challenges (Infrastructure Canada 2019). None of the 20 proposals that qualified to the final round addressed the problem of innovative or smart asset rehabilitation. Hence, there is a need not only to define the term "smart rehabilitation and maintenance" and draw attention to it as one of the main components of smart cities, but also to develop techniques to improve rehabilitation decision making to save time and money and sustain the infrastructure services to the public.

Fig. 2.1: Key Components of Smart Facility Management (Advancer Global n.d)

The rise of various technological innovations such as big data analytics, the Internet of Things (IoT), and mobile internet access can foster collaboration among citizens. Cities will become "smarter" only with the right combination of technology, policy innovation, and civic engagement and collaboration (Anand and Navio-Marco 2018). Researchers are undertaking enormous efforts trying to investigate how smart city components can interact with one another (e.g. Ben Letaifa, 2015; Colldahl et al. 2013; Meijer and Bolivar 2015) but introducing smart asset management as one of the overarching smart concepts that can act as a connecting layer between smart assets and other smart city components (e.g. smart citizens as users of said assets) has not been addressed. Also, none of the global initiatives investigated and presented earlier aimed to tackle asset management issues. Hence, the proposed research aims to address both the conceptual and the technological gap related to smart asset management and smart rehabilitation and maintenance.

2.3 Asset Management for Civil Infrastructure

Major components of a well-functioning city are the civil infrastructure assets as they impact every aspect of the residents' lives and the health and integrity of said assets greatly affect the city's potential for economic and social growth. Factors such as age, harsh environmental conditions, and consistent intensive usage put them under duress which leads to their deterioration. Typically, these factors are counteracted by regular inspection and maintenance, but this is not always the case due to limited funding. Accumulation of shortfalls produced by consistent funding limitations creates a backlog of needed repairs. Such backlogs can be as big as $17.2 billion as in the case of Canadian universities (Johnson 2020) or even $16.3 billion as in the case of Ontario schools (Rushowy 2019). According to the Toronto District School Board (TDSB), over 20,000 repairs for 583 schools are required with a total value of $3.7 billion as of July 2021 (TDSB 2021). However, provincial funding is only $300 million per year. Similarly, in the USA, the most recent ASCE infrastructure report (Fig. 2.2) gave schools a grade of D+. Furthermore, the report states that extra $380 billion are needed between now and 2029 to keep schools at an acceptable working condition (ASCE 2021) as more than half of the schools need to update and/or replace their building systems (ASCE 2021).

To reduce the backlog, effective management of assets is essential to ensure adequate and long-term serviceability. However, this is not always an easy task due to budget limitations and operational limitations in terms of minimizing downtime due to maintenance and repairs. Hence, asset management systems were introduced to help managers find the optimal timing and methodology for repairs to maximize the value of the allocated budgets (Elhakeem and Hegazy 2010). Asset management systems involve both strategic and operational functions to help organizations perform capital renewal, rehabilitation, and upgrades for their inventory. Ideally, asset management systems incorporate the following functions: (1) performance assessment through inspection, identification of defects, and evaluation of the level of service; (2) deterioration modeling to predict the changes in asset performance over time; (3) analyzing and selecting the most appropriate renewal type (e.g., minor, major, or full replacement); (4) studying the life cycle cost of the asset to enhance the decision making process and extend the asset's lifespan; (5) ranking the assets according to performance priorities (e.g., condition, importance, etc.) and allocating rehabilitation funds based on such ranking; and (6) implementing the decision taken by the previous steps and assessing the condition of the asset post-rehabilitation. Those six functions are displayed in Fig. 2.3.

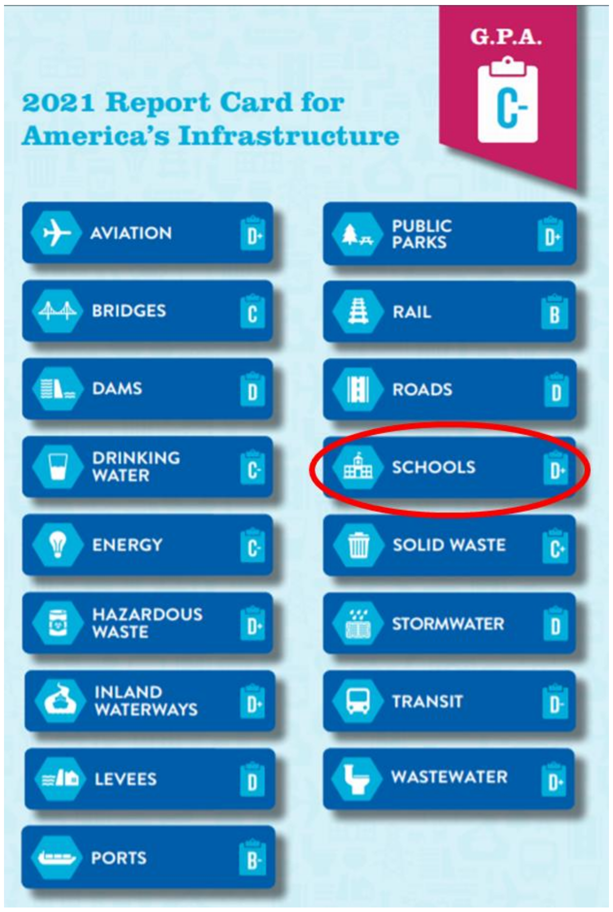


Fig. 2.2: ASCE 2021 Infrastructure Report

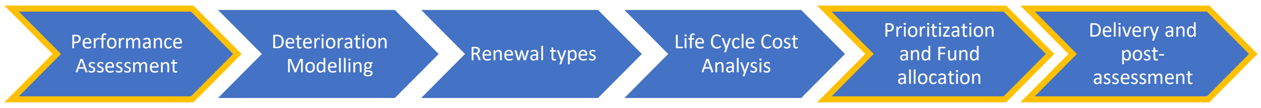


Fig. 2.3: Main Asset Management Functions (Adapted from Abdel-Monem and Ali 2010)

This research aims to enhance asset management frameworks by addressing three main phases of smart city asset management (highlighted in Fig. 2.3): performance assessment through implementing computer vision and Artificial intelligence techniques to enhance and automate the inspection process, prioritization and fund allocation through the use of novel data mining and optimization techniques, and the delivery phase by utilizing scheduling techniques that are better suited to these type of works (repetitive scheduling techniques).

2.3.1 Inspection Research

Currently, most inspections are done visually using semi-automated methods that are time-consuming. The Facility Management department (FM) sends inspectors to assess the building conditions and file reports estimating any required maintenance work. Visual inspections are widely used for preliminary and regular inspections because of their effectiveness in detecting external defects such as cracks and spalling (Omar et al. 2017). This is of extreme benefit as many degradation conditions often exhibit visual symptoms. Visual inspections aim to ensure the integrity of a structure by looking at its critical components and note any visual damages or changes that warrant further attention (Sweeny and Unsworth 2010). However, subjectivity is inevitable in visual inspections as different inspectors may evaluate the same structure differently (Dawood et al. 2018). Furthermore, some parts of the structure are not accessible for inspectors, making an overall assessment sometimes impractical (Dawood et al. 2018). Once the reports are submitted, the FM has to manually price, prioritize, schedule, and allocate resources to act on the reports received. This process is lengthy, subjective, and error-prone. Therefore, manually administering hundreds of these reports proves to be resource-consuming, if not problematic, for FMs given the tight budgets and time frames. An inspection site visit typically takes 4 hours to be completed and for each hour spent in the field for inspection, additional three hours are spent in the office to generate the reports (Abou Shaar 2012). As an example of how time-consuming inspection is (using current methods) Ontario ministry of education has issued a bid in 2010 seeking a company to inspect a total of 4800 schools over a five-year period (MERX 2011). In addition, the inspection results are not always consistent and depend on the inspectors' training and experience levels. Despite those shortcomings, manual visual inspection still proves to be to most suitable inspection approach for most building components (Elhakeem and Hegazy 2010).

Errors and gaps produced by manual assessments not only waste time and money, but can also lead to catastrophic events, such as the collapse of the I-35W highway bridge in Minneapolis, MN which caused 13 deaths and 145 injuries (Koch et al. 2015). One of the most recent events is the failure of Oroville dam spillways which took place in February 2017, forcing more than 188,000 Californians to evacuate and creating a 45-foot deep, 300-foot wide, and 500-foot long hole in the ground (Graham 2017). The dam was not on the California governor's "wish list"; a list of $100 billion worth of key assets targeted for investment for rehabilitation or construction purposes (CNBC 2017). The forensic report mentioned that the main cause of failure, vulnerabilities in the chute slab, was not identified in any of the inspections despite their frequency, rigorousness, and the fact that they are undertaken by different investigators representing different parties and following different guidelines (France et al. 2018). In Italy, while inspections were done properly and the Italian government was aware of the fact that the metal cables of a highway bridge in Genoa were corroded which reduced the bridge's strength by 20%, no action was taken to reduce the loads on the bridge (e.g. limit traffic, ban heavy trucks, etc.) which has led to its collapse killing 43 people and forcing 600 others to evacuate (Associated Press 2018).

Based on the discussion above, asset management, in its current form, has its huge challenges. Acquiring the condition of assets using traditional inspection means is time-consuming, less accurate, and subjective. As such, many assets end up in the same category in terms of condition, and this creates a large problem in allocating rehabilitation funds to the most deserving assets. Rehabilitation funds, therefore, are often spent by doing "some for all" (doing minor work for multiple assets) or "all for some" (doing all the work required for a single or a small number of assets) (Anand and Navio-Marco 2018). The first approach spreads the available resources too thinly to the point that it might create an illusion of slow progress (Anand and Navio-Marco 2018). On the other hand, the second approach can create a sense of inequality (Anand and Navio-Marco 2018). For this reason, Smart Asset Management Systems that focus on rehabilitation, which is missing in existing smart governance frameworks, become an essential component of smart cities and need to include: smart data collection and smart delivery of rehabilitation work to avoid service disruption to the public.

In general, existing inspection and asset management systems have little capabilities regarding how to deal with images. As seen in Table 2.1, typical inspection software focuses on report creation more than analysis, with none of the software packages can automatically detect or quantify defects. Images are merely included within the automated report with no semantic information automatically extracted. At best, some software packages allow for manually adding markups and annotations on the provided images (Table 2.1). Some asset management software can offer more services related to extracting information from images. They can store it as part of the asset data portfolio (VFA 2021), time and location stamp it (EZMaxMobile 2021), allow for manual highlighting and markups (Home Inspector Pro 2021), or even integrate it with a BIM model (ARCHIBUS 2021), but the building condition will have to be inputted by the user to be later on extrapolated using other information such as building age and undertaken repair works, which are also user inputs. In addition, only one software among the ones featured in Table 2.1 tries to address the delivery phase of the required rehabilitation work. Therefore, the proposed research deals with these two areas in particular: image-based analysis for automated inspection; and smart delivery of rehabilitation work.

Section 2.4 provides an overview of the state-of-the-art in the areas of the latest technologies in image analysis that can revolutionize the detection and classification of defects in different domains (which can be utilized to improve inspection); and data mining which can upgrade the prioritization and fund allocation decision-making process.

Table 2.1: Inspection Applications Feature Comparison

		Home Inspector Pro	Horizon	Inspectheck	Spectora	Jobber	HappyCo	HomeGauge	HomInspect	ReportHost	Chapps	Link Inspect Pro	zInspector
Visual Tools	Mark on Pictures	√	√		√		√	√	√	√	√	√	√
	Take Pictures		√	√	√		√	√	√	√	√	√	√
Interface	Assign Jobs to Inspectors		√		√	√	√				√	√	
	Hierarchy & Reporting Levels				√						√	√	√
	Condition Assessment			√	√		√						
	Condition Prediction						√						
	Custom Report Generation	√	√		√			√		√	√	√	√
	Per-Component Condition History											√	
Deep learning & Automation	Automated Defect recognition												
	Automated Quantity takeoff												
	Automated Prioritization & estimation							√					
	Automated scheduling					√							

2.3.2 Decision Making for Fund Allocation

Efficient asset management requires frequent monitoring and inspection of all assets and individual components to ensure their physical and functional fitness and to identify the items that are most worthy of the limited funds for capital renewal. However, the data in inspection reports often lack granularity and end up assigning critical priority to many more assets than can be funded (Ahluwalia and Hegazy 2010). This forces asset managers to rely on their subjective judgment for asset prioritization and limited-fund allocation. Capital renewal challenges are exacerbated for organizations that administer a large number of facilities, such as the Toronto District School Board (TDSB) in Canada, which owns nearly 600 schools, a sizable portion of which are older than 40 years old. To upgrade school conditions and reduce the growing renewal backlog ($4.2 billion in 2021), the provincial government provides yearly capital renewal funds to school boards. In the 2020-2021 budget, the TDSB was allocated $312 million for school renewals (TDSB 2021). However, it remains a challenge every year on how to allocate the available budget to the most deserving assets. Therefore, it has become even more critical for upper management to appropriately and objectively allocate limited funds to address the most pressing needs.

Current fund allocation decisions are typically analyzed using spreadsheets. A multi-year capital plan is then put into place where management decides which assets to repair immediately and which to push back for future years (Mostafa et al. 2021). Thus, this process would benefit greatly from an objective tool that would make use of the available optimization techniques to facilitate a simpler and faster fund allocation method with a good level of accuracy.

The low level of granularity and the inability to efficiently utilize inspection data has led to catastrophic events in some situations. Examples include the failure of the Oroville dam spillways explained in 2.3.1, as well as the failure of the Morandi bridge in 2018 that claimed the lives of 43 people in Italy (Piangiani 2020). As such, there is a need to develop a smart data-driven system that can provide timely, detailed, and unbiased insights on how different assets should be assessed and ranked in terms of rehabilitation/renewal needs. With limited funds, there have been calls that governments should adopt a “Moneyball” approach: a data-driven investment approach similar to what Oakland A’s manager Billy Beane used to build a top baseball team on a limited budget (Adriaens 2019). Finally, there is a need to find an objective way to answer the question “Among the many assets of similar inspection results, which ones are most deserving of renewal funds?”.

2.3.3 Repetitive Scheduling: Delivery of Asset Rehabilitation Works

Repetitive projects, by definition, consist of a group of activities that are repeated over multiple units. Repetitive projects can be linear, such as pipelines and roads, vertical, such as high-rise buildings, or scattered, such as multiple housing projects (Fig. 2.4). One key aspect for efficient scheduling of repetitive projects is allowing crews to perform their works and move from one unit to the other with minimal interference so that the crews develop a learning momentum, saving time and cost and benefiting from the economy of scale. As such, rehabilitation work can be considered under the umbrella of scattered repetitive projects. Scheduling those projects is a challenging task because the works are repetitive in nature but take place in diverse locations.

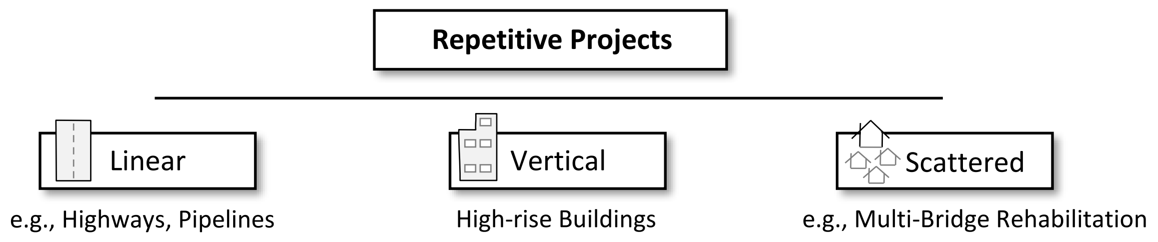


Fig. 2.4: Types of Repetitive Projects

2.3.3.1 Scheduling for Repetitive Projects

Scheduling repetitive projects is a challenging process and commonly used techniques, such as CPM, are inadequate for multiple reasons. In the case of repetitive projects, CPM networks become more complicated as they include copies of the same activities but assigned to different units. This makes the schedule difficult to understand or visualize (Su and Lucko, 2016). For repetitive projects, CPM plans each activity directly after the conclusion of its predecessor overlooking the importance of having the crews maintain work continuity while moving from one unit to another. For example, faster crews would be idle for a period of time till all predecessor activities that employ slower crews are complete. To address the CPM drawbacks, techniques have been developed in an attempt to synchronize resources, maintain work continuity, incorporate non-repetitive tasks within repetitive projects, respect project deadlines, and account for learning curve effects. Examples of the developed techniques include the Line Of Balance (LOB) (Arditi and Albulak 1986); the Linear Scheduling Model (Harmelink and Rowings 1998); and the Repetitive Scheduling Method (Harris and Ioannou 1998). Most repetitive scheduling methods incorporate the CPM network analysis to consider the logical relationship within each repetitive unit, and among the repetitive units as well (Hegazy 2002; Suhail and Neale 1994).

One of the key advantages of repetitive scheduling methods is their ability to show the large information about a repetitive schedule in a legible manner. The Line of balance (LOB), for example, plots the activities on a time vs. units axes as opposed to the time vs. activities axes used in bar charts (Fig. 2.5). In the figure, a total of 15 activities (3 activities repeated over 5 units) are shown in a relatively small chart area. Furthermore, LOB charts provide information about crew assignments and delivery rates, a feature that is not available in CPM. For example, Fig. 2.20 shows that activities A and B are each performed by one crew that moves from one unit to another at a certain rate (slope of the line). Activity C, on the other hand, is performed by three crews; the first crew moves from unit 1 to unit 4, the second crew moves from unit 2 to unit 5, and the third crew is only assigned to unit 3.

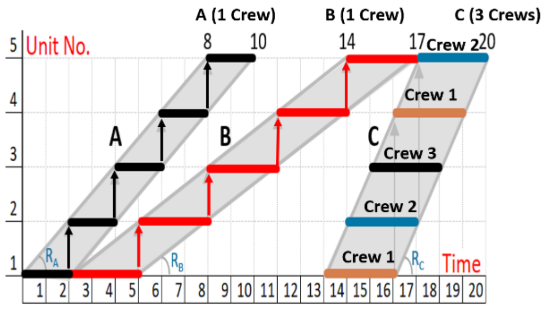


Fig. 2.5: LOB Schedule Representation of Three Activities along 5 Units

The first step in existing CPM/LOB calculations is to use conventional CPM formulae to calculate the time needed to finish one unit (T1). Then, since all the units (N) need to be completed before the deadline (DL), the rate of delivery of the units (R) is calculated using Equation 2.1. Equation 2.1 incorporates the total float of the activity (TF) as an adjustment factor that reduces the delivery rate of non-critical activities and thus requires fewer crews. Accordingly, the number of required crews is calculated based on the required rate (R) and the duration of the activity (D) using Equation 2.2, and then the rate of delivery is adjusted based on the new number of crews after rounding (because having fractions of a crew is impractical) through Equation 2.3. An example can be seen in Fig. 2.6.

[2.1] Task i desired rate $(R_i) = \frac{N-1}{(DL - T_1 + TF_i)}$

[2.2] Task necessary crews (C_i) = Roundup $(D_i \times R_i)$

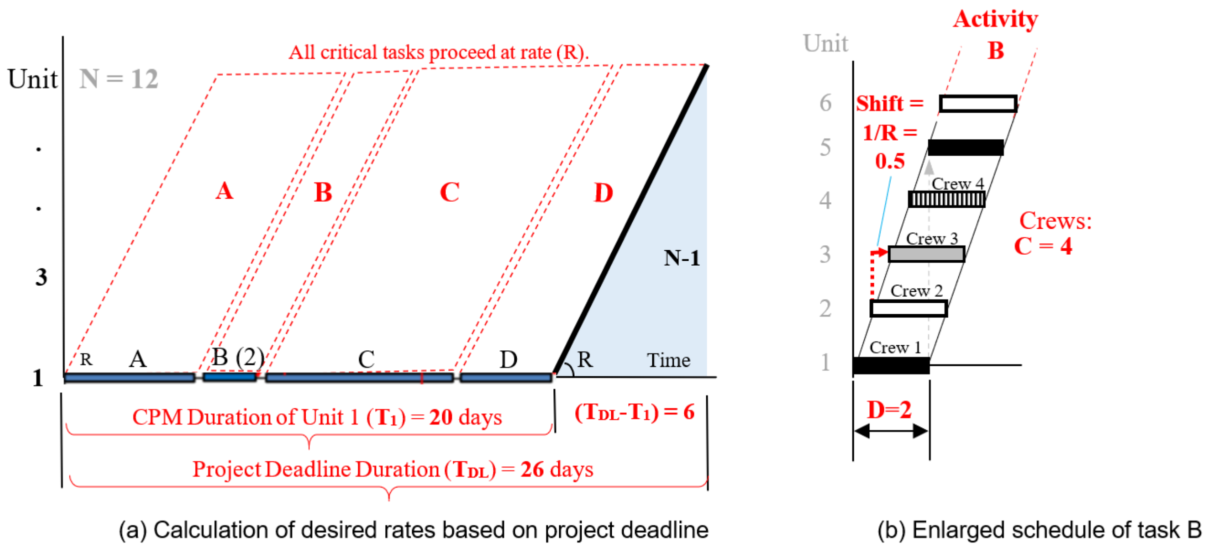
[2.3] Task actual rate $(R_i) = C_i / D_i$

In an effort to provide a more practical crew assignment, a new derivation of the needed crews has been developed for the more practical case of using parallel crews. The crews are arranged into *S* cycles of *C* crews to achieve the required delivery rate using equations 2.4-2.6. These equations can be seen in action in the example provided in Fig. 2.7, where the final arrangement of this task is 3 crews engaged in 4 cycles to complete all units.

[2.4] Initial cycles S_i of C_i crews = $\frac{(DL - T_1)}{D_i} + 1$

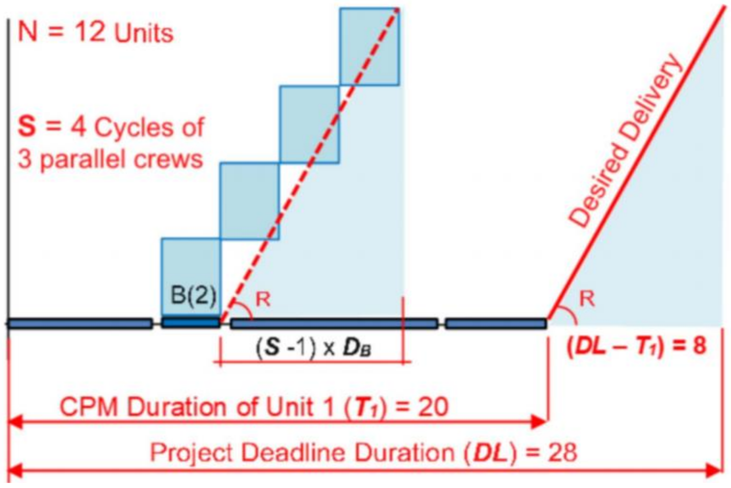
[2.5] No. of Crews C_i = Roundup (N / S_i); $1 \leq C_i \leq N$ & $C_i \leq$ Crew-Limit$_i$

[2.6] Actual Cycles S_i = Roundup (N / C_i)



Unit
N = 12
All critical tasks proceed at rate (R).
A
B
C
D
N-1
3
1
R
A
B (2)
C
D
R
Time
CPM Duration of Unit 1 (T1) = 20 days
(TDL-T1) = 6
Project Deadline Duration (TDL) = 26 days
(a) Calculation of desired rates based on project deadline
Activity B
Unit
6
5
4
3
2
1
Shift = 1/R = 0.5
Crew 4
Crew 3
Crew 2
Crew 1
Crews: C = 4
D=2
(b) Enlarged schedule of task B

Fig. 2.6: CPM/LOB Analysis of the Tasks' Required Shifted Crews to Meet the Deadline



N = 12 Units
S = 4 Cycles of 3 parallel crews
B(2)
R
(S -1) x DB
Desired Delivery
R
(DL – T1) = 8
CPM Duration of Unit 1 (T1) = 20
Project Deadline Duration (DL) = 28

Fig. 2.7: Modified CPM-LOB analysis of the required parallel crews.

To schedule a successor task that may have different crews/durations from its predecessor, researchers (e.g., Hegazy and Kamarah 2008; Laramee 1983) use a unique process of initially drawing the successor task starting at some time in the future. Afterward, a proper shift time is calculated to bring the task back to immediately follow the predecessor (Fig. 2.8). This approach is referred to as the "**Delta-Shift**" approach. As seen in Fig. 2.8, the delta-shift approach is effective in the case of scheduling tasks with non-identical units.

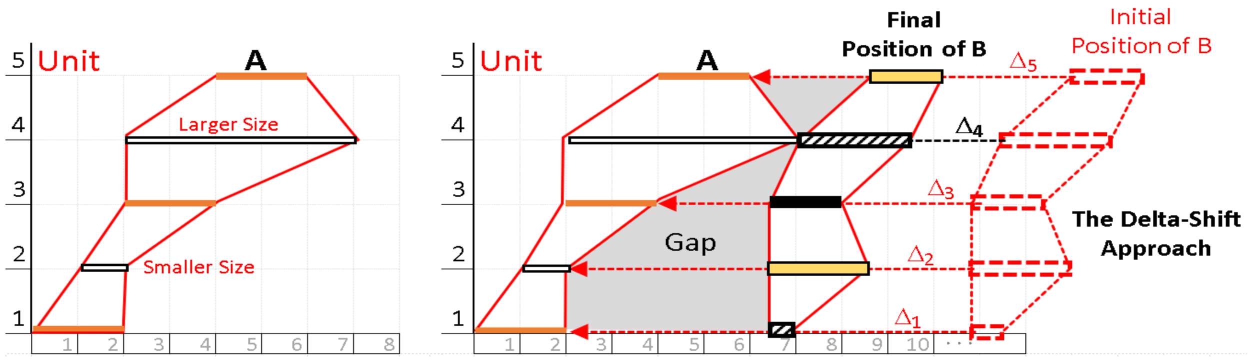


Fig. 2.8: Using the Delta-Shift Approach to Schedule non-Identical Units

Further research and development in repetitive scheduling have been taking place since the late 1990s with a particular focus on schedule optimization. Hegazy and Wasef (2001) developed a model that would integrate CPM and LOB techniques and use genetic algorithms to determine the optimum combination of construction methods, number of crews, and interruptions for each repetitive activity that would minimize the total project cost (direct and indirect costs, interruption costs, liquidated damages). Hyari and Elrayes (2006) developed a multi-objective optimization model to minimize project duration while maximizing work continuity. Derham (2008) developed a multi-objective genetic algorithm-based model to minimize both project cost and duration. Long and Ohsato (2009) developed a multi-objective model for minimizing project cost and/or duration for repetitive schedules. Ali and Elazouni (2009) integrated a CPM/LOB model with a cash flow model to optimize the project cash flow and generate financially feasible schedules. Agrama (2012) presented a multi-objective genetic optimization model able to minimize the project duration, work interruption, and the number of crews. Aziz (2013) developed a model that would optimize the tender offer for a repetitive project, taking into account schedule objectives (minimizing cost and duration) while maximizing the project's net present value. Dolabi (2014) presented two heuristic algorithms to achieve optimal crew formations so the project would meet a certain deadline, but those algorithms are only valid if the activities are serial (finish to start relationships with only one predecessor per activity). Huang et al. (2016) used genetic algorithms to solve multimode time-cost-tradeoff problems considering soft logic. Zou et al. (2017) presented a mixed-integer linear programming model for solving deadline satisfaction problems in LOB scheduling. Altuwaim and Elrayes (2018) used a two-step approach to develop an optimization model that would minimize project duration as well as interruptions.

Despite all those efforts, repetitive schedules still exhibit deadline violations even when the necessary computations are applied. This is because of multiple reasons such as rounding of crews, rounding of start times, crew availability limitations, or other reasons that change the geometry of the activity/crew assignment and thus introduces schedule gaps that lead to project duration extensions.

2.3.3.2 Scheduling for Scattered Repetitive Projects

Scattered projects are projects whose units are not in a single location. Hence, rehabilitation work can fall under this category as the assets where the rehabilitation work is taking place are present in multiple locations. Scattered projects are the most challenging to schedule due to a variety of reasons. First of all, not being bound by a single geographical location means that the work in each site is independently affected by its local conditions such as weather and traffic. Therefore, work at a given site should be scheduled when the site conditions allow for maximum productivity. Furthermore, project managers needed to have the flexibility in changing the sequence of assigned sites and not being tied to a single sequence that has to govern all activities, allowing to take into consideration multiple factors such as transportation costs and interruption to facilities operations. For example, the electrical crew may proceed in a different sequence than the HVAC crew. Having different working sequences makes plotting a scattered repetitive schedule problematic as can be seen in Fig. 2.9. Activity A is using one crew that moves in the order 2-6-7-1-3-5-4 (map on the left-hand side) while activity B uses two crews that move in the sequences 2-6-7-1, and 2-5-4, respectively (map on the right-hand side). With a schedule with many activities, it may not be possible to define the correct order of units on the vertical axis that makes the schedule readable.

To resolve this issue, Kamarah (2019) proposed a generic schedule representation where the site index is demonstrated on the activity bars instead of on a fixed axis (Fig. 2.10). Among the attempt to address the challenges of scattered repetitive scheduling, Hegazy et al. (2004) presented a genetic-algorithm-based scheduling model for efficient scheduling and resource optimization of scattered repetitive projects. That model was further developed by Kamarah (2019) into a computer prototype that automates the scheduling, control, and cost optimization of scattered repetitive projects.

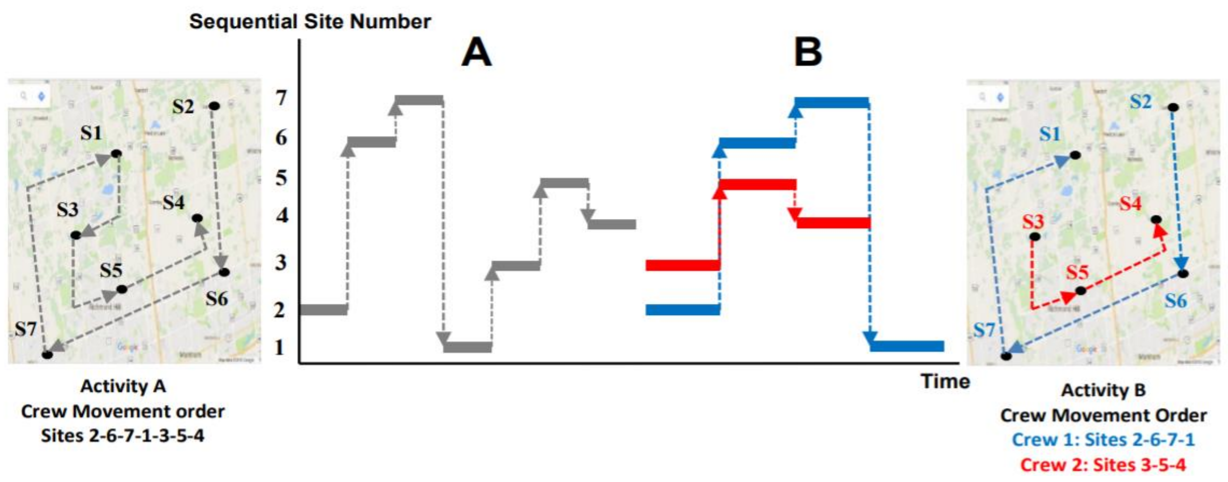


Fig. 2.9: Traditional LOB for a Scattered Repetitive Project (Kamarah 2019)

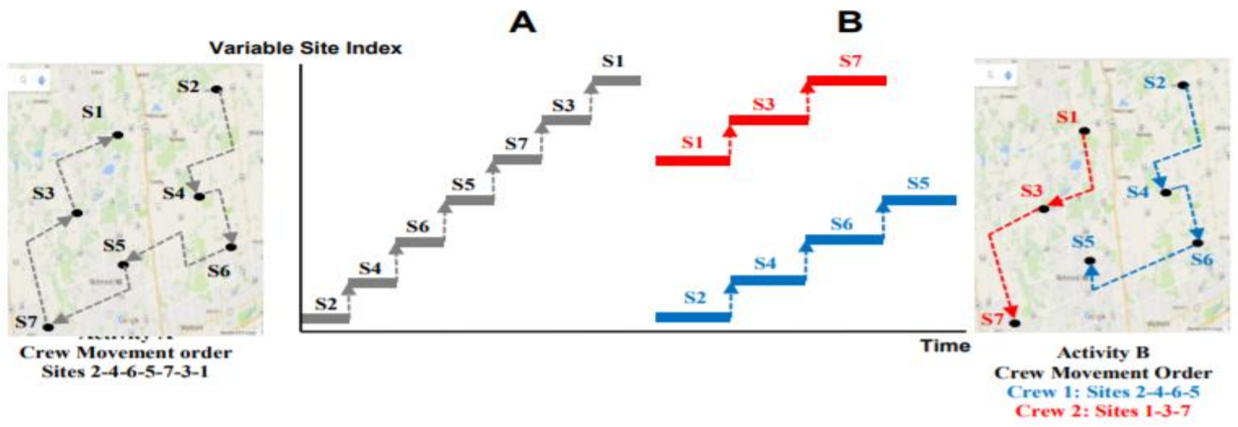


Fig. 2.10: Scattered Repetitive Schedule with Variable Site Index (Kamarah 2019)

Repetitive scheduling calculations require prior knowledge of the number of units, maximum available crews, as well as the project deadline. This poses a challenge when it is applied to rehabilitation work because the number of units is sometimes not known. As mentioned in chapter 1, 22,000 repairs are required in over 500 schools in Toronto and these repairs can only take place in the summer period when the schools are not active. Hence, there is a “packaging” problem in terms of how many, and which, schools can be fixed in this limited period of time, before solving the conventional scattered repetitive scheduling problem in terms of determining the optimal ordering and crew assignments. If we assumed that the scattered repetitive scheduling is a traveling salesman problem, where the objective is finding the least expensive routes to visit all states, then applying scattered repetitive scheduling to rehabilitation work imposes another problem of which states to visit in the first place. Furthermore, in municipalities, the resource limit for this type of work might not necessarily be limited to the number of maintenance crews doing the work (because the works can be subcontracted), but rather the number of in-house inspectors that oversee and inspect the work.

2.4 Advanced Analytics

The recent advances in artificial intelligence, computer vision, deep learning, and other data analysis technologies have allowed to automate many engineering tasks. This has allowed engineering professionals to acquire and analyze more up-to-date data to make decisions more accurately, which has led to cost and time reductions. Such advances allowed not only to analyze numerical data but also extract meaningful information directly from text and photos. However, the construction industry has been deemed to be relatively conservative in terms of adopting data-driven technology innovations to improve safety and productivity (Busta 2016). The construction industry is currently one of the least digitized industries in the world according to MGI’s digitization index (Manyika et al. 2016). For example, it is estimated that current site managers consume almost half their time manually collecting and processing progress monitoring data before making a decision (Deng et al. 2020). Hence there is a clear need for the use of artificial intelligence and advanced analytics to achieve maximum efficiency.

2.4.1 Computer Vision and Image Analysis Techniques

Computer vision and image-based learning techniques allow for computers to automatically analyze visual data such as images. The reason why computer vision has attracted various researchers in multiple other fields is primarily because images are easy to collect in a non-intrusive manner and they are often readily available (e.g. security CCTV cameras). The benefits of using image-based learning techniques are that they reduce the human subjectivity and time for manual inspections, and being able to inspect locations that are inaccessible by human inspectors. For those reasons, many state highway agencies are replacing manual surveys with automated systems that can collect high-resolution images and are able to detect cracks as small as 1-mm long (Wang et al. 2015).

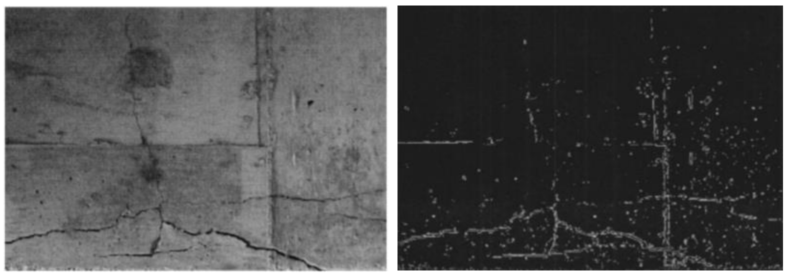
2.4.1.1 Edge Detection

Edge detection refers to the use of special filters for the purpose of detecting edges in an image (such as a crack) so they can be easily identified and located. Edges are detected based on discontinuities in image color and/or brightness. Points where such discontinuities occur are identified as edges. Cracks in a 2-dimensional image are classified as edges, and therefore existing edge detection algorithms can be used for crack detection purposes (Dorafshan et al. 2018). Further techniques can combine those points to form straight lines as well as identify corners based on the intersections of the lines detected earlier. Examples of common edge detection filters are in Table 2.2.

Table 2.2: Commonly Used Edge Detection Filters

Filter name	Composition
Sobel	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = C_x; \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = C_y$
Prewitt	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = G_x \quad ; \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = G_y$
Roberts	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} = R_x; \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = R_y$
Laplacian of Gaussian (LoG)	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

One of the most commonly used edge detectors is the Canny edge detector (Fig 2.11). It starts by convolving the original image (Fig. 2.11a) with a spatial mask (the Sobel mask is the one most commonly used) producing a first-order partial derivative for the pixel at the center of the mask in both the x (Cx) and the y (Cy) directions (Abdel-Qader et al. 2003). Two thresholds, a minimum and a maximum, are then set. If the pixel value is above the maximum threshold, it is considered a “strong edge”. If it is below the minimum threshold, it is not an edge. If it is between the two thresholds, it is considered an edge only if it was connected to a strong edge. The end result is a binary image that includes edges only (Fig. 2.11b).



a) Original Image b) Canny Image

Fig. 2.11: Canny Edge Detection (Abdelqader et al. 2003)

Dorafshan et al. (2018) compared the performance of six different edge detection filters (Roberts, Prewitt, Sobel, Laplacian of Gaussian (LOG), Butterworth, and Gaussian filter) and an AlexNet-based Deep Convolutional Neural Network (DCNN) by applying them to a 100-image (3402 sub-images) dataset of concrete panels for crack detection purposes. An example comparing the performance of the edge detection filters is shown in Fig. 2.12 where the filters (Fig. 2.12c-h) are trying to detect the crack present in Fig. 2.12a, Fig. 2.12b represents the ground truth (i.e. perfect detection). The experiments have shown that the LoG filter was the most efficient in terms of both accuracy and computation time. However, using DCNNs is more optimal, detecting finer cracks with higher accuracy.

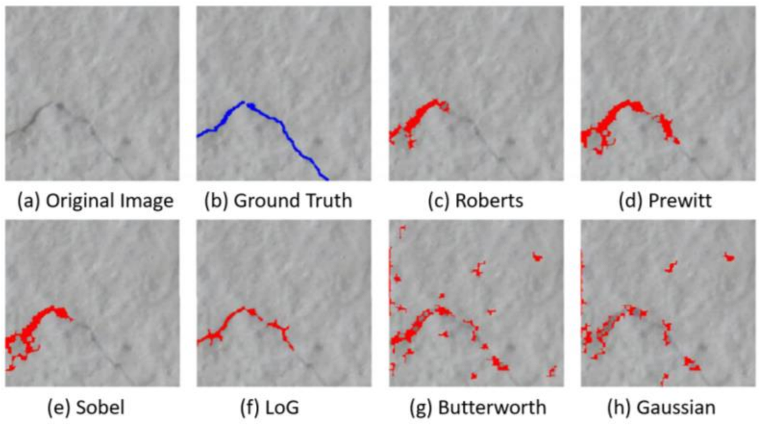


Fig. 2.12: Various Edge Detectors Performance on a Sample 0.02mm crack (Dorafshan et al. 2018)

2.4.1.2 Object Detection and Classification

As the name suggests, these techniques aim to detect objects of interest from images and videos as well as classify them into predefined categories (e.g., detecting a crack). Intuitively, to be able to classify objects through images, first, there is a need to detect or identify them. Such identification typically takes place by recognizing the objects' distinctive features (e.g. all circles are round). From those facts, there are many techniques used to achieve object detection and classification and often it requires combining them for improved results. Some of the common techniques are:

Feature Extraction Algorithms: a feature is defined as a function of one or more measurements that quantifies some significant characteristics of the object (Choras 2007). Feature extraction algorithms aim to use such descriptive features to detect objects of interest inside more cluttered scenes. An example is in Fig. 2.13 where an object (a Tim Hortons Gift Card) is being detected inside a larger scene. Available feature extraction and template matching algorithms include Scale Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF, used in Fig. 2.13), and others.

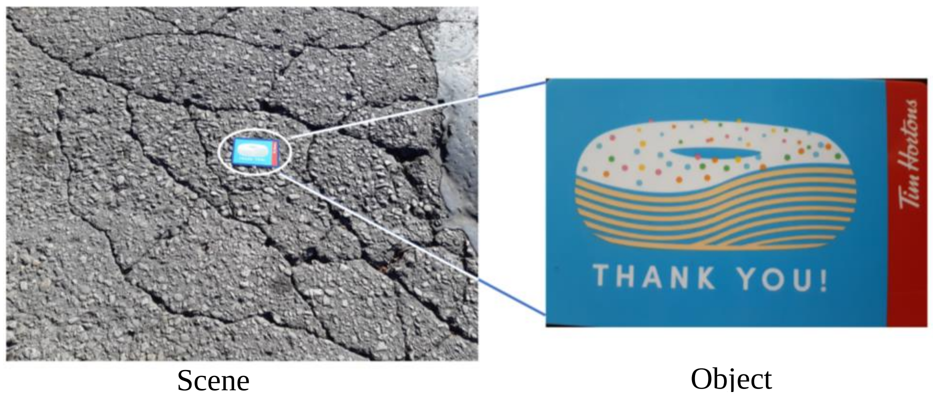


Fig. 2.13: Example of the Use of Speeded Up Robust Features (SURF) Feature Detection Algorithm

Histogram of Oriented Gradients (HOG): A feature descriptor of images by a set of local histograms. The idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions (Dalal and Triggs 2005). Hence, the image is divided into cells of a predefined size, then the occurrences of gradient orientation in each cell are counted to build the descriptor vector. Finally, normalization is performed to regulate the variability in the image (Suard et al. 2006). HOG pays huge attention to the shape of the detected object. Hence, Azhar et al. (2016) considered the use of HOG for pothole detection as potholes have no fixed shapes. They developed an automated pothole detection system that combined HOG with a Naïve-Bayes classifier achieving 90% accuracy on a 120-image dataset.

Support Vector Machines (SVM): While not used exclusively for computer vision purposes, SVM is one of the most powerful binary classification techniques (Seong et al. 2017). The SVM classifier aims to find an optimal hyperplane that separates samples into two classes (Aylien 2016, example in Fig. 2.14). In essence, SVM can be thought of as a high-dimension discriminant analysis, where the objective is finding the thresholding function that achieves the best binary classification of data based on their parameters.

Hyperplane

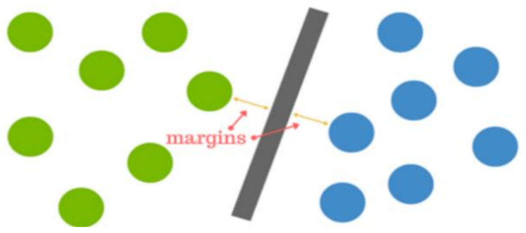


Fig. 2.14: Support Vector Machine (Aylien 2016)

Convolutional Neural Networks (CNNs): CNN is a type of artificial neural network (based on the human brain structure) that is inspired by the visual cortex of animals. An example is shown in Fig. 2.15 where a handwritten text is classified into the correct digit through layers of convolutional and pooling operators. The challenge in creating a neural network lies in the choice of the layers type, numbers, operators, and order. CNNs are a type of deep learning methods because they are characterized by having multiple hidden neuron layers where each of the layers focuses on extracting a specific feature(s) (Aloysius and Geetha 2017). Among the most famous CNN-based object detection algorithms, primarily for its speed, is the YOLO (You-Only-Look-Once) algorithm (Redmon et al. 2016). Girshick et al. (2014) added region proposals as an attempt to reduce the computational time by suggesting regions to investigate instead of analyzing the entire image. Examples of publicly available CNN building platforms include PyTorch and Tensorflow.

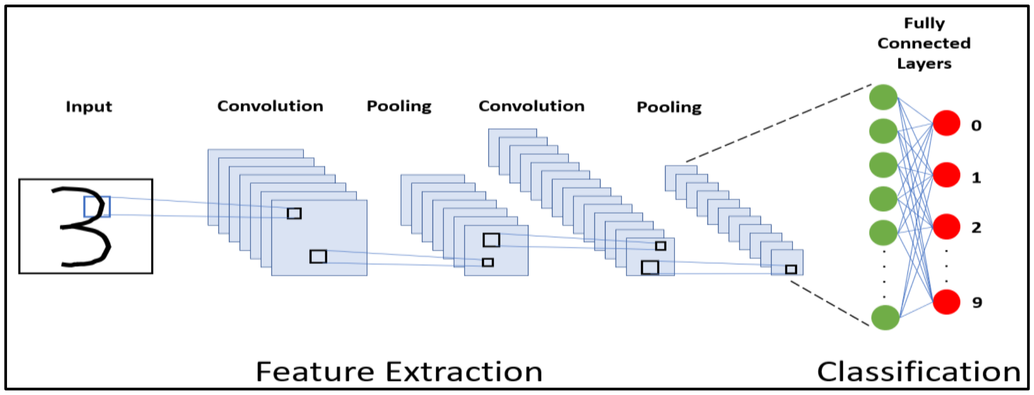


Fig. 2.15: Example of Convolutional Neural Network Architecture

2.4.1.3 Object Measurement

In the literature, two main methods are used to obtain real-life measurements from mages: homography, and photogrammetry. These are explained as follows:

Homography: The homography matrix (H) maps two different images of the same scene to one another (Bovik 2005). That is, every point on the first image corresponds to a point on the second image. If x maps to x’, then x’=Hx. Homography is a 3x3 matrix (h11, h12, h13, ….., h33) yet it has only eight degrees of freedom because it is set to a scale (h33=1). Hence, a four-point correspondence is required to obtain the homography matrix. Fig. 2.16 shows a real-life object being recorded by images from two different viewpoints creating two different images. While the four points on the original object form orthogonal lines, this is not the case in either image due to the distortion caused by the camera lens. Such distortion can be rectified as there exists a homography that relates the image to the real-world scene (Bovik 2005). Consequently, homography can be used to directly obtain real-life measurements from captured images as the homography matrix can find the correlation between the coordinates of a certain point(s) within the image and its corresponding coordinates in the real scene. Fig. 2.17 shows the equation to calculate the homography matrix. In case of having more than four correspondences, a process called RANSAC (RANdom SAmple Consensus) is used to reduce correspondence errors (Dubrofsky 2007). RANSAC can be best explained as solving an optimization problem, choosing the four-point correspondences that create a homography matrix which yields the minimum distance between the real location of features on the original image and the locations projected from their corresponding feature locations on the second image.

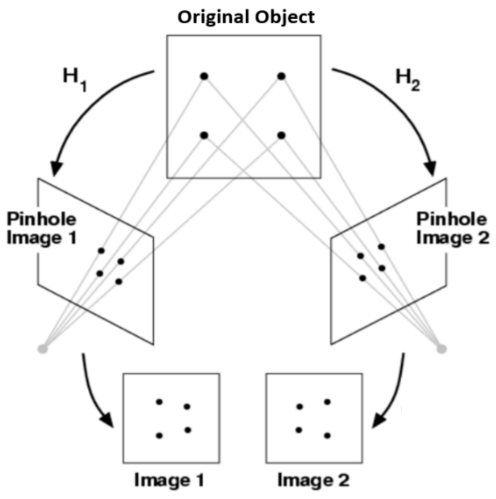


Fig. 2.16: Similar Scene Captured from Multiple Angles Produce Images that can be Mapped to one another (and to the Original Scene)

$$\begin{array}{l}\textbf{Point 1}\\ \\ \textbf{Point 2}\\ \\ \textbf{Point 3}\\ \\ \textbf{Point 4}\\ \\ \end{array}\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix}$$

Fig. 2.17: Equation for Calculating the Homography Matrix

Photogrammetry: Using feature detection and homography techniques illustrated earlier, photogrammetry, often referred to as Structure from Motion (SfM), aims to reconstruct a 3D model of a scene from its projections captured in 2D images (Schonberger and Frahm 2016, Moulon et al. 2012). It is widely used in mapping, surveying, and historical preservation applications (e.g., Hidayat and Cahyono 2016, Jalandoni et al. 2018, Wang et al. 2019b, Shretha et al. 2017). It relies on a simple principle based on similarity of triangles. An example is shown in Fig. 2.18 where a 3D object is projected on the 2D image.

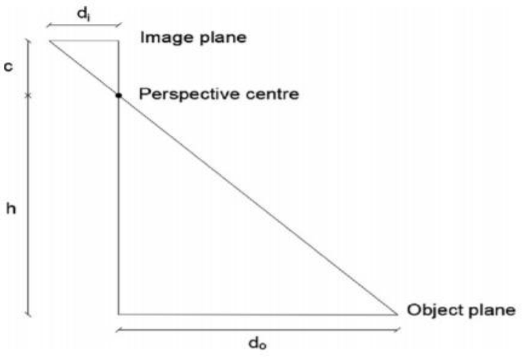


Fig. 2.18: Relationship between Object Plane and Image Plane

SfM algorithm (e.g., Zhang and Xie 2018, Yang et al. 2013) relies on feature matching between images to estimate the camera positions and consequently generate the point cloud. Hence, it is recommended that a minimum overlap of 60% must be present between images to achieve good results (Mikhail et al. 2001). With the proper scale, real measurements can be obtained either directly from the point cloud or by using triangulation to refer to the point cloud through multiple images that show the same feature being measured.

2.4.2 Unsupervised Clustering

Data clustering is a technique that aims to classify the data into “clusters” of similar attributes (Aggrawal and Zhai 2012). This is considered to be an unsupervised technique as it does not require the algorithm to be previously introduced to the data. According to Jain (2010), clustering can help understand the underlying data structure, identify significant features, and organize the data into understandable groups. In the literature, different clustering procedures have been developed, each with its own assumptions regarding the nature of a "cluster" (Jordan and Mitchell 2015).

2.4.2.1 Common Clustering Techniques

The four clustering techniques used in roofing analysis fall under the *partitioning* category. That is, the clusters are created based on the proximity (i.e., similarity) of the data points to one another. Partitioning clustering is robust, efficient with large datasets, and can be optimized to find the best clusters if the number of clusters is predetermined (Shah and Jivani 2013).

Canopy Clustering: Canopy clustering is simple, fast, yet highly accurate (Sharma et al. 2014). The first stage is to divide the data into overlapping subsets (i.e., canopies) based on distance thresholds T1>T2 as shown in Fig. 2.19 (based on McCallum et al. 2000). These thresholds can be set manually or obtained through cross-validation. For each point, the distance between the point and the centers of clusters is examined. If the distance is smaller than T1, then this point is added to the cluster (otherwise the point is considered to be the center of a new cluster). If the distance is smaller than T2, then the point is removed from the set (McCallum et al. 2000). This way, points that are very close to one another are removed to avoid redundant processing in the subsequent stage. The next step is centroid calculation using a more rigorous distance metric (McCallum et al. 2000; Sharma et al. 2014).

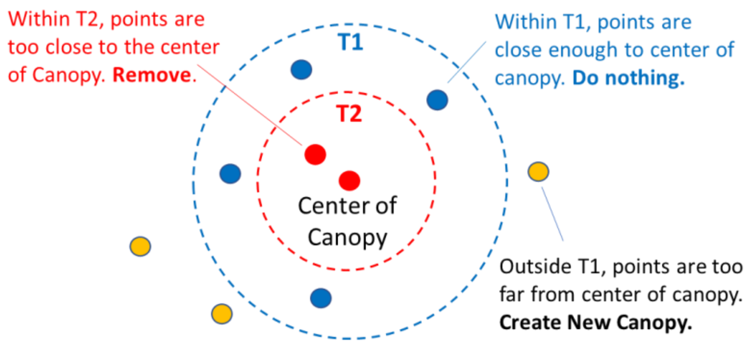

Within T2, points are too close to the center of Canopy. **Remove.**
T1
T2
Within T1, points are close enough to center of canopy. **Do nothing.**
Center of Canopy
Outside T1, points are too far from center of canopy. **Create New Canopy.**

Fig. 2.19: Relationship between Object Plane and Image Plane

K-Means Algorithm: Developed by Hartigan and Wong (1979), K-Means is a well-known partitioning-based algorithm for grouping objects into clusters such that the within-cluster sum of squares is minimized. After specifying the number of required clusters (k), the algorithm randomly chooses k points to serve as the initial centroids of the clusters, and all other points are then assigned to the centroid they are closest to. Afterward, for each cluster, a new centroid is computed by averaging the feature vectors of all data points inside that cluster (Hartigan and Wong 1979). The data points are reassigned to the clusters, based on the new centroids, and then new centroids are calculated. This process is repeated until convergence.

Farthest First (FF) Clustering: In essence, the Farthest-First (FF) clustering algorithm operates in a similar way to the K-Means algorithm in terms of centroid selection and cluster assignment. However, FF chooses the point "farthest away from other cluster centers" as the new cluster center (Sharma et al. 2012). Thus, unlike K-Means, FF does not need a second pass to revise the cluster centroids, which reduces the processing time. Therefore, cluster centroids created by FF are real data points as opposed to geometric centers created by averaging the data points' attributes (Devi et al. 2020).

Expectation-Maximization (EM) Clustering: The EM algorithm is an iterative method that assumes the dataset can be modeled as a linear combination of multiple Gaussian distributions (Abu Abbas 2008). As such, EM aims to find the parameters of the probability distribution best describing the shape of the cluster where the probability (i.e., log-likelihood) that each data point belongs to a certain cluster is highest (Devi and Gandhi 2015). After randomly initializing the cluster shape parameters, the algorithm iterates between estimating the log-likelihood using the current cluster shape parameters, and recomputing those parameters to "maximize" the expected log-likelihood from the previous estimation step (Sharma et al. 2012). As such, the algorithm assigns each data point a probability distribution belonging to a certain cluster (Seghal and Garg 2014).

2.4.3 Applications in the Construction Domain

Data mining utilizes sensory, image, and textual data to support decision-making. As such, data mining and image analysis techniques have been utilized in many engineering and construction applications, including condition assessment and asset management applications. Examples of these applications are presented and discussed in the following subsections.

2.4.3.1 Image Analysis Research in Construction

Research in the field of computer vision has produced valuable numerous benefits in various industries and applications, where the most evident those applications can be seen in drones and autonomous vehicles (e.g. Mobileye 2019). The construction industry has also benefited from computer vision techniques in terms of improving workers' safety and productivity on construction sites. For example, Son et al. (2019) were able to use convolutional neural networks to detect workers within the construction site under varying poses and backgrounds (Fig. 2.20), achieving a 94.1% accuracy rate with an average processing speed of 5 frames per second.

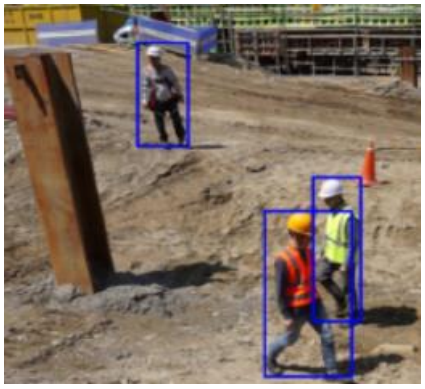


Fig. 2.20: Using CNNs to Detect Workers on Site (Son et al. 2019)

Seong et al. (2017) compared the performance of three types of classifiers (support vector machines, artificial neural networks, and logistic regression) and two different color spaces (Lab and HSV) regarding their suitability for a safety-vest detection system and found that using Support Vector Machines had the most desirable accuracy. To mitigate the likelihood of falls from height, Fang et al. (2018) developed an image-based system to detect if the workers are wearing safety harnesses (Fig. 2.21). The system relies on two CNNs; one detects the workers while the other detects the safety harness.

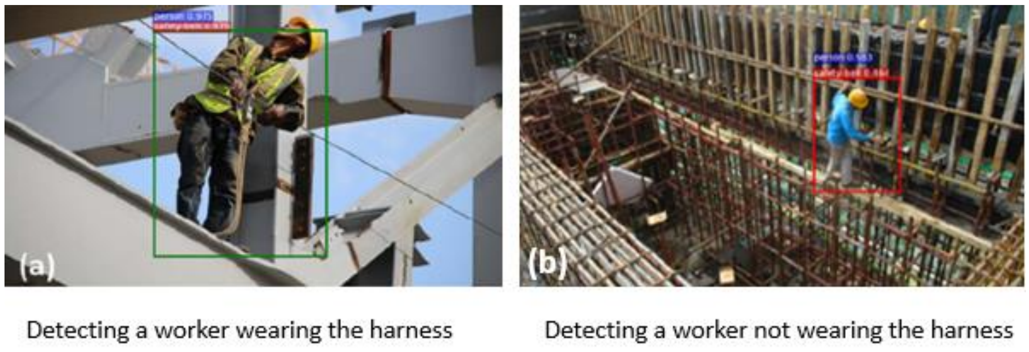

(a)
(b)
Detecting a worker wearing the harness
Detecting a worker not wearing the harness

Fig. 2.21: Detecting workers and safety harnesses using CNNs (Fang et al. 2018)

Using a video sequence, Roberts and Golparvar-Fard (2019) were able to develop a model capable of detecting which activity a piece of equipment is performed by utilizing convolutional neural networks, hidden Markov models, Gaussian mixture models, and support vector machine classifiers. One of the benefits is being able to determine how much time a piece of equipment is being in use

(Fig. 2.22). The only sort of quantification that exists was along the time domain, where the time the equipment spends doing a certain activity is being quantified.

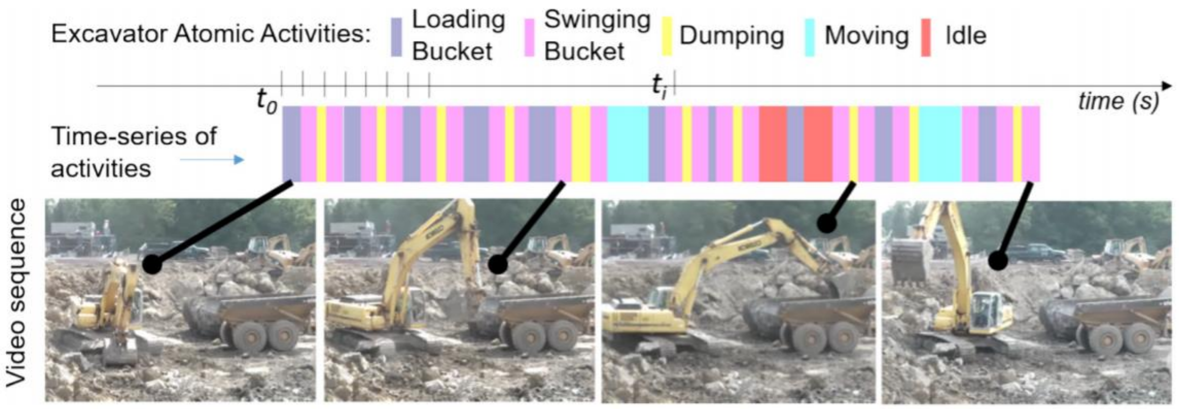


Fig. 2.22: Equipment Activity Classification (Roberts and Golparvar-Fard 2019)

Yang et al. (2013) have used SfM among other stereo-imaging-based techniques to reconstruct 3D models that can be used for Augmented Reality (AR) purposes by project managers and stakeholders. Golparvar-Fard et al. (2009) created an as-built model from recorded images of the construction site, comparing it with a 4D-BIM model for progress monitoring purposes (Fig 2.23).

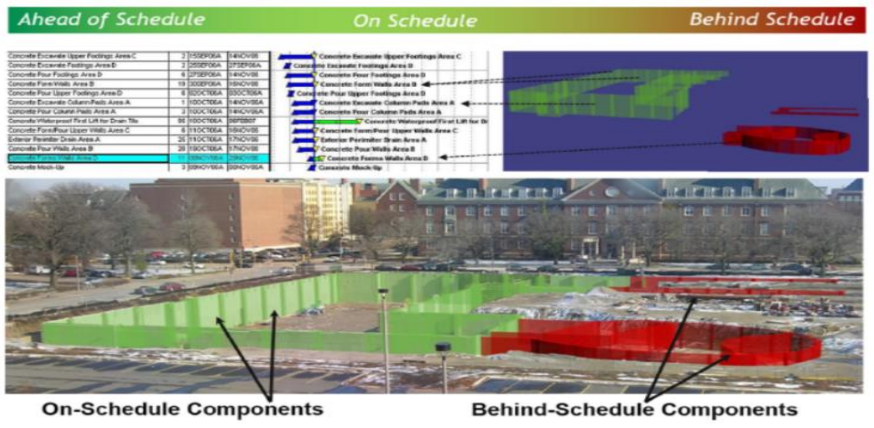


Fig. 2.23: Superimposing 4D-BIM on Time Lapse Images (Golparvar-Fard et al. 2009)

2.4.3.2 Image Analysis in the Inspection Domain

Research works implementing image analysis techniques for structural health monitoring purposes are featured in Table 2.3. As seen in Table 2.3, neural networks is the most common technique utilized for defect detection. It is possible to categorize literature efforts in the application of image analysis in the inspection domain into two categories: efforts for detection of defects; and efforts for quantification of defect size. These are discussed as follows.

Table 2.3: Summary of Image-Based Analysis Applications in Inspection

			Objective			Tool				
No.	**Reference**	**Subject**	Detection	Classification	Quantification	Neural Networks	Edge detectors	Feature Matching	Phtogrammetry/SfM	Other
1	Yudin et al. (2021)	Roofing Defect Detection	Y	Y	Y	Y		Y		
2	Perez and Tah (2021)	Interior Building Defect Detection	Y	Y		Y				
3	Perez et al. (2019)	Interior Building Defect Detection	Y	Y		Y				
4	Napolitano and Glisic (2019)	Masonry Crack Inspection	Y	Y	Y				Y	
5	Cabo et al. (2019)	Local Structural Displacement	Y		Y			Y		
6	Wang et al. (2019b)	Masonry Damage Detection	Y			Y				
7	Liang (2019)	Post-Disaster Inspection Of Columns	Y			Y				
8	Luo et al. (2019)	Concrete Crack Detection	Y							Y
9	Wang et al. (2019a)	Concrete Crack Detection	Y							Y
10	Dung and Anh (2019)	Concrete Crack Detection	Y			Y				
11	Liu et al. (2019)	Concrete Crack Detection	Y			Y				
12	Kim and Cho (2019)	Crack Detection In Concrete Walls	Y		Y	Y				
13	Choi et al. (2018)	Façade Inspections							Y	
14	Dawood et al. (2018)	Moisture Marks Detection	Y		Y	Y	Y			
15	Kumar et al. (2018)	Sewer Defect Detection	Y			Y				
16	Cheng and Wang (2018)	Sewer Defects Detection	Y			Y				
17	Dorafshan et al. (2018)	Concrete Crack Detection	Y			Y	Y			
18	Hoang and Nnguyen (2018)	Crack Detection Of Concrete Walls	Y				Y			
19	Makantasis et al. (2018)	Tunnel Defect Detection	Y			Y				Y
20	Yousaf et al. (2018)	Pavement Potholes Detection	Y					Y		Y
21	Chen and Jahanshani (2018)	Crack Detection In Concrete Bridges	Y			Y				
22	Doulamis et al. (2018)	Crack Detection In Tunnels	Y			Y				
23	Cha et al. (2017a)	Concrete Crack Detection	Y			Y				
24	Hezaveh et al. (2017)	Roofing Defect Detection	Y			Y				
25	Chen et al. (2017)	Bridge Crack Inspection	Y			Y				
26	Zhang et al. (2017)	Pavement Crack Detection	Y			Y				
27	Cha et al. (2017b)	Concrete Cracks, Steel Corrosion	Y			Y				
28	Azhar et al. (2016)	Pavement Potholes Detection	Y							Y
29	Zhang et al. (2016)	Pavement Crack Detection	Y				Y			
30	Yeum and Dyke (2015)	Bridge Crack Detection	Y				Y			

Detection Applications: Most of the ongoing research in the field of utilizing image analysis techniques for inspections and structural health monitoring purposes is geared toward crack detection. For example, Luo et al. (2019) develop a crack detection algorithm that is six times faster than its conventional counterparts without sacrificing accuracy (Fig. 2.24). Their interesting approach processes the image to extract vertical and horizontal cracks separately then fuses them to produce a final image where the cracks are highlighted. Wang et al. (2019a) developed a learning model for concrete crack detection that achieved 97% while maintaining efficient training and testing speeds (0.76s for an image with a resolution of 4608 x 3456 pixels). Such an approach, however, requires a lot of training data. To utilize a small data set, Liu et al. (2019) developed a U-net fully convolutional network to detect concrete cracks (Fig. 2.25) that reached higher accuracy with smaller datasets compared to the fully convolutional network approach used by other researchers such as Dung and Anh (2019). To do that, they first extracted the features using conventional CNN, then used feature fusion to obtain higher precision. Kim and Cho (2019) used region-based CNN to detect and quantify concrete cracks based on their width. The method was able to successfully detect cracks that are wider than 0.3 mm while finer cracks exhibited larger errors due to image accuracy issues (1 image pixel = 0.224mm). Chen et al. (2017) combined image processing techniques (image filtering, background subtraction, and neural networks) with self-organizing map optimization (SOMO) technique to develop a bridge crack inspection model with 90% accuracy. Cha et al. (2017a) used CNNs to detect concrete cracks that achieved 98% accuracy for both training and testing sets, before expanding his detection objectives to include steel corrosion, bolt corrosion, and steel delamination (Cha et al. 2017b). Despite the impressive accuracy of the model, such accuracy was obtained using a training dataset of 40,000 different crack patches, a luxury that may not be afforded by other researchers.

Another prominent area of research is pavement defect detection. Yousaf et al. (2018) proposed a pothole detection and localization scheme relying on the SIFT feature extraction algorithm and support vector machine. Testing said scheme has shown its capability of pothole identification with 95% accuracy. Zhang et al. (2017) proposed a novel CNN architecture, CrackNet, for automated pixel-level crack detection on asphalt surfaces. An experiment using a 200-image testing dataset showed that CrackNet can achieve high Precision (90.13%) and Recall (87.63%) simultaneously. Furthermore, CrackNet can be used in conjunction with the data collection software because of its capability to utilize parallel computing techniques.

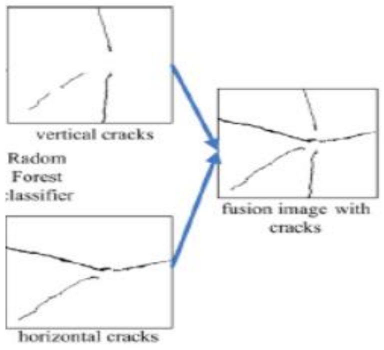


Fig. 2.24: Detection by fusing detected horizontal and vertical cracks (Luo et al. 2019)

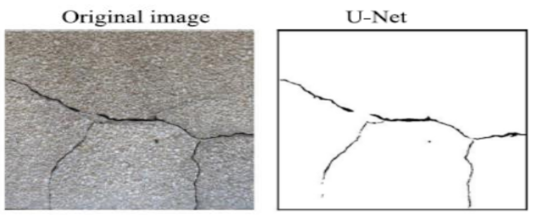


Fig. 2.25: U-net CNNs for Crack Detection (Liu et al. 2019)

Works in other inspection "subdomains" include the CNN-based model developed by Makantasis et al. (2018) for tunnel inspections, using the SURF feature detection algorithm to estimate structural displacements (Cabo et al. 2019), the ANN-based model developed by Dawood et al. (2017) for detection of moisture marks in subway networks, the CNN-based model developed by Wang et al. (2019b) able to automatically detect efflorescence and spalling damages in historic masonry buildings, and the CNN-based schemes developed by Cheng and Wang (2018) and Kumar et al. (2018) to automatically detect cracks, deposits, and root intrusions in sewers from CCTV inspection videos.

In terms of recent efforts aimed at inspecting "non-structural" elements such as roofing, Perez et al. (2019) used VGG-16 CNN architecture to develop a model capable of detecting interior building defects such as mould growth and paint deterioration and reached an average accuracy of 89.25%. In a more recent effort (Perez and Tah 2021) this CNN architecture was then replaced with a MobileNet CNN with the aim of developing a smartphone application capable of real-time detections, but the model accuracy dropped to 80%. In both cases, roofing inspections were not addressed. Hezaveh et al. (2017) developed a CNN model with three convolutional layers and two fully connected layers to detect hail effects in roof shingles. The model achieved 83.4% accuracy and the use of deeper CNN architectures was proposed as a solution to further increase accuracy. Yudin et al. (2021) targeted the development of a comprehensive automated roof detection framework, using image segmentation to identify multiple defects within the same image as well as quantify their sizes. However, the proposed model has low accuracy (mean accuracy < 65%) and long processing time (> 2.5 sec/image).

Quantification Applications: Compared to their detection/classification counterparts, photogrammetry-based techniques for defect measurements and quantification are yet to gain the same popularity. Napolitano and Glisic (2019) attempted to used photogrammetry to inspect cracks in masonry structures. Their system did not directly measure the crack width, but it was calculated as the distance between the bricks where the crack is detected, and did not attempt to measure the crack length at all. A model that was tested using a real-life experiment (a steel frame building in West Lafayette, IN) is the one developed by Choi et al. (2018) to inspect building facades. The developed model uses a drone for automated image collection, then utilizes SfM techniques to generate a rectified photo (orthophoto) of the entire building façade under inspection. Inspectors can then manually select regions of interest for further inspection. A diagram outlining the full model framework is in Fig. 2.26. While the developed model intended to save the inspectors the time, effort, and safety hazards related to physically inspecting the building façade by recreating a scaled orthophoto of it, the damage assessment and quantification phase remains a source for errors and inconsistencies as it is still conducted manually, Hence, it can be concluded that applying photogrammetry for inspection purposes has not been fully capitalized on. Furthermore, since the objective of the inspection is to detect and measure crack dimensions, then just using the homography calculations (a subset from the SfM algorithm featured in Fig. 2.26) might be able to achieve reasonable results without the need for extensive computations required for the entire algorithm.

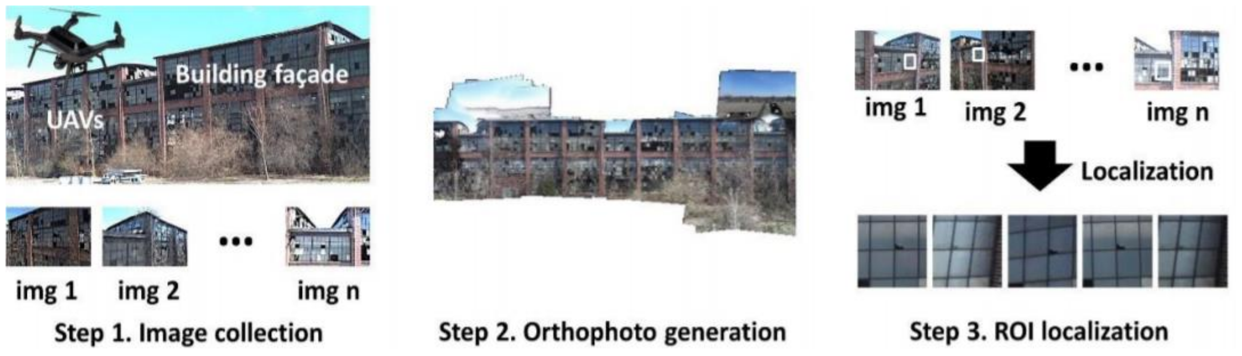


Fig. 2.26: Automated SfM Façade Inspection Model (Choi et al. 2018)

Multiple observations can be drawn based on the previous discussion as well as the works presented in Table 2.3. First, it can be noticed that most researchers have focused on detecting the cracks, a handful of researchers have attempted to estimate the crack width, yet none have considered trying to calculate the length of the crack or quantify the size of the damaged area for quantity take-off purposes. Second, in terms of areas of application, relatively little work has attempted to address "non-structural" elements such as building interiors and roofs, despite their importance towards upholding the functional integrity of the asset. Third, present works only adopt a binary detection approach (defect vs. no defect) without investigating the level of severity of the defect detected, which is the main information needed from inspection to facilitate decision making for repair prioritization, which will be the target of this study. Hence, an analysis component of the overall original image to help draw more useful information by looking at the bigger picture is currently missing.

2.4.3.3 Data Mining and Clustering Applications in Construction and Asset Rehabilitation

Typically, data mining applications involve the use of clustering techniques to identify which part of the data belongs to a certain category of information. Similarly, text mining aims to analyze data from textual reports to retrieve information that supports the decision-making process. This technique was used in various engineering applications and produced beneficial results. For example, Al Hattab (2021) combined text mining with social network analysis to examine the performance of BIM and its relation to sustainability over a 15-year period, by analyzing 523 journal articles. Williams and Betak (2016), used text mining to analyze equipment accident reports from the American Federal Railroad Association, a publicly available database, and identify major themes in railroad equipment accidents. A similar approach was used by Lv and El-Gohary (2016) to extract the key phrases describing project stakeholders' concerns received in emails and public hearings, before classifying them into groups based on topic, to help practitioners detect key concerns at the initial stages of highway projects. Zhao et al. (2016) analyzed occupational safety reports and investigations pertaining to electrocution events and was able to identify activities and decision mistakes that increase the worker's safety risk, thus developing a set of decision-making chains to improve workplace safety.

Some recent efforts exist in applying data mining in the maintenance, inspection, and condition assessment domains, which involve large data sets such as inspection data for bridges, buildings, and other infrastructure systems. Liu and El-Gohary (2017) developed an ontology-based semi-supervised model for information extraction from bridge inspection reports. The model was applied to 11 reports and was able to identify deficiency type, severity, as well as required maintenance actions. Martinez et al. (2020) used predictive modeling to forecast the condition of bridges from bridge characteristics (e.g., size, structural type, etc.) and historical conditions. Gunay et al. (2019) used text-mining to detect failure patterns in building components from textual data in the facility management database. The model uses clustering to filter out the work orders that address failures using rule-mining to identify the coexistence tendencies of certain keywords. Using a similar concept, Mo et al. (2017) analyzed more than 80,000 maintenance requests to properly detect the urgency of a maintenance request based on its textual description and assign the required maintenance crews. Several other studies “mined” different sources of data to support efficient maintenance and optimum operation of building systems. These include the works of Wang et al (2021) to facilitate efficient optimal control strategy for HVAC systems; Zhou et al. (2019) to optimize the operational parameters for chiller plants; and Zhou et al. (2021) of detecting anomalies of daily energy consumption patterns. Few efforts also focused on building renovations and retrofitting. Ren et al. (2019) used actual smart meter data of 666 households and used clustering techniques to identify the groups of households to retrofit their heating systems to cost-effectively maximize energy savings. Kamari et al. (2021) also developed a BIM-based decision support system to generate and evaluates various dwelling renovation scenarios in a Danish context. The study clusters the generated renovation scenarios using sustainability Key Performance Indicators (e.g., energy consumption, investment cost, indoor thermal comfort, etc.).

Based on the above literature review, clustering and text mining techniques can be applied most effectively to large inspection datasets using the most relevant attributes. Various studies have looked into investigating asset failures and supporting maintenance activities. To the author's knowledge, no study exists on using data mining to analyze the inspection reports of buildings and their many systems and sub-systems, for the purpose of identifying the most critical items and supporting fund-allocation decisions. These decisions are most challenging for the owners of many buildings, such as TDSB, particularly since the school inventory involves a large age range; the inspection is done subjectively and often inconsistently by a large number of inspectors; and inspection is done at a high level that leaves many assets at the same criticality level. These factors require careful design of a data mining system in order to be useful and practical. Among the many building components, the paper focuses on roofing as a major component that requires frequent inspections and extensive capital renewal activities.

2.5 Summary of Research Gaps

Currently, most inspections for buildings take place in a manual fashion, either visually or with simplistic hand-held devices, with the inspection process being largely subjective and dependent on the experience of the inspectors. Also, current asset management software act mainly as repositories of the manually collected data, and lack smart decision support for defect quantification, work packaging, and efficient delivery planning for scattered rehabilitation tasks. Image-based analysis techniques have diverse abilities to extract information directly from collected images and videos, thus having good potential to automate the inspection process. However, the literature review revealed that most research is geared towards the detection of defects without classification or quantification. Classifying the defects based on severity is a challenging problem because the different classification categories (e.g., high damage, medium damage, or low damage) tend to overlap and even trained professionals produce inconsistent classifications. Even with proper inspection frameworks, existing inspection reports offer the data without enough granularity to facilitate the prioritization and fund allocation processes, forcing the decision-maker to perform these decisions manually. This is challenging, especially if the asset portfolio is large, and is highly sensitive to the decision maker's biases. Scattered repetitive scheduling seems to be the best way to tackle the delivery of rehabilitation work, but there is little work regards improving the scheduling, optimization, and visualization aspects. Furthermore, applying repetitive scheduling to rehabilitation work poses a new challenge in terms of proper packaging and determining the number of units on which repetitive scheduling calculations will be applied.

Chapter 3: Data Collection and Proposed Framework for Smart Asset Rehabilitation

3.1 Introduction

This chapter introduces the components of the proposed smart asset rehabilitation framework, addressing the inspection, prioritization, and fund allocation, and delivery phases. Key building components are presented and roofing is selected as a key asset that requires detailed asset management. For the criticality assessment, the framework uses Image analysis (Convolutional Neural Networks) to detect and quantify damages and then integrates image analysis with text mining of inspection reports to classify roofs according to their condition and identify the roofs that are most worthy of the limited rehabilitation funds. The short-listed roofs are then passed to the work packaging and project delivery phase. For this purpose, the proposed framework treats roof repairs as scattered repetitive units, thus saving time and cost of the economy of scale. New CPM-LOB formulations and visualization have been developed to address the practical challenges that are commonly encountered in delivering scattered projects and violate meeting deadlines. In addition to explaining the different components of the framework, the process of acquiring the necessary data used in developing and validating the model is also presented.

3.2 Data Collection and Analysis of Key Building Assets

This research has been conducted in collaboration with the Toronto District School Board (TDSB), which is the largest school board in Canada, owning more than 550 schools and other buildings in the GTA area. For TDSB and other large owner organizations whose buildings exhibit a large range of age, inspection plays an important role as a first step in sustaining the healthy performance of those buildings while abiding by the tight budgetary constraints. Typically, external inspection consultants are hired to inspect the schools, over a five-year period, and submit individual school reports that provide data about the condition of all school systems and subsystems, as defined in the standard hierarchy of Fig. 3.1, and suggest rehabilitation strategies (called events) for selected components. The inspection process follows the ASTM E2018-15 (ANSI 2021) condition assessment standard and includes visual analysis, interviews with school representatives, and reviews of building documents. Representative photos of asset conditions are also documented and included in the reports. For each component in the hierarchy of Fig. 3.1, TDSB has historical data about its typical life span, unit cost, and possible defects, which are useful for the analysis.

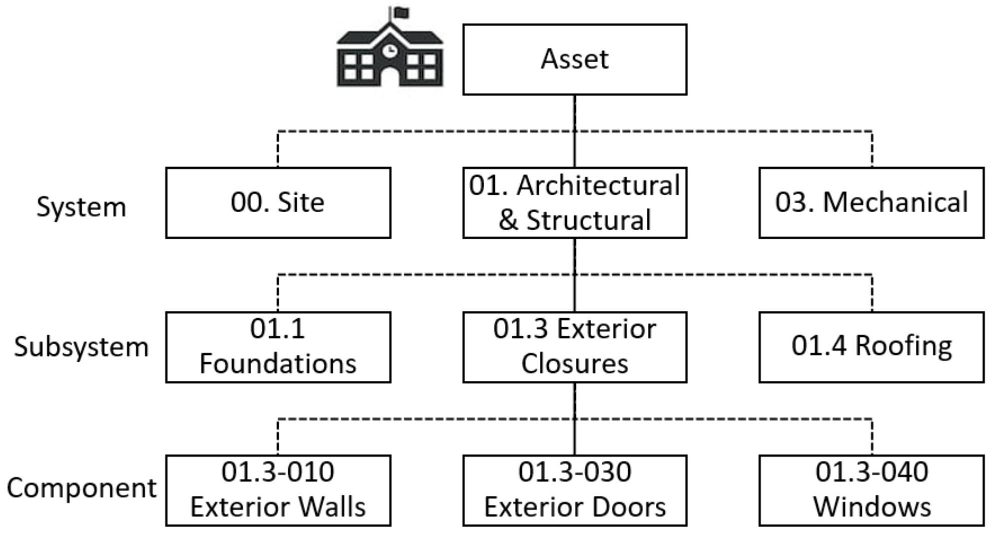


Fig. 3.1: TDSB Building Hierarchy

Among the large number of schools owned by TDSB, inspection reports of 400 schools were obtained as a sample to conduct this study. Each report (pdf file with 20 to 40 pages) starts with the overall assessment of the school systems, followed by a list of the recommended repair/replacement events. Using a custom Macro program, Information related to more than 15,000 unique rehabilitation events was obtained as a result of the data extraction process. The schools under investigation were constructed between 1887 and 1999, with sizes ranging from 500 to nearly 70,000 square meters. The combined cost of all rehabilitation events required for all schools is over \$1.1 billion. High-priority events (35% of the total events) account for half of the total required rehabilitation costs. And those high priority events need to be accomplished within five years only (2003-2007), which is problematic to TDSB, given the financial and time constraints (e.g., much of the rehabilitation work can only take place over the summer when schools are closed). In the absence of a more granular classification, the prioritization and fund allocation efforts become very challenging.

Among all the events, the building components that take the largest share of rehabilitation costs are shown in Fig. 3.2. As shown in the figure, Roofing represents the costliest component with the majority of its events being labeled as high priority (550 or 71% of all roofing events) and thus has been focused upon in this paper. The total rehabilitation cost required to fix all roofing elements is approximately \$140 million, about 14% of all rehabilitation needs of TDSB schools. High-priority roofing events require over \$120 million, making it the component in direst need of rehabilitation work. These numbers, therefore, highlight the research problem indicated earlier, i.e., the daunting task of identifying the roofs that are most eligible for rehabilitation, given the many roofs in high priority and the very limited funds available.

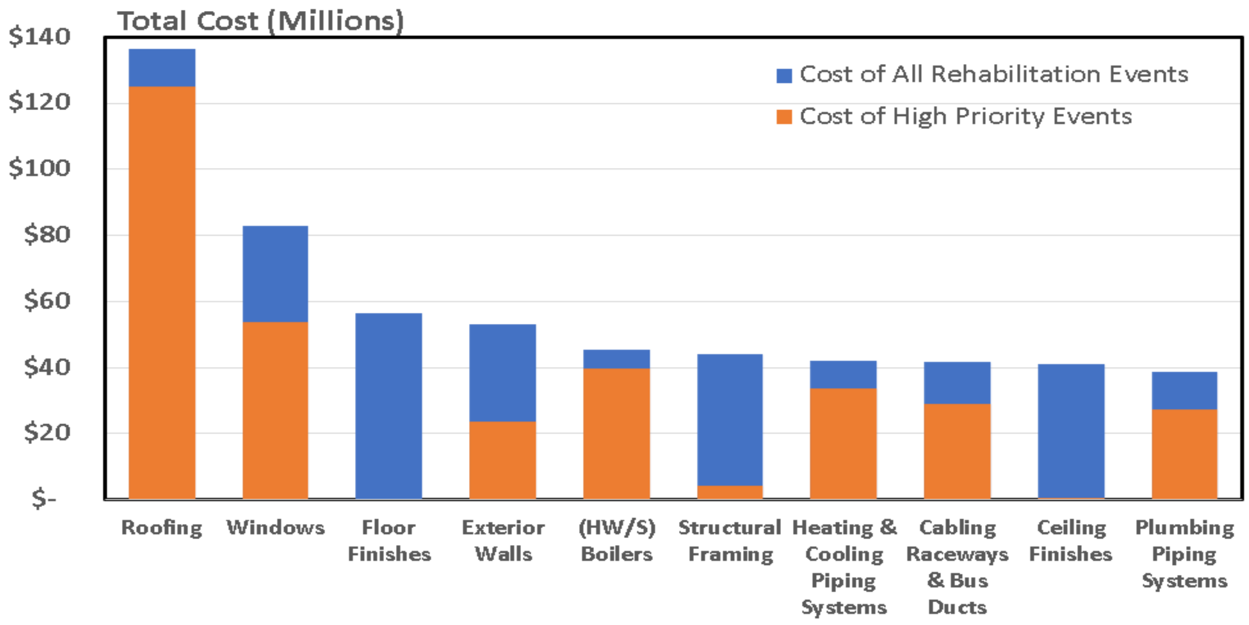


Fig. 3.2: Components with the Highest Total Event Cost

In addition to being a costly component, roofing was selected as the main focus for this study because it gets frequently damaged as it is exposed to the environment, its repairs can either be done using in-house staff and resources or subcontracted to external professionals, and require repetitive scheduling to manage the delivery of scattered roof locations.

3.3 Components of Proposed Framework

With roofing as an example asset, the proposed framework for smart rehabilitation is presented in Fig. 3.3. The model incorporates three main components as follows:

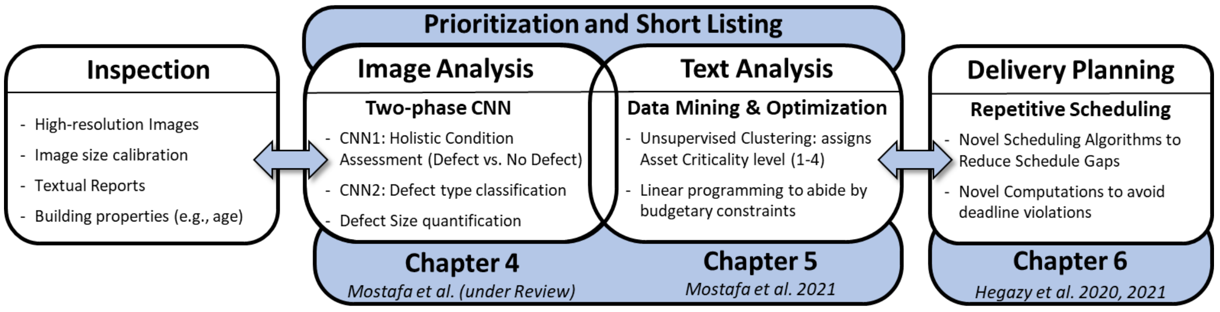

Prioritization and Short Listing
Inspection
- High-resolution Images
- Image size calibration
- Textual Reports
- Building properties (e.g., age)
Image Analysis
Two-phase CNN
- CNN1: Holistic Condition Assessment (Defect vs. No Defect)
- CNN2: Defect type classification
- Defect Size quantification
Text Analysis
Data Mining & Optimization
- Unsupervised Clustering: assigns Asset Criticality level (1-4)
- Linear programming to abide by budgetary constraints
Delivery Planning
Repetitive Scheduling
- Novel Scheduling Algorithms to Reduce Schedule Gaps
- Novel Computations to avoid deadline violations
Chapter 4
Mostafa et al. (under Review)
Chapter 5
Mostafa et al. 2021
Chapter 6
Hegazy et al. 2020, 2021

Fig. 3.3: Components of Proposed Smart Asset Rehabilitation Framework

3.3.1 Inspection

This part is done by expert inspectors and it includes multiple phases. Inspectors perform manual inspections of the roof to assess its current conditions. These inspections are mostly visual (looking for visual defects such as cracks), but sometimes include the use of simple semi-automated tools. Other phases of the inspection process include interviewing building representatives to gain insights about the building performance, as well as reviewing old documents to understand the building history (Mostafa et al. 2021). The results of these inspection activities are combined in a report that highlights the current condition of the asset and suggests future rehabilitation actions if needed.

3.3.2 Prioritization and short listing

a. Utilizing Inspection Images

Because the images in the TDSB inspection reports were of low resolution, other roof images were taken from the University of Waterloo campus buildings for the purpose of the image analysis development. The University of Waterloo is one of the largest universities in Canada and its campus includes more than 40 buildings. After meeting with representatives of plant operations, access to the roofs was granted and images of 21 buildings were collected. The process of the image collection and labeling is discussed in detail in section 4.2, while the visited buildings are highlighted on the campus map in Fig. 3.4.

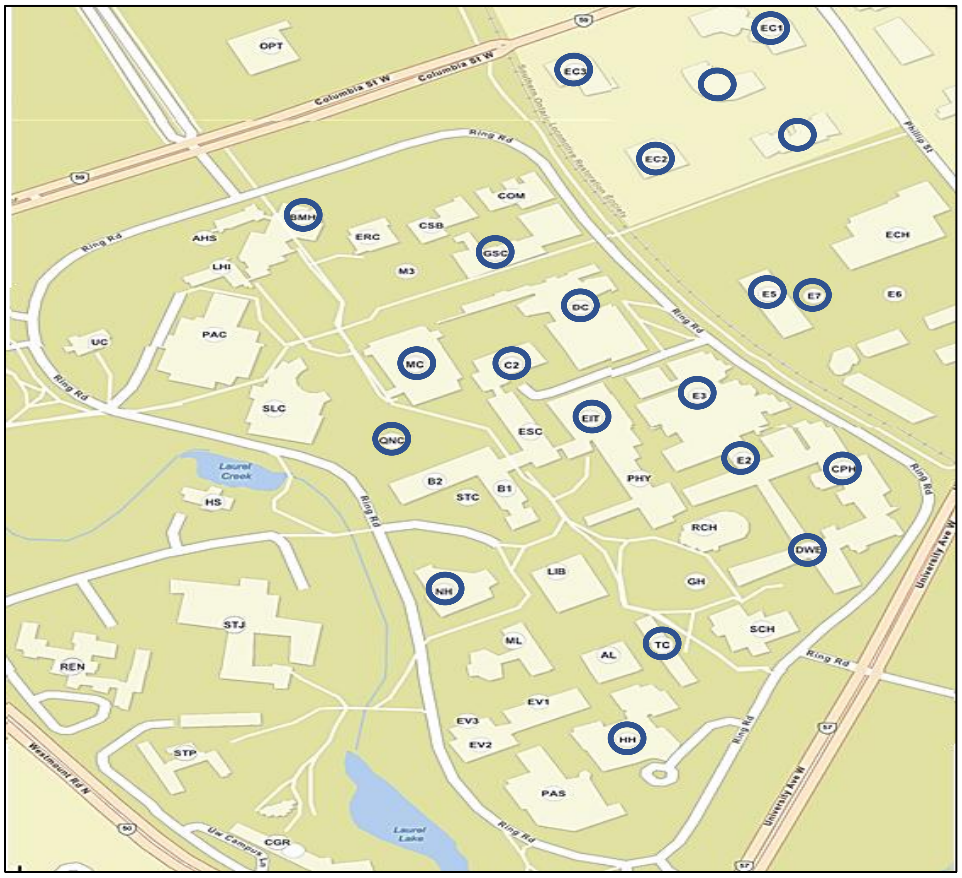


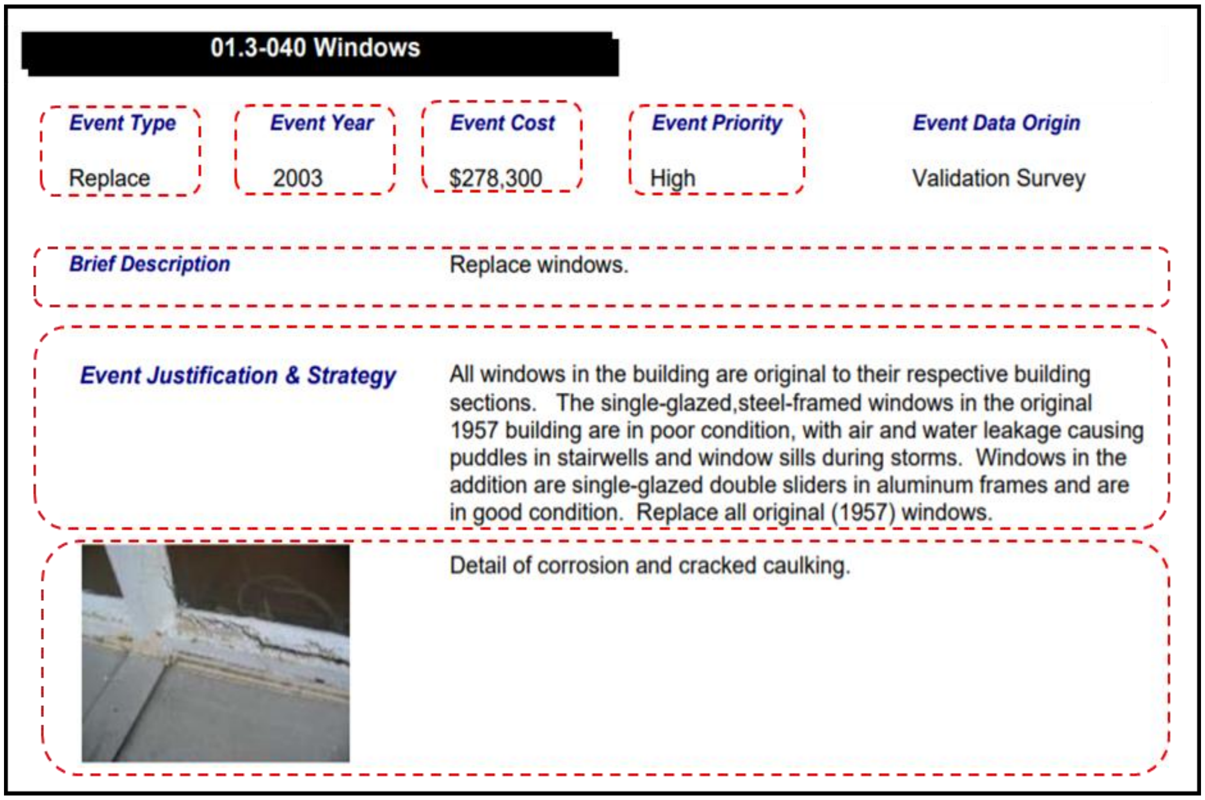
Fig. 3.4: Locations of the University of Waterloo buildings where roof images were taken

The first module developed as part of the proposed asset rehabilitation framework aims to efficiently analyze the inspection images. It is composed of two CNN models; one for defect detection, and the other for defect type classification and size quantification. In its current form, an inspector can collect images of the roof or record a video and then feed the collected images to the model, where classifications are obtained automatically. Consequently, after successful defect quantification, linking the model to a work packaging and estimation database (e.g., RS means) can produce an automated estimate for the required rehabilitation work.

b. Data Mining of Textual Reports

The second module incorporates additional building information such as age and damage description information obtained by textual data mining of the inspection reports, in addition to the results of image analysis, and categorizes the building into one of four categories according to its current condition and its need for repairs. Then, the fund allocation system can now perform an optimization procedure based on the building condition and required rehabilitation costs to better utilize the available funds.

The inspection reports of 400 the schools used in this study are pdf files with 20 to 40 pages each. Extracting useful information from the inspection reports, however, was very challenging as the inspection is conducted at a high level, with details about the specific defects and extent of damage embedded in the "Event justification" textual description, and the text descriptions of related photos. The specified "Event Priority" subjectively classifies the event into three main categories (High, Medium, and Low), thus the chance of having many events in the same category is very high, without further granularity to help guide the fund allocation process. The sample event in Fig. 3.5 shows the following information that was entered during inspection, and extracted in this study from inspection reports using a VBA macro code developed by the authors to automate the extraction of PDF data:



01.3-040 Windows

Event Type	Event Year	Event Cost	Event Priority	Event Data Origin
Replace	2003	$278,300	High	Validation Survey

Brief Description: Replace windows.

Event Justification & Strategy: All windows in the building are original to their respective building sections. The single-glazed,steel-framed windows in the original 1957 building are in poor condition, with air and water leakage causing puddles in stairwells and window sills during storms. Windows in the addition are single-glazed double sliders in aluminum frames and are in good condition. Replace all original (1957) windows.

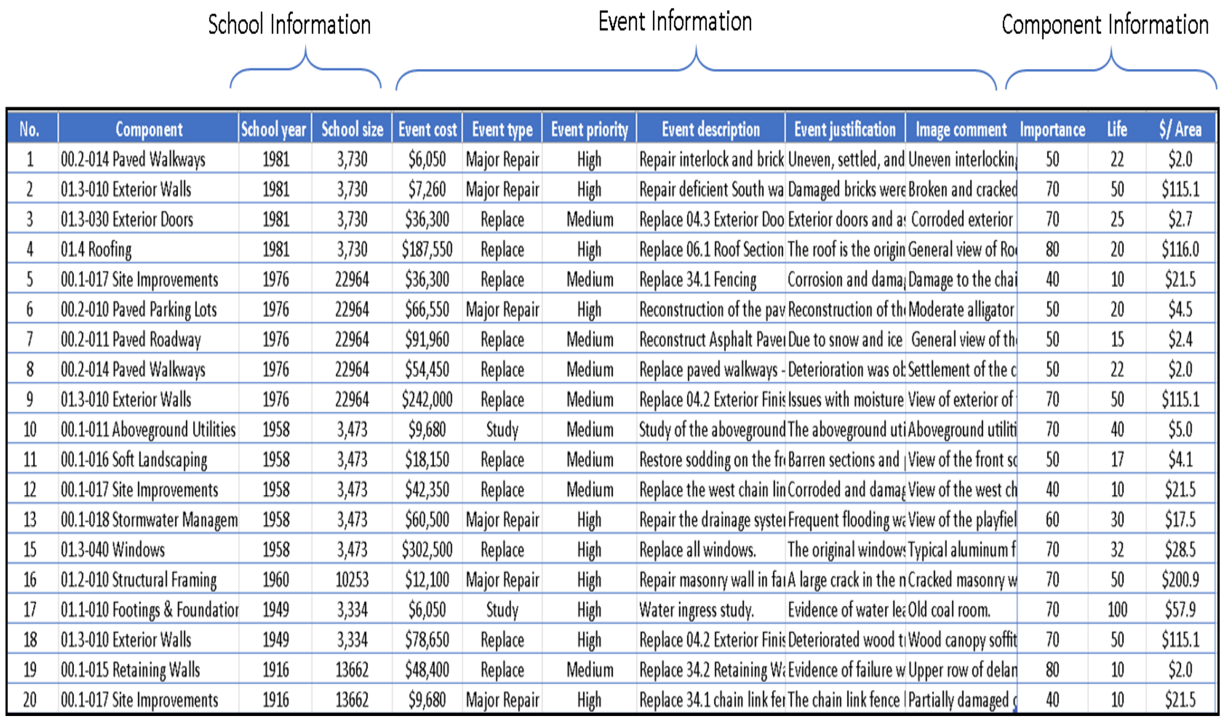
Detail of corrosion and cracked caulking.

Fig. 3.5: Example of TDSB Event Documentation

- Event Type: Specifies one of three possible suggestions: (1) Replace the component; (2) Repair the component, or (3) Conduct Further Study to evaluate the component's condition;
- Event Year: Suggested rehabilitation year by the inspector, and is typically within five years from the date of inspection;

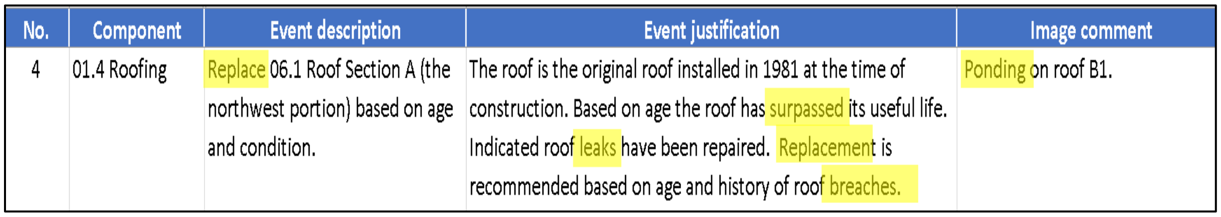
- Event Cost: The event costs were computer-generated, based on a template created by the inspector using various construction cost estimation reference databases such as RS Means, Whitestone, and Hanscomb pricing guides, as well as the TDSB's own database;
- Event priority: The inspector's assessment of the event's urgency: High, Medium, or Low;
- Event Description: A brief description of the event nature;
- Event Justification: A detailed explanation of the component's current condition and defects (if any) as well as any implications that may result from delaying the rehabilitation event;
- Photos with defect samples.

Following the data extraction process, a database of all extracted events was compiled, as shown in Fig. 3.6, with information added about each school (age and size), and component information from TDSB databases (importance, theoretical life, and unit cost). A sample of the textual information for a roofing event is shown in Fig. 3.7, describing the various defects noticed during inspection.


School Information
Event Information
Component Information

No.	Component	School year	School size	Event cost	Event type	Event priority	Event description	Event justification	Image comment	Importance	Life	$/ Area
1	00.2-014 Paved Walkways	1981	3,730	$6,050	Major Repair	High	Repair interlock and brick	Uneven, settled, and	Uneven interlockin	50	22	$2.0
2	01.3-010 Exterior Walls	1981	3,730	$7,260	Major Repair	High	Repair deficient South wa	Damaged bricks were	Broken and cracked	70	50	$115.1
3	01.3-030 Exterior Doors	1981	3,730	$36,300	Replace	Medium	Replace 04.3 Exterior Doo	Exterior doors and a	Corroded exterior	70	25	$2.7
4	01.4 Roofing	1981	3,730	$187,550	Replace	High	Replace 06.1 Roof Section	The roof is the origin	General view of Ro	80	20	$116.0
5	00.1-017 Site Improvements	1976	22964	$36,300	Replace	Medium	Replace 34.1 Fencing	Corrosion and dama	Damage to the chai	40	10	$21.5
6	00.2-010 Paved Parking Lots	1976	22964	$66,550	Major Repair	High	Reconstruction of the pav	Reconstruction of th	Moderate alligator	50	20	$4.5
7	00.2-011 Paved Roadway	1976	22964	$91,960	Replace	Medium	Reconstruct Asphalt Pave	Due to snow and ice	General view of th	50	15	$2.4
8	00.2-014 Paved Walkways	1976	22964	$54,450	Replace	Medium	Replace paved walkways -	Deterioration was ob	Settlement of the c	50	22	$2.0
9	01.3-010 Exterior Walls	1976	22964	$242,000	Replace	Medium	Replace 04.2 Exterior Finis	Issues with moisture	View of exterior of	70	50	$115.1
10	00.1-011 Aboveground Utilities	1958	3,473	$9,680	Study	Medium	Study of the aboveground	The aboveground uti	Aboveground utiliti	70	40	$5.0
11	00.1-016 Soft Landscaping	1958	3,473	$18,150	Replace	Medium	Restore sodding on the fr	Barren sections and	View of the front sc	50	17	$4.1
12	00.1-017 Site Improvements	1958	3,473	$42,350	Replace	Medium	Replace the west chain lin	Corroded and damag	View of the west ch	40	10	$21.5
13	00.1-018 Stormwater Managem	1958	3,473	$60,500	Major Repair	High	Repair the drainage syste	Frequent flooding wa	View of the playfiel	60	30	$17.5
15	01.3-040 Windows	1958	3,473	$302,500	Replace	High	Replace all windows.	The original window	Typical aluminum f	70	32	$28.5
16	01.2-010 Structural Framing	1960	10253	$12,100	Major Repair	High	Repair masonry wall in fa	A large crack in the n	Cracked masonry w	70	50	$200.9
17	01.1-010 Footings & Foundatior	1949	3,334	$6,050	Study	High	Water ingress study.	Evidence of water lea	Old coal room.	70	100	$57.9
18	01.3-010 Exterior Walls	1949	3,334	$78,650	Replace	High	Replace 04.2 Exterior Finis	Deteriorated wood t	Wood canopy soffit	70	50	$115.1
19	00.1-015 Retaining Walls	1916	13662	$48,400	Replace	Medium	Replace 34.2 Retaining Wa	Evidence of failure w	Upper row of delan	80	10	$2.0
20	00.1-017 Site Improvements	1916	13662	$9,680	Major Repair	High	Replace 34.1 chain link fe	The chain link fence	Partially damaged	40	10	$21.5

Fig. 3.6: Database of Suggested Rehabilitation Events



No.	Component	Event description	Event justification	Image comment
4	01.4 Roofing	Replace 06.1 Roof Section A (the northwest portion) based on age and condition.	The roof is the original roof installed in 1981 at the time of construction. Based on age the roof has surpassed its useful life. Indicated roof leaks have been repaired. Replacement is recommended based on age and history of roof breaches.	Ponding on roof B1.

Fig. 3.7: Sample of Textual Description for a Roofing Event

3.3.3 Delivery planning

Once the most critical rehabilitation events are identified through the first two modules, the third and final module aims to develop an efficient delivery plan. But, instead of treating every rehabilitation event as a separate project, this module treats them as units of a repetitive project. Such treatment allows for the use of repetitive scheduling techniques such as LOB calculations which saves time and cost by applying economies of scale and capitalizing on worker's momentum and learning curve. In reality, current LOB calculations fail to account for practical constraints such as unidentical unit sizes, non-integer start and finish times, and varying delivery rates which lead to project overruns. This module incorporates novel computations, visualizations, and algorithms to remedy these shortfalls.

3.4 Summary

This chapter has introduced the components of the proposed roof rehabilitation framework. The framework consists of three main components; a) Image-based analysis; b) Text-based analysis; and c) Repetitive Scheduling. First, the framework determines the size and type of defects that the roof exhibits based on analyzing photos of the roof. Then, the results of the analysis are combined with text-based analysis that investigates other features of the roof such as age to assign the roof to one of four criticality levels based on how imminent the rehabilitation work has to be, before an optimization model selects the roofs that are most worthy of repair based on the limited budget. Finally, the final model addresses the selected roofs and produces an optimized delivery schedule for the required rehabilitation work. To account for the scattered and complex nature of the infrastructure renewal projects, new computations, visualizations, and algorithms are presented to overcome the shortcomings of conventional LOB calculations. Each of the three modules shall be discussed in more detail in the coming chapters.

Chapter 4: Convolutional Neural Network for Defect Detection and Classification

4.1 Introduction

As highlighted in Fig. 4.1, this chapter introduces the first module of the roofing rehabilitation framework; an image analysis model that uses Convolutional Neural Networks (CNNs) for defect detection and type classification directly from roofing images. As seen in Fig. 4.2, the proposed model is a two-phased model, with the first phase responsible for damage detection while the second phase analyzes the defected images to determine the damage type. The chapter starts with the experimental setup implemented for data collection and preprocessing, before discussing the proposed architecture for the different phases of the model. Finally, the results of the model validation are presented.

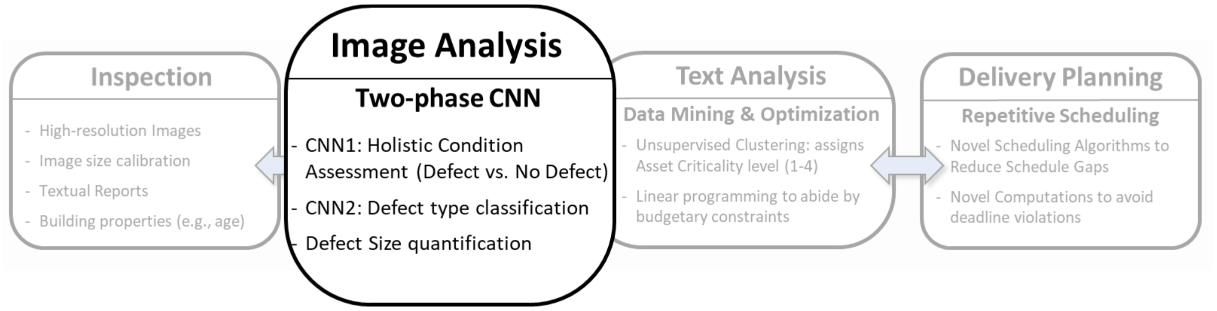

Inspection
- High-resolution Images
- Image size calibration
- Textual Reports
- Building properties (e.g., age)
Image Analysis
Two-phase CNN
- CNN1: Holistic Condition Assessment (Defect vs. No Defect)
- CNN2: Defect type classification
- Defect Size quantification
Text Analysis
Data Mining & Optimization
- Unsupervised Clustering: assigns Asset Criticality level (1-4)
- Linear programming to abide by budgetary constraints
Delivery Planning
Repetitive Scheduling
- Novel Scheduling Algorithms to Reduce Schedule Gaps
- Novel Computations to avoid deadline violations

Fig. 4.1: First Module of the Proposed Framework

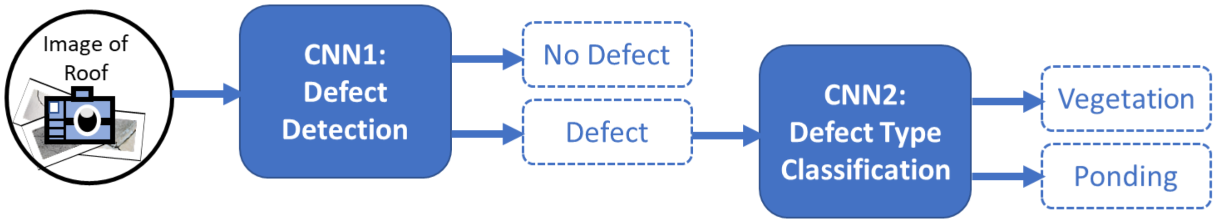


Fig. 4.2: Proposed CNN-based Module

4.2 Experimental Setup

A 16-megapixel phone camera with a field of view (FOV) of 78 degrees was used for the image collection process. Image resolution was set to 1,920x1,080 pixels. To eliminate distortion, the camera was mounted on a selfie stick and maintained parallel to the ground surface at all times. The camera was also kept at a constant height (waist level). This meant that all the photos taken correspond to areas of the same size. As shown in Fig. 4.3, the size of the area captured in the photo frame was calibrated using an object of known size (black folder, 30X22.5cm). It was found that the total area captured in the photo is 1.5X1.12m.

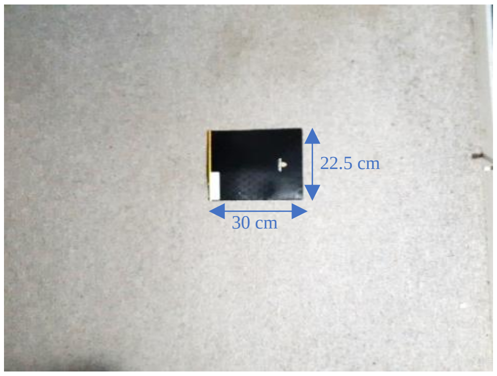
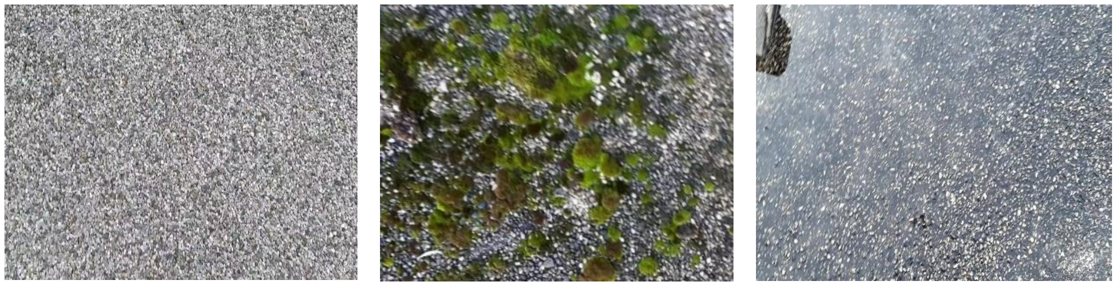


Fig. 4.3: Reference Image used for Calibration

To capture a large number of images, the camera was set to “video mode” and a video traversing the roof surface was recorded. Then, image frames were extracted from the recorded video. The videos were recorded at a speed of 30 frames per second, and 1 in 10 frames was extracted. This corresponds to an image capturing speed of 3 images per second. This resulted in over 11,000 images in total. The retrieved images were then manually labeled according to the defect type they exhibit. The most prevalent damage types were vegetation (183 images) and water ponding (254 images). Examples of the labelled images showing the three categories (no defect, vegetation, and ponding) are in Fig. 4.4 as well as Appendix A. Images were labelled as “defected” if the defect covers 20% or more of the overall image.



(a) No Defect (b) Vegetation (c) Ponding

Fig. 4.4: Example of Labelled Images

4.2.1 Dataset Augmentation and Splitting

Due to the relatively small number of images showing the different defects (496 images out of more than 11,000), data augmentation techniques were used. The images were rotated to the left by 90 degrees, to the right by 90 degrees, flipped horizontally, and flipped vertically. An example of the data augmentation process is in Fig. 4.5. This has increased the size of the image dataset to 2480 images. In addition to increasing the size of the dataset, models trained with a dataset that has undergone data augmentation exhibit better generalization and more adaptability to different architectures as opposed to parameter fine-tuning (Hernandez-Garcia and Konig 2019). To ensure that the dataset used for training has an equal representation of both categories (defects and no defects), 2608 images that show no defects were randomly selected to be part of the training dataset, bringing the total size of the dataset to 5088 images.

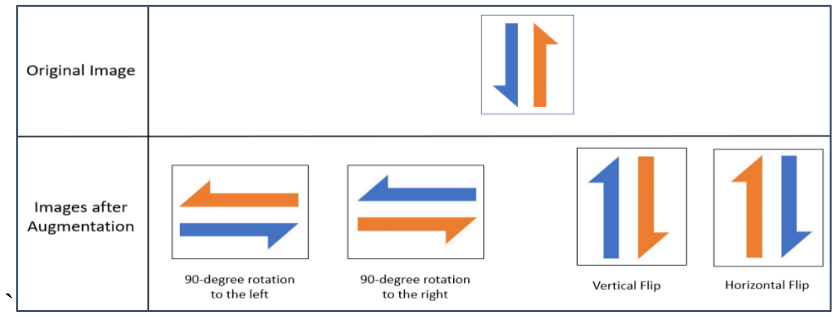

Original Image
Images after Augmentation
90-degree rotation to the left
90-degree rotation to the right
Vertical Flip
Horizontal Flip

Fig. 4.5: Example of Image Augmentation

4.3 Proposed CNN Architecture

Despite the presence of many powerful networks such as ResNet, AlexNet, VGG, GoogleNet, and others, the original Zeiler-Fergus network architecture (Zeiler and Fergus 2014) which won the Large-Scale Visual Recognition Challenge 2013 is still being used by many researchers today to perform their classification tasks (e.g., Cha et al. 2017b, Deng et al. 2019). This study uses a simpler version of the Zeiler-Fergus network whose architecture can be seen in Fig. 4.6. The parameters of the convolutional and maxpooling layers are denoted as (number of kernels@length and width of each kernel). For example, in the first convolution layer, the input image is convolved with 24 different kernels, each having a length and width of 10 pixels. The third dimension of the kernel is always equal to the third dimension of the input (i.e., the number of channels). Hence, the third dimension of the convolutional kernels in the first layer is equal to three. The output of the first convolutional procedure will have 24 channels (one channel for each convolution kernel). As such, the maxpooling operation is performed at the first phase using 24 kernels whose dimensions are 4x4x24. This is because the objective of maxpooling is to perform down sampling takes place along the length and width of each channel.

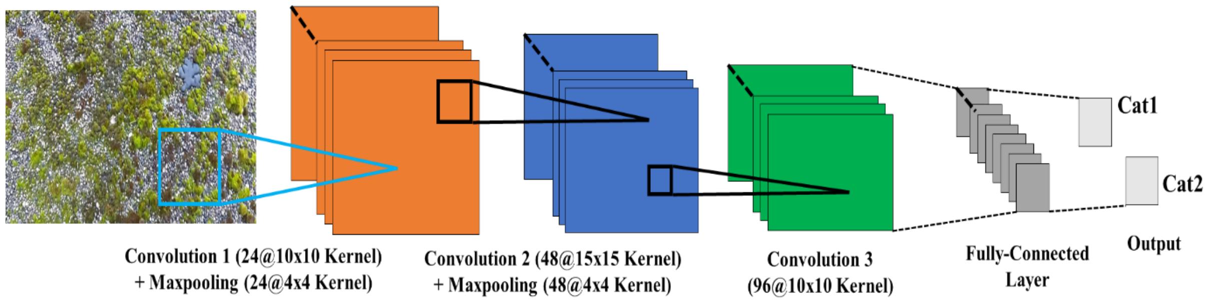

Cat1
Cat2
Output
Convolution 1 (24@10x10 Kernel) + Maxpooling (24@4x4 Kernel)
Convolution 2 (48@15x15 Kernel) + Maxpooling (48@4x4 Kernel)
Convolution 3 (96@10x10 Kernel)
Fully-Connected Layer

Fig. 4.6: Overall Architecture of the Proposed CNN

The rectified linear unit (ReLu) function is used as the activation function. To save computing time and to accommodate the different image orientations, all images are resized into 256x256 before the start of the training, validation, or prediction phases. The same architecture is used for both networks: type classification, and severity classification. As such, the output categories 1,2 refer to *no defect*, and *defect* in the case of CNN1, but refer to *vegetation*, and *Ponding* in the case of CNN2. The final output is calculated using a softmax function, which calculates the probability of the input belonging to each output class before announcing the output as the class with the highest probability.

To write the goal of CNN (minimize the difference between predicted classification and correct classification) in a mathematical form, then the objective is to minimize the loss function $E=D-f(W)$. Where D is the “correct output” while $f(W)$ is the predicted output as a function of the input parameters W. The loss function is minimized by estimating the impact of changing the parameter values on the loss function (i.e., the derivative of the loss function with respect to the parameter values). To solve this minimization problem, this network relies on the Adaptive Moment Estimation (ADAM) optimizer (Kingma and Ba 2015). ADAM is an algorithm for first-order gradient-based optimization that applies conventional stochastic gradient descent principles but adds a factor (γ) depending on the direction of the gradients (higher values for gradients that point in the same direction) to prevent oscillation and reach convergence faster (Kingma and Ba 2015). Finally, this operation is performed after a batch of inputs is fed to the model (64 images in the case of the model used in this paper). This serves two purposes; it reduces the overall computation time since the process is done once after each batch as opposed to once after each image, and prevents the model from overfitting based on the parameters of one image only.

4.3.1 Advantages of the Two Step Approach

Multiple experiments were conducted to reach the most efficient model. For example, the proposed two-step approach was compared to a multi-classifier CNN that aims to detect and classify the defects in one step (i.e., the CNN outputs are: no defect, vegetation, or ponding). Although it had a comparable accuracy level to the two-step approach when it operated on the training and testing datasets, performing the campus-wide deployment has highlighted various issues with the single-step multi-classifier. First, the performance of the defect detection (vegetation vs. ponding) phase has deteriorated. Second, and most important, a large number of images that originally showed defects (26%) were classified as no defects, which leads to an underestimation of the roof condition. This is because minimizing the number of class labels increases the accuracy of the model, as images can be misclassified in the case of having too many labels compiled into the same classifier. This has been verified by a case study, which will be explained in more detail later in this chapter.

4.4 Code Development

The network was developed and compiled using the Scientific Python Development Environment (Spyder®), part of the Anaconda® scientific programming distribution (Individual Edition). Anaconda was chosen as it already contains and/or easily supports the installation of many python packages that are essential to this work such as PyTorch, NumPy, and Pandas. The developed code is composed of five different modules as follows (Full code in Appendix B):

1. *Model*: includes the model architecture in terms of the layer composition and activation functions.
2. *Dataset*: responsible for retrieving the image dataset, augmenting the images to fit the network requirements, translating the textual labels to numerical values for training, and calculating the weighted sampling ratios in case of training with unbalanced data.
3. *Myutils*: includes necessary functions for interpreting the network results, such as calculating the network accuracy in the training phase, and producing the output labels in the prediction phase (Screenshot in Fig. 4.7)
4. *Train*: responsible for executing the entire training phase. This includes:

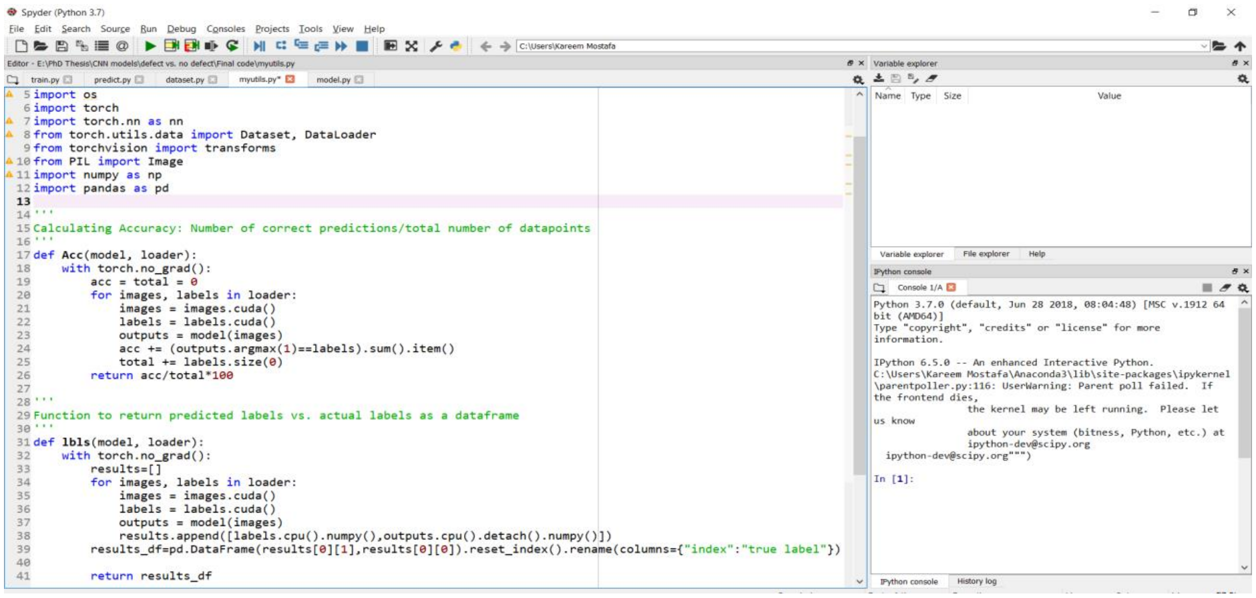
a. Calling the *Dataset* module to retrieve the training and validation datasets and feed them to the CNN;

b. Set the different hyperparameters such as the seed value (for experimentation purposes, number of training epochs (and loop through them), the batch size, and the learning rate;

c. Report the training results (training and validation accuracy for each training epoch); and

d. Save the model parameters to be used for the prediction phase

5. *Predict:* responsible for executing the entire prediction phase. Including loading the model and the prediction dataset, and reporting the prediction results.



Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Kareem Mostafa
Editor - E:\PhD Thesis\CNN models\defect vs. no defect\Final code\myutils.py
train.py
predict.py
dataset.py
myutils.py*
model.py
Variable explorer
Name Type Size Value
Variable explorer File explorer Help
IPython console
Console 1/A
IPython console History log

```python
import os
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
from PIL import Image
import numpy as np
import pandas as pd

'''
Calculating Accuracy: Number of correct predictions/total number of datapoints
'''
def Acc(model, loader):
    with torch.no_grad():
        acc = total = 0
        for images, labels in loader:
            images = images.cuda()
            labels = labels.cuda()
            outputs = model(images)
            acc += (outputs.argmax(1)==labels).sum().item()
            total += labels.size(0)
        return acc/total*100

'''
Function to return predicted labels vs. actual labels as a dataframe
'''
def lbls(model, loader):
    with torch.no_grad():
        results=[]
        for images, labels in loader:
            images = images.cuda()
            labels = labels.cuda()
            outputs = model(images)
            results.append([labels.cpu().numpy(),outputs.cpu().detach().numpy()])
        results_df=pd.DataFrame(results[0][1],results[0][0]).reset_index().rename(columns={"index":"true label"})

        return results_df
```

```
Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64
bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 6.5.0 -- An enhanced Interactive Python.
C:\Users\Kareem Mostafa\Anaconda3\lib\site-packages\ipykernel
\parentpoller.py:116: UserWarning: Parent poll failed.  If
the frontend dies,
                the kernel may be left running.  Please let
us know
                about your system (bitness, Python, etc.) at
                ipython-dev@scipy.org
  ipython-dev@scipy.org""")

In [1]:
```

Fig. 4.7: Screenshot of *Myutils* Module in the Developed Code

4.5 Implementation Details and Results

All experiments were performed using the python programming language (CUDA 7.5) on a personal laptop with Core i7-10750H@2.6GHz CPU, 16GB RAM, and 4GB NVIDIA GeForce GTX 1650 Graphical Processing Unit (GPU). The learning rate was set to be 0.0001 while the batch size was set to 64 images. All datasets have undergone a 75/25 split where 25% of the images were set aside to test the model’s predictive ability, while the remaining 75% went through another 75/25 split for training and validation purposes, respectively.

4.5.1 Detection Phase (Defect Vs. No Defect)

The first developed CNN was to classify the image as defect or no defect. It was trained for 1000 epochs with an average processing time of 3 minutes per epoch (2862 images). The classification accuracy (i.e., percentage of correct classifications) was used as the evaluation metric of the CNN performance. The model accuracy for the training and validation datasets, per epoch, are presented in Fig. 4.8. The graph shows that the model exhibits satisfying performance, achieving close to 95% accuracy on both datasets.

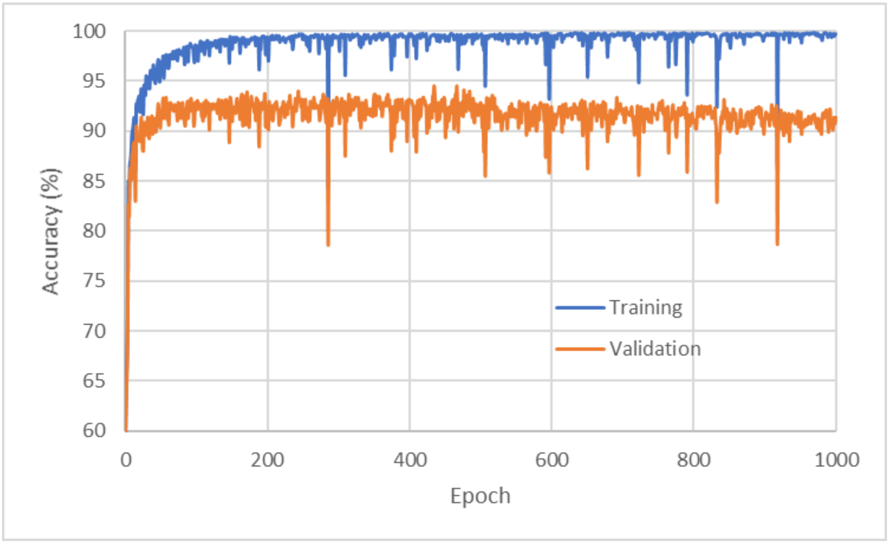


Fig. 4.8: Model Accuracy for Training and Validation Datasets

To further validate the performance of the model, it was tested on the prediction dataset, a portion of the images that the model did not see during the training phase. Table 4.1 shows the model prediction results. It can be seen that the model exhibits good prediction results overall, achieving 94.9% accuracy. Furthermore, the amount of false negatives (i.e., images with defects that were misclassified) is less than 3%, which means that asset management personnel can trust the model's outputs.

Table 4.1: Confusion Matrix for the Model Prediction Results

		True Label	
		No Defect	Defect
Predicted Label	No Defect	605	46
	Defect	19	594

With the limited availability of image classification models related to roofing defects in the literature, it was challenging to compare the performance of the proposed model to the state of the art. Table 4.2 compares the model to existing roofing defect detection models in terms of accuracy and speed. In addition, to gain a better sense of the model performance, Table 4.3 compares the proposed model to similar models created for different purposes such as pavement (e.g., Li et al. 2020; Mei and Gul 2020), and Concrete (Cha et al. 2018) defect detection. The objective of these comparisons is to show where the proposed model stands in terms of its effectiveness for its intended purpose, compared to the stat of the art models used in different fields such as concrete or pavement inspections.

Table 4.2: Comparing the Proposed Model to Existing Roofing Detection Models

Model	Accuracy	Processing Time per Image
Proposed Model	**94.9%**	**< 0.01 sec**
Hezaveh et al. 2017	83.4%	N/A
Yudin et al. 2021	65%	> 2.5 sec

Table 4.3: Comparing between the Proposed Model and Others in the Literature

Model	Purpose	Accuracy
Li et al. 2020	Pavement Crack Detection	94%
Mei and Gul 2020	Pavement Crack Detection	92%
Perez et al. 2019	Building Defect Detection	89.1%
Cha et al. 2017b	Concrete Defect Detection	89.7%
Proposed Model	**Roofing Defect Detection**	**94.9%**

4.5.2 Classification Phase

A second CNN model was developed with the purpose of classifying the defects according to their type. For this experiment, only the images that include vegetation and ponding defects (2185 images) were used. The model was trained for 1000 epochs and had a faster training time (approx. 1 minute per epoch) because the dataset was smaller (1230 images). The model accuracy for the training and validation datasets, per epoch, are presented in Fig. 4.9. It can be seen that not only the classification model was able to achieve higher accuracy for both training and testing datasets, but it was done at a smaller number of epochs. This was further clarified when the model performance was tested against the prediction dataset, yielding 97% accuracy (compared to 94.5% accuracy of the detection model).

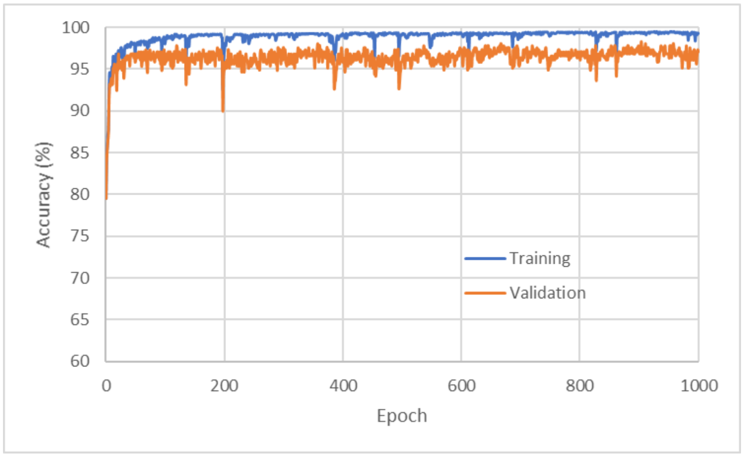


Fig. 4.9: Model Accuracy for Training and Validation Datasets

4.6 Model Validation

Based on the promising results shown by the model on the training, testing, and prediction datasets. The next step was to validate the model's performance in real-life scenarios. This will be explained in this subsection in two phases. First, a detailed analysis of the results of two buildings will be presented as validation case studies. Next, the results of large-scale deployment over 21 buildings in the University of Waterloo are presented.

4.6.1 Experiments on Individual Buildings

To show the potential of the proposed method, two buildings (Chemistry 2 and Douglas Wright Engineering) were used as case studies. The locations of the selected buildings are highlighted in Fig. 4.10 (a zoomed-in version of the map previously shown in Fig. 3.4). Photos were collected in a manner similar to the one explained earlier as part of this study's experimental setup.

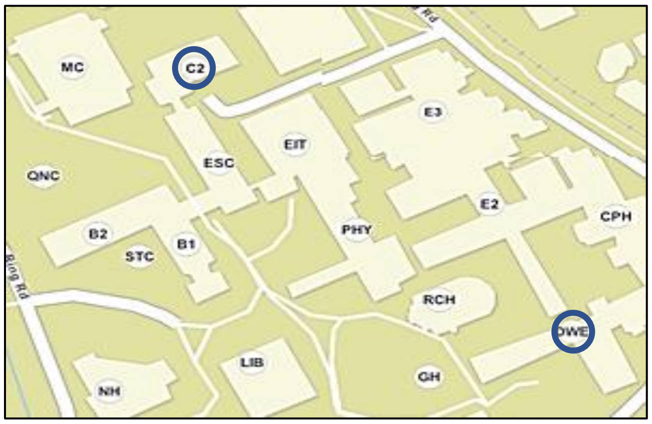


Fig. 4.10: Case Study Buildings

Example 1: Douglas Wright Engineering (DWE) Building

To further show the potential of the proposed method, the Douglas Wright Engineering (DWE) building was selected as a case study and investigated in more detail. A total of 494 images were collected to be used in this case study. First, the images were manually examined to determine whether they include a defect. Then, all 494 images were used as inputs for CNN1 responsible for defect detection. Detection Results of CNN1 are in Table 4.4.

Table 4.4: Detection Results of CNN1

		True Label	
		No Defect	Defect
Predicted Label	No Defect	383	--
	Defect	9	102

As seen in Table 4.4, no defects were missed by the CNN1, but there are 9 images (3.6%) that were misclassified as defects even though they were not. Next, the 111 images that were initially classified as "defects" were used as input for the second CNN responsible for defect type classification. Results are shown in Table 4.5.

Table 4.5: Detection Results of CNN2

		True Label		
		Ponding	Vegetation	No defect (misclassification)
Predicted Label	Vegetation	8	56	9
	Ponding	38	--	--

Example 2: Chemistry 2 (C2) building

The Chemistry 2 building (C2) was then selected as a second case study. A total of 490 images were collected. First, the images were manually examined to determine whether they include a defect. Then, all 490 images were used as inputs for the first CNN (CNN1) responsible for defect detection. Detection Results of CNN1 are in Table 4.6.

Table 4.6: Detection Results of CNN1

		True Label	
		No Defect	Defect
Predicted Label	No Defect	327	--
	Defect	48	116

As seen in Table 4.6, 48 images were misclassified as defects even though they were not (Examples in Fig. 4.11). Reasons for that include shadows (Fig. 4.6a) and areas where there is a change in the surface texture (Fig. 4.11b). It is important to mention that no defects were missed by CNN1. Next, the 164 images that were initially classified as "defects" were used as input for the second CNN responsible for defect type classification. Results are shown in Table 4.7.

Table 4.7: Detection Results of CNN2

		True Label			
		Ponding	Vegetation	Flashing	No defect (misclassification)
Predicted Label	Vegetation	--	97	--	8
	Ponding	--	--	17	40

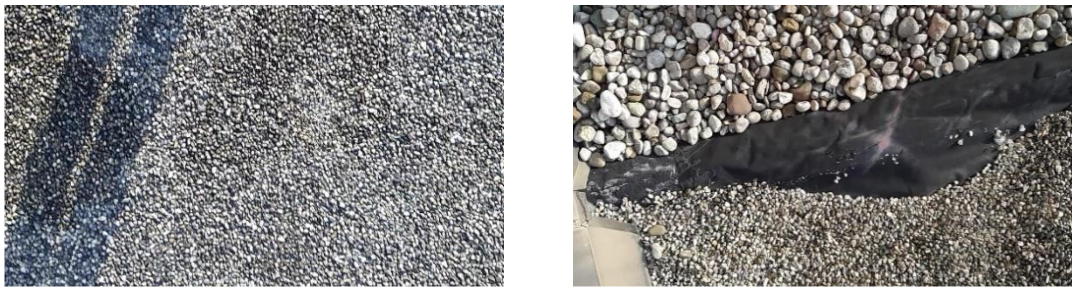

(a) Shadows (b) Change in Surface Texture

Fig. 4.11: Examples of misclassifications

An important observation is that all the images that were misclassified as "defects" because shadows were present were eventually classified as a ponding defect, while the ones that had a change in the surface texture were classified as a Vegetation defect. There were images that showed flashing defects (example in Fig. 4.12). Since there was no specific category for flashing defects due to the absence of sufficient images to properly train the model, these images were eventually classified as ponding defects. Finally, it was noticed that the lower the severity of the vegetation, the higher the probability of it being misclassified.

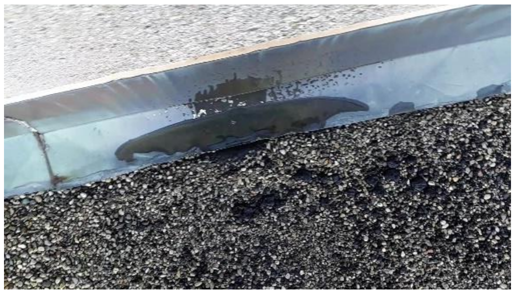


Fig. 4.12: Example of a Flashing Defect

Table 4.8 compares the classification results of the proposed two-phased system to that of a single-phase multiclassifier. As expected, it can be seen in Table 4.8 that the accuracy of the proposed model is superior to that of the single-phase model. Also, the number of false negatives (i.e., images with defects that were misclassified to include no defects) has increased in the case of the single-phase model. This is problematic to asset management personnel as underestimating the severity of the asset would delay the delivery of the necessary repairs, which increases the risk of the asset failure.

Table 4.8: Comparison of Results between Two-Step and Single-Step Classifiers

	Two-Step Classifier	**Single-Step Multiclassifier**
No. of Images	490	490
Correct Classifications	442	403
Accuracy	90%	82%
False Negatives (i.e., Missed Defects)	--	36

4.6.2 Campus-Wide Deployment

The model was applied to inspect the roofs of the 21 buildings highlighted in Fig. 3.4. Table 4.9 shows the full results of the two-phase model. For each roof, the total number of images collected and the number of images detected as defects (results of CNN1) are shown in columns 2 and 3, respectively. Column 4 shows, as a percentage, how much of the collected images have been classified as non-defective. This can be used as an indicator for the overall roof integrity as shown in the heat map in Fig. 4.13. The results of the second phase are shown in columns 5-8. The number of images that exhibit each defect type are in columns 5 and 6. As the size of each picture frame is known (1.5X1.12m), the number of images (columns 5 and 6) are then used to calculate the total defect area, listed in columns 7 and 8.

According to the inspected defects, there are two possible ways to repair a roof. First, a full replacement for the defective area. Such replacement is supposed to restore the roof to its original condition (i.e., 100%) and clear all defects, regardless of type. The second method is to simply clean up the vegetation accumulation. This method is less costly, but it only removes vegetation defects and leaves the ponded areas in their existing condition. As such, the improvement obtained from cleaning up vegetation can be calculated as *improvement due to replacement * (area of vegetation defect / total defected area)*.

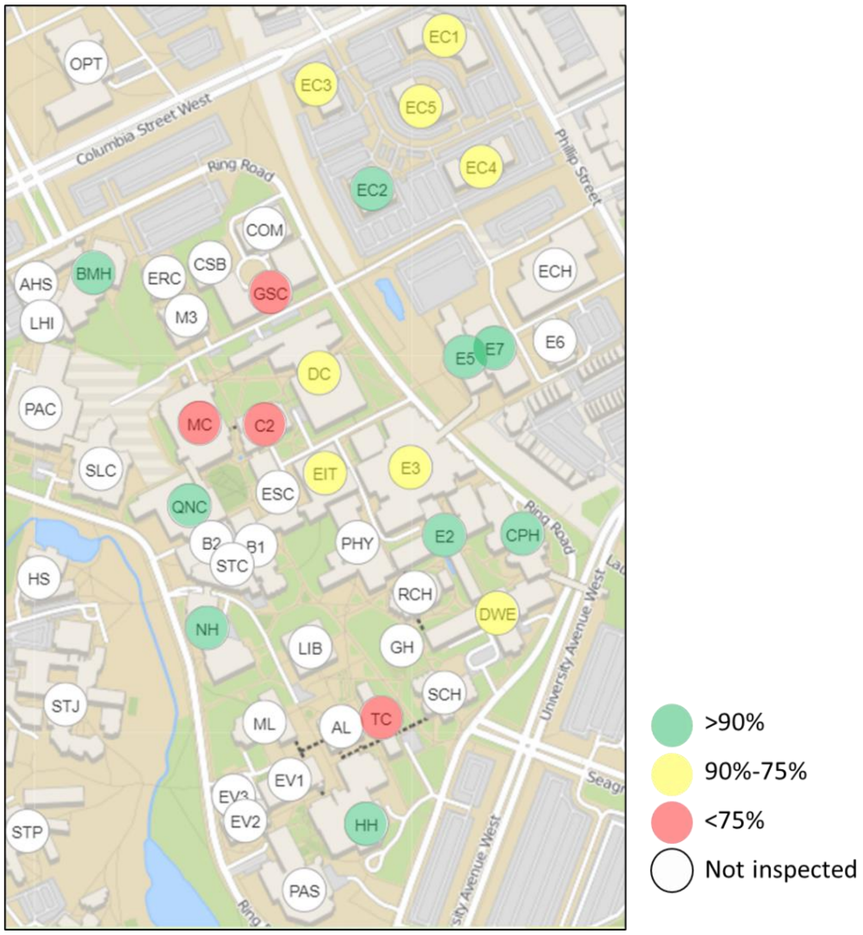


Fig. 4.13: A heat Map of University of Waterloo Campus Buildings According to the Integrity of their Roofs (based on data from Table 4.9: Column 4)

Table 4.9: Results of the Proposed Two-Phase Model, Applied to Roofs of 21 Buildings

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Building	**Total No. of Images**	**No. of images with defect**	**% of no defect images**	**No. of Ponding Images**	**No. of Vegetation Images**	**Ponding Area (m^2)**	**Vegetation Area (m^2)**
BMH	507	23	95%	23	0	38.64	0
C2	490	164	67%	58	106	97.44	178.08
CPH	436	9	98%	7	1	11.76	1.68
DC	816	108	87%	62	44	104.16	73.92
DWE	494	111	78%	38	73	63.84	122.64
E2	542	31	95%	13	15	21.84	25.2
E3	316	35	89%	2	22	3.36	36.96
E5	727	10	99%	3	3	5.04	5.04
E7	646	51	92%	20	17	33.6	28.56
EC1	646	84	87%	55	29	92.4	48.72
EC2	748	53	93%	16	33	26.88	55.44
EC3*	822	204	75%	102	58	171.36	97.44
EC4	673	74	89%	52	20	87.36	33.6
EC5	490	41	92%	22	15	36.96	25.2
EIT	245	35	86%	22	13	36.96	21.84
GSC	178	95	47%	89	6	149.52	10.08
HH	243	22	91%	19	3	31.92	5.04
MC*	932	298	68%	246	52	413.28	87.36
NH	401	35	91%	15	10	25.2	16.8
QNC	404	19	95%	15	2	25.2	3.36
TC*	626	274	56%	151	123	253.68	206.64

*: Snow buildup and defrosting have led to an increased no. of defected images

It is assumed that the cost for replacing the defected area of the roof is $300 per square meter of defected area, while the vegetation cleanup cost is $100 per square meter of vegetation defect. As such, the cost for fixing each roof can be calculated based on the size of the defected area calculated in columns 7 and 8 in Table 4.9. With the data in column 4 in Table 4.9 being used to indicate the existing roof condition, a simple optimization problem was then developed to maximize the value of the rehabilitation efforts (difference between existing and post-repair roof conditions) while abiding by budgetary constraints (budget limit = $400,000). The formulation of this problem can be seen in Figs. 4.14 and 4.15. Fig. 4.15 shows that, according to the existing budget limits, all buildings can undergo rehabilitation. However, the method of rehabilitation will differ depending on the building condition and the type of existing defects. Only 13 buildings will undergo full replacement, while vegetation cleanup is sufficient for the other 8 buildings. The total cost of the rehabilitation efforts is $391,608 (less than the $400,000 budget limit) and improves the overall condition of the buildings by an average of 12%.

Decision Variables:

R_i: *Perform full Replacement at Roof i* C_i: *Perform cleanup at Roof i* **(Binary)**

Objective Function:

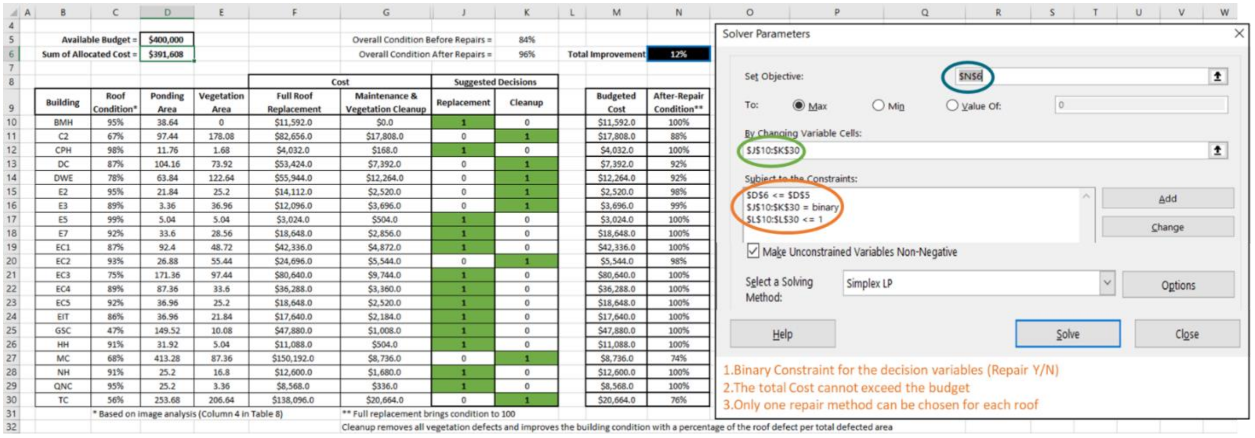
$$Max\ Z = \sum_{21}^{i=1} R_i * Condition\ after\ Replacement(i) + C_i * Condition\ after\ Cleanup(i) - Existing\ Condition\ (i)$$

Constraints:

$$\sum_{21}^{i=1} R_i * Cost\ of\ Replacement(i) + C_i * Cost\ of\ Cleanup(i) \leq Budget$$

$R_i + C_i \leq 1$ *For all i* (Conduct only one form of repair for each roof)

Fig. 4.14: Mathematical Formulation of Solver Setup



Available Budget = $400,000
Sum of Allocated Cost = $391,608

Overall Condition Before Repairs = 84%
Overall Condition After Repairs = 96%
Total Improvement 12%

				Cost		Suggested Decisions			
Building	Roof Condition*	Ponding Area	Vegetation Area	Full Roof Replacement	Maintenance & Vegetation Cleanup	Replacement	Cleanup	Budgeted Cost	After-Repair Condition**
BMH	95%	38.64	0	$11,592.0	$0.0	1	0	$11,592.0	100%
C2	67%	97.44	178.08	$82,656.0	$17,808.0	0	1	$17,808.0	88%
CPH	98%	11.76	1.68	$4,032.0	$168.0	1	0	$4,032.0	100%
DC	87%	104.16	73.92	$53,424.0	$7,392.0	0	1	$7,392.0	92%
DWE	78%	63.84	122.64	$55,944.0	$12,264.0	0	1	$12,264.0	92%
E2	95%	21.84	25.2	$14,112.0	$2,520.0	0	1	$2,520.0	98%
E3	89%	3.36	36.96	$12,096.0	$3,696.0	0	1	$3,696.0	99%
E5	99%	5.04	5.04	$3,024.0	$504.0	1	0	$3,024.0	100%
E7	92%	33.6	28.56	$18,648.0	$2,856.0	1	0	$18,648.0	100%
EC1	87%	92.4	48.72	$42,336.0	$4,872.0	1	0	$42,336.0	100%
EC2	93%	26.88	55.44	$24,696.0	$5,544.0	0	1	$5,544.0	98%
EC3	75%	171.36	97.44	$80,640.0	$9,744.0	1	0	$80,640.0	100%
EC4	89%	87.36	33.6	$36,288.0	$3,360.0	1	0	$36,288.0	100%
EC5	92%	36.96	25.2	$18,648.0	$2,520.0	1	0	$18,648.0	100%
EIT	86%	36.96	21.84	$17,640.0	$2,184.0	1	0	$17,640.0	100%
GSC	47%	149.52	10.08	$47,880.0	$1,008.0	1	0	$47,880.0	100%
HH	91%	31.92	5.04	$11,088.0	$504.0	1	0	$11,088.0	100%
MC	68%	413.28	87.36	$150,192.0	$8,736.0	0	1	$8,736.0	74%
NH	91%	25.2	16.8	$12,600.0	$1,680.0	1	0	$12,600.0	100%
QNC	95%	25.2	3.36	$8,568.0	$336.0	1	0	$8,568.0	100%
TC	56%	253.68	206.64	$138,096.0	$20,664.0	0	1	$20,664.0	76%

* Based on image analysis (Column 4 in Table 8)

** Full replacement brings condition to 100

Cleanup removes all vegetation defects and improves the building condition with a percentage of the roof defect per total defected area

Solver Parameters

Set Objective: $N$6

To: [✓ Max] Min Value Of: 0

By Changing Variable Cells: $J$10:$K$30

Subject to the Constraints:
$D$6 <= $D$5
$J$10:$K$30 = binary
$L$10:$L$30 <= 1

[✓] Make Unconstrained Variables Non-Negative

Select a Solving Method: Simplex LP

1.Binary Constraint for the decision variables (Repair Y/N)
2.The total Cost cannot exceed the budget
3.Only one repair method can be chosen for each roof

Fig. 4.15: Screen Capture of the Solver Setup

4.7 Potential Improvements

One of the potential improvements is collecting images that cover larger areas to reduce the data collection burden. In that case, simply measuring the area of defect as the total area that the image covers will significantly overestimate the defect size and, consequently, the criticality of the building condition. To that end, homography can be used to extract the area of defect from the captured image. Experiments to inspect the effectiveness of homography have been conducted on images of pavements that show cracks. In this experiment, a Tim Hortons gift card of known dimensions was used to calibrate the image. Then, the start and end points of the individual cracks were manually selected, and the lengths were automatically calculated. Fig. 4.16 shows the true (Fig. 4.16a) and calculated (Fig. 4.16b) crack lengths, highlighted on the individual cracks, while Table 4.10 shows the calculation results and errors. Table 4.10 shows that the average error between the ground truth and the image-based measurements is 4.23%, thus demonstrating its potential for future study and development.

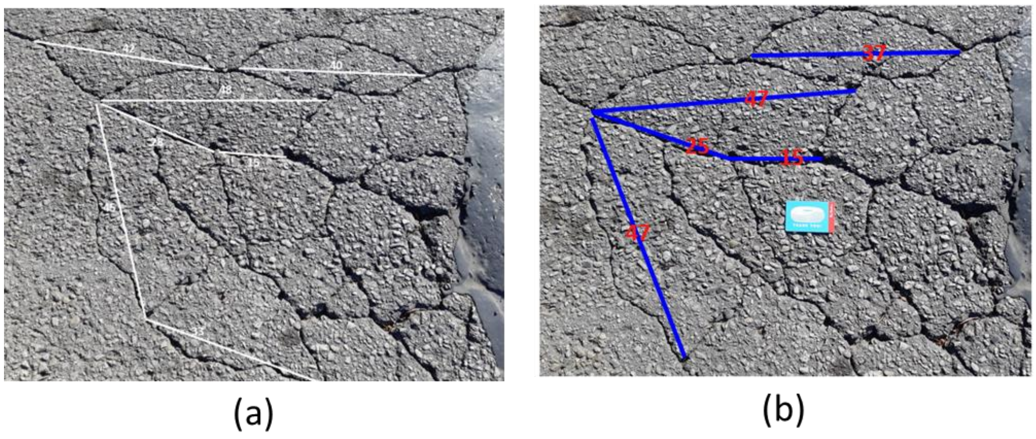


Fig. 4.16: Ground Truth (a) and Homography-Calculated (b) Crack Lengths

Table 4.10: Comparison Between Image-Based Crack Measurements (Fig. 4.16b) And Their Ground Truth Counterparts (Fig. 4.16a).

#	Image-Based (cm)	Actual (cm)	Error (%)
1	37	40	7.5
2	47	48	2.1
3	26	28	7.1
4	15	15	0
5	47	45	4.44

The same technique was used to calculate the area of cracks. A polygon surrounding the cracks was created whose points were manually picked from the image (Fig. 4.17). Then the shoelace algorithm (Equation 4.1) was used to calculate the area of the polygon where homography was used to translate the picked points from the image plane coordinates to the real-world plane coordinates. The overall area came out to be 4652 cm^2, representing approximately 75% of the overall area. It is noted for these preliminary experiments, the points marking the boundaries of the cracked area were picked manually. This will be automated in the remaining part of the work.

[4.1] $$A = |x_1y_2 + x_2y_3 + \ldots\ldots + x_{n-1}y_n - x_2y_1 - x_3y_2 - \ldots\ldots - x_ny_{n-1} - x_1y_n|$$

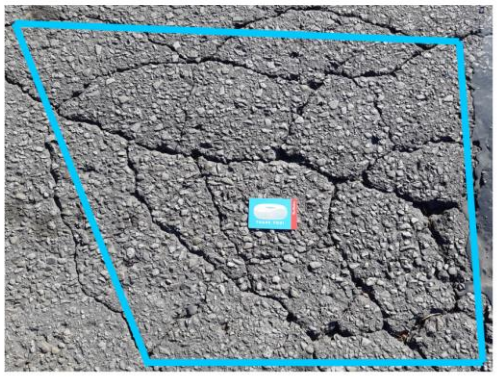


Fig. 4.17: Polygon used to Calculate the Overall Area of Cracks

Other potential improvements include the use of infrared imaging to detect hidden defects, using image registration information to pinpoint the defect location within the structure, and connecting the defect information obtained from the image analysis to a BIM model.

4.8 Conclusion

This chapter has presented the first module of the roof rehabilitation framework; a two-step CNN model is proposed to detect roofing defects based on 2D images, as well as classify them according to their type. More than 5,500 images from different roofs across the University of Waterloo campus were used to train and test the model's classification power. The model has shown promising results, achieving approximately 95% accuracy level during the detection phase, 97% accuracy during the classification phase, and exhibiting no major biases. The images used for the model are of constant real-life dimensions (1.5x1.12 m) which helps decision-makers classify the roofs according to the extent of the defects (the more the images of the defects, the bigger the size of the damage and the more critical the roofing condition) as well as quantify the damage size for appropriate rehabilitation work packaging and estimating. The images are originally obtained from video recordings which means that the model, if given the proper computational resources, is capable of performing real-time detections. This is further proven by the model performance speed (0.08 sec/image for both phases). The proposed model has surpassed other models existing in the literature and it is expected that the model would remain on par with, if not surpass, existing notable CNN architectures used in the industry such as ResNet and VGG. For example, VGG system architecture includes 12 layers while the proposed model has only 9. Hence, it is expected that the proposed model would be the faster performer, with little to no accuracy drawbacks.

The model, in its current form, is faster and more efficient than manual inspections. The total length of videos captured for all buildings is about 90 minutes. As mentioned earlier, the CNN takes around 0.08 seconds per frame to complete its analysis. This means that all 11,000 images can be analyzed in around 15 minutes. Assuming an extra minute is needed to adjust the camera for each building, and an extra minute is needed to extract the frames from each video. Then the total time needed to complete the entire roof inspection process for all 21 buildings is less than three hours.

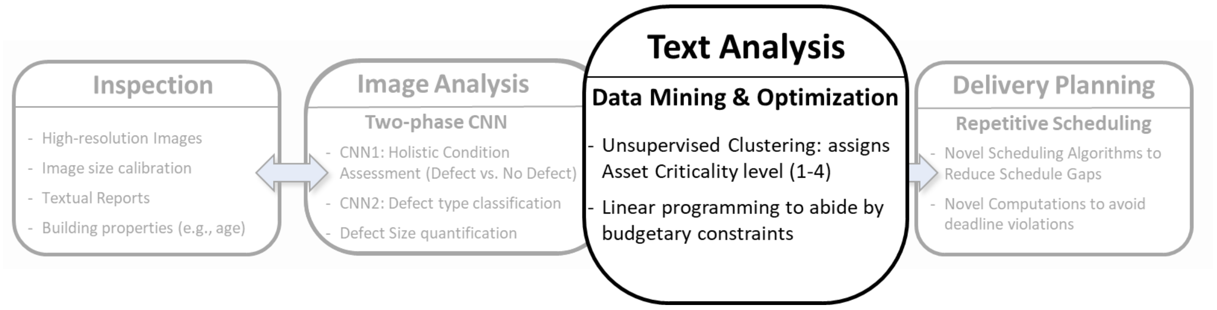
Inspecting a building using the conventional methods typically takes around 3 hours (Kamarah 2019) and for every hour of on-site inspection, 3 hours are spent in the office to analyze and document the inspection observations (Abou Shaar 2012). Assuming the onsite visits will remain the same, each building will require nine hours of office work to analyze and document the collected data to provide the final assessment. Assuming that one of the nine hours are dedicated for analysis of roofing inspection data, analyzing the data for the 21 buildings would take a total of 21 hours. As such, the proposed method saves more than 18 hours of work (over 85%). Faster inspections mean that they can be performed more frequently which makes the inspection data more updated and relevant to the decision-making process. Finally, the assessments performed by the proposed model are consistent, objective, and truly reflect the condition of the asset.

Part of the inspection process, currently, includes interviews with building representatives and reviewing old documents to gain insights about the current state of the building (Mostafa et al. 2021). Analyzing this information is discussed in chapter 5 and added to the outcome of the CNN analysis for a more accurate assessment and fund allocation data mining framework.

Chapter 5: Data Mining for Prioritization and Fund Allocation

5.1 Introduction

As shown in Fig. 5.1, this chapter introduces the second module of the roofing rehabilitation framework; a data-mining model that relies on textual data describing different building parameters such as age and description to prioritize the different buildings according to their need of the rehabilitation work. To reduce the effect of human bias, the proposed model uses unsupervised clustering to group the buildings into four categories according to their condition and need for repairs. Since the data acquisition process has previously been explained in section 3.4, this chapter starts by discussing the different processes implemented to extract useful information from the acquired reports. Then, the different clustering approaches are then explained along with the author's conclusions and recommendations. Finally, the impact of integrating defect information from images (Chapter 4) is introduced and an optimization framework is implemented accordingly.



Inspection
- High-resolution Images
- Image size calibration
- Textual Reports
- Building properties (e.g., age)
Image Analysis
Two-phase CNN
- CNN1: Holistic Condition Assessment (Defect vs. No Defect)
- CNN2: Defect type classification
- Defect Size quantification
Text Analysis
Data Mining & Optimization
- Unsupervised Clustering: assigns Asset Criticality level (1-4)
- Linear programming to abide by budgetary constraints
Delivery Planning
Repetitive Scheduling
- Novel Scheduling Algorithms to Reduce Schedule Gaps
- Novel Computations to avoid deadline violations

Fig. 5.1: Second Module of the Proposed Framework

5.2 Keyword Selection for Defect Categorization

The first step was examining the list of potential defects originally developed by TDSB and samples of the inspection reports. For roofs, TDSB has a list of eight possible defects, stated as a general guide for inspectors, as shown on the top part of Fig. 5.2. Some of these defects overlap, for example, "Water penetration" and "Leaks at penetrations". Furthermore, after examining samples of the inspection reports, it was found that the descriptions do not refer to the original defect list, rather, the inspector's own explanation of the various defects. An example of the nature of the information provided by TDSB inspectors can be found in Fig. 3.5. To help with text mining for matching statements in the inspector's descriptions to a particular damage category, roof defects have been summarized into four generic defect categories: Damage, Leak, Drainage, and Obsolete, as shown at the bottom of Fig. 5.2. The category “Damage” includes all damaged roofing elements. Thus, the more the “Damage” text appears in the event descriptions, the higher the extent of roof damage. It was also found that some rehabilitation events were scheduled not because of a specific defect, but because the roof has exceeded its service life. These were accounted for by including them under the “Obsolete” category. Once the generic defect categories were defined, the sample of the roof event descriptions was further examined to find unique keywords that, when seen in the text, could indicate the type of damage associated with the roof under inspection. A sample of the developed keywords is shown below each defect category in Fig. 5.2.

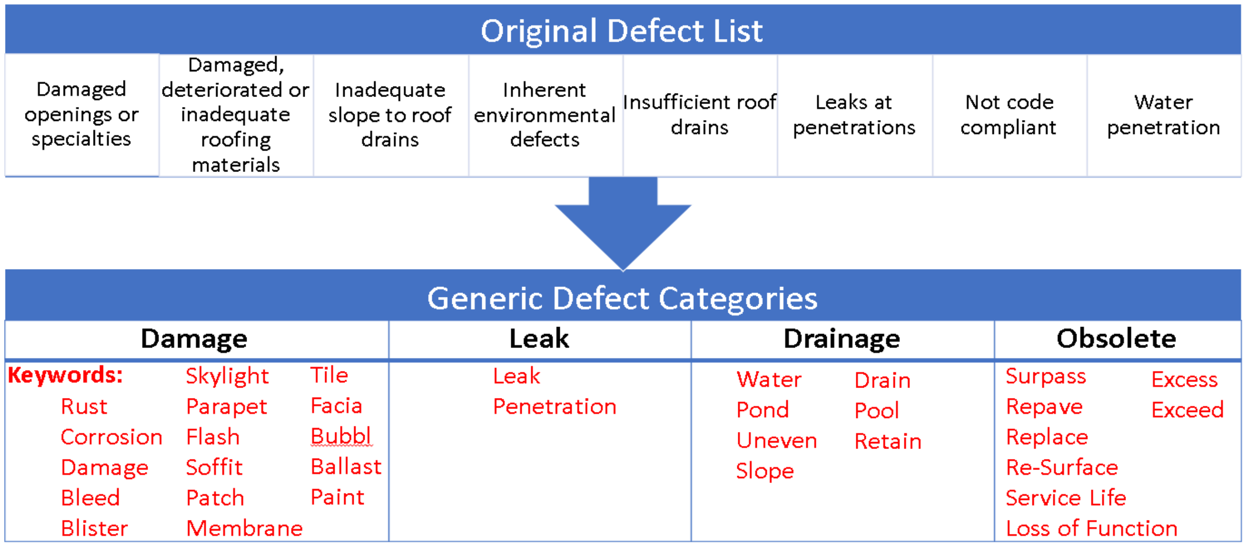


Fig. 5.2: Generic Defect Categories and Related Keywords

Once the keywords were defined for the 4 categories of defects, a VBA macro was developed to search the “Event Justification”, “Event Description” and “Image Comment” text fields of every rehabilitation event with all the keywords in Fig. 5.2. The macro sums the number of times the keywords of each Defect category are used (referred to as Defect_Count) in each event, as an indication of the severity of the defect. In addition, the overall sum is also tallied (referred to as “Total_Count”). As such, the higher the Total_Count, the worse the condition of the roof. The pseudocode for calculating the Total Count for each building roof is shown in Fig. 5.3.

```
For each **Rehabilitation_Event**
  **Full_Text** = Event_Description + Event Justification + Image_Comment
  Total_Count = 0
  For i = 1 to 4                ' Defect categories 1 to 4
        Defect_Count (i) = 0
        For each **Word** in **Keywork_List (i)**
            If **Word** in **Full_Text** Then
                Defect_Count (i) = Defect_Count (i) + 1
                Total_Count = Total_Count + 1
            End If
        Next Word
  Next i
  Save **Defect_Count (1 to 4)**
  Save **Total_Count**
Next Event
```

Fig. 5.3: Pseudocode for Text Mining

Based on the results of the text mining analysis, 73% of all roofing events indicate roofing damage and over 98% of those events are deemed to be of high priority and require a total cost of $87.3 million. Since this amount of required funds is much higher than the budget limit, it is necessary to go deeper and identify the top critical roofs and their rehabilitation funding needs.

5.3 Principal Component Analysis (PCA)

Before attempting data clustering, Principal Component Analysis (PCA) was performed to reduce the dimensionality of the dataset. PCA is a data reduction technique that produces a new set of variables (principal components) that better capture the variations in the data (Wilks 2011). It was noticed that there is a high correlation between the *"Event Type"* and *"Event Priority"* categorical variables. For example, all events of the type *Major Repair* had the priority of *High*. As such, PCA was performed in an attempt to replace these two variables with a single representative attribute. Prior to conducting PCA, the variables had to be binarized, as shown in Table 5.1.

Table 5.1: Binarization of Event-Type and Event-Priority Values

EVENT TYPE		**EVENT PRIORITY**	
REPLACE	Major Repair	High	Medium
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1

The binary values in Table 5.1 were then used for PCA using the scikit-learn package present in the python programming environment, which has a built-in module for PCA. The analysis resulted in identifying a single attribute (Principal Component), called *T&P* in this paper, that represents both the event type and event priority attributes. Based on the PCA analysis results, the *T&P* value associated with each roof rehabilitation event was calculated and used as a parameter to facilitate efficient clustering of the data.

5.4 Implementation and Results of Clustering

To implement the various clustering methods, the well-known WEKA (2020) software was used. WEKA is a java-based open-source software developed by the University of Waikato in New Zealand, and has an extensive collection of unsupervised and supervised data mining and machine learning algorithms (data preprocessing, classification, clustering, regression, etc.). The three clustering methods (Canopy, K-Means, and Farthest-First) were then applied to the data of the roofing rehabilitation events, and the performance of the three methods were compared to one another.

To apply the three clustering techniques, three important parameters were used: the *Total_Count* of defects for each event; the *Event Age*; and the *T&P* parameter determined by the Principal Component Analysis (PCA) discussed earlier. Based on the results of the different clustering methods, the number of roofing rehabilitation events assigned to each cluster by the various clustering methods is shown in Table 5.2. It is important to mention that, to ensure consistency among the different algorithms, the cluster order has been presented such that the higher the cluster number, the more critical the asset condition (e.g., Cluster2 has more critical assets than Cluster1). This was done manually after examining the cluster information produced as part of the output of each clustering algorithm. Other clustering algorithms (e.g., decision trees, expectation maximization) were inspected, but their performance and explainability were found to be inferior to the three algorithms studied here.

Table 5.2: Number of Events in each Cluster based on Different Clustering Methods

	Canopy	**FF**	**Kmeans**
Cluster1	2%	2%	53%
Cluster2	4%	4%	24%
Cluster3	6%	20%	10%
Cluster4	88%	74%	13%

Looking at Table 5.2, it is possible to see that the algorithms assign most of the data to one or two clusters. For example, the EM algorithm assigned almost all the data to clusters 3 and 4, while the Kmeans assigned more than half the data points to Cluster1. With Cluster4 representing the most critical events that are in most need of rehabilitation funds, the Kmeans clustering algorithm proves to be the most useful in terms of aiding the decision-maker to abide by the smallest budget limit. It was also noticed that the events that were classified by the Kmeans algorithm to be of highest priority (i.e., Cluster4), received the same classification from the other algorithms. To examine the results of the Kmeans algorithm more closely, a 3D visual representation of the datapoint assigned to each cluster is shown in Fig. 5.4, with the data of Cluster 4 highlighted. From this figure, it seems that the Kmeans algorithm assigns the data points to their respective clusters mainly based on *Event_Age* and *Total_Count* parameters while allocating little weight to the effect of the third parameter (*T&P*).

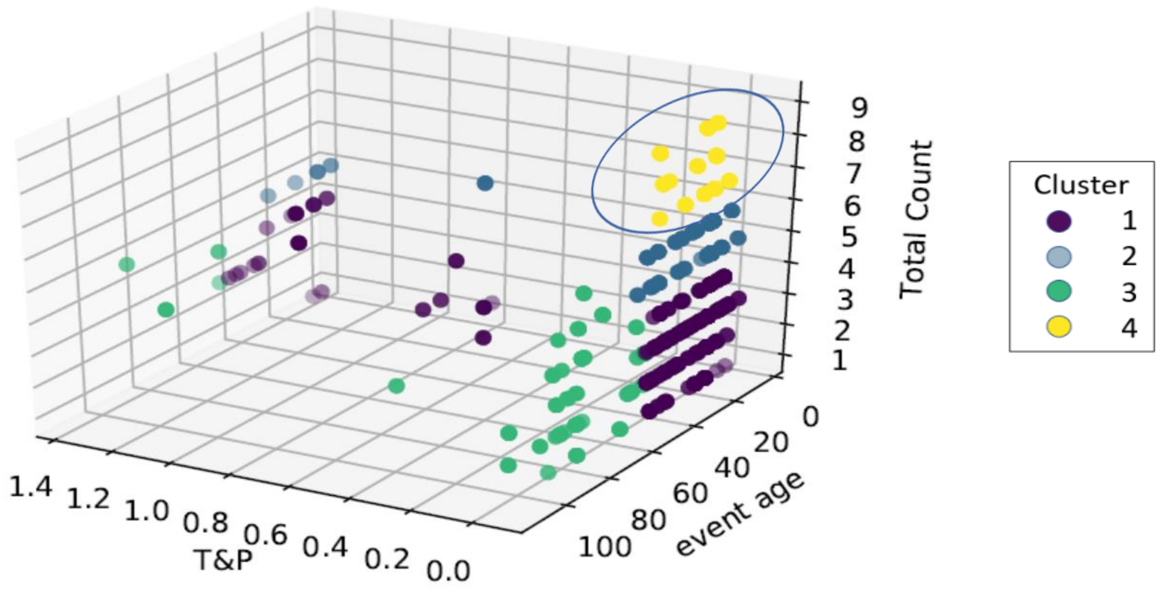


Fig. 5.4: Clustering Results of the Kmeans Algorithm

5.5 Incorporating Image-based Defect Information

First of all, the textual analysis presented earlier was done on an event-by-event basis. However, the image-based analysis conducted in Chapter 4 was done on a building-by-building basis. Hence, to combine the two approaches, the event information obtained from the TDSB data was aggregated to represent the condition of every school. Next, damage information as a percentage of the total roofing area was obtained from the analysis conducted on the University of Waterloo buildings and used as an assumption for the range of defect size the different TDSB buildings would experience. Finally, the

Kmeans clustering technique was reapplied on the new dataset (school-based, including visual defect information) using three parameters; School age, Total Count of defect keywords, and visual defect information represented as a percentage of the total roof area. For notational convenience, the three parameters are referred to as *Age, T_C,* and *% of defects*. The *T&P* parameter used in previous clustering attempts (section 5.4) was disregarded for two reasons; first, to reduce the model bias as the components of the *T&P* parameters are based on the inspector's personal judgment, and second, because the conducted analysis showed that the *T&P* parameter plays a little role in the clustering process (refer to Fig. 5.4 and the supporting narrative). The total number of rehabilitation events required for each school was not included in the clustering analysis because this piece of information will not be available for the model in the future as the inputs will become age and damage description information (per building).

Two different approaches were implemented regarding aggregating the different rehabilitation events. The first approach was to sum the *Defect Count* values for all the events for a given school and present that as the new *T_C* value of the school (referred to as *Sum of T_C*). The second approach was to set the new *T_C* value of the school to be equal to the average *Defect Count* values for the events concerning that school (referred to as *Average of T_C*). The results for the clustering attempts using the two approaches can be visualized in Figs. 5.5 and 5.6, respectively.

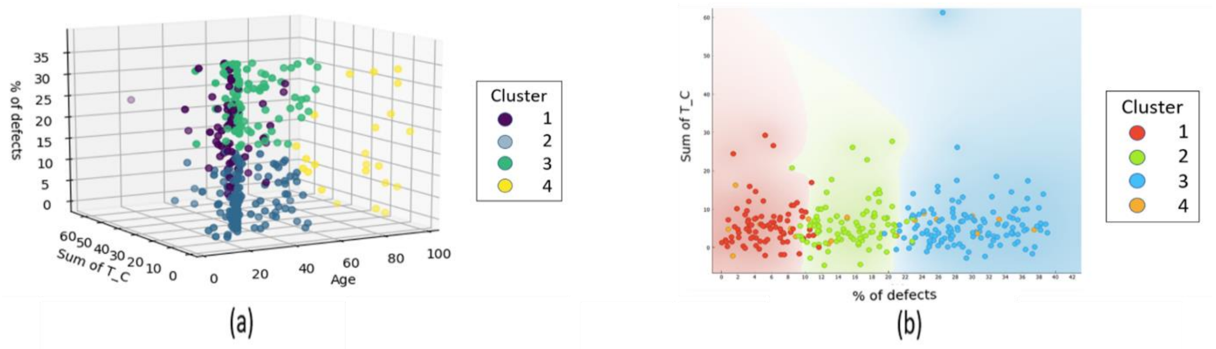


Fig. 5.5: Clustering Results using *Sum of T_C* Values

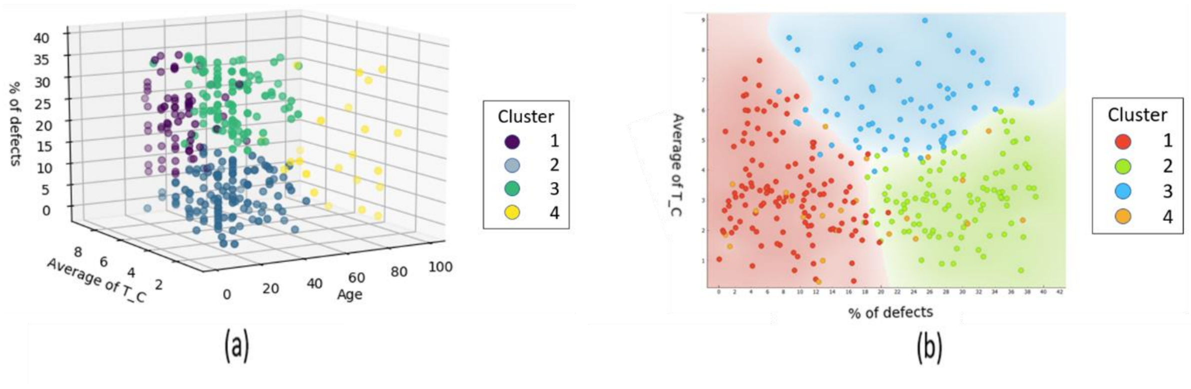


Fig. 5.6: Clustering Results using *Average of T_C* Values

Figs. 5.5a and 5.6a show a 3D visualization of the clustering results using both approaches. It can be seen that both approaches agree on separating the old buildings (*Age* > 50) into a separate cluster, then used the *% of defects* and *T_C* data to create the other two clusters. As such, Figs. 5.5b and 5.6b show a 2D visualization of the clustering results based on the *% of defects* and *T_C* data. It can be seen that in the case of using the *Sum of T_C* parameter for clustering (Fig. 5.5b), the clustering occurs primarily based on the *% of defects* parameter and the *T_C* parameter is almost ignored. However, when *Average of T_C* is used (Fig. 5.6b), the clustering takes into account both parameters. This may be attributed to the relatively uniform spread of the datapoint along the *Average of T_C* axis as opposed to their spread along the *Sum of T_C* axis. The number of schools and total rehabilitation costs required based on both clustering attempts are in Table 5.3. Although the cost required to repair all schools in cluster 4 (most critical) is 16% more when *Average of T_C* is used as the clustering parameter, the total cost to repair the schools in the highest two categories (cluster4 and cluster3) is 30% less, and the number of schools that need to undergo immediate rehabilitation is almost 50% less. As such, clustering results based on *Average of T_C* are used for testing the optimization model in the later stages of this work.

Table 5.3: No. of Schools and Total Rehabilitation Cost Required for Each Cluster based on Different Clustering Parameters

	Clustering using *Sum of T_C*		**Clustering using *Average of T_C***	
Cluster	**No. of Schools**	**Total Cost**	**No. of Schools**	**Total Cost**
1	82	$30,044,784	125	$41,903,994
2	89	$32,210,200	101	$35,802,763
3	127	$51,436,568	67	$34,944,195
4	18	$6,054,644	23	$7,095,244
Grand Total	**316**	**$119,746,196**	**316**	**$119,746,196**

5.6 Fund Allocation Optimization

According to the latest operating budget for The Toronto District School Board (TDSB), \$31.4 million are being allocated for renewal events for all 550 schools under their jurisdiction (TDSB 2020). Assuming that 10% of the aforementioned budget is dedicated to roofing work, and the fact that only 400 schools were analyzed as part of the work presented in this chapter, the budget limit was set to be \$2.3 million. As such, an optimization problem was structured using the schools categorized in clusters 3 and 4 (mathematical formulation in Fig. 5.7, screenshot in Fig. 5.8). The objective is to improve the condition of the entire asset portfolio by selecting which schools to repair, without going over budget. The condition improvement gained from repairing each school corresponds to the cluster to which the school was assigned. For example, repairing a school from cluster4 yields 4 points, while repairing a school from cluster3 yields 3 points. Out of the 90 schools in clusters 3 and 4 combined (Table 5.3), the solver results recommended repairing only 20 schools to achieve the maximum improvement with the given budget. 12 schools from cluster 4 (out of 23) and 8 schools from cluster 3 (out of 67) were deemed most worthy of immediate repair to maximize the use of the current rehabilitation budget.

Decision Variables:

R_i: *Conduct Repairs for School i* **(Binary)**

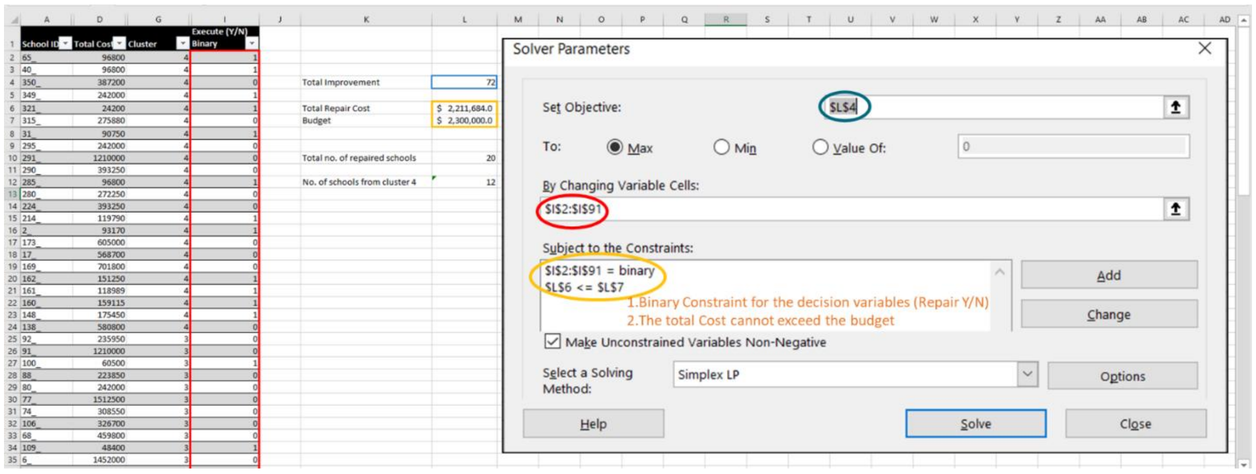
Objective Function:

$$Max\ Z = \sum_{90}^{i=1} R_i * Cluster\ Value\ for\ School\ (i)$$

Constraints:

$$\sum_{90}^{i=1} R_i * Cost\ of\ Repair(i) \leq Budget$$

Fig. 5.7: Mathematical Formulation of the Solver Setup



	A	D	G	I
1	School ID	Total Cost	Cluster	Execute (Y/N) Binary
2	65_	96800	4	1
3	40_	96800	4	1
4	350_	387200	4	0
5	349_	242000	4	1
6	321_	24200	4	1
7	315_	275880	4	0
8	31_	90750	4	1
9	295_	242000	4	0
10	291_	1210000	4	0
11	290_	393250	4	0
12	285_	96800	4	1
13	280_	272250	4	0
14	224_	393250	4	0
15	214_	119790	4	1
16	2_	93170	4	1
17	173_	605000	4	0
18	17_	568700	4	0
19	169_	701800	4	0
20	162_	151250	4	1
21	161_	118989	4	1
22	160_	159115	4	1
23	148_	175450	4	1
24	138_	580800	4	0
25	92_	235950	3	0
26	91_	1210000	3	0
27	100_	60500	3	1
28	88_	223850	3	0
29	80_	242000	3	0
30	77_	1512500	3	0
31	74_	308550	3	0
32	106_	326700	3	0
33	68_	459800	3	0
34	109_	48400	3	1
35	6_	1452000	3	0

K	L
Total Improvement	72
Total Repair Cost	\$ 2,211,684.0
Budget	\$ 2,300,000.0
Total no. of repaired schools	20
No. of schools from cluster 4	12

Solver Parameters

Set Objective: \$L\$4

To: [✓ Max] Min Value Of: 0

By Changing Variable Cells: \$I\$2:\$I\$91

Subject to the Constraints:
\$I\$2:\$I\$91 = binary
\$L\$6 <= \$L\$7

1.Binary Constraint for the decision variables (Repair Y/N)
2.The total Cost cannot exceed the budget

[✓] Make Unconstrained Variables Non-Negative

Select a Solving Method: Simplex LP

Fig. 5.8: Screen Capture of the Solver Setup

5.6.1 Sensitivity Analysis

The sensitivity of the fund allocation model to changes in the estimated repair cost for each school as well as the allowed budget are investigated and will be discussed in this subsection.

5.6.1.1 Sensitivity to Changes in Estimated Costs

Monte Carlo Simulation was used to investigate the impact of changes in the estimated repair costs on the funding decisions obtained by the model. To simulate the worst-case scenario, only cost increases were investigated in these simulations. A total of 50 iterations were conducted and in each iteration the cost of each school took a random value between 100% and 115% of the original estimated cost. Histograms showing the number of schools selected for repairs as well as the overall improvement are show in Figs. 5.9 and 5.10, respectively. Figs. 5.9 and 5.10 show that the same results obtained from the deterministic model discussed earlier (i.e., repairing 20 schools with total improvement = 72) is achieved approximately 85% of the time. This shows the resilience of the proposed model considering the studied variations (cost increase up to 15%).

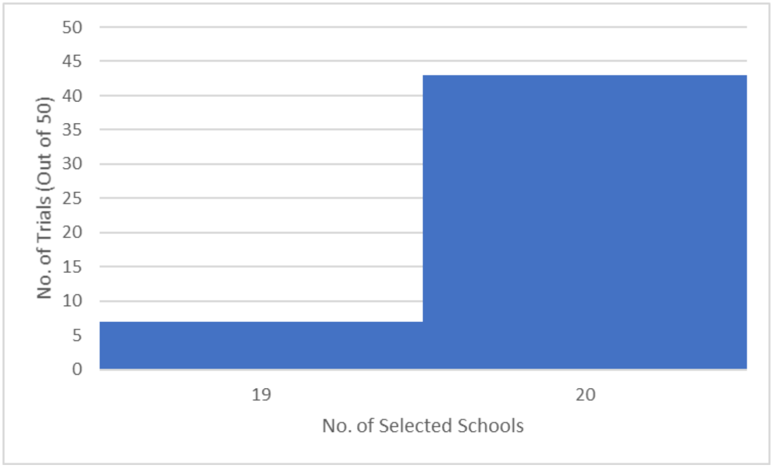


Fig. 5.9: Histogram for the Number of Schools Selected for Repairs

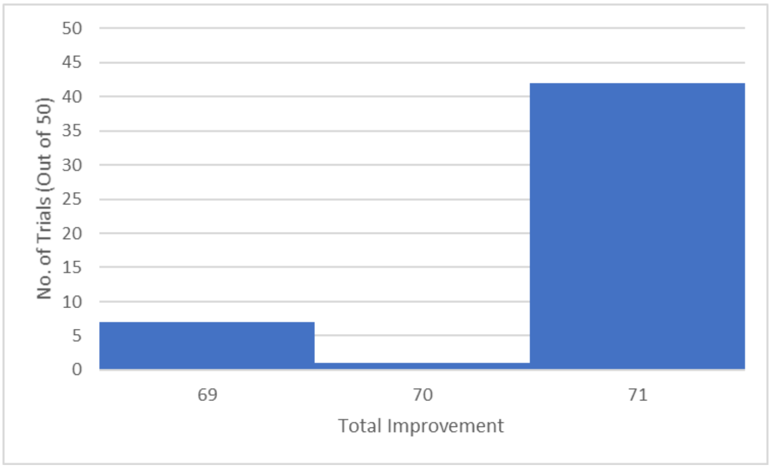


Fig. 5.10: Histogram for the Total Improvement Achieved by the Monte Carlo Simulations

To further Illustrate the resilience of the model, the individual results of the Monte Carlo simulations were investigated. Fig. 5.11 shows how many times each school was selected by the model through the different iterations. It can be seen from the Fig. that 15 schools were selected 100% of the time (50 out of 50 trials), five schools were selected more than 80% of the time, and only three schools (IDs: 349_, 295_, and 236_) were selected less than five times.

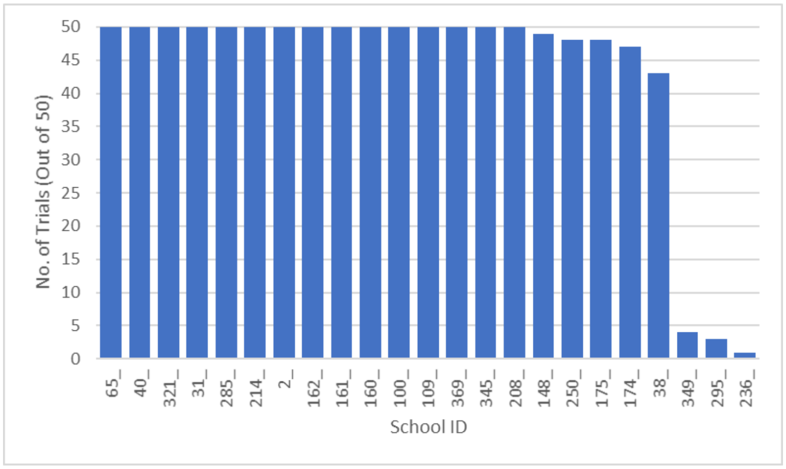


Fig. 5.11: Number of Times Each School was Selected for Repair (Out of 50 Iterations)

5.6.1.2 Sensitivity to Changes in the Allocated Budget

To study the sensitivity of the model to changes in the allocated budget, four different budget values were used as the financial constraint for the optimization model. The four values selected were in $100,00 decrements from one another. The results of the different optimization attempts are shown in Table 5.4. Table 5.4 shows the resilience of the model against changes in budget and the effectiveness of the optimization efforts, as 15% decrease in the budget only lead to a 5% decrease in the overall improvement of the asset portfolio, with only one school being left out compared to the results obtained using the original budget.

Table 5.4: Results of Sensitivity Analysis with Respect to the Allocated Budget

Budget	Total Improvement	No. of Schools
$ 2,300,000	72	20
$ 2,200,000	71	20
$ 2,100,000	69	19
$ 2,000,000	68	19

5.7 Conclusion

This chapter has presented the second module of the roofing rehabilitation framework; a data-mining model that relies on textual data describing different building parameters such as age and description to prioritize the different buildings according to their need of the rehabilitation work. In addition, the analysis technique adopted in this chapter has been combined with the pictorial data analysis presented in Chapter 4 to provide a comprehensive assessment. Accordingly, a linear optimization problem was formulated to select which schools should be repaired immediately to maximize the improvement of the entire asset portfolio while abiding by budgetary constraints.

Given the number of schools in need of rehabilitation, proper repetitive scheduling techniques will be implemented and novel scheduling computations and visualizations will be developed to assist with developing an efficient plan to execute the required rehabilitation tasks within the constraints of time and budget (e.g., Hegazy et al. 2020, 2021). Such implementation is discussed in more detail in Chapter 6.

Chapter 6: Delivery Planning and Scheduling

6.1 Introduction

As seen in Fig. 6.1, this chapter introduces the third and final module of the roofing rehabilitation framework; a repetitive scheduling-based approach to optimize the delivery planning and scheduling for the required rehabilitation works. This chapter starts by introducing the novel contributions in terms of repetitive scheduling visualizations, computations, and algorithms. These developments are then applied to a case study of a 20-unit rehabilitation project, based on the data analyzed in Chapter 5.

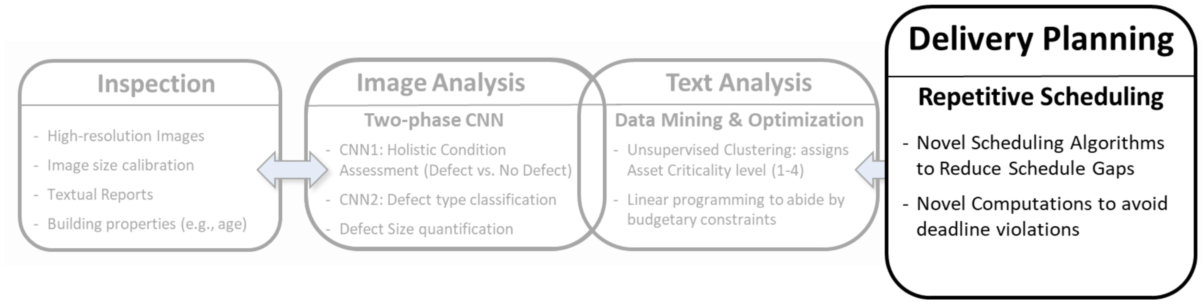

Inspection
- High-resolution Images
- Image size calibration
- Textual Reports
- Building properties (e.g., age)
Image Analysis
Two-phase CNN
- CNN1: Holistic Condition Assessment (Defect vs. No Defect)
- CNN2: Defect type classification
- Defect Size quantification
Text Analysis
Data Mining & Optimization
- Unsupervised Clustering: assigns Asset Criticality level (1-4)
- Linear programming to abide by budgetary constraints
Delivery Planning
Repetitive Scheduling
- Novel Scheduling Algorithms to Reduce Schedule Gaps
- Novel Computations to avoid deadline violations

Fig. 6.1: Third and Final Module of the Proposed Framework

6.2 Novel Visualizations (Duration-Distance Chart)

Typically, all repetitive scheduling methods present the schedule in one of two ways (Fig. 6.1): (1) the line of balance (LOB) visual, showing the activities of each unit on a separate horizontal line (i.e., the vertical axis is the unit index, i.e., unit 1, 2, or 3, etc.), similar to a bar chart; and (2) the flowline visual showing time versus the distance (on the vertical axis) that mark the start (ST) and finish (FN) locations of each unit (Fig. 6.1b).

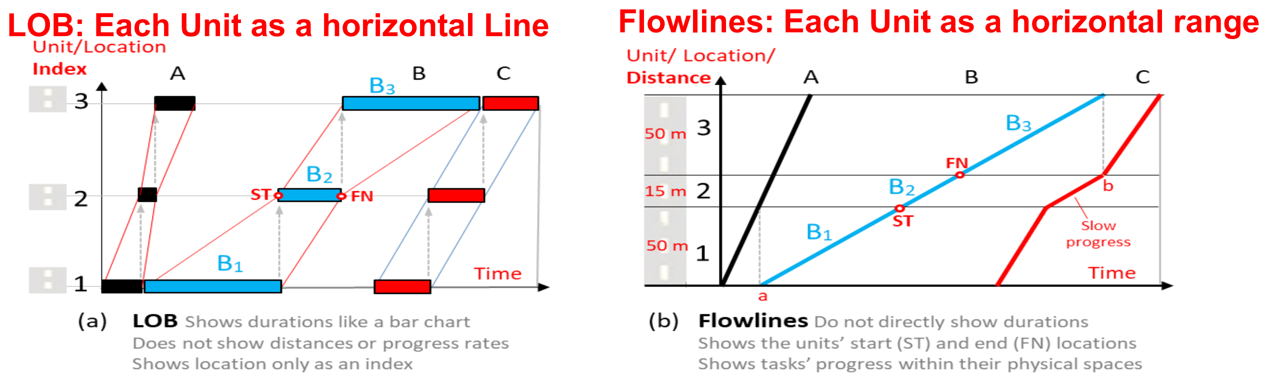

LOB: Each Unit as a horizontal Line
Unit/Location
Index
A
B
C
3
2
1
B3
B2
B1
ST
FN
Time
(a) LOB Shows durations like a bar chart
Does not show distances or progress rates
Shows location only as an index
Flowlines: Each Unit as a horizontal range
Unit/ Location/
Distance
A
B
C
50 m
15 m
50 m
3
2
1
B3
B2
B1
FN
ST
a
b
Slow progress
Time
(b) Flowlines Do not directly show durations
Shows the units' start (ST) and end (FN) locations
Shows tasks' progress within their physical spaces

Fig. 6.2: Differences between LOB and Flowline visuals

Fig. 6.2 shows a schedule of 3 sequential activities (A-B-C), each employing a single crew that moves along 3 non-identical units. Flowlines make use of the vertical axis to show the progress rate (slope) within each repetitive unit, as a function of both unit size (vertical projection) as well as required duration (horizontal projection). For example, looking at the LOB diagram (Fig. 6.2a) would give the false impression that the amount of work required for activity C in each unit is the same. However, looking at the flowline diagram (Fig. 6.2b) actually shows that while unit 2 requires a smaller amount of work, the work is advancing at a slower rate leading to a duration equal to that required by the larger units. On the other hand, The LOB visual is closer to a bar chart visual that clearly shows durations and crew movements. In addition, LOB visuals clearly show the task envelop and thus allows effortless visual ability to check if any task violates the precedence relations by intersecting with its predecessor. In the flowline chart, on the other hand, where the activity envelops is not shown, care has to be taken in this regard. For example, task B in unit 1 (i.e., B_1 in Fig. 6.2b) has to start on or after time (a) which is the finish of its A_1 predecessor of task A in unit 1. Similarly, task C_3 in unit 3 can only start after time (b) which is the finish of its B_3 predecessor.

Because LOB and Flowline representations have their unique advantages, using either one means missing the advantages of the other visual. To combine the benefits of LOB and Flowline visuals, a new visual called Duration-Distance (DD) chart has been proposed, as shown in Fig. 6.3. In this new visual, the distance (on the vertical axis) is shown with the flowlines in the foreground. Background bars are also included, similar to LOB to define task durations and task envelops. As such, the DD chart combines the benefits of LOB and Flowline. It gives a visual envelope for each task to help in checking for interferences in the duration-distance zones of tasks. This chart, as such, suits all types of repetitive projects. Computerizing this chart, it is possible to set the viewing preferences to on/off for the flowlines, the background bars, or both, to suit different users.

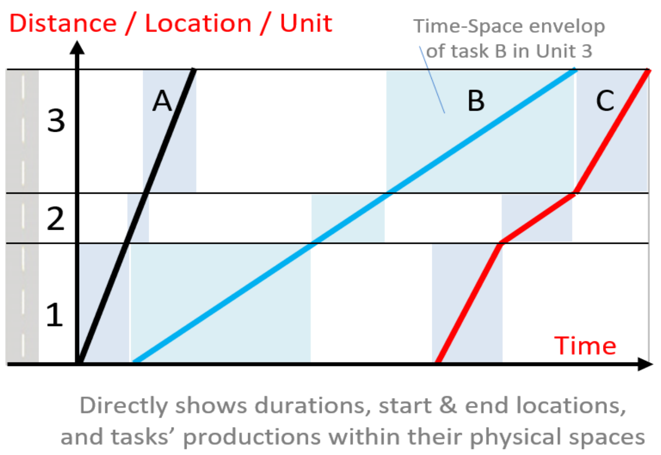


Fig. 6.3: Duration-Distance (DD) chart combines the benefits of LOB and Flowline visuals.

6.3 Novel Computations

This section discusses the novel computations developed to enhance the scheduling of repetitive projects. In this section, two formulae are presented. First, a formula to avoid deadline violations in the case of relaxed deadlines. Second, a formula to calculate designed interruptions.

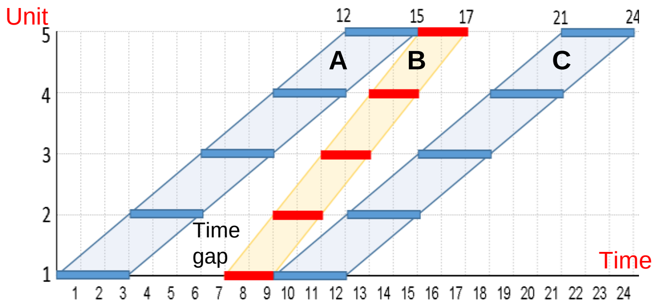
6.3.1 Preventing Schedule Delays in the case of Relaxed Deadlines

As a small example, consider the small 3-activity project shown in Fig. 6.4, with a deadline of 20 days and 5 repetitive units. The standard LOB calculations using equations 2.1-2.3 are shown in the figure. Similar calculations for the parallel crew arrangement using equations 2.4-2.6 are shown in Fig. 6.5.

$$\begin{tikzcd}\boxed{A\ (3)} \arrow[r, dash] & \boxed{B\ (2)} \arrow[r, dash] & \boxed{C\ (3)}\end{tikzcd}$$

Deadline (*DL*) = **20** days, CPM (T_1) = 8 days, Units (N) = 5

	Duration **(D)**	Desired Rate, $\frac{N-1}{(DL - T_1 + TF_i)}$	Necessary Crews, **(C) = Roundup (D_i x R_i)**	Actual Rate **(R) = C/D**	1/R
A	3	4/(20-8) = 0.33	1	0.33	3 days
B	2		1	0.50	2 days
C	3		1	0.33	3 days



Notes:

1. The relaxed deadline led to the use of only one crew in each task.

2. Initially, the CPM duration of first unit = 8 days, however, with one crew, time gap is introduced, making first unit 12 days. .

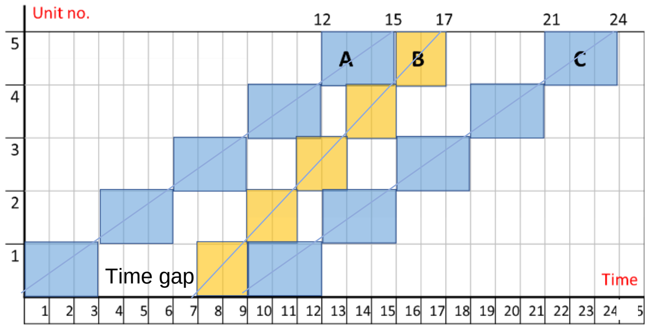
Fig. 6.4: CPM/LOB Schedule for Shifted Crews exceeds the Deadline

$$\begin{tikzcd}\boxed{A\ (3)} \arrow[r, dash] & \boxed{B\ (2)} \arrow[r, dash] & \boxed{C\ (3)}\end{tikzcd}$$

Deadline (T_{DL}) = 20 days

CPM (T_1) = 8 days Units (N) = 5

	Duration **D**	No. of Cycles (S) **(T_{DL}-T_1) / D_i + 1**	Necessary crews **C = Roundup (N/S)**	Actual Cycles **S = Roundup (N/C)**
A	3	(20-8)/3 + 1= 5	Roundup (5/5) = 1	5/1 = 5
B	2	(20-8)/2 + 1= 7	Roundup (5/7) = 1	5/1 = 5
C	3	(20-8)/3 + 1= 5	Roundup (5/5) = 1	5/1 = 5



Notes:

1. The relaxed deadline led to the use of only one crew in each task.

2. Initially, the CPM duration of first unit = 8 days, however, with one crew, time gap is introduced, making first unit 12 days.

Fig. 6.5: CPM/LOB schedule for Parallel Crews exceeds the Deadline

As seen in the example presented in Figs. 6.4, 6.5. Using LOB formulae resulted in a schedule that is 24 days long. This violates the original deadline constraint of 20 days. The first unit alone is completed in 12 days as opposed to the original CPM duration of 8 days. This implies that using a larger CPM duration in the LOB calculation would ultimately create a schedule that satisfies the deadline constraints. However, using an excessively large duration would require the use of more resources (i.e., extra crews) which corresponds to higher costs. Hence, there needs to be a balance between resource usage and achieving early completion.

Computationally, calculating the number of needed crews can be seen as a function of (DL – T_1), which is the difference between the deadline (DL) and the CPM duration (T_1) for one unit. As such, the smaller the (DL – T_1) value, the more crews to use. Therefore, reducing the (DL - T_1) value can be achieved by either reducing the deadline duration or increasing the CPM's T_1 duration to account for the expected schedule gaps. As such, the proposed computation in Fig. 6.6 checks if the project deadline is so relaxed that it may lead to violations. It does so by dividing the gap between the duration of one unit (T_1) and the deadline into five segments of length *G*. Then, if the produced LOB schedule using the original deadline resulted in a deadline violation (i.e., the original deadline is too relaxed), it adds *G* days to T_1 and re-performs the LOB calculations using the new deadline. This loop terminates once a schedule is reached that satisfies the original deadline constraint to avoid overusing resources.

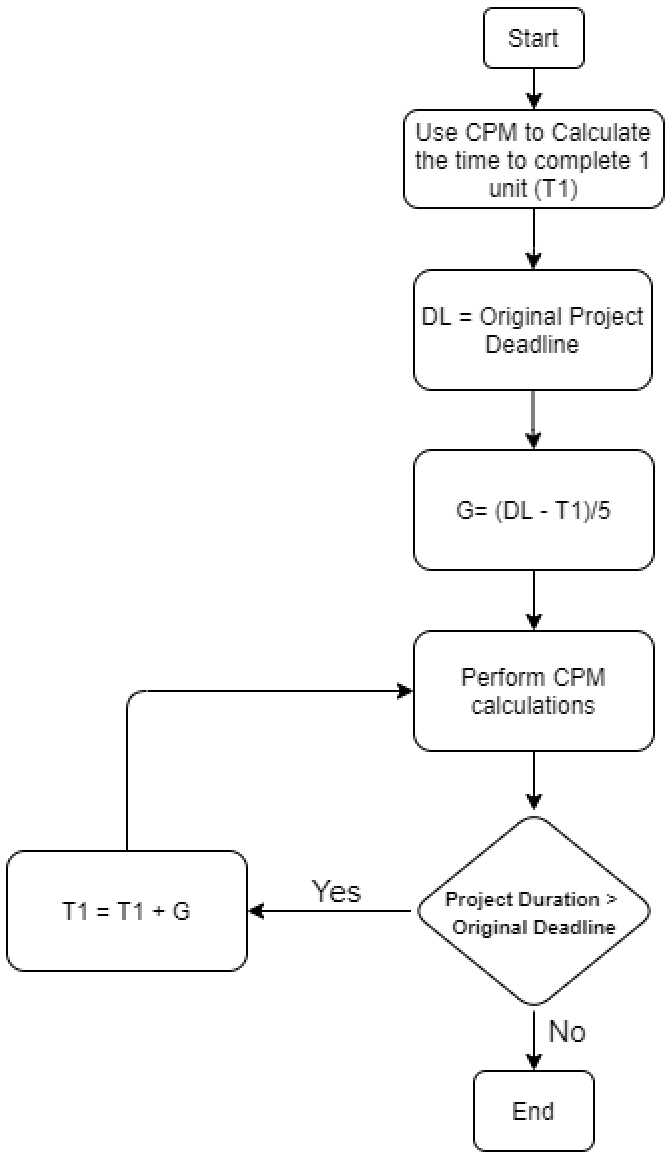
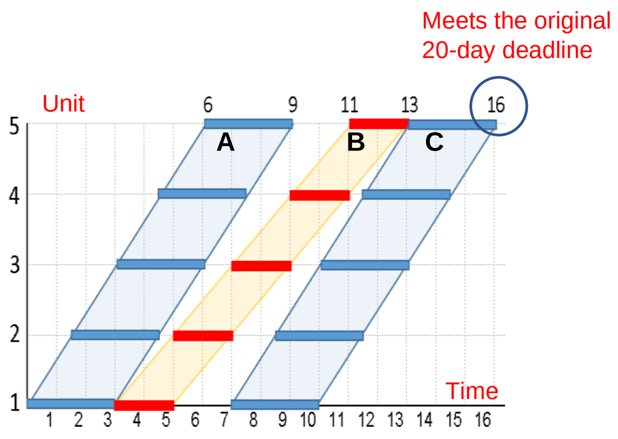
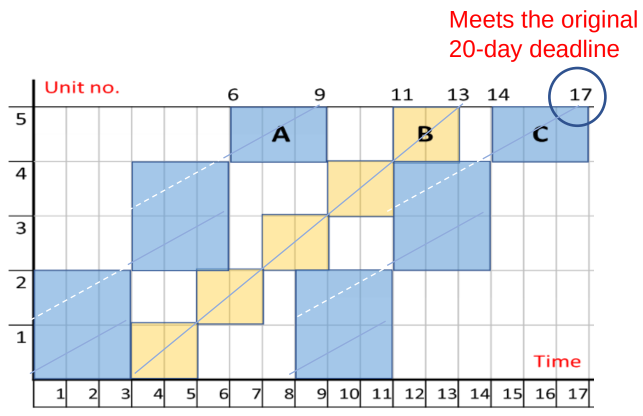

Start
Use CPM to Calculate the time to complete 1 unit (T1)
DL = Original Project Deadline
G= (DL - T1)/5
Perform CPM calculations
Project Duration > Original Deadline
Yes
T1 = T1 + G
No
End

Fig. 6.6: Flowchart of the Improved CPM/LOB Computation

The developed computation in Fig. 6.6 is applied on the schedule originally introduced in Fig. 6.4. Since the original deadline was 20 days and the CPM duration was 8, the difference between the two values will be divided into four segments of equal length (G = (20 – 8) /5 = 2.4 days). As shown before in Fig. 6.4, when the deadline was set to 20 days, the resulting CPM/LOB schedule ended up as 24 days, which is unacceptably beyond the deadline long. Hence, the introduced computation loop introduced in its first cycle a longer T_1 of 10.4 days was used (T_1 + G = 8 + 2.4). The calculations and the resulting schedules using this deadline can be seen in Fig. 6.6, for shifted and parallel crews, respectively. Since the durations of the improved schedules are less than the project's original deadline of 20 days, the process terminates after one cycle of the loop. If that was not the case, a new T_1 of 12.8 days (10.4 + 2.4) would have been proposed and a new schedule would have been developed using the revised deadline. The parallel-crews schedule of Fig. 6.7b maintains crew continuity while putting all crews with integer start times.



(a) Improved schedule of shifted crews

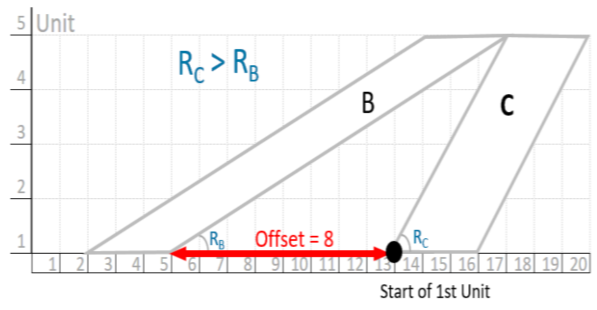


(b) Improved schedule of parallel crews

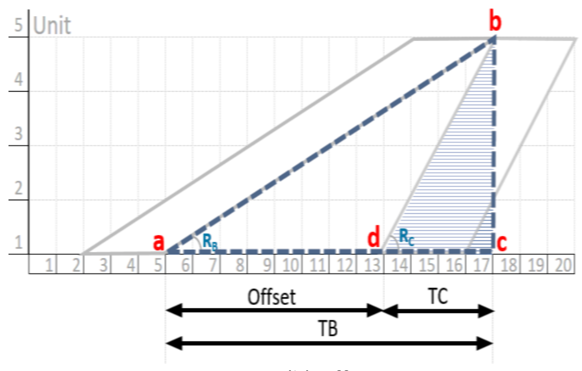
Fig. 6.7: Schedules developed using the Proposed Computation meet the Original Deadline

6.3.2 Calculating Designed Interruptions

To present a formulation for the start-time offset, let's consider activities B and C in Fig. 6.8. As schematically shown in Fig. 6.8a, the start-time offset for activity C is the amount of delay time from the end of the predecessor's unit 1 to the start of activity C in unit 1, which depends on the difference in rates of progress between the two activities. To facilitate the calculation, Fig. 6.8b highlights two right-angle triangles: a-b-c (related to activity B) with a base of TB and height of N-1; and d-b-c (related to activity C) with a base of TC and height of N-1. From these two triangles, the Start-Time Offset (***STO_C***) for activity C can be formulated as follows:



(a) Start-Time Offset for activity C.



1. (b) Offset = TB -

Fig. 6.8: Formulating the Start-Time Offset

[6.1] Start-Time Offset from task B to task C = TB − TC

$$= \frac{N-1}{R_B} - \frac{N-1}{R_C} = (N-1)\left(\frac{1}{R_B} - \frac{1}{R_C}\right)$$

Applying Equation 6.1 to the case in Fig. 6.8, ***STO***$\boldsymbol{STO_C}$ = 4 (3 − 1) = 8 days, as shown in the figure. The equation applies to the situation when task (C) has a higher rate than the predecessor (B), otherwise, it produces a negative value in the opposite case. To generalize this equation for all cases (including cases of parallel rates or when the successor is slower than the predecessor), it is possible to re-write equation 6.1, avoiding negative values, as follows:

[6.2] Start-Time Offset of any task *i* ($\boldsymbol{STO_i}$) = $\text{Max}\left[0\ ,\ (N-1)\left(\frac{1}{R_{predecessor}} - \frac{1}{R_i}\right)\right]$

Equation 6.2, as such, does not take any negative value, rather, only positive or zero. The equation, therefore, is generic and can apply to all tasks during the forward-pass scheduling process. As such, using Equation 6.2 during the forward-pass of repetitive scheduling, all tasks are scheduled bottom-up, thus having a more systematic and easy-to-follow process.

For the purpose of plotting the LOB chart, only the lower-left corner of the activity diagram which represents the start date of that activity at unit 1 is required. This is calculated in Equation 6.3, while the start of any unit *j* of activity *i* is calculated in Equation 6.4:

[6.3] Start of activity *i* at unit 1 = Finish Time of unit 1 of Predecessor + STO_i

[6.4] Start of activity *i* at unit j = Start of activity *i* at unit 1 (Equation 6) + $\frac{j-1}{R_i}$

It is noted that in the case activity *i* has multiple predecessors, Equations 6.2-6.4 are calculated for each predecessor and the highest value is used.

To present a formulation for achieving better synchronization among the activities' delivery rates through designed interruption, the case in Fig. 6.9 is used, where a faster activity (B) follows a slow one (A). As schematically shown in Fig. 6.9a, the start-time offset for activity B is shown, and can be formulated using Equation 6.3. Assuming one interruption at N/2 is sufficient, the interruption time that allows the bottom half of activity B to start earlier is shown in the figure. As shown, the Offset time of B = Y + Duration of B + Interruption time; where Y can be calculated using Equation 6.3, taking into account the only first half of activity B units, therefore:

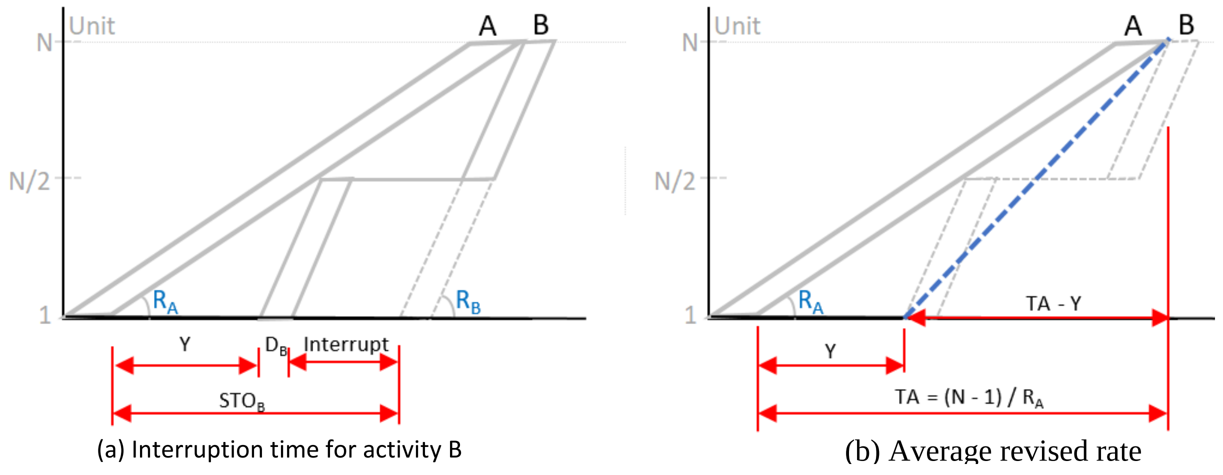


Fig. 6.9: Formulating Task Interruption Time

[6.5] Interruption Time = STO of B − Y (STO of B for N/2 units) − Duration of B =

$$= (N-1)\left(\frac{1}{R_A} - \frac{1}{R_B}\right) - ((N-1)/2)\left(\frac{1}{R_A} - \frac{1}{R_B}\right) - D_B$$

Thus, applying the interruption, the bottom part activity B (with N/2 units) starts with an offset of Y:

[6.6] Start-Time Offset of B with interruption = Y = $((N-1)/2)\left(\frac{1}{R_A} - \frac{1}{R_B}\right)$

Using this formulation, it can be seen that the lower part of activity B started early, thus, changing the average delivery rate of B (dashed line in Fig. 6.9b) to a lower rate than the original rate of B, and becoming more closer in its rate to its predecessor (A). This average modified rate of B can also be formulated, based on the presentation in Fig. 6.8b, as follows:

[6.7] Average rate of B with interruption = $\left(\frac{N-1}{TA-Y}\right) = \left(\frac{N-1}{\frac{N-1}{R_A}-Y}\right)$

As a demonstration of the calculations for interruption time, the previous example of activities B and C in Fig. 6.8 is continued. Interruption time is calculated using Equation 6.5 as 1 day, as shown in Fig. 6.10. Also, the revised rate of activity C after interruption is calculated using Equation 6.7 as 0.5 units per day.

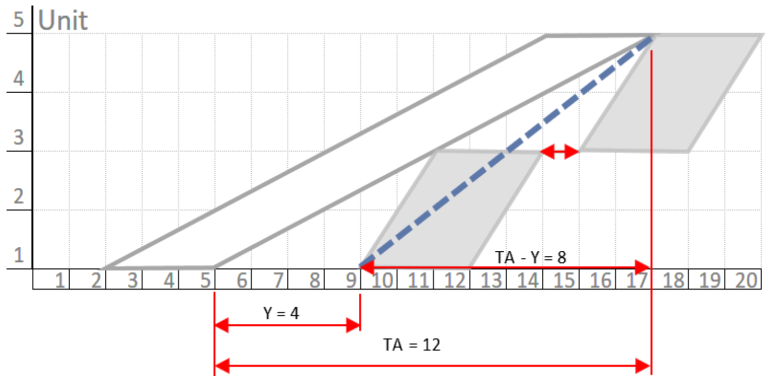


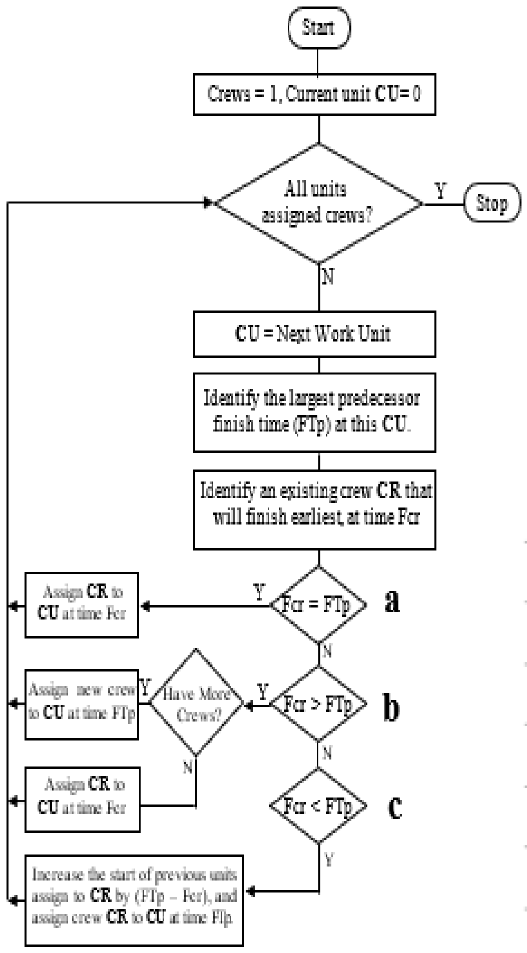
Fig. 6.10: Interruption Time for Example Activities

6.4 Novel Algorithm (First-Come-First-Serve)

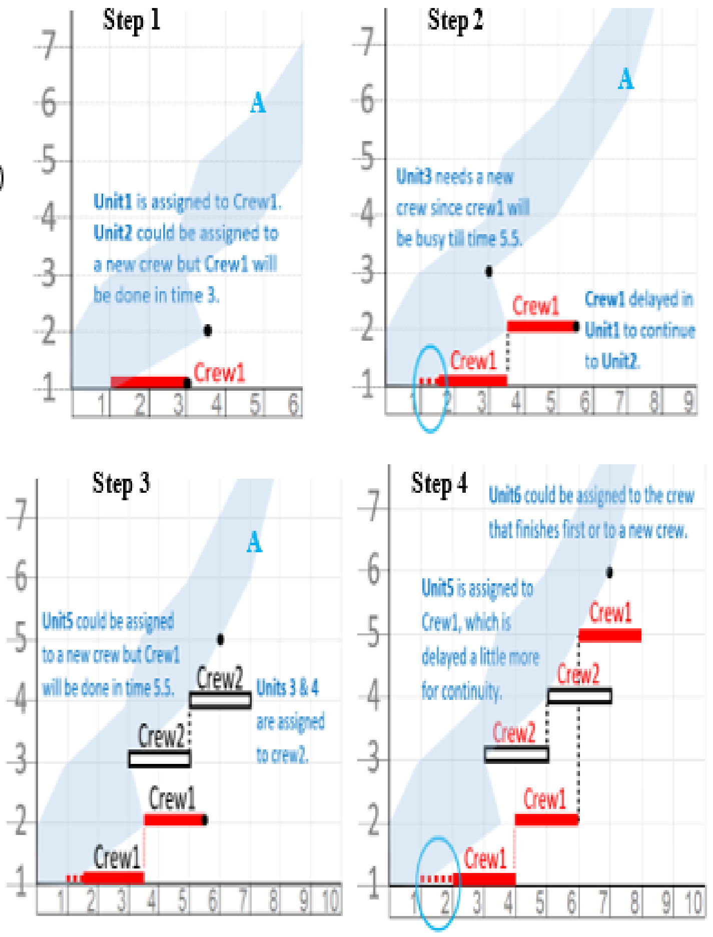
Typically, crews are assigned to units in sequential order. For example, if an activity has 3 crews, they are assigned to units 1, 2, and 3. These crews will then move to units 4, 5, and 6 in the same order, etc. While this sets which crew will work in which unit, this often creates unnecessary time gaps and possible project delays, especially in the case of non-identical units.

Knowing that the rehabilitation work required is different from one unit to another, a generalized crew assignment framework was developed that facilitates task synchronization and maintains crew continuities while not being tied to a rigid crew assignment strategy or needs to calculate interruption times beforehand. Fig. 6.11 illustrates our newly proposed, First-Come-First-Serve (FCFS), method. This method deals with each crew at a time following the flowchart in Fig. 6.11(a) and can be seen in action in Fig. 6.11(b). A detailed explanation is as follows:

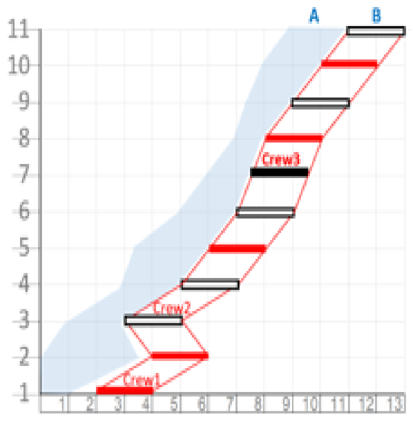
1. In the first step, all crews are available (Fcr=0). Hence, crew 1 is assigned to the first unit and its finish time is updated to reflect the finish time of activity B in the first unit.
2. In step 2, the finish time of the predecessor to activity B in the second unit (FTp) is greater than the updated finish time of crew 1. Hence, crew 1 is assigned to the second unit. To maintain work continuity, the start of crew 1 at the first unit is delayed by the difference between FTp of the second unit and the scheduled finish of the first unit. Finally, the scheduled finish of crew 1 (Fcr) is updated to reflect the finish of activity B in the second unit
3. In step 3, FTp of the third unit is smaller than the scheduled finish of crew 1. Crew 2 is assigned to that unit and its Fcr is updated accordingly. FTp of the fourth unit was less than Fcr of crew 1 but equals to that of crew 2. Therefore, crew 2 was assigned to the fourth unit.
4. Similarly, in step 4, crew 1 is assigned to unit 5, the start dates of its previous units (units 1 and 2) are delayed to remove work interruption, and its Fcr is updated to reflect the finish of the activity in unit 5.
5. The same procedure explained in the previous step is followed for the rest of the units, and the final schedule is featured in Fig. 6.11(c).


Start
Crews = 1, Current unit CU= 0
All units assigned crews?
Y
Stop
N
CU = Next Work Unit
Identify the largest predecessor finish time (FTp) at this CU.
Identify an existing crew CR that will finish earliest, at time Fcr
Y
Fcr = FTp
a
Assign CR to CU at time Fcr
N
Y
Fcr > FTp
b
Have More Crews?
Y
Assign new crew to CU at time FTp
N
Assign CR to CU at time Fcr
N
Fcr < FTp
c
Y
Increase the start of previous units assign to CR by (FTp – Fcr), and assign crew CR to CU at time FTp

(a) Flowchart of the FCFS crew assignment



(b) Steps in the FCFS crew assignment applied to task B



(c) Completed Schedule of activity B using FCFS assignment

Fig. 6.11: Flowchart and Steps of the Proposed FCFS Crew Assignment Process

6.5 Validation Example

To demonstrate the effectiveness of the proposed enhancement to repetitive scheduling visualizations and computations, the example of Dolabi et al. (2014) was solved using the proposed method and its results were compared to Dolabi et al.'s solutions. The example is a 10-unit highway project where each unit contains 24 sequential activities, with a project deadline of 240 days. Table 6.1 shows the data for the validation example (e.g., task durations) as well as the results obtained by Dolabi et al. (2014) and the proposed model.

Table 6.1: Validation Example Data and Comparison of Results

Task ID	Task Duration	Crews of Dolabi et al.'s Solutions			Crews of Proposed Solutions
		CPM/ LOB	Heuristic (HLOB)	Heuristic (SHLOB)	
1	6	1	3	3	3
2	4	1	2	2	2
3	12	2	4	4	5
4	9	2	3	3	4
5	13	2	4	4	5
6	10	2	3	3	4
7	2	1	1	1	1
8	4	1	1	2	2
9	9	2	2	4	4
10	5	1	1	2	2
11	8	2	2	3	3
12	7	1	2	2	3
13	9	2	3	2	4
14	8	2	3	2	3
15	20	3	8	5	8
16	7	1	3	2	3
17	9	2	4	3	4
18	7	1	4	2	3
19	6	1	4	2	3
20	8	2	6	3	3
21	1	1	1	1	1
22	3	1	3	2	2
23	5	1	5	3	2
24	4	1	4	3	2
	Total crews	36	76	63	76
	Project duration	405.5	233	239	226

In table 6.1, it can be seen that the conventional CPM/LOB calculations without the use of the deadline validation loop produces a schedule of 405.5 days (column 3 in Table 6.1). This is unacceptable as it violates the project's 240-day deadline. To resolve this issue, Dolabi et al. (2014) used complex heuristics to reach a project duration of 233 and 239 days. In this model, however, the deadline validation loop (subsection 6.3.1) was used to change the value of T_1 till a satisfactory project duration was met. Furthermore, using the FCFS crew assignment algorithm produced a schedule with smaller gaps which translated to a smaller duration even through the same number of crews, 76, was used (column 6 in Table 6.1). Fig. 6.12 shows the schedule developed by the proposed algorithms.

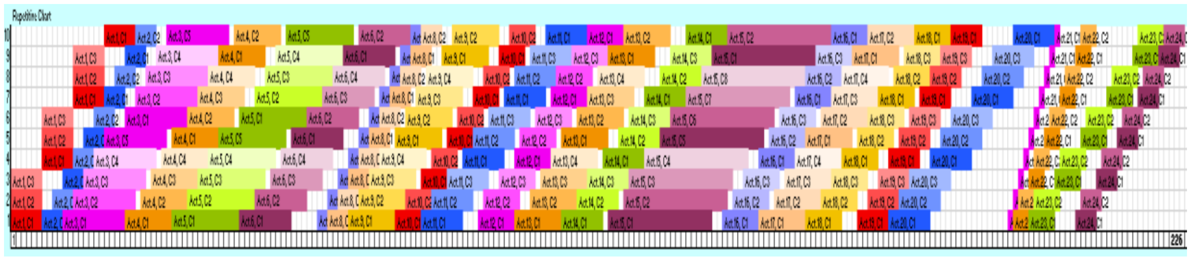


Fig. 6.12: Developed Schedule for the Validation Example Using the Proposed Model

It can be noticed in Fig. 6.12 that there is a relatively huge schedule gap between activities 20 and 21 because of the relatively fast delivery rate of activity 21. Hence, introducing a designed interruption to activity 21 could produce a schedule with an even shorter duration. To test this hypothesis, a four-day interruption was introduced after the fourth unit. The new schedule can be seen in Fig. 6.13. As expected, the introduced interruption reduced the schedule gap which in turn reduced the overall project duration, bringing it down to 224 days. In future developments, the selection of the interruption duration and location (i.e., after which unit) will be automated and be part of the scheduling automation procedure.

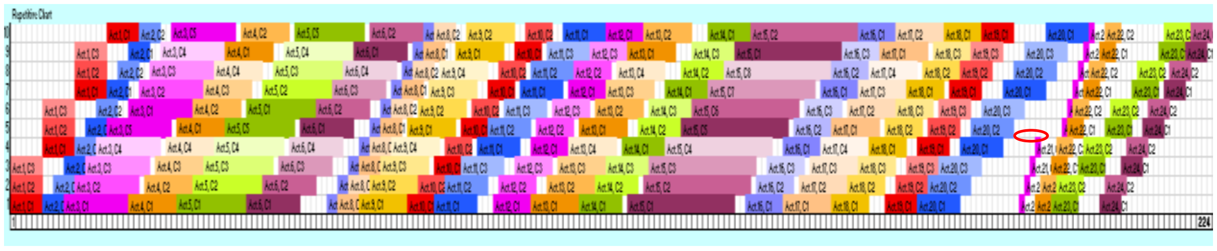


Fig. 6.13: Developed Schedule for the Validation Example After Introducing Interruption

6.6 Application Case Study

Based on the analysis conducted in Chapter 5, 20 units were deemed most worthy of immediate repairs given the budgetary limitations. As such, the novel formulations and visualizations presented earlier in this chapter will be used here to develop an optimized schedule for the delivery of the necessary rehabilitation activities.

6.6.1 Activity Parameters and Order of Execution

Table 6.2 shows the data concerning the units to be repaired. For each school (1-20), the total cost is presented in the second column. The area of the roof to be repaired, presented in the third column, is estimated by dividing the total cost of repair by $300. This value for repairs was obtained by consulting with members of the University of Waterloo Plant Operations. Based on the estimated area, the estimated duration based on R.S. Means data for each activity is presented in columns 4-7. Transportation duration to move from one unit to the other is ignored, and it is assumed that these activities will follow a finish-to-start relationship.

As seen in Table 6.2, the schools are arranged in descending order based on their size. This is because executing the works in that order is more efficient than going from the smallest school to the largest. A proof using a simple example (two activities conducted over three units using two crews) can be seen in Fig. 6.14.

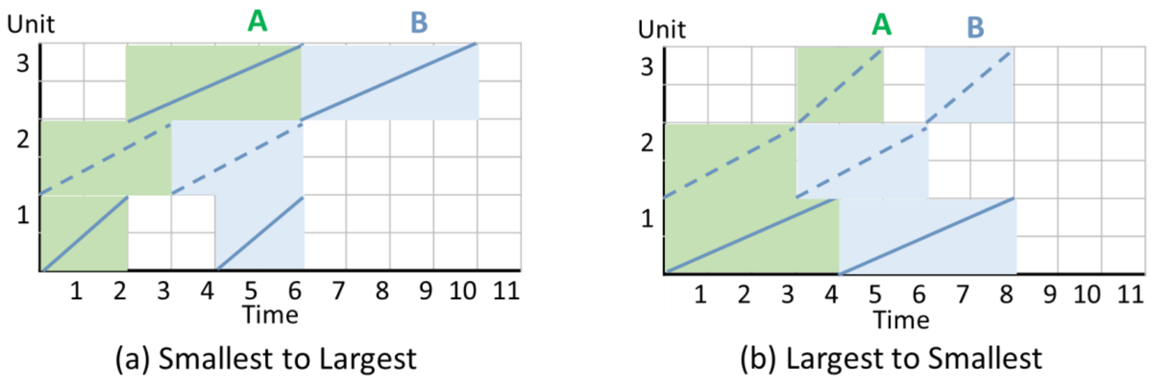


Fig. 6.14: Choosing the most Efficient Order of Execution

Table 6.2: Activity Information for Schools Undergoing Roof Rehabilitation

			Activity Duration (Days)			
School ID	**Cost**	**Size (m2)**	**removal & cleanup**	**asphalt base sheet**	**fiber felt**	**flood coat with gravel surfacing**
1	242000	807	3	4	5	4
2	175450	585	2	3	4	3
3	159115	530	2	3	3	3
4	151250	504	2	3	3	3
5	127050	424	2	2	3	2
6	121000	403	2	2	3	2
7	121000	403	2	2	3	2
8	119790	399	1	2	3	2
9	118989	397	1	2	3	2
10	114950	383	1	2	3	2
11	96800	323	1	2	2	2
12	96800	323	1	2	2	2
13	96800	323	1	2	2	2
14	93170	311	1	2	2	2
15	93170	311	1	2	2	2
16	90750	303	1	2	2	2
17	60500	202	1	1	2	1
18	60500	202	1	1	2	1
19	48400	161	1	1	1	1
20	24200	81	1	1	1	1

6.6.2 Rehabilitation Schedule using FCFS

45 days were selected as the deadline to finish all the required rehabilitation works. While typically repairs take place within the two-month summer vacation period (July and August), the 45-day deadline was selected to allow for contingencies. Based on the duration information presented in Table 6.2, it was assumed that the typical unit will have a duration of two days per activity. As such, according to the basic CPM/LOB calculations, the duration of one unit (T_1) is eight days and the rate of delivery is therefore 19/(45-8) = 0.51. This means that each activity will require 2 crews to deliver the project within the allotted duration. The developed schedule is shown in Fig. 6.15. The developed schedule is 33 days long which satisfies the deadline constraint. It can be seen that the developed schedule has minimal gaps while maintaining work continuity, thus validating the efficiency of the proposed FCFS approach.

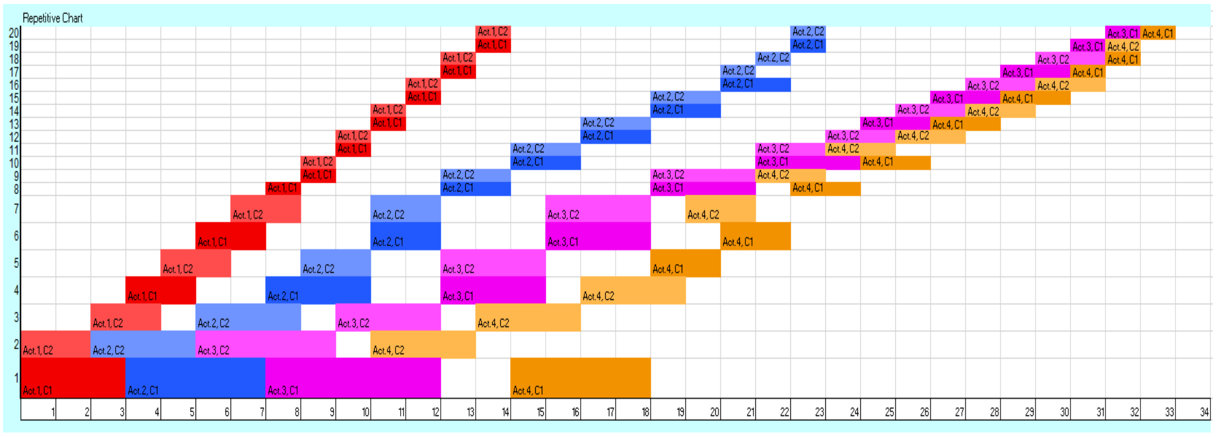


Fig. 6.15: FCFS Schedule to meet a 45-day Deadline

A different scenario was examined where the rehabilitation works for all units need to be performed within 30 days only. In that case, the rate of delivery becomes 19/(30-8) = 0.86 and the required number of crews remains two. However, using only two crews per activity produces the same schedule in Fig. 6.15 which has a duration of 33 days. This is unacceptable as it violates the now stricter 30-day deadline. Therefore, the deadline checking loop presented in Fig. 6.6 comes into effect. First, G is calculated to be (30-8)/5 = 4.4 days. Hence, the CPM calculations are redone for the first loop with a new value for T_1 which equals 8+4.4 = 12.4 days. Accordingly, a new rate of delivery is calculated to be 1.08 which means that 3 crews will be required for each activity (roundup(3*1.08)). The developed schedule using the new crew information is in Fig. 6.16. It can be seen that the new schedule has a duration of 26 days which complies with the 30-day deadline. This proves the versatility and the effectiveness of the proposed approach.

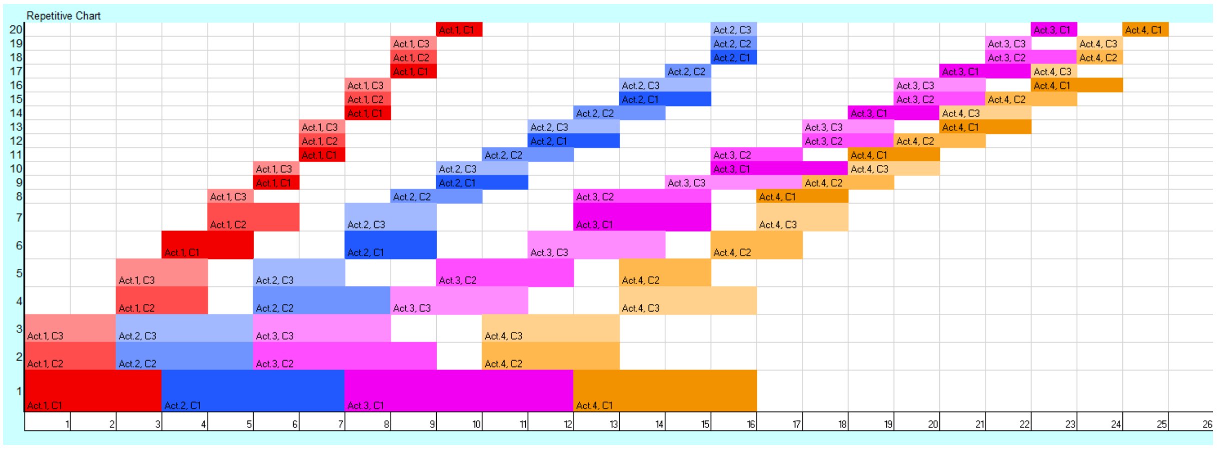


Fig. 6.16: FCFS Schedule to meet a 30-day Deadline

6.7 Conclusion

This chapter has presented the third and final module of the proposed roofing rehabilitation framework; the use of novel repetitive scheduling techniques to facilitate the delivery of the required rehabilitation work. Novel computations, visualizations, and scheduling algorithms were proposed and applied to a case study based on the data analyzed by the previous modules (Chapters 4 and 5). Computationally, schedules developed using the FCFS algorithm exhibit minimum gaps while maintaining work continuity and abiding by the deadline constraints. The deadline checking loop allows for adjusting the project schedule by manipulating one parameter only (T_1) as opposed to tweaking every activity manually. Visually, the novel duration-distance charts now have extra information regarding the size of each unit, represented by the height of each activity block. The presented case study demonstrates the proposed scheduling developments and proves their effectiveness compared to traditional methods as well as the generic applicability to all types of repetitive projects.

Chapter 7: Conclusion and Future Research

7.1 Introduction

The typical vision of a smart city focuses on building new smart assets. This vision, however, overlooks the need for a “smart rehabilitation” framework that addresses the conditions of the existing infrastructure assets and preserves their condition and acceptable level of service. Currently, public organizations and managers of large asset portfolios have significant challenges keeping up with the multibillion-dollar maintenance backlogs of their assets, especially when many assets are old and funding is inadequate. As such, improving the existing asset management frameworks and streamlining its processes is a must.

The current asset management practices regarding asset inspections, prioritization and fund allocation, and rehabilitation delivery have been investigated. These processes are done manually and often consider each asset separately. This deprives these organizations of the benefits of applying data-driven inspection and decision support systems. Such benefits include faster and more objective decisions, as well as the ability to analyze all assets in unison which helps enhance the overall service level of the entire portfolio. Creating a rehabilitation delivery schedule that considers the repetitiveness of the tasks across multiple units allows for cost and time savings by reaping the benefits of repetitive schedules such as momentum, learning curve, and economy of scale. Having an effective data-driven asset management framework would optimize the use of the limited rehabilitation funds to improve the conditions of the different assets and reduce the repair backlog.

7.2 Research Summary

The primary goal of this research is to establish “smart rehabilitation” as a major component of the smart asset management layer of smart cities. Specifically, this research utilizes machine learning tools, such as computer vision and data mining, and repetitive scheduling techniques to develop an automated framework for smart city rehabilitation. The framework includes different functions that perform efficient condition assessment, prioritization and fund allocation, and delivery planning of time-critical and cost-critical rehabilitation works.

Regarding the condition assessment phase, the manual nature of conducting inspections was seen as the main issue to address. To that end, computer vision was used to develop an automated system to inspect roofing elements. The system would detect and classify defects according to their type directly from collected images. The proposed system is composed of two CNN models; one for detection, and the other for classification. Each individual CNN is composed of five convolutional and pooling layers (3+2), an activation layer, a dropout layer, a fully connected layer, and a final output layer. The model performance speed is, on average, 0.08 sec/image for both phases. This means that the model is faster than manual inspections. Faster inspections mean that they can be performed more frequently which makes the inspection data more updated and relevant to the decision-making process.

Regarding the prioritization and fund allocation processes, a comprehensive study was conducted on the inspection reports produced by Toronto District School Board (TDSB) inspectors. It was found that the reports do not offer the level of detail necessary to perform such prioritization activities, which requires the asset manager to manually discern the contents of the reports and rely on their own personal experience and biases to reach a decision. To address this issue, data mining and unsupervised clustering were used to create a model capable of analyzing the textual information available in different reports and categorizing the schools into one of four categories (1-4, 4 being the neediest for repair). The model incorporates multiple parameters such as the building age and description, as well as the damage description provided by the inspectors. A second version of the clustering model was then developed which includes the data collected from the automated image-based inspection module, represented as the percentage of the roof being damaged.

The delivery phase was tackled with the aim of incorporating repetitive scheduling techniques into asset rehabilitation delivery projects as opposed to treating each asset separately. To that end, current drawbacks within the existing repetitive scheduling computations were highlighted and remedies were introduced. This study introduced novel visualizations (duration-distance chart), computations (scheduled interruptions, and preventing deadline violations), and algorithms (first-come-first-serve) to develop repetitive schedules with minimum duration and maximum continuity.

7.3 Research Contributions

Based on the above summary, the research contributions of this research can be encapsulated in the following points:

- Better understanding of challenges in managing large asset portfolios with huge backlogs and limited budgets;
- Development of an automated image-based system capable of detecting, classifying, and quantifying defects directly from collected images;

- Development of a data-mining-based system for prioritization and fund allocation maximizing the gained benefits from performed rehabilitation works while meeting budgetary constraints;
- Development of new representation of repetitive schedules that incorporates more information related to the size of the work in each unit;
- Development of new repetitive scheduling computations to avoid deadline violations and reduce schedule gaps; and
- Development of a novel crew assignment algorithm for repetitive schedules to reduce the project duration by developing more cost-effective and compact schedules.

7.4 Future Research

7.4.1 Potential Research Related to Inspection and Condition Assessment:

- The performance speed of the image-based inspection model has shown its potential for performing real-time analysis. This means the inspection process can be automated using drones where, in addition to identifying the damage type, the damage location can be pinpointed using image registration or GIS techniques. This would help better identify the repair strategy (e.g., an overall replacement vs. a localized repair).
- Faster inspections allow for more frequent inspections, this would help develop models that better track and forecast the damage progression, improving the effectiveness of future preventative maintenance frameworks. Incorporation with BIM can enable the development of “as-damaged” models where different what-if scenarios for building rehabilitation strategies can effectively tested and analyzed
- Some roofing defect types (e.g., cracks, blisters, flashing) almost did not appear in the collected images, and therefore were not extensively experienced by the model, this should be investigated and be part of future model improvements.
- Image analysis is incapable of analyzing internal roofing defects (e.g., internal cracks, saturation of the roof insulation) and can be obscured by the roof coverings and other appliances. The use of infrared imagery should be investigated to overcome these challenges.
- The images were collected in a specific way such that they represent constant real-life dimensions (1.5x1.12 m) which allows to automatically quantify the damage size. As such, full

automation of defect sizing and quantification to reduce the data collection burden by collecting images that represent larger roof areas as well as reduce the constraints on the data collection methods.

- Collecting images that represent larger roof areas require the use of homography to identify the size of the damage within the image, as opposed to estimating the size of the damage to be equivalent to the total area covered by the image.
- While the developed model is only applicable to roofing, the same underlying technology (i.e., CNN) can be utilized to develop similar models that tackle different building assets such as structural elements, doors, and windows to name a few. These models could be then aggregated to develop a comprehensive building assessment model that considers all building elements.
- Applying feature extraction techniques to resolve issues such as shadows and markings, especially when the model is applied to building elements where this is a prevalent issue (e.g., parking lots).
- Using feedback loops as part of the model architecture to enhance its accuracy by retraining the model against datapoints that were misclassified to examine the source of misclassifications and avoid them in future iterations.

7.4.2 Future Research Related to Prioritization and Fund Allocation:

- Development of a more comprehensive optimization framework that accounts for multi-year investments and different delivery methods and/or rehabilitation strategies (e.g., different repair methods).
- Including probabilistic analyses that study the reliability of the funding decisions against changes to the rehabilitation costs and/or budgets.
- Collaborating with the school of social work and/or school of public health, incorporating social parameters into the optimization framework (e.g., the demographics of the community served by the asset)
- Investigating how the different assets interact with one another. For example, an inoperable school means that another school will be overloaded because of the increased size of the student population, which would affect its structural integrity.

7.4.3 Future Research Related to Scheduling and Delivery

- Although the simple scheduling example provided in Chapter 6 showed that it was faster to execute the units in descending order according to their size, there is a possibility to have different arrangements that provide a shorter project duration. This is better examined by treating repetitive scheduling as a traveling salesman/vehicle routing problem.
- Including interruption calculations as well as duration and location (i.e., after which unit) selection into the automated optimization procedure
- Consideration of interruption-related impacts in terms of demobilization and re-mobilization, and how that affects the momentum and productivity of the crews.
- Study the effect of changing the activity delivery rate midway through the project and its effects on the project duration and resources.
- Development of repetitive scheduling visuals and computations that consider project progress.
- Examining the applicability of the proposed repetitive scheduling algorithm on real-world megaprojects both in terms of practicality and abiding by the project's requirements and complex constraints, as well as the algorithm's computational requirements.

7.4.4 Future Integration with Smart City Initiatives

- Reach out to different cities to develop long-term strategic simulations using tools such as system dynamics to investigate policy issues related to the budgeting of new versus rehabilitation projects and the impact on backlog.
- Extend the smart rehabilitation work to other infrastructure domains such as the transportation network (roads and bridges), water/sewer, etc.

References

Abdel-Monem, M. S., and Ali, A. I. (2010). "Spreadsheet-based system for sustainable asset management." CSCE 2010 General Conference, Winnipeg, MB.

Abdel-Qader, I., Abudayyeh, O., and Kelly, M. E. (2003). "Analysis of Edge-Detection Techniques for Crack Identification in Bridges." Journal of Computing in Civil Engineering, 17(4), 255–263.

Abou Shaar, B. (2012). Adaptable Three-Dimensional System for Building Inspection Management. Dissertation. Department of Civil and Environmental Engineering, University of Waterloo

Abu Abbas, O. M. (2008). "Comparisons Between Data Clustering Algorithms." International Arab Journal of Information Technology, 5(3), 320-325.

Advancer Global (n.d.). "Smart FM." Advancer Global.

Aggarwal, C. C., and Zhai, C. (2012). "A survey of text classification algorithms." Mining Text Data, Springer, US, 163-222.

Ahluwalia, S. S., and Hegazy, T. (2010). "Roof deterioration and impact: A questionnaire survey." Journal of Retail & Leisure Property, 9(4), 337–348.

Al Hattab, M. (2021). "The dynamic evolution of synergies between BIM and sustainability: A text mining and network theory approach," Journal of Building Engineering, 37.

Alawadhi, S., Aldama-Nalda, A., Chourabi, H., Gil-Garcia, J. R., Leung, S., Mellouli, S., Nam, T., Pardo, T. A., Scholl, H. J., and Walker, S. (2012). "Building understanding of smart city initiatives." Lecture Notes in Computer Science, 40–53.

Albino, V., Berardi, U., and Dangelico, R. M. (2015). "Smart cities: Definitions, dimensions, performance, and initiatives." Journal of Urban Technology, 22(1), 3–21.

Allam, Z., and Dhunny, Z. A. (2019). "On big data, artificial intelligence and smart cities" Cities, 89, 80-91.

American Society for Civil Engineers (2021), *2021 Report Card for America's Infrastructure*

Anand, P. b, and Navío-Marco, J. (2018). "Governance and economics of smart cities: opportunities and challenges." Telecommunications Policy, 42(10), 795–799.

ANSI (2021). "ASTM2018-15 Standard Guide For Property Condition Assessments: Baseline Property Condition Assessment Process," https://webstore.ansi.org/Standards/ASTM/ASTME201815?gclid=%20EAIaIQobChMIkZDq-pP67gIVDUeRBR15_QlmEAAYASAAEgLkzvD_BwE , Accessed Feb. 19, 2021

Archibus Integrated Workplace Management System, (2021) https://archibus.com/

Azhar, K., Murtaza, F., Yousaf, M. H., and Habib, H. A. (2016). "Computer vision based detection and localization of potholes in asphalt pavement images." 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE).

Bawany, N., and Shamsi, J. (2015). "Smart City Architecture: Vision and Challenges." International Journal of Advanced Computer Science and Applications, 6(11).

Bovik, A. C. (2005). Handbook of image and video processing. Academic Press, Boston, MA.

Ben Letaifa, S. (2015). "How to strategize smart cities: Revealing the smart model." Journal of Business Research, 68(7), 1414–1419.

Cabo, C., Ordóñez, C., Muñiz-Calvente, M., Lozano, M., and Ismael, G. (2019). "A hybrid SURF-DIC algorithm to estimate local displacements in structures using low-cost conventional cameras." Engineering Failure Analysis, 104, 807–815.

Camboim, G. F., Zawislak, P. A., and Pufal, N. A. (2019). "Driving elements to make cities smarter: Evidences from European projects." Technological Forecasting and Social Change, 142, 154–167.

Cha, Y.-J., Choi, W., and Büyüköztürk, O. (2017a). "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks." Computer-Aided Civil and Infrastructure Engineering, 32(5), 361–378.

Cha, Y.-J., Choi, W., Suh, G., Mahmoudkhani, S., and Büyüköztürk, O. (2017b). "Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types." Computer-Aided Civil and Infrastructure Engineering, 33(9), 731–747.

Challawala, A., Ogundiya, K., and Patel, H.(2020). "The Future of Smart Cities." Barclays.

Chapps: Mobile Property Inspection Apps, (2021) https://www.chapps.com/us/

Cheng, J. C., and Wang, M. (2018). "Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques." Automation in Construction, 95, 155–171.

Choi, J., Yeum, C., M., Dyke, S., J., and Jahanshani, M. R. (2018). "Computer-Aided Approach for Rapid Post-Event Visual Evaluation of a Building Façade." *Sensors*, 18(9), 3017.

Choras, R. S. (2007). "Image Feature Extraction Techniques and Their Applications for CBIR and Biometrics Systems." *International Journal of Biology and Biomedical Engineering*, 1(1), 6-16.

City of Ottawa. (2017). Smart City 2.0.

Colldahl, C., Frey, S., & Kelemen, J. (2013). “Smart cities: Strategic sustainable development for an urban world.” Karlskrona, Sweden: Blekinge Institute of Technology.

Collins, A., Lonard, A., Cox, A., Greco, S., and Gianpiero, T. (2017). “Report on urban policies for building smart cities” PERCEVIE: Perception and Evaluation of Regional and Cohesion Policies by Europeans and Identification with the Values of Europe

Dalal, N., and Triggs, B. (2005). “Histograms of Oriented Gradients for Human Detection.” IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05).

Daniels, J. (2017). “California’s troubled Oroville Dam wasn’t on Gov. Brown’s infrastructure ‘wish list’.” CNBC.

Deng, H., Hong, H., Luo, D., Deng, Y., and Su, C. (2020). “Automatic Indoor Construction Process Monitoring for Tiles Based on BIM and Computer Vision.” Journal of Construction Engineering and Management, 146(1), 04019095.

Deng, J., Lu, Y., and Lee, V. C. S. (2019). “Concrete crack detection with handwriting script interferences using faster region-based convolutional neural network.” Computer-Aided Civil and Infrastructure Engineering, 35(4), 373–388.

Devi, R. D. H., Bai, A., and Nagarajan, N. (2020). “A novel hybrid approach for diagnosing diabetes mellitus using farthest first and support vector machine algorithms.” Obesity Medicine, 17, 100152.

Dirks, S., and Keeling, M., (2009). “A Vision of Smarter Cities: How Cities Can Lead the Way into a Prosperous and Sustainable Future”. Retrieved from: https://www.ibm.com/downloads/cas/2JYLM4ZA

Dorafshan, S., Thomas, R. J., and Maguire, M. (2018). “Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete.” Construction and Building Materials, 186, 1031–1045.

Dung, C. V., and Anh, L. D. (2019). “Autonomous concrete crack detection using deep fully convolutional neural network.” Automation in Construction, 99, 52–58.

EZMaxMobile: Work Synchronization App, (2021) https://interprosoft.com/products-services/ezmaxmobile/

Fang, W., Ding, L., Luo, H., and Love, P. E. (2018). “Falls from heights: A computer vision-based approach for safety harness detection.” Automation in Construction, 91, 53–61.

Giffinger, R., and Gudrun, H. (2010). “Smart Cities Ranking: An Effective Instrument for the Positioning of Cities?” ACE Architecture, City and Environment 4: 12, 7–25.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” 2014 IEEE Conference on Computer Vision and Pattern Recognition.

Graham, D. A. (2017). “How did the Oroville Dam Crisis get so Dire?” The Atlantic

Gunay, H. B., Shen, W., and Yang, C. (2018). “Text-mining building maintenance work orders for component fault frequency.” Building Research & Information, 47(5), 518–533.

HappyCo: Real-Time Property Management, (2021) https://happy.co/

Hartigan, J. A., and Wong, M. A. (1979) “A k-means clustering algorithm”, Journal of Applied Statistics, 28, 100-108.

Hegazy, T. (2002), “Computer-Based Construction Project Management”. Prentice Hall, Upper Saddle River, NJ, USA.

Hegazy, T., Mostafa, K., and Ojulari, S. (2021). “Tetris-inspired approach for generating tightly-packed repetitive schedules.” Automation in Construction. 124: 103601.

Hegazy, T., Saad, D. A., and Mostafa, K. (2020). " Enhanced Repetitive-Scheduling Computation and Visualization." Journal of Construction Engineering and Management, 146(10), 04020118

Hernández-García, A., and König, P. (2018). “Further Advantages of Data Augmentation on Convolutional Neural Networks.” Artificial Neural Networks and Machine Learning – ICANN 2018 Lecture Notes in Computer Science, 95–103.

Hezaveh, M. M., Kanan, C., Salvaggio, C. (2017). “Roof Damage Assessment using Deep Learning” 2017 Applied Imagery Pattern Recognition Workshop (AIPR).

Hoang, N., Nguyen, Q., and Tran, V. (2018). “Automatic recognition of asphalt pavement cracks using metaheuristic optimized edge detection algorithms and convolution neural network.” Automation in Construction, 94, 203–213.

Hoang, N.-D., and Nguyen, Q.-L. (2018). “Metaheuristic Optimized Edge Detection for Recognition of Concrete Wall Cracks: A Comparative Study on the Performances of Roberts, Prewitt, Canny, and Sobel Algorithms.” Advances in Civil Engineering, 2018, 1–16.

Hollands, R. G. (2008). “Will the real smart city please stand up? Intelligent, progressive or entrepreneurial?” City, 12(3), 303–320.

Home Inspector Pro: Professional, Interactive, and Easy to Read Reports, (2021) https://www.homeinspectorpro.com/

HomeGauge: The future of the inspection industry, (2021) https://www.homegauge.com/

HomInspect Inspection Software, (2021) https://hominspect.net/

Horizon Inspection Software by Carson Dunlop, (2021) https://www.carsondunlop.com/landing/kaplan.html

Infrastructure Canada-Competition One (2019).

InspectCheck: Property Inspection App, (2021) https://www.inspectcheck.com/

Jain, A.K., (2010). "Data clustering: 50 years beyond k-means". Pattern Recognition Letters, 31(8), 651–666.

Jobber: Site Inspection Checklists and Job Forms, (2021) https://getjobber.com/

Johnson, D., (2020). "Deferred maintenance: Universities can't keep up with expensive upkeep and repairs." *Maclean's*.

Jordan, M.I., and Mitchell, T.M. (2015). "Machine learning: trends, perspectives, and prospects." Science, 349(6245), 255-260.

Kamarah, E. (2019). "Framework for scheduling, controlling, and delivery planning for scattered repetitive infrastructure rehabilitation projects." Ph.D. dissertation, Dept. of Civil and Environmental Engineering, Univ. of Waterloo.

Kamari, A., Kirkegaard, P.K., Schultz, C.P.L. (2021). "PARADIS - A process integrating tool for rapid generation and evaluation of holistic renovation scenarios," Journal of Building Engineering, 34,

Kim, B., and Cho, S. (2019). "Image-based concrete crack assessment using mask and region-based convolutional neural network." Structural Control and Health Monitoring.

Kingma, D. P., Ba, J. L. (2015). "ADAM: A Method for Stochastic Optimization". International Conference for Learning Representations, San Diego.

Kumar, S. S., Abraham, D. M., Jahanshahi, M. R., Iseley, T., and Starr, J. (2018). "Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks." Automation in Construction, 91, 273–283.

Li, B., Wang, K. C. P, Zhang, A., Yang, E., and Wang, G. (2020). "Automatic classification of pavement crack using deep convolutional neural network." International Journal of Pavement Engineering, 21(4), 457-463.

Link Inspect Pro: Property Inspection Software, (2021) https://linkinspectpro.com/

Liu, K., and El-Gohary, N. (2017). “Ontology-based semi-supervised conditional random fields for automated information extraction from bridge inspection reports.” Automation in Construction, 81, 313–327.

Liu, Z., Cao, Y., Wang, Y., and Wang, W. (2019). “Computer vision-based concrete crack detection using U-net fully convolutional networks.” Automation in Construction, 104, 129–139.

Luo, Q., Ge, B., and Tian, Q. (2019). “A fast adaptive crack detection algorithm based on a double-edge extraction operator of FSM.” Construction and Building Materials, 204, 244–254.

Lv, X., & El-Gohary, N. (2016). “Text Analytics for Supporting Stakeholder Opinion Mining for Large-scale Highway Projects.” Procedia Engineering, 145, 518-524.

Makantasis, K., Protopapadakis, E., Doulamis, A., Doulamis, N., and Loupos, C. (2015). “Deep Convolutional Neural Networks for efficient vision based tunnel inspection.” 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP).

Manville, C., Cochrane, G., Cave, J., Millard, J., Pederson, J. K., Thaarup, R. K., Liebe, A., Wissner, M., Massink, R., and Kotternik, B. (2014). “Mapping Smart Cities in the EU” European Parliament’s Directorate General for Internal Policies

Martinez, P., Mohamed, E., Mohsen, O., and Mohamed, Y. (2020). “Comparative Study of Data Mining Models for Prediction of Bridge Future Conditions.” Journal of Performance of Constructed Facilities, 34(1), 04019108.

McCallum, A., Nigam, K., and Ungar, L. H. (2000). “Efficient Clustering of High-Dimensional Data Sets with Application to Reference Matching”, International Conference on Knowledge Discovery and Data Mining, 169-178.

Meijer, A., and Bolívar, M.P.R., (2016). “Governing the smart city: a review of the literature on smart urban governance.” International Review of Administrative Sciences, 82 (2), 392–408.

MERX (2011). "RFP: Condition Assessment Program for Education Facilities in Ontario " MERX, Ministry of Government Services, Ontario Shared Services Supply Chain Management Procurement Advisory Branch.

Mo, Y., Zhao, D., Syal, M., and Aziz, A. (2017). “Construction Work Plan Prediction for Facility Management Using Text Mining.” Computing in Civil Engineering 2017.

Mohanty, P. S., Choppali, U., and Kougianos, E. (2016). “Everything You Wanted to Know about Smart Cities” IEEE Consumer Electronics Magazine

Mostafa, K., Attalla, A., and Hegazy, T. (2021). “Data mining of school inspection reports to identify the assets with top renewal priority.” Journal of Building Engineering, 41, 102404.

Nam, T., and Pardo, T. A. (2011). “Smart city as urban innovation: Focusing on Management, Policy, and Context.” Proceedings of the 5th International Conference on Theory and Practice of Electronic Governance - ICEGOV '11.

Oralhan, Z., Oralhan, B., and Yiğit, Y. (2017). “Smart city application: Internet of things (IoT) technologies based smart waste collection using data mining approach and ant colony optimization.” International Arab Journal of Information Technology, 14(4), 423–427.

Patel, M. (2019). “Understanding the Role of Smart City & its Components in the IoT Era.” eInfochips.

Perez, H., and Tah, J. H. (2021). “Deep learning smartphone application for real-time detection of defects in buildings.” Structural Control and Health Monitoring, 28(7).

Perez, H., Tah, J. H., and Mosavi, A. (2019). “Deep Learning for Detecting Building Defects Using Convolutional Neural Networks.” Sensors, 19(16), 3556.

Piangiani, G. (2020). “Poor Maintenance and Construction Flaws Are Cited in Italy Bridge Collapse” The New York Times.

Pramanik, M. I., Lau, R. Y. K., Demirkan, H., and Azad, M. A. K. (2017). “Smart health: Big data enabled health paradigm within smart cities.” Expert Systems with Applications, 87, 370–383.

Raja, A. K., and Pang, Z. (2016). “High accuracy indoor localization for robot-based fine-grain inspection of smart buildings.” IEEE International Conference on Industrial Technology (ICIT)

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). “You Only Look Once: Unified, Real-Time Object Detection.” IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Rena, G., Heob, Y., Sunikka-Blanka, M. (2019). “Investigating an adequate level of modelling for retrofit decision-making: A case study of a British semi-detached house,” 38.

Reporthost Home Inspection Software, (2021) https://www.reporthost.com/

Roberts, D., and Golparvar-Fard, M. (2019). “End-to-end vision-based detection, tracking and activity analysis of earthmoving equipment filmed at ground level.” Automation in Construction, 105, 102811.

Sachgau, O. (2016). “Schools crumbling amid $1B repair shortfall.” thestar.com.

Rushowy, K. (2019). “Repair backlog in Ontario schools hits $16.3 billion.” thestar.com.

Seghal, G., and Garg K. (2014). “Comparison of Various Clustering Algorithms.”, International Journal of Computer Science and Information Technologies, 5(3), 3074-3076

Seong, H., Choi, H., Cho, H., Lee, S., Son, H., and Kim, C. (2017). “Vision-Based Safety Vest Detection in a Construction Scene.” Proceedings of the 34th International Symposium on Automation and Robotics in Construction (ISARC).

Sharma, N., Bajpai, A., and Litoriya, R. (2012). “Comparison the various clustering algorithms of weka tools.”, International Journal of Emerging Technology and Advanced Engineering, 2(5), 73-80.

Sharma, S., Tiwari, R., Shukla, A., and Yadav, J. (2014). “Canopy Clustering Based Multi Robot Area Exploration.”, Third International Conference on Advances in Control and Optimization of Dynamical Systems, Kanpur, India.

Silva, B. N., Khan, M., and Han, K. (2018). “Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities.” Sustainable Cities and Society, 38, 697–713.

Smart Brantford (n.d.). “Smart Governance: Smart Brantford.”

SmartCitiesWorld (2017). “Smart cities services worth $225bn by 2026.”.

Son, H., Choi, H., Seong, H., and Kim, C. (2019). “Detection of construction workers under varying poses and changing background in image sequences via very deep residual networks.” Automation in Construction, 99, 27–38.

Spectora: Top-Rated Home Inspection Software, (2021) https://www.spectora.com/

TDSB (2021). “Toronto District School Board, Renewal Needs Backlog.” https://www.tdsb.on.ca/About-Us/Accountability/Renewal-Needs-Backlog-and-Facility-Condition-Index/Renewal-Needs-Backlog, Accessed Aug. 11, 2021.

TDSB (2020). “2020-2021 Operating Budget.” https://www.tdsb.on.ca/Portals/0/docs/5_1.pdf, Accessed Jul. 7, 2021

VFA capital Planning Software, (2021) https://www.gordian.com/resources/vfa-capital-planning-software/

Vuppulurri, P. (2020) “Investing In Innovation: The Rise Of The Smart City.” Forbes.

Wang, B., Li, Y., Zhao, W., Zhang, Z., Zhang, Y., and Wang, Z. (2019a). “Effective Crack Damage Detection Using Multilayer Sparse Feature Representation and Incremental Extreme Learning Machine.” Applied Sciences, 9(3), 614.

Wang, J., Hou, J., Chen, J., Fu, Q., Huang, G. (2021). “Data mining approach for improving the optimal control of HVAC systems: An event-driven strategy,” Journal of Building Engineering, 39.

Wang, K. C., Li, Q. J., Yang, G., Zhan, Y., and Qiu, Y. (2015). “Network level pavement evaluation with 1 mm 3D survey system.” Journal of Traffic and Transportation Engineering (English Edition), 2(6), 391–398.

Wang, N., Zhao, X., Zhao, P., Zhang, Y., Zou, Z., and Ou, J. (2019b). “Automatic damage detection of historic masonry buildings based on mobile deep learning.” Automation in Construction, 103, 53–66.

Weka 3: Data Mining with Open Source Machine Learning in Java, (2020) https://www.cs.waikato.ac.nz/ml/weka/..

Williams, T. P., and Betak, J. F. (2016). “Identifying Themes in Railroad Equipment Accidents Using Text Mining and Text Visualization.” International Conference on Transportation and Development 2016.

Yang, M.-D., Chao, C.-F., Huang, K.-S., Lu, L.-Y., and Chen, Y.-P. (2013). “Image-based 3d scene reconstruction and exploration in augmented reality.” *Automation in Construction*, 33, 48–60.

Yousaf, M. H., Azhar, K., Murtaza, F., and Hussain, F. (2018). “Visual analysis of asphalt pavement for detection and localization of potholes.” Advanced Engineering Informatics, 38, 527–537.

Yudin, D. A., Adeshkin, V., Dolzhenko, A. V., Polyakov, A., and Naumov, A. E. (2021) “Roof Defect Segmentation on Aerial Images Using Neural Networks” Advances in Neural Computation, Machine Learning, and Cognitive Research IV, 175-183

Zeiler, M., & Fergus, R. (2013). “Visualizing and understanding convolutional networks.” European Conference on Computer Vision, 8689, 818-833

Zhang, A., Wang, K. C. P., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q., and Chen, C. (2017). “Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network.” Computer-Aided Civil and Infrastructure Engineering, 32(10), 805–819.

Zhao, D., McCoy, A. P., Kleiner, B. M., Du, J., and Smith-Jackson, T. L. (2016). "Decision making chains in electrical safety for construction workers." Journal of Construction Engineering and Management, 142(1), 04015055.

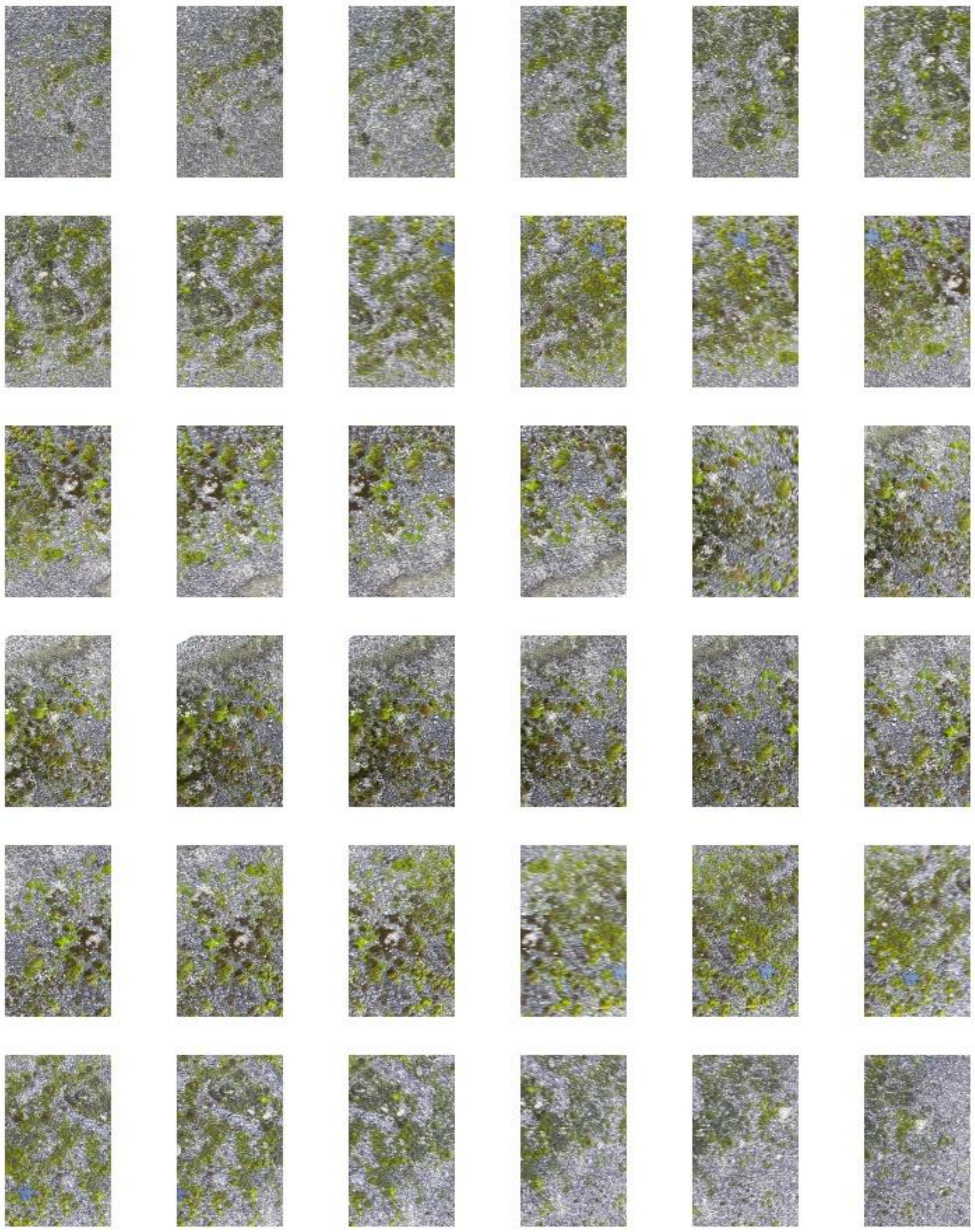
Zhou, X., Yang, T., Liang, L. Zi, X., Yan, J., Pan, D. (2021). “Anomaly detection method of daily energy consumption patterns for central air conditioning systems,” Journal of Building Engineering, 38.

Zhou, X., Wanga, B., Liang, L., Yana, J., Pana, D. (2019). “An operational parameter optimization method based on association rules mining for chiller plant,” Journal of Building Engineering, 26.

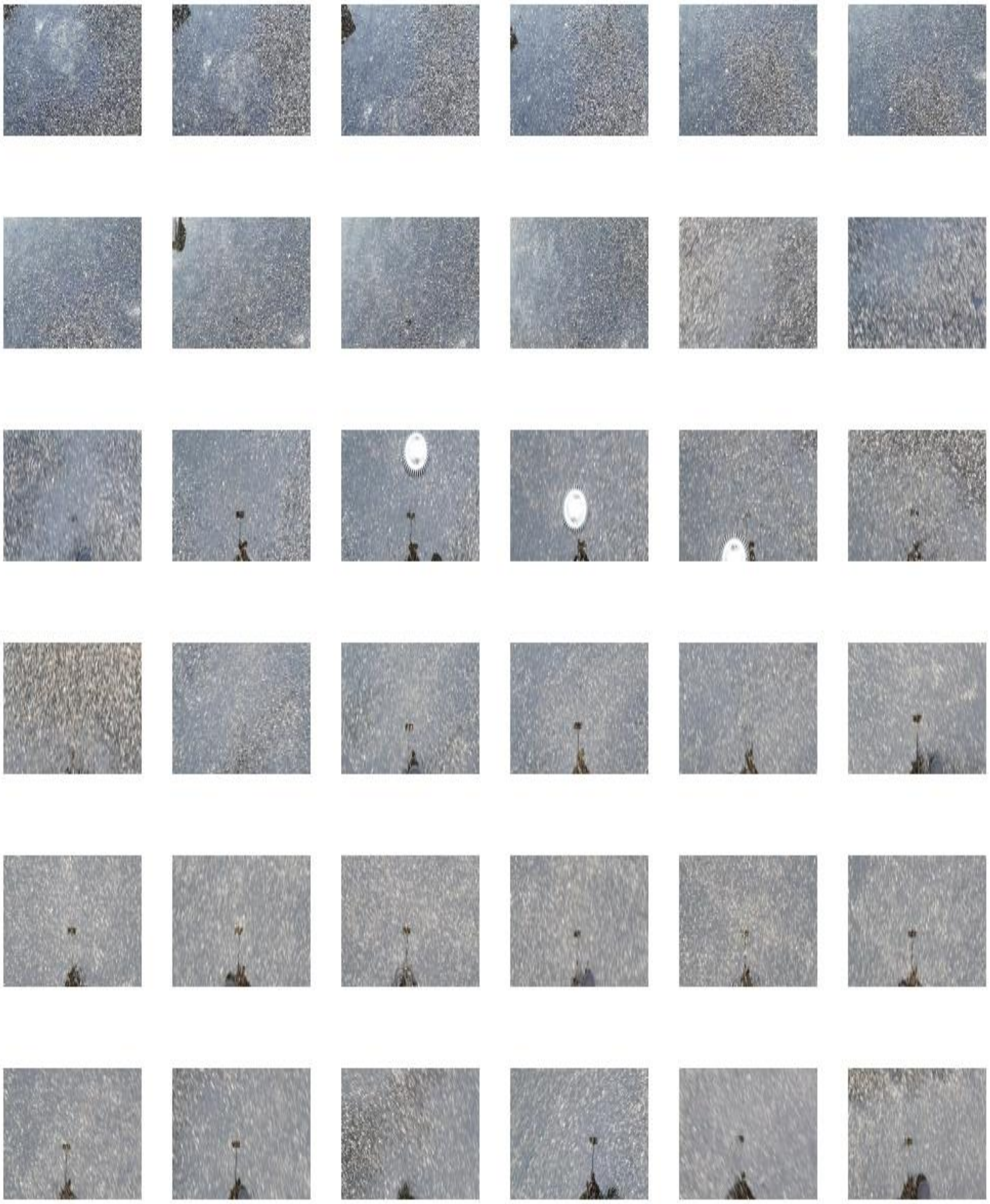
Zinspector: Property Inspection Solution, (2021) https://www.zinspector.com/

Appendix A: Sample of Collected Images

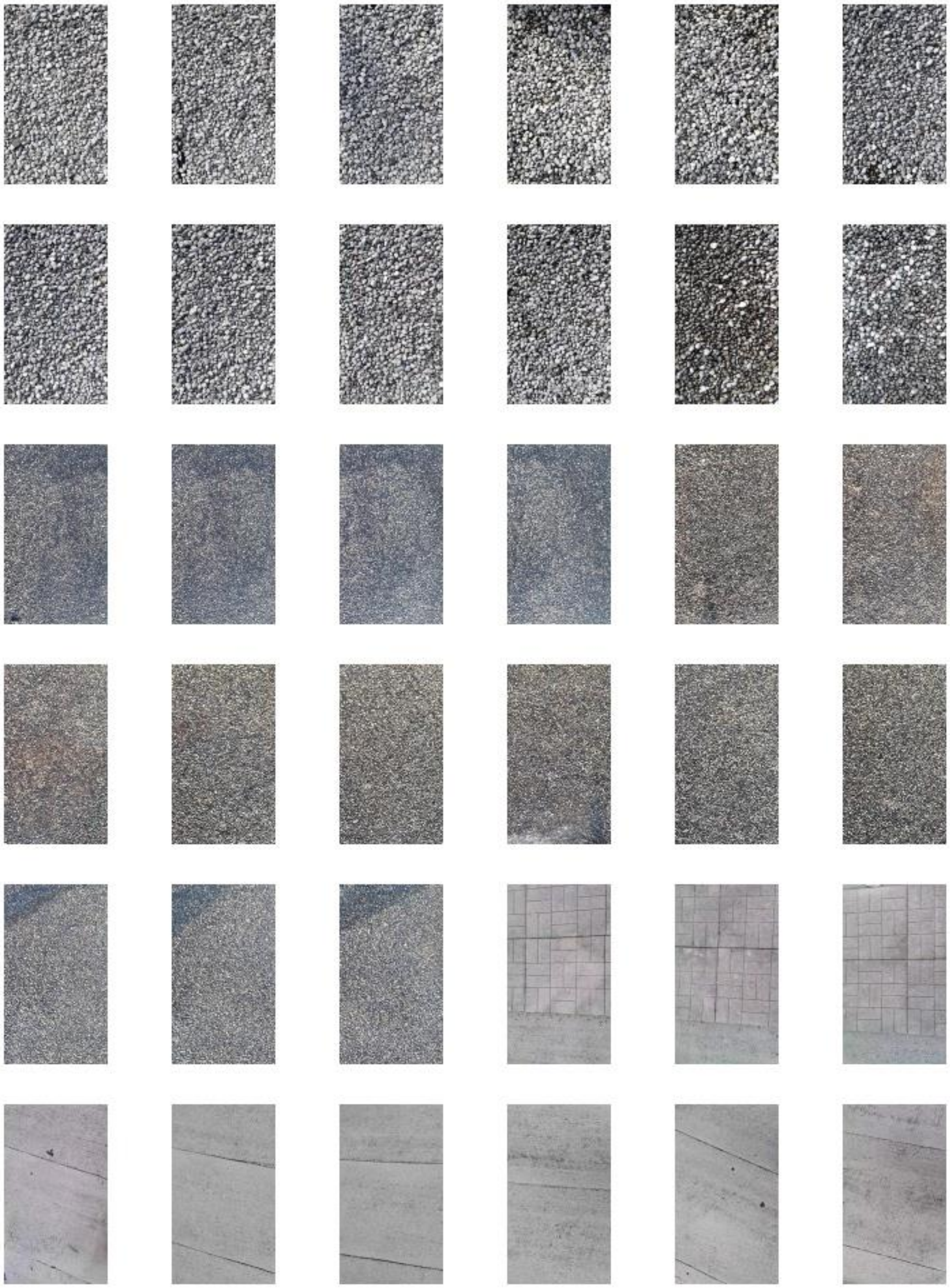
Vegetation Images



Ponding Images



No Defect Image



Appendix B: Python Code for Convolutional Neural Network

Model.py Module

```
# import pytorch CNN libraries
import torch
import torch.nn as nn

#Define the CNN architecture
class CnnPHD(nn.Module):
  def __init__(self):
    super(CnnPHD, self).__init__()
    self.conv1 = nn.Conv2d(in_channels=3, out_channels=24, kernel_size=20, stride=2)
    self.pool1 = nn.MaxPool2d(kernel_size=7, stride=2)
    self.conv2 = nn.Conv2d(24, 48, 15, stride=2)
    self.pool2 = nn.MaxPool2d(kernel_size=4,stride=2)
    self.conv3 = nn.Conv2d(48,96,10,stride=2)
    self.conv4 = nn.Conv2d(96,2,1,stride=1)
  def forward(self,x):
    L1 = self.conv1(x)
    L2 = self.pool2(L1)
    L3 = self.conv2(L2)
    L4 = self.pool2(L3)
    L5 = self.conv3(L4)
    L6 = nn.ReLU(inplace=True)(L5)
    L6 = nn.Dropout2d(p=0.5)(L6)
    L7 = self.conv4(L6)
    L8 = nn.Softmax(dim=1)(L7)
    L8 = L8.reshape(-1,2)
    return L8
```

Dataset.py Module

```
# importing necessary libraries:
import os
import torch
from PIL import Image
from torch.utils.data import Dataset


class slicesDataset(Dataset):

  # Going to image directory and collecting all JPG images

   def __init__(self, root ,train=True, transform = None):
      Dataset.__init__(self)
      images_dir = os.path.join(root,'')
      images = os.listdir(images_dir)
      self.images = [os.path.join(images_dir, k) for k in images if 'jpg' in k]
      self.images.sort()

      self.transform = transform
      self.train = train

   #Image labelling: image label=1 if it has the letters 'CLR' in its name indicating no defects
   #For Image Classification CNN, image label=1 if it has the letters ‘VEG’ in its name indicating a
vegetation defect

   def __getitem__(self, index):
      img_dir = self.images[index]
      img = Image.open(img_dir).resize([256,256])
      img = self.transform(img)
      if self.train:
         lbe = 1 if 'CLR' in img_dir else 0
         return img, lbe
      return img

   def __len__(self):
      return len(self.images)

  #Weighted sampling for the batches, depending on the ratio between the images carrying different
labels

   def __weight__(self):
      pos = sum(1 for x in self.images if 'CLR' in x)
      neg = len(self.images) - pos
      weights = [1/neg, 1/pos]
      class_weights = torch.FloatTensor(weights)
      return class_weights
```

Myutils.py Module

```
#Import necessary libraries
import os
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
from PIL import Image
import numpy as np
import pandas as pd

#Calculating Accuracy: Number of correct predictions/total number of datapoints
def Acc(model, loader):
    with torch.no_grad():
        acc = total = 0
        for images, labels in loader:
            images = images.cuda()
            labels = labels.cuda()
            outputs = model(images)
            acc += (outputs.argmax(1)==labels).sum().item()
            total += labels.size(0)
        return acc/total*100

#Function to return predicted labels vs. actual labels as a dataframe
def lbls(model, loader):
    with torch.no_grad():
        results=[]
        for images, labels in loader:
            images = images.cuda()
            labels = labels.cuda()
            outputs = model(images)
            results.append([labels.cpu().numpy(),outputs.cpu().detach().numpy()])
        results_df=pd.DataFrame(results[0][1],results[0][0]).reset_index().rename(columns={"index":"true
label"})

        return results_df

#Transforming the images to a format that the CNN can read
transform=transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize(mean=[.5,.5,0.5],std=[.5,.5,0.5])
])
```

Train.py Module

```
#Import necessary libraries
import os
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
from PIL import Image
import numpy as np
from dataset import slicesDataset
from model import CnnPHD
import logging
import copy
import pandas as pd
from myutils import Acc, transform, lbls
from datetime import datetime

#Hyperparameters
batch_size = 64
num_epochs = 1000
num_workers = 0 #means use all GPU power
lr = 0.0001 # learning rate
torch.cuda.manual_seed(7)
torch.manual_seed(7)

torch.backends.cudnn.enabled = False  #avoid CUDA OUT OF MEMORY error

#Loading the images for training and validation datasets, refer to the "dataset.py" functions for more
detail
train_dataset = slicesDataset('../train_images', transform=transform)
val_dataset = slicesDataset('../val_images', transform=transform)
train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size,
                shuffle=True,num_workers=num_workers, pin_memory=True)
val_loader = DataLoader(dataset=val_dataset, batch_size=batch_size,
                shuffle=True, num_workers=num_workers, pin_memory=True)


#Calling the CNN model
init_cnn = CnnPHD().cuda()
print(Acc(init_cnn, val_loader))

history = []
cnn = copy.deepcopy(init_cnn)

#Loss function and optimizer
criterion = nn.CrossEntropyLoss(weight=train_dataset.__weight__().cuda())
optimizer = torch.optim.Adam(cnn.parameters(), lr=lr)
logging.basicConfig(filename='../logs/logger_cnn_adam.log', level=logging.INFO)
total_step = len(train_loader)
```

Train.py Module (Cont.)

```
start_time=datetime.now()
print("TRAINING START at: ",start_time)

#The Training Loop
for epoch in range(num_epochs):
   totalloss = 0
   for i, (images, labels) in enumerate(train_loader):

      #Forward pass
      images = images.cuda()
      labels = labels.cuda()
      outputs = cnn(images)
      loss = criterion(outputs, labels)
      totalloss += loss*images.size(0)

     #Backward pass
      optimizer.zero_grad()
      loss.backward()
      optimizer.step()

      #Store Model Accuracy
      if i==total_step-1:
         valacc =  Acc(cnn,val_loader)  #refer to "myutils" functions for more detail
         trainacc = Acc(cnn,train_loader)
         history.append([totalloss.item()/len(train_dataset),trainacc,valacc])

      #Saving and Timestamping the model parameters
         torch.save(cnn.state_dict(),\
          '../trained_models/cnn_adam_'+datetime.now().strftime('%Y-%m-
%d_%H%M_')+str(epoch+1)+'.pkl')

#Display function, so we know how much work is left
   if epoch%100==0:
      print('Epoch [{}/{}], Time is: '
            .format(epoch + 1, num_epochs, i + 1), datetime.now())

#Report total training time for all epochs
finish_time=datetime.now()
print("total Training time=",finish_time-start_time)

#Save the accuracy results for training and validation (create Figs. 4.8, 4.9)
history_df=pd.DataFrame(history,columns=['loss','Training accuracy(%)','Validation accuracy(%)'])
history_df.to_csv('../History.csv',index=True)
```

Predict.py Module

```
#Import Necessary Libraries
import os
import torch
from PIL import Image
import numpy as np
from model import CnnPHD
from myutils import transform, Acc, lbls
import pandas as pd
from dataset import slicesDataset
from torch.utils.data import Dataset, DataLoader

torch.backends.cudnn.enabled = False

#Loading image folders (one folder per school)
# '../' refers to the folder above the working directory (i.e., folder above the folder containing the code files)
big_folder = '../FreeVideoToJPGConverter/'
big_dir = os.listdir(big_folder)
big_dir.sort()
big_list = [big_folder+k+'/' for k in big_dir if '.' not in k]
big_list.sort()
i=0
#Preparing images from within the folder to be loaded by the dataset.py module (folder-by-folder)
for folder in big_list:
   img_folder = folder
   img_dir = os.listdir(img_folder)
   img_dir.sort()
   img_list = [img_folder+k for k in img_dir]
   img_list.sort()

#Loading model parameters
   cnn = CnnPHD().cuda()
   cnn.load_state_dict(torch.load('../trained_models/cnn_adam_163.pkl'))

   #Loading image dataset
   pred_dataset = slicesDataset(img_folder, transform=transform)
   pred_loader = DataLoader(dataset=pred_dataset, batch_size=len(img_dir),
                  shuffle=False, num_workers=0, pin_memory=True)

   #performing prediction and saving results
   pred_df=lbls(cnn,pred_loader)
   #print(Acc(cnn,pred_loader))
   pred_df.index=img_dir
   pred_filename=big_folder+big_dir[i]+'.csv'
   i=i+1
   pred_df.to_csv(pred_filename,index=True)
```