Doctoral Dissertation

Automatic semantic classification of photogrammetry-based image feature points and point cloud using deep learning

SAOVANA NATTHAPOL

January 2021

Graduate School of Engineering,
Osaka University

Abstract

Digital images from a construction site such as an infrastructure project are constantly used as a tool for visualization because of their easy utilization and cost-effectiveness. These images can then be processed into photogrammetric techniques such as Structure from Motion (SfM) to construct point clouds of an existed scene based on image features inside the overlapping of the scene. These features can be detected and matched using feature detector and descriptor algorithms, which is the fundamental step of numerous photogrammetric processes. However, feature matching is prone to error from the excessive features of noises and outliers, especially with the infrastructure scene, where are full of the uncontrollable light condition and feature-rich components such as vegetation and boulders. The concepts that each feature detector and descriptor differently implements also provide advantages and disadvantages among these algorithms, in which it should be validated first before using with each expertise but in infrastructure scope, it has not yet to be seen. Moreover, the final output point clouds from photogrammetric techniques are usually full of noises and unwanted features that do not belong to the object of interest. These point clouds need to be classified and eliminated the unwanted cloud points out before utilization. Achieving the classified point cloud manually is troublesome, labor-intensive, and costly, therefore there is a need to support this work process, in which a deep convolutional neural network (DCNN) may prove its usefulness. Nevertheless, the major drawback of implementing DCNN is the necessity to have a gigantic amount of training data, which is a three-dimensional point cloud. Acquiring the training point cloud in the outdoor scene is very difficult due to the preparation of work and there are less public data available in this field. Thus, the image feature points and point cloud classification should be further investigated for the probability to use other training data other than the three-dimensional point cloud.

This study proposed four steps of work in order to improve the current image feature points and point cloud classification for infrastructure projects. First, the performance of each feature detector and descriptor algorithm was compared to each other to find a suitable algorithm for infrastructure scope. Next, the effect of unwanted features with the feature matching was investigated and evaluated the result if the unwanted features were to be removed before feature matching. Then, the investigation was expanded to the effect of unwanted features with the point cloud generation. A novel system to eliminate the unwanted features based on semantic segmentation was also proposed to facilitate the workload of practitioners. Finally, another new approach to classify a three-dimensional point cloud from SfM is proposed. Point cloud Classification based on image-based Instance Segmentation (PCIS) utilizes the mask images from instance segmentation to classify the object of interest in the two-dimensional scene and converts these masks using the corresponding camera pose solved from SfM into the three-dimension environment. The cloud points that located inside the projection from these masks and their respective camera position are classified as the cloud points of the class as same as the mask. All of this research is beneficial to both professional practitioners and academic staff. Site practitioners can utilize the proposed system to facilitate the workload of unwanted feature removal and point cloud classification. On the other hand, this research fills the knowledge gap that unwanted feature removal should be implemented before the feature matching and other academic staff can gain the benefit of

having a suggestion for choosing feature detector and descriptor algorithm when the domain of the problem relates to infrastructures.

Preface

This dissertation is the original work by Natthapol Saovana under the supervision of Professor Nobuyoshi Yabuki. Three journal articles and three international conference proceedings that relate to this dissertation have been submitted or published and are listed below.

Journal articles:

1. Saovana, N., Yabuki, N., and Fukuda, T. (2020). Performance Assessment of Feature Detector and Descriptor Algorithms for Scaffold Digital Images. Journal of Applied Survey Technology (JAST), 31, 13–24.
2. Saovana, N., Yabuki, N., and Fukuda, T. (2020). Development of an unwanted-feature removal system for Structure from Motion of repetitive infrastructure piers using deep learning. Advanced Engineering Informatics, 46(July), 101169. https://doi.org/10.1016/j.aei.2020.101169
3. Saovana, N., Yabuki, N., and Fukuda, T. Automated point cloud classification using an image-based instance segmentation for Structure from Motion. *Automation in Construction*, Under review.

International conference proceedings:

1. Saovana, N., Yabuki, N., and Fukuda, T. (2019). A Performance Evaluation of Feature Detectors and Descriptors for Unmodified Infrastructure Site Digital Images. Proceedings of the 4th International Conference on Civil and Building Engineering Informatics (ICCBEI2019), 113–120.
2. Saovana, N., Yabuki, N., and Fukuda, T. (2019). A Quantitative Effect Evaluation of the Unwanted Features Removal of Infrastructure Digital Images. Proceedings of the 19th International Conference on Construction Applications of Virtual Reality (CONVR2019), 62–71.
3. Saovana, N., Yabuki, N., and Fukuda, T. (2020). An image feature classification system for piers based on semantic segmentation with deep learning. Proceedings of the 20th International Conference on Construction Applications of Virtual Reality (CONVR2020), 325–335.

Acknowledgment

This doctoral dissertation would not have been possible without the assistance, support, or guidance of the following people and organizations.

First of all, I would like to express my deepest gratitude to Professor Nobuyoshi Yabuki, my supervisor, who constantly has supported me in every way I can imagine, especially in the critical last months before dissertation submission. He has plentifully allocated his time to me so that I can have more time to conduct my research and graduate in time. I would also like to show my gratitude to Associate Professor Tomohiro Fukuda. His critical thinking is very good and beneficial for my manuscripts. He can always point out details for better development.

Next, I would like to thank my family, Chanvit, Subhavadee, and Natthida Saovana, my father, mother, and sister, respectively, and my beloved, Kanyanut Keskomon. They always support me, both physically and mentally, and are there for me when I need assistance.

I would like to thank all of my friends in Yabuki Lab, for example, Mr. Izutsu, Ms. Manuel, Ms. Yixi, Mr. Ishikawa, Mr. Tanaka, Mr. Yamamoto, Mr. Zhu, and Mr. Kubota, and my Thai friends in Osaka. I cannot imagine my life in Japan without them.

I am greatly thankful to the dissertation committee for spending their time reading this dissertation and providing me valuable feedback and recommendations. I also would like to thank Professor Masanori Sawaki for being the vice referee for my Ph.D. examination.

Lastly, I would like to show my gratitude for the financial support from the Royal Thai Government Scholarship. Without this scholarship, I would have stopped pursuing my dream of becoming a Ph.D. graduate. I would like to thank all of the staff in the Royal Thai Embassy also for all the warm and kind support they have given me for all these years.

I will treasure these three years as one of my greatest life experiences for sure.

Table of contents

List of figures

List of tables

Chapter 1
Introduction

1.1 Background and problem statements

Digital images are widely utilized to track progress on construction sites (Jadidi et al., 2014; Yang et al., 2015) and monitor infrastructure health (Agnisarman et al., 2019) because of their ease of use and are cost-effectiveness (Hamledari et al., 2017). These images can be processed using Structure from Motion (SfM), which is an inexpensive yet powerful photogrammetric technique for constructing point clouds or models of existing scenes, structures, and objects (Golparvar-Fard et al., 2015; Raoult et al., 2016; Snavely et al., 2008). Although laser scanners can also produce point cloud similar to photogrammetric techniques with a better precision up to a millimeter accuracy (Tang et al., 2010), there are various limitations that need to be considered such as the cumbersome scan-point planning (Wang et al., 2018), prohibitively expensiveness of the laser scanner (Bosché, 2010), and long calibration time that prolongs the work duration (Golparvar-Fard et al., 2015). These difficulties of using laser scanners make image-based methods such as SfM still be desirable when the processing for point cloud generation is needed (Lu and Lee, 2017).

By using point clouds or models, changes over time can be monitored, which is useful for equipment tracking (Fang et al., 2016), construction progress estimation (Brilakis et al., 2011), and infrastructure inspection (Rashidi and Karan, 2018). Feature detection and description algorithms can be used to detect important features in each image by matching the same pixel in the overlap between images. These matched features are then merged into the same coordinate environment, which is called ‘image registration’ (Tareen and Saleem, 2018).

Image registration plays a vital role in image processing (Domokos et al., 2008), especially in remote sensing and surveying, which have numerous challenges such as continual changes in lighting conditions, such as from cloud formation (Simonson et al., 2007). Variation in the image background (Oron et al., 2018), such as that caused by the movement of trees and vehicles, can also decrease quality during the registration process. The pixels of unwanted features can lead to false feature matches or decrease true feature matches inside the region of interest (ROI) (Saovana et al., 2019b). High levels of complexity can disturb the separation between good and bad matches in the image (Bian et al., 2017), causing the feature detection and description algorithms to discard numerous correct matches to ensure the accuracy of the entire feature matching process (Lin et al., 2018), thereby decreasing the image registration quality. Furthermore, these false matches, together with unwanted features outside the ROI, can generate a point cloud during the SfM process that consists of undesirable components such as vegetation, buildings, and parked vehicles that obstruct target areas and decrease the overall quality of the point cloud and models (Ning et al., 2018).

Furthermore, a plain point cloud does not provide much information (Su et al., 2013). It has to be classified into objects inside these points for further processing and visualization such as a bridge pier in an infrastructure scene (Saovana et al., 2020a) and traffic signs in an urban area (Rastiveis et al., 2020). Yet, point cloud classification and cleaning are tedious and troublesome work (Weidner et al., 2019), particularly when they are handled manually.

Numerous studies have tried to facilitate this problem by proposing automated techniques, among which one of the most important methods is to use deep learning systems to automate the process (Uy et al., 2019). Nevertheless, deep learning requires a huge amount of training data in order to obtain high accuracy (Merkurjev, 2020), which makes it very challenging for point cloud classification of infrastructure piers due to the lack of training point cloud data, resulting in overfitting (Arief et al., 2019). Moreover, infrastructure scenes are untidy, containing vegetation and boulders, which usually makes the point cloud outputs from the SfM technique to be full of noise (Zhu et al., 2017) and have varying point density (Kang and Yang, 2018), and this leads to errors when utilized without proper classification (Park and Guldmann, 2019).

To eliminate the difficulty of deep learning training using point clouds, various studies have transformed the input into images (Hackel et al., 2017) because it is easier to obtain more training data and deep learning also performs well in this scenario (Roveri et al., 2018). However, the images generated from point clouds are not realistic owing to the variety of camera directions, which is different from images of real-world scenes (Fu et al., 2020) that have constraints, such as the sky always has to be higher than the ground. Therefore, images from the actual scene need to be used in order to preserve the realistic camera poses in the solved camera parameters from the SfM technique, but the combination of this information with point clouds for classification (Pessoa et al., 2019) is yet to be implemented.

Considering these opportunities and limitations, a new system based on an instance segmentation technique for digital images to classify SfM point clouds called point cloud classification based on image-based instance segmentation (PCIS) is proposed. First, the original images of an actual construction scene are processed through a pre-trained deep convolutional neural network (DCNN) to segment out the objects of interest and form mask images for the subsequent steps. These original images are then used as inputs for the SfM process to construct a point cloud and solve for the camera poses and parameters such as rotation matrix, translation matrix, and focal length. Next, the mask images from the DCNN are placed at their calculated positions instead of their corresponding input images in the created point cloud scene based on the camera variables solved in the previous step. Finally, the respective projection rays are constructed from each camera position through the mask to the point cloud to form a three-dimensional mask, and the cloud points that are located inside this mask are classified as the cloud points of the object of interest. The effect of unwanted features with the feature matching and point cloud generation is also investigated. They can be eliminated when the masks are created by removing the features or point clouds that belong to the unwanted class.

PCIS can solve the main problem of deep learning with point cloud classification, which is the lack of point cloud training data. The training images required by PCIS are taken from the actual construction site, where these images have already been captured as a part of the work (Jadidi et al., 2014; Yang et al., 2015). Therefore, there is no need for further data acquisition. Various studies of automated point cloud classification have paid attention to the difficulties of handcrafted point geometry (Wen et al., 2020). However, our work process does not require any adjustment except the camera parameters, which can be solved automatically by the SfM process.

1.2 Research objectives

Automatic image feature points and point cloud classification can relieve a gigantic workload from site practitioners by supporting the classification, cleaning, and raising the cloud points inside the constructed infrastructure scene based on photogrammetric techniques such as SfM, which is usually challenging because of the feature-rich components, and wide coverage of the site. It can transform the ROI point cloud into a very small minority inside the entire point cloud. Moreover, due to the lack of available point cloud data, the developed system must be able to implement digital images from a daily work basis as training data for point cloud classification. Therefore, the objectives of this research are as follows:

(1) To evaluate the quantitative performance of feature detector and descriptor algorithms when they have to manage digital infrastructure scene images.
(2) To investigate the effect of unwanted feature removal on feature matching and point cloud generation.

(3) To develop a method that can automatically classify and eliminate unwanted features based on digital site images.
(4) To develop a method that can classify image features in two-dimensional scenes and cloud points from a photogrammetric technique in a three-dimensional environment based on digital site images.

1.3 Research significance

This research will provide an understanding of the photogrammetric technique from the fundamental level such as feature detector and descriptor algorithms with image features to higher levels such as unwanted feature removal and point cloud classification. This research presents various methods for improving photogrammetric work procedures, which will be beneficial for both practitioners and academic staff. The practitioners can implement the proposed automatic system in this study to manage the unwanted features and point clouds when they visualize their job site with SfM. On the other hand, the academic staff can use this work to support the selection of feature detector and descriptor algorithms for the infrastructure scope and utilize these experimental results to discuss further improvement in photogrammetry and deep learning.

1.4 Research scope

The focused environment of this research is the civil infrastructure component such as monorail piers and bridge piers because there is a research gap between indoor and outdoor scenes in the photogrammetric scope with deep learning. The outdoor environment is challenging because it is more random to factors such as the uncontrollable light condition and full of unique-shape objects with a feature-rich texture that can complicate the feature matching such as the moving vegetation and boulders.

1.5 Overview of the dissertation

In order to facilitate the labor-intensive and time-consuming feature point and point cloud classification, the understanding from the fundamental level of photogrammetric techniques is needed to be investigated. This dissertation is separated into 7 chapters, which the main concepts of each chapter are shown in **Figure 1.1**. The next chapter presents about the related literature with this dissertation. Next, the evaluation of feature detector and descriptor algorithms is conducted in Chapter 3, which is related to the first research objective. Then, the selected algorithms from Chapter 3 are used to evaluate the effect between unwanted feature removal with feature matching that relates to the second research objective. The evaluation is further expanded to compare the influence of unwanted feature removal with the point cloud generation in Chapter 5 along with the development of an automated unwanted feature removal based on semantic segmentation. This chapter associates with the second and third

research objectives. After the unwanted features can be managed, there is still a need for a novel method to facilitate the traditional classification that requires handcrafted classifiers or numerous feature points and point clouds for training. In Chapter 6, an automated classification system for feature points and point clouds based on image segmentation deep learning is demonstrated, which relates to the last research objective. Finally, the last chapter, Chapter 7, concludes all of the findings in each chapter. The detail of each chapter are further explained as follows:

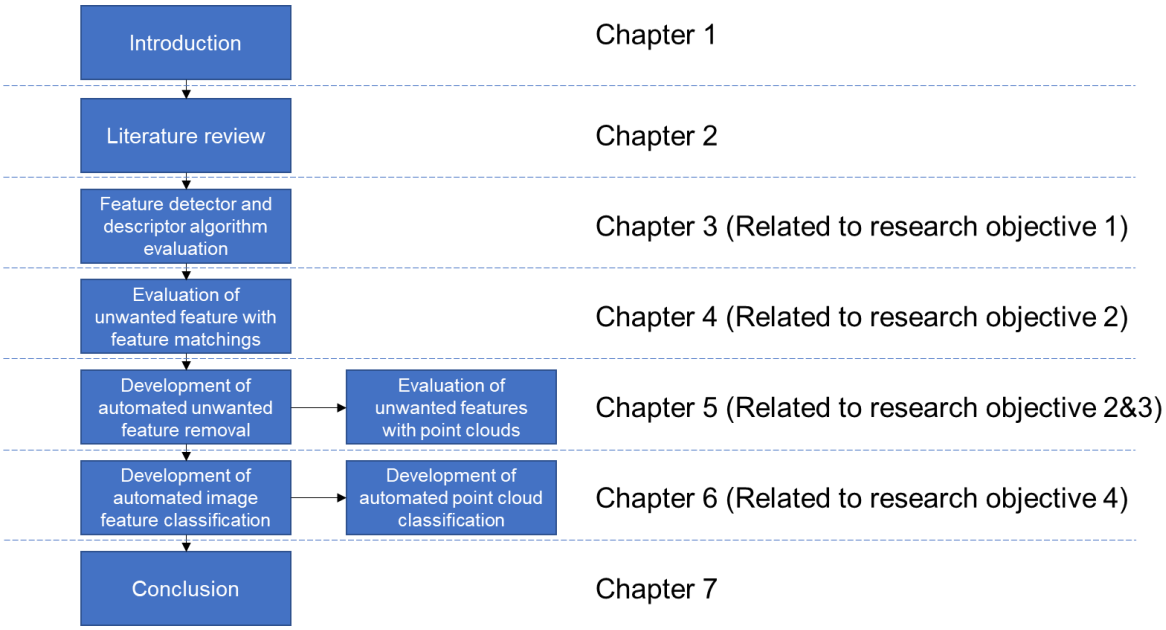


Figure 1.1 Overview of the main concepts in each chapter

Chapter 1 Introduction:

This chapter explains the background information, problem statements, research objectives, research significance, research scope, and overview of all chapters in this dissertation.

Chapter 2 Literature review:

This chapter presents the related works with this study. It is separated into four main sections. The first section explains the information of computer vision that relates to this research. It is further separated into four subsections, which are image features, feature matching, SfM, and deep learning. The second section provides a brief review of infrastructure visualization. The third chapter explains the limitations of the current workflow between site visualization and deep learning. The fourth chapter summarizes this chapter.

Chapter 3 The validation of feature detector and descriptor algorithms:

A comparative study about the performance of each feature detector and descriptor algorithm is described in this chapter. The sample images in this section are the piers of a bridge motorway. The results are compared in two aspects, which are ROI feature detection and quantitative comparison of feature matching and computational time. The sequence of the study is separated into five parts, namely the proposed method, experiment, result, discussion, and conclusion of this chapter.

Chapter 4 Effect evaluation of the unwanted features on feature matching:

This chapter evaluates the relationship between unwanted features and feature matching. The input images of civil infrastructures are used as testing samples. First, unwanted features are removed from the input images. Then, feature matching was done on these images. Finally, the quantitative performances between feature matchings of the original images and images with unwanted feature removed are compared. The results are discussed and summarized.

Chapter 5 Unwanted feature removal with point clouds:

This chapter further investigates the effect of unwanted features with point clouds. The testing samples are constructed from three sources, the original images, the unwanted feature removal images based on a new proposed system using semantic segmentation from DCNN, and the manually unwanted feature removal images. Finally, the in-depth comparison between the alignable images, point cloud, and computational time are conducted, discussed, and analyzed.

Chapter 6 Automated feature points and point cloud classification:

This chapter presents the development of a novel feature point and point cloud classification based on instance segmentation from DCNN. The training sample is the digital images from the construction site. After the training is finished the testing images are processed into an SfM application to construct a point cloud. The classification is based on the mask images generated from the trained DCNN and camera poses solved from SfM. The final result is compared between samples and verify the performance of the new system and the new approach from two-dimensional real-world images.

Chapter 7 Conclusions:

This chapter concludes the entire research by giving a brief summary, contributions of the research, and limitations.

Chapter 2
Literature review

This chapter presents the literature that is related to the work in this dissertation. It is separated into four sections. The first section focuses on computer vision, which can be divided into other four subsections: 1) image features, 2) feature matching, 3) SfM, and 4) deep learning. The second section guides about infrastructure visualization. The third section explains the limitations of current photogrammetric works with infrastructure scope. In this section, it is further split into three subsections, which are the lack of fundamental evaluation, the lack of automated robust tools, and the explanation of the effect from unwanted features with photogrammetric techniques. The last section summarizes all the detail in this chapter.

2.1 Computer vision

2.1.1 Image features

Image features, in this aspect, are the important areas or pixels in an image that are selected by the feature detector algorithm based on the difference of intensity with their adjacent area (Li et al., 2015). These features can be detected as edges, corners, and blobs inside an image. Edges are lines constructed from the disjunction of pixel intensity. An example algorithm is the Canny edge detector (Canny, 1986), which utilizes an adaptive thresholding detector with hysteresis to eliminate the weaker edges than the lowest threshold. The result of the Canny edge detector is the contours constructed from the detected edges (Bradski and Kaehler, 2008).

Corners are formed by the points of intersection between two or more edges. Harris corner detector (Harris and Stephens, 1988) is one of the famous algorithms in this category. It also utilizes hysteresis procedures like a Canny edge detector. The output of this algorithm is five classes, namely the

background, two classes of corners, and two classes of edges. The algorithm deletes the exceeded and isolated edges and then, joins short and discontinuous edges together. The ends of these edges that form corners are shown as the result of the detection.

Finally, blobs or areas of interest in an image are the locations that contain many similar points. The algorithms that work on blob detection operate for both point and area detections. However, point detection relies on corner detection but finding the area of interest is instead a kind of image segmentation (Li et al., 2015). The representative point of each blob is chosen from the difference of intensity between samples inside that blobs. Examples of blob detection are scale-invariant feature transform (SIFT) (Lowe, 2004) and speeded up robust features (SURF) (Bay et al., 2008). SIFT utilizes the difference of gaussians (DoG), which is roughly gained from the approximation of laplacian of gaussian (LoG), to find a promising area that may contain points of interest. A DoG is the difference of Gaussian blurring of the same image but dissimilar sigma value finding the extreme point inside a set of neighbors over the image, both in the same scale and space (Bradski and Kaehler, 2008). This procedure is continuously processed in every octave of the Gaussian pyramid setting of the algorithm. When the rough areas are specified, they are then precisely sought through to find the feature inside. The extreme points inside each area that do not surpass the predefined edge and contrast thresholds will be removed. On the other hand, SURF implements Box Filter with LoG that can process rapidly and collaterally in many parts of an image. SURF also utilizes the determinant of the Hessian matrix for the location and scale of the detection, which is different from the sigma and thresholds of SIFT.

Image feature from digital images is a fundamental method in the computer vision domain (Apollonio et al., 2014). They can be used to track equipment and staff on site (Brilakis et al., 2011), template matching (Oron et al., 2018), and image stitching (Qu et al., 2015). They can then be further processed through numerous photogrammetric techniques, for example, Structure from Motion (Westoby et al., 2012) and Multi-View Stereo (Furukawa and Ponce, 2010), to construct a point cloud or model based on the matchings of these features solving for the intrinsic information of the camera.

2.1.2 Feature matching

Feature matching can be done by implementing a feature detector and descriptor algorithm, which is one of the key innovations in computer vision (Musialski et al., 2013). They were invented in the 1980s (Westoby et al., 2012). Feature matching is the first step of numerous photogrammetric techniques such as SfM and multi-view stereo (MVS) (Furukawa and Ponce, 2010). When processing a group of images into a photogrammetric application, the features inside each image can be firstly searched by feature detector algorithms. Then, when all of the features are detected, the algorithms called feature descriptors can match the same feature across images by searching through the unique vector of the intensity values around the feature of interest (Kaehler and Bradski, 2016), which is called image alignment. The output of image alignment is the intrinsic parameters of the image capturing tool and the positions in two- or

three-dimensional positions of each feature called image registration (Tareen and Saleem, 2018), which can be further utilized constructing the point cloud of the scene or object of interest. However, these vectors are very sensitive to any image distortions, for instance, rotation and zooming. Therefore, choosing the feature description that has robustness over these differentiations is the key to a better number of matchings and dispersing ratio over the region of interest (ROI), which will lead to the more complete outputs of photogrammetric techniques.

There are numerous feature detectors and descriptors, which have their own advantages and disadvantages, to be implemented. Among these feature detectors and descriptors, Scale Invariant Feature Transform (SIFT) (Lowe, 2004), Speeded Up Robust Features (SURF) (Bay et al., 2008), KAZE (Alcantarilla et al., 2012), Accelerated-KAZE (AKAZE) (Alcantarilla et al., 2013), Oriented FAST and Rotated BRIEF (ORB) (Rublee et al., 2011), and Binary Robust Invariant Scalable Keypoints (BRISK) (Leutenegger et al., 2012) have their own feature detectors and descriptors inside their algorithms. They are robust to image distortion such as rotation, affine transformation, and the zooming in or out between images (Tareen and Saleem, 2018). Moreover, they can be utilized easily with the built-in libraries inside OpenCV (Bradski, 2000). The detail of these algorithms is shown as follows.

2.1.2.1 SIFT

SIFT was first introduced by Lowe (2004). As mentioned in the previous subsection, it utilized the difference of Gaussian (DoG) pyramid and Hessian matrix to sort out the entire region of interest. The maximum value in the pyramid was chosen to be the prospective feature of that region. The DoG was the approximation values of Laplacian of Gaussian (LoG), which was able to decrease the computational resource. The descriptor inside split the surrounding of the feature of interest and formed into a 16x16 pixel with sub-blocks inside, using 128 bin to store the orientation. The disadvantage of SIFT was that it required a lot of computational costs.

2.1.2.2 SURF

SURF was developed by Bay et al. (2008). It deployed a box filter to calculate the determinant of the Hessian matrix. The descriptor of SURF utilized 64 bit which was able to be extended to 128 bits to cope with the affine distortion. Moreover, it required a lower computational resource than SIFT.

2.1.2.3 KAZE

KAZE was proposed by Alcantarilla et al. (2012). The main different point between KAZE with other older algorithms was the usage of non-linear Partial Differential Equations (PDE), instead of LoG and DoG. PDE was developed to replace Gaussian scale-spaces, which LoG and DoG relied on, due to the reason that Gaussian flattening blurred the image. Meanwhile, the non-linear diffusion filtering that KAZE used kept the sharpness of the input image. The maximum value inside the diffusion filtering was chosen to be the feature of the KAZE algorithm. For the descriptor, it investigated the strong

orientation around the feature of interest. Anyway, KAZE was very computation-exhausting, which the improved version of KAZE named AKAZE was later introduced in 2013.

2.1.2.4 AKAZE

Due to the resource hungriness of KAZE, AKAZE was invented to improve on this issue (Alcantarilla et al., 2013). The non-linear diffusion filtering in AKAZE was based on Fast Explicit Diffusion (FED), which worked more efficiently than the one in KAZE. The detector was upgraded from KAZE by using Scharr filters and the descriptor was based on Modified Local Difference Binary (MLDB).

2.1.2.5 ORB

ORB was presented by Rublee et al. (2011). It was a combination of the corner detection of Features from Accelerated Segment Test (FAST) (Rosten and Drummond, 2005) and the descriptor of Binary Robust Independent Elementary Features (BRIEF) (Calonder et al., 2010). FAST detector utilized Harris Corner to detect the feature in each region. Furthermore, the original descriptor of BRIEF was not good (Rublee et al., 2011), therefore it was modified to be utilized in ORB by rotating it as the name suggested.

2.1.2.6 BRISK

BRISK was developed by Leutenegger et al. (2012). BRISK implemented the AGAST algorithm (Mair et al., 2010) to detect the corner and then, found the features by utilizing the FAST score. The descriptor used the grey score around the feature point of interest to find the light direction. These values were collected into the binary to earn the true direction of the descriptor.

2.1.3 SfM

SfM was first mentioned around 1990 by a study in the computer vision expertise (Westoby et al., 2012). It has been proved that it is compatible with various type of photo and video capturing devices, for instance, cameras (Fathi and Brilakis, 2013), action cameras (Raoult et al., 2016), unmanned aerial vehicles (UAVs) (Carbonneau and Dietrich, 2017), and mobile phones (Wrózyński et al., 2017). The concept of the SfM technique was described by Westoby et al. (2012) as a low-cost photogrammetry operation to reconstruct a high-resolution model from the overlapping of various images. The difference between the SfM technique and the conventional method is that all of the parameters required for the calculation, for instance, the geometry of the view and camera locations, can be solved automatically by the matchings between features in the overlapping between images.

First, the images of the structure or scene of interest are processed to let the feature detector and descriptor algorithm detect the features inside these images. When these features are detected, they can be iteratively processed into non-linear least-squares minimization (Westoby et al., 2012) to achieve the fundamental matrix (Bartoli and Sturm, 2004).

The fundamental matrix is a 3 × 3, rank 2 matrix that shows the epipolar geometry gained from the projective mapping from feature points to their corresponding epipolar lines (Hartley et al., 2011; Jog et al., 2011). After the fundamental matrix is gained, it can be generalized for the essential matrix (Hartley et al., 2011) by implementing matrix multiplication between fundamental matrix, camera calibration matrix, and transposed camera calibration matrix. The essential matrix consists of five degrees of freedom (DoF), instead of seven from the fundamental matrix, which are rotation matrix, 3 DoF translation, and scale ambiguity. Once the essential matrix is achieved, the camera poses can be calculated. The camera pose consists of 6 DoF, which three of them are the rotation (roll, pitch, yaw) and others are the translation (X, Y, Z) in the real world environment (Nüchter and Hertzberg, 2008). The outcome of this step is four pose configurations that each solution yields a candidate position to reconstruct a point. The true reconstructed position can be verified by testing which position is in front of their respective cameras (Hartley et al., 2011).

a) b)

Figure 2.1 A pair of images with black epipolar lines show the motion between views, which image b) has a translation in the x-axis with some rotation from the image a

These true points are stored in a set of candidates that can appear in a three-dimension (3D) environment. New images are continuously added into this workflow to expand the size of the candidate set. Then, Perspective-n-Points or PnP problem (Hartley et al., 2011) is implemented to form the 3D environment of the reconstructed point cloud. Each pair of points that exists in at least three images are iteratively selected to be the output in a 3D environment (Westoby et al., 2012). However, these points cannot be directly utilized because PnP is very sensitive to outliers. Consequently, noise filtering algorithms such as random sample consensus (RANSAC) (Fischler and Bolles, 1981) should be implemented before further utilization (Hu et al., 2018).

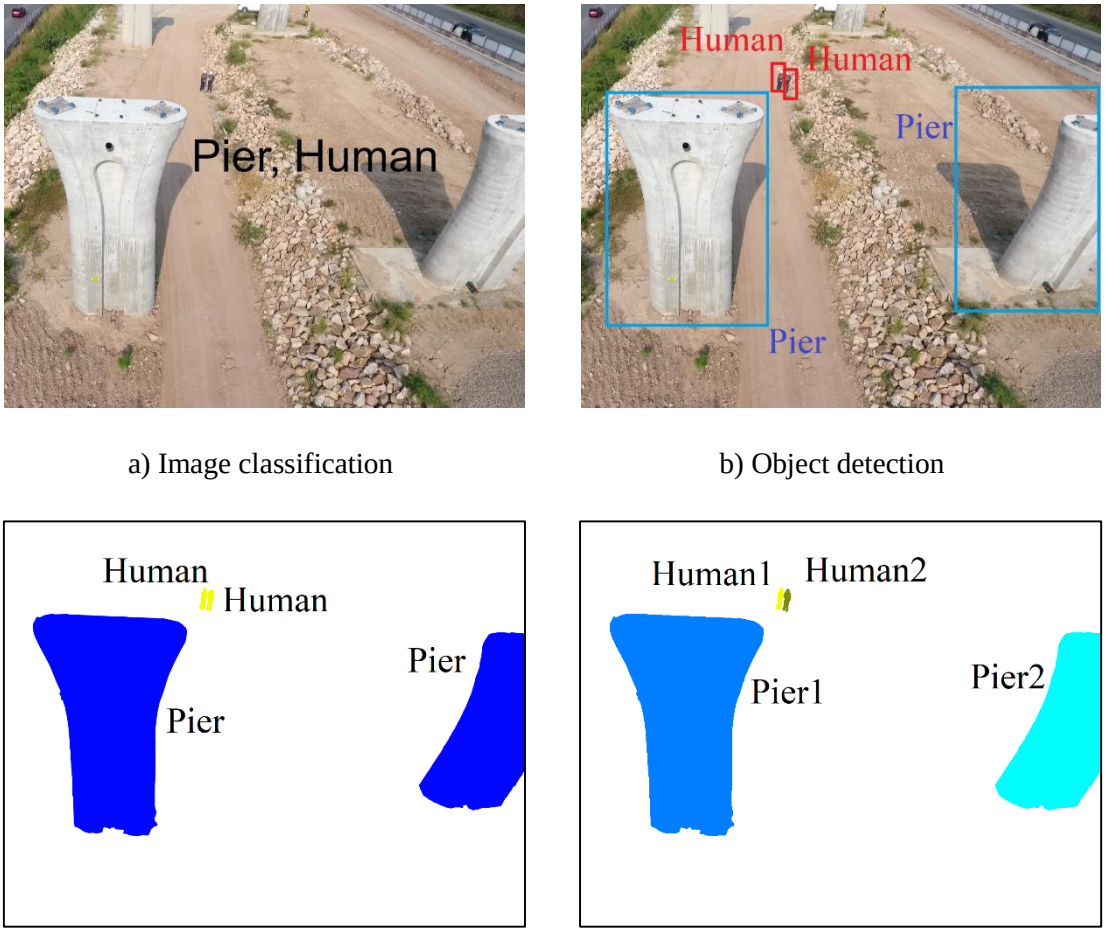
Although the outliers are mostly rejected by RANSAC in the previous process, minor noises from the iterative work process can accumulate and distort the output point cloud (Hu et al., 2018). In this case, camera poses and positions of points in the 3D environment from the previous process can be further corrected by implementing bundle adjustment (Triggs et al., 2010). Bundle adjustment is a non-linear optimization that tries to minimize the error from the projection by adjusting camera intrinsic parameters with the external parameters inside each image.

2.1.4 Deep learning

Deep learning is a subcategory of machine learning, which enables machines to comprehend information like humans. Deep learning applications can be constructed using various types of architecture, the most well-known is the convolutional neural network (CNN) (Cheng and Wang, 2018; Garcia-Garcia et al., 2018; Ma et al., 2019). CNNs utilize layers of units that arrange themselves in a manner similar to neurons in the human brain to understand the meaning of inputted information. In computer vision applications, CNNs generally perform well in terms of accuracy and efficiency (Garcia-Garcia et al., 2018). Cheng et al. (2018) described the iterative workflow of a CNN in computer vision as follows: 1) feature detection, 2) forward propagation of the detected features with other parameters to forecast which classes each pixel belongs to, 3) evaluation of the difference between the forecasted image with the ground truth, and 4) sending the error back to the neurons to adjust the parameters inside. A subcategory of CNNs called 'deep CNN' (DCNN) utilizes numerous layers inside the CNN to allow the algorithm to understand more information.

Garcia-Garcia et al. (2018) separated image segmentation, which is a crucial process in computer vision that helps the computer understand the information in images, into four categories. The first category is image classification, which allows the computer to understand what objects are in the image without specifying the locations of these objects. For further understanding, object localization, the second category, can be applied to let the computer draw bounding boxes over the objects of interest to show their locations. This level of technology can give the approximate locations of each object but is not sufficiently accurate to handle problems that require a more precise determination of location or area (Chen et al., 2019). The third category is semantic segmentation, in which the algorithm gives the exact pixels of each object class in the image. Ma et al. (2019) suggested that DCNN-based semantic segmentation was likely to become widely adopted due to its high accuracy. The fourth category is an upgraded version of semantic segmentation, called 'instance segmentation', in which the algorithm detects objects of each class as separate objects. **Figure 2.2** shows an example of four types of image segmentation.

Although generating readily usable data by applying deep learning and computer vision to civil infrastructure remains a challenge (Spencer et al., 2019), in recent years, deep learning has been applied to images related to civil engineering assist with the inspection and visualization tasks. One of the most useful applications of this deep learning is structural health inspections, research into which Spencer et al. (2019) separated into the heuristic feature-based and deep learning-based studies. Heuristic feature-based methods rely on thresholds or classifiers, such as edge detection (Spacek, 1986), region growing (Li et al., 2019), histogram shape-based threshold (Koch and Brilakis, 2011), and wavelet-based algorithm (Jahanshahi and Masri, 2013), to generate the output. However, the automation level of these



a) Image classification

b) Object detection

c) Semantic segmentation

d) Instance segmentation

Figure 2.2 Example of four types of image segmentation

techniques is low because they require manual or semi-manual tuning for each specific problem (Spencer et al., 2019). Thus, the utilization of deep learning-based methods continues to expand in the field of civil infrastructure because it is robust and can be applied to many different types of problems, especially semantic segmentation (Cheng and Wang, 2018; Li et al., 2019; Zhang et al., 2018). Examples of research in this field are as follows. Zhang et al. (2018) developed CrackNet and CrackNet II, which are tools for detecting cracks in the pavement. CrackNet II is a CNN with fully learnable layers. They trained their algorithm on 3,000 3D images, resized from a resolution of 4,096 × 2,048 pixels to 1,024 × 512 pixels to reduce the computational workload. CrackNet II achieved 90.2% precision and a recall rate of 89.06%. The main limitation of their work is that their algorithm cannot distinguish between pavement cracks and joint grooves. Cheng and Wang (2018) proposed the use of closed-circuit television images with a faster region-based CNN called the 'modified Zhang-Fergus network' (Zeiler and Fergus, 2014) to draw bounding boxes over defects, such as tree roots, cracks, infiltrations, and deposits in sewers. They trained their system on images with a resolution of 224 × 224 pixels. Due to the low number of available images, they implemented an image augmentation technique

to generate a training dataset of 3,000 images. They applied vertical and horizontal flipping, rotation, and color tuning with their images. Their system achieved a mean average precision of 0.83.

Images for training are not limited to photographs of the actual scene. Training images can be synthesized from a computer-aided design (CAD) model or captured from a miniature construction model. Kolar et al. (2018) developed a DCNN to detect guardrails in an image from VGG-16 architecture (Simonyan and Zisserman, 2015). They utilized synthesized images from a CAD model in different image capturing positions. Then, these images were placed over construction scenes and augmented by flipping, rotating, adjusting the color of the image. The total number of training images was 4,000 images, half of which were positive (the image included guardrails) and the other half was negative (the image did not include guardrails). Their system was able to detect guardrails at a mean precision of 96.0% and a recall rate of 98.2% for a single guardrail and a mean precision of 86.0% and a recall rate of 76.1% for multiple guardrails. False-positive images included pictures of scaffolds and reinforced steel bars. Izutsu et al. (2019) trained an object detection and semantic segmentation system using YoloV3 (Redmon et al., 2015) and U-Net (Ronneberger et al., 2015) architecture to recognize the steel frame components of a building under construction based on 2,000 images of a miniature construction model. The results showed that the system achieved an intersection over union (IoU) higher than 80% for the segmentation of the miniature construction model and just below 80% for the beam and column parts of the actual structure. However, the joint achieved an IoU of only around 50% due to the complexity of its shape.

2.2 Infrastructure visualization

Civil infrastructure such as highways and elevated railways are usually built using cyclic construction processes, for example, constructing numerous road sections and building structures from location to location. Civil infrastructure projects are expansive, which makes them challenging to monitor and inspect using traditional methods such as conducting site visits to assess construction progress (Behnam et al., 2016) and evaluate structural health (Tang et al., 2009).

Tools have been introduced to support the monitoring and inspection of civil infrastructure, including laser scanners, cameras, and drones. A laser scanner captures the geometry of a structure by emitting a laser and calculating the positions of the surfaces that reflected the laser back to the scanner to generate a point cloud (Tang et al., 2009). Laser scanners can generate hundreds of thousands of points per second with an accuracy ranging from a millimeter to a centimeter (Tang et al., 2010).

Although laser scanning can capture objects with high accuracy, there are numerous limitations to consider when selecting the data acquisition technique. First, the scanning strategy should be planned in advance to ensure that the structure is scanned completely and efficiently (Wang et al., 2018). Second, laser scanners may be prohibitively expensive (Bosché, 2010). Third, the accuracy of laser scans is

sometimes affected by the so-called ‘mixed-pixel effect’ (Tang et al., 2009), which results when the laser beam reflects off of two surfaces at different depths. This means that the sensor receives the reflected laser beam back from both surfaces at different times, which leads to a miscalculation of the distance. Fourth, laser scanners frequently require sensor calibration, which increases the time before they are ready to use (Golparvar-Fard et al., 2015). Finally, point clouds generated from laser scans of civil infrastructure require extremely large amounts of computational memory for processing (Hidaka et al., 2018).

Cameras and drones can also contribute to monitoring and inspection by capturing digital images and videos of structures, such as those used by the construction industry for decades (Brilakis et al., 2005). On-site workers often capture hundreds or thousands of images before the start of construction, depending on the scale of the project (Park et al., 2018). These images are used for documentation (Zhu and Brilakis, 2010), including progress reports (Golparvar-Fard et al., 2009) and structural inspection reports (Dawood et al., 2018). Although images captured using these tools have lower spatial accuracy than images captured using laser scanners, cameras and drones are easier to use and more affordable (Braun et al., 2015) and their images and videos can be processed using photogrammetric techniques such as SfM to produce point clouds similar to those generated by laser scanning.

2.3 Limitations of current photogrammetric works with infrastructure scope

2.3.1 Lack of fundamental evaluation

Digital images with feature detector and descriptor algorithms in civil engineering have been studied for a while now, for instance, rectangle concrete columns detection (Zhu and Brilakis, 2010), sparse point cloud generation using features from two calibrated cameras (Fathi and Brilakis, 2011), bricks calculation (Hui et al., 2014), absolute-scale point cloud generation of civil structures from a cube (Rashidi et al., 2015), an inspection of frame manufacturing from CAD images (Martinez et al., 2019), and crack detection in civil infrastructure using digital images (Kong and Li, 2019). However, the feature detector and descriptor algorithms have not been focused on as the main topic in the civil engineering scope. Studies often verified the performance of their system, not the performance of feature detectors and descriptors themselves.

Although there are numerous available feature detector and descriptor algorithms, each algorithm has its own advantages and disadvantages over separated expertise. In order to use a suitable feature detector and descriptor algorithms with new expertise, they should be tested before use (Rusinol et al., 2015). Işık and Özkan (2014) tested several algorithms on the Oxford dataset. They recommended the usage of ORB that was able to perform well with this image dataset. Hietanen et al. (2016) evaluated various feature detector and descriptor algorithms on Caltech (Fei-Fei et al., 2006) and ImageNet classes (Deng et al., 2009). They evaluated that the combination of a detector with SIFT descriptor was able to

outperform other algorithms. Tareen and Saleem (2018) appraised algorithms with numerous image datasets such as Oxford, MATLAB, and OpenCV. Their result showed that ORB was able to perform well, especially when the user-specified ORB to detect up to 1,000 feature points so that it could decrease the low confidence features and be able to match these features rapidly. In conclusion, ORB and SIFT have recommendations when the public image dataset is utilized as the sample of the implementation.

On the other hand, when the topic came to more specific expertise, their recommendations did not go along with the recommendation from the public image dataset domain. Mouats et al. (2018) argued that most of the studies about performance evaluations were mainly focused on clear and bright images. Therefore, they proposed the evaluation of feature detector and descriptor algorithms on night goggle images, which are blurry and dark. Their result demonstrated that SIFT had a very high performance with images with noises. They suggested that SIFT was very slow and might cause problems when the computational time is considered. Rusinol et al. (2015) evaluated the performance of these algorithms with thousands of document images and the result revealed that SIFT had extraordinary high performance within this domain. Its accuracy and computational speed were high. Moreover, BRISK and ORB also displayed promising performance having a slower decimal speed level than SIFT. Cowan et al. (2017) utilized videos from UAVs to let the users choose the area where they want the UAVs to land. Their result showed that BRISK was able to perform well with the videos which were taken from UAVs. Finally, Saha et al. (2018) assessed the performance of feature detector and descriptor algorithms with retinal images and found that SIFT and SURF were able to execute comparatively well.

It can be seen that there are different kinds of recommended algorithms depending on the topic of the dataset. Therefore, there is a need to evaluate the feature detector and descriptor algorithms before the actual utilization because a specific algorithm that can perform well with some expertises does not guarantee that it can execute well with other study topics (Rusinol et al., 2015).

2.3.2 Lack of automated robust tools

Three-dimensional point clouds are a fundamental source of data in various fields of study (Wang et al., 2019) for visualizing scenes and the existence of objects in which differences can be identified by comparing point clouds captured at different times. Numerous studies in civil engineering have implemented point clouds to assist with traditional work such as rock slope supervision (Weidner et al., 2019), equipment tracking (Fang et al., 2016), and powerline inspection (Jung et al., 2020). Nevertheless, these point clouds needed to be classified into classes before utilization (Pessoa et al., 2019; Wang, 2020). Point cloud classification or segmentation is a field that has attracted attention for many years but still requires further investigation (Wen et al., 2020). Numerous studies have tried to facilitate this cumbersome and time-consuming work by proposing automation based on three types of

methods, namely, point-based, object-based, and voxel-based (Kang and Yang, 2018; Politz et al., 2018).

Point-based classification considers each point separately by calculating the distance between the point of interest and its neighbor points (Li et al., 2019) using classifiers such as k nearest ratio (Altman, 1992) and perceptual organization (Lee and Schenk, 2002). The dispersion characteristics of the cloud points around the neighborhood of the point cloud of interest, for instance, the direction of vector normals and curvatures, are then used to represent this neighborhood along with the value from the classifier in the first step (Li et al., 2019). These values reveal traits of the cloud point called local features, and the classification is performed based on learning algorithms such as support vector machine (Boser et al., 1992) and random forests (Ho, 1995). Teo (2015) implemented Euclidean clustering to classify the point cloud from airborne wave light detection and ranging (LiDAR) using point distances and attributes to group cloud points that do not exceed some threshold, which was predefined by the user. The main drawbacks of this method are that the quality of the classification depends significantly on the initial value setting of the classifier and extensive calculation time is needed (Li et al., 2019).

The next type of classification is object-based techniques in which the classifier groups points into clusters and calculates the parameters such as vector normals and curvatures as a whole, which makes this process work nearly the same as how humans try to classify point clouds (Teo, 2015). Antonarakis et al. (2008) utilized object-based classification with the point clouds from LiDAR to separate forest and other types of land using the elevation surface and intensity of the cloud points. Their results showed that their algorithm was able to classify water and bare earth correctly, but differences between forest types complicated the classification. Politz et al. (2018) utilized object-based point cloud classification to specify the cloud points between water and road classes from a digital terrain model, which was the top view of a point cloud of a highway. They developed a convolutional neural network (CNN) to receive training data that were small patches of the point cloud model. Their results showed that their CNN correctly classified between water areas and roads at an accuracy of approximately 75%.

The third type of classification is voxel-based. A voxel is a representation of a three-dimensional grid that is divided from the input point cloud to simplify the input for further processing. Example values inside each voxel are voxel occupancy (Meng et al., 2019), visibility (Kim et al., 2019), color (Predescu and Triantafyllidis, 2018), and normal vectors that have both location and direction of the surface (Wang et al., 2019). Moreover, voxel-based subsampling can also be implemented to eliminate noise and outliers and also solve for differences in point cloud density (Xu et al., 2019), which encourages numerous studies to utilize this kind of classification in their research (Kang and Yang, 2018). Meng et al. (2019) proposed a new network for point cloud classification utilizing the occupancy of each voxel. The voxel values were then calculated for local parameters by interpolated variational autoencoder

combined with radial basis functions. The validating datasets were from public datasets, and the results were competitive with other novel point cloud segmentation algorithms. Kim et al. (2019) implemented the use of voxel grids containing visibility to optimize for the traveling route of a mobile laser scanning robot based on the point cloud of the scene taken from an unmanned aerial vehicle. The problem was defined as minimizing the occluded view for scanning. The results gave only about 3% of the incompleteness of the testing scene. Garcia-Garcia et al. (2016) presented PointNet which uses occupancy grids as the main concept of classification. They explained that occupancy grids are convenient for training due to the shape cues and array-like information. They specified the size of the occupancy grid as 300 in each dimension and 5 for each single voxel size. These occupancy grids were further processed through their proposed CNN that fully utilized point clouds as inputs and gained a 77.6% success rate with high classification speed.

However, numerous point cloud classification techniques implement handcrafted feature classifiers (Wen et al., 2020). Although the success of these algorithms relies on the excellence of the threshold setting, even a good setting may not be able to guarantee perfect classification (Garcia-Garcia et al., 2016). Automatic point cloud classification is a challenging task because point clouds usually have irregular point dispersion, noise, and overlapping (Merkurjev, 2020), which means that although there are some similar objects inside the point cloud, if their point densities and regularities are not the same, the algorithm classifying the point cloud might not be able to recognize as the same object (Zhao et al., 2018). The quality of classification by machine learning also heavily depends on the training data, and unless there is a huge amount of training data, the weight parameters inside the hidden layers of the network can be affected with the performance decreased by overfitting (Xu et al., 2020). Moreover, point clouds from laser scanners and cameras are greatly affected by the position of the capturing tools, with areas that are near or located perpendicular to the tool generating more points than those farther away or inclined objects (Hackel et al., 2017). This can be a serious issue if there are too few samples.

Due to these difficulties, some studies have transformed the input data from the point cloud directly into other formats (Hackel et al., 2017). For example, McClune et al. (2016) implemented edge detection to find the lines of roofs from orthophotos. They then utilized dense image matching to match the found edges with the point cloud constructed from these images. The cloud points inside the edges were considered to be the roof inside the point cloud. Their results achieved up to 85.5% completion for the flat roof, which was considered to be good when using (Hackel et al., 2017). Vu et al. (2012) proposed the use of dense image matching to support the local feature matching in point-based classification. They utilized two-dimensional Delaunay triangulation (Delaunay, 1934) to find feature points that have the shortest distance from the local feature of interest in a pair of images from the input data, which was the images from the actual scene. The triangulation was then formed based on the nearest distance features from each image and the point cloud of the scene. Furthermore, if there were any unused features from other images with a distance shorter than half of the epipolar band when the

triangle was projected, these features were also used. The output of this process was a set of used features, each in a tuple, that was further utilized as the visibility of each point. Sfikas et al. (2018) proposed the utilization of panoramic pictures from three-dimensional point clouds as an input for their CNN. The surface of each input point cloud was projected onto a cylinder shape that was unfolded into a panoramic picture and used for training a descriptor for further point cloud classification. The results showed that the method achieved competitive results with other networks that implemented a three-dimensional point cloud as an input.

It can be seen that although point cloud classification is output, the input does not have to be point clouds, which can ameliorate the problem of a lack of three-dimensional point cloud training data. However, the use of artificial data prepared from secondary sources such as computer-aided models usually makes the classification less general (Uy et al., 2019). For instance, the classification network has to deal with data from actual images that are often aligned in the vertical axis, but the synthetic data contains various irregular orientations (Fu et al., 2020). Moreover, the photogrammetric point cloud of a real-world scene usually has numerous unrelated cloud points such as noise and unwanted features, particularly for challenging scenes from civil infrastructure sites that often contain a large number of rocks, boulders, and vegetation that have more features than the smooth surface of the civil structure. This situation can complicate feature detection algorithms or handcrafted classifiers because there are only a small number of cloud points belonging to the object of interest, and statistical outlier removal could accidentally remove them (Saovana et al., 2020a).

2.3.3 Effect of unwanted features with photogrammetric techniques

Unfortunately, the output of the SfM process usually contains numerous outliers (Ning et al., 2018) that dramatically lower the accuracy of the 3D reconstruction (Balta et al., 2018). Removing this noisy information can lower the computational cost and increase the accuracy of the work process (Balta et al., 2018). Rusu (2007) proposed the use of semantic segmentation of indoor environments using a statistical technique from Zhang (1994) to remove unwanted features from the input point cloud before segmentation. Balta et al. (2018) developed a system for grouping the point cloud into clusters and calculating the mean number of points inside the cluster. Each cluster with fewer than the average number of points was then removed by the system. However, these methods decrease unwanted features or noise only from the scene as a whole which can be problematic because ROIs in the point cloud with fewer than the average number of points will be removed by the system. Ning et al. (2018) separated outliers into isolated and non-isolated outlier groups. They utilized the local density of the point cloud to statistically justify whether a group of outliers was isolated. Then, they used local plane fitting to transform the non-isolated outliers, which were very close to the ROI point cloud, into the same plane as the ROI point cloud. Unfortunately, when non-isolated outliers are very dense, the ROI can deform and outliers will not be eliminated, as was the case in Balta et al. (2018). Jahanshahi et al. (2017)

proposed using an adaptive resection-intersection bundle adjustment algorithm to iteratively reduce outliers from 3D feature positions inside a scene. They set the probability of a feature inside the aerial image input being an outlier at 5% and 10% in some testing cases, in which a feature with a reprojection error exceeding the threshold was treated as an outlier. This approach has the same limitation as the system of Balta et al. (2018); that is, if the ROI does not cover the entirety of the images, it may decrease the features present in the ROI. Jung et al. (2019) suggested removing unwanted features in the point cloud of a road by separating the axes inside the point cloud. Because of the two-dimensional nature of roads running along the ground, they were able to separate the point cloud of the road from the point cloud of other three-dimensional objects. However, they could not remove the grass features because they share similar characteristics to road markings, which have long thin lines and high contrast with the road surface (Jung et al., 2019). It should be noted that this method cannot be applied to three-dimensional infrastructures such as piers and bridges because their structures contain heights, which is different from the nature of roads that lies on a horizontal plane.

Moreover, large areas of rough terrain are usually difficult to reconstruct as point clouds (Balta et al., 2018) due to the large number of features that can cause unwanted cloud points such as boulders and vegetation. Therefore, other than the noise and outliers that have to be removed, it is necessary to remove as many unwanted features as possible from outdoor settings in images of civil infrastructures, in which in this aspect, there are several presented noise and outlier removal techniques. However, current methods usually tackle only the noise and outliers as a whole with statistical techniques for the outdoor environment that usually contains a gigantic amount of cloud points. These techniques treat the cloud points that do not align with the majority of the scene point cloud as noise and outliers and remove them without the consideration of whether those removed cloud points are the ROI of the scene. This is a serious issue when applying the current techniques with the outdoor scene of infrastructure sites where are rich in features from surrounding but the ROI such as the concrete and steel piers have fewer features because of their smooth and textureless surfaces (Saovana et al., 2019b). Thus, the removal of these unwanted features such as noise, vegetation, and extraneous ground surfaces from the infrastructure scenes is still manually removed during the current work process (Lu et al., 2019). Furthermore, current noise and unwanted-feature filterings of photogrammetric techniques mainly focus on dealing with the constructed point clouds although removing these undesirable features at the input process can raise the number of matchings inside the ROI and greatly decrease the computational time of the entire process (Saovana et al., 2019b).

Semantic and instance segmentations have the potential to be applied to repetitive infrastructure projects (Hidaka et al., 2018) because they can highlight the ROI inside the scene likes human interpretation although they are just a small part of the images. Nevertheless, the segmentation to filter outliers from scenes is normally utilized with the indoor environment such as the detection of mugs, plates, and boxes (Rusu et al., 2007) and planar structures such as walls and floors (Czerniawski et al.,

2018). On the other hand, the utilization of segmentation with outdoor scenes is still limited to roads and their assets (Garcia-Garcia et al., 2018; Golparvar-Fard et al., 2015). Therefore, in this study, the use of segmentation techniques for removing unwanted features in images of infrastructure was also investigated.

There are three overview advantages of removing unwanted features according to Carrivick et al. (2016): 1) reducing file size, 2) decreasing computational time, and 3) improving the quality of the ROI output. The previous studies also showed that removing the unwanted features out of the input images before implementing feature matching can raise the number of matching inside the ROI of the images (Saovana et al., 2019a, 2019b). However, less matching of the overall data can complicate the camera position (Rahal et al., 2018) and affect the overall matching quality because good matches are not sufficient for controlling bad matches (Lin et al., 2018). Therefore, a quantitative evaluation of the SfM model is necessary to understand the effect of removing unwanted features on the SfM technique.

2.4 Summary

Computer vision has been implemented with infrastructure visualization for a while now. The current approach that is widely used is to utilize the feature and point cloud from feature detector and descriptor algorithms and photogrammetric techniques such as SfM to compare differences over time. However, although various studies have investigated this topic, the fundamental aspects such as the validation to find the most suitable feature detector and descriptor algorithms still remain unknown. Moreover, before the utilization of these features and point clouds, there is a need that these points must be classified to be more intuitive and useful, in which the classification of these points is troublesome, labor-intensive, and time-consuming. Most of the current classification techniques rely on the handcraft classifier that their performances require on the knowledge and experience of the practitioners who set the parameters, which make these classifiers fragile and not robust to other samples.

One of the solutions to this problem is to use deep learning to classify these points. Deep learning is a subset of machine learning that allows the computer to understand the problem and solve it with the pre-trained knowledge inside the network layers. Nevertheless, the bottleneck that obstructs the implementation of deep learning is the availability of the training data such as feature formation and point cloud in two- and three-dimensional environments, respectively, which is very rare for infrastructure scope to have these data available. Thus, a new and robust method to support the classification of these points for infrastructure is needed.

The final presented difficulty is the effect of unwanted features on the feature matching and output point cloud from the photogrammetric method. In theory, these unwanted features complicate the matching of features and need to be eliminated before processing into a photogrammetric application but there are no quantitative clarifications for this solution. On the other hand, removing unwanted features raises

the concern that it can also provide complexity to the matching algorithm due to the lower number of features of the scene. Consequently, the validation of unwanted features on feature matching and point cloud generation must also be investigated.

Chapter 3

The validation of feature detector and descriptor algorithms

As mentioned in the previous chapters, the current practice in site monitoring utilizes vision-based techniques to recognize the existence of structures and other facilities (Braun et al., 2018). Vision-based techniques can be mainly grouped into two types, which are the image recognition from digital images or videos and the point cloud from photogrammetric procedures. By implementing these techniques, they have to rely on feature detectors to automatically detect meaningful pixels and feature descriptors to match these pixels between several images to solve for the intrinsic parameters of the image capturing tool. Consequently, the quality of these vision-based techniques significantly depends on feature detector and descriptor algorithms (Saovana et al., 2019b). However, the studies on this aspect with the infrastructure-related components have not yet to be seen.

Thus, in this chapter, an evaluation between feature detector and descriptor algorithms is conducted. First, the method used to validate is explained. Then, the validation result based on the samples, which are infrastructure piers and scaffolds, is demonstrated. Finally, the result is discussed and concluded. The algorithms that can provide satisfactory results will be utilized for the next chapters.

3.1 Proposed method of performance evaluation of feature detector and descriptor algorithms for infrastructure site components

The evaluation is separated into five parts, which are shown in **Figure 3.1**. Infrastructure digital images are utilized as the input of this study. These images are classified based on their original capturing tools, put into a classified dataset, and processed through an image processing software to be manually

cropped. These cropped images are also formed other datasets that have no background to visualize the impact of the background. Moreover, the rim of these cropped images will be served as the ROI of the testing. Next, the system implementing feature detector and descriptor algorithms is developed to extract and match features between each image pair. For the ROI coverage testing, every image from datasets that have the background is processed into the system to mark the feature inside each image. These images are further processed into an image processing software to calculate the number of pixels that each feature detector and descriptor can detect. After the number of pixels inside the entire picture is achieved, the ROI is utilized to find the number of pixels inside the ROI and calculate the coverage that each algorithm can detect. Then, every dataset is processed into the system again to do the matching between each image pair. In this part, each algorithm detects and highlights the features inside each image. Afterward, each feature inside the first image is utilized as a reference to find other similar features inside the second image. If a similar feature is found, the system will match these two features by drawing a yellow line between these two features. The evaluation is separated into two criteria, which are the performance of the matching and the computational time in each step. These criteria are two important factors for image registration. The higher number of good matching shows a better ability to find the similarity between images. Meanwhile, the computational time shows the rapidness of the system, which will be necessary when the speed of the image registration is concerned. Finally, the results are compared, discussed, and concluded.

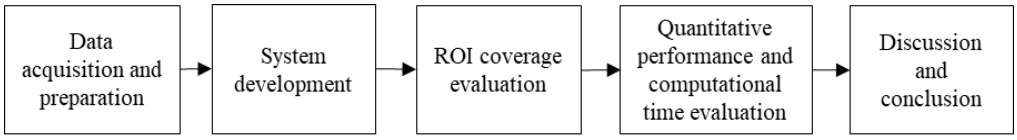


Figure 3.1 Research procedure

3.2 Experimentation based on infrastructure site images

Eight digital image pairs from a real infrastructure project in Thailand were captured by a drone and a camera. This case study was chosen because it contained numerous types of smooth surface reinforced concrete structures and had a disordered environment. The drone used in this study was DJI Phantom 4 and the camera was Olympus EM-10 Mk III. **Figure 3.2** shows the sample images from each image pair. Then, these images were manually cropped to extract the interested structures and utilized as the ROI for further evaluations. **Figure 3.3** shows the example of the original image and the cropped image that its rim will be utilized as the boundary of the ROI coverage evaluation. Consequently, there were four datasets which were images from a camera, images from a camera that have no background, images from a drone, and images from a drone that have no background. Each dataset consisted of eight images (four image pairs).

Next, both ROI and normal images were processed through a system programmed on Spyder with OpenCV 3.4.2 to detect the feature points inside these pictures. The specifications of the testing computer were Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz and 8.00 GB RAM. Feature detector and descriptor algorithms that were evaluated in this research were SIFT, SURF (64-Floats), SURF128 (Extended SURF that used 128-Floats), KAZE, AKAZE, ORB, and BRISK. The output of the system was black images that the positions of feature points were marked in red color. These images were processed through Photoshop CS2 to find the number of red pixels and calculate the percentage of ROI coverage (red pixels inside the yellow line). **Figure 3.4a** shows the output image from the system. The yellow line in **Figure 3.4b** was manually marked to show the boundary of the ROI. It used the rim of **Figure 3.3b** as the reference.

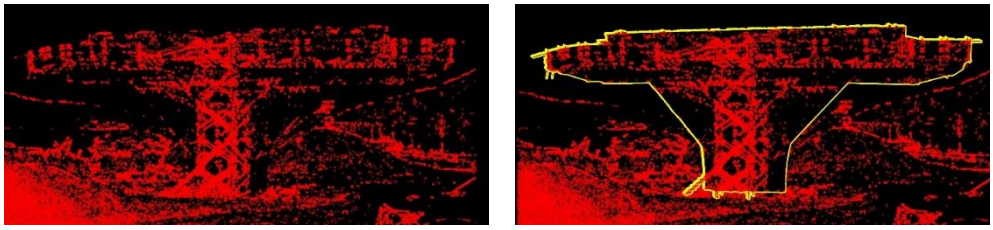
a) Pictures from a camera b) Pictures from a drone

Figure 3.2 Sample pictures inside each dataset

a) Original image b) Cropped image

Figure 3.3 Example of the original image and the cropped structure inside no background dataset

The next part was the evaluation of the feature matching by each feature detector and descriptor algorithm. The matching scheme between features was Nearest Neighbor Distance Ratio, which utilized the first two nearest neighbors from a feature set of the first image as the references and then, searched



a) Output image from the system b) Boundary of the ROI

Figure 3.4 Output image after processed **Figure 3.3a** through the system and the boundary of the ROI

through similar features from another picture. The threshold ratio of this matching scheme was set at 0.7 to eliminate the low confidence matches. For SIFT, SURF, SURF128, and KAZE descriptors matching, Least Absolute Deviations or L1-norm was utilized for the image registration. Meanwhile, Hamming distance (Robinson, 2008) was used for AKAZE, ORB, and BRISK descriptors matching.

When the matching was ongoing, it had a high probability to form bad matchings that will affect the quality of the image registration. Therefore, these bad matchings had to be removed by using a RANdom Sample Consensus (RANSAC) algorithm (Fischler and Bolles, 1981). The iteration of RANSAC was set at 2,000 iterations and 99.5% confidence like Tareen and Saleem (2018). In order to raise the confidence of the time measurement of each feature detector and descriptor algorithm, the entire matching was executed 100 times for each algorithm and each image pair to minimize the risk of computational errors.

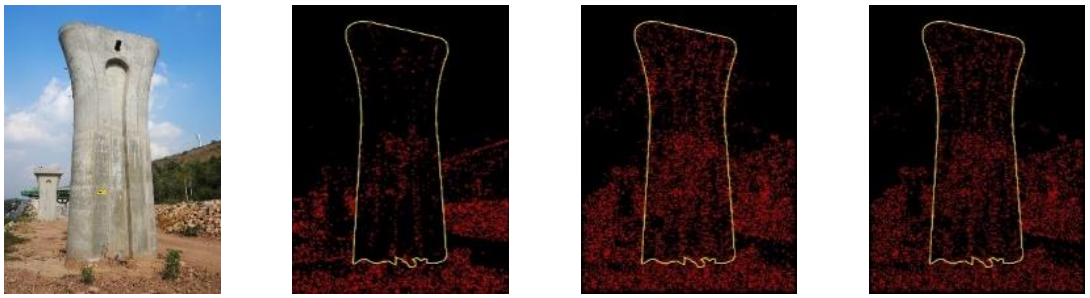
3.3 Results of the validation between feature detector and descriptor algorithms

In this section, it is separated into two subsections based on the validation, which are the comparison of the detection coverage and the performance evaluation of each algorithm.

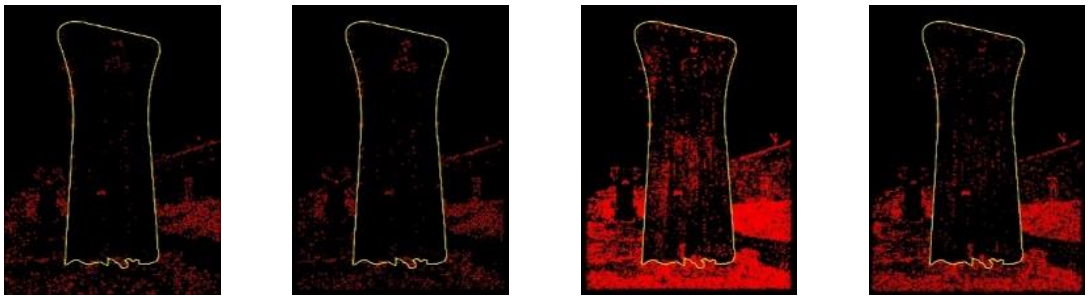
3.3.1 Comparison of the detection coverage

Firstly, sixteen images from camera and drone datasets were processed into the system. Each sample was evaluated separately between each feature detector and descriptor algorithm. **Figure 3.5a)** shows the sample image from the camera dataset **Figure 3.5b)** to **Figure 3.5h)** shows the output from each feature detector and descriptor algorithm. Red dots were features that each algorithm detected and the yellow lines were manually drawn to show the ROI boundary of the image. **Table 3.1** shows the result of the example in **Figure 3.5**. The calculation started by finding the number of pixels in the original image. In this case, the original image was 1080 × 1440 pixels, which equaled to 1,555,200 pixels. Then, this amount was utilized to divide the total detected features of each algorithm to find the

percentage of the coverage for the entire image. Next, the numbers of red dots inside the ROI were calculated to be divided by the number of pixels inside the ROI, which was 484,526 pixels in this case.



a) Original image b) SIFT c) SURF d) SURF128



e) KAZE f) AKAZE g) ORB h) BRISK

Figure 3.5 Sample coverage results of this study

Table 3.1 Result from the evaluation of **Figure 3.5**

Feature detector and descriptor algorithms	Total detected features (Pixels)	Percentage of detected features compared to the entire image (%)	Total detected features inside ROI (Pixels)	ROI Coverage (%)
SIFT	136,933	8.805	29,225	6.032
SURF	258,399	16.615	**125,206**	**25.841**
SURF128	204,238	13.133	91,274	18.838
KAZE	73,593	4.732	15,102	3.117
AKAZE	63,047	4.054	15,807	3.262
ORB	**340,587**	**21.900**	78,718	16.246
BRISK	210,614	13.543	25,140	5.189

From **Table 3.1**, it could be seen that ORB can detect the highest amount of features inside an image. However, when focusing inside the ROI of the image, SURF could do better with the coverage of about 25% of the entire ROI. Furthermore, all of the results from every image pair were used to calculate the average percentage of the coverage. The final average result of this part is shown in **Table 3.2**. The final result went along with the example in Table 1 and demonstrated that ORB had the highest performance in detecting features inside an image. Anyway, SURF had the highest percentage of the features detection inside ROI.

Table 3.2 Comparison of the average percentages of detected features and ROI coverage

Feature detector and descriptor algorithms	Average percentages of detected features compared to the entire picture (%)	Average ROI coverages by detected features (%)
SIFT	11.552	7.877
SURF	18.864	**23.802**
SURF128	15.130	17.218
KAZE	6.955	6.946
AKAZE	6.672	7.034
ORB	**30.882**	20.976
BRISK	19.964	12.020

3.3.2 Performance evaluation of each feature detector and descriptor algorithm

In this part, all of the image pair, sixteen in total, were processed into the system to find the matching performance of each feature detector and descriptor algorithm. **Figure 3.6** shows the example results of an image pair from the camera dataset and **Table 3.3** shows the summary of the performance evaluation between feature detector and descriptor algorithms. Please be noted that µ stands for Micro (10^{-6}).

The result showed that ORB could detect the highest number of features in every case. The number of matched features in the datasets that contained fewer details such as no background datasets were very competitive between ORB and SURF algorithms, which on average, ORB could perform slightly better than SURF. However, the number of matching from the ORB algorithm decreased drastically after RANSAC removed the outlier matching. SURF had the highest amount of inliers instead of ORB. For the time spent in each step, although ORB could detect the highest amount of features in each image, ORB still spent the least period of time in cases that had less detail such as the datasets that had no background. For the datasets that had a background, SURF128 could perform faster. Furthermore, AKAZE spent the least amount of time on average for features matching and outliers rejecting. Finally, SURF128 utilized the least amount of time on average for the entire process.

3.4 Discussion

In general, ORB was a very promising algorithm for detecting features inside infrastructure images. It could detect the biggest amount of features and still utilized the shortest period of time in order to detect them in cases that had less detail like no background datasets. BRISK and SURF were the other two algorithms that could detect the desirable amount of features inside images. ORB had a great performance in feature matching but SURF also had high performance, especially in the dataset that

a) Features and matchings by SIFT

b) Features and matchings by SURF

c) Features and matchings by SURF128

d) Features and matchings by KAZE

e) Features and matchings by AKAZE

f) Features and matchings by ORB

g) Features and matchings by BRISK

Figure 3.6 The sample matching results of this study

had less detail like the camera dataset that had no background. Moreover, although ORB had a considerably high amount of matching, more than half of these matchings were rejected by RANSAC as poor matchings. The huge amount of time that ORB used to do the matching reflected on this aspect also because the higher amount of detected features meant the higher possible matchings. However, in this case, these matchings could be wrong and waste the time instead. These matchings also did not cover the ROI of the images as expected because it could detect features up to about 30% of the image but could detect only about 20% of the ROI. SURF had the highest amount of ROI coverage which might result in a better quality model of ROI. Although it could detect around 18% of the image, it could detect the ROI up to about 24%.

Table 3.3 Summary of the performance evaluation between feature detector and descriptor algorithms

Feature detector and descriptor algorithms	Detected features (Points)		Matched features (Pairs)	Matched inliers (Pairs)	Time for features detected (µSecond)		Time for features matched (µSec.)	Time for outliers to be rejected (µSec.)	Total time (µSec.)
	1st image	2nd image			1st image	2nd image			
Camera dataset with no background									
SIFT	1,522	1,470	190	121	12.431	10.265	0.971	0.112	23.779
SURF	4,130	3,839	**396**	**236**	7.941	6.652	2.262	0.106	16.961
SURF128	2,716	2,551	312	202	3.457	2.830	1.034	0.102	7.423
KAZE	1,167	1,306	103	50	61.432	50.729	0.428	0.109	112.697
AKAZE	1,205	1,244	77	36	10.936	8.763	**0.389**	**0.088**	20.176
ORB	**7,007**	**7,115**	308	162	**1.525**	**1.288**	4.341	0.100	**7.254**
BRISK	2,142	2,221	104	61	3.390	3.153	0.928	0.095	7.566
Camera dataset									
SIFT	10,974	7,876	248	119	12.178	11.691	23.406	0.084	47.359
SURF	14,626	13,141	480	**250**	10.787	10.095	23.046	0.098	44.025
SURF128	10,749	9,510	387	214	**4.963**	4.640	12.450	0.093	22.147
KAZE	5,474	5,110	102	42	54.821	54.658	3.975	0.088	113.542
AKAZE	4,821	4,117	76	24	9.813	9.651	**2.273**	**0.076**	**21.813**
ORB	**63,308**	**47,596**	**499**	114	5.029	**4.010**	144.843	0.089	153.971
BRISK	20,619	15,610	141	64	9.404	7.684	31.587	0.078	48.753
Drone dataset with no background									
SIFT	1,200	1,200	108	66	8.057	8.011	0.766	0.070	16.904
SURF	3,079	2,988	290	**178**	5.245	5.246	1.680	0.075	12.245
SURF128	2,108	2,054	215	144	2.488	2.341	0.861	0.073	**5.763**
KAZE	1,190	1,144	136	63	36.585	36.155	0.425	0.075	73.240
AKAZE	1,101	1,063	77	42	6.854	6.587	**0.285**	**0.061**	13.786
ORB	**9,192**	**8,880**	**318**	172	**1.191**	**1.160**	7.811	0.076	10.239
BRISK	3,171	3,036	130	83	3.116	3.042	2.098	0.068	8.325
Drone dataset									
SIFT	11,196	11,879	418	146	9.790	9.912	26.515	0.076	46.293
SURF	17,220	16,884	529	202	10.122	10.043	29.247	0.083	49.495
SURF128	12,778	12,487	413	159	**4.701**	**4.615**	17.482	0.076	26.875
KAZE	7,455	7,575	457	190	39.119	38.208	5.873	0.083	83.284
AKAZE	6,397	6,626	186	67	7.632	7.450	**3.289**	**0.061**	**18.432**
ORB	**69,771**	**70,982**	**805**	**222**	5.344	5.254	187.449	0.081	198.128
BRISK	28,510	27,932	459	202	11.249	10.870	61.542	0.080	83.740
Average of all image pairs from all dataset									
SIFT	6,223	5,606	241	113	10.614	9.970	12.914	0.086	33.584
SURF	9,764	9,213	424	**216**	8.524	8.009	14.059	0.090	30.682
SURF128	7,088	6,651	332	180	3.902	3.607	7.957	0.086	**15.552**
KAZE	3,821	3,784	199	86	47.989	44.937	2.675	0.089	95.691
AKAZE	3,381	3,263	104	42	8.809	8.113	**1.559**	**0.071**	18.552
ORB	**37,319**	**33,643**	**483**	167	**3.272**	**2.928**	86.111	0.087	92.398
BRISK	13,610	12,200	208	103	6.790	6.187	24.039	0.080	37.096

AKAZE could match features rapidly. It spent the smallest amount of time finishing the matching and the outliers rejecting. Besides, although the speed was very fast compared to other algorithms, its

matching number was very poor and had the smallest amount of final matchings. Finally, SURF128 was the overall fastest algorithm when it had to detect and match features inside unmodified images of infrastructures. The ROI coverage was considered satisfactory with the third rank behind SURF and ORB.

3.5 Conclusion of this chapter

This chapter presents a performance evaluation between seven feature detector and descriptor algorithms, which are SIFT, SURF, SURF128, KAZE, AKAZE, ORB, and BRISK. They have both a detector and a descriptor available in their algorithms. They are also robust to the rotation, the affine transformation, and the zooming of images. The evaluation used unmodified infrastructure digital images as testing datasets. This study can serve as a suggestion for choosing the feature detector and descriptor algorithms with the infrastructure domain. The results showed that ORB had the highest performance in detecting features. It also spent the least amount of time in finding these features. Anyway, if the image had a gigantic amount of features, it would slow down this process and lose to SURF128. SURF had a competitive amount of detected features but the dispersion of detected features inside the ROI was better than ORB and had the most promising result in ROI coverage testing. AKAZE was the fastest algorithm to finish the matching and outliers rejecting. However, AKAZE performed poorly when it came to the quantity of feature matching. Interestingly, SURF128 spent the least amount of time on the overall process. Moreover, it could also detect the features inside the ROI very well and got the third rank in the ROI coverage testing.

The limitation of this validation was the number of the sample, which was small (sixteen image pairs, thirty-two images in total). Therefore, more images should be implemented to solidify the result.

Chapter 4
Effect evaluation of the unwanted features on feature matching

In the previous chapter, it can be seen that there were only a few features that were detected by the feature detector algorithm. This is one of the problems from utilizing feature detector and descriptor algorithms with unmodified infrastructure images. These images usually contain numerous features from surrounding such as boulders, vegetation, and terrain, resulting in the ROI becomes the minority of the image and decrease the probability that these features of interest to be matched and used in the further steps of the photogrammetric techniques without getting removed. By removing these unwanted features out before implementing feature matching may relieve this problem but also leads to the lack of matching due to the loss of matchable features. This vague situation needs to be clarified before the unwanted feature removal can be implemented with the infrastructure images to support photogrammetric work processes.

4.1 Proposed evaluation method

The proposed method was separated into five parts, which were shown in **Figure 4.1**. The first process was data acquisition and preparation. Infrastructure digital images that were taken from a camera were utilized as the input of the system. The camera which was used in this study is Olympus EM-10 Mk III. A total of one hundred images, for instance, roads, bridges, dams, tunnels, and waterways, were formed into fifty image pairs for image matching. These images were also processed into an image editing application to be manually removed the unwanted feature to form the ROI of each image. Therefore, there were fifty image pairs for both original and modified datasets, which had no unwanted features.

Moreover, the rims of the images that their backgrounds were removed were used as the boundary of the ROI for the evaluation in the further stage. All of fifty image pairs from both sets can be found in https://drive.google.com/drive/u/1/folders/14OZO6dQ3mIYLCjBQHRNCsJeA81x7yLZE.

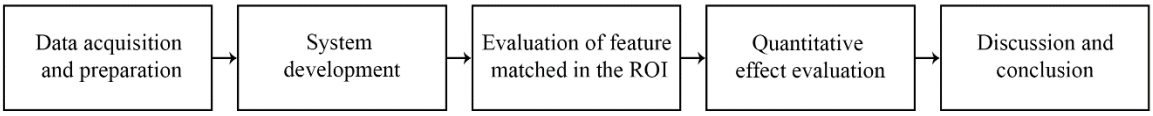
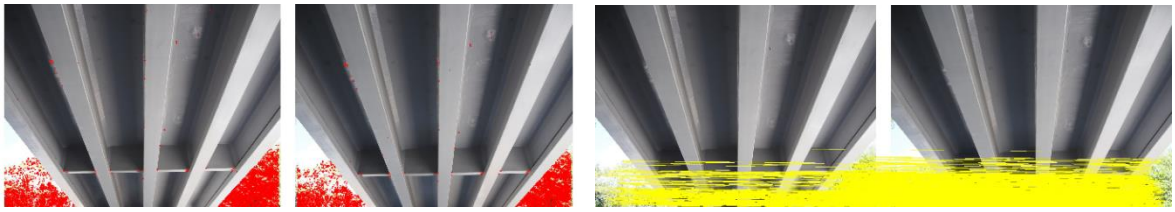


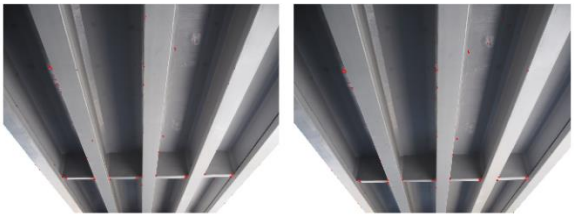
Figure 4.1 Overview of the evaluation of the effect of unwanted feature removal with feature matching

The system for evaluation was developed using Spyder with OpenCV 3.4.2 to detect features and do the matching. The specification of the programming computer was Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz and 8.00 GB RAM. Feature detector and descriptor algorithms that were tested in this study were SURF (64-Floats), SURF128 (Extended SURF, which used 128-Floats), ORB, and BRISK. These algorithms were chosen because they had a high performance with the infrastructure domain in the previous study (Saovana et al., 2019a). Moreover, they also had both feature detectors and feature descriptors that could be easily utilized in OpenCV. The output of the system were the images with red dots showing the detected features in each input image and yellow lines showing the matching between two input images. **Figure 4.2** shows the detected features and matching results from one of the image pairs, both with and without unwanted features.

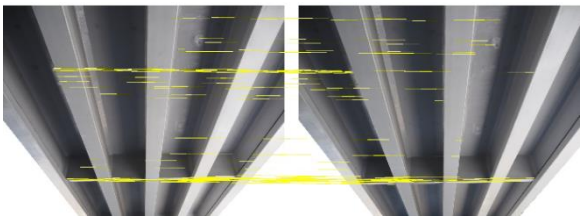


a) Detected features of the unmodified image pair

b) Matching of the unmodified image pair



c) Detected features of the image pair that unwanted features were removed



d) Matchings of the image pair which unwanted features were removed

Figure 4.2 Samples of the detected features and matchings by ORB

Then, these images with red dots were processed through Photoshop CS2 to calculate the number of features matched inside the ROI. In this process, the boundary for the ROI was obtained from the rim of the corresponding image without unwanted features. For example, **Figure 4.3** shows images of a

pedestrian crossing bridge which is the 4th sample in the dataset. **Figure 4.3a** is the unmodified images. **Figure 4.3b** is **Figure 4.3a** that was removed the unwanted features. **Figure 4.3c** shows the boundary that used the rim of the middle pair for the ROI coverage evaluation. This boundary was utilized to calculate the number of features matched in the ROI for the unmodified images. Next, the numbers of features matched inside the ROI of each original image pair were compared with their corresponding image pair that unwanted features were removed. The result of the comparing was the different percentage between the number of matches before and after unwanted features were removed. However, if there are no matchings inside the ROI before the unwanted features removal but instead have matchings after that, the different percentage will be infinity because the divisor is zero. Therefore, samples that fell into this case must be removed from the averaged data and presented in the total number of the case instead. After that, the average total number of features and matchings with their computational time in each step were processed and compared between each algorithm to verify the effect between modified and unwanted features removed images. The data was further analyzed to find the different percentages of each algorithm and concluded.

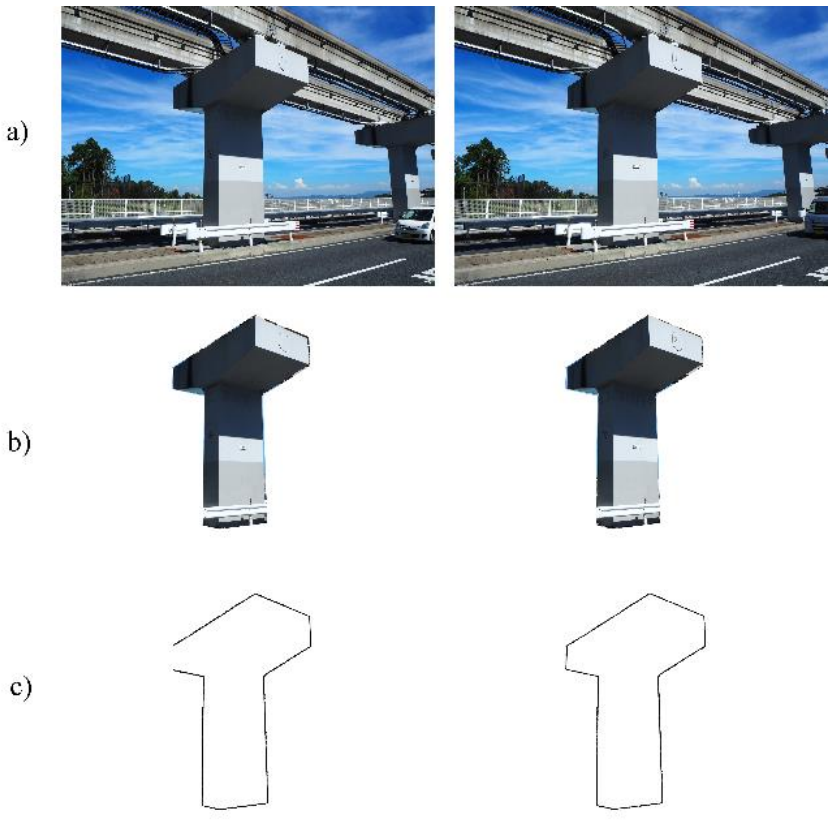


Figure 4.3 Example of a sample image pair from the dataset, a) Original image pair, b) Unwanted feature removal pair, and c) Boundary for the ROI evaluation

4.2 Results of the effect of unwanted features

The results of the study were separated into two parts, which are the comparison between the number of matching inside ROI and the quantitative effect of the unwanted features removal.

4.2.1 Number of matching inside ROI

The result of this part was shown in Table 1. However, please be noted this table only shows the final result of this procedure. The entire result of each sample and each algorithm can be found in the appendix of this study in the same link with all of the image samples in chapter 3. In addition, abbreviations in Table 1 were utilized in the table in the appendix. A total of fifty image pairs of various infrastructure digital images were processed through the system with different feature detector and descriptor algorithms. The result showed that unwanted features removal could support every feature detector and descriptor algorithm in order to match the features inside the ROI. They were able to detect all of the ROI in every sample. Moreover, ORB gained the biggest benefit from unwanted features removal because it could match 22.6% more features inside the ROI. Meanwhile, other algorithms in the study detected about 7 to 12% more features inside the ROI and the overall average was 13.18%.

Table 4.1 Evaluation of the effect of unwanted features removal

Feature detector and descriptor algorithms	Average percentage of increasing matchings inside the ROI after unwanted features removal (%)
BRISK (B)	10.82
ORB (O)	22.60
SURF128 (S128)	7.36
SURF (S)	11.94
Average	13.18

4.2.2 Evaluation of the effect on performance

In this part, the result was separated into two sub-topics, which were the detected features and matchings and computational time.

4.2.2.1 Detected features and matchings

The result in this part was separated into three tables as shown in **Table 4.2** to **Table 4.4**. **Table 4.2** shows the number of detected features and the matchings of the dataset that had only unmodified images. Meanwhile, **Table 4.3** shows the number of detected features and the matchings of the dataset that had unwanted features removed. Finally, **Table 4.4** shows the difference percentage between the data before and after the unwanted features removal. It can be seen that by removing the unwanted features, the features that could be detected was decreased by about 68% (69.16% for the first images

in each image pair and 67.58% for the second images in each image pair). Moreover, the matching was also decreased by 75.36% and 77.65% for both before and after outlier elimination by RANSAC, respectively.

Table 4.2 Detected features and matchings from the original image dataset

Feature detector and descriptor algorithms	Detected features of the 1st image (points)	Detected features of the 2nd image (points)	Matched features (pairs)	Matched inliers (pairs)
SURF	20,882	20,425	3,369	2,721
SURF128	15,147	14,889	2,405	1,940
ORB	80,014	78,263	7,140	5,845
BRISK	28,603	27,779	3,182	2,595

Table 4.3 Detected features and matchings from the unwanted features removed image dataset

Feature detector and descriptor algorithms	Detected features of the 1st image (points)	Detected features of the 2nd image (points)	Matched features (pairs)	Matched inliers (pairs)
SURF	7,523	7,607	970	700
SURF128	5,264	5,358	678	500
ORB	21,898	22,457	1,429	1,048
BRISK	7,215	7,709	686	518

Table 4.4 Different percentage of detected features and matchings between two datasets

Feature detector and descriptor algorithms	Detected features of the 1st image (%)	Detected features of the 2nd image (%)	Matched features (%)	Matched inliers (%)
SURF	63.97	62.75	71.21	74.28
SURF128	65.25	64.01	71.81	74.24
ORB	72.63	71.31	79.99	82.06
BRISK	74.77	72.25	78.43	80.03
Average	69.16	67.58	75.36	77.65

4.2.2.2 Computational time

Subsequently, the computational time that was used according to the work process in the previous sub-topic was demonstrated in three tables from **Table 4.5** to **Table 4.7**. **Table 4.5** shows the computational time that was utilized in each step, namely features detection of both images, feature matching, outliers rejection, and time in total, for the dataset that contained only original images. Next, **Table 4.6** shows the computational time of the same topics as **Table 4.5** but with the dataset that unwanted features were removed. Lastly, **Table 4.7** shows the different percentages of the time used to process between both datasets. From the result tables, the biggest amount of time that could be decreased was the time for features matching. It was diminished by about 82% if the unwanted features removal was implemented.

For the total amount of time spent for the entire process, it was decreased to the average of about 72% which ORB and BRISK were the top two algorithms that can gain the most benefit from unwanted features removal. The time used for algorithms to detect features was also reduced for all of the algorithms. The average of the decreased time was about 46% for the entire image dataset (46.98% for the first images in each pair and 45.70% for the second images in each pair). Finally, the time that RANSAC spent to eliminate the outliers was also reduced by 55.38%.

Table 4.5 Computational time in each process of the original image dataset

Feature detector and descriptor algorithms	Time for detected features of the 1st image (µSecond)	Time for detected features of the 2nd image (µSecond)	Time for features matching (µSecond)	Time for outliers to be rejected (µSecond)	Total time (µSecond)
SURF	13.389	13.178	48.882	0.342	75.791
SURF128	6.024	5.947	26.645	0.267	38.883
ORB	6.093	5.955	315.989	0.634	328.671
BRISK	12.194	11.839	83.873	0.327	108.233

Table 4.6 Computational time in each process of the unwanted features removed image dataset

Feature detector and descriptor algorithms	Time for detected features of the 1st image (µSecond)	Time for detected features of the 2nd image (µSecond)	Time for features matching (µSecond)	Time for outliers to be rejected (µSecond)	Total time (µSecond)
SURF	8.941	8.829	9.871	0.165	27.807
SURF128	3.876	3.937	5.268	0.142	13.223
ORB	2.426	2.434	54.370	0.203	59.432
BRISK	5.016	5.105	13.566	0.148	23.836

Table 4.7 Different percentage of the computational time in each process between two datasets

Feature detector and descriptor algorithms	Time for detected features of the 1st image (%)	Time for detected features of the 2nd image (%)	Time for features matching (%)	Time for outliers to be rejected (%)	Total time (%)
SURF	33.22	33.00	79.81	51.74	63.31
SURF128	35.65	33.81	80.23	46.95	65.99
ORB	60.18	59.12	82.79	68.05	81.92
BRISK	58.86	56.88	83.83	54.76	77.98
Average	46.98	45.70	81.66	55.38	72.30

4.3 Discussion on unwanted feature removal with feature matching

The evaluation of the effect from unwanted features removal was tested with one hundred infrastructure digital images (fifty image pairs) in two datasets, original and unwanted features removed. The result showed that by removing unwanted features, it was able to help the feature detector and descriptor

algorithms to focus more on the ROI and hence, had more matching inside the ROI. The reason behind this situation was the algorithms had lower confidence to match the features inside the ROI so, it instead matched the features that it had more confidence. Although the distance between image taking positions is only a centimeter away, this problem can still occur. **Figure 4.4** shows the samples that an algorithm was not able to match features inside the ROI greatly when using the original images (The matching of 32nd sample (Monorail pier) and 40th sample by BRISK (Bridge girder)). The algorithm was able to detect some features inside the ROI (The red pixels inside **Figure 4.4c** and **Figure 4.4d**). However, it matched the uninterested features such as buildings, rails, and trees instead of the ROI (**Figure 4.4e** and **Figure 4.4f**). By removing unwanted features, it assisted these algorithms to shift their matchings into the ROI directly (**Figure 4.4g** and **Figure 4.4h**) thus, the following process such as 3D triangulation could gain benefits from these matchings inside the ROI. Based on **Table 4.1**, the average percentage of the increasing matching inside the ROI was 13.18%. Furthermore, ORB gained the biggest benefit of this method. It was able to match 22.60% more matchings inside the ROI after the unwanted features removal.

For the quantitative performance evaluation, the features that each algorithm could detect were about 68% decreased. The matchings were also decreased to an average of 75.36% and 77.65% for both before and after RANSAC implementation, respectively. These results might affect the overall matching because the number of good matches may be too few to suppress the bad matchings (Lin et al., 2018). Therefore, it should be further analyzed to know the quality of the output model whether it is affected.

Anyway, removing the unwanted features out of the input images had a great advantage for the computational time. The required time for features detection was about 46% decreased or nearly half of the original images procession time. Moreover, the matching time was also 81.66% reduced and due to the less amount of matchings, the outliers rejection by RANSAC also spent 55.38% less amount of time. Finally, the total average time for the entire process was 72.30% decreased.

4.4 Conclusion of this chapter

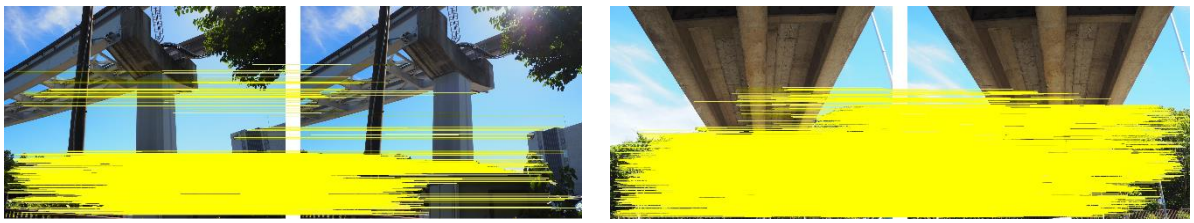
This chapter presents a quantitative performance evaluation of the unwanted features removal with four feature detector and descriptor algorithms, which were SURF, SURF128, ORB, and BRISK. These four algorithms were chosen because they performed well with the infrastructure digital images based on the previous chapter. There were also easy to be utilized and robust to image distortions. The evaluation was conducted on various infrastructure digital images such as roads, bridges, and dams. The results of this study showed that although unwanted features removal decreased about 68% and 75% of detected features and matchings, respectively, it was able to increase the matchings inside the ROI by 13.18% and reduce the total computational time by 72.30%. Moreover, ORB gained the biggest benefit from

a) Detected features from the original

b) Detected features from the original image

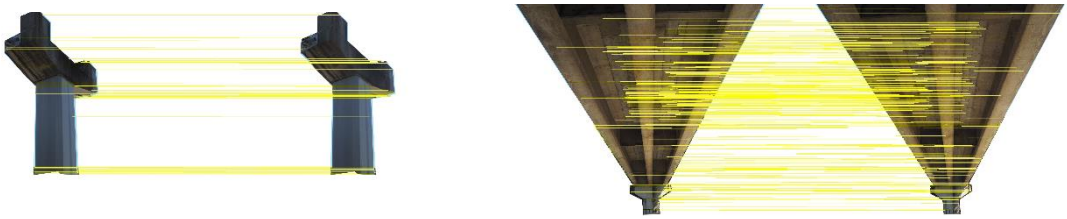
c) Detected features from the image pair that unwanted features were removed

d) Detected features from the image pair that unwanted features were removed



e) Matchings from the original image

f) Matchings from the original image pair



g) Matchings from the image pair that unwanted features were removed

h) Matchings from the image pair that unwanted features were removed

Figure 4.4 Sample matching results that their matching performance of the ROI were better after unwanted features removal

unwanted features removal by having 22.60% more matching inside the ROI and 81.92% less computational time. This study can be implemented as proof that by removing the unwanted features out of the input images, it can increase the matchings inside the ROI and reduce the computational time in order to process these images.

Chapter 5
Unwanted feature removal with point clouds

From the previous chapter, the number of feature matching can be raised by removing unwanted features before the matching. By gaining the bigger number of feature matching, it may refer to the higher number of point cloud that can be registered. However, two aspects need to be further clarified. First, if the number of matching can be increased, can the number of point cloud be quantitatively raised? Second, based on previous chapters, the unwanted feature removal is currently implemented manually, which is time-consuming and labor-intensive. There must be a novel method to support this process of work. One of the possible solutions is to implement deep learning with this scope of work but no studies have conducted the unwanted feature removal with deep learning yet. Thus, in this chapter, the prospective solution of using semantic segmentation with unwanted feature removal is investigated.

5.1 Proposed methodology of unwanted feature removal with point clouds

5.1.1 Overview

In this research, a new method using a DCNN to automatically segment and remove unwanted features from input images for the SfM process is proposed. The proposed methodology is shown in **Figure 5.1**. First, images of the target structures are taken. These images are manually labeled and processed by the developed system utilizing the DCNN, which automatically segments the target structures based on the trained knowledge. The system then removes other features from the images based on the specifications input by the user. Next, to assess the performance of the proposed system, three testing sets are created: the original images, the images generated by the proposed system, and the manually modified images.

The three test sets are processed separately by the SfM application between each testing set in each sample. Finally, the results of each testing set are compared to evaluate the performance of the proposed system.

Figure 5.1 Proposed methodology

5.1.2 Data acquisition

The images used as input for the proposed methodology can be captured using various tools, including cameras, drones, and mobile phones. However, these images should be taken from numerous angles due to the limitations of the SfM technique. Furthermore, the overlap between the images is necessary to raise the quality of the generated point cloud by allowing the feature detection and description algorithm to run during the feature matching process.

5.1.3 System development

To replace the labor-intensive and time-consuming manual removal of unwanted features, two main automated processes were proposed: the use of a DCNN and simple pixel replacement. The main objective of the DCNN is to identify which pixels belong to the classes specified as unwanted by the user. These pixels are removed by replacing the unwanted pixel with a white pixel (255 RGB). These processes are detailed in the following sections.

5.1.3.1 Proposed DCNN

The development of the DCNN can be separated into three steps, which are network composition, network training, and network evaluation, as shown in **Figure 5.2**. First, a DCNN model was selected for use in the proposed system. Then, the DCNN was trained on the image dataset from the target structure to understand the scene. Finally, the results were evaluated to verify the performance of the network.

Figure 5.2 Development procedure of the proposed system

The composition of the network in this methodology can be separated into two parts: an encoder and a decoder. The main task of the encoder is to discover the features of the ROI in the input image by

utilizing convolutional layers with the rectified linear units (ReLUs) activation function and max-pooling layers. The use of convolutional layers with ReLUs is beneficial because they can converge onto the correct trend in the training data up to six times faster than the same network using the tanh function (Krizhevsky et al., 2017). The max-pooling layers decrease the size of the multi-channel feature map by searching for and returning the maximum values inside each neighborhood of the feature map to protect the translation of the detected feature in the feature map (Goodfellow et al., 2016).

The main tasks of the decoder are to increase the size of the feature map and determine the position of each object class. The decoder consists of convolutional layers with ReLUs, up-convolutional layers, and 1 × 1 convolutional layers. Upsampling is performed by the up-convolutional layers, which double the size of the feature maps at the beginning and reduce the number of channels by half. The 1 × 1 convolutional layer can be implemented at the final step to adjust the number of channels inside the final layer to match the number of classes in the training set (Ronneberger et al., 2015).

Normally, the encoder and decoder sections function as separate processes. However, copy-and-pasting the feature maps from the encoder to the decoder can increase the performance of the network, especially at the borders of each convolutional layer (Amirkolaee and Arefi, 2019; Ronneberger et al., 2015). Therefore, it is advantageous to implement a copy-and-paste procedure between the encoder and decoder sections.

Numerous studies (Golparvar-Fard et al., 2015; Goodfellow et al., 2016; Li et al., 2020) have indicated that having more layers in the network architecture results in higher quality and a more generalized outcome. Thus, it is useful to observe the training results to check whether the network is performing well (Krizhevsky et al., 2017). Many evaluations can benchmark the performance of the system, including precision rate, recall rate, F1-score, and IoU.

Training data is crucial to the performance of a DCNN. It must relate to the scope of the research objective. However, the designs in civil infrastructure projects vary from project to project. Thus, it can be challenging to find a suitable public dataset that can be used for the universal detection of infrastructure piers. Some studies (Bosché, 2010; Kolar et al., 2018) have proposed synthesizing artificial data, such as CAD models, for machine learning detection. However, in the real world, the ROI can be obstructed by objects such as buildings, vegetation, and vehicles, which can make it difficult for the DCNN to make an accurate prediction. To train the DCNN to overcome these challenges, images from real-world scenes were utilized as the training images.

Moreover, although a DCNN requires a large number of training images for the network to converge onto the correct trend for prediction and achieve satisfactory results, it is very labor-intensive to capture thousands of images from the actual site. By rotating, flipping, and adjusting the brightness of the input images, the total number of training images can be increased, thereby increasing the rate of understanding and the accuracy of the results (Amirkolaee and Arefi, 2019).

After the training is complete, cross-validation is conducted. Cross-validation is a method for evaluating the quality of a network to determine whether it is generalized and can be utilized with other datasets (German et al., 2012) without bias toward the data from the testing set (Ibtehaz and Rahman, 2020). One of the most commonly used cross-validation methods is 10-fold cross-validation (McLachlan et al., 2004), in which the training data are randomly separated into 10 parts. Each part is continuously implemented as an evaluation set for the other 9 parts as the training set. The performance of each training is averaged to determine the performance of the network using the training set. At the same time, the standard deviation is calculated to determine the generalization of the network over the dataset (German et al., 2012).

5.1.3.2 Automated unwanted-feature removal

The methodology of automated unwanted-feature removal is shown in **Figure 5.3**. First, the unwanted classes and the input images are processed into the system. Second, the input images are segmented by the DCNN based on the trained dataset. Finally, the DCNN removes pixels belonging to the unwanted classes from the images.

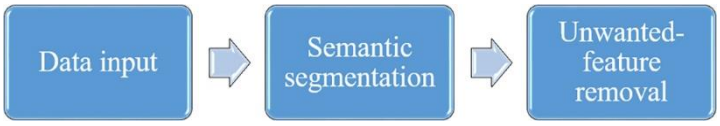


Figure 5.3 Methodology of the proposed automated unwanted-feature removal system

However, in cases where there is insufficient computational memory to process the original images due to their large size, pixel replacement can be implemented using reduced-size images. The proposed work process of unwanted-feature removal from reduced-sized images is shown in **Figure 5.4**.

For example, if the original image (image I) has a resolution of 4,000 × 3,000 pixels, it will cost a gigantic amount of computational memory and might cause an error with the computation. Therefore, its size can be reduced by the DCNN and used as the input for semantic segmentation. After the semantic segmentation is complete, the resolution of the output is reduced to 512 × 384 pixels (image S). Then, it is restored to a resolution of 4,000 × 3,000 pixels (image E) without passing through the network. However, enlarging the smaller image causes the image to become blurry, which if it is directly utilized as an input for the SfM process, can affect the quality of the SfM output (Liu et al., 2016). Therefore, in order to preserve the detail for the further SfM process, the image used for the unwanted-feature removal process is primarily constructed from a copy of the original image (image U) modified using image E as a reference. Given the formula for the pixel value as

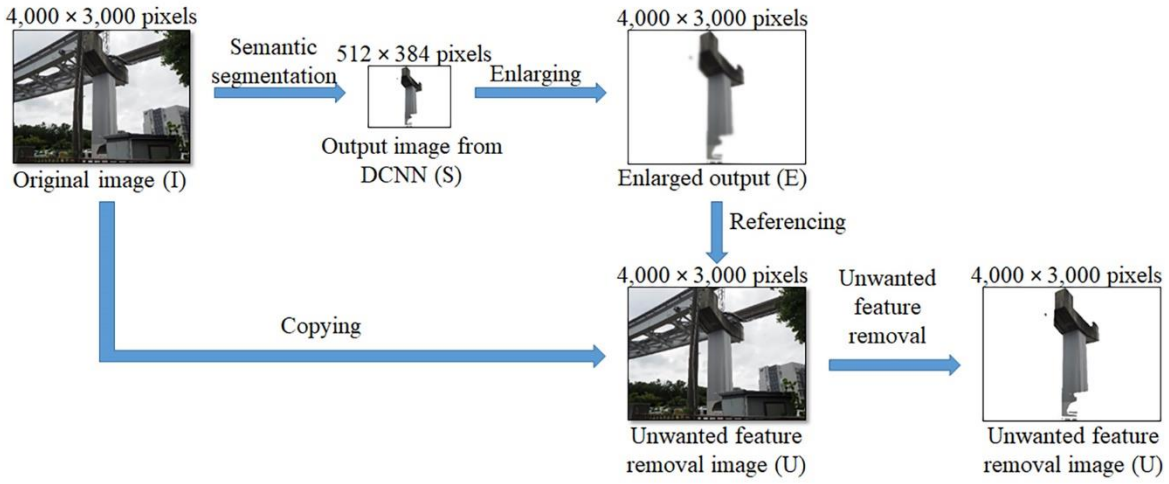


Figure 5.4 Proposed process of unwanted-feature removal from reduced-sized images

$$p_e(x, y) = (r, g, b), \tag{5.1}$$

where p_e is the pixel at the coordinate (x, y) of the image E for any integer $x \in [0..w]$, and integer $y \in [0..h]$, where w is the width of the original image and h is the height of the original image, r, g, and b represent the value of red, green, and blue channels of pixel p_e, respectively. The value of the pixel inside image U can be described as

$$p_u(x, y) = p_e(x, y), if\ p_e(x, y) = (255, 255, 255), \tag{5.2}$$

where p_u is the pixel at the coordinate (x, y) of image U, which has the same size as the original image. The pixel value of (255, 255, 255) means that pixel is white. It is worth mentioning that if the ROI in the original image already contains many white pixels, the value of the removal can be changed from (255, 255, 255) to any value the user specifies. The unwanted-feature removal is complete when $p_u(0, 0)$ to $p_u(w, h)$ have been completely processed.

5.1.4 Testing set creation

To benchmark the quality of the SfM model from the proposed system, three testing sets were created from the unmodified images, the images modified by the system, and the images modified based on the ground truth of the system. These testing sets were created separately for each sample and processed into an SfM application to construct the output model of each testing set.

5.1.5 SfM process

The methodology of the SfM process used in this study is shown in **Figure 5.5**. The process starts by uploading the images from each testing set to the SfM application to perform image alignment and complete image registration. The result from this step is the determination of the features and camera positions in three-dimensional space that will serve as the initial parameters for dense cloud

construction. The construction of the dense cloud generates the point cloud based on the depth map from these camera positions. However, the point cloud generated from these processes usually contains a lot of noise and outliers. Therefore, threshold filtering is used to eliminate low-confidence cloud points from the output before the evaluation.

The threshold of each cloud point can be calculated by comparing the number of matching that formed the respective cloud point with the numbers of matching from other points in the same scene. Higher confidence of a cloud point shows that the cloud point of interest is constructed from a huge amount of matchings from various image pairs compared to the number of matching from other points in the same scene (Agisoft LLC, 2019).

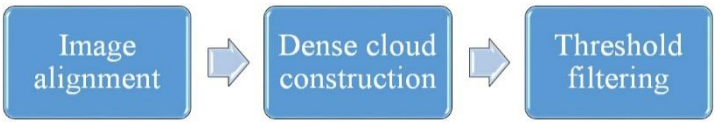


Figure 5.5 Methodology of the SfM process

5.1.6 Result comparison

Normally, the number of features, the cloud points, and computational time are benchmarked between each sample (Anil et al., 2013; Jung et al., 2016; Rashidi et al., 2013; Saovana et al., 2019a). However, comparing between alignable images to obtain more in-depth information for judging the performance of unwanted-feature removal by the DCNN using the SfM process was also proposed.

The number of input images plays a vital role in the quality of the output point cloud when using the SfM technique because the more images there are, the more overlap there will be. This greater overlap helps the feature detection and description algorithm match the same features from different images more easily. Thus, the SfM application can efficiently align these images and produce a higher-quality point cloud. Concerns were expressed by Lin et al. (Lin et al., 2018) and Rahal et al.(Rahal et al., 2018) that removing some features from the input images might affect the overall SfM process due to reduced or possibly erroneous feature matching. To investigate this issue, the evaluation of the number of alignable images is also implemented as part of this study.

In addition, file size is considered to be one of the main difficulties in processing point clouds (Gao et al., 2015). Large file sizes reduce the ability to manage and transfer these files. Thus, it is important to verify this aspect as well.

5.2 Experimentation

5.2.1 Data acquisition

Images used for input were collected from real-world structures. The equipment used in this study was a camera and a drone, the specifications of which are shown in **Table 5.1**. The drone was used to capture images of a concrete motorway pier in Nakhon Ratchasima Province, Thailand, and the camera was used to capture images of a composite monorail pier in Suita, Osaka Prefecture. The piers were treated as the ROIs of each sample.

Table 5.1 Specifications of image capturing tools

Specification	Drone	Camera
Name	DJI Phantom 4	Olympus EM-10 Mk III
Image type	PNG	JPG
Image size (pixels)	1,920 × 1,080	4,608 × 3,456

5.2.2 Development of the unwanted-feature removal system

5.2.2.1 Network composition

The foundation of the proposed system relied on U-Net (Ronneberger et al., 2015), which was selected because its architecture contains deep multiple layers of encoder and decoder parts with a copy-and-paste implementation that can achieve high-speed detection with satisfactory accuracy in many areas (Garcia-Garcia et al., 2018; Izutsu et al., 2019; Ronneberger et al., 2015). However, any network with a similar architecture and functionality could also have been used instead.

The network workflow is as follows: receive the input image, attempt to understand the objects in the image, and formulate a prediction based on this understanding. U-Net has convolutional and pooling layers that decrease the size of the image and increase the number of channels. U-Net uses a max-pooling algorithm to determine the maximum values within each rectangular region (Goodfellow et al., 2016). Feature maps are copied and cropped at every other convolutional layer to the counterpart side, which gives the architecture of the network a "U" shape. By applying this method, the loss of information at the border of the image from stride 2 of 2 × 2 max-pooling can be solved by downsampling. Then, after 10 convolutional layers, the downsized image is upsized and the number of channels is decreased. The input image has to pass through the network for a total of 23 convolutional layers until the number of channel equals the number of assigned classes by a 1 × 1 convolutional layer or a deconvolutional layer (Garcia-Garcia et al., 2018). Ronneberger et al. (2015) suggested using an input image size that 2 × 2 max-pooling can be applied through the image evenly in both horizontal and vertical axes. **Figure 5.6** shows the U-Net architecture when it detects 2 classes in an image, in this case, piers and background.

5.2.2.2 Network training

The motorway pier and monorail pier datasets were trained separately because doing so yielded better accuracy compared with training them together (Amirkolaee and Arefi, 2019). **Figure 5.7** shows sample images from both datasets. Each image was manually labeled into 5 classes (piers, roads, vegetation, terrain, and vehicles) using LabelMe (Russell et al., 2008). Unlabeled pixels were automatically classified as background. **Figure 5.8** shows the sample manual labeled images as the ground truth for training from both datasets.

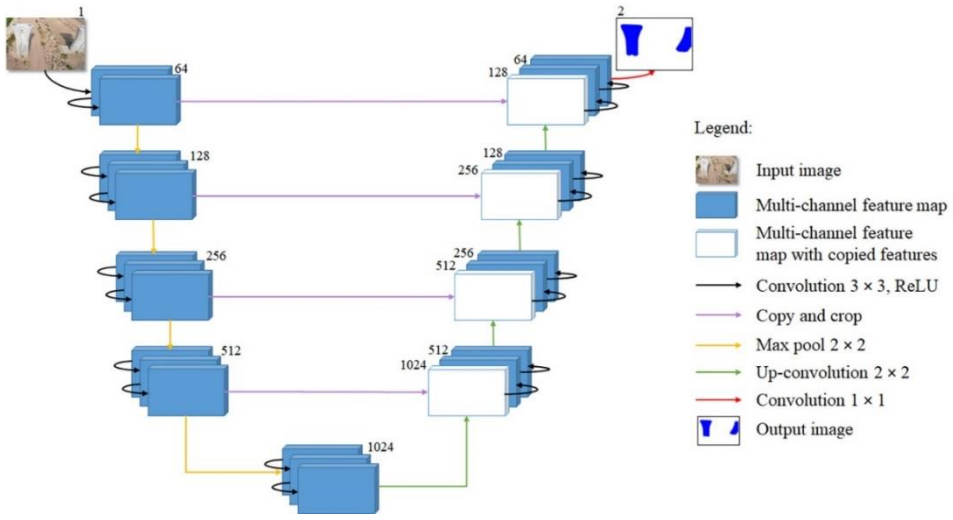


Figure 5.6 U-Net architecture when processing 2 classes of objects
The numbers at the corner of each feature map are the number of channels

a) Sample image from the motorway dataset b) Sample image from the monorail dataset

Figure 5.7 Sample images from the infrastructure datasets

The labeled images in each dataset were clustered into training and testing sets. However, these numbers were very small compared with public datasets, which may have contained thousands of images, so data augmentation was implemented to increase the number of training images using horizontal flipping, up to 10° angle rotation, and random brightness adjustment. Vertical flipping was not applied because it does not reflect reality, such as the sky being at the bottom and the road being at the top of the picture. In addition, the images were downsized to a maximum of 512 pixels to ensure that there was enough

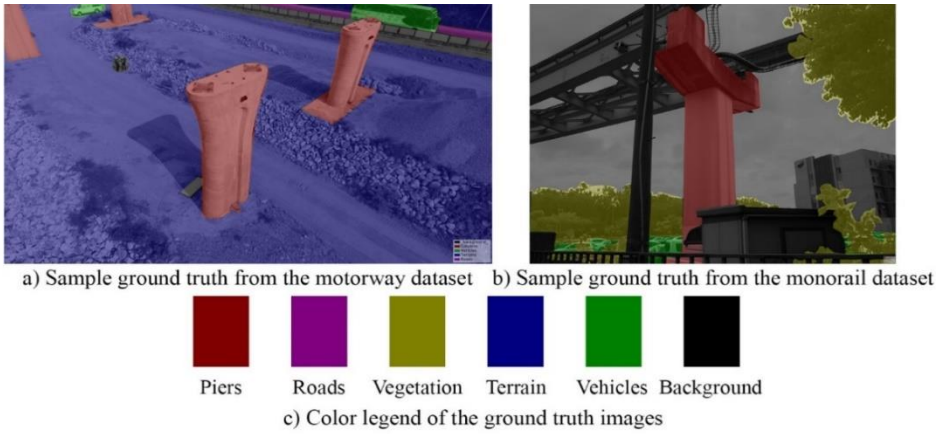


Figure 5.8 Sample manual labeled images as ground truth for training from both datasets

memory for the training procedure. Details on the training and testing sets from the motorway and monorail datasets are shown in **Table 5.2**. Training was performed on a computer with an Intel® Core™ i7-8086K CPU @ 4.00GHz with 32.0 GB of RAM and an NVIDIA GeForce RTX 2080 Ti graphics card. The model was implemented using an open-source deep learning platform named TensorFlow (Abadi et al., 2005). The training was set to 300 epochs at a batch size of 1, and a learning rate of 0.00001. The decay rate of both samples was specified as 0.995. The network attempted to minimize the loss of the training when processing each epoch.

Table 5.2 Details on the training and testing sets from the motorway and monorail datasets

Dataset	Training set			Testing set
	Original images	Augmented images	Total images	
Motorway	150	1,850	2,000	100
Monorail	100	1,900	2,000	100

5.2.2.3 Network evaluation

To evaluate the system, precision, recall, F1-score (F1), and IoU were utilized to quantify the performance of the network. The calculations of these parameters are shown in Equations 5.3 to 5.6:

$$\text{Precision} = \frac{SP_C}{SP_T}, \tag{5.3}$$

$$\text{Recall} = \frac{SP_C}{SP_{GT}}, \tag{5.4}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{5.5}$$

$$\text{IoU} = \frac{SP_C \cap SP_{GT}}{SP_T \cup SP_{GT}}, \tag{5.6}$$

where SP_C is the number of correctly segmented pixels, SP_T is the total number of pixels segmented by the DCNN, and SP_{GT} is the total number of labeled pixels in the ground truth images. For the IoU, $SP_C \cap SP_{GT}$ is the number of pixels correctly segmented by the algorithm that are intersected with the labeled pixels inside the ground truth image, and $SP_T \cup SP_{GT}$ is the number of pixels segmented by the algorithm plus the number of pixels in the ground truth image that were not segmented by the system.

To further validate the system, 10-fold cross-validation (10-fold CV) (McLachlan et al., 2004) was also implemented to validate the generalization of the dataset. After the cross-validation was complete, the performance values of each parameter were summarized and averaged to find the mean values of each parameter. Next, the system was tested on the testing set of each sample to evaluate the performance of the system. **Table 5.3** shows the performance evaluation of the network after training on both datasets for 300 epochs with the evaluation from 10-fold CV and testing set. Moreover, the average losses of each dataset are shown in **Figure 5.9**. The sample predicted image that utilized **Figure 5.7** as an input and **Figure 5.8** as the labeled ground truth is shown in **Figure 5.10**.

Table 5.3 Performance evaluation of the proposed network at 300 epochs

Parameter	Motorway dataset		Monorail dataset	
	10-fold CV	Testing set	10-fold CV	Testing set
Precision	98.02%	97.88%	97.80%	95.91%
Recall	97.86%	97.74%	97.74%	95.61%
F1-score	97.94%	97.81%	97.77%	95.76%
IoU	85.26%	84.12%	85.40%	79.04%

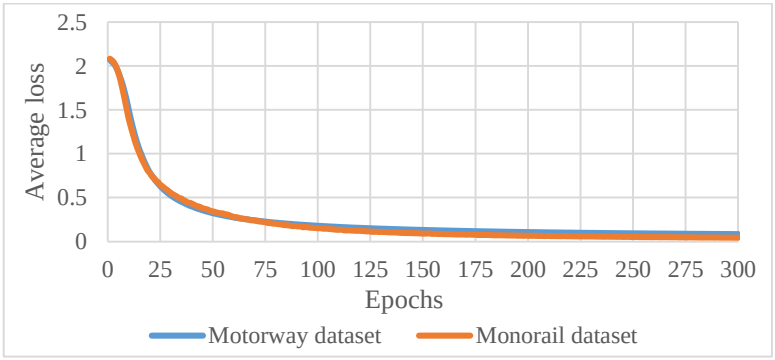


Figure 5.9 Average loss when training on the motorway and monorail datasets

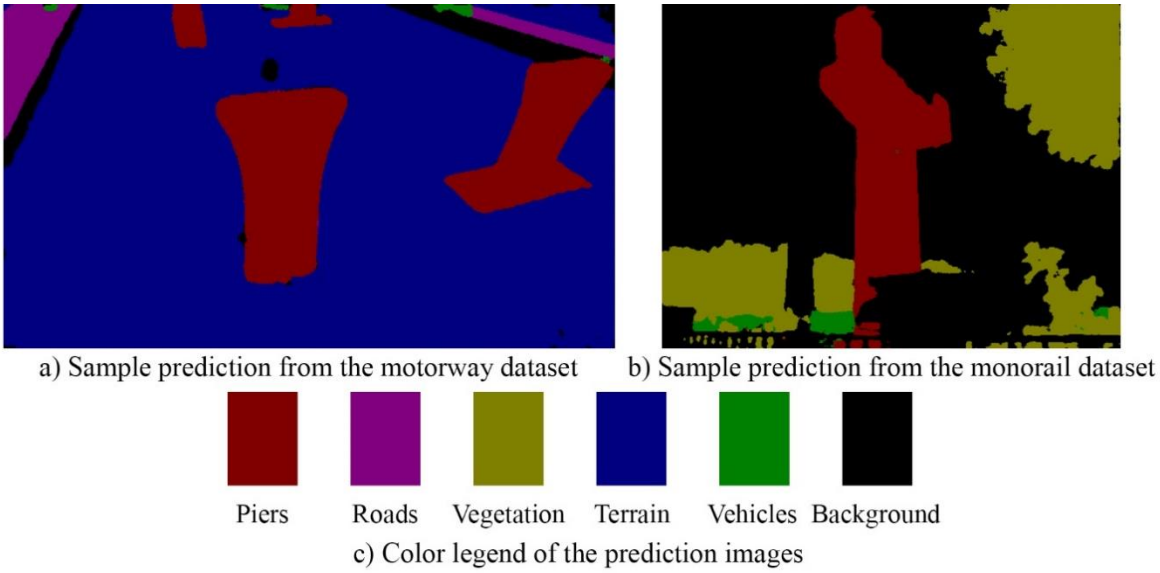


Figure 5.10 Sample predicted images from motorway and monorail datasets in this chapter

The performance evaluation results showed that the network was trained thoroughly because it was able to achieve over 97% in every aspect of evaluation except for the IoU, which the network correctly identified about 85% of the time. The evaluation results showed that the network performed well on the testing sets, especially with the dataset from the motorway pier, achieving nearly the same performance with a 10-fold CV. Meanwhile, the performance on the monorail testing set was about 2% lower for precision, recall, and F1-score. The IoU on the testing set decreased to 79.04% due to the complexity of the dataset.

5.2.2.4 Unwanted-feature removal programming

The workflow of the unwanted-feature removal application is shown in **Figure 5.11** along with **Algorithm 5.1**, which shows the implementation using the DCNN. First, the input images (*I*) and label classes (*C*) are processed into the system by the user. The unwanted classes (*UC*) can be selected from the 6 classes ($j = 6$) that were utilized in the training of the DCNN (i.e., piers, roads, vegetation, terrain, vehicles, and background). In this study, *UC* had 5 classes ($k = 5$) inside, which comprised the abovementioned classes except for piers. Next, the pre-trained DCNN segmented the input images according to the 6 classes (*S*). The function **try** was used to check whether sufficient computational memory was available. If so, the system was able to remove the pixels belonging to the specified unwanted classes in the original resolution (*S*). The system returned the images with unwanted features replaced by white pixels. It is important to note that the Exchangeable Image File Format (EXIF), which contains important data such as georeferencing information, is still attached to the output images.

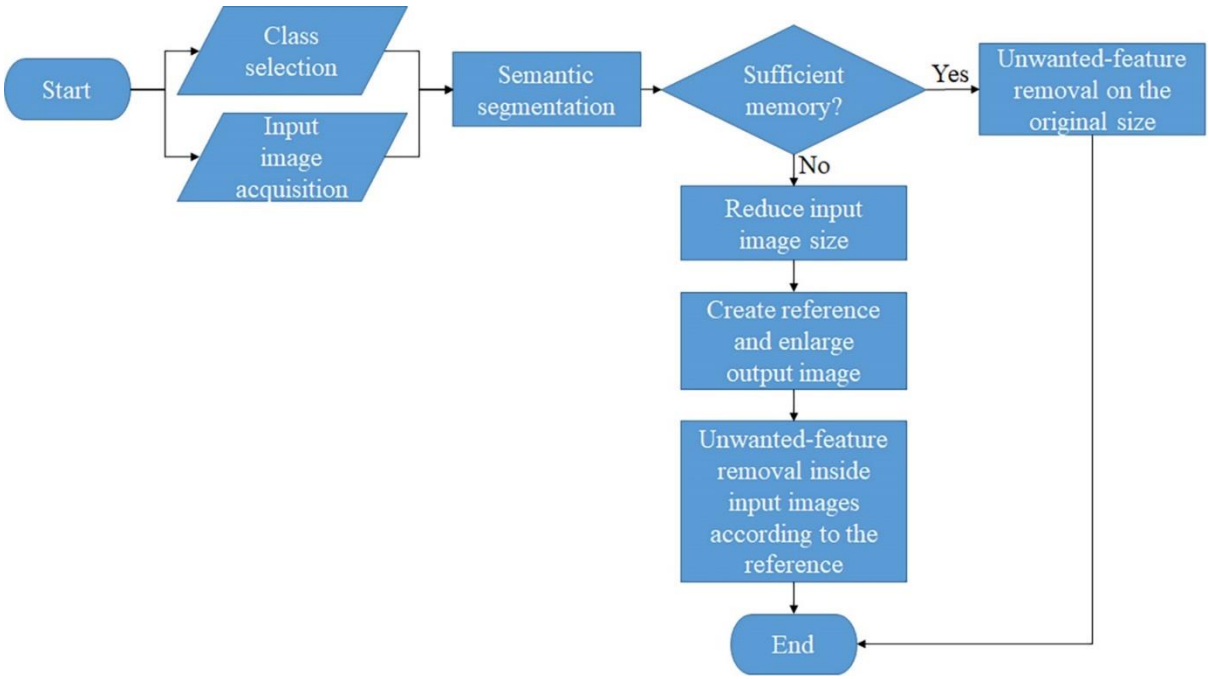


Figure 5.11 Workflow of the unwanted-feature removal application

When the system was able to detect insufficient memory availability, the size of the input and output images had to be reduced. In this study, input images were reduced to 512 pixels (w_1 and h_1 = 512), similar to the size of the training images. The output images were restored by the system to the size (w_0, h_0) of the original image (I_i), which caused some blurriness in the images (E_i). To prevent the loss of detail, this enlarged output was used only as a reference for unwanted-feature removal. Next, the system made a copy of the original image (U_i) and implemented the unwanted-feature removal process according to the algorithms in Equation 5.1 and 5.2. The final output was images with the same EXIF data as the original images, but with the unwanted features removed.

To further illustrate this process, **Figure 5.12** shows examples of images at each step. The only target class, in this case, was piers (red). Therefore, the other classes shown in **Figure 5.12b** (which were roads, terrain, vegetation, vehicles, and background), were automatically removed from the original image (**Figure 5.12a**). **Figure 5.12c** shows the output image after the unwanted features were removed by the DCNN. All other pixels that did not belong to the pier class were removed by the DCNN. **Figure 5.12d** shows the color legend used in this segmentation.

5.2.3 Testing set creation

The two samples discussed in **Section 5.2.2** were further processed into testing sets to evaluate the performance of the unwanted-feature removal system. The number of images in the motorway and monorail samples were 250 and 254, respectively. These images were from the testing sets of each respective sample in **Table 5.2** combined with new images taken from the corresponding sites. Images

Algorithm 5.1: Unwanted-feature removal using the DCNN

Input: Original images $I = \{I_i | i=1,2,3,\ldots,n\}$; Label classes $C = \{C_i | i=1,2,3,\ldots,j\}$ when C_i is the color of the class (r_i, g_i, b_i)

Output: Images with unwanted features removed

1. The user selects unwanted classes $UC = \{UC_i | i=1,2,3,\ldots,k\}$ from C with the color for removal (r_r, g_r, b_r)
2. **for** I_i in I:
3. Load I_i
4. Extract the width (w_0) and height (h_0) of I_i
5. **try:**
6. Process I_i into the DCNN to get the segmented image S_i based on the label classes C
7. Remove pixels that belong to UC and save
8. **except**:
9. Reduce the size of I_i to (w_1,h_1), process into the DCNN to get S_i based on C
10. Remove pixels that belong to UC
11. Enlarge image S_i to the same size (w_0,h_0) of I_i and define it as image E_i
12. Copy image I_i and define it as image U_i
13. Define $p_u(x,y)$ = the color value of the pixel at the coordinate (x, y) of image U_i
14. Define $x = 0$, $y = 0$, and $p_e(x,y)$ = the color value of the pixel at the coordinate (x, y) of E_i
15. **while** $x < w_1$:
16. **while** $y < h_1$:
17. $p_e(x,y) = (r, g, b)$
18. **if** $(r, g, b) \in UC$
19. $p_e(x,y) = (r_r, g_r, b_r)$
20. Save image U_i
21. **else:**
22. **continue**

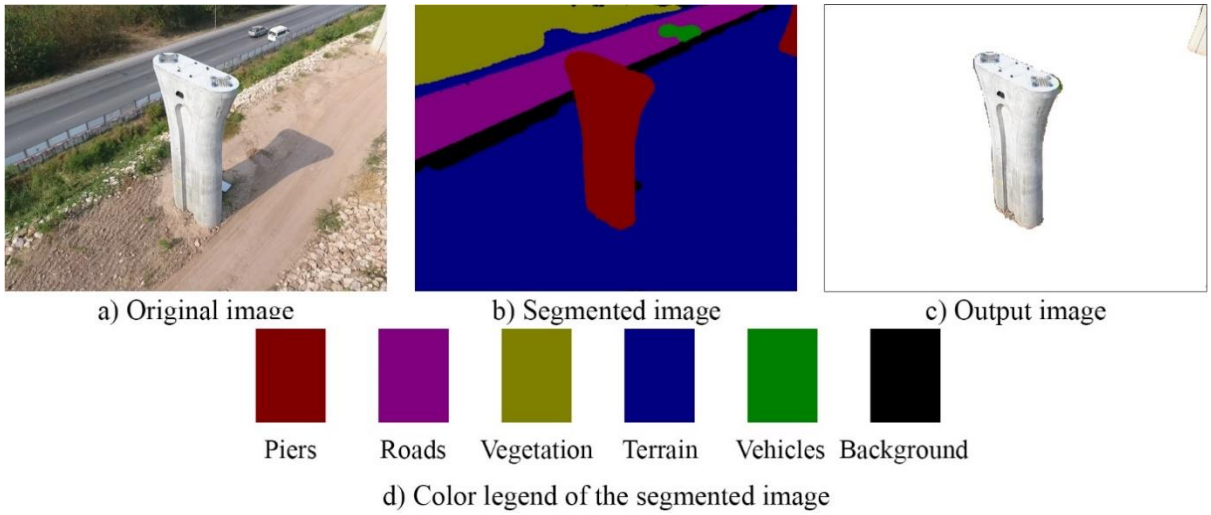


Figure 5.12 Example of the unwanted-feature removal from the proposed DCNN

from training sets were not utilized to eliminate the bias from the training. The images of each sample were copied and further processed into three testing sets (i.e., the original testing set, the DCNN testing set, and the ground truth testing set) to evaluate both sample images. The original testing set comprised the images of each sample without any modifications. The ground truth testing set was clustered by processing the original images through an image editing application to manually remove the unwanted features. The original and the ground truth images are shown in **Figure 5.13**. These 250 motorway and 254 monorail images were processed into the developed system, had the DCNN remove the unwanted features, and used the output images as the DCNN testing set. An example output image is shown in **Figure 5.12c**.

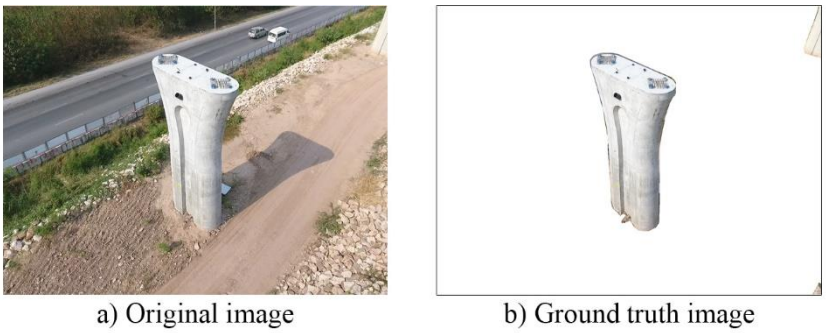


Figure 5.13 Comparison of original and ground truth images

5.2.3.1 Motorway sample

The motorway sample comprised images of a motorway pier, which was constructed with reinforced concrete. The top part of the pier was slightly wider than the body. The plastered surface of the pier had few features. **Figure 5.14** to **Figure 5.16** show the sample images in each testing set.

The ROI in this sample comprised the entire pier structures, including the drainage pipes. The soil, boulders, vegetation, cars, and roads were excluded from the ROI and were manually removed for the ground truth testing set. The DCNN was able to remove most of the unwanted features, but the drainage pipe at the bottom of the pier was slightly cut out due to the similarity between the drainage pipe and the soil on the ground.

Figure 5.14 Samples from the original motorway testing set

Figure 5.15 Samples from the motorway sample with unwanted features removed by the DCNN

Figure 5.16 Samples from the motorway sample with unwanted features manually removed

5.2.3.2 Monorail sample

The monorail sample comprised images of monorail piers constructed with reinforced concrete covered in a steel casing. The top part of the pier was finished with plain concrete but the surface of the pier was made from reinforced steel, which did not have many features to be detected. **Figure 5.17** to **Figure 5.19** show sample images from each testing set.

The reason black is chosen for the removal color in this sample is that the red, green, and blue color values in the images were weakly contrasted against the light and shadows (Jung et al., 2019). When the images were captured, the steel casing of the monorail pier reflected the sunlight and distorted the actual color. In this case, the color of the steel casing appeared white, which, when combined with its smooth surface and few features and placed over a white background, could complicate the construction of the point cloud because the image registration algorithm might have difficulty in distinguishing between the boundary of the pier and the background. Thus, black was chosen as the removal color in this sample.

The ROI in this sample comprised the piers and the surrounding guardrails. The traffic cones next to the piers, the cables, and the steel plates at the top of the pier were excluded from the ROI and were manually removed along with other objects, such as utility poles, signs, trees, cars, and buildings, in the ground truth testing set. The DCNN performed well on this sample and eliminated most of the unwanted features, including the rings of the traffic cone.

Figure 5.17 Samples from the original monorail testing set

Figure 5.18 Samples in the monorail sample with unwanted features removed by the DCNN

Figure 5.19 Samples from the monorail sample with unwanted features were manually removed

5.2.4 SfM process

Next, the three testing sets for each sample were independently used as inputs to construct their respective SfM point clouds in Agisoft Metashape (version 1.6.0, 64 bit) (Agisoft LLC, 2019). The processes were separated into three steps in Agisoft Metashape: image alignment, dense point cloud creation, and outlier and unwanted-feature cleaning. The SfM workflow in this study was performed on a computer with an Intel® Core™ i7-7700 CPU @ 3.60 GHz and 48.0 GB of RAM. The computational times of each step were also measured for further evaluation.

First, the input images were processed into Agisoft Metashape and image alignment was applied. The quality of the alignment was set at ultra-high without any further modified parameters. The images were processed to identify similar features between images, after which these features were matched and tie points were constructed. Finally, the features were matched in the three-dimensional environment.

The motorway sample presented some challenges. The top of the pier was too high relative to the surrounding environment and was therefore excluded from the normal work region by the application. Therefore, the work region of the project had to be manually enlarged for the original testing set. The DCNN and ground truth testing sets did not require any adjustments because the work region already covered the entire pier.

Next, the matched features were used to create depth maps for the point cloud generation. The quality of this process was set at high, rather than ultra high, to ensure the sufficient memory. Other parameters were not modified and the default values of the application were used. The point cloud generation process took from minutes to hours depending on the complexity of the input images and the number of matched features.

The point cloud outputs of the SfM process usually contained noise and outliers, which had to be removed before the evaluation. Noise was filtered out using the threshold function in Agisoft Metashape. Cloud points in the output that had confidence lower than the threshold of 5 and 2 for the motorway pier and monorail pier samples, respectively, were eliminated to decrease noise without the risk of bias from manual removal. However, due to the large amount of cloud points in the original testing set, the ROI was manually selected to let the application calculate the number of cloud points inside the ROI after applying the threshold filtering.

5.3 Results of the quantitative unwanted feature removal evaluation

In this section, the variables of the outputs from each sample were compared to verify the performance of the proposed system that removed unwanted features using the DCNN. The comparison was separated into three aspects: the number of aligned images, the number of features and cloud points, and computational time.

5.3.1 Number of aligned images

The number of images aligned by the application without human intervention is shown in **Figure 5.20**, which shows that applying unwanted-feature removal to these samples did not reduce the number of alignable images.

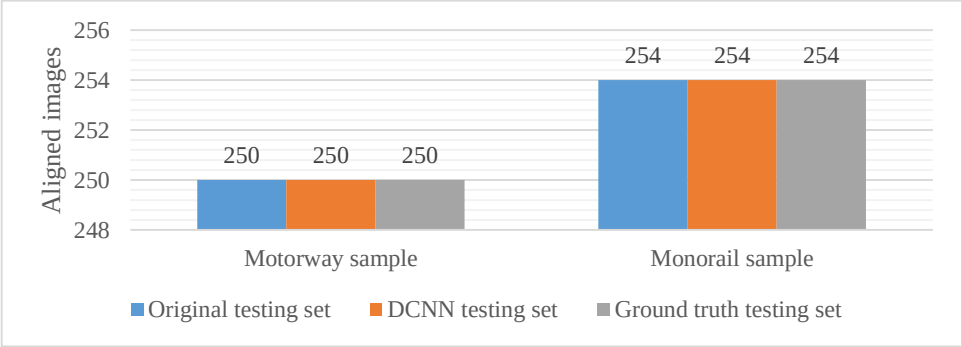


Figure 5.20 Number of aligned images in each testing set

5.3.2 Features and point clouds

The numbers of features and cloud points were determined and separated into subsections: the number of features, the total number of cloud points, and the ROI point cloud. To obtain these values, each testing set was processed by the application five times. The average values from these computations were used as the representative values of each testing set to reduce the processing error.

5.3.2.1 Number of features

The numbers of detected features for each sample are shown in **Figure 5.21**. In each sample, the numbers of features in the original testing set were the highest, followed by the ground truth and the DCNN testing sets. The features of each sample are shown in **Figure 5.22** and **Figure 5.23.** These features were created by Agisoft Metashape after image alignment was complete.

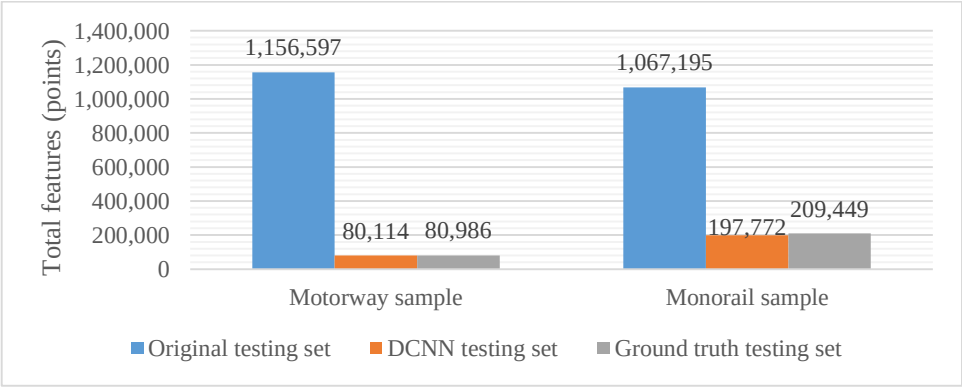


Figure 5.21 Total features in each testing set

In the motorway sample (**Figure 5.22**), most of the features representing classes other than the piers were removed by the developed DCNN, leaving only two piers in the center of **Figure 5.22b**, with some features from more distant piers, as in the ground truth testing set. The difference between the number of point clouds between these outputs was minimal.

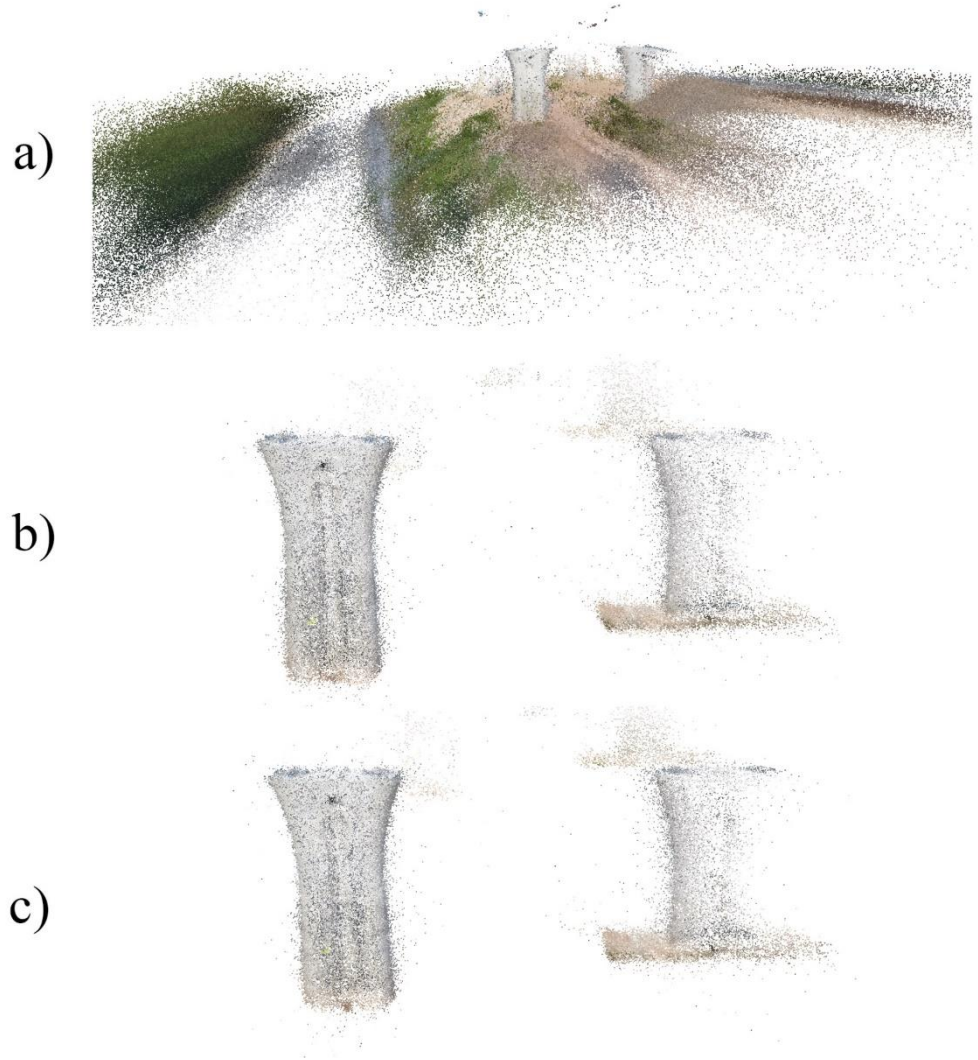


Figure 5.22 Features of each testing set from the motorway sample:
a) original testing set, b) DCNN testing set, and c) ground truth testing set

Due to a large number of features in the monorail sample (**Figure 5.23**), it was difficult to identify the piers among the noisy feature points (**Figure 5.23a**) resulting from processing the original testing set. The output from processing the DCNN and the ground truth testing sets, yielded only the ROI piers, as shown in **Figure 5.23b** and **Figure 5.23c**, respectively. However, as shown in **Figure 5.23c**, the output from the ground truth testing set had more cloud points generated from the feature matching.

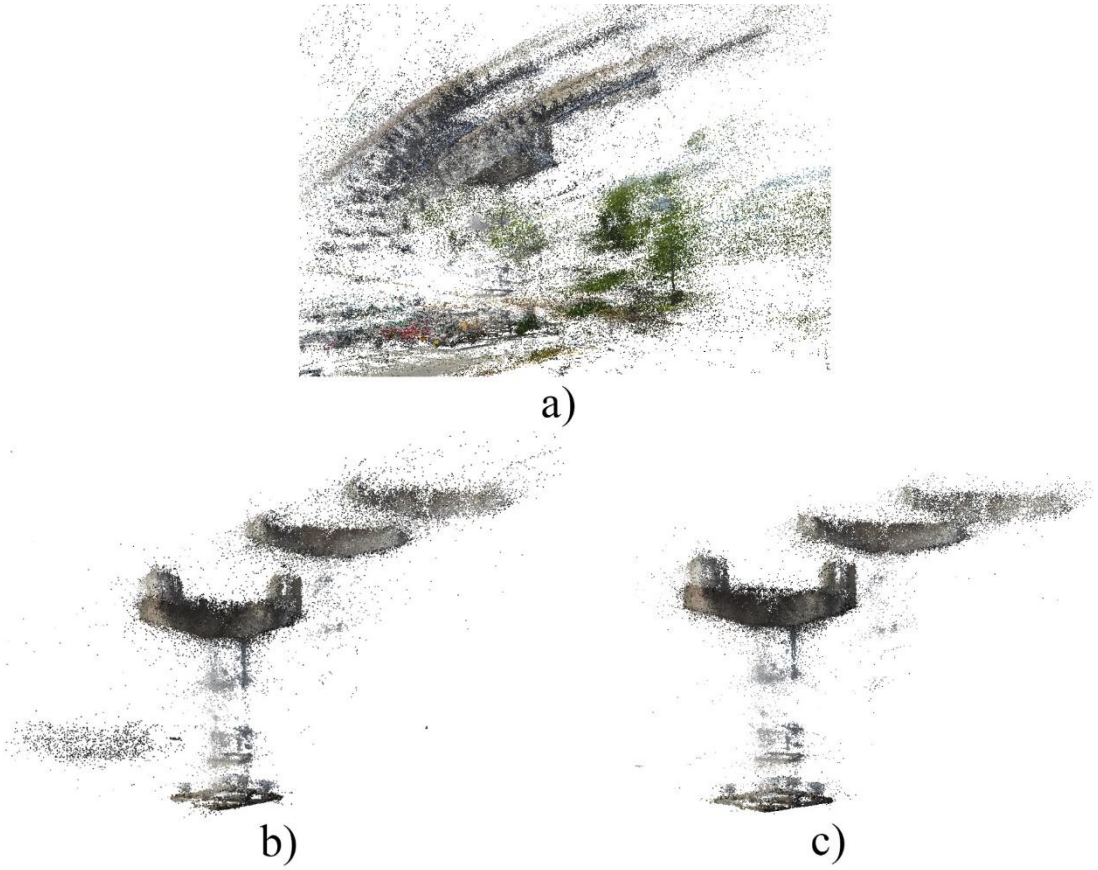
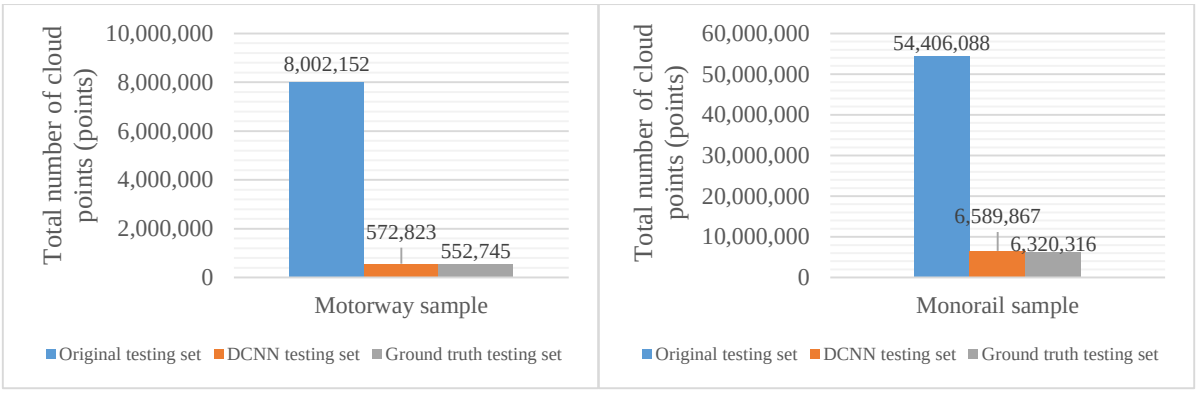


Figure 5.23 Features of each testing set from the monorail sample:
a) original testing set, b) DCNN testing set, and c) ground truth testing set

5.3.2.2 Total cloud points

The detected features were processed and used as the input to generate a depth map for point cloud creation. The total number of cloud points in each sample is shown in **Figure 5.24**. Unexpectedly, although the number of features from the ground truth testing set was higher than that from the DCNN testing set, the number of total cloud points generated from the DCNN testing set was marginally higher than that from the ground truth testing set.

Figure 5.25 shows the point cloud of the motorway sample. **Figure 5.25a** shows the point cloud generated from the original testing set, which was reconstructed clearly, due to the large number of features identified in the previous step. However, two incomplete piers at the back of the scene were constructed poorly due to the low confidence threshold compared to other parts inside the scene such as the ground and boulders as shown in **Figure 5.26**. Therefore, they were counted as noise and not included in the further steps of evaluation. The point cloud from the DCNN testing set (**Figure 5.25b**) is clearer, showing only two piers with some noise that had to be filtered out. The point cloud from the ground truth testing set shown in **Figure 5.25c** also had noise resulting from the previous steps.



a) b)

Figure 5.24 Comparison of total cloud points in each testing set between samples,
a) Number of cloud points in the motorway sample, b) Number of cloud points in the monorail sample

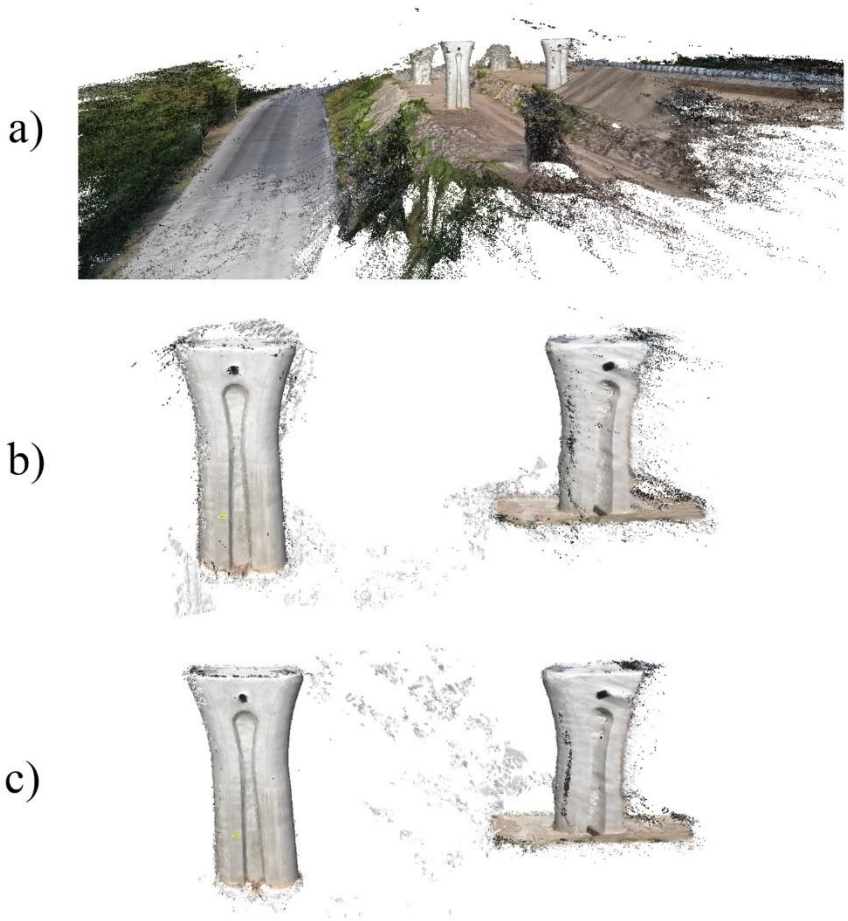


Figure 5.25 Point clouds of each testing set from the motorway sample:
a) original testing set, b) DCNN testing set, and c) ground truth testing set

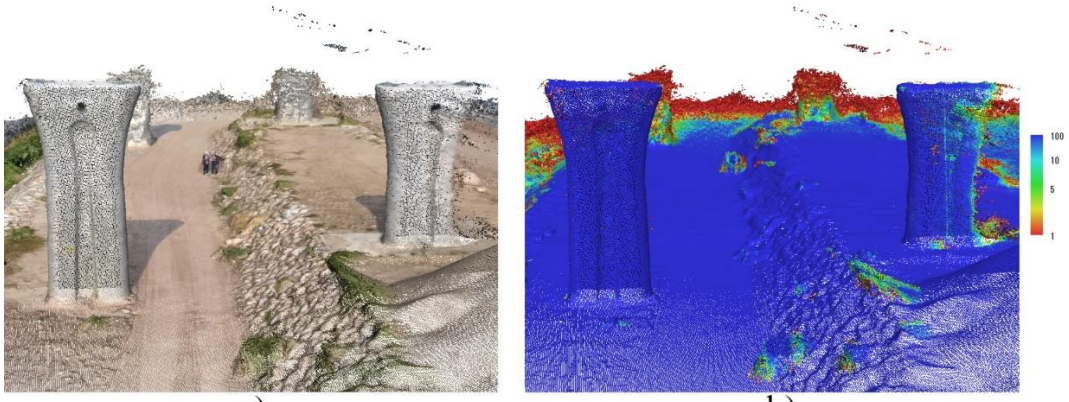


Figure 5.26 Visualization of the confidence threshold of **Figure 5.25**a:
a) Close-up point cloud of **Figure 5.25**a and b) Confidence threshold of a

The point clouds of the monorail sample are shown in **Figure 5.27**. In **Figure 5.27a**, several unnecessary components, such as vehicles, railway tracks, and trees that resulted from processing the original testing set, partially occluded the piers. The scene was made clearer using the developed system, as demonstrated in **Figure 5.27b**. which left only the two ROI piers and some noise. There was less noise around the piers in **Figure 5.27c** from the ground truth testing set, as demonstrated by the numbers shown in **Figure 5.24**.

5.3.2.3 Point cloud of the ROI

Finally, the raw point clouds were modified by filtering out points that the application determined to be of lower confidence than the specified threshold. However, there were still too many points in the outputs from the original testing set due to the excessive number of features, which raised the confidence of the feature matching. Thus, the cloud points inside the ROI in the original testing set were manually selected according to the boundary of the ROI specified by the user. **Figure 5.28** shows the number of cloud points in the ROI for each sample. Surprisingly, although the total number of features and cloud points of the original testing set were much higher than those of the other sets, the number of cloud points in the ROI was actually lower compared with the other sets. The testing set that had the highest number of the cloud points inside the ROI was the ground truth testing set, which had all features other than those of the ROI removed beforehand. The point cloud of the ROI from the DCNN testing set was slightly smaller than that from the ground truth testing set.

Figure 5.29 shows close-ups of ROI point clouds from the motorway sample. The point cloud from the original testing set (**Figure 5.29a**) is visibly less detailed compared with the other sets due to the difference in cloud point numbers, as shown in **Figure 5.28a**. Meanwhile, the point cloud from the DCNN testing set (**Figure 5.29b**) was more complete, judging from the number of cloud points and the

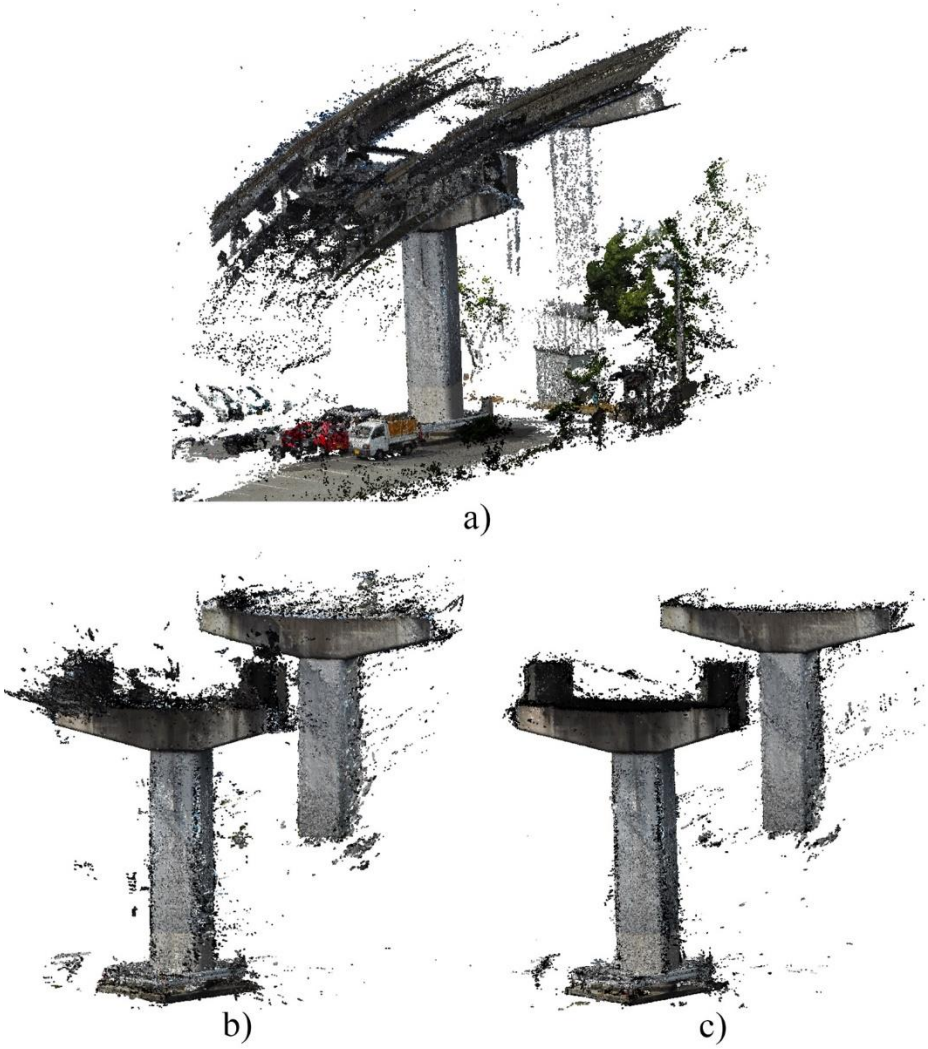
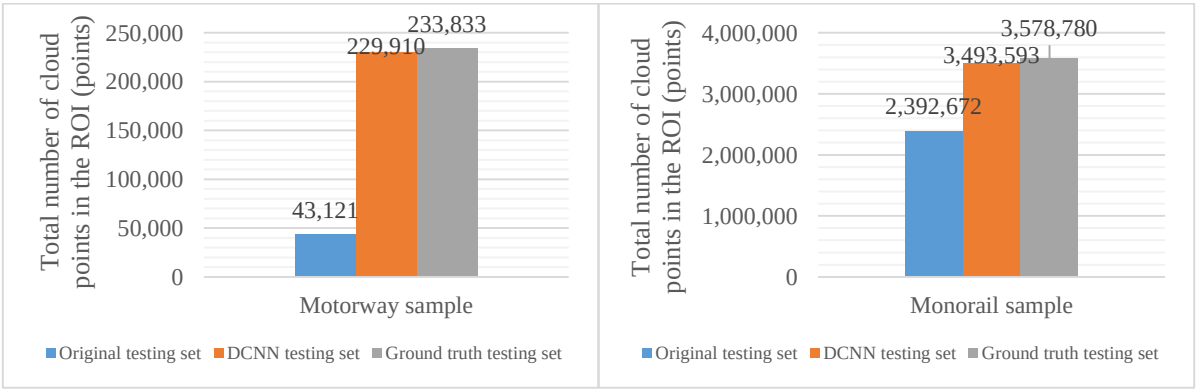


Figure 5.27 Point cloud of each testing set from the monorail sample:
a) original testing set, b) DCNN testing set, and c) ground truth testing set



a) b)

Figure 5.28 Numbers of cloud points in the ROI of each testing set: a) number of cloud points in the ROI of the motorway sample and b) number of cloud points in the ROI of the monorail sample

denseness of the point cloud of the pipe that is clearly visible at the top of the pier. However, the pipe at the bottom is still difficult to distinguish from the ground. The pipes in the point cloud from the ground truth testing set (**Figure 5.29c**) are easy to distinguish because there was less noise after filtering.

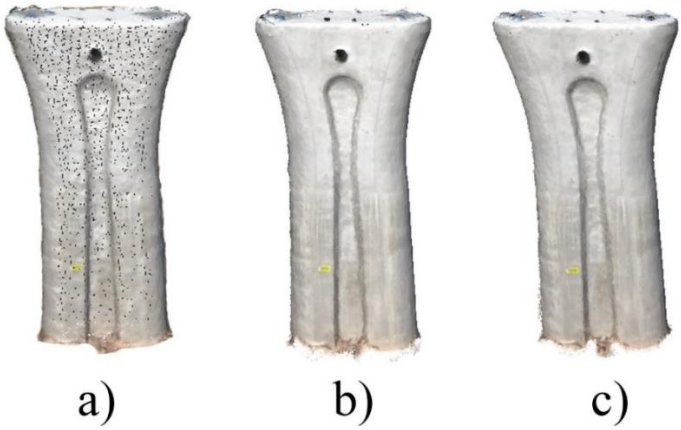


Figure 5.29 ROI point clouds of each testing set from the motorway sample:
a) original testing set, b) DCNN testing set, and c) ground truth testing set

Figure 5.30 shows three point clouds from the monorail sample. The point cloud from the original testing set shown in **Figure 5.30a** has some voids at the top of the pier and on the surface of the pier. The guardrails in the front and back were also distorted by noise. Although there is some roughness in the pier similar to the point cloud from the original testing set, the top part of the point cloud from the DCNN testing set in **Figure 5.30b** is denser and the guardrail is detailed. The point cloud from the ground truth testing set (**Figure 5.30c**) is as detailed as that from the same testing set in the motorway pier sample.

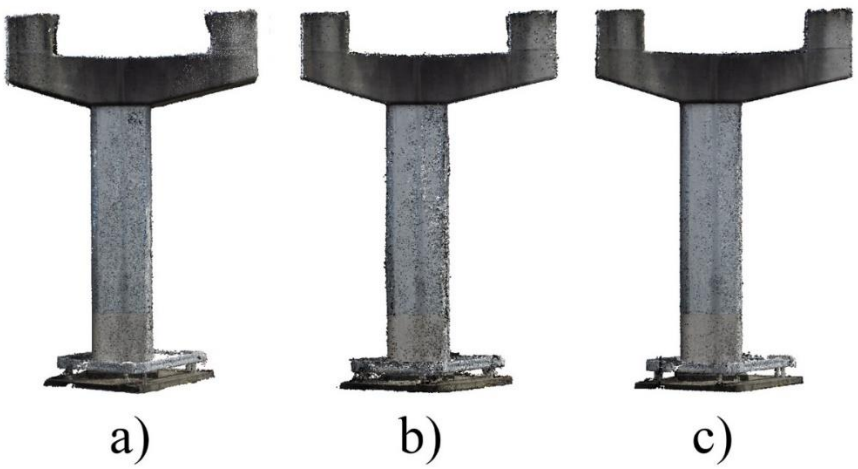


Figure 5.30 Cleaned point cloud of each testing set from the monorail sample:
a) original testing set, b) DCNN testing set, and c) ground truth testing set

5.3.3 Computational time

The last aspect of the evaluation was the amount of computational time required for each step of the SfM implementation. The time sequence was separated into three major phases: pre-processing, processing, and post-processing. **Figure 5.31** shows the pre-processing time before the acquired images can be utilized as inputs for the SfM application. The ground truth testing set consumed the largest amount of time because the manual labeling of unwanted features was labor-intensive and time-consuming. Meanwhile, the original testing set required hardly any time if the images could be aligned without any human intervention. There was some pre-processing time for the original testing set in the motorway sample because the height of the piers exceeded that of the surroundings. Thus, the top parts of the piers were cut off in the workspace of the application and needed some manual adjustment to facilitate the boundary of the dense cloud generation. Finally, the DCNN testing set also required some time for the network to filter out the unwanted features, but less time than was required for the ground truth testing set.

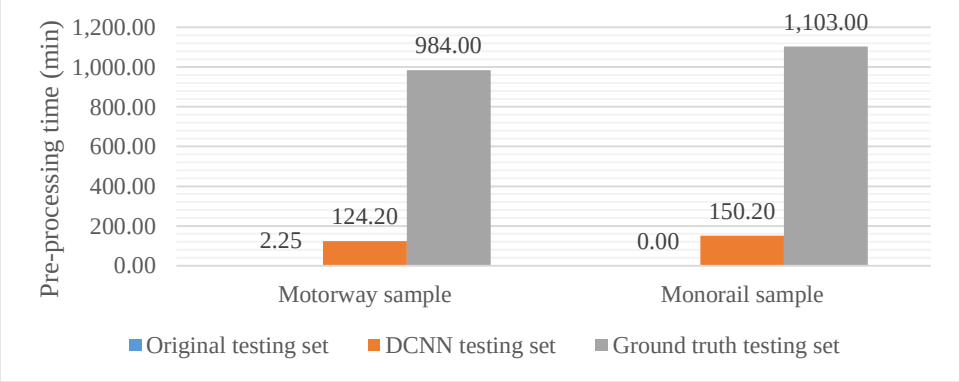


Figure 5.31 Comparison of pre-processing times between testing sets

Figure 5.32 shows the total computational time required for the SfM process, including feature matching, image alignment, depth map generation, and point cloud creation. The original testing set required the largest amount of time due to its large amount of information. In the motorway sample, there was no computational time difference between the DCNN and the ground truth testing sets but the DCNN testing set required slightly more time compared with the ground truth testing set in the monorail sample. Nevertheless, both DCNN and ground truth testing sets required a relatively small amount of time compared with the original testing set.

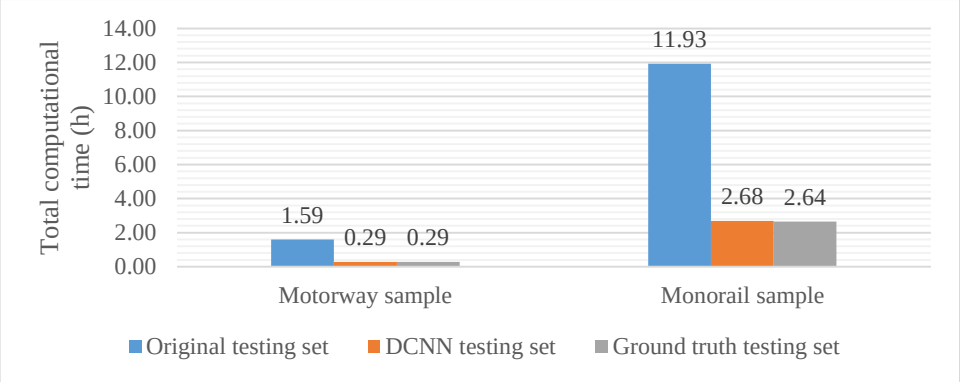


Figure 5.32 Total computational time required for the SfM process in each testing set

The last graph in this section is **Figure 5.33**, which shows the total amount of post-processing time required for each testing set. In this case, post-processing consisted of adjustment and manual point cloud selection to target only cloud points in the ROI. The graph trend was the same in both samples, with the original testing set requiring the largest amount of post-processing time. Meanwhile, the DCNN and the ground truth testing sets required the same amount of time because they did not require any adjustment other than the automated threshold filtering.

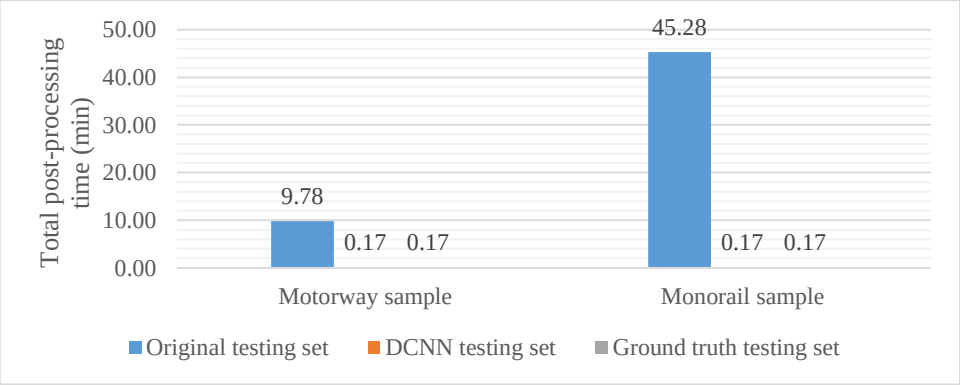


Figure 5.33 Total post-processing time required for each testing set

5.4 Discussion on unwanted feature removal evaluation with point cloud

After processing the original, DCNN, and ground truth testing sets for each of the motorway and monorail pier samples, the evaluation results showed that removing unwanted features from the input of the SfM process raised the number of cloud points inside the ROI, and hence the quality of the output model, was increased. The number of cloud points in the ROI and the percentage of difference in cloud points between the original and the other testing sets are shown in **Table 5.4**. By applying unwanted-feature removal, the DCNN and the ground truth testing set were able to increase the number of cloud points in the ROI by 5.33 and 5.42 times, respectively, in the motorway sample. Meanwhile, in the

monorail sample, the number of cloud points in the ROI was increased by 1.46 and 1.50 times using the DCNN and ground truth, respectively, as the processing input. These results are due to the removal of unwanted features shifting the concentration of the feature matching algorithm to the ROI (Saovana et al., 2019b); more feature matches means more cloud points that will be generated in the subsequent steps of the workflow.

Table 5.4 Percentage of difference between the numbers of cloud points in the ROI between the original and the other testing sets

Sample	Original testing set (A)	DCNN testing set (B)	Ground truth testing set (C)	Difference percentage between the DCNN and the original testing set (B/A) x 100	Difference percentage between the manual and the original testing set (C/A) x 100
Motorway	43,121	229,910	233,833	533.17%	542.27%
Monorail	2,392,672	3,493,593	3,578,780	146.01%	149.57%

Furthermore, applying unwanted-feature removal did not complicate the image alignment because all of the testing sets had the same number of aligned images, as shown in **Figure 5.20**. Unwanted-feature removal substantially reduced total computational time compared with the original testing set in every sample as well as post-processing time because noise resulting from feature matching of images in which unwanted features have already been removed can be easily and automatically filtered out using the threshold filtering. Comparisons between the point cloud and the confidence threshold before and after the filtering of the original and the DCNN testing sets are shown in **Figure 5.34** and **Figure 5.35**, respectively. Without the application of unwanted-feature removal in the image shown in **Figure 5.34**, unwanted features such as boulders, vegetation, and people were not removed due to the high confidence threshold of these points (blue areas in **Figure 5.34b**). Only the small areas of noise and the more distant cloud points were eliminated by threshold filtering. In contrast, the ROI point cloud shown in **Figure 5.35** was neatly and automatically cleaned up by the threshold filtering, although there was more noise in **Figure 5.35a**. This noise was created by uncertainty in the feature matching and depth map calculation, which resulted in very low confidence (red areas in **Figure 5.35b**). Thus, threshold filtering worked very well with this problem and removed these low-confidence points from the output, as shown in **Figure 5.35d** which has no red areas. To quantify this phenomenon, the percentages of unwanted features were calculated, as shown in **Table 5.5** for the motorway pier sample and in **Table 5.6** for the monorail pier sample.

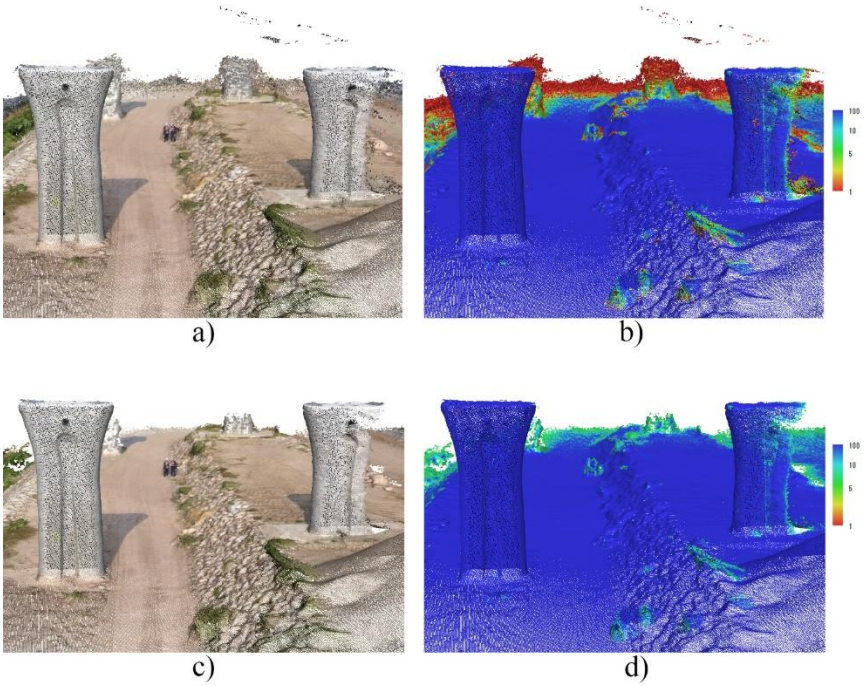


Figure 5.34 Comparison before and after implementing threshold filtering of the point cloud from the processing of the original testing set in the motorway pier sample: a) point cloud with noise and unwanted features, b) confidence threshold of each point cloud in **Figure 5.34a**, c) point cloud after setting the threshold filter to 5, and d) confidence threshold of each point cloud in **Figure 5.34c**

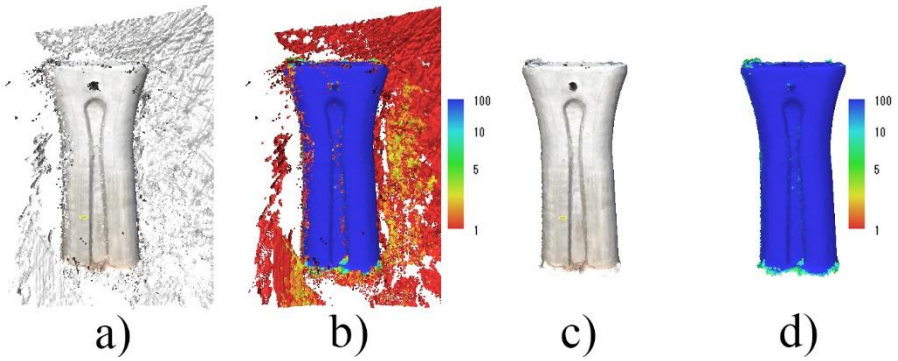


Figure 5.35 Comparison before and after implementing threshold filtering of the point cloud from the processing of the DCNN testing set in the motorway pier sample: a) ROI point cloud with noise, b) confidence threshold of each point cloud in **Figure 5.35a**, c) ROI point cloud after setting the threshold filter to 5, and d) confidence threshold of each point cloud in **Figure 5.35c**

Table 5.5 Percentage of the unwanted cloud points in each testing set of the motorway pier sample

Testing set	Total cloud points (A)	ROI cloud points (B)	Unwanted cloud points (A−B)	Percentage of ROI cloud points (B/A) × 100	Percentage of unwanted cloud points ((A−B)/A) × 100
Original	8,002,152	43,121	7,959,031	0.54%	99.46%
DCNN	572,823	229,910	342,913	40.14%	59.86%
Ground truth	552,745	233,833	318,912	42.30%	57.70%

Table 5.6 Percentage of the unwanted cloud points in each testing set of the monorail pier sample

Testing set	Total cloud points (A)	ROI cloud points (B)	Unwanted cloud points (A−B)	Percentage of ROI cloud points (B/A) × 100	Percentage of unwanted cloud points ((A−B)/A) × 100
Original	54,406,088	2,392,672	52,013,416	4.40%	95.60%
DCNN	6,589,867	3,493,593	3,096,274	53.01%	46.99%
Ground truth	6,320,316	3,578,780	2,741,536	56.62%	43.38%

Without the application of unwanted-feature removal in the original testing set, 99.46% of the resulting point cloud comprised the surrounding environment, including the soil, vegetation, and terrain. Meanwhile, in the DCNN and the ground truth testing set, unwanted features were automatically removed, resulting in about 40% and 50% more ROI cloud points than the original testing set in the motorway and monorail samples, respectively.

However, manually removing unwanted features at the beginning of the work process was very labor-intensive and time-consuming. The pre-processing time for the ground truth testing set was longer than that of the other testing sets. Combining the total time spent from pre-processing to post-processing, the ground truth testing set required the most time of the three testing sets in each sample. **Figure 5.36** shows the total processing time for each testing set.

Using the DCNN to assist the removal of unwanted features is possible. Although the removal performance of the DCNN was slightly behind the usage of the ground truth due to the difference in the prediction of unwanted features from the DCNN and the ground truth, the DCNN was able to substantially reduce the processing time required for SfM. The DCNN performed well in the monorail sample by substantially reducing the total computational time (**Figure 5.32**). As shown in **Figure 5.36**, the DCNN testing set required the least amount of time, less than the original and the ground truth testing sets. To further compare these values, **Table 5.7** and **Table 5.8** show the percentage of differences in ROI cloud points, pre-processing time, and total processing time between the ground truth testing set and the DCNN testing set of both samples.

As shown in **Table 5.7** and **Table 5.8**, the number of ROI cloud points in the DCNN testing set was lower than that in the ground truth testing set by only 1.68% for the motorway pier sample and by 2.38% for the monorail pier sample. Meanwhile, the DCNN was able to reduce the pre-processing time by 87.38% and 86.38% compared with the ground truth in the motorway and monorail samples, respectively. In addition, the total processing time was also decreased by 85.85% for the motorway sample and 75.33% for the monorail sample when the DCNN assisted with unwanted-feature removal. Thus, unwanted-feature removal using the DCNN is applicable to the SfM process for infrastructure

inspection and monitoring because it enables features outside the ROI to be eliminated, thereby substantially reducing the required computational time.

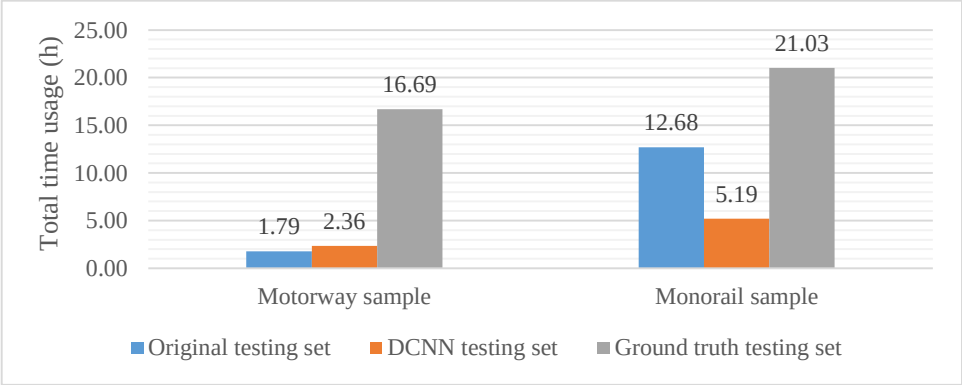


Figure 5.36 Total processing time required for each testing set

Table 5.7 Parameter comparison between the ground truth and the DCNN testing sets in the motorway pier sample

Parameters	Ground truth testing set (A)	DCNN testing set (B)	Percentage difference ((A−B)/A) × 100
ROI cloud points	233,833	229,910	1.68%
Pre-processing time (min)	984.00	124.20	87.38%
Total processing time (h)	16.69	2.36	85.85%

Table 5.8 Parameter comparison between the ground truth and the DCNN testing sets in the monorail pier sample

Parameters	Ground truth testing set (A)	DCNN testing set (B)	Percentage difference ((A−B)/A) × 100
ROI cloud points	5,476,374	4,573,565	2.38%
Pre-processing time (min)	1,103.00	150.20	86.38%
Total processing time (h)	20.46	5.56	75.33%

5.5 Conclusion of this chapter

Based on previous studies (Saovana et al., 2019a, 2019b), manual removing unwanted features from images processed using the SfM technique can increase the number of feature matches in the ROI and reduce the computational time required to complete the entire process. However, although a higher number of feature matches in the ROI can be achieved, the total number of features in the entire scene can be drastically decreased and complicate the overall process of the SfM technique. This uncertainty needs to be clarified and quantified before unwanted-feature removal can be applied to the SfM technique, which studies on this scope have not yet to be seen. Furthermore, manually removing unwanted features in previous studies is very labor-intensive and time-consuming, especially when it is

applied to the large number of images required as input for the SfM technique in construction monitoring and infrastructure inspection. Thus, a new procedure for assisting with unwanted-feature removal should be investigated.

This chapter proposed a novel method for assisting with unwanted-feature removal in images of repetitive infrastructure such as piers by using deep learning to reduce the pre-processing of the manual unwanted-feature removal and the post-processing of the conventional workflow. The developed DCNN was trained on the digital images of real-world infrastructure to understand the meaning of each pixel, enabling it to aid the SfM process by removing pixels belonging to unwanted classes specified by the user. The precision rate, recall rate, and F1-score of the proposed network after training on two datasets (i.e., motorway pier and monorail pier) with a 10-fold CV for 300 epochs were over 97%, whereas the IoU of both samples was 85.26% and 85.40% for the motorway and monorail datasets, respectively, in which the small details that did not belong to the ROI structure such as traffic cones and signs were completely removed from the input of the DCNN dataset. When the testing sets were evaluated, their performance showed that the datasets were generalized.

Next, three testing sets (i.e., the original testing set, the DCNN testing set, and the ground truth testing set) were created to verify the effect of unwanted-feature removal with the SfM process. The original testing set comprised unmodified sample images from the datasets. The DCNN testing set was generated by processing the original images into the proposed system based on the developed DCNN. For the last testing set, unwanted features were manually removed using the ground truth of the DCNN training as the reference.

The evaluation results showed that the DCNN-assisted unwanted-feature removal was promising. The main advantage of the developed system was the 85.85% and 75.33% reductions in the computational time required for manual processing in the motorway monorail samples, respectively, which was achieved without complicating the image alignment. The use of threshold filtering with the DCNN was also investigated and the results showed that it worked very well with the output from the DCNN by automatically removing noise and substantially reducing post-processing time.

For the amount of ROI point clouds, the result showed that without the utilization of the proposed approach in the first step of SfM, the ROI point cloud created by the traditional method were only 0.54% of the entire point cloud output from the motorway sample and 4.40% of the final point cloud from the monorail sample. These percentages of point clouds can be easily treated as outliers by the traditional noise filters and get removed from the point cloud outputs. Implementing the proposed system was able to increase the number of cloud points in the ROI by 5.33 and 1.46 times for the motorway and monorail samples, respectively. The DCNN can be used to remove unwanted features, thereby substantially reducing the computational time, especially with the projects that have a large number of images with few differences, such as those used in the monitoring and inspection of infrastructure. On-site inspectors

do not have to remove unwanted features such as the ground, vegetation, and boulders whenever monitoring or inspection is needed; they can train the DCNN once and use it right away and continue using it until the project is finished or the structure is demolished. The faster speed and higher number of ROI point clouds can support the decision-making to be more rapid and reliable, which is crucial to the management of the site practitioners and stakeholders.

The images used in this study are from actual piers in Thailand and Japan. Therefore, the knowledge learned by the developed network can be applied only to similar structures (Alex Braun and Borrmann, 2019). It is very challenging to create a generalized dataset encompassing all types of civil infrastructure because infrastructure designs vary from location to location (Spencer et al., 2019). Nevertheless, this proposed methodology can be applied to numerous structures in other projects.

Finally, this proposed methodology has some limitations. First, the number of images with good overlapping in the ROI is crucial to the success of unwanted-feature removal because the number of features in the processed images is already low. If the number of overlapping images is not high enough, the image features might not be appropriately matched and the images might not be aligned. Due to this reason, in case the image capturing cannot be done properly for the entire site at once, it is better to split the process into smaller parts of the site or individual piers for better reconstruction. The second limitation is that the user cannot intervene with the decision of the system when it decides which pixels belong to which class due to the black-box nature of deep learning. The system might remove parts of the ROI if sufficient training and validation have not been performed. The final limitation is that the developed methodology was tested on only two samples, which although the concept of utilizing semantic segmentation to segment the unwanted-features out of the SfM input is very general and applicable to any shapes of infrastructures if the data for training is available, it should be further verified with additional samples to increase its reliability and generalization.

Chapter 6
Automated feature points and point cloud classification

The benefit and an innovative point from the proposed approach in chapter 5 is the higher amount of the ROI point cloud as demonstrated in **Table 5.5** and **Table 5.6**. Processing the images from the developed system into the SfM process can raise the number of the ROI point cloud from 43,121 points in the motorway sample to be 229,910 points, which can increase the quality of the output model as shown in **Figure 5.29**. Traditional noise and outlier removal implementing the removal at the end of the entire process, which does not raise the number of the final ROI point cloud like what this approach presented and still require manual intervention if the practitioners do not wish to use the entire scene because there are still undesirable features left in the scene such as the vegetation and boulders. The computational time was also decreased because all of the undesirable features such as boulders and the vegetation, which are rich in features, inside the images were removed as can be seen from **Figure 5.36** that when combining all of the processing time, the proposed approach can decrease the total computational time of the SfM process in the Monorail sample lower than the original workflow.

Nevertheless, the current process utilized semantic segmentation with the digital image, in which although semantic segmentation is powerful enough to classify the scene into classes already, it still faces some difficulties when the object classification inside the class is needed. Therefore, instance segmentation should be implemented when a clear boundary between objects in the same class is essential. In this chapter, the novel automated classification algorithm is presented and validated firstly with image features inside a two-dimensional scene. Then, the proposed algorithm is expanded to

handle the three-dimensional environment to classify the point cloud of the infrastructure scene based on the digital image from a daily work basis on the site.

6.1 Automated feature points classification

Image feature detection from digital images is a fundamental method in the computer vision domain (Apollonio et al., 2014; Li et al., 2015). By classifying these features enables the opportunity to better manage the further work processes such as feature matching, image registration, and point cloud generation from photogrammetric techniques. However, although image feature extraction by utilizing feature detection algorithms can rapidly discover the meaningful pixels, the image matching from these features is prone to error from outliers that can decrease the matching accuracy (Ali and Whitehead, 2014) and reduce the overall quality, especially when it has to deal with challenging scenes such as a rough terrain of infrastructure scenes with numerous unnecessary features of the vegetation and boulders (Saovana et al., 2019b).

Thus, there is a need for a novel system to assist the image feature classification to better specify the features of interest from the infrastructure sites. One of the promising techniques that may be able to support this work process is the semantic image segmentation from deep learning algorithms, which can give the boundary of the object of interest inside each image based on the pre-trained knowledge. Nevertheless, there are no studies that focus on these aspects. Therefore, to fill this research gap, this subsection proposes the development of a new system to automatically classify the features of interest inside infrastructure images based on pre-trained deep learning.

6.1.1 Experimentation for automated classification of image features

The experimentation of this subchapter is shown in **Figure 6.1**. First, the features inside test images were specified by a feature detector algorithm. Then, the test images are utilized as inputs for the pre-trained DCNN to segment the object of interest inside each image. Next, the contour of the segmented image outputs from the previous step is extracted and used as the boundary of the object of interest. Finally, the boundary is processed into Delaunay triangulation (Delaunay, 1934) to find the features inside the object of interest. The output of the system is further compared to the manual image editing to validate the performance of the system.

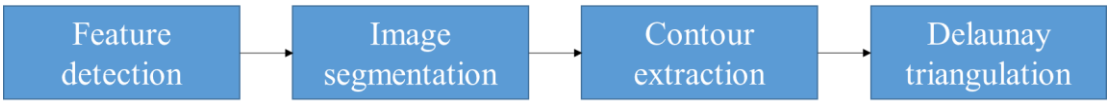


Figure 6.1 Experimental procedure of this part

The testing image set was captured from an infrastructure site in Nakhon Ratchasima, Thailand. The image capturing equipment was a DJI Phantom 4 drone. The resolution of the image was 1,920 × 1,080

pixels in PNG format. The object of interest in this dataset is the piers inside the images. The feature detector algorithm in this study was the SURF detector (Bay et al., 2008), which had a satisfactory result with infrastructure images (Saovana et al., 2019a, 2019b). The setting of the algorithm is the default value from OpenCV 3.4.2 (Bradski and Kaehler, 2008), which is an open-source library for image management. The specification of the testing computer is Intel® Core™ i5-8400 CPU @ 2.80GHz and 8.00 GB RAM. **Figure 6.2** shows the detected features from the SURF algorithm.

a) Original input image b) SURF features detected by the algorithm (red dots)

Figure 6.2 Sample of detected features from the SURF algorithm

The DCNN in this experiment utilized the architecture inspired by U-Net (Ronneberger et al., 2015). It contains encoder and decoder parts, which also connected with the copy-and-paste process that can raise the quality of the segmentation result. U-Net has been recognized for its high accuracy in many scopes of works (Abrams et al., 2019; Garcia-Garcia et al., 2018; Izutsu et al., 2019). The DCNN was trained on one thousand images and tested on one hundred twenty-five images. It should be noted that the system had never seen the test data in order to also prove the generalization of the system. The detailed information on train and test sets are shown in **Table 6.1**. Data augmentation was implemented to raise the number of images in the train set by up to 10° angle rotation, horizontal flipping, and brightness adjusting.

Table 6.1 Detailed information on train and test sets

Parameter	Training set			Test set
	Original images	Augmented images	Total images	
Number of images	125	875	1,000	125

Training was conducted on a computer that has specifications as follows: Intel® Core™ i7-8086K CPU @ 4.00GHz with 32.0 GB of RAM and an NVIDIA GeForce RTX 2080 Ti graphics card. The DCNN was performed on Tensorflow, which is an open-source platform for deep learning. Three hundred epochs were set as the number of training with the batch size of one, the learning rate of 0.00001, and the decay rate of 0.995. These hyperparameters were set so the system can slowly reach the most optimal answer without overfitting. The optimization strategy for learning is to minimize the loss in each epoch

of training. There are two classes in the training, which were piers and background. The validation of the network was based on four aspects, namely precision, recall, F1 score, and intersection over union (IoU). The formulas utilized to calculate for these values are shown as Equations 6.1 – 6.4:

$$\text{Precision} = \frac{SP_C}{SP_T} \tag{6.1}$$

$$\text{Recall} = \frac{SP_C}{SP_{GT}} \tag{6.2}$$

$$\text{F1} = \frac{2 \text{ x Precision x Recall}}{\text{Precision} + \text{Recall}} \tag{6.3}$$

$$\text{IoU} = \frac{SP_C \cap SP_{GT}}{SP_T \cup SP_{GT}} \tag{6.4}$$

where SP_C is the number of true positive segmented pixels, SP_T is the total number of pixels that DCNN decided to segment, and SP_{GT} is the total number of ground truth pixels. In the case of IoU, $SP_C \cap SP_{GT}$ is the number of true positive pixels from the segmented image that intersects with the ground truth pixels, and $SP_T \cup SP_{GT}$ is the number of pixels that formed from the union of segmented pixels from the DCNN and the ground truth.

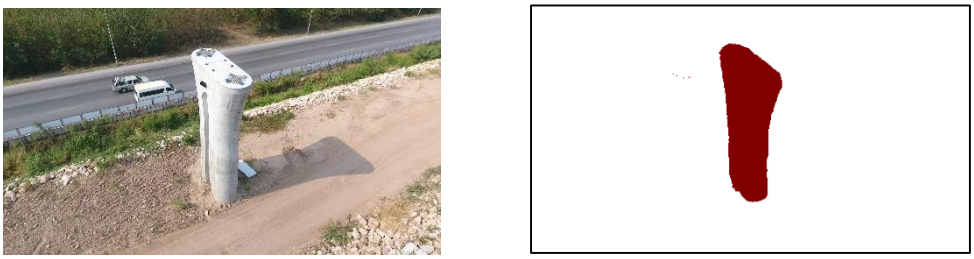
Moreover, 5-fold cross-validation (5-fold CV) was also utilized to quantify the performance of the DCNN on the train set. *k*-fold cross-validation is a method to evaluate the generalization of the network over the training images (German et al., 2012). The cross-validation was conducted by separating the images inside the train set into five groups. Then, each group was implemented as the validation set in each network training. After five times of training, all of the validated performances were averaged to find the values represented the performance of the network over the train set. **Table 6.2** Final performance of the DCNN in the proposed system over the training and test sets shows the final performance of the DCNN in this study.

Table 6.2 Final performance of the DCNN in the proposed system over the training and test sets

Parameters	Training set (5-fold CV)	Test set
Precision	97.78%	97.52%
Recall	97.62%	97.43%
F1 score	97.70%	97.48%
IoU	82.60%	83.45%

The performance of the network over train and test sets was satisfactory with the result of over 97% in precision, recall, and F1 score, in which the values over the test set are slightly lower than the ones from the test set, showing the good generalization between train and test sets. However, although the performance of the DCNN over a test set should normally be lower than the one from a train set, the network can perform on the test set better than the train set in the aspect of IoU. This situation can be

inferred as the data augmentation such as rotating and brightness adjusting complicated the training of the network and decreased the performance of the training set. On the other hand, all of the images in the test set were general and easy to understand. **Figure 6.3** shows an example of a segmented image.

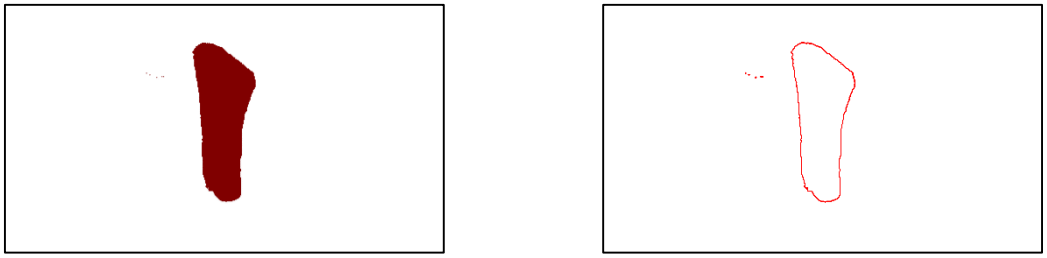


a) Original input image

b) Segmented image of **Figure 6.3a** by the DCNN (Red color shows the pier position)

Figure 6.3 Example of a segmented image

Next, the segmented images of each test image were processed to extract the boundary of the piers inside each image. Canny edge detector (Canny, 1986) was implemented to extract the boundary of the segmented images. Canny edge detector utilizes thresholding with hysteresis, therefore the maximum and minimum thresholds must be set. There was a recommendation to set the maximum threshold equals two times the minimum threshold (Canny, 1986). However, the minimum threshold varies by cases and needs to be examined in order to get the optimal value. Thus, the optimal thresh for extracting the boundary was also investigated. **Figure 6.4** shows an example of boundary extraction.

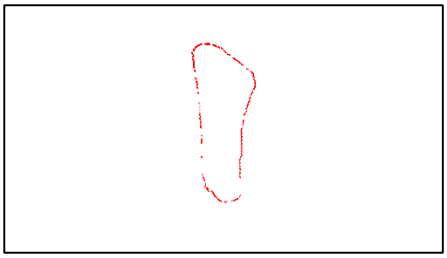


a) Segmented image from **Figure 6.3b**

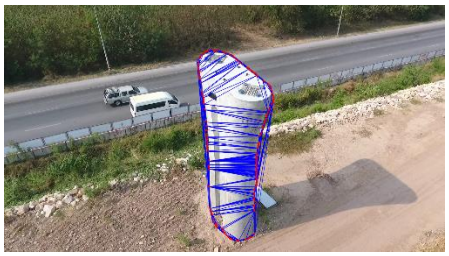
b) Boundary extraction from image **Figure 6.4a**

Figure 6.4 Example of an extracted boundary image

This boundary was further processed for Delaunay triangulation (Delaunay, 1934) by extracting the vertices on the boundary and utilizing these points to form the triangulation. **Figure 6.5** shows the result after applying the Delaunay triangulation. The blue color was the triangles that were formed based on these vertices. These triangles were implemented with the InCircle test proposed by Guibas and Stolfi (1985) to check which features from the feature detection step are in the boundary of the piers. In order to reduce the complexity from noises in the segmentation stage, the boundaries smaller than twenty square pixels were not processed in this step.



a) Vertices of the boundary from **Figure 6.4b**



b) Delaunay triangulation (blue lines) from vertices in **Figure 6.5a**

Figure 6.5 Example of Delaunay triangulation from vertices on the boundary of the object of interest

The concept of the InCircle test is for a set of points P = {P_1, P_2, P_3} as shown in **Figure 6.6**, if point Q is inside the circumcircle of P_1 to P_3, the determinant of these points will be a positive value. The formula for the determinant checking is shown in Equation 6.5.

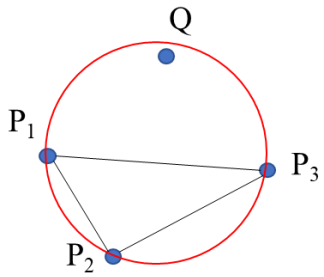


Figure 6.6 Point Q and a circumcircle made from Delaunay triangulation of P1 to P3

$$\begin{vmatrix} P_{1x} & P_{1y} & P_{1x}^2+P_{1y}^2 & 1 \\ P_{2x} & P_{2y} & P_{2x}^2+P_{2y}^2 & 1 \\ P_{3x} & P_{3y} & P_{3x}^2+P_{3y}^2 & 1 \\ Q_x & Q_y & Q_x^2+Q_y^2 & 1 \end{vmatrix} = \begin{vmatrix} P_{1x}-Q_x & P_{1y}-Q_y & (P_{1x}^2-Q_x^2)+(P_{1y}^2-Q_y^2) \\ P_{2x}-Q_x & P_{2y}-Q_y & (P_{2x}^2-Q_x^2)+(P_{2y}^2-Q_y^2) \\ P_{3x}-Q_x & P_{3y}-Q_y & (P_{3x}^2-Q_x^2)+(P_{3y}^2-Q_y^2) \end{vmatrix}$$

$$= \begin{vmatrix} P_{1x}-Q_x & P_{1y}-Q_y & (P_{1x}-Q_x)^2+(P_{1y}-Q_y)^2 \\ P_{2x}-Q_x & P_{2y}-Q_y & (P_{2x}-Q_x)^2+(P_{2y}-Q_y)^2 \\ P_{3x}-Q_x & P_{3y}-Q_y & (P_{3x}-Q_x)^2+(P_{3y}-Q_y)^2 \end{vmatrix} \quad (6.5)$$

The features that were able to be substituted into Equation 6.5 and make the determinant to be a positive value were added into a set of features that were inside the pier. **Figure 6.7** shows the comparison between all features detected by the SURF detector and the features inside the pier classified by the proposed system. The output of the system was validated with the manual editing process and the quantitative performance was also presented in the next chapter.

a) All features inside the original image b) Features inside the pier from the proposed system

Figure 6.7 Comparison between images of all SURF features
and SURF features inside the pier (red dots)

6.1.2 Results of the automated feature classification

The results from processing one hundred twenty-five images of the test image set into the proposed system with different thresholds of Canny edge detector are shown in **Table 6.3** compared with manual background removal. Please be noted that these values were gained from the average of one hundred processing times of each image inside the testing set in order to decrease the effect of computational errors during the testing.

Table 6.3 Results from the proposed system processed at different thresholds

Parameters	Detected contour (Contour)	Pier features (Point)	Boundary point extraction (Microsecond)	Feature classification using Delaunay triangulation (Microsecond)	Total computational time (Microsecond)
Manual background removal		1,863			
Threshold = 1	1,427	1,855	10.588	1.832	12.420
Threshold = 50	1,427	1,855	10.790	1.867	12.657
Threshold = 100	1,427	1,855	10.808	1.872	12.680
Threshold = 150	1,427	1,855	10.828	1.874	12.702
Threshold = 200	1,427	1,855	11.015	1.898	12.913
Threshold = 250	1,427	1,855	11.026	1.901	12.927

From **Table 6.3**, there are no differences for the numbers of detected contours by the Canny edge detector although the thresholds were set differently. The numbers of pier features are also the same in each threshold. The amount of the pier features processed by the proposed system is not much dissimilar (-0.43%) to the baseline of manual background removal to find the number of features inside the pier.

To further illustrate the trend of the computational times between different thresholds, **Figure 6.8** shows the comparison between these values. The computational time for feature classification using Delaunay triangulation has nearly no differences between the values of threshold. However, the required time for

boundary point extraction is different based on the number of the threshold. The bigger amounts of thresholds slow down the process, in which when combining the total computational time, the setting with higher numbers of threshold delays the finishing time of the proposed system as well.

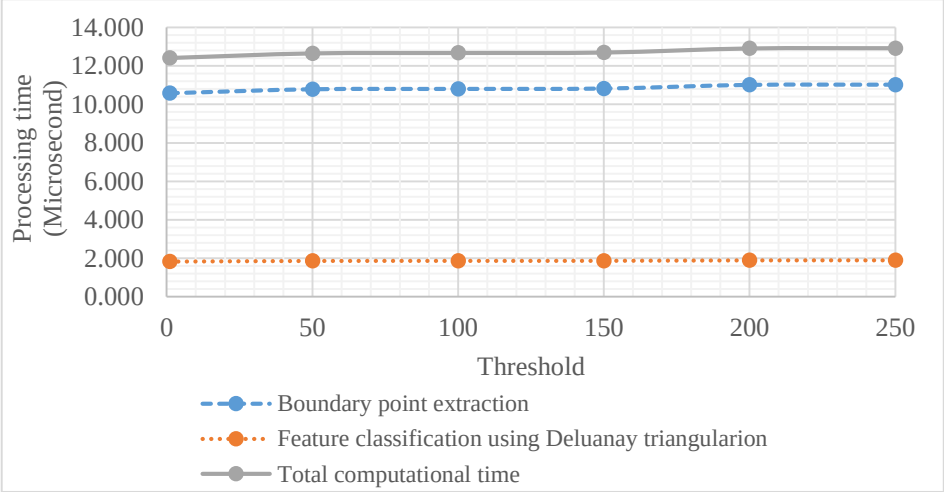


Figure 6.8 Comparison between processing time in each step at different thresholds

6.1.3 Discussion on the automated feature classification

The proposed system that the minimum threshold of the Canny edge detector was set at various values was evaluated with the manual background removal about the number of detected pier features. Furthermore, the computational times of each threshold were also collected and compared to validate the effect of thresholding with the processing of the proposed system.

According to the number of detected features, utilizing the proposed system did not complicate the feature detection of the original images. There is only a 0.43% difference between the detected features by the proposed system compared to the manual background removal of the original image. Moreover, removing the background to find the features inside the pier also has some disadvantages with the SURF detector as can be seen in **Figure 6.9**. Although the background of **Figure 6.9a** was removed completely, the SURF detector still detected some features outside the pier due to its detection procedure that separates images into various small batches through the entire image. Therefore, some features in the clean background can still be a candidate of the detection inside some image batches, in which this situation might be able to complicate the feature matching when processed into further steps of photogrammetric techniques and reduce the quality of the output.

For the computational time, setting a higher threshold resulted in taking more time to finish the entire process of feature classification. The reason is because of the process that the Canny algorithm used to detect edges inside an image. The detector applies the maximum threshold, which is two times higher than the minimum one, to find the edges that have an intensity lower than the maximum threshold. Next, the Canny algorithm tracks the detected edges along their lines until these edges have lower

intensity than the minimum threshold. By applying high minimum threshold values means that the algorithm has to track the intensity in a wider range and requiring more time to finish the computation. However, a segmented image from the DCNN does not have much difference in intensity, in which the setting of a higher threshold is unnecessary and resource-consuming as shown in **Table 6.3** that there is no difference in the number of detected contours although the thresholds were different. Thus, applying a low threshold is encouraged when utilizing a Canny edge detector with a segmented image.

a) Detected features from manual removal b) Detected features from the proposed system

Figure 6.9 Comparison of detected pier features (red dots) from the manual process and the proposed system

6.2 Automated point cloud classification

From the previous subchapter, it can be seen that implementing Delaunay triangulation with two-dimensional segmented images can classify the feature points as similar to the manual process. Therefore, the procedure in Subchapter 6.1 is extended to be implemented with the three-dimensional classification of point clouds.

6.2.1 Proposed methodology for automated point cloud classification

6.2.1.1 Overview of the novel point cloud classification based on digital images

In this section, a novel work process in order to classify three-dimensional point clouds from SfM based on their real-world images called Point cloud Classification based on image-based Instance Segmentation (PCIS) is presented. The research methodology was expanded from the first sections in this subchapter and is shown in **Figure 6.10**. First, a DCNN to automatically segment and create two-dimensional masks from input images of the SfM process was utilized. The original images were then processed into an SfM application to be solved for the camera parameters and construct a point cloud. Next, the mask images of each original image were placed at their respective positions inside the generated point cloud environment based on camera parameters solved by an SfM application. Each mask was further projected from their corresponding camera positions through their mask images onto

the point cloud to construct three-dimensional masks. Finally, the cloud points located inside the created three-dimensional masks were classified as the point cloud of the same class as the mask from DCNN based on results of Delaunay triangulation.

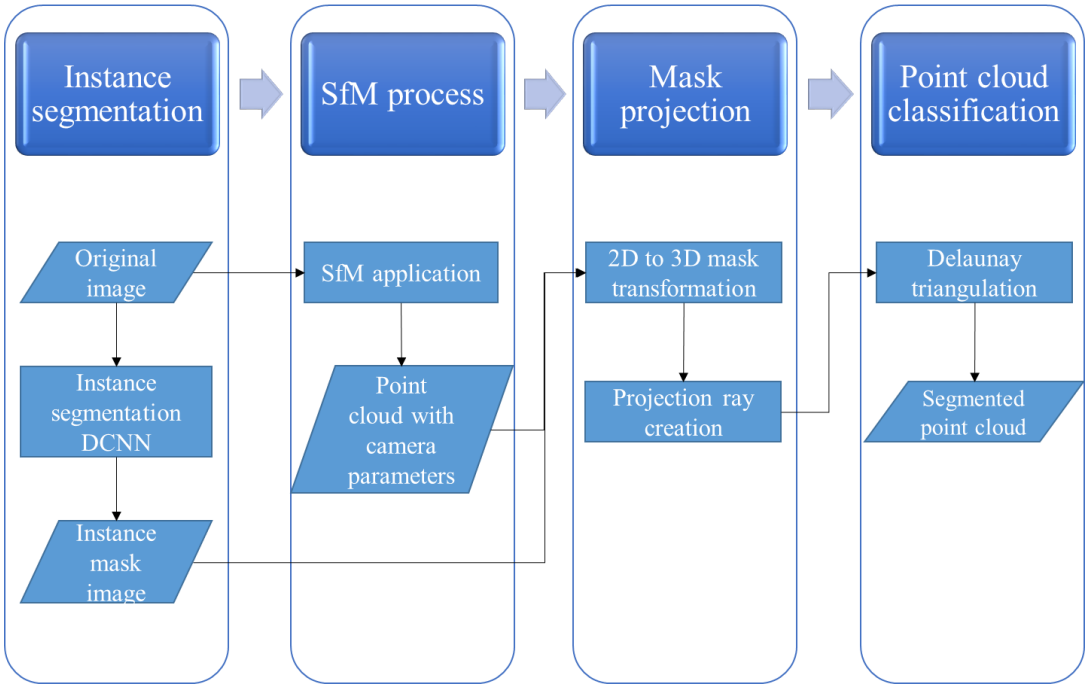


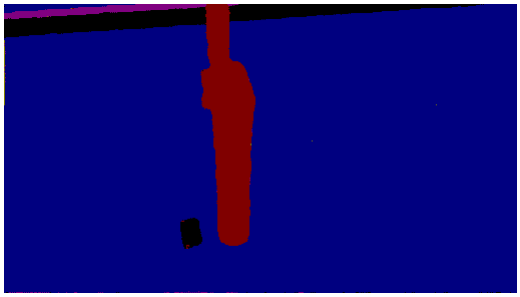
Figure 6.10 Proposed methodology for classification based on digital images

6.2.1.2 Image-based instance segmentation

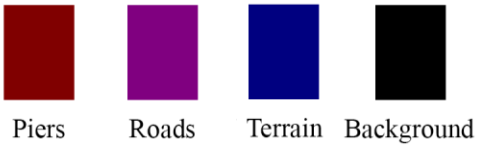
Although numerous studies have been implemented using semantic segmentation, which is a technique that allows a computer to categorize each pixel into classes (Garcia-Garcia et al., 2018), it was found that providing classes to each pixel is not enough to solve some problems in the field of civil engineering. To further illustrate this issue, U-Net (Ronneberger et al., 2015), which is a semantic segmentation network that is widely used in various studies (Abrams et al., 2019; Ibtehaz and Rahman, 2020; Izutsu et al., 2019), was trained with the same dataset as the validation sample. In **Figure 6.11**, it can be seen that from this view, the boundary between the front and back piers cannot be distinguished by the semantic segmented image from U-Net, resulting in difficulty when a site practitioner would like to automatically inspect only one pier in this picture.

Therefore, instance segmentation was implemented in this system. The difference between semantic segmentation and instance segmentation is that instance segmentation also separates each object inside the same class, which can solve the problem in this case by giving a clear boundary between the two piers. The network of PCIS was inspired by YOLACT presented by Bolya et al. (2019). Numerous

a) Original image



b) Semantic segmented image from U-Net (Ronneberger et al., 2015)



c) Color legend of the segmented image

Figure 6.11 Example of semantic segmentation result that cannot separate two piers

studies have utilized (Chang et al., 2020) and been inspired (Yang et al., 2020) by YOLACT. YOLACT was also often used as a benchmark for other studies on instance segmentation owing to its good performance (Chen et al., 2020; Peng et al., 2020; Wang et al., 2020). YOLACT separates instance segmentation into two sub-processes, which are rough prediction based on the protonet network to detect any instances in the image, and the precise detection of each detected instance based on a mask coefficient. Non-maximum suppression (Rothe et al., 2015) is implemented using the latter result to increase the confidence. The results that pass non-maximum suppression can then be linearly merged with the results from the rough prediction. 5-fold cross-validation (5-fold CV) (McLachlan et al., 2004) was also implemented to check whether generalization is achieved between the training set and test set.

6.2.1.3 SfM process

The outputs from this step are the point cloud and other camera parameters such as the rotation, translation, and focal length of each image. The point cloud is then exported in (x, y, z) coordinates in the reconstructed scene. Furthermore, the rotation matrix (R) is calculated based on the solved camera angles as the rotation around each axis. Three vector rotations equal to θ angle, each around x-, y-, and z-axis according to the right-hand rule, can be expressed as rotation matrices as in Equations 6.6–6.8:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{6.6}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{6.7}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.8}$$

To combine these matrices into one general matrix, matrix multiplication can be implemented as shown in Equation 6.9,

$$R_i = R_{xi}(\alpha)R_{yi}(\beta)R_{zi}(\gamma) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

$$= \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ \sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{6.9}$$

where R_i is the rotation matrix of image i and $R_{xi}(\alpha)$, $R_{yi}(\beta)$, and $R_{zi}(\gamma)$ are the rotation matrices around the x-, y-, and z-axis of each image for angles α, β, and γ, respectively. Next, the translation of each image is the distance between the origin of the point cloud environment and each image. In a three-dimensional scene, the translation can be written as a 1 x 3 matrix of the form as shown in Equation 6.10,

$$T_i = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{6.10}$$

where T_i is the translation matrix of image i and X_i, Y_i, and Z_i are the distances between the position of the image and the origin of the point cloud. Finally, the focal length is the distance between the camera sensor and the position at which the rays of light converge to form the sharpest image. To avoid unnecessary complexity in the photogrammetric techniques, the focal length needs to remain the same by not changing the zoom function of the camera capturing tool (Hackl et al., 2018).

6.2.1.4 Mask creation

After determining the parameters from each image, the values are used to create three-dimensional masks for further point cloud classification. This method was inspired by the pinhole camera model, which is the most basic model of a camera (Hartley and Zisserman, 2003). **Figure 6.12** shows the concept of a pinhole camera model. At the image plane of image i, where z equals the focal length (f), a point (x, y) inside this image plane can be matched with the cloud point (X, Y, Z) inside the global point cloud by using the theorem of similar triangles with a projection ray shown in Equation 6.11,

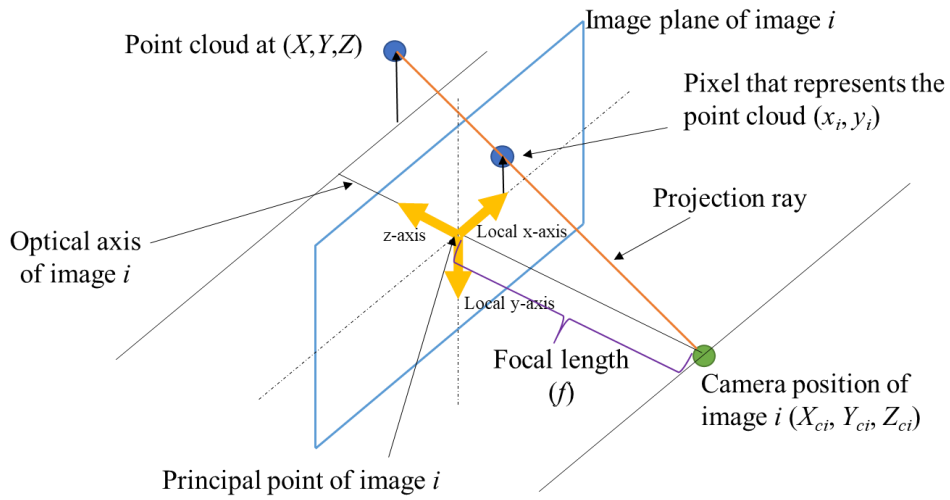


Figure 6.12 Concept of a pinhole camera model

$$(X, Y, Z) = \left(\frac{fx_i}{|Z - Z_{ci}|}, \frac{fy_i}{|Z - Z_{ci}|}, Z\right) \tag{6.11}$$

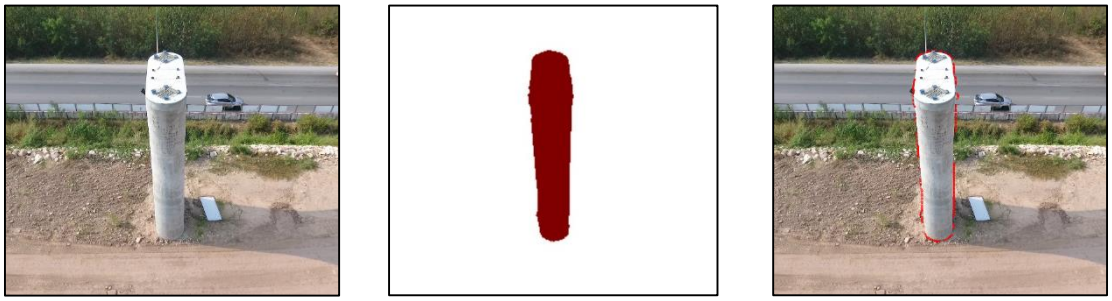
where $|Z - Z_{ci}|$ is the distance between the camera (X_{ci}, Y_{ci}, Z_{ci}) and the cloud point (X, Y, Z) in the global Z-axis, and (x_i, y_i) is the coordinate of the pixel inside the image plane with respect to the point cloud environment. Note that (x_i, y_i) has to be in the point cloud environment in order to get the correct position because currently, it is still inside the local two-dimensional pixel environment. To overcome this problem, the rotation and translation matrices output from the SfM process were utilized to transform this point in the image plane into the global point cloud environment by implementing matrix multiplication between the corresponding rotation matrix R_i and the pixel (x_i, y_i), then adding translation using the translation matrix T_i. The formula is shown in Equation 6.12,

$$\begin{bmatrix} X_{ip} \\ Y_{ip} \\ Z_{ip} \end{bmatrix} = R_i \begin{bmatrix} x_i \\ y_i \\ 0 \end{bmatrix} + T_i \tag{6.12}$$

where X_{ip}, Y_{ip}, and Z_{ip} are the position of the newly transformed pixel (x_i, y_i) as a three-dimension point in the X-, Y-, and Z-axis respectively, R_i is the 3 × 3 rotation matrix of image i, and T_i is the 1 × 3 translation matrix of image i. It can be seen that the parameter z_i is omitted because the coordinate (x_i, y_i) is inside a two-dimensional environment. Therefore, there is no value along the z-axis.

By implementing this proposed method to create masks for point cloud classification, a mask image from the DCNN in the first step is substituted for its corresponding original image from the SfM process at the image plane. A set of vertices from the mask are then used as the coordinates (x_i, y_i) inside Equation 6.12 to be transformed into three-dimensional coordinates. This process is performed continuously until every vertex is transformed. **Figure 6.13** shows an example of the mask vertices.

The red dots shown in **Figure 6.13c** are the vertices obtained by processing the mask image **Figure 6.13b** of the pier classification inside **Figure 6.13a**.



a) Original image b) Mask image c) Vertices from the mask image

Figure 6.13 Example of mask vertices

Although the positions of these vertices in a three-dimensional point cloud environment is obtained, these vertices cannot form projection rays because each vertex does not have another point to form a vector of the projection ray. Therefore, the origins of these rays are still needed, which is the position of the camera in the three-dimensional scene. From **Figure 6.12**, the distance between the camera and the image plane is known to be equaled to the focal length. Consequently, the camera position can also be converted into the three-dimensional environment by applying Equation 6.13, which is derived from Equation 6.13

$$\begin{bmatrix} X_{ci} \\ Y_{ci} \\ Z_{ci} \end{bmatrix} = R_i \begin{bmatrix} 0 \\ 0 \\ -f \end{bmatrix} + T_i \tag{6.13}$$

where the coordinate (X_{ci}, Y_{ci}, Z_{ci}) is the position of the camera of interest in the three-dimensional scene, R_i and T_i are the rotation and translation matrices of the corresponding image, and f is the focal length of the camera that captured the image. (0, 0, -f) is substituted for the position in two dimensions because the camera is known to be located behind the image plane at the focal length distance. Therefore, a negative focal length is used for the camera position transformation.

After the positions of the camera and vertices in the three-dimensional environment have been determined, the projection is performed by creating a vector using the camera position (X_{ci}, Y_{ci}, Z_{ci}) as the start point, the vertex of the mask boundary as the end point, and simply multiplying by a factor to extend the vector through the image plane to the point cloud scene. This factor can be fine-tuned or automatically adjusted based on the distance such as the ratio between the origin and the camera position, and the origin to the image plane. To further illustrate this, **Figure 6.14** shows the concept of the three-dimensional mask projection.

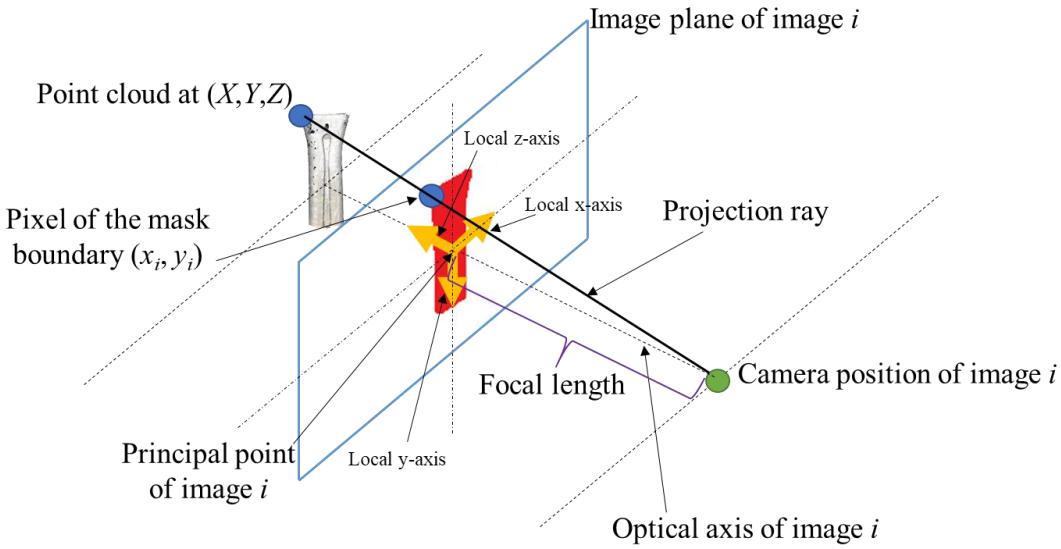
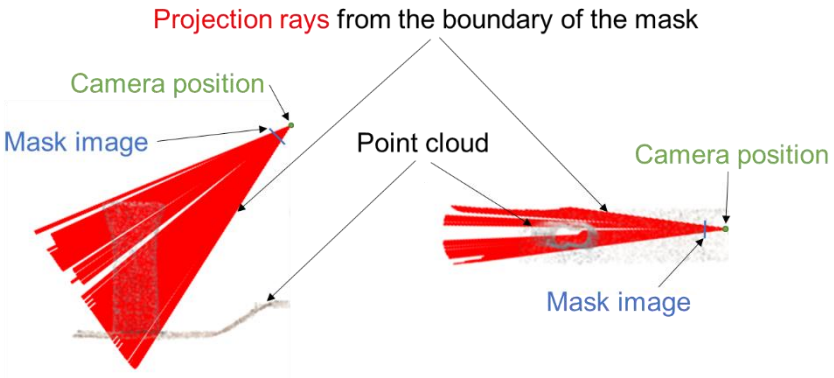


Figure 6.14 Concept of three-dimensional mask creation in PCIS

The mask projection process is performed iteratively until all of the vertices are used. The new camera is then selected and the steps are continued until all of the cameras are implemented. The output of this step is the three-dimensional masks projected from each camera to the point cloud. **Figure 6.15** shows an example of a three-dimensional mask from processing one camera with its corresponding mask image of an infrastructure pier.



a) Front view b) Top view

Figure 6.15 Example of a three-dimensional mask created by processing a camera and its corresponding mask image into PCIS

6.2.1.5 Point cloud classification

The final process in PCIS is to classify the point cloud based on the masks constructed in the previous step. The input to this step is the point cloud constructed from the SfM process and the three-dimensional masks. Each mask is used to check if the tested cloud points belong to the class segmented

by the two-dimensional instance segmentation. The pseudo-code of the classification system is presented in **Algorithm 6.1**. First, a set of masks is loaded into the system. The cloud points of interest are then processed into the system to check whether they are inside the mask based on Delaunay triangulation (Delaunay, 1934). If the testing cloud point is inside the mask, it is then added to the set of points detected by that mask. When all of the cloud points have been tested against the first mask, the next mask is used and all of the cloud points are processed again until all masks have been verified against every cloud point.

The concept of Delaunay triangulation was that when a set of point P is processed from P_1 to P_n to form a triangulation using the Delaunay algorithm, there are no other points P_i that can be inside the circumcircles constructed from three points of the same set P (Guibas and Stolfi, 1985). Delaunay triangulation has been widely used in computer vision techniques such as feature matching (Jiang and Jiang, 2019), boundary detection (Brie et al., 2016), and surface reestablishing (Mineo et al., 2019). A previous research (Saovana et al., 2020b) also implemented Delaunay triangulation to extract the two-dimension features inside a mask image with satisfactory results. In this study, the use of Delaunay triangulation is extended to implementation by a three-dimensional point cloud instead of simple two-dimensional features. the InCircle algorithm is utilized, which was first introduced by Guibas and Stolfi (1985), to check whether the cloud point of interest is inside the checking mask. PCIS implemented SciPy's Delaunay class and find_simplex function (Virtanen et al., 2020) to solve the three-dimensional InCircle problem.

The output from this step is the cloud points that were inside at least one of the three-dimensional projections from the mask images. **Figure 6.16** demonstrates an example of a classified point cloud that passes the projected mask testing. Although the cloud points that are located inside the masks can be determined already, there is a need to filter out cloud points that have the potential to be noise or unwanted features because they were accidentally detected by poor segmentation masks or incorrect camera poses. Therefore, filtering needs to be performed before the final classified point cloud can actually be utilized (Rastiveis et al., 2020).

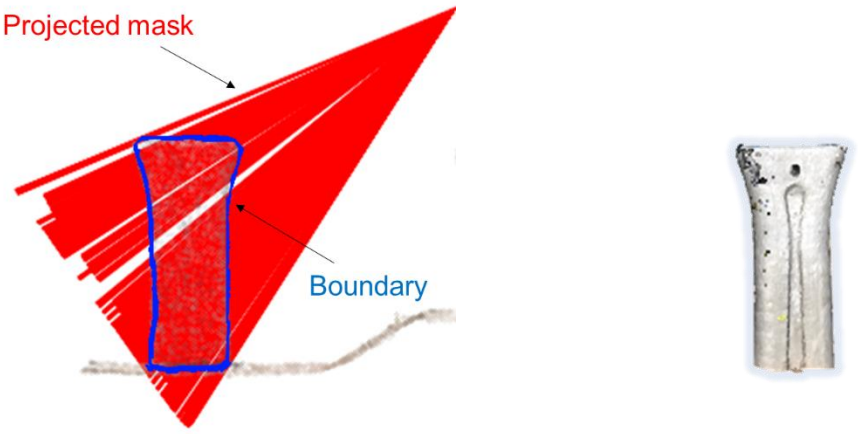
A confidence threshold checking was implemented inside PCIS to verify that the prospective cloud points have been properly classified by various masks. However, although filtering can increase the precision of a system, it can also decrease the recall rate. Therefore, in the validation step, various criteria is implemented to validate the performance of PCIS, namely, precision, recall, and F1-score.

Algorithm 6.1: Point cloud classification based on image masks

Input: Three-dimensional masks $M = \{M_i | i=1,2,3,\ldots,n\}$; Cloud points $C = \{C_i | i=1,2,3,\ldots,n\}$; Confidence threshold T

Output: A set of cloud points that are classified as the segmented object of interest $S=\{S_i | i=1,2,3,\ldots,n\}$

```
1. Load M and C
2. for Mi in M:
3.     Define a set S to temporary store the cloud points located inside Mi
4.     for Ci in C:
5.     #Check if Ci is in Mi with by processing into SciPy's Delaunay class
       and find_simplex function
6.         if Ci is inside Mi:
7.             Add Ci into S
8. Calculate the frequency of each Ci that appears in S
9. for Ci in S:
10.    if the frequency of Ci < T:
11.        Remove Ci
12.        else:
13.            Keep Ci as Si
```



a) Boundary of the object of interest b) Classified cloud points based on a projected mask

Figure 6.16 Example of the point cloud that passes the InCircle testing

6.2.2 Experimental validation for PCIS

To validate PCIS, images from a real case study were used. The case study was a civil infrastructure construction site because it was challenging for PCIS to work on a huge scene with numerous complexities, such as uncontrollable light and an untidy environment containing many objects with image features such as boulders and vegetation resulting in making the objects of interest, which were civil infrastructure piers, a minority point cloud of the scene.

The dataset was created by utilizing a drone, namely DJI Phantom 4, to capture multiple images focusing on a pier so that the effect of a lack of good mask images against another pier was further investigated. The details of the combination between training and test sets are shown in **Table 6.4** One hundred fifty images were utilized with an augmentation technique to increase the number of training data to one thousand images. The test set contained another one hundred images that the network had never seen when training.

Table 6.4 The information on the number of training and testing images

Parameter	Training set			Test set
	Original images	Augmented images	Total images	
Number of images	150	850	1,000	100

Moreover, to verify the generalization of the training and test sets, 5-fold CV was also implemented with the training data by separating the training data into five subsets and letting one of the subsets be an evaluation set in each training. Training was executed on a computer with an Intel® Core™ i7-8086K CPU @ 4.00GHz with 32.0 GB of RAM and an NVIDIA GeForce RTX 2080 Ti graphics card. The number of training iterations was set to sixty thousand to eliminate the risk of overfitting due to a low number of training data. The used batch size was equal to five. The learning rate, momentum, and decay rate were set to 0.001, 0.9, and 0.0005, respectively. The performance justification of the network followed the primary COCO validation style that utilizes the mean average precision (mAP) calculated from the threshold of intersection over union (IoU) from 0.5 to 0.95 in steps of 0.05. An example of the original image and the corresponding labeled image for training are shown in **Figure 6.17**. The five classes that were implemented in this study were piers, roads, vegetation, terrain, and background. Note that the object of interest in this study was only the pier class.

6.2.3 Results of PCIS

6.2.3.1 Instance segmentation result

After the training was finished, the performance of this network was evaluated. The performance was satisfactory, able to achieve 23.06 mAP from 5-class segmentation with 5-fold CV when it was trained and gained 23.02 mAP on the testing showing the generalization between the training and test sets even

a) Original image

b) Label for training in this study

c) Color legend

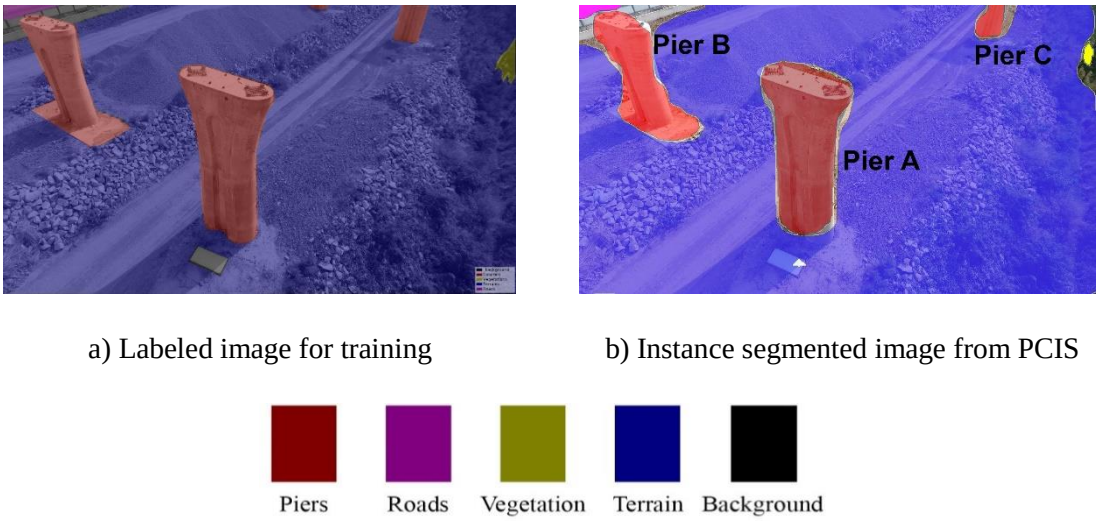
Figure 6.17 Example of the training image and its label with the color dictionary of the study

though it had never seen the test set before. **Figure 6.18** shows the labeled image for testing with its respective instance segmented image from the network. Note that the different shades of red have no meaning except for better illustration to show that this network can separate objects within the same class. It can be seen that most of the pixels were segmented correctly except only where there was overlap, and although this case has also been reported by other research (Chen et al., 2020), the performance of the YOLACT architecture is still far from disappointing and inspires new advances and developments (Chang et al., 2020; Yang et al., 2020).

6.2.3.2 Point cloud classification performance

The testing point cloud was constructed by processing the original images from the test set. The SfM application in this study was Agisoft Metashape (version 1.6.0, 64 bit) (Agisoft LLC, 2019). The used functions and settings were photo-alignment and dense point cloud creation. Photo-alignment was adjusted to ultra-high quality. On the other hand, the quality of dense point cloud creation was set at high to reduce the computational time. When the processing finished, 761,784 points had been created. The point cloud for the ground truth of the validation was separated into two samples, which were two piers inside the scene. Pier A was the focus of the image capture, with all of the images containing this pier. In contrast, Pier B appeared in approximately forty images only. **Table 6.5** shows a summary of the numbers of cloud points. Due to the lack of image features in civil infrastructure, particularly in wild scenes containing vegetation, boulders, and terrain, the percentage of cloud points corresponding to the structure is very low compared with the cloud points of the surroundings. This issue is very challenging for statistical point cloud classification because the target points can be easily neglected

and pin-point classification is needed in order to obtain high classification performance. The point cloud scene is also visualized in **Figure 6.19**.



a) Labeled image for training

b) Instance segmented image from PCIS

c) Color legend

Figure 6.18 Sample of a segmented image from PCIS compared with its respective labeled image for testing

Table 6.5 Summary of testing point cloud data

Samples	Number of cloud points	Percentage (%)
Pier A	11,985	1.57
Pier B	10,916	1.43
Background	738,883	96.99
Total	761,784	100.00

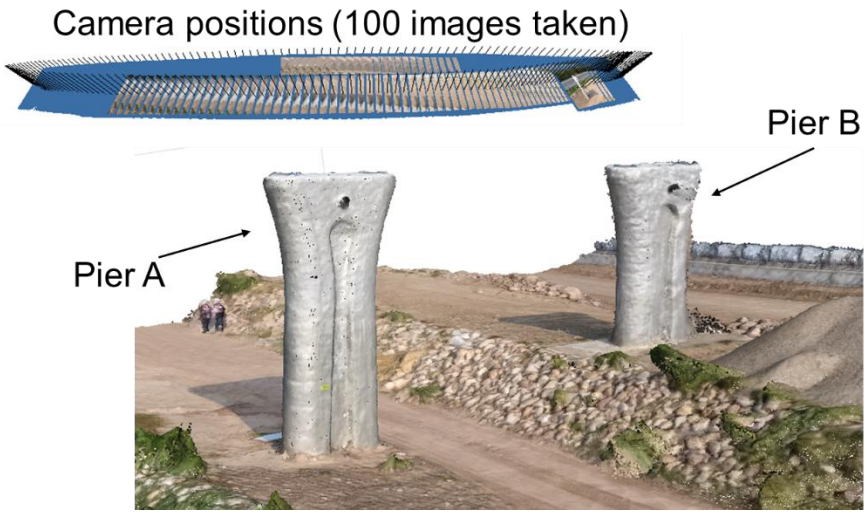


Figure 6.19 Testing point cloud with camera positions from Agisoft Metashape

Precision, recall, and F1-score were used to judge the performance of the PCIS network. Equations 6.14–6.16 show how these criteria were calculated:

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}} \tag{6.14}$$

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \tag{6.15}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.16}$$

where true positive is the number of correctly classified cloud points by the system, false positive is the number of cloud points that should not be classified but the system decided to, and false negative is the number of cloud points that should be classified by the system but did not get recognized.

The testing point cloud was processed into PCIS to determine the classification by using masks from the trained DCNN. After classification, filtering was performed to eliminate cloud points that had a lower detection frequency than various thresholds. These thresholds were called outlier removal percentages such that if the total number of masks that can detect a point cloud is lower than the specified percentage, the cloud point is not classified and removed from the final classification list as an outlier or noise of the process. For example, if the percentage is set to 5, then if there are one hundred mask images, a cloud point will be classified as the class of interest if there are at least five mask images that can detect this cloud point. Cloud points that have fewer than five masks detected are removed from the final classification list.

Figure 6.20 shows the results of the classification of Pier A, which was the main focus of the camera. The highest recall rate of PCIS over Pier A was very high at nearly 1.00, which means that 100% of the cloud points of Pier A were recognized by PCIS when the outlier removal percentage was set to lower than 10. **Figure 6.21** shows the classified final point cloud of Pier A at 10% outlier removal. For removal percentages higher than 10, the recall dropped due to the removal of some ground truth cloud points that had lower confidence.

In contrast to recall, the precision rate of PCIS without removal was lower, at about 0.66, and drastically increased as the removal percentage was increased due to the elimination of false positive cloud points. **Figure 6.22** shows the output of the classification at 20% outlier removal, which has the highest F1-score ratio of 0.96.

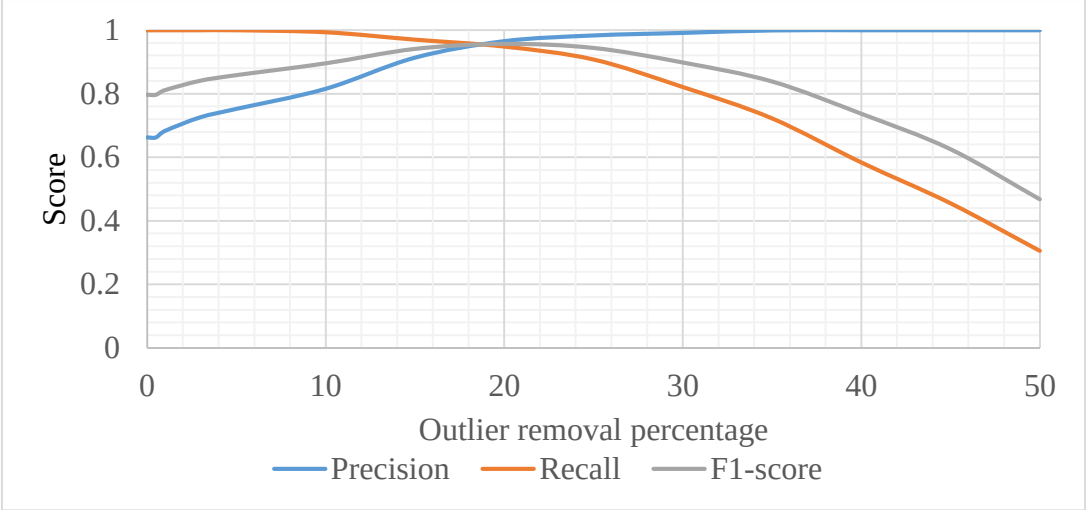


Figure 6.20 Performance graph of PCIS at various outlier removal percentages for Pier A classification

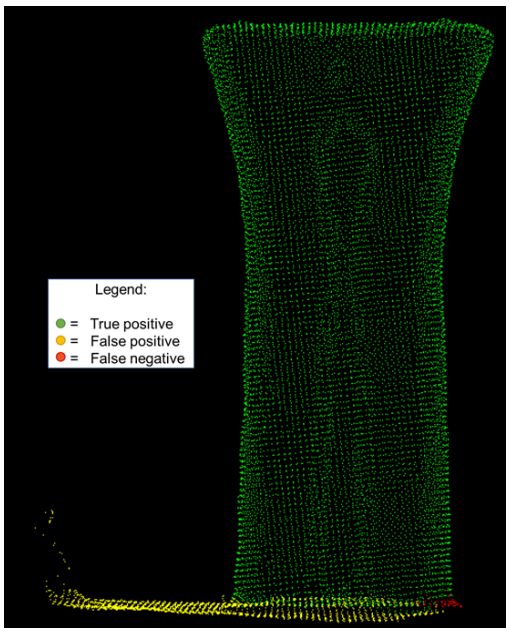


Figure 6.21 Final classified point cloud of Pier A at 10% outlier removal

To evaluate the effect of coverage deficiency, the Pier B sample was also tested with the system. Pier B appeared in only approximately forty images since it was not the focus of the image acquisition. A performance graph of the point cloud classification for Pier B is shown in **Figure 6.23**.

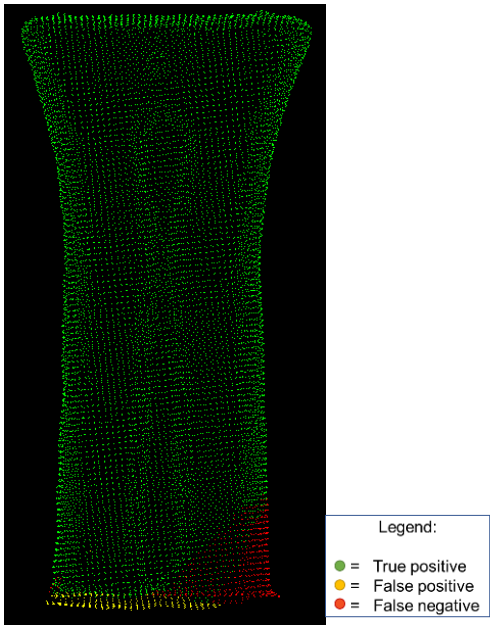


Figure 6.22 Final classified point cloud of Pier A at 20% outlier removal

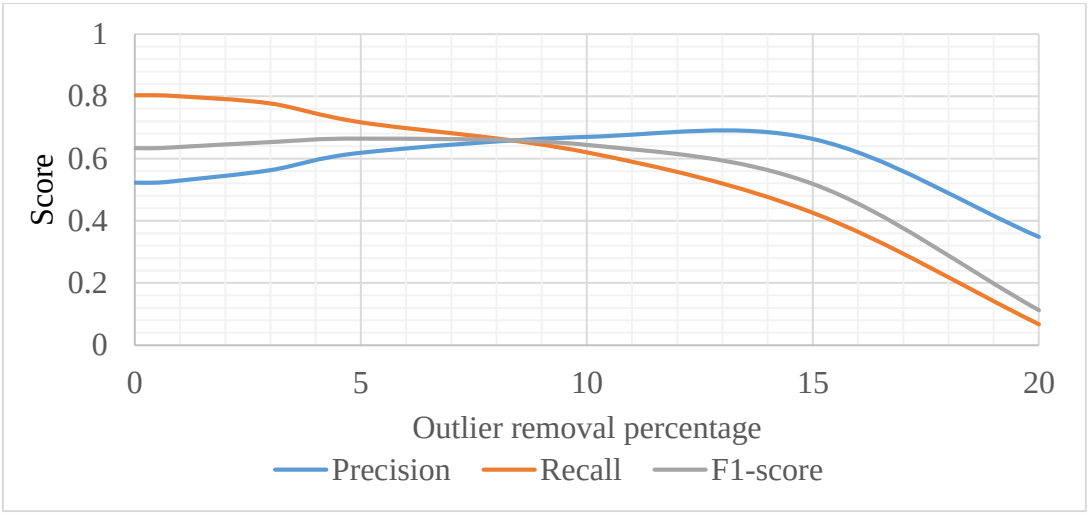


Figure 6.23 Performance graph of PCIS at various outlier removal percentage for Pier B classification

Although there was a lack of images, PCIS was still able to obtain 0.8 recall and 0.52 precision without removal. **Figure 6.24** shows the final output of Pier B at an 8% outlier removal threshold, which was the highest performance that PCIS can perform with this sample. The precision, recall, and F1-score at 8% outlier removal were about 0.68.

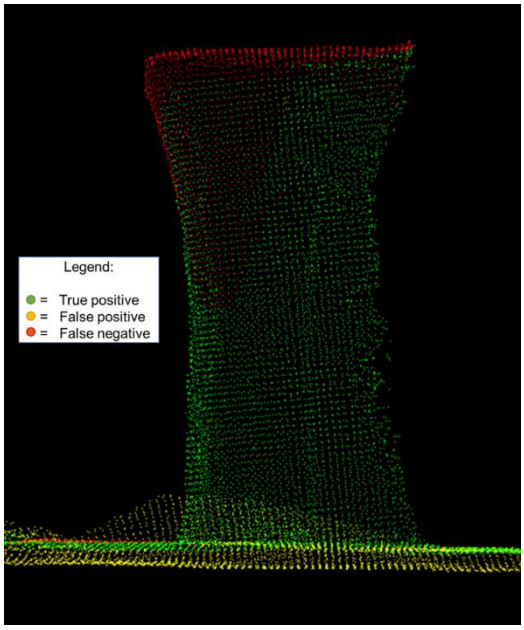


Figure 6.24 Final classified point cloud of Pier B at 8% outlier removal

6.2.4 Discussion on the automated classification of the point cloud from PCIS

After evaluation using Pier A and Pier B as test samples, the performance of PCIS on Pier A was found to be very satisfactory due to sufficient coverage of the mask images. From the graph performance of Pier A shown in **Figure 6.20** the peak performance of PCIS was at 20% of outlier removal. The overall performance of the system based on the F1-score is the highest at 0.96. The recall decreased because the final point cloud was over-filtered and too many true positive cloud points were removed, which increased the precision as shown in **Figure 6.22**. It can be seen that there were nearly no false positive cloud points left inside the classified point cloud. Nevertheless, there was an increase in the false negative cloud points at the bottom of the pier. Note that in **Figure 6.21**, which is the resulting point cloud from 10% outlier removal, the last outlier removal percentage that kept the recall nearly at 1.00, although there were some false positive cloud points as highlighted in yellow that hinder the performance by decreasing the precision, the cloud points of the pier were successfully detected with high detail as shown in green. There were very few ground truth cloud points that were not detected and marked as false negative cloud points (red) at the base of the pier that will hardly affect further processing.

For Pier B, on the other hand, the PCIS peak performance was at 8% of the outlier removal as shown in **Figure 6.23**. The highest performance of PCIS was still considerably high at an F1-score equal to 0.66 at 8% outlier removal. **Figure 6.24** shows that there were some false negative cloud points at the top of the pier and there were also some false positive cloud points in the gravel near the base of the pier. This shows that images from different views are very important for the success of PCIS because

if there are more images from different points of view, it would be possible to support projection angels for more valid cloud points and suppress these obstacles out of the final point clouds.

Furthermore, the performance graphs also show that having more images from different views can increase the level of outlier removal percentage before the classification performance decreases. The amount of adjustable removal percentage can provide leverage for the practitioners to justify their final point cloud output. If the detail of the object of interest is not strictly necessary, the percentage of outlier removal can be increased to get a tighter classification with higher precision, which can save time in the removal of unwanted cloud points while still achieving high-quality classified cloud points. However, if the final point cloud needs to be substantial, the removal percentage can be reduced to gain more recall and have more cloud points classified. With sufficient image coverage, PCIS can provide critical relief for practitioners removing noise and classifying the point cloud, which is considered to be a tedious, labor-intensive, and time-consuming task, especially when dealing with the huge scenes of civil infrastructure projects.

6.3 Conclusion of this chapter

Point clouds are widely used for visualization, particularly in civil engineering such as for inspection and progress tracking. They can be constructed by using photogrammetric techniques such as SfM, which is the process of using overlapping between images to solve for camera parameters and produce a point cloud based on the detected image features. However, these point clouds need to be classified before utilization, at which point cloud classification is considered to be labor-intensive and time-consuming. Numerous studies have therefore tried to alleviate this problem. Although one of the main solutions is to use deep learning to classify these point clouds, the use of deep learning is still far from maturity due to various complexities such as the lack of point cloud training data and unrealistic synthetic images.

In this study, digital images from a real scene were used based on the fact that images are currently captured as part of the daily work at a construction site. These images were first utilized as training data for the DCNN, which was inspired by YOLACT (Bolya et al., 2019). The test images were then processed into Agisoft Metashape (Agisoft LLC, 2019) to construct a point cloud of the testing scene. Next, these test images were used as the input for the DCNN for instancing segments and producing mask images. The mask images were further utilized as the image plane for projection in the point cloud scene based on the solved camera poses to create three-dimensional masks from camera locations through the mask images to the point cloud. Finally, cloud points located inside more of these masks than the outlier removal percentage were selected as the final classified cloud points of the object of interest.

The examples used in the validation experiment were a pair of piers from a civil infrastructure construction project. Only one of the piers is the main focus of the image capturing in order to validate the effect of coverage deficiency. The results found that with sufficient view coverage of the object of interest, PCIS can correctly classify the point cloud at an F1-score of 0.96, which is extremely high for a challenging scene that is usually unordered, huge, and full of image feature-rich elements such as vegetation and boulders, which can complicate the automatic point cloud classification. Moreover, this methodology is very simple and able to be applied to other fields of work.

Chapter 7
Conclusion

7.1 Summary

A point cloud is a visualization tool that is currently widely used in the computer vision aspect. It can show the existence at the scanned time and when comparing between each discrete-time, the difference can be inspected. These point clouds can be constructed from photogrammetric techniques such as SfM that uses digital images from various views as inputs and let feature detector and descriptor algorithms match the image features inside the overlapping between input images. Another output of the SfM is the camera poses of each image comparing their direction with the constructed point cloud scene.

Nevertheless, the feature detection and description, which is the fundamental process of photogrammetric techniques, is prone to error and needs to be validated before the implementation, which has not yet to be seen with the infrastructure scope. Due to the reasons that the infrastructure scene is usually complex, having a gigantic size with numerous feature-rich texture components such as boulders, vegetation, and terrain, resulting in the excessive unwanted point cloud of the scene.

These feature points and point clouds need to be classified before the utilization, which if implemented manually, is cumbersome and labor-intensive. Therefore, some studies have implemented the automated point cloud classification by using a handcrafted classifier or deep learning from three-dimensional point clouds. However, these handcrafted classifiers require an expert to fine-tuning the parameters inside and deep learning require numerous point clouds to be used as training data.

Therefore, an evaluation to find the suitable feature detector and descriptor algorithm for infrastructure scope was conducted with the development of a novel image feature and point cloud classification

system that does not implement handcrafted classifiers and point cloud training data. The result of the feature detector and descriptor algorithm evaluation shows that the SURF algorithm can perform well with every scope of feature matching. Meanwhile, removing the unwanted features out of the input image of an SfM application can raise the number of feature matching inside the ROI, hence generate more high confidence cloud points in the final point cloud output. The performance of the presented automatic unwanted feature removal and feature classification based on semantic segmentation were also promising to provide the results nearly equal to the manual removal.

The proposed system for point cloud classification based on instance segmentation of real-world digital images called PCIS also provided satisfactory result with up to 0.96 F1-Score, confirming the usability of two-dimensional real-world images, which can be acquired easily from a daily work basis, to be used as the training data for deep learning. This process can relieve the bottleneck problem of deep learning due to the lack of point cloud training data in the outfield environment.

7.2 Research Contributions

This research contributes to the further understanding of the photogrammetric technique with infrastructure scope, from the most fundamental aspect, which is image feature, to the advanced work process of point cloud classification based on deep learning.

This study presents the quantitative performance evaluations of feature detector and descriptor algorithms when implemented with infrastructure scope that has not yet to be seen. It further investigates the effect of unwanted features inside the infrastructure scene with feature matching and point cloud generation. Finally, it proposes the PCIS system to automatically classify the image features and point clouds based on semantic and instance segmentation, which can relieve the traditional method of classification that has to use handcraft algorithms or a gigantic amount of point cloud data to train the network.

All experiments in this research deliver in-depth quantitative comparison between the traditional and the newly proposed method in order to understand the effect and validate the usability of the novel work processes such as unwanted feature removal based on semantic segmentation and point cloud classification using on-site digital images as training data. Although the proposed system was evaluated from case studies, it can be applied to other structures by forming the new input data, training the network, and utilizing the system with the new pre-trained knowledge. Furthermore, the proposed system can release the workload of practitioners such as unwanted feature removal and point cloud classification by automated the removal and classification based on the trained data inside the network of the PCIS system.

7.3 Limitations and future research

The proposed methods have some limitations that should be addressed in future studies. The limitations of each method are highlighted as follows.

The limitations of the unwanted feature removal are that first, the number of images with good overlapping in the ROI is crucial to the success of unwanted-feature removal because the number of features in the processed images is already low. If the number of overlapping images is not high enough, the image features might not be appropriately matched and the images might not be aligned. Due to this reason, in case the image capturing cannot be done properly for the entire site at once, it is better to split the process into smaller parts of the site or individual piers for the better reconstruction. The second limitation is that the user cannot intervene with the decision of the system when it decides which pixels belong to which class due to the black-box nature of deep learning. The system might remove parts of the ROI if sufficient training and validation have not been performed. The final limitation is that the developed methodology was tested on only two samples, which although the concept of utilizing semantic segmentation to segment the unwanted-features out of the SfM input is very general and applicable to any shapes of infrastructures if the data for training is available, it should be further verified with additional samples to increase its reliability and generalization.

Next, the limitations of PCIS are that view coverage of the object of interest plays a vital role in the success of the classification, and although the concept of using DCNN to support the work process was implemented, DCNN usually works in a black-box manner, rejecting intervention from the user, and the quality of the performance is directly influenced by the training. The architecture of DCNN can also affect the performance of the system such as what happened in the network where it could not properly segment the border between detected objects. Nevertheless, the classification based on these mask images was still satisfactory, which means that utilizing networks with a similar architecture and performance to this proposed method may also be fine. Moreover, more samples should be implemented to verify the usability of PCIS such as the structures with multiple components and images that do not cover the entire structure.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2005). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. https://doi.org/10.1080/09548980500300507

Abrams, J. F., Vashishtha, A., Seth, T. W., Nguyen, A., Mohamed, A., Wieser, S., Kuijper, A., Wilting, A., and Mukhopadhyay, A. (2019). Habitat-Net: Segmentation of habitat images using deep learning. *Ecological Informatics*, *51*(May), 121–128. https://doi.org/10.1101/483222

Agisoft LLC. (2019). *Agisoft Metashape User Manual: Professional Edition, Version 1.6*. https://www.agisoft.com/pdf/metashape-pro_1_6_en.pdf

Agnisarman, S., Lopes, S., Chalil Madathil, K., Piratla, K., and Gramopadhye, A. (2019). A survey of automation-enabled human-in-the-loop systems for infrastructure visual inspection. *Automation in Construction*, *97*(April 2018), 52–76. https://doi.org/10.1016/j.autcon.2018.10.019

Alcantarilla, P. F., Bartoli, A., and Davison, A. J. (2012). KAZE Features. *ECCV 2012, Part VI*, 214–227. https://doi.org/10.1007/978-3-642-33783-3_16

Alcantarilla, P. f., Nuevo, J., and Bartoli, A. (2013). Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. *Proceedings of the British Machine Vision Conference 2013*, 13.1-13.11. https://doi.org/10.5244/C.27.13

Ali, H., and Whitehead, A. (2014). Modern to Historic Image Matching: ORB / SURF an Effective Matching Technique. *International Conference on Computers and Their Applications*, 321–326.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *American Statistician*, *46*(3), 175–185. https://doi.org/10.1080/00031305.1992.10475879

Amirkolaee, H. A., and Arefi, H. (2019). Height estimation from single aerial images using a deep

convolutional encoder-decoder network. *ISPRS Journal of Photogrammetry and Remote Sensing*, *149*(July 2018), 50–66. https://doi.org/10.1016/j.isprsjprs.2019.01.013

Anil, E. B., Tang, P., Akinci, B., and Huber, D. (2013). Deviation analysis method for the assessment of the quality of the as-is Building Information Models generated from point cloud data. *Automation in Construction*, *35*, 507–516. https://doi.org/10.1016/j.autcon.2013.06.003

Antonarakis, A. S., Richards, K. S., and Brasington, J. (2008). Object-based land cover classification using airborne LiDAR. *Remote Sensing of Environment*, *112*(6), 2988–2998. https://doi.org/10.1016/j.rse.2008.02.004

Apollonio, F. I., Ballabeni, A., Gaiani, M., and Remondino, F. (2014). Evaluation of feature-based methods for automated network orientation. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *40*(5), 47–54. https://doi.org/10.5194/isprsarchives-XL-5-47-2014

Arief, H. A. ari, Indahl, U. G., Strand, G. H., and Tveite, H. (2019). Addressing overfitting on point cloud classification using Atrous XCRF. *ISPRS Journal of Photogrammetry and Remote Sensing*, *155*(July), 90–101. https://doi.org/10.1016/j.isprsjprs.2019.07.002

Balta, H., Velagic, J., Bosschaerts, W., De Cubber, G., and Siciliano, B. (2018). Fast Statistical Outlier Removal Based Method for Large 3D Point Clouds of Outdoor Environments. *IFAC-PapersOnLine*, *51*(22), 348–353. https://doi.org/10.1016/j.ifacol.2018.11.566

Bartoli, A., and Sturm, P. (2004). Non-Linear Estimation of the Fundamental Matrix With Minimal Parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Institute of Electrical and Electronics Engineers*, *26*, 426–432. https://hal.archives-ouvertes.fr/hal-00092887

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, *110*(3), 346–359. https://doi.org/10.1016/j.cviu.2007.09.014

Behnam, A., Wickramasinghe, D. C., Ghaffar, M. A. A., Vu, T. T., Tang, Y. H., and Isa, H. B. M. (2016). Automated progress monitoring system for linear infrastructure projects using satellite remote sensing. *Automation in Construction*, *68*, 114–127. https://doi.org/10.1016/j.autcon.2016.05.002

Bian, J., Lin, W. Y., Matsushita, Y., Yeung, S. K., Nguyen, T. D., and Cheng, M. M. (2017). GMS: Grid-based motion statistics for fast, ultra-robust feature correspondence. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2828–2837. https://doi.org/10.1109/CVPR.2017.302

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2019). YOLACT: Real-time instance segmentation. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 9156–9165. https://doi.org/10.1109/ICCV.2019.00925

Bosché, F. (2010). Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. *Advanced Engineering Informatics*, *24*(1), 107–118. https://doi.org/10.1016/j.aei.2009.08.006

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). Training Algorithm Margin for Optimal Classifiers. *COLT '92: Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144–152.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Bradski, G., and Kaehler, A. (2008). *Learning OpenCV* (M. Loukides (ed.); First Edit). O'REILLY. https://doi.org/10.1109/MRA.2009.933612

Braun, Alex, and Borrmann, A. (2019). Combining inverse photogrammetry and BIM for automated labeling of construction site images for machine learning. *Automation in Construction*, *106*(January), 102879. https://doi.org/10.1016/j.autcon.2019.102879

Braun, Alexander, Tuttas, S., Borrmann, A., and Stilla, U. (2015). A concept for automated construction progress monitoring using BIM-based geometric constraints and photogrammetric point clouds. *Journal of Information Technology in Construction*, *20*(November 2014), 68–79.

Braun, Alexander, Tuttas, S., Stilla, U., and Borrmann, A. (2018). Process- and computer vision-based detection of as-built components on construction sites. *Proceedings of the 35th International Symposium on Automation and Robotics in Construction and Mining*, 7. https://publications.cms.bgu.tum.de/2018_braun_isarc.pdf

Brie, D., Bombardier, V., Baeteman, G., and Bennis, A. (2016). Local surface sampling step estimation for extracting boundaries of planar point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *119*, 309–319. https://doi.org/10.1016/j.isprsjprs.2016.06.006

Brilakis, I., Fathi, H., and Rashidi, A. (2011). Progressive 3D reconstruction of infrastructure with videogrammetry. *Automation in Construction*, *20*(7), 884–895. https://doi.org/10.1016/j.autcon.2011.03.005

Brilakis, I., Park, M. W., and Jog, G. (2011). Automated vision tracking of project related entities. *Advanced Engineering Informatics*, *25*(4), 713–724. https://doi.org/10.1016/j.aei.2011.01.003

Brilakis, I., Soibelman, L., and Shinagawa, Y. (2005). Material-Based Construction Site Image Retrieval. *Journal of Computing in Civil Engineering*, *19*(4), 341–355. https://doi.org/10.1061/(ASCE)0887-3801(2005)19:4(341)

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). BRIEF: Binary robust independent elementary features. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-642-15561-1_56

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*, *No.6*, 679–698. https://doi.org/10.1109/ASICON.2011.6157287

Carbonneau, P. E., and Dietrich, J. T. (2017). Cost-effective non-metric photogrammetry from consumer-grade sUAS: implications for direct georeferencing of structure from motion photogrammetry. *Earth Surface Processes and Landforms*, *42*(3), 473–486. https://doi.org/10.1002/esp.4012

Carrivick, J. L., Smith, M. W., and Quincey, D. J. (2016). *Structure from Motion in the Geosciences* (1st ed.). Wiley-Blackwell.

Chang, M.-C., Zhao, G., Pandey, A. K., Pulver, A., and Tu, P. (2020). Railcar Detection, Identification and Tracking for Rail Yard Management. *Image Processing (ICIP) 2020 IEEE International Conference*, 2271–2275. https://doi.org/10.1109/icip40778.2020.9190763

Chen, H., Sun, K., Tian, Z., Shen, C., Huang, Y., and Yan, Y. (2020). Blendmask: Top-down meets bottom-up for instance segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8570–8578. https://doi.org/10.1109/CVPR42600.2020.00860

Chen, J., Kira, Z., and Cho, Y. K. (2019). Deep Learning Approach to Point Cloud Scene Understanding for Automated Scan to 3D Reconstruction. *Journal of Computing in Civil Engineering*, *33*(4), 04019027. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000842

Cheng, J. C. P., and Wang, M. (2018). Automated detection of sewer pipe defects in closed-circuit television images using deep learning techniques. *Automation in Construction*, *95*(June), 155–171. https://doi.org/10.1016/j.autcon.2018.08.006

Cowan, B., Imanberdiyev, N., Fu, C., Dong, Y., and Kayacan, E. (2017). A performance evaluation of detectors and descriptors for UAV visual tracking. *2016 14th International Conference on Control, Automation, Robotics and Vision, ICARCV 2016*. https://doi.org/10.1109/ICARCV.2016.7838649

Czerniawski, T., Sankaran, B., Nahangi, M., Haas, C., and Leite, F. (2018). 6D DBSCAN-based segmentation of building point clouds for planar object classification. *Automation in Construction*, *88*(December 2017), 44–58. https://doi.org/10.1016/j.autcon.2017.12.029

Dawood, T., Zhu, Z., and Zayed, T. (2018). Computer Vision–Based Model for Moisture Marks Detection and Recognition in Subway Networks. *Journal of Computing in Civil Engineering*, *32*(2), 04017079. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000728

Delaunay, B. (1934). Sur la sphère vide. *Bulletin de l'Académie Des Sciences de l'URSS, Classe Des Sciences Mathématiques et Naturelles*, *6*, 793–800.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. *CVPR09*.

Domokos, C., Kato, Z., and Francos, J. M. (2008). Parametric estimation of affine deformations of binary images. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, *3*, 889–892. https://doi.org/10.1109/ICASSP.2008.4517753

Fang, Y., Chen, J., Cho, Y. K., and Zhang Peiyao. (2016). A Point Cloud-Vision Hybrid Approach for 3D Location Tracking of Mobile Construction Assets. *ISARC Proceedings*, *Isarc*, 613–620. https://doi.org/10.22260/ISARC2016/0074

Fathi, H., and Brilakis, I. (2011). Automated sparse 3D point cloud generation of infrastructure using its distinctive visual features. *Advanced Engineering Informatics*, *25*(4), 760–770. https://doi.org/10.1016/j.aei.2011.06.001

Fathi, H., and Brilakis, I. (2013). A videogrammetric as-built data collection method for digital fabrication of sheet metal roof panels. *Advanced Engineering Informatics*, *27*(4), 466–476. https://doi.org/10.1016/j.aei.2013.04.006

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(4), 594–611. https://doi.org/10.1109/TPAMI.2006.79

Fischler, M. A., and Bolles, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, *24*(6), 381–395. https://doi.org/10.1145/358669.358692

Fu, K., Chang, Z., Zhang, Y., Xu, G., Zhang, K., and Sun, X. (2020). Rotation-aware and multi-scale convolutional neural network for object detection in remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, *161*(January), 294–308. https://doi.org/10.1016/j.isprsjprs.2020.01.025

Furukawa, Y., and Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(8), 1362–1376. https://doi.org/10.1109/TPAMI.2009.161

Gao, T., Akinci, B., Ergan, S., and Garrett, J. (2015). An approach to combine progressively captured point clouds for BIM update. *Advanced Engineering Informatics*, *29*(4), 1001–1012. https://doi.org/10.1016/j.aei.2015.08.005

Garcia-Garcia, A., Gomez-Donoso, F., Garcia-Rodriguez, J., Orts-Escolano, S., Cazorla, M., and Azorin-Lopez, J. (2016). PointNet: A 3D Convolutional Neural Network for real-time object class recognition. *Proceedings of the International Joint Conference on Neural Networks*, *2016-Octob*, 1578–1584. https://doi.org/10.1109/IJCNN.2016.7727386

Garcia-Garcia, Alberto, Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Martinez-Gonzalez, P., and Garcia-Rodriguez, J. (2018). A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing Journal*, *70*, 41–65. https://doi.org/10.1016/j.asoc.2018.05.018

German, S., Brilakis, I., and Desroches, R. (2012). Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments. *Advanced Engineering Informatics*, *26*(4), 846–858. https://doi.org/10.1016/j.aei.2012.06.005

Golparvar-Fard, M., Balali, V., and de la Garza, J. M. (2015). Segmentation and Recognition of Highway Assets Using Image-Based 3D Point Clouds and Semantic Texton Forests. *Journal of Computing in Civil Engineering*, *29*(1), 04014023. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000283

Golparvar-Fard, M., Peña-Mora, F., Arboleda, C. A., and Lee, S. (2009). Visualization of Construction Progress Monitoring with 4D Simulation Model Overlaid on Time-Lapsed Photographs. *JOURNAL OF COMPUTING IN CIVIL ENGINEERING*, *23*(6), 391–404. https://doi.org/10.1061/(ASCE)0887-3801(2009)23

Golparvar-Fard, M., Peña-Mora, F., and Savarese, S. (2015). Automated Progress Monitoring Using Unordered Daily Construction Photographs and IFC-Based Building Information Models. *Journal of Computing in Civil Engineering*, *29*(1), 04014025. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000205

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org/

Guibas, L., and Stolfi, J. (1985). Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi. *ACM Transactions on Graphics (TOG)*, *4*(2), 74–123. https://doi.org/10.1145/282918.282923

Hackel, T., Wegner, J. D., and Schindler, K. (2017). Joint classification and contour extraction of large 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *130*, 231–245. https://doi.org/10.1016/j.isprsjprs.2017.05.012

Hackl, J., Adey, B. T., Woźniak, M., and Schümperlin, O. (2018). Use of Unmanned Aerial Vehicle Photogrammetry to Obtain Topographical Information to Improve Bridge Risk Assessment. *Journal of Infrastructure Systems*, *24*(1), Content ID 04017041. https://doi.org/10.1061/(ASCE)IS.1943-555X.0000393

Hamledari, H., McCabe, B., Davari, S., and Shahi, A. (2017). Automated Schedule and Progress Updating of IFC-Based 4D BIMs. *Journal of Computing in Civil Engineering*, *31*(4), 04017012. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000660

Harris, C., and Stephens, M. (1988). A Combined Corner and Edge Detector. *Alvey Vision Conference*,

50. https://doi.org/10.5244/c.2.23

Hartley, R., and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press.

Hartley, R., Zisserman, A., Hartley, R., and Zisserman, A. (2011). Epipolar Geometry and the Fundamental Matrix. In *Multiple View Geometry in Computer Vision* (pp. 239–261). https://doi.org/10.1017/cbo9780511811685.014

Hidaka, N., Michikawa, T., Motamedi, A., Yabuki, N., and Fukuda, T. (2018). Polygonization of point clouds of repetitive components in civil infrastructure based on geometric similarities. *Automation in Construction*, *86*(August 2017), 99–117. https://doi.org/10.1016/j.autcon.2017.10.014

Hietanen, A., Lankinen, J., Kämäräinen, J. K., Buch, A. G., and Krüger, N. (2016). A comparison of feature detectors and descriptors for object class matching. *Neurocomputing*, *184*, 3–12. https://doi.org/10.1016/j.neucom.2015.08.106

Ho, T. K. (1995). Random decision forests. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, *1*, 278–282. https://doi.org/10.1109/ICDAR.1995.598994

Hu, Q., Luo, J., Hu, G., Duan, W., and Zhou, H. (2018). 3D Point Cloud Generation Using Incremental Structure-from-Motion. *Journal of Physics: Conference Series*, *1087*(6). https://doi.org/10.1088/1742-6596/1087/6/062031

Hui, L., Park, M., and Brilakis, I. (2014). Automated In-Place Brick Counting for Facade Construction Progress Estimation. *Computing in Civil and Building Engineering*, *29*(6), 958–965. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000423.

Ibtehaz, N., and Rahman, M. S. (2020). MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation. *Neural Networks*, *121*, 74–87. https://doi.org/10.1016/j.neunet.2019.08.025

Işık, Ş., and Özkan, K. (2014). A Comparative Evaluation of Well-known Feature Detectors and Descriptors. *International Journal of Applied Mathematics, Electronics and Computers*, *3*(1), 1. https://doi.org/10.18100/ijamec.60004

Izutsu, R., Yabuki, N., and Fukuda, T. (2019). As-built detection of steel frame structure using deep learning. *The 4th International Conference on Civil and Building Engineering Informatics (ICCBEI)*, 25–32.

Jadidi, H., Ravanshadnia, M., and Alipour, M. H. (2014). Visualization of As-Built Progress Data Using ConstructionSite Photographs: Two Case Studies. *Proceedings of the 31st International Symposium on Automation and Robotics in Construction and Mining (ISARC)*, *Isarc*. https://doi.org/10.22260/isarc2014/0096

Jahanshahi, M.R., Chen, F.-C., Ansar, A., Padgett, C. W., Clouse, D., and Bayard, D. S. (2017).

Accurate and Robust Scene Reconstruction in the Presence of Misassociated Features for Aerial Sensing. *Journal of Computing in Civil Engineering*, *31*(6), 1–11. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000702

Jahanshahi, Mohammad R., and Masri, S. F. (2013). Parametric performance evaluation of wavelet-based corrosion detection algorithms for condition assessment of civil infrastructure systems. *Journal of Computing in Civil Engineering*, *27*(4), 345–357. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000225

Jiang, S., and Jiang, W. (2019). Reliable image matching via photometric and geometric constraints structured by Delaunay triangulation. *ISPRS Journal of Photogrammetry and Remote Sensing*, *153*(April), 1–20. https://doi.org/10.1016/j.isprsjprs.2019.04.006

Jog, G. M., Fathi, H., and Brilakis, I. (2011). Automated computation of the fundamental matrix for vision based construction site applications. *Advanced Engineering Informatics*, *25*(4), 725–735. https://doi.org/10.1016/j.aei.2011.03.005

Jung, J., Che, E., Olsen, M. J., and Parrish, C. (2019). Efficient and robust lane marking extraction from mobile lidar point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *147*(June 2018), 1–18. https://doi.org/10.1016/j.isprsjprs.2018.11.012

Jung, J., Che, E., Olsen, M. J., and Shafer, K. C. (2020). Automated and efficient powerline extraction from laser scanning data using a voxel-based subsampling with hierarchical approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, *163*(July 2019), 343–361. https://doi.org/10.1016/j.isprsjprs.2020.03.018

Jung, J., Hong, S., Yoon, S., Kim, J., and Heo, J. (2016). Automated 3D Wireframe Modeling of Indoor Structures from Point Clouds Using Constrained Least-Squares Adjustment for As-Built BIM. *Journal of Computing in Civil Engineering*, *30*(4), 04015074. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000556

Kaehler, A., and Bradski, G. (2016). *Learning OpenCV 3* (D. Schanafelt (ed.); 1st ed.). O'Reilly Media, Inc.

Kang, Z., and Yang, J. (2018). A probabilistic graphical model for the classification of mobile LiDAR point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *143*(September 2018), 108–123. https://doi.org/10.1016/j.isprsjprs.2018.04.018

Kim, P., Park, J., Cho, Y. K., and Kang, J. (2019). UAV-assisted autonomous mobile robot navigation for as-is 3D data collection and registration in cluttered environments. *Automation in Construction*, *106*. https://doi.org/10.1016/j.autcon.2019.102918

Koch, C., and Brilakis, I. (2011). Pothole detection in asphalt pavement images. *Advanced Engineering Informatics*, *25*(3), 507–515. https://doi.org/10.1016/j.aei.2011.01.002

Kolar, Z., Chen, H., and Luo, X. (2018). Transfer learning and deep convolutional neural networks for safety guardrail detection in 2D images. *Automation in Construction*, *89*(May 2017), 58–70. https://doi.org/10.1016/j.autcon.2018.01.003

Kong, X., and Li, J. (2019). Non-contact fatigue crack detection in civil infrastructure through image overlapping and crack breathing sensing. *Automation in Construction*, *99*(December 2018), 125–139. https://doi.org/10.1016/j.autcon.2018.12.011

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Lee, I., and Schenk, T. (2002). Perceptual organization of 3D surface points. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *34*(January 2002).

Leutenegger, S., Chli, M., and Siegwart, R. Y. (2012). BRISK: Binary Robust Invariant Scalable Keypoints Stefan. *2011 International Conference on Computer Vision*. https://doi.org/10.1109/ICCV.2011.6126542

Li, F., Lehtomäki, M., Oude Elberink, S., Vosselman, G., Kukko, A., Puttonen, E., Chen, Y., and Hyyppä, J. (2019). Semantic segmentation of road furniture in mobile laser scanning data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *154*(November 2018), 98–113. https://doi.org/10.1016/j.isprsjprs.2019.06.001

Li, K., Wan, G., Cheng, G., Meng, L., and Han, J. (2020). Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, *159*(May 2019), 296–307. https://doi.org/10.1016/j.isprsjprs.2019.11.023

Li, Yali, Wang, S., Tian, Q., and Ding, X. (2015). A survey of recent advances in visual feature detection. *Neurocomputing*, *149*(PB), 736–751. https://doi.org/10.1016/j.neucom.2014.08.003

Li, Yong, Tong, G., Du, X., Yang, X., Zhang, J., and Yang, L. (2019). A single point-based multilevel features fusion and pyramid neighborhood optimization method for ALS point cloud classification. *Applied Sciences (Switzerland)*, *9*(5). https://doi.org/10.3390/app9050951

Li, Yuan, Wu, B., and Ge, X. (2019). Structural segmentation and classification of mobile laser scanning point clouds with large variations in point density. *ISPRS Journal of Photogrammetry and Remote Sensing*, *153*(May), 151–165. https://doi.org/10.1016/j.isprsjprs.2019.05.007

Lin, W. Y., Wang, F., Cheng, M. M., Yeung, S. K., Torr, P. H. S., Do, M. N., and Lu, J. (2018). CODE: Coherence Based Decision Boundaries for Feature Correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(1), 34–47. https://doi.org/10.1109/TPAMI.2017.2652468

Liu, Y., Cho, S., and Fan, J. (2016). Concrete Crack Assessment Using Digital Image Processing and 3D Scene Reconstruction. *Journal of Computing in Civil Engineering*, *30*(1), 1–19. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000446.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, *60*(2), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

Lu, Q., and Lee, S. (2017). Image-Based Technologies for Constructing As-Is Building Information Models for Existing Buildings. *Journal of Computing in Civil Engineering*, *31*(4), 04017005. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000652

Lu, R., Brilakis, I., and Middleton, C. R. (2019). Detection of Structural Components in Point Clouds of Existing RC Bridges. *Computer-Aided Civil and Infrastructure Engineering*, *34*(3), 191–212. https://doi.org/10.1111/mice.12407

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., and Johnson, B. A. (2019). Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS Journal of Photogrammetry and Remote Sensing*, *152*(April), 166–177. https://doi.org/10.1016/j.isprsjprs.2019.04.015

Mair, E., Hager, G. D., Burschka, D., Suppa, M., and Hirzinger, G. (2010). Adaptive and Generic Corner Detection Based on the Accelerated Segment Test. *In: Daniilidis K., Maragos P., Paragios N. (Eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science*, *6312*, 183–196. https://doi.org/10.1007/978-3-642-15552-9_14

Martinez, P., Ahmad, R., and Al-Hussein, M. (2019). A vision-based system for pre-inspection of steel frame manufacturing. *Automation in Construction*, *97*(November 2018), 151–163. https://doi.org/10.1016/j.autcon.2018.10.021

McClune, A. P., Mills, J. P., Miller, P. E., and Holland, D. A. (2016). Automatic 3d building reconstruction from a dense image matching dataset. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *41*, 641–648. https://doi.org/10.5194/isprsarchives-XLI-B3-641-2016

McLachlan, G. J., Do, K.-A., and Ambroise, C. (2004). *Analyzing Microarray Gene Expression Data*. John Wiley & Sons, Inc.

Meng, H. Y., Gao, L., Lai, Y. K., and Manocha, Di. (2019). VV-net: Voxel VAE net with group convolutions for point cloud segmentation. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 8499–8507. https://doi.org/10.1109/ICCV.2019.00859

Merkurjev, E. (2020). A fast graph-based data classification method with applications to 3D sensory data in the form of point clouds. *Pattern Recognition Letters*, *136*, 154–160. https://doi.org/10.1016/j.patrec.2020.06.005

Mineo, C., Pierce, S. G., and Summan, R. (2019). Novel algorithms for 3D surface point cloud boundary

detection and edge reconstruction. *Journal of Computational Design and Engineering*, *6*(1), 81–91. https://doi.org/10.1016/j.jcde.2018.02.001

Mouats, T., Aouf, N., Nam, D., and Vidas, S. (2018). Performance Evaluation of Feature Detectors and Descriptors Beyond the Visible. In *Journal of Intelligent and Robotic Systems: Theory and Applications* (Vol. 92, Issue 1). Journal of Intelligent & Robotic Systems. https://doi.org/10.1007/s10846-017-0762-8

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Van Gool, L., and Purgathofer, W. (2013). A survey of urban reconstruction. *Computer Graphics Forum*, *32*(6), 146–177. https://doi.org/10.1111/cgf.12077

Ning, X., Li, F., Tian, G., and Wang, Y. (2018). An efficient outlier removal method for scattered point cloud data. *PLoS ONE*, *13*(8), 1–22. https://doi.org/10.1371/journal.pone.0201280

Nüchter, A., and Hertzberg, J. (2008). Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, *56*(11), 915–926. https://doi.org/10.1016/j.robot.2008.08.001

Oron, S., Dekel, T., Xue, T., Freeman, W. T., and Avidan, S. (2018). Best-Buddies Similarity - Robust Template Matching Using Mutual Nearest Neighbors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(8), 1799–1813. https://doi.org/10.1109/TPAMI.2017.2737424

Park, J., Cai, H., and Perissin, D. (2018). Bringing Information to the Field: Automated Photo Registration and 4D BIM. *Journal of Computing in Civil Engineering*, *32*(2), 04017084. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000740

Park, Y., and Guldmann, J. M. (2019). Creating 3D city models with building footprints and LIDAR point cloud classification: A machine learning approach. *Computers, Environment and Urban Systems*, *75*(January), 76–89. https://doi.org/10.1016/j.compenvurbsys.2019.01.004

Peng, S., Jiang, W., Pi, H., Li, X., Bao, H., and Zhou, X. (2020). Deep snake for real-time instance segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8530–8539. https://doi.org/10.1109/CVPR42600.2020.00856

Pessoa, G. G., Amorim, A., Galo, M., and Galo, M. de L. B. T. (2019). Photogrammetric point cloud classification based on geometric and radiometric data integration. *Boletim de Ciencias Geodesicas*, *25*(Special Issue), 1–17. https://doi.org/10.1590/s1982-21702019000S00001

Politz, F., Kazimi, B., and Sester, M. (2018). Classification of Laser Scanning Data Using Deep Learning. *Remote Sensing*, *27*(9), 597–610. http://www.mdpi.com/2072-4292/8/9/730

Predescu, A.-D., and Triantafyllidis, G. (2018). Real-time, anthropomorphic 3-D scanning and voxel display system using consumer depth cameras as an interactive means of individual artistic expression through light. *SHS Web of Conferences*, *43*, 01002. https://doi.org/10.1051/shsconf/20184301002

Qu, Z., Lin, S. P., Ju, F. R., and Liu, L. (2015). The Improved Algorithm of Fast Panorama Stitching for Image Sequence and Reducing the Distortion Errors. *Mathematical Problems in Engineering*, *2015*. https://doi.org/10.1155/2015/428076

Rahal, R., Asmar, D., Shammas, E., and Ghanem, B. (2018). Object Oriented Structure from Motion: Can a Scribble Help? *13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018)*, *5*(Visigrapp), 541–548. https://doi.org/10.5220/0006596005410548

Raoult, V., David, P. A., Dupont, S. F., Mathewson, C. P., O'Neill, S. J., Powell, N. N., and Williamson, J. E. (2016). GoPros™ as an underwater photogrammetry tool for citizen science. *PeerJ*, *4*, e1960. https://doi.org/10.7717/peerj.1960

Rashidi, A., Brilakis, I. and Vela, P. A. (2013). Built Infrastructure Point Cloud Data Cleaning: An Overview of Gap Filling Algorithms. *Proceedings of the 13th International Conference on Construction Applications of Virtual Reality, October*, 585–593.

Rashidi, A., Brilakis, I., and Vela, P. (2015). Generating Absolute-Scale Point Cloud Data of Built Infrastructure Scenes Using a Monocular Camera Setting. *Journal of Computing in Civil Engineering*, *29*(9), 04014089. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000414

Rashidi, A., and Karan, E. (2018). Video to BrIM: Automated 3D As-Built Documentation of Bridges. *Journal of Performance of Constructed Facilities*, *32*(3), 04018026. https://doi.org/10.1061/(asce)cf.1943-5509.0001163

Rastiveis, H., Shams, A., Sarasua, W. A., and Li, J. (2020). Automated extraction of lane markings from mobile LiDAR point clouds based on fuzzy inference. *ISPRS Journal of Photogrammetry and Remote Sensing*, *160*(June 2019), 149–166. https://doi.org/10.1016/j.isprsjprs.2019.12.009

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. https://doi.org/10.1109/CVPR.2016.91

Robinson, D. J. S. (2008). *An Introduction to Abstract Algebra*. De Gruyter. https://doi.org/10.1515/9783110198164

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. Wells, and A. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science* (Vol. 9351). Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

Rosten, E., and Drummond, T. (2005). Fusing Points and Lines for High Performance Tracking. *Proceedings of the IEEE International Conference on Computer Vision*, 1508–1511.

Rothe, R., Guillaumin, M., and Gool, L. Van. (2015). Non-Maximum Suppression for Object Detection by Passing Messages between Windows. *Asian Conference on Computer Vision (ACCV 2014)*.

https://doi.org/10.1007/978-3-319-16865-4_19

Roveri, R., Rahmann, L., Oztireli, A. C., and Gross, M. (2018). A Network Architecture for Point Cloud Classification via Automatic Depth Images Generation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4176–4184. https://doi.org/10.1109/CVPR.2018.00439

Rublee, E., Rabaud, V., and Konolige, K. (2011). ORB : an efficient alternative to SIFT or SURF. *Intl. Conf. Computer Vision*, 1–5. https://doi.org/10.1002/art.38045

Rusinol, M., Chazalon, J., Ogier, J. M., and Llados, J. (2015). A comparative study of local detectors and descriptors for mobile document classification. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, *2015-Novem*, 596–600. https://doi.org/10.1109/ICDAR.2015.7333831

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, *77*(1–3), 157–173. https://doi.org/10.1007/s11263-007-0090-8

Rusu, R. B., Blodow, N., Marton, Z., Soos, A., and Beetz, M. (2007). Towards 3D object maps for autonomous household robots. *IEEE International Conference on Intelligent Robots and Systems*, 3191–3198. https://doi.org/10.1109/IROS.2007.4399309

Saha, S. K., Xiao, D., Frost, S., and Kanagasingam, Y. (2018). Performance Evaluation of State-of-the-Art Local Feature Detectors and Descriptors in the Context of Longitudinal Registration of Retinal Images. *Journal of Medical Systems*, *42*(4). https://doi.org/10.1007/s10916-018-0911-z

Saovana, N., Yabuki, N., and Fukuda, T. (2020a). Development of an unwanted-feature removal system for Structure from Motion of repetitive infrastructure piers using deep learning. *Advanced Engineering Informatics*, *46*(July), 101169. https://doi.org/10.1016/j.aei.2020.101169

Saovana, N., Yabuki, N., and Fukuda, T. (2019a). A Performance Evaluation of Feature Detectors and Descriptors for Unmodified Infrastructure Site Digital Images. *4th International Conference on Civil and Building Engineering Informatics (ICCBEI2019)*, 113–120.

Saovana, N., Yabuki, N., and Fukuda, T. (2019b). A Quantitative Effect Evaluation of the Unwanted Features Removal of Infrastructure Digital Images. *19th International Conference on Construction Applications of Virtual Reality (CONVR2019)*, 62–71.

Saovana, N., Yabuki, N., and Fukuda, T. (2020b). An image feature classification system for piers based on semantic segmentation with deep learning. *Proceedings of the 20th International Conference on Construction Applications of Virtual Reality (CONVR2020)*, 325–335.

Sfikas, K., Pratikakis, I., and Theoharis, T. (2018). Ensemble of PANORAMA-based convolutional neural networks for 3D model classification and retrieval. *Computers and Graphics (Pergamon)*,

71, 208–218. https://doi.org/10.1016/j.cag.2017.12.001

Simonson, K. M., Drescher, S. M., and Tanner, F. R. (2007). A statistics-based approach to binary image registration with uncertainty analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(1), 112–125. https://doi.org/10.1109/TPAMI.2007.250603

Simonyan, K., and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–14.

Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Modeling the world from Internet photo collections. *International Journal of Computer Vision*, *80*(2), 189–210. https://doi.org/10.1007/s11263-007-0107-3

Spacek, L. A. (1986). Edge detection and motion detection. *Image and Vision Computing*, *4*(1), 43–56. https://doi.org/10.1016/0262-8856(86)90007-7

Spencer, B. F., Hoskere, V., and Narazaki, Y. (2019). Advances in Computer Vision-Based Civil Infrastructure Inspection and Monitoring. *Engineering*, *5*(2), 199–222. https://doi.org/10.1016/j.eng.2018.11.030

Su, J., Srivastava, A., and Huffer, F. W. (2013). Detection, classification and estimation of individual shapes in 2D and 3D point clouds. *Computational Statistics and Data Analysis*, *58*(1), 227–241. https://doi.org/10.1016/j.csda.2012.09.008

Tang, P., Akinci, B., and Huber, D. (2009). Quantification of edge loss of laser scanned data at spatial discontinuities. *Automation in Construction*, *18*(8), 1070–1083. https://doi.org/10.1016/j.autcon.2009.07.001

Tang, P., Huber, D., Akinci, B., Lipman, R., and Lytle, A. (2010). Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in Construction*, *19*(7), 829–843. https://doi.org/10.1016/j.autcon.2010.06.007

Tareen, S. A. K., and Saleem, Z. (2018). A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. *2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development, ICoMET 2018 - Proceedings*, *2018-Janua*, 1–10. https://doi.org/10.1109/ICOMET.2018.8346440

Teo, T. A. (2015). Object-based point clouds classification using airborne waveform LiDAR. *ACRS 2015 - 36th Asian Conference on Remote Sensing: Fostering Resilient Growth in Asia, Proceedings*, *2008*. https://www.scopus.com/record/display.uri?eid=2-s2.0-84964047855&origin=inward&txGid=31bd8892600e4c3d7277aa9b0666647a

Triggs, B., Mclauchlan, P., Hartley, R., and Fitzgibbon, A. (2010). Bundle Adjustment – A Modern Synthesis. *International Workshop on Vision Algorithms*, 298–372. https://doi.org/10.1007/3-540-

44480-7_21

Uy, M. A., Pham, Q. H., Hua, B. S., Nguyen, T., and Yeung, S. K. (2019). Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 1588–1597. https://doi.org/10.1109/ICCV.2019.00167

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Mulbregt, P. van. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*(3), 261–272.

Vu, H. H., Labatut, P., Pons, J. P., and Keriven, R. (2012). High accuracy and visibility-consistent dense multiview stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(5), 889–901. https://doi.org/10.1109/TPAMI.2011.172

Wang, C., Cheng, M., Sohel, F., Bennamoun, M., and Li, J. (2019). NormalNet: A voxel-based CNN for 3D object classification and retrieval. *Neurocomputing*, *323*, 139–147. https://doi.org/10.1016/j.neucom.2018.09.075

Wang, D. (2020). Unsupervised semantic and instance segmentation of forest point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *165*(March), 86–97. https://doi.org/10.1016/j.isprsjprs.2020.04.020

Wang, L., Meng, W., Xi, R., Zhang, Y., Ma, C., Lu, L., and Zhang, X. (2019). 3D point cloud analysis and classification in large-scale scene based on deep learning. *IEEE Access*, *7*, 55649–55658. https://doi.org/10.1109/ACCESS.2019.2909742

Wang, Q., Sohn, H., and Cheng, J. C. P. (2018). Automatic As-Built BIM Creation of Precast Concrete Bridge Deck Panels Using Laser Scan Data. *Journal of Computing in Civil Engineering*, *32*(3), 1–17. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000754

Wang, Y., Xu, Z., Shen, H., Cheng, B., and Yang, L. (2020). CenterMask: Single shot instance segmentation with point representation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 9310–9318. https://doi.org/10.1109/CVPR42600.2020.00933

Weidner, L., Walton, G., and Kromer, R. (2019). Classification methods for point clouds in rock slope monitoring: A novel machine learning approach and comparative analysis. *Engineering Geology*, *263*(October), 105326. https://doi.org/10.1016/j.enggeo.2019.105326

Wen, C., Yang, L., Li, X., Peng, L., and Chi, T. (2020). Directionally constrained fully convolutional neural network for airborne LiDAR point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, *162*(August 2019), 50–62. https://doi.org/10.1016/j.isprsjprs.2020.02.004

Westoby, M. J., Brasington, J., Glasser, N. F., Hambrey, M. J., and Reynolds, J. M. (2012). "Structure-from-Motion" photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, *179*(June 2018), 300–314. https://doi.org/10.1016/j.geomorph.2012.08.021

Wrózyński, R., Pyszny, K., Sojka, M., Przybyła, C., and Murat-Błazejewska, S. (2017). Ground volume assessment using "Structure from Motion" photogrammetry with a smartphone and a compact camera. *Open Geosciences*, *9*(1), 281–294. https://doi.org/10.1515/geo-2017-0023

Xu, X., Caulfield, S., Amaro, J., Falcao, G., and Moloney, D. (2020). 1.2 Watt Classification of 3D Voxel Based Point-clouds using a CNN on a Neural Compute Stick. *Neurocomputing*, *393*, 165–174. https://doi.org/10.1016/j.neucom.2018.10.114

Xu, Y., Boerner, R., Yao, W., Hoegner, L., and Stilla, U. (2019). Pairwise coarse registration of point clouds in urban scenes using voxel-based 4-planes congruent sets. *ISPRS Journal of Photogrammetry and Remote Sensing*, *151*(September 2018), 106–123. https://doi.org/10.1016/j.isprsjprs.2019.02.015

Yang, J., Park, M. W., Vela, P. A., and Golparvar-Fard, M. (2015). Construction performance monitoring via still images, time-lapse photos, and video streams: Now, tomorrow, and the future. *Advanced Engineering Informatics*, *29*(2), 211–224. https://doi.org/10.1016/j.aei.2015.01.011

Yang, P., Yang, G., Gong, X., Wu, P., Han, X., Wu, J., and Chen, C. (2020). Instance Segmentation Network with Self-Distillation for Scene Text Detection. *IEEE Access*, *8*, 45825–45836. https://doi.org/10.1109/ACCESS.2020.2978225

Zeiler, M. D., and Fergus, R. (2014). Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8689 LNCS*(PART 1), 818–833. https://doi.org/10.1007/978-3-319-10590-1_53

Zhang, A., Wang, K. C. P., Fei, Y., Liu, Y., Tao, S., Chen, C., Li, J. Q., and Li, B. (2018). Deep Learning–Based Fully Automated Pavement Crack Detection on 3D Asphalt Surfaces with an Improved CrackNet. *Journal of Computing in Civil Engineering*, *32*(5), 04018041. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000775

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, *13*(2), 119–152. https://doi.org/10.1007/BF01427149

Zhao, Y., Hu, Q., and Hu, W. (2018). A point cloud classification approach based on vertical structures of ground objects. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(3), 2427–2434. https://doi.org/10.5194/isprs-archives-XLII-3-2427-2018

Zhu, Q., Li, Y., Hu, H., and Wu, B. (2017). Robust point cloud classification based on multi-level semantic relationships for urban scenes. *ISPRS Journal of Photogrammetry and Remote Sensing*,

129, 86–102. https://doi.org/10.1016/j.isprsjprs.2017.04.022

Zhu, Z., and Brilakis, I. (2010). Concrete Column Recognition in Images and Videos. *Journal of Computing in Civil Engineering*, *24*(6), 478–487. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000053