Machine Learning for Improved Detection and Segmentation of Building Boundary

Salem Algarni

A thesis presented for the degree of Doctor of Philosophy

School of Engineering

Cardiff University

February 2022

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed ..... (candidate)

Date .....

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ..... (candidate)

Date .....

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

Abstract

The first step in rescuing and mitigating the losses from natural or man-made disasters is to assess damaged assets, including services, utilities and infrastructure, such as buildings. However, manual visual analysis of the affected buildings can be time consuming and labour intensive. Automatic detection of buildings, on the other hand, has the potential to overcome the limitations of conventional approaches. This thesis reviews the existing methods for the automated detection of objects using multi-source geospatial data and presents two novel post processing techniques. Effective building segmentation and recognition techniques are also investigated. Artificial intelligence techniques have been used to identify building boundaries in automated building-detection applications. Compared with other neural network models, the convolutional neural network (CNN) architectures based on supervised and unsupervised approaches provide better results by looking at the image details as spatial information of the entity in the frame. This research incorporates the improved semantic detection ability of Region-based Convolutional Neural Network (Mask R-CNN) and the segmentation refining capability of the conditional random field (CRF)s. Mask R-CNN uses a pre-trained network to recognise the boundary boxes around buildings. It also provides contour key points around buildings that are masked in satellite images. This thesis proposes two novel post-processing techniques that operate by modifying and detecting the building's relative orientation properties and combining the key points predicted by the two head neural networks to modify the predicted contour with

the help of the proposed novel snap algorithms. The results show significant improvements in the accuracy of boundary detection compared with the state-of-the-art techniques of 2.5%, 4.6% and 1% for F1-Score, Intersection over Union also known as Jacard coefficient (IoU), and overall pixel accuracy, respectively. CNNs have proven to be powerful tools for a wide range of image processing tasks where they can be used to automatically learn mid-level and high-level concepts from raw data, such as images. Finally, the results highlight the potential of further approaches to these applications, such as the planning of infrastructure.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor Prof. Monjur Mourshed, Professor of Sustainable Engineering at Cardiff University, who has supported my PhD journey. This thesis would not have been possible without his guidance and valuable assistance. His immense knowledge and great experience has encouraged me both in my academic research and in my daily life. He is the true definition of a leader and the ultimate role model; I have learned from his personal skills and I hope to carry this forward throughout my career. For her unconditional love, to my mother, I would say, "all useful happenings, accomplishments and success in my life are because of your support and prayers for me every day. May Allah bless and protect you forever."
Most importantly, I am grateful for my family's (my wife, my sons, Mohammad and Ahmed, and my daughte, Orchid) unconditional, unequivocal, and loving support during my PhD journey, and I would say to them, "I won't forget your sacrifice and patience with me forever." I am forever thankful to my brothers, especially Ali, and to my nephews, for their support and encouragement.
Deepest thanks to my colleagues and true friends (Qulail, Mohammed and Abdulrazaq), who keep me grounded, remind me of what is essential in life, and always support my journey. I would like to express my gratitude to all members of my sponsors, King Fahd Security College, for their kind motivation and funding of my study; I ask God to allow me to return this scientific trust.

Contents

List of Figures

List of Tables

Acronyms

AI	artificial intelligence.
ANN	Artificial Neural Networks.
ANNs	Artificial neural networks.
CNN	convolutional neural network.
CRF	conditional random field.
DCNN	deep convolutional neural network.
DEM	digital elevation model.
DGPF	Digital Government Policy Framework.
DMP	differential morphological profile.
DNN	Deep neural networks.
DSM	digital surface model.
DTM	digital terrain modelling.
EaNet	edge-aware neural network.
FAIR	Facebook AI Research.
FCN	fully convolutional networks.
FPN	feature pyramid network.

GAN	generative adversarial networks.
GPUs	graphics processing Units.
IoU	Intersection over Union also known as Jacard coefficient.
IR	infrared.
ISPRS	International society for photogrammetry and remote sensing.
LiDAR	Light Detection and Ranging.
LKPP	large kernel pyramid pooling.
LoD	levels of details.
LSTM	Long short-term memory.
Mask R-CNN	Region-based Convolutional Neural Network.
MBR	minimum bounding rectangle.
MLCG	multi-level context-guided classification.
MLP	multi-layer perceptron.
MRS	multi-resolution segmentation.
NIR	Near Infrared Region.
OBIA	object-based image analysis.
OCNN	Object-based CNN.
R-CNN	Region-based Convolutional Neural Network.
RAM	random access memory.
ResNet	residual neural network.
ResNeXt	residual neural network NeXt.

RF	random forest decision tree classifier.
RGB	Red, Green, and Blue.
RNN	recurrent neural networks.
RoI	region of interest.
RoIAlign	aligning region of interest.
RoIPool	region of interest pooling.
RPN	Region Proposal Network.
SAGAN	self-attention-GAN.
SAR	synthetic-aperture radar.
SegNet	special deep architecture.
SSIM	Structural similarity index measure.
SSP	Segmentation Scale Parameter.
SSPs	segmentation scale parameters.
SVM	support vector machines.
TLS	terrestrial laser scanning.
TOP	true orthophoto.
U-Net	You Only Look Once.
VHR	very high resolution.

Chapter 1

Introduction

This chapter introduces the purpose of this research. In addition, it explains the need for better techniques to exploit the power of more than one artificial neural network and describes the building boundary refining post-processing techniques for automatic building detection. Then it explains the motivation behind exploiting the information available within building satellite imagery in the form of vectorised boundary data. This chapter also explores some of the problem associated with the existing techniques for automatic building detection. Finally, this chapter presents the main objectives of the research, lists the contributions that have been made, and outlines the structure of this thesis.

1.1 Background

Most buildings follow a rigid geometrical structure. Consequently, the top views of a building usually have a definite shape (e.g., rectangular, square or a combination of any other geometrical shape). This narrows down the entire shape of the building into a smaller space, where the task will be to estimate a path that allows this shape to be maintained. To navigate through the shape of the building, it is necessary to know its contours and major corner points.

Following the rapid development of spaceborne imaging, geospatial object detection of remote sensing imagery has attracted increasing interest. However, most of the previously proposed object detectors are very sensitive to object deformations, such as scaling and rotation. Recently, geospatial object detection has

received considerable attention in the remote-sensing community. The main challenges and difficulties are that objects in optical remote sensing imagery usually suffer from misrepresentations that are caused by scaling, offset, rotation, illumination, and atmospheric conditions, which inevitably degrade the performance of the detection [1].

Academic research in this area can be roughly categorised by template matching-based, object-based, and machine learning-based methods [2]. Unfortunately, these approaches mostly fail to capture rotation-related properties for small-scale training samples. Object detection in very high resolution (VHR) remote sensing images determines whether or not a given aerial or satellite image includes one or more objects belonging to the class of interest and then detects the locations of each predicted object in the image [2]. In remote sensing, the term 'object' mainly refers to man-made objects with clear borders that are independent of backdrop (e.g., buildings, cars, storage tanks, etc.). Meanwhile, object detection in remote sensing images is a prominent image processing issue that is essential for strategic planning and recreational applications. However, this is still a difficult challenge thanks to the varying visual presentation of objects caused by occlusion, lighting, shadow, perspective fluctuation, resolution, polarisation, noise, and so on. Furthermore, the enormous expansion in the quantity and intensity of remote sensing images causes incredibly high processing costs, which raises the complexities of object detection for near-real-time applications. While object detection and semantic segmentation have made significant progress in recent years, the issue of instance segmentation remains complicated. In the remote sensing field of extreme segmentation, the target needs to not only detect individual object instances, but is also required to provide a mask for each instance with geospatial information (e.g., the coordinate points and the orientation of the objects).

Given the strength of meta-algorithms, this thesis will demonstrate some of the methods (e.g., segmentation) that will act as a meta-algorithm in terms of speed,

accuracy, and simplicity. These methods may be used in both future research of instance segmentation and in many other applications of instance-level understanding.

Over the last few years, instance segmentation approaches have been commonly classified into two groups. The first group includes the popular Region-based Convolutional Neural Network (Mask R-CNN) methods [3], which typically begin with a proposal for segmentation levels. Classifiers are then trained to classify these proposals into somatic classes. The second group includes the successful fully convolutional networks (FCN) methods [4]. They typically begin with full image somatic segmentation results and then learn a particular split to divide them into individual instances.

Figure 1.1 illustrates building roof tops, which has two separate structures joined under one roof. As can be seen, the image has a high amount of shadow on the top side. It also has a dark region between the region where the building changes from one section to another. These types of shadows around the building's image provide challenges to artificial intelligence (AI) based methods, and the way in which they classify and detect the contours around the boundaries of buildings.

Another example of some of the challenges that are faced when using this technique, as can be seen in this image in the figure above, is that there are trees over a small part of the roof top, which cover the top view and hide the edge of the building. These types of obstructions also pose challenges to the AI based methods that are used to classify and detect the contours around the boundaries of buildings.

The current state of the art segmentation networks are able to identify the location of a building and its segmentation mask on a given image. Although it is easy to obtain the contour of the building using the available segmentation mask, it may not be possible to maintain the rigid geometrical structure.

Figure 1.1: Top mosaic 09cm R RGB from VHR satellite imagery that shows buildings with useful geometrical information (source: area 8, Vaihingen, Germany dataset).

Achieving higher accuracy when dealing with the building structure is also important because it will provide improved engineering estimates (i.e., the size, orientation, and shape) of the building to achieve the many goals of the applications that were mentioned earlier.

One of the motivations for this research work was to understand the challenges faced by the current AI based methods that are used to detect the accurate contours of buildings and then improve this accuracy by utilising the geometrical information available within building images.

As can be seen in Figure 1.1, buildings have some very useful information that

is available in the form of corner points and principal directions. Hence, this research is motivated by the need to extract and utilise the key corner points and directional information to refine the contours that are detected by post-processing with the help of AI based approaches to reach the outputs.

1.2 Associated Problems

Although automatic building detection techniques use feature extraction and object detection techniques, they rely heavily on object segmentation. Moreover, a number of deep learning techniques are exploited for automatic building detection tasks. While traditional deep learning techniques use supervised learning, recent deep learning techniques (e.g., pre-training a convolutional neural network (CNN), generative adversarial networks (GAN), etc.) provide better unsupervised results for images without labels.

VHR images of buildings contain a lot of useful information (e.g., roof boundaries, shapes, edges, corners etc.), which has been the focus of the prior art in building detection. Many satellite images contain buildings of different scales and sizes, which are available together in a single image. Similarly, some relative directional information is available, which can be exploited. Therefore, a different perspective is required to look for more semantic information in building images; for example, a great deal of semantic information is available, such as 'roof tops', 'building wall', 'single hip', 'multiple hips', 'hexagonal gazebo', 'shed', and so on [5].

While many types of building data sources are available (e.g., Light Detection and Ranging (LiDAR), digital elevation model (DEM), digital terrain modelling (DTM), and point cloud data etc.), the complexity of processing these types of data using various artificial neural networks to detect building boundaries, and to then further classify and extract semantic information from it, is high. In addition, because the amount of data to be processed is so large, the computa-

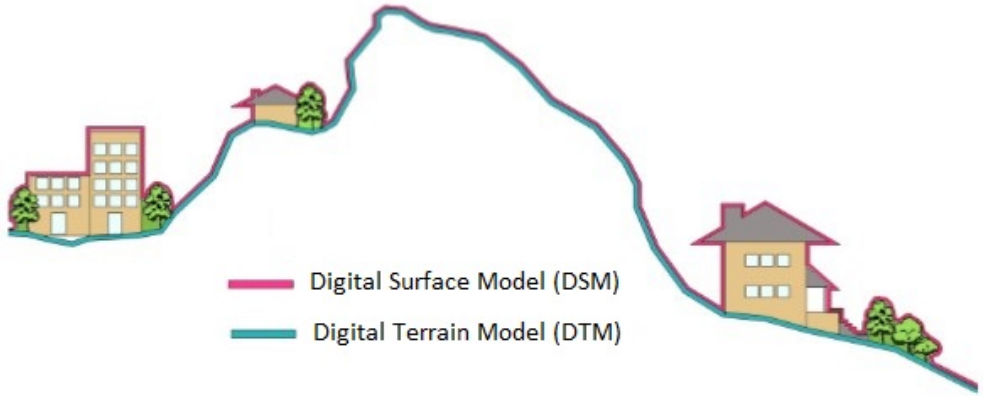


Figure 1.2: Comparison between DTM and DSM.

tional power required and the training time needed are significant. Traditional remote sensing classification algorithms are often manually developed during their feature design, using simple structure and architecture classifiers. Feature extraction is performed on small imaging areas (e.g., objects, regions, image patches, or super-pixels), employing specific algorithms. The original spectral feature space is then transmuted into a compact, abstract illustration. This will result in spatial characteristics that can even have a significant impact on the performance of a classifier.

Several techniques can be used to extract levels of details (LoD) from building data. In particular, many researchers have attempted to utilise AI approaches by extracting the available semantic information and processing different data sources. The primary focus of these approaches and their associated problems are as follows:

- Computational complexity in the significant high-dimensional data processing.
- A lack of data uniqueness or performance.
- A reduction the computational difficulty of training an actual detection method.

- Verifying normalisation to various detection methods.

This thesis aims to assess these problems. It also describes some recent theoretical and practical developments over the last few years. Based on these issues, this thesis aims to address the following research questions and hypotheses:

- **What are the optimum data types to extract the appropriate LoD information from buildings?**

  The primary aim of many recently published research papers in the field of automatic building detection has been to process different remote sensing data (e.g., LiDAR, DEM, DTM, point cloud and satellite images) to extract building roof top information for detection.

  Therefore, the first hypothesis of this research work states: "Satellite imagery data is sufficient to provide accurate LoD information for roof top detection."

- **Can a novel method be developed based on deep learning techniques to detect complex building boundaries using VHR remotely sensed images?**

  Therefore, the second hypothesis of this research work states: "An architecture method that is based on deep learning techniques can be used to classify and extract localised contours around building boundaries using VHR remotely-sensed images."

- **Can post-processing techniques improve the output of the deep learning methods to extract further information from the detected buildings?**

  The published research papers in the field of automatic building detection have focused on improving the output using post-processing techniques combined with classification. They have used smoothed image classification by

averaging over segmented regions or by employing a conditional random field (CRF) approach. Zhao et al.[6] proposed a contour-preserving CNN method for semantic segmentation and post-processing to smooth the classification results. Marmanis and Fu [7] used FCN-based approaches for dense classification and CRF to refine region boundaries. These methods have used post-processing to fix, refine and smooth the boundary issues caused by CNN models.

Therefore, the third hypothesis of this research work states: "Post-processing techniques can significantly improve the output for detection accuracy."

- Can any of the deep learning architecture-based features that are available be converted into vectors to support the post-processing calculation to determine functional geometrical building information for LoD information extraction?

  Kaiming He and a team of researchers developed Mask R-CNN [3], which is a well-known and influential deep learning architecture, to explore Facebook. This architecture outputs three primary detection features: bounding box, mask and keypoint. Keypoint detection in images comes from a desire to extract facial emotions and a heatmap from an image. A heatmap is a vector that contains primarily zero values, and has the same height and width as the input image. A keypoint is a positive pixel in the middle of the image. Networks can then be trained to look for frequent patterns around this keypoint and learn how to detect them in images. Each class has its own heatmap to avoid keypoint clusters that can disrupt training.

  Therefore, the fourth hypothesis of this research work states: "Detection of the vectors performed by deep learning architecture-based can provide vital information to help post-processing determine useful geometrical information for building detection."

1.3 Objectives

The main objective of this PhD research is to extract high-level details of objects, particularly buildings, using high-resolution satellite imagery with ground truth data. This thesis develops deep learning neural network architectures that are adapted to meet remote sensing requirements, such as detecting accurate shapes and the geospatial information of objects in a large urban area. To attain this aim, the following specific objectives are posed:

1 To review state-of-the-art methodologies, obstacles and results achieved through various methods, which are compared and illustrated to identify viable alternatives for automated urban objects extraction from contemporary satellite imaging data.

2 To use less computationally demanding satellite imagery with ground truth information to classify and help extract useful building information in the form of accurate boundaries and direction information.

3 To devise architectures that are based on deep learning techniques to help classify and extract localised contours around building boundaries.

4 To develop deep learning neural network architectures using VHR imagery with ground truth data.

5 To develop post-processing techniques that can extract the directional vectorised information that is available in the building boundary images, which should improve the accuracy of roof top detection.

6 To extract building orientation direction with a high level of accuracy using these post-processing techniques.

7 To validate the trained deep learning neural network model using test images to evaluate unsupervised results for the building boundaries using these test models.

8 To validate the developing post-processing techniques on the extracted building boundaries and confirm that the detection accuracies have improved.

9 To suggest future directions of research.

1.4 Contributions

The main contributions of this research are that by only using existing VHR satellite images with ground truth data, a deep learning neural network architecture model was developed and used to propose viable solutions. The models achieved an overall pixel accuracy of approximately 93%, and 94% for automated urban extraction object. The contributions that have been derived from this study were obtained by using a Mask R-CNN approach with novel post-processing techniques, which included:

1 The first contribution is that the Mask R-CNN approach that is used in this research differs in many aspects when compared to other contemporary approaches. The proposed method uses (Mask R-CNN with recently developed deep and deeper backbone architectures sizes such as residual neural network (ResNet)-50 or ResNet-101 has been used as the feature backbone. The model with backbone size ResNet-50 contains two structural heads that can predict the masks and the keypoints to exploit the directional information that can be learned during the training process from the building images. As a result, high accuracy semantic segmentation of the buildings is achieved compared with the previous GAN model. Moreover, by increasing the model's size by upgrading the backbone using the aResNet-101, a further improvement in semantic segmentation of the building is achieved, which increases overall pixel accuracy by approximately 93%, and 94%. Finally, additional improvements are obtained by applying CRF-based on the final

output for both model sizes, which increased the accuracy of the predicted results accuracy by approximately 0.8%, and 0.9%.

2 The second contribution of this research is development of novel post-processing techniques (method 1) to detect the semantic segmentation of buildings to improve the shape of the building's boundaries. Both methods proposed rounds of snapping contour algorithms. Meanwhile, each round selected reference lines based on all of the lines within the contour's proximity before, during and after the training setup. Following the detection of these lines, another round of voting algorithms selects new references with the maximum support in a particular direction or angle. Updated reference lines serve as a new foundation for snapping contoured procedures to update the initial round. As a result, the precision and overall accuracy of building shape output are improved by 2%, and 0.5%, respectively, compared with the output obtained by the GAN model, giving a total overall pixel accuracy of 93.6%, and 94.5%.

3 The third contribution of this work is to provide novel post-processing techniques (method 2), which will be divided into three stages. The proposed algorithm will group the predicted keypoints and contour points at a certain distance in the first stage. Common reference points are found for the building using the lines through the adjacent keypoints. The second stage of the algorithm finds the orientation of the building by calculating the North azimuth angle of each line and joining the adjacent keypoints aligned to the North azimuth direction. The algorithm selects the reference that supports the maximum number of angles based on the new reference point using these angles. Finally, the third stage manipulates a point that can be used as a fixed reference point for the keypoints selected to compare all of the lines' directions passing through these points and reference one with respect to the azimuth direction. As a result, one or more of these lines, which

contained the maximum number of angle directions based on azimuth, will be selected and act as a known side of the building's perimeters. When one side of the building is defined, the other sides can be calculated as parallel or perpendicular to the assigned reference line. The proposed algorithm adopts a modified snap contour procedure and this significantly improves the accuracy of the boundaries by approximately $5.8\%, 2.4\%, 3.8\%,$ and 1% of the Precision, F1-score, IOU and overall pixel accuracy, respectively, to achieve a total overall pixel accuracy of $94.0\%,$ and 95.2%. This approach has helped achieve a competitive result when compared with the state-of-the-art in the literature.

1.5 Organisation of This Thesis

This thesis contains six chapters, including the present chapter. A brief description of each of the chapters follows:

Chapter 2 - Research Background: This chapter reviews the literature related to automatic building detection techniques, focusing on common image processing techniques that are based on advanced state-of-the-art AI-based approaches. This chapter also provides an overview of the deep learning and post-processing techniques using the novel methodology that has evolved throughout the research work described in this thesis.

Chapter 3 - Methodology: This chapter explains the philosophical approach of the research methods that have been adopted and refined to develop a new approach to detect accurate building boundaries by using a powerful combination of artificial neural networks and novel post-processing techniques to exploit the vectorised information that is inherently available in satellite images of buildings. This chapter presents the design of the research in the form of the experimental setup and theoretical methodological framework. It also describes their working

to justify several aspects of the proposed methodology. Finally, this chapter discusses some of the performance metrics that are used to evaluate the performance of the proposed methods with respect to state-of-the-art techniques.

Chapter 4 - Results: This chapter presents the findings of the research. In particular, it discusses the results with respect to the improvements that have been achieved. First, the results achieved using GAN-*pix2pix* based approaches are discussed, together with the results attained from a combination of self-attention-GAN (SAGAN) and post-processing using CRF and the proposed building boundary refinement technique. This chapter also discusses the results obtained using Mask R-CNN and various post-processing techniques involving CRF and the proposed building boundary refinement techniques. The results are presented in the form of building images to provide visual qualitative feedback on the boundaries detected by the proposed algorithms, and also in the form of a comparison of various performance metrics. Finally, this chapter will also provide some insights related to the requirements of computational time and training behaviour.

Chapter 5 **- Discussion:** This chapter discusses the research findings and the implications of the results, particularly the results obtained from utilising the power of combined neural networks with the proposed post-processing techniques. This chapter will also discuss the practical applications and limitations of this research. In addition, it critically evaluates this study in terms of how well the findings address the original research questions.

Chapter 6 **- Conclusion and Future Work:** This chapter presents the key-points of this thesis by summarising what was achieved, highlighting the key findings, and reviewing its contributions to the growing body of knowledge on this subject. The conclusions drawn from the research findings are provided in this chapter, together with some indications of how these findings might be useful for both researchers and practitioners. Some recommendations for future work are also given.

Finally, this thesis includes appendixes for reference and to detail the software setup, which includes a list of the various software programs that were used to achieve the desired research work. A list of the research publications drawn from this thesis are also included as an appendix.

1.6 Summary

This thesis consists of six main chapters, chapter one includes the research background and motivation for this research. chapter two presents the associated problems of the research, including the research hypothesis. chapter three presents the scope of the research, and chapter four presents study objectives. chapter five provides the scientific contribution of the present.

Chapter 2

Literature Review

2.1 Introduction

Automatic building detection has emerged as an essential tool in a wide range of applications, such as land utilisation mapping, change detection, urban planning and disaster management. Meanwhile, automatic building detection system research has evolved from traditional image segmentation processing on 2D images to the complex 3D reconstruction of buildings using sophisticated 3D LiDAR, multi-spectral, multi-imagery data. While the amount of information being processed is heading towards "Big Data", the spatial pattern has often benefited from multi-core graphics cards. Further progress is also provided in the form of sophisticated image processing and computer vision algorithms. Having a range of data sources available in different forms (e.g., LiDAR, satellite imagery, infrared, DTM, point cloud, terrestrial laser scanning (TLS), synthetic-aperture radar (SAR), etc.) can assist in identifying and visualising complex features by providing faster early detection. Image processing and machine learning based techniques rely on post-processing techniques and supervised methods in which the dataset needs to be properly labelled for classification. Moreover, the latest AI techniques provide us with an opportunity to work on unsupervised data without labelling.

The research set out in this thesis focuses on utilising VHR remote sensing images, along with ground truth with advanced deep learning techniques, to provide accurate automatic building detection. The rest of this chapter is organised as follows. Section 2.2 reviews the relevant literature: Section 2.3 describes the satel-

lite imagery dataset that will be used for the research work. Section 2.3 reviews the various image processing and machine learning techniques in the literature of automatic building detection. Section 2.4 reviews the artificial intelligence-based techniques that are available in the literature on automatic building detection. Finally, Section 2.5 examines the various parameters that have been used to compare the performance of the different methodologies available for automatic building detection.

2.2 Review of the Methods Literature

A systematic review of the literature was undertaken to explore all aspects of automatic building detection techniques. This review begins with an exploratory reconnaissance, involving a preliminary literature search that explored the methods and data sources that have been used with the satellite imagery of buildings and their details (e.g., roofs, walls, etc.). The main topics that have been identified are automatic building extraction, rooftop print extraction, and the boundaries of planar segmentation and/or extraction of building geometry. This preliminary search showed the possibility of sub-dividing the area of investigation and then selecting the most relevant aspects for a detailed systematic review. A preliminary search for relevant publications on automatic building extraction was conducted using Google Scholar, Science Direct, Scopus, and Web of Science. An advanced search of these databases was then conducted by categorising the keywords into three-word groups and combining them using the Boolean operator 'AND', as shown in Table 2.1. The search was conducted in two stages. In the first stage, keywords associated with the model, objectives, and geographical extent were used.

Table 2.1: Keywords used in the advanced search

Data source	Aims	Technique
LiDAR imaging	Terrestrial laser scanning	Superpixels
Multi-spectral imagery	Urban environment	Image segmentation
Multi-sensor fusion	3D roof reconstruction	Edge detection
Point cloud	Automatic object extraction	Colour histogram
digital terrain	Building detection	Clustering
		Deep learning segmentation
		Pattern recognition neural network

In the second stage, the model, objectives, methods, and analysis measures were applied. The initial search results from each stage were renewed by applying the following inclusion criteria:

- Publications in the English language only.
- Publications related to the reconstruction of buildings (as opposed to medical usage).
- Publications from peer reviewed scientific journals in the remote sensing field.
- The literature was updated where new papers in the above areas were published during the time the research was conducted.

In this manner, 138 papers were excluded and 219 articles were retained for the analysis. These articles were then grouped under intersections between the data source, segmentation, and object detection, as outlined in Table 2.2. Content analysis was used for all of the analyses conducted in this research. The next steps in the search were undertaken with the objective of extracting pertinent information on 3D modelling for building extraction. An examination of the titles and abstracts of the articles was undertaken to identify data sources and categorised under: (a) infrared (IR) with Red, Green, and Blue (RGB) indexed

maps; (b) multi-spectral imagery including fluorescence; and (c) LiDAR. This examination narrowed the number of articles down to 194. The final step involved a thorough reading of each article to analyse its content, focusing specifically on automatic object extraction for many different dimensions as a complete system. This step further narrowed down the number of relevant articles to 51, as shown in Table 2.3.

Table 2.2: Data sources for object- and pixel-based techniques.

Data source	Segmentation and object detection-based technique							
	2D			3D			Artificial	
	pixel-based	spatial grouping pixels	image-based	voxel-based	volume-based	object-based	neural network	Total
RGB [a,b]	[8–26]	[27–69]	[70–93]	–	–	[94]	[95–110]	99 (45%)
Infrared with RGB [c]	[111]	–	[112]	–	–	[113]	[114–120]	9 (4%)
Multi-spectral imagery	[121–123]	[124]	[125–127]	[128–130]	[131]	–	[132–135]	11 (5%)
LiDAR[d]	[136–139]	[1, 140–143]	[144–146]	[147–151]	[152–158]	[159–167]	[168–171]	30 (14%)
LiDARS+RGB	–	[172–175]	[176, 177]	–	[178]	[179]	[180]	8 (4%)
LiDARs+Multi-spectral	[181, 182]	[183, 184]	[185–188]	[189, 190]	[191, 192]	[193, 194]	[195]	13 (6%)
All sources, including LiDAR[e]	[196]	[197–199]	[200–202]	–	–	[203–206]	[207–214]	15 (7%)
Other sensors[f,g]	[215]	[216–218]	[219–221]	[222–225]	[226–228]	[229]	[230–253]	34 (16%)
Number	28 (13%)	56 (26%)	37 (17%)	11 (5%)	12 (5%)	16 (7%)	59 (27%)	219 (100%)

[a] "Image segmentation", "colour histogram", and "edge detection" were more relevant for processing 2D images.

[b] "Superpixels" are now more relevant for processing 2D images.

[c] Many references for "urban environment" were associated to a greater extent with using (indexed maps) in GIS.

[d] Many references for "terrestrial laser scanning" were associated with building construction technologies and using LiDAR as a measurement tool.

[e] This category was more relevant to "3D roof reconstruction", and the studies involved non-relevant data, such as synthetic aperture radar"SAR" images.

[f] "multi-sensor fusion" and "mutual information." were not as relevant, and could not be used separately

[f,g]"SAR" and magnetic resonance imaging "MRI" sensors were included in the "other sensors" category to demonstrate the differences in processing methods.

Table 2.3: Automatic building detection techniques that use multiple data sources.

Processing methods used bases in Discontinuity[d] or Similarity[s]	**Spectral (colour) or Texture map**	**Multiple images with Digital Elevation Modelling (DEM) and Digital Terrain Modelling (DTM)**	**LiDAR** height map	**Publications**
Global thresholding	✓	✓	✓	[254–256]
Variable thresholding	–	✓	–	[257]
Multiple thresholding	✓	–	✓	[258–262]
Canny edge detecting	✓	✓	–	[263, 264]
Edge detection multi-methods	✓	✓	–	[265, 266]
Region growing methods	✓	✓	–	[267–271]
Splitting and merging methods	✓	✓	–	[272–276]
Hard clustering	–	✓	✓	[277–280]
Soft clustering	✓	–	✓	[281–283]
Watershed	–	✓	✓	[284, 285]
Partial differential equation	–	–	✓	[286]
Artificial neural network	–	✓	✓	[287–292]
Convolutional neural network-based	✓	✓	✓	[3, 7, 293–302]

All of the above results could be categorised in two main common factors, which are: the first is that all publications used satellite imagery as their primary data sources; the second is that all pixels classification methods can be classified as discontinuity-based or similarity-based. The literature was further narrowed down by focusing on studies that use VHR remote sensing images. The following section will present different aspects of satellite imagery, accompanied by well-known methodologies for pixel classification.

2.3 Satellite Imagery Data

Satellite images have been accessible by the general public since 1972, starting with the launch of Landsat-1, which was initially known as ERTS-1 [303]. Various types of sensors have since been introduced. SPOT-1 HRV was launched in 1986 and is still one of the most significant optical sensors from the perspective of survey and mapping because it enables stereo-coverage. Furthermore, SPOT-1 HRV was designed to support topographic mapping along with 3D map construction

[303]. IKONOS was launched in 1999. A number of imaging satellites have since been launched that provide images with even better resolutions. The WorldView-3 state-of-the-art satellite was launched in August 2014, which allows images with a resolution of up to 31cm to be taken. These improvements have allowed satellite imagery to become widely available for many uses, including urban planning, and mapping and 3D modelling of cities. Furthermore, due to their increased availability, competition has increased among the providers of those images, leading to lower costs for the users.

Pixel size determines spatial resolution. Furthermore, when selecting satellite images for a specific application, it is necessary to reconcile the trade-off between spatial and temporal resolution requirements [304]. For instance, a high temporal resolution is critical in emergency situations such as hurricanes and landslides because emergency situations can quickly change, necessitating frequent observations throughout the day. Meanwhile, planning applications for urban infrastructure changes require spatial understanding over a longer time period [305], so annual observations may suffice. Nonetheless, both of these examples may necessitate high spatial resolution images to allow for a thorough examination of these processes [306]. In other cases, such as monitoring rapidly changing situations like the weather, a high temporal resolution is required. As a result, operational weather forecasts require high temporal resolution satellite observations, even if this reduces spatial resolution. As a result, each remote sensing application has its own set of resolution requirements that must be met [307].

While the terms used for specific resolutions differ, a description of some the most commonly used terms follows; as put forward by reference [303]. Satellite image classification considers the resolution required, including low resolution of $\geq 30m$ and $< 300m$, medium resolution of $\geq 5m$ and $< 30m$, high resolution of $\geq 1.0m$ and $< 5.0\ m$, and VHR of $< 1m$. Table 2.4 illustrates some of the high resolution satellite optical image sensors that are now available for use. These sensors pro-

Table 2.4: Commercially available satellite imagery and sensors

Satellite	**Bands** **Pan[1]/MS[2]**	**Resolution** **Pan/MS(m)**	Launch Date	**Organisation**
IKONOS	1/4	0.82/3.2	1999	GeoEye Inc., (DigitalGlobe Inc),USA
Quick Bird	1/4	0.61/2.44	2001	DigitalGlobe, (DigitalGlobe Inc., 1992),USA
WorldView-1	1/-	0.45/-	2007	DigitalGlobe, (DigitalGlobe Inc.),USA
GeoEye-1	1/4	0.41/1.65	2008	GeoEye Inc.,(DigitalGlobe Inc.) ,USA
WorldView-2	1/8	0.46/1.8	2009	DigitalGlobe, (DigitalGlobe Inc.),USA
Pleiades-1A+&1B	1/4	0.7/2.8	2011	(Astrium Services) Airbus Defence and Space, France
KOMPSAT 3	1/4	0.7/2.8	2012	Korea Aerospace Research Institute, South Korea
SkySat-1	1/4	0.9/2.0	2013	Skybox Imaging, USA
SkySat-2	1/4	0.9/2.0	2014	Skybox Imaging, USA
WorldView-3	1/28	0.31/1.24	2014	DigitalGlobe, USA (DigitalGlobe Inc.)

[1] "Pan :Panchromatic "

[2] "MS : Multispectral."

vide imagery of the Earth's surface by sensing reflectance in both the visible and near-visible regions of the electromagnetic spectrum[1]. While only passive sensors are used, this table clearly shows that the maximum resolution of civilian use satellite imagery has now reached 0.31m, which was achieved by WorldView-3. Given that VHR satellite images have been used to conduct this current research, this is the main focus of the following table, despite the availability of higher resolution data from other sources (e.g., aerial imagery or LiDAR[2]).

A single band can cover the visible parts of the electromagnetic spectrum. In a double band spectrum for remote sensing, there are three or more bands in the narrow wavelength range within the visible and near visible range, including infrared and parts of the electromagnetic spectrum [309]. Satellite imagery is beneficial for the extraction of the required information regarding urban segmentation [310]. A wide range of free satellite imagery sources are available, such

[1] A multi-spectral image is made up of several channels, each of which contain radiation measured in specific wavelength ranges for each pixel [308], and LiDAR.

[2] involves using sensors that transmit energy in a narrow frequency range as shown in Table 2.3.

as USGS, LandViewer, and NASA Earthdata Search. Satellite images are a rich data resource because they provide three bands or more of representation of all of the layout, including buildings and trees. In addition, many other useful aspects are revealed in these images (e.g., shadows), which can be used as features for the 3D representation of objects in the image [311].

The complexity of building recognition from satellite images can be attributed to the wide variety of building shapes and sizes, various roof covering materials, the existence of multiple objects of similar shapes on the satellite image, and also roof overhangs that hide the actual building edge and interference from surrounding objects and their shadows [311]. Acknowledging these issues, this research aims to explore the development of effective building segmentation and recognition methods using VHR remote sensing satellite images. The results will provide a solid basis for further research in this area by shedding light on the level of detail that is most suitable for automatic object extraction, building detection, and 3D roof reconstruction in urban environments [312].

The dataset that is used in this research work comprises VHR satellite imagery that is taken using state-of-the-art approaches, because these types of images are typically used due to their high resolution, which includes all details in the pixels, as well as spatial information location data, for example, true orthophoto (TOP) These images represent an area of Vaihingen (a city in Germany), and was taken by the German Association of Photogrammetry and Remote Sensing Digital Government Policy Framework (DGPF). These datasets consist of very high resolution true orthophoto (TOP) tiles and corresponding digital surface models (DSMs) derived from dense image matching techniques. Furthermore, unlike normal images, these are Near Infrared Region (NIR) images and differ from normal RGB images because they contain a near infrared channel instead of the normal blue channel. There are 33 images in the dataset, which are known as tiles. Each tile represents a major area of the city. The tiles are roughly

2500×2000 pixels and have been taken with a ground sampling of uniform distance of $9cm$.

2.4 Image Processing Techniques for Automatic Building Detection

Several approaches can be used to solve the problems associated with rooftop detection and building reconstruction segmentation. One of the most common involves the detection of linear features in various images, the subsequent comparison of these features, and the construction of a 3D model based on the results of the comparison [266]. This approach can provide excellent results if information from multiple images can be collected, it simply requires several images of the same area. However, alternative approaches are needed because in many cases it is not possible to collect such images, including because observation satellites can have fairly orbital periods.

Another approach involves using the structural, contextual and spectral information of the image [256, 257, 260, 265]. This image processing technique approach assumes that only one image is used to detect buildings. Contour detection algorithms are used to achieve the required result, which involve various modifications to filter [308, 313]; in fact, the last two infamous methods can both be used for edge detection. In addition, the colour and brightness of individual elements can be analysed to further increase the accuracy.

Some algorithms find and conduct shadow analysis, which can lead to high accuracy. However, their accuracies are low when compared to methods based on LiDAR data analysis. Consequently, the use of these algorithms is not advised for the 3D reconstruction of buildings. Another approach for the automatic reconstruction of 3D building roof models has been presented by integrating airborne

LiDAR data and optical multi-view aerial imagery [144]. A coarse-to-fine LiDAR data segmentation is proposed to separate a building's LiDAR points into a set of roof planar segments, which involves initial segmentation using normal point estimation and segmentation refinement. A point-based integration mechanism is then proposed for 3D step line determination by incorporating the segmented rooftop points and 2D lines extracted from the optical multi-view aerial images, with which 3D roof models are reconstructed.

Another approach focuses on building a dense height matrix using digital terrain modelling DTM and DEM. This requires the use of several images. In this case, the height-map can be estimated directly, with sub-millimetre accuracy. Algorithms based on this approach provide a high percentage of object detection and a low probability of false positivesv[261, 277]. For machine learning, the trend is usually that a higher percentage of object detection is more than 90% percent; in addition, regarding the main matrix to compare the results when the false positive is measured and found to be low, the accuracy will be high, and vice versa. The details of the matrix used are presented in Chapter Three.

As mentioned earlier, several segmentation methods can be used to automate the extraction of buildings by drawing on multiple data sources; which are shown in Table 2.2.

Threshold and clustering are the most commonly used techniques in the similarity approach for LiDAR data, such as in [142]. These approaches have been taken into consideration for automatic building detection and localisation. They have also been compared and evaluated using high-spatial-resolution images and LiDAR data with threshold-based to classify objects. The thresholding-based approach is based on two threshold values: the first refers to the minimum height to be considered as a building, which is defined using the LiDAR data; and the second refers to the presence of vegetation, which is defined by the spectral response. Object-based classification follows a standard object-based image classification

scheme, such as segmentation and clustering, to extract the target features.

Following the detection process that was explained earlier, the DTM and DEM can be processed to detect and reconstruct buildings. However, such generalised methods for image extraction [267, 271, 289] provide rather inaccurate results. In addition, the construction of an exact elevation matrix implies a complete orientation of the images. Furthermore, the construction of the exact elevation matrix means that the images are fully oriented (i.e., all the points correspond between the image time requirements).

Automatic building detection using different sources of data (e.g., LiDAR, DTM, DEM, multi-spectral data or fusion data) requires huge computation. In fact, achieving 2D and 3D reconstruction needs different sources of data, which requires high GPU power and computational time.

While many feature extraction techniques are available for 2D image processing, Table 2.2 describes some of the 2D feature extraction and segmentation techniques that are used in the automatic building detection literature.

2.4.1 Discontinuity-based Approaches

In a discontinuity-based approach, "the partitions or sub-division of an image is based on some abrupt changes in the intensity level of images "[314]. Furthermore, discontinuity-based segmentation usually involves either isolated points detection, line detection or edge detection. Point detection is the simplest form of discontinuity that is found in digital images, and these discontinuities can be detected by running a mask over each point on a particular image [314] as can be seen in Figure 2.1. Line detection is the next complex discontinuity approach, which involves the use of two masks that will reveal the lines of pixels in a specific direction. In other words, by using a line detector mask, it is possible to discover all of the lines oriented in a certain direction that are one pixel thick. However,

and most relevant to the current research, edge detection is the most commonly used approach and it is very useful for detecting discontinuities in an image.

For most practical applications, isolated points and lines with the same pixel thickness are rare; that is, in most cases finding an isolated point or line with the same pixel value is almost impossible when referring to the image intensity and contrast. Therefore, edge detection is commonly used for grey level discontinuity segmentation. An edge forms the boundary between two areas with distinct levels of intensity. For example, if an image changes from dark to white, then the changes in intensity (first-order and second-order derivatives) can be examined to find the intensity and gradient of the edge [314].

$$\sum_{i=0}^{9} w_i \; z_i \tag{2.1}$$

where:
- w_i is filter weight ;
- z_i is pixel magnitude

Edge detection and line extraction algorithms: Edges are areas where the goal is to identify points in an image in which the image brightness changes sharply or edges can characterise boundaries, and are therefore a problem of fundamental importance in building detection. Edges in images are areas with strong intensity contrasts, with a jump in intensity from one pixel to the next.

- The “Canny edge detector "[263, 264] is the most popular edge detection technique and it uses a multi-stage algorithm to detect a wide range of edges on images. It was developed in 1987 by John F. Canny. For the Canny algorithm, a Gaussian filter is first applied for smoothness [79], and then magnitude of the equation

z1	z2	z3
z4	z5	z6
z7	z8	z9

(a) sub-image

-1	-1	-1
-1	8	-1
-1	-1	-1

(b) detection point

-1	-1	-1
2	2	2
-1	-1	-1

(c) horizontal line

-1	2	-1
-1	2	-1
-1	2	-1

(e) vertical line

-1	-1	2
-1	2	-1
2	-1	-1

(f) diagonal +45

2	-1	-1
-1	2	-1
-1	-1	2

(g) diagonal −45

Figure 2.1: Discontinuity approach common masks used to detection objects.

The Canny edge detection algorithm's process can be broken down into five steps:

1-Use a Gaussian filter kernel of size $2k+1$ $2k+1$ is given by Equation 2.3 to smooth the image and remove noise . 2-Identify the image's intensity gradients calculated by Equation 2.4. 3-Use gradient magnitude thresholding or lower bound cut-off suppression to eliminate spurious edge detection responses calculated by Equation 2.5. 4-Use a double threshold to identify potential edges. consequently Track edge by hysteresis to complete edge detection by suppressing all weak edges that are not connected to strong edges.

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{2.2}$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.3}$$

Equations 2.2 and 2.3 from [79] show the Gaussian distribution in the 1D and 2D cases, where σ is the distribution's standard deviation. The idea

behind Gaussian smoothing is to use these distributions as a point spread function to create a filtering mask, and then blur an image using convolution. Since images are typically stored as discrete pixel values, a discrete Gaussian function estimation on the filtering mask would be required before conducting the convolution.

$$m = \sqrt{G_x^2 + G_y^2} \tag{2.4}$$

$$\theta = arctan\frac{G_y}{G_x} \tag{2.5}$$

where:

- m is the magnitude and direction of gradient ;
- G is Gaussian filter ;
- G_x direction coordinates of the point being considered in$X$;
- G_y direction coordinates of the point being considered in Y;

After that, no maximum suppression is applied to press out the pixels that are not maximum. This results in edge pixels being provided.

- The "Hough transform-based edge detector "[313] is a well-known edge detection technique, which works on the principle of transforming the spatial image data into parametric data. It then uses a voting scheme to select the regions in the parametric space with the majority of points available. The simplest form of the Hough transform finds lines and works by converting the image pixels into the parametric space of (θ and p). As shown in Figure 2.2, three points have fallen on a single line available on image spatial (x,y) plane maps on a single point (p_0,θ_0). Similarly, all of the points falling on the line in (x,y) are mapped onto a single point (p_0,θ_0).

- Mean shift segmentation [301]: This method is similar to region split and merge, which converts the image into homogeneous tiles based on the closest neighbour's pixel values and calculates the similarity inside each pixel group.

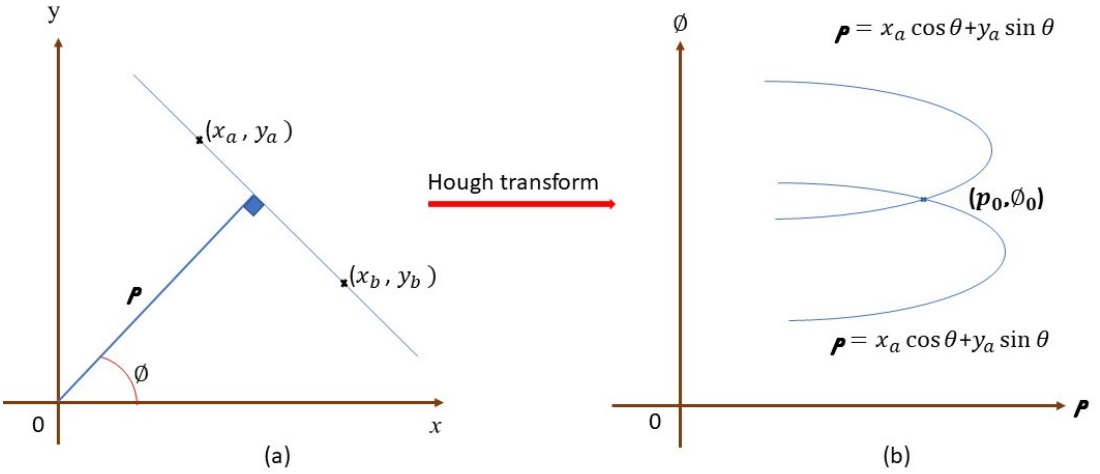


Figure 2.2: Hough transform: (a) Cartesian coordinate, and (b) polar coordinate system parameter.

This method is based on defining a window with a specific circular size centred on a specific pixel in the image. The defined window is shifted based on the mean value of the pixels' intensity in the window. The calculated mean value will be used to find the magnitude and direction for shifting the original window for the next iteration. This is repeated until the result converges and the vector value change is so small that it does not need, effectively, to be shifted any more. Finally, the mean colour of the final iteration will be assigned to the starting location of the window's initial location. The previous operation tries to smooth the image by finding the mode inside a specific local region in the image, and it later applies the mode value to the whole specific region. This technique leads to better results in segmentation than using a thresholding method because it maintains the edges of the objects and removes the noise that is not related to the building.

- Corner Detection Algorithm [297]: In images, a corner is a junction of two edges, where an edge is a sudden change in image brightness. The corner detection algorithms, such as the Harris corner detector, assume that corners are related to local maxima at each pixel of the image. If the

local maximum is higher than a certain threshold, then the pixel is declared a corner.

- Enhanced snake model [112]: The snake model, also known as active contour modelling using energy minimising, is a deformable spline that is influenced by constraint and image forces that pull it towards object contours and internal forces that resist deformation. Snakes may be understood as a special case of the general technique of matching a deformable model to an image by means of energy minimisation. In 2D, the active shape model represents a discrete version of this approach, which takes advantage of the point distribution model to restrict the shape range to an explicit domain that is learnt from a training set. The enhanced snake model works based on a dot that is selected within the object to be segmented using the snake convergence techniques, along with providing a means to convergence to the concave, as explained in the next section.

2.4.2 Similarity-based Approaches

Similarity-based approaches facilitate the generalisation of several machine learning and pattern recognition methods, and can support the development of a framework for classification. Furthermore, these approaches focus on a group of pixels with similar correlation characteristics. While discontinuity focuses on a single area of interest and examines the neighbouring pixel to build up a picture, similarity-based approaches start with a group of pixels that have some common correlation regarding the area of interest (e.g., pixel value, colour or location). A brief description of these approaches follows:

- “Pixel grouping process" [268, 270] is a fundamental technique that works on connecting or grouping pixels that have similarity in one object. The

similarity of pixels is decided based on the similarity of a pixel's intensity values compared to its neighbouring pixels' intensity values.

- "Connected pixel analysis" [239] is a technique that looks at four or eight neighbours of the central pixel and then puts the neighbouring pixel into a set of similar pixels. In this way, various sets of pixels are grouped which have similar intensity values. Similar values of pixels are usually available on building boundaries. Hence, this technique is used to determine the boundaries of the buildings.

- "Region growing approach" [267–271] is a region-based image segmentation, also known as pixel-based image segmentation. It examines neighbouring pixels for initial seed points and then calculates whether or not those pixels can be added to the region.

- "Region splitting and merging method" [272–276] is a technique in which an image is first subdivided into a set of arbitrary disjointed regions. The regions are merged back if some similarity criteria are met. However, if the similarity criteria are not met, then the regions are further split into more small regions. A similarity check is again done and a decision of merging the regions or further splitting into smaller regions will be made. The same method is repeated until various regions having similarity check satisfied are procured. In the case of buildings, most of the roofs will have similar regions. Hence, after applying this method, roofs regions will be extracted.

- Region growing by voting [286] is a method that relies on a line extraction algorithm, such as the Hough transform (as explained in the discontinuity approach). However, it works differently than the pixel grouping method. The pixel grouping methods fail on high resolution images, such as IKONOS. In the voting strategy, the algorithm focuses on the lines of the buildings by taking a point on the roof of the building as a starting point and then defines a small area around that point. The lines near the starting

point are extracted using a line extraction algorithm and then, based on their location and orientation, 'voting' is used to opt out the best line to be used for building extraction.

- Watershed method [284, 285] uses a greyscale image that is based on catchment basins and watershed lines.

- Morphology operations, such as differential morphological profile (DMP) are also used in building detection [133]; DMP is used to supply image structural information. Information related to the building's assumed size and position is inferred from the DMP. The DMP is also used to detect the building's shadow, which is consequently used to provide appropriate data about the proposed size and position of the related building.

- "Clustering" is an unsupervised method that is used to classify the pixels of an image into multiple determined numbers of classes. In hard clustering, each pixel is allocated to one particular class, whereas in soft clustering a probability is set for each pixel for every class. The pose clustering method [315] performs object recognition by determining hypothetical object poses and finding clusters of the poses in the space of legal object position. This method is based on a voting process in which the majority position of components in an edge are used to determine the position and location of that edge. Pose clustering is also called hypothesis accumulation and generalised Hough transform, which is characterised by a "parallel" accumulation of low level evidence followed by a maxima or clustering step that selects the pose hypotheses with strong support from the set of evidence. Various methods can be used for the building extraction to be achieved.

  For example, the following method has been used: first, effective segmentation of the building region was used to extract the components of a building roof's outline from its background; and second, pose clustering is then

used to adjust the direction of roof outline components and building corner locations[280].

Also within the scope of clustering, the following approaches can be used:

- Threshold-based approach : The main process tools in similarity approaches are briefly described in the following:

1 “Global thresholding" [254–256] is used to produce a binary image by defining a pixel intensity threshold: values below the threshold are changed to zero and above the threshold set as one.

2 Variable thresholding [257] is a kind of adaptive thresholding in which the value of the threshold varies for every sub-image; whereas multiple thresholding marks multiple levels of thresholds, which sets the same value for every pixel between certain thresholds.

3 Spectrum thresholding works in the frequency domain. Basically, the spatial information available in the image is converted into a frequency or multi-resolution spectrum with the help of Fourier analysis [316] or wavelet transform based approaches [217, 218]. The power spectrum is computed from the frequency domain and a threshold is used to separate out different spectrum having similarity using filters. The filtered spectral information is then transformed back in to spatial images and can be used further for object detection.

At the end of the processing techniques for automatic building detection section, there is a comparative analysis of various image segmentation techniques based on the discontinuity-based approach and similarity-based approaches. To evaluate these two approaches, with the state of the literature with respect to an important aspect:

- The performance of the algorithms or methods, in terms of computational power and detection features accurately.
- The performance with complex images as well as complex data sources.
- The simplicity of using them in computer vision and applying post-processing manipulation with machine learning outputs.

2.5 Artificial neural networks (ANNs) in Remote Sensing: An Overview

In recent years, object detection tasks have become widespread. One explanation for this may be the highly effective meta-algorithms that have resulted from AI developments over the past decade, which have provided a foundation for several further developments, such as CNN, FCN [4], Region-based Convolutional Neural Network (R-CNN) [317], and so on. In the case of semantic segmentation, the same reasoning applies. The convolution neural network from the FCN system can be used it as a meta-algorithm for this problem. Since the publication of the fully convolutional network FCN in 2015, nearly all semantic segmentation methods have been based on some form of a fully convolutional network.

Given the strength of meta-algorithms, this thesis will demonstrate some of the methods (e.g., segmentation) that will act as a meta-algorithm in terms of speed, accuracy, and simplicity. These methods may be used in both future research of instance segmentation and in many other applications of instance-level understanding.

Over the last few years, instance segmentation approaches have been commonly classified into two groups. The first group includes the popular R-CNN methods, which typically begin with a proposal for segmentation levels. Classifiers are then trained to classify these proposals into somatic classes. The second group includes

the successful FCN methods. They typically begin with full image somatic segmentation results and then learn a particular split to divide them into individual instances.

What exactly is Mask R-CNN? Simply, it is the best combination between the two groups. Mask R-CNN is a fast R-CNN method that incorporates a FCN on each region of interest (RoI). As a result, it serves as an algorithm for a complex problem. Another advantage of Mask R-CNN is the aligning region of interest (RoIAlign) process, which is an improved version of the regular region of interest pooling (RoIPool) operation. The RoI-align operation does not struggle with condensation. Naturally, the regions must be mapped onto the future map, and then linear interpolation must be used to extract a fixed-dimensional output. Because there is no condensation, no information is lost in this phase. In contrast to the popular RoIPool operation, the original RoIPool operation was not engineered for segmentation. Because it is designed for object detection bounding box, some pixel-to-pixel precision has been lost, such as in the case of using GAN. In addition, theRoIPool process divides the image into pixel-by-pixel alignment due to the pooling, which results in some information loss. The `ROI-Align` operation has resolved these issues (e.g., segmentation). Another aspect of Mask R-CNN is that FCNs are naturally constructed in a pixel-to-pixel alignment fashion due to fully convolutional head for predicting masks.Recent research has shown that convolution neural feature representations are highly effective in large-scale image recognition, object detection, and semantic segmentation [318–320]. A few extra convolution layers on each RoI helps to predict a very accurate mask and simultaneously helped to resize these regions to identify objects with a softmax classifier. Mask R-CNN is a meta-algorithm that can support several implementations in various fields and applications that, due to the Mask R-CNN, can be compatible with many other advances, such as highlighting mining and many other developments to the backbone architecture such as ResNet or residual neural network NeXt (ResNeXt) has been used as the feature backbone [321], and feature

pyramid network (FPN) has been used as the backbone in [322]. Mask-RCNN achieved instance segmentation contribution results when used on the COCO dataset, which are comparable to the first three winners of the cocoa competition in the previous three years. Given that Mask-RCNN is a meta-algorithm, it can easily support the improvements of their architectures, such as replacing the features from ResNet to ResNeXt [321]. The Mask R-CNN framework also enhances object detection. The results of bounding box detection on the COCO dataset are shown. When comparing RoIAlign activity to RoIPool with FPN as a reference, RoIAlign increases by about one point. Due to multitask learning, the training Mask R-CNN and the bounding box will see another round of one-point improvement. However, some images are extremely difficult to segment because (for example) several objects are surrounded by other objects in the same group, which can be a complex case for semantic segmentation.

Disconnected objects are another problematic case that Mask R-CNN faces, which can pose a problem for those mass loads that are dependent on grouping. Meanwhile, Mask R-CNN effects can detect and segment tiny objects in another example. After seeing all of these outcomes, one may believe that rivalry has been resolved, but this is not the case. There are also many failure events. For example, the detection of masks can continue to fail if segmentation detection continues to fail, then the detection of segmentation may also fail. Missing items and false masks are another possibility. More importantly, understanding is not solved; for example, a building may look similar to a large truck when viewed from above. Mask R-CNN can easily be expanded to perform additional tasks, such as object keypoint detection. As a result, the extension makes it much easier to display a specific object's keypoints as a single hot mask or ground-truth mask, and then the Mask R-CNN framework can be used to execute this objective. In other words, this approach is a single framework that can support bounding box detection, Mask segmentation, and keypoint detection, which meets the remote sensing requirement mentioned earlier. The previous sections highlight evaluative short-

comings with regard to spatial fusion, remote sensing, and big data for urban modelling. In general, this review examines the current state of the research in this area to identify the knowledge gaps and propose future research directions, such as whether or not it is advisable to solve the problem using ANNs or machine learning. Machine learning is a process through which a machine becomes capable of predicting required outputs for a given input, and ANN architecture is used in the learning process. Overall, each ANN architecture consists of a number of layers. Each layer consists of multiple nodes, known as neurons, that are connected with the following layer through weights. These weights are optimised to the input map with a given output label during training or learning. In 1958, the psychologist Frank Rosenblatt invented the first ANN, known as the Perceptron, which was intended to model how visual data is processed by the human brain and how object recognition is learned. The Perceptron generated a lot of excitement in machine learning research and related industries thanks to many breakthrough results in areas such as computer vision (an AI field which trains computers to interpret and comprehend the visual world) and image processing (a method of performing certain operations on images to extract some useful information from them). Although the concepts of ANN and machine learning have been around for a long time, they were not useful initially because of hardware limitations. However, thanks to the advances made in hardware from the start of 21st century, and especially after 2010, it is now possible to implement these complex architectures. graphics processing Units (GPUs) can apply parallel processing to a task to speed up the process and reduce the computational time. In addition, hardware components can have huge amount of random access memory random access memory (RAM).

2.5.1 Artificial Neural Networks (ANN) and Deep neural networks (DNN)

Recently, computational advances, coupled with huge dataset availability, have brought a promising new option to the ANN family, called DNN. Basically, DNN is a subset of ANN. While ANN can be shallow or deep depending on how many layers between input and output, DNN has multiple hidden layers (thus the expression deep);see the flowchart in Figure 2.3. Due to the availability of this big data, efficient parallel processing hardware and parameter optimisation techniques, DNN have shown promising potential in a variety of machine learning tasks(e.g., pattern recognition). Furthermore, 2D and 3D feature extraction and classification techniques have been used as supervised or unsupervised methods using deep learning. Single or multi-data types and prepossessing have been used (e.g., data-augmentation, RoI extraction, image cropping, etc.). The feature maps are directly fed into a softmax classifier for classification before receiving the final outputs.

2.5.2 Convolutional Neural Network (CNN)

CNN have been one of the most influential innovations in the field of computer vision. They have performed much better than traditional computer vision and they have produced state-of-the-art results. These neural networks have also proven to be successful in many different real-life case studies and applications, such as image classification, object detection, segmentation and face recognition. Figure 2.4 shows a CNN, which typically has convolution layers, max pooling layers and ReLU layers and a FCN for classification.

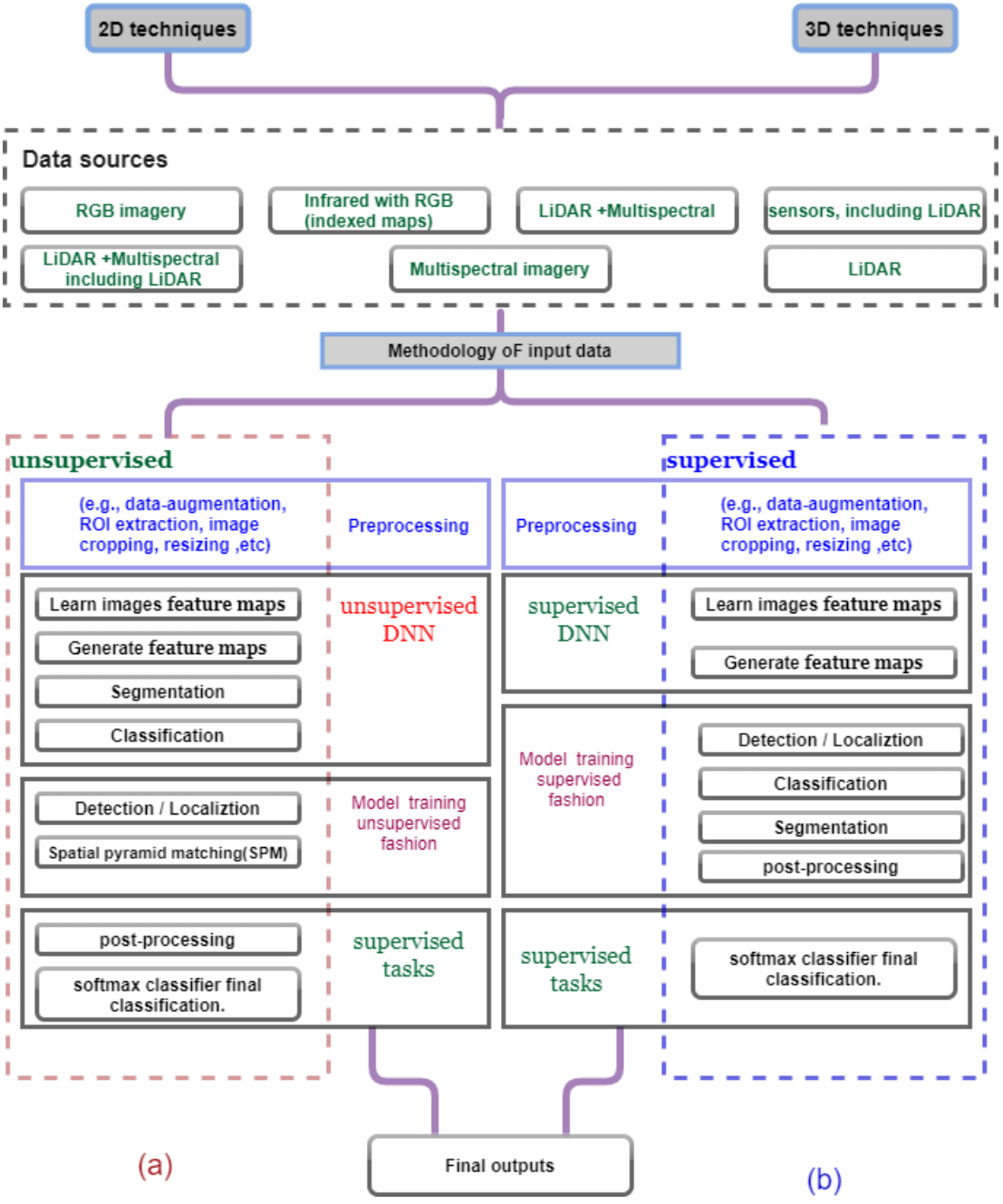


Figure 2.3: Flowcharts of DNNs showing 2D and 3D feature extraction and classification techniques using: (a) unsupervised method, and (b) supervised method.

2.5.3 Region-based Convolutional Neural Network(R-CNN)

R-CNN is another technique that can be used for parts of images as an input compared to a whole image.

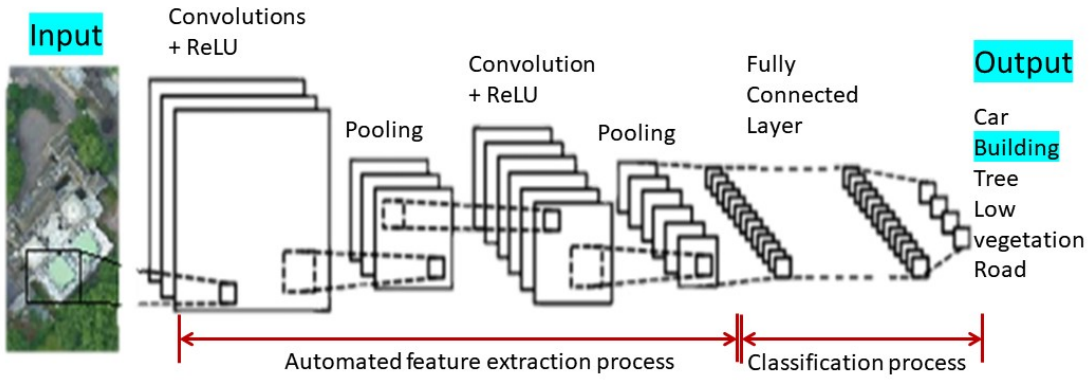


Figure 2.4: Typical CNN architecture

Further improvements have been made to R-CNN, namely fast R-CNN and faster R-CNN. All of these techniques can be used for object detection, and therefore for roof extraction from images. With the rapid improvement in methods for object detection, R-CNN are supervised learning algorithm models that are used for computer vision; more precisely, object detection. The primary purpose of creating R-CNN is to produce bounding boxes that are based on the characteristics class (e.g., car or buildings) that can identify an object segmentation, classification, and localisation. Over the last few years, various R-CNN architectural versions have been published. The following sections discusses some of the leading architectural contributions that have been published.

- In November 2013, R-CNN starts by extracting RoI from an input image using Selective Search. Each RoI is a rectangle that can represent the boundary of an object in the image. However, there could be up to thousands of RoIs depending on the model's size. Each RoI is then loaded into a neural network and generates an output features map. A variety of support vector machine classifiers can be used to specify the type of object found within each RoI's detected output features[296].

- In April 2015, Fast R-CNN was developed by Ross Girshick et al. [297]. While the original R-CNN independently computed the neural network fea-

tures on each of RoI, which is enormous in many cases, Fast R-CNN runs the neural network input once on the whole image by using selective search to generate the RoI. At the end of the network is a new method called RoIPool, which slices out the generated RoI from the network's output tensor, and then reshapes and classifies it. This manner of R-CNN and FAST R-CNN can be found across the whole input space, using selective search to determine RoI. However, the drawback of selective search is that it is a slow and time-consuming operation, which affects the network's performance.

- In June 2016,[3] developed an object detection algorithm that lets the network learn how to detect and generate the RoI by itself. Instead of using a selective search algorithm, they named this method Faster R-CNN.

- The final and main variation of the R-CNN family resulting from a recent summit is called Mask R-CNN [290]. Object detection is a computer vision task that involves localising one or more objects within an image and then classification of each of these objects in the image. However, this challenging computer vision task requires both auspicious object localisation to locate and draw a bounding box around each object in an image and object classification to predict the correct class of the localised object. An extension of object detection marks the specific pixels in the image that belong to each detected object instead of using rough bounding boxes during object localisation. This more complex version of the problem is usually referred to as object segmentation or semantic segmentation.

Mask R-CNN builds on top of this by adding a third head, where it performs classification and localisation. It also creates the instance segmentation mask for a given RoI, which helps to vector and separate the pixels surrounding the boundary of objects, and allows the keypoint to be extracted. The following section analyses this in more detail.

2.5.4 Mask R-CNN Architecture

Mask R-CNN builds upon the idea of Faster R-CNN. In the case of Mask R-CNN [290], the first stage is identical to that of the Fast R-CNN. Here, the network uses Region Proposal Network (RPN) to propose possible regions where objects are likely to be present. In the second stage, along with bounding box and object classification, the network uses FCNs to generate the segmentation mask for each instance of the objects. For these segmentations, the network employs a special operation called RoIAlign, which is responsible for generating pixel-to-pixel instance segmentation. Each of these masks are of size $m \times m$, thus allowing the network to preserve spatial information about the objects. The loss on this network is now on each of the sampled RoI, and is given as:

$$\text{Loss}(L) = L_{cls} + L_{box} + L_{mask} + L_{kps}, \tag{2.6}$$

where, Loss(L) total loss, (L_{cls}) classification loss, (L_{box}) bounding-box loss, (L_{mask}) segmentation mask loss and (L_{kps}) keypoint loss.

2.5.5 Generative Adversarial Network (GAN)

As discussed previously, there is a need to automate detection and reconstruction. Deep learning can play a vital role in addressing this need. In 2014, a deep learning technique known as GAN was introduced by Ian Goodfellow [298] and his colleagues [292]. In this system, two neural networks competed with each other in a game. This technique learns to generate new data with the same statistics as the training set. For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to the human observer. Although initially proposed as a form of a generative model for unsupervised learning, it has also proven useful for semi-supervised learning, fully supervised learning and reinforcement learning. There are two parts to the GAN system, the generator and discriminator, and both indulge in a game such that the generator

rates the fake image and the discriminator checks whether or not it is fake. The generator optimises itself to generate images as close to the label as possible using a supervised approach, and the discriminator optimises itself to distinguish between fake and real images. This optimisation goes on until the generator starts generating images close to reality, which are not easily distinguishable by the discriminator as fake. A deep learning approach to attain reconstruction in an urban environment using GAN is very effective, as shown in Figure 2.5.

a b c

Figure 2.5: Sample from Vaihingen area 32 showing: (a) top mosaic (b) ground truth and (c) the output using *pix2pix GAN*.

In simple terms, the discriminator is responsible for classifying the contents from the generator as fake and the actual data/examples to be real. The job of the generator is to generate samples that can fool the discriminator.

To mathematically present this idea, the generator takes a sample distribution $\boldsymbol{z}$ and creates a sample $x = G(z; \theta^{(g)})$ that is parameterised on feature θ. The role of the discriminator is to predict the probability $D(x; \theta^{(d)})$, which quantifies the chance that x is real data.

As shown in Figure 2.6, new GAN papers are being published nearly every month and researchers are becoming increasingly inventive in their approach to naming them. This is the product of an intriguing experiment in which GANs were compiled.

This thesis will use different variants of GANs, which are dependent on the loss function and the way that they are formed. The methodology chapter will describe the *pix2pix GAN* [299] and *SAGAN-based* [300], which will allow the segmentation to be performed.

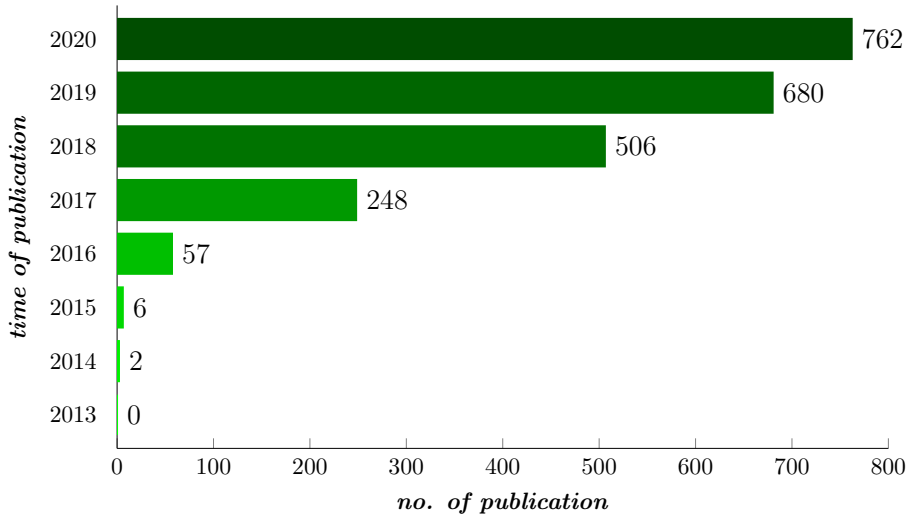


Figure 2.6: Cumulative numbers of published papers contained generative adversarial network (GAN) [323].

2.5.6 A Critical Review of the State-of-the-Art in AI Architecture for Object Detection

Object segmentation based methods lack the ability to provide semantic information. Recently, many AI methods have been applied to provide better feature extraction techniques, which can help to extract semantic building information from satellite building images [324]. To extract good semantic and instance information, it is important to extract accurate geometrical information along with accurate directional information. The following paragraphs highlight the most famous examples of object based segmentation in timeline order:

Zhao et al. [6] has mainly focused on classifying the different types of classes (e.g.,

car, buildings, tree, etc.) that are available in the image simultaneously. This research has compared the results with FCN -based methods and concludes that FCN alone is not suitable for building detection. Therefore, it proposes detecting pixel wise features using CNN, and then detecting edges using spectral domain segmentation and contour detection. It combines the pixel-wise features for detecting CNN and contours these to generate edge preserving semantic segments. A contextual graph model was built to capture the interrelationships between different classes of objects. This work involved a contour-preserving strategy to combine semantic labels and objects' shapes, thus generating semantic segments. This paper reports accuracy and kappa coefficients for Vaihingen and Beijing datasets. In addition, it focused on multiple class classification by relying heavily on simple statistics at the pixel-level, which is probably not suitable for the recognition of complex objects such as buildings.

An end-to-end study was conducted on trainable deep convolutional neural network (DCNN) for semantic segmentation with built-in awareness of semantically meaningful boundaries [7]. The proposed method overcomes the issue of the loss of high-frequency details by combining semantic segmentation with semantically informed edge detection, thus making class boundaries explicit in the model. This research proposes to use different combinations of networks and has provided results for three different DCNN architectures, as follows: (1) boundary detection added to the SEGNET encoder-decoder architecture, 2) boundary detection added to FCN-type model, and 3) setting up a high-end classifier ensemble. Their best model achieved more than 90% overall accuracy on the International society for photogrammetry and remote sensing (ISPRS) Vaihingen benchmark. Although this research has provided a guideline for using building boundary information in finding meaningful boundaries, it utilised DEM information along with image information. In addition, the proposed architectures are bulky and need to be improved.

A recent study [325] has applied Mask R-CNN for building segmentation which improved the statistical analysis of binary classification, such as F1-Score, Intersection over Union also known as Jacard coefficient (IoU) and Precision, when compared with other segmentation architectures, such FCN or You Only Look Once (U-Net). Meanwhile, another study by Pham et al. [301]has focused on extracting semantic features using keypoint. This study considers segmenting an individual building by detecting several keypoints. The keypoints that are detected are subsequently reformulated as a closed polygon, which is the semantic boundary of the building.

A building is considered to have a set of keypoints [325, 326]. In the first step, the feature maps are extracted using CNN, followed by the RPN, which slides over the feature maps to generate candidate bounding boxes where buildings may exist. For each bounding box, local features are acquired using RoIAlign. FCN is then applied to the features, and it predicts the heat map of the keypoints. Once the keypoints are extracted from the heat map, they are grouped into boundaries in a geometric way. Finally, the buildings of interest are delineated with these boundaries as a polygon map.

In addition another research study has proposed a three step process, which is termed the multi-level context-guided classification (MLCG) method using Object-based CNN (OCNN) for land cover classification [291]. During the first step, objects with a fine boundary and high internal compactness are first segmented as functional units using a multi-resolution segmentation (MRS) algorithm. During the second step, the per-object classification is run by means of MLCG based on OCNN. The contour-preserving objects are clipped from images using a minimum bounding rectangle (MBR) mask policy. Object oriented context patches are then produced by means of masking the windows with flexible sizes on each object. At the same time, the object deformation coefficient is derived during the object resize operation. It is then used as a supplement for

geometric characteristics for object discrimination. Finally, the deep independent object features and contextual information (i.e., the features extracted from context patches) are fused in the proposed OCNN. Instead of extracting features from patches to represent objects, MLCG-OCNN utilises objects and context patches as inputs.

During the third step, a CRF graph model is employed to explore the contextual information of neighbouring pixels to further improve the classification results. This work utilised OCNN CRF to provide object-level contextual guidance and CRF to achieve pixel-level contextual guidance.

The proposed method of OCNN and CRF [292] claims to provide a superior performance by extracting in reference to a contextual patch, which differs from normal patches generated by CNN-based approaches.

In this research, achieving a proper patch size is a crucial task. Therefore, this research utilised segmentation scale parameters (SSPs) in the MRS. This study tested various SSPs parameters on Vaihingen and Potsdam datasets, and came to an experimental value of 20 to 50 for Vaihingen images and 40 to 100 for the Potsdam images, respectively.

Available edge information has been utilised for building detection, but classification may also be needed [292]. This paper proposes a standalone end-to-end edge-aware neural network (EaNet) for urban scene semantic segmentation. For semantic consistency preservation inside objects, the EaNet model incorporates a large kernel pyramid pooling (LKPP) module to capture the rich multi-scale context with strong continuous feature relations. To effectively separate confusing objects with sharp contours, a Dice-based edge-aware loss (EA loss) function was devised to guide the EaNet to refine both the pixel- and image-level edge information directly from semantic segmentation prediction. This architecture uses a combination of Conv pool network along with an encoder-decoder framework based on LKPP network. The results have been reported using three datasets,

which are: Cityscapes, ISPRS Vaihingen, and the WHU Aerial Building datasets. This research proposed combining multi-scale object segmentation and boundary refinement and achieved 80.9% IOU and 96.4% F1-score for building class with the Vaihingen dataset.

The current research work has focused on exploiting boundary refinement in a multi-scale object segmentation framework. However, it has been identified that although the Mask R-CNN and CRF-based frameworks are computationally demanding, they can provide better results if precise bounding co-ordinates are extracted along with the boundaries [292].

The use of CNN (a popular image dataset for image classification) to win the ImageNet competition reduced the classification error from 26 to 15 [327]. In addition, a range of ways have been developed to utilise CNN. For example, one application [328] investigated the application of CNN for individual building extraction from side-facing images extracted from Google StreetView, hence, open sources from Google AIP style transfer pixel to pixel translate, without any reference style, were used. The results from Google Maps at a resolution of 512 x 512, with the model trained using images at 256 $\times$ 256 resolution, were run with larger images convolutionally, and the contrast was adjusted for clarity.

Authors have proposed a unified and effective deep CNN-based approach for the simultaneous segmentation of multi-class objects in images with large-scale variability in remote sensing [296], . They used a redesigned CNN feature extractor to perform the detection. Pixel-based classification, or mapping using neural networks (which is the architecture U-Net), is clearly based on this kind of approach. The architecture of the U-Net, as the name suggests, is a U-shape, where convolution is done on the input image, which is then deconvolved to map an output image. All of the architectures and techniques that have been discussed so far can be used to detect or find rooftop information from images. Using an image-based approach such as segmented-CNN has become possible, which uses whole or par-

tial images as the input data. However, in this case, a huge number of images are required for network training. Therefore, a compromise can be made by using certain rules and a specific principle for the preliminary spatial clustering of pixels. This technique may be described as being based on 'superpixels'. With the transition to 3D, the situation becomes even more radical, given the increasing number of pixels (which are called voxels in 3D).

Finally, given that the segmentation map alone is not sufficient to preserve the sharp edges present in human-made structures, this thesis intends to add the power of CRFs [302] to the principal architecture of Mask R-CNN to preserve the building's shape and geometry while generating the mask.

Consequently, predicting corner keypoints and refining the prediction using CRF will allow the development of new algorithms that are able to predict the building's geometry with high accuracy. FCN is used to generate a heatmap for each of the keypoints and it extracts the points from the heatmap. The following section presents the concept of CRF in more detail.

2.5.7 Conditional Random Fields - CRFs

A CRF is an undirected graphical model of a random variable $\boldsymbol{X}$ that describes a sequential observation. In this graphical model, any random variable $\boldsymbol{X}$ belongs to a set of labels $L = \{L_1, L_2, \dots L_k\}$. Any pixel u at position i will have to pay an unary cost $\phi_u(X_u = x_u|I)$ to assign a label of x_i for pixel u in the image $\boldsymbol{I}$. The image $\boldsymbol{I}$ has a unique color assigned to each label. Panboonyuen et al. [302] report that CRF considers the low-level information captured by the local interactions of pixels and edges. In addition, their results show that this method outperforms all of the baselines in terms of precision and recall.

This research examines how to implement a pairwise cost, which is the relationship between the pixels and the object edge, and can be represented as $\phi_{u,v}(X_u =$

$x_u, X_v = x_v|I)$. This cost function applies the Potts' Model, which assigns zero value when the labels are different and a certain value λ when the two labels are the same. The energy and pairwise costs can now be combined to calculate the total energy cost of the model, as follows:

$$E(x, I) = \sum_{u \in V} \phi_u(X_u = x_u|I) + \sum_{u,v \in E} \phi_{u,v}(X_u = x_u, X_v = x_v|I) \tag{2.7}$$

where E is the energy cost for the given configuration of x

CRF will then have to minimise this energy cost for the given configuration of x. After the use of CRFs on the segmentation result, the final output was satisfactory. However, the result was missing the structural rigidity that is present within the buildings. Consequently, this thesis exploits the geometry present in the segmentation to provide a better structure for the detected buildings.

2.6 Summary

The feature design of traditional remote sensing classification methods is typically hand engineered. There are also have classifiers with a shallow structure and architecture, usually with two separate yet complementary steps for carrying out feature extraction and classification [329]. Feature extraction is conducted using particular operators on local portions of the image, such as objects, regions, image patches or super-pixels. The original spectral feature space is transformed into compact and abstract representations, which can easily be separated using a classifier [330]. These transformed spatial features can then be utilised, along with the original spectra, to train certain types of supervised classifiers for tasks such as supporting vector machine to understand the semantic content of the imagery that is input [331]. A classifier's performance is greatly influenced by the transformations that are used and the resultant spatial features. Some typical

examples of these kinds of operators are mathematical morphology [332], texture descriptors [333], and oriented gradients [334]. Hand engineering features usually involves a monotonous process of trial and error to extract and select features [329]. Furthermore, hand-coded features are typically task specific and are only useful to address a certain region or specific problem. In addition, using low-level features followed by shallow classifier architectures is not effective enough to mine the underlying semantics or functions because of the lack of high level feature representations [335]. Therefore, to date, classification performance has been limited with regard to the use of spectrally and structurally complex VHR images.

Remote sensing classification methods, such as pixel-based machine learning methods (e.g., multi-layer perceptron (MLP) ,support vector machines (SVM), and pixel-based random forest decision tree classifier (RF) followed by object-based SVM acting as the object-based image analysis (OBIA) paradigm) are on the same level as the existing deep learning methods (e.g., patch-based CNN; CNN with segmented object averaging, CNN withCRF, and FCN with CRF). The use of these classification methods requires satellite and aerial imagery as the input data source, which are classified into several land use categories (e.g., buildings, roads and trees). The accuracy range of pixel-based using MLPis 75% to 81% [336, 337], which is comparable to pixel-based SVM at 75% to 83%, and pixel-based RF at 74% to 83%. In comparison to pixel-based approaches, object-based SVM has slightly higher accuracy, at 80% to 85%, because it takes the spatial context into account. However, hand-coded features or rules are used for these traditional methods, which is problematic because they are hard to design and are based on the knowledge and expertise of the individual user.
Meanwhile, deep learning methods can automatically learn feature representations and they have shown promising performance outcomes in recent years. For example, CNN offers a wider range of accuracy than traditional approaches, ranging from by overall accuracy, at 85% to 88% [338]. However, there is a risk of

losing spatial resolution and the edges of ground features may be blurred. Nevertheless, the latest developments in deep learning methods have shown progress with regard to these problems, including through post-processing (e.g., an accuracy range of 82% to 86% of the CNN has been achieved by integrating the segmented object's averaging during the post-classification process). Patch-based CNNs, along with CRF, are likely to further improve accuracy by using class specific probability distribution along the boundaries, with results achieving 85% to 87% accuracy [302]. Attention is also being paid tospecial deep architecture (SegNet) in the remote sensing community to conduct pixel-wise semantic labelling, which has shown at least 30% performance improvement accuracy range [339]. These types of networks can be generalised as being fully convolutional networks FCN that have different extensions, such as dilated convolution or CRF post-processing, to conduct semantic segmentation. For FCN with CRF, the accuracy range is 87% to 93%, and achieves $> 90\%$ overall accuracy on the ISPRS Vaihingen (on average) for buildings [7].

In summary, the average accuracy of traditional pixel-based machine learning methods and object-based methods is around 80%. Meanwhile, the majority of the deep learning methods that have been reviewed here have reached an accuracy of $> 85\%$ classification, on average. All of the currently available methods have their own types of problems and challenges. Therefore, the goal of this research study has been to develop novel deep-learning methods that increase the accuracy of boundary classification to reveal the most accurate building detection, and extraction and classification approaches.

Chapter 3

Methodology

3.1 Introduction

The methods that used in this research are based on the review of the literature and an exploration of previous research studies. The literature review highlighted the need to utilise the vectorised information available from the building images to achieve better performance measures for automatic building detection. This thesis proposes to use of AI-based techniques to improve on the automatic detection of building boundaries.

This research has reused and developed two models to perform image detection: the first is GAN*pix2pix* [299] which is one of the latest deep learning methodologies, and its variant attention-GANs; the second is *detectron2* [340], which is used as the Mask R-CNN to detect key-points, as well as for instance segmentation. For the Mask R-CNN, two different backbone architectures (ResNet-50 and ResNet-101) were used for the experiment. Both of the networks were pre-trained on the ImageNet dataset. This thesis will also exploit the latest refinement output techniques for deep learning. Furthermore, two novel post-processing techniques have been developed, which extract and use vital information available from VHR imagery in the form of vectors that help to identify the inherent features of buildings.

The mask R-CNN that has been used is the latest and fastest deep learning approaches. It combines the power of CNN, Region Proposal Network RPN and a FCN to provide an accurate mask of buildings from satellite imagery. This thesis

will also discuss some important key-points, which this research work proposes to use to derive vectorised information for use in post-processing to modify and improve on the detected contours. CRFs have also been used to smooth out any random irregularities that may have been introduced by Mask R-CNN in the detected contours.

Both the CNN- and GAN-based methods use the same concept involving the use of ground truth information during the training of the deep artificial neural network. While GANs provide faster results, the repetitiveness of the results are not guaranteed when the images are resized and re-sampled. Accordingly, in this scenario, Mask R-CNN provides better results. This thesis further evaluates the proposed methods by examining two network backbones, namely: ResNet-50 and ResNet-101.

3.2 Building Boundary Detection Methods

To recap, the following hypotheses have been defined in Section 1.2:

1 The first hypothesis of this research work states: "Satellite imagery data is sufficient to provide accurate LoD information for roof top detection."

2 The second hypothesis of this research work states: "An architecture method that is based on deep learning techniques can be used to classify and extract localised contours around building boundaries using VHR remotely-sensed images."

3 The third hypothesis of this research work states: "Post-processing techniques can significantly improve the output for detection accuracy."

4 The fourth hypothesis of this research work states: "Detection of the vectors performed by deep learning architecture-based can provide vital informa-

tion to help post-processing determine useful geometrical information for building detection."

Based on the literature review, and up to date state-of-the-art approaches involving DNN, the experiments were conducted by taking two different machine learning approaches: 1) GAN and 2) Mask R-CNN based network. The sections that follow summarise the machine-learning techniques used in conjunction with various contour refinement post-processing techniques to test the hypotheses.

3.3 Generative Adversarial Network GAN

GAN was initially applied to explore approaches to post-processing, as described in the literature review. A large number of applications have since applied a GAN architecture. This thesis will use several variant GANs, depending on the loss function and the way that they are formed. The following sections describe *pix2pix* GANs and SAGAN, which have been used to perform the segmentation. This led to a clearer understanding of the detention processes, which forms the basis for the next steps.

3.3.1 *pix2pix* GAN Model

This section demonstrates how to build and train a *pix2pix* (in some of the Literature it is called conditional generative adversarial network (GAN)), which learns mapping from input images to output images, as described in the literature review. The model is *pix2pix* by Isola et al. [341]. The architecture of the network contains a generator with a U-Net-based architecture and a discriminator represented by a convolutional Patch GAN classifier (proposed in the *pix2pix* paper). The network assigns a label (or class) to each input image in an image classification task. However, in some cases, it is necessary to know the shape of that

object, which pixel belongs to which object, and so on. In this case, assigning a weight to each image pixel will be needed. The generator process connects the layers by value weight between zero and one for each pixel through the generator process. The weight contains the pixel value and something called bias value, then the sigmoid function was applied to the final weight to increase the values, which were close to one, and lowered the values, which were close to zero, as can been seen in Figure 3.2.

This can formalised as Equations (3.1 and 3.2)

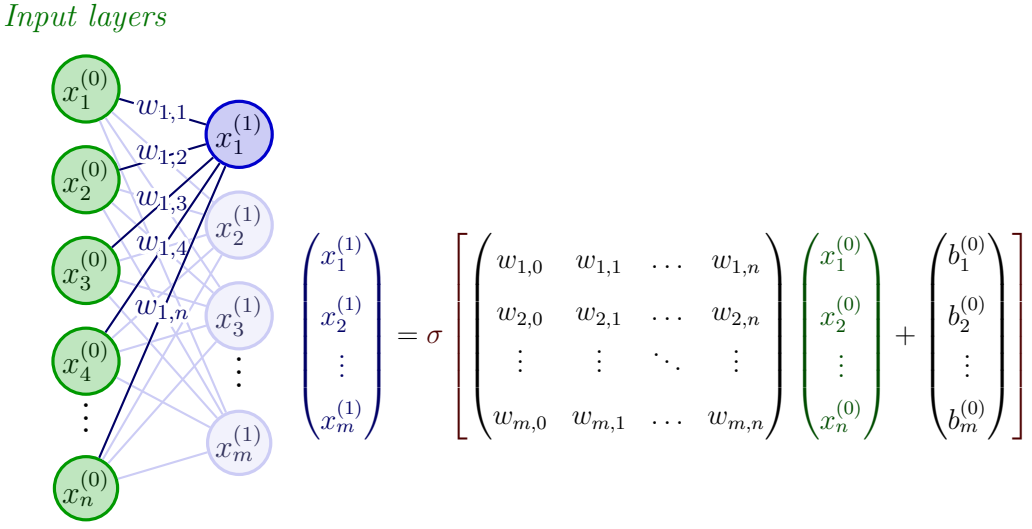


Figure 3.1: a partial view of GANs shows how a layer is weighted

$$x_1^{(1)} = \sigma\left(w_0 x_1^{(0)} + w_1 x_1^{(0)} + \ldots + w_n x_n^{(0)} + b_1^{(0)}\right) \tag{3.1}$$

$$x_m^{(n)} = \sigma\left(\sum_{i=1}^{n} w_i x_i^{(0)} + b_m^{(0)} + \ldots + w_m^{(n)} x_m^{(n)} + b_m^{(n)}\right) \tag{3.2}$$

where: x_0 is the magnitude and direction of gradient; x_m is Gaussian filter and $W^{(0)} \in \mathbb{R}^{m \times n}$.

The optimism generated output when the result of the Equations (3.1 and 3.2) is equal or close to one, which happens when the networks have a well-connected weight and reflect the predictions closed to authentic images. To evaluate the generator results, it is necessary to calculate the error (in AI called log Loss). To understand this idea behind, suppose there is actual label1 with a value of one and assume the generator prediction $G(x)$ 0.1; that means it is a lousy prediction generated by the networks because the error was huge. on the other hand, if the prediction $G(x) = 0.9$, that means the network is doing well. To formulate this case with the notice of negative natural logarithm of 0.1 equal to 2.3 and 0.9 equals 0.1, representing the correct reflection of the predictions. Therefore GAN used log-loss function to evaluate the prediction results as follows:

$$G(x)_e = -\ln(x) \tag{3.3}$$

$$D(x)_e = -\ln(1-(x)) \tag{3.4}$$

where: $G(x)_e$ loss function (error) from noise (prediction), and $D(x)_e$ loss function (error) from images. Figure 3.3 demonstrated this clearly, as seen in the graph, negative natural logarithm of $G(x)$ is significant increase the error when the prediction value of x is close to zero and small when the prediction value of x is close to one. The opposite with function $D(x)$ in case of the error of the image.

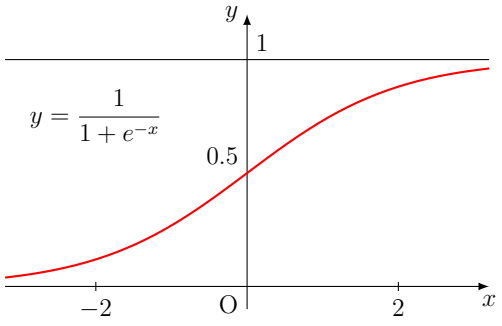


Figure 3.2: Sigmoid function

- **Backpropagation**

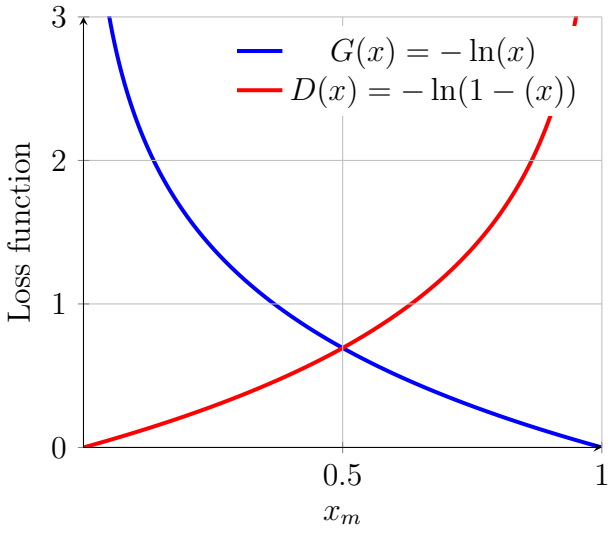


Figure 3.3: Loss function diagram and sigmoid function

After training both discriminator and generator and evaluating this, the next step is called backpropagation, which is mainly used to reevaluate and recalculate the networks weighted to decrease the generator error by and increase the weighted (x) as shown in Figure 3.4, as well as to decrease the discriminator error by decreasing the weighted (x). To do this, GAN uses a gradient descent process, which takes the first derivative of the log-loss function to calculate the gradient direction of the most significant growth. Then, it takes a tiny step into the negative of the gradient to find new parameters that decrease this error as much as possible. after that, it goes forward pass calculating the prediction and then calculating the error based on the previously defined log-loss. As a sequence, the first derivative of the log-loss function for the whole network with new weight has been calculated using the **Chain rule**. as a result, the perdition error will improve, and the following step shows the first derivative of the log-loss function:

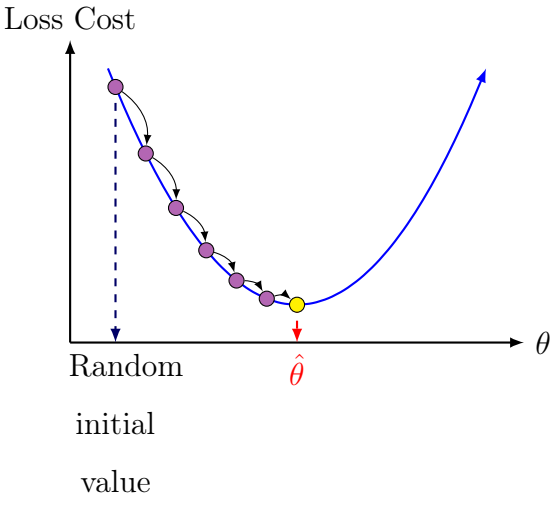


Figure 3.4: Gradient descent process.

- **Derivatives of Loss function (error) from images**

  Back propagation was used to track the weights, using the method gradient descent, therefore it was necessary to apply the derivatives using the following equations: Equations 3.1, 3.2, 3.3 and Equation 3.4 the following steps to find the first derivatives of these equations using Chain rule :

  ∵ Equation 3.2, therefore :

$$
\begin{aligned}
D(x) &= \sigma\left(x_1 w_1 + b_1 + x_2 w_2 + b_2 + \ldots\ldots + x_n w_n + b_n\right) \\
\frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial D} \cdot \frac{\partial D}{\partial w_i} \\
&= \frac{-1}{D(x)} \cdot \sigma\left(\sum_{j=1}^{m} x_j w_j + b\right)\left[1 - \sigma\left(\sum_{j=1}^{m} x_j w_j + b\right)\right] x_i \\
&= \frac{-1}{D(x)} \cdot D(x)[1 - D(x)] x_i \\
&= -[1 - D(x)] x_i \\
\frac{\partial E}{\partial b} &= \frac{\partial E}{\partial D} \cdot \frac{\partial D}{\partial b} = -[1 - D(x)]
\end{aligned}
\tag{3.5}
$$

  ∴ the first derivative of Discriminator becomes:

$$
E = -\ln(1 - D(x)) \tag{3.6}
$$

$\because$ Equation 3.3, therefore:

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial D} \cdot \frac{\partial D}{\partial w_i} \\
&= \frac{1}{1 - D(x)} \cdot \sigma\left(\sum_{j=1}^{m} x_j w_j + b\right)\left[1 - \sigma\left(\sum_{j=1}^{m} x_j w_j + b\right)\right] x_i \\
&= \frac{1}{1 - D(x)} \cdot D(x)[1 - D(x)] x_i \\
&= D(x) x_i \\
\frac{\partial E}{\partial b} &= \frac{\partial E}{\partial D} \cdot \frac{\partial D}{\partial b} = D(x)
\end{aligned} \tag{3.7}$$

$\therefore$ the first derivative of Generator error becomes:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{3.8}$$

$$\begin{aligned}
&\min_G \max_D \left(E_{x \sim p_{data}}[\log D(x)] + E_{z \sim p(z)}[\log(1 - D(G(z)))]\right) \\
=&\min_G \int_X \left(p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x) + p_G(x)}\right) dx \\
=&\min_G \left(E_{x \sim p_{data}}\left[\log \frac{2}{2} \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + E_{x \sim p_G}\left[\log \frac{2}{2} \frac{p_G(x)}{p_{data}(x) + p_G(x)}\right]\right) \\
=&\min_G \left(E_{x \sim p_{data}}\left[\log \frac{2 * p_{data}(x)}{p_{data}(x) + p_G(x)}\right] + E_{x \sim p_G}\left[\log \frac{2 * p_G(x)}{p_{data}(x) + p_G(x)}\right] - \log 4\right)
\end{aligned} \tag{3.9}$$

Optimal Discriminator in first term Equation 3.9:

$$D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} \tag{3.10}$$

$\because$ Kullback-LeiblerD_{KL}[1] defined the difference between two references as a divergence ratio of the probability distribution by:

[1] D_{KL} is a mathematical statistics theorem, which is also called relative entropy.

Theorem 1

$$KL(p,q) = E_{x\sim p}\left[\log\frac{p(x)}{q(x)}\right] \tag{3.11}$$

$\therefore$ Then Equation 3.9 becomes:

$$= \min_G \left(KL\left(p_{data}, \frac{p_{data}+p_G}{2}\right) + KL\left(p_G, \frac{p_{data}+p_G}{2}\right) - \log 4 \right) \tag{3.12}$$

$\because$ in Jensen-Shannon's theory [2] $JSD_{(\mathrm{p,\ q})}$ is:

Theorem 2

$$JSD(p,q) = \frac{1}{2}KL\left(p, \frac{p+q}{2}\right) + \frac{1}{2}KL\left(q, \frac{p+q}{2}\right) \tag{3.13}$$

$\therefore$ Equation 3.12 Optimal Discriminator becomes:

$$= \min_G \left(2 * JSD\left(p_{\mathrm{data}}, p_G\right) - \log 4\right) \tag{3.14}$$

Summary: the global minimum of the minimax game happens when:

$$\begin{aligned} &\text{when } D_G^* = \frac{p_{\mathrm{data}(x)}}{p_{\mathrm{data}(x)+p_G(x)}} && \text{(Optimal discriminator for any G)} \\ &\text{when } p_G(x) = p_{\mathrm{data}(x)} && \text{(Optimal generator for optimal D)} \end{aligned} \tag{3.15}$$

Unlike normal GANs where a generator maps a random noise vector $\boldsymbol{z}$ into an image $\boldsymbol{y}$, $G : z \to y$, conditional GANs map an observed image $\boldsymbol{x}$ and a random noise vector$\boldsymbol{z}$ into an image. The objective function for the conditional GANs can be given as:

$$L_{cGAN}(D,G) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))] \tag{3.16}$$

The *pix2pix* GAN introduces L1 distance into the loss function because it can produce outputs that are close to the images sampled. Consequently, *pix2pix* GAN is responsible for optimising the following objective function

$$G* = \arg\min_G \max_D L_{cGAN}(D,G) + \lambda \mathbb{E}_{x,y,z}[||y - G(x,z)||_1] \tag{3.17}$$

[2] This is another method to measure the similarity between two probability distributions.

3.3.2 Attention block with GAN (*SAGAN*)

Attention-based GANs are relatively new concepts and were first introduced by *SAGAN*, which implemented non-local convolution networks. These networks are based on the idea of self-attention and they allow the network to pass non-local parameters that are present in the data.

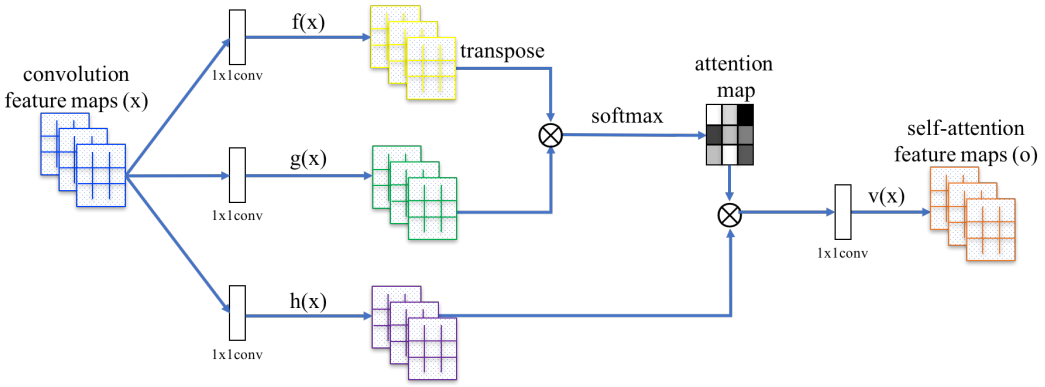


Figure 3.5: The self-attention mechanism that is used in *SAGAN*.

The attention block is shown in the diagram above. This idea has been borrowed from several papers. The primary focus is on the idea of transformers that was originally proposed by Vaswani et al. [300]. SAGAN allows the network to estimate the weights on one part by estimating from the position across the whole area. In simple terms, attention allows the network to estimate the parts that it should focus on. As a result, the network can learn non-local relationships. Given that there are some relationships with the structure of the input samples in the case city, the assumption is that the network will be able to figure out how buildings are structured and where they are located.

In the original paper, for the self-attention mechanism (see Figure 3.6), the transformer represents the input in the form of a key **K** and value **V** pair. It then uses a third representation of query to calculate the attention. The mapping from

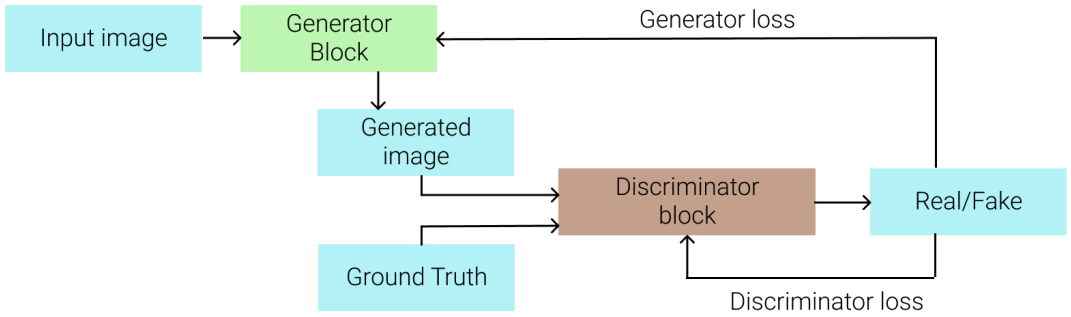


Figure 3.6: Overall block diagram representing SAGAN

query (Q) to the key-value $(K-V)$ pair produces the attention score:

$$\text{Attention}(\mathbf{Q},\, \mathbf{K},\, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}})V \tag{3.18}$$

This attention function provides the foundation for *GAN*. In *SAGAN*, the query and key are represented by $f(x)$ and $g(x)$, which compute the affinity between pixels at locations i and j, while the value is represented by $h(x)$, which computes the representation of pixels at position i. This attention is also similar to the previous equation 3.18. The inputs to $\boldsymbol{f}$ and $\boldsymbol{g}$ represent the features $\boldsymbol{x} \in \mathbb{R}^{C\times N}$. We can visualise this similarity from the Equations that follow:

$$\beta_{j,i} = \frac{\exp(s_{ij})}{\sum_{i=1}^{N}\exp(s_{ij})},\ \text{where}\ s_{ij} = \boldsymbol{f}(\boldsymbol{x_i})^T\boldsymbol{g}(\boldsymbol{x_j}), \tag{3.19}$$

where $\beta_{j,i}$ indicates the extent to which the model attends to the i^{th} location when synthesising the j^{th} region. Here, C is the number of channels and N is the number of feature locations of features from the previous hidden layer.

$$\boldsymbol{o_j} = \boldsymbol{v}\left(\sum_{i=1}^{N}\beta_{j,i}\boldsymbol{h}(\boldsymbol{x_i})\right),\ \boldsymbol{h}(\boldsymbol{x_i}) = \boldsymbol{W_h}\boldsymbol{x_i},\ \boldsymbol{v}(\boldsymbol{x_i}) = \boldsymbol{W_v}\boldsymbol{x_i}. \tag{3.20}$$

In this formulation, $\boldsymbol{W_g} \in \mathbb{R}^{\bar{C}\times C}$, $\boldsymbol{W_f} \in \mathbb{R}^{\bar{C}\times C}$, $\boldsymbol{W_h} \in \mathbb{R}^{\bar{C}\times C}$, and $\boldsymbol{W_v} \in \mathbb{R}^{C\times\bar{C}}$ are the learned weight matrices, which are implemented as 1×1 convolutions.

Building on this idea, this thesis introduces a new architecture that adds an attention unit to each of the upstream layers in the U-Net Figure 5.1. This new

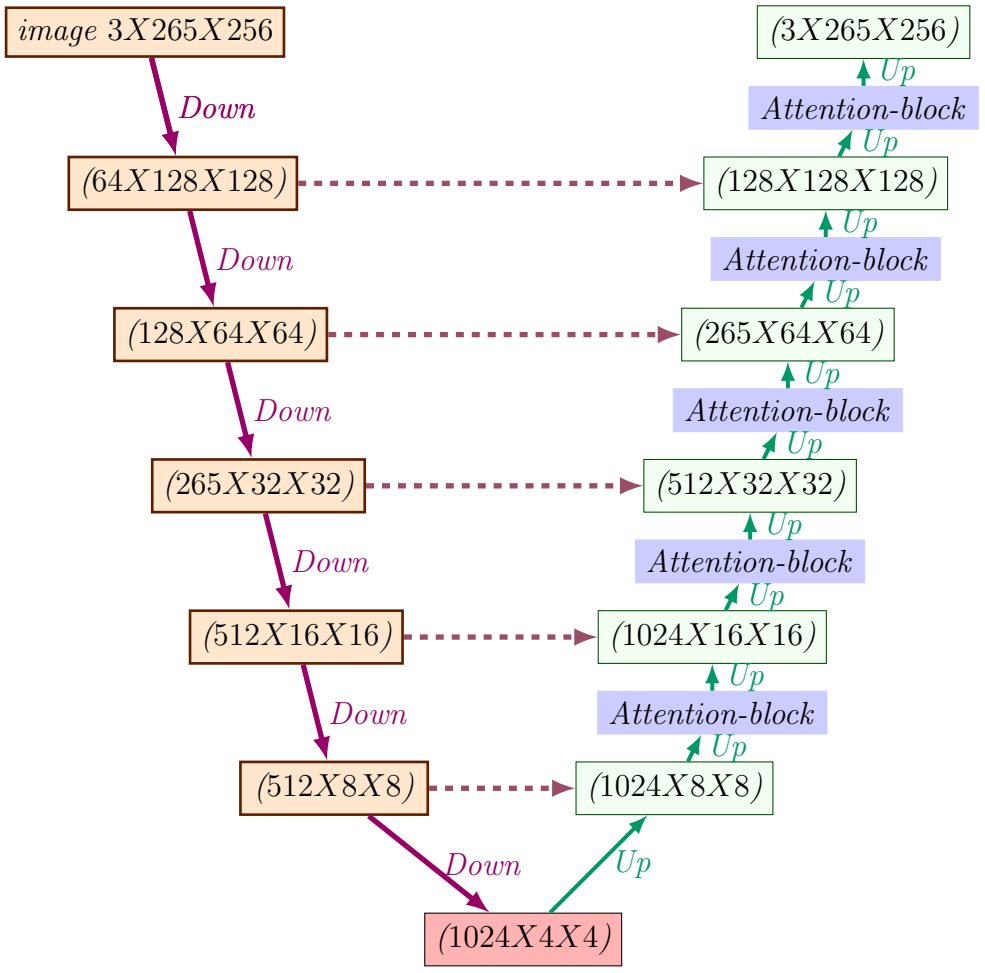


Figure 3.7: SAGAN uses the mechanism of attention-block during up and down-samples of generator the training setup.

architecture allows the network to attend to the required part. The detail is shown in the following .

Finally, the loss function LS-GAN is used to train the model, which is given as:

$$\begin{aligned}\min_D V_{\text{LSGAN}}(D) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x})-b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-a)^2\right] \\ \min_G V_{\text{LSGAN}}(G) =& \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}\left[(D(G(\boldsymbol{z}))-c)^2\right],\end{aligned} \tag{3.21}$$

where a denotes the labels for fake data, b denotes labels for the real data, c denotes the value that G wants D to believe for fake data.

The results obtained from the GANs are stochastic in nature. This means that

they are not suitable for the task of remote sensing, where the building location barely changes. Consequently, this thesis goes on to utilise the state-of-the-art methods on image segmentation, using Mask R-CNN. The issues mentioned above will be demonstrated in more detail in the experimental setup section.

3.4 Mask R-CNN

Mask R-CNN is a region-based CNN that builds on the faster R-CNN, which is is an object detection architecture that detects object present within the image in two stages. In the first stage, the model uses the RPN to detect the possible locations of the objects and to predict a bounding box for the objects. The second stage extracts features from the box using a method called as RoIPool, which performs the required classification and regression of the bounding box. Mask R-CNN builds on this by adding a third head, which not only performs classification and regression but also creates the segmentation mask for a given RoI (as can be seen in the architecture section).

3.4.1 Architectural Details

As mentioned earlier, Mask R-CNN builds on the idea of Faster R-CNN. In the case of mask R-CNN, the first stage is identical to that of Fast R-CNN. Here, the network uses RPN to propose possible regions where objects are likely to be present. In the second stage, along with bounding box and object classification, the network uses FCNs to generate the segmentation mask for each instance of the objects. For these segmentations, the network employs a special operation called RoI-align that is responsible for generating pixel to pixel segmentation. Each of these masks are of size $m \times m$, which allows the network to preserve the spatial information.

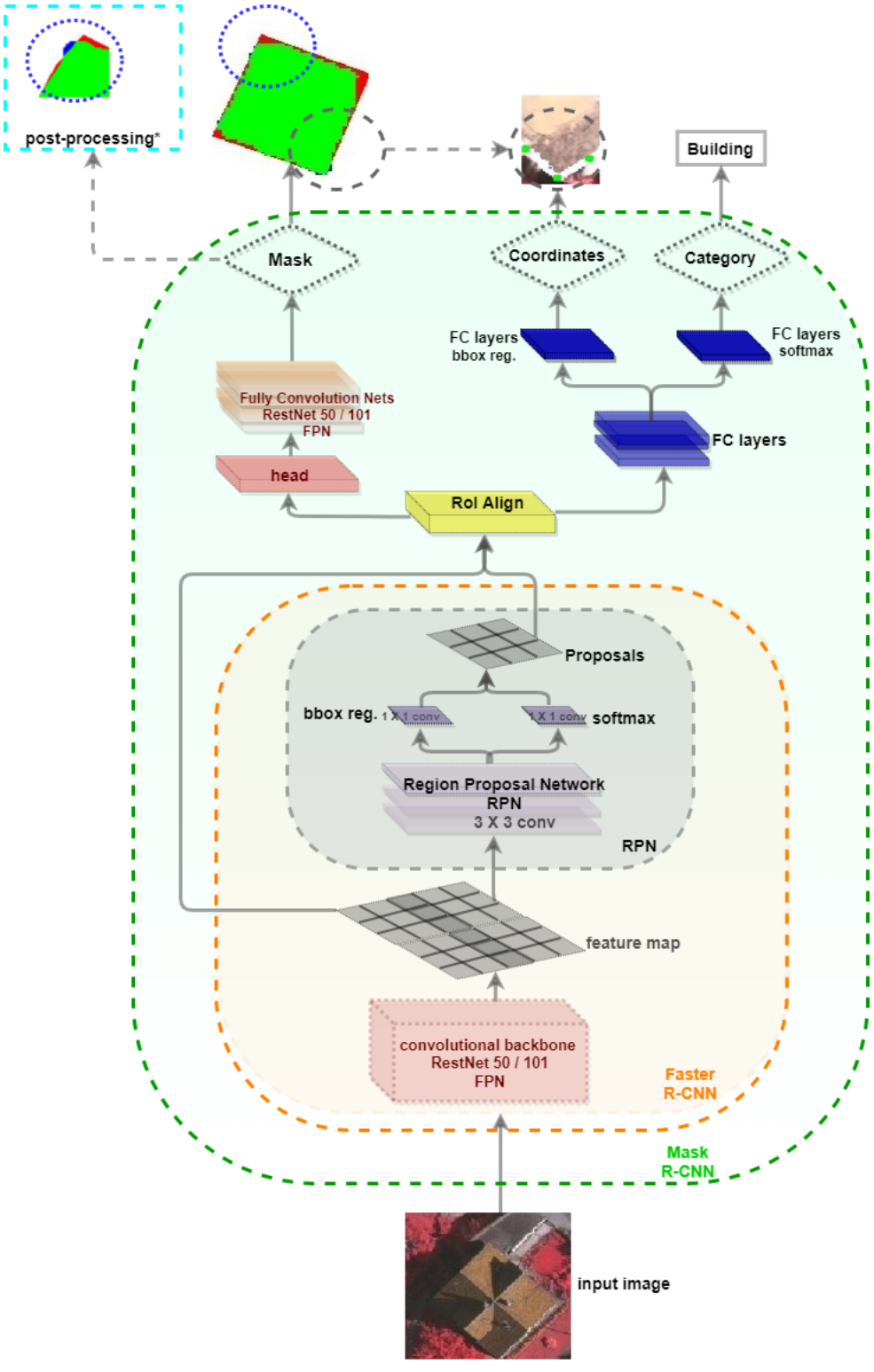


Figure 3.8: Architectural details of Mask R-CNN, showing how the network uses FCN to generate a heatmap at the top of the Architecture head.

In this research, one extra step is added along with the segmentation map to detect corners as keypoints. The idea here is to preserve the shape of the building along with its geometry. However, the segmentation map alone is not sufficient to preserve the sharp edges that are present in man-made structures, such as buildings. Consequently, predicting corner keypoints will help developed a new algorithm that can estimate the geometry of the building. To do so, the network uses FCN to generate a heatmap for each of the keypoints. It then extracts the points from the heatmap, as can be seen in Figure 3.8 and it will be described in more detail in the experimental setup of the model.

3.5 Post-processing Techniques

DontPrintSemicolon

A man-made building generally follows a rigid geometrical structure. Consequently, segmentation masks for buildings are mostly rectangular or a regular shape, and are suitable even if the angles differ. Even in other normal cases, the buildings will have a definite and standard building structure that is generally a well-known practice. This narrows the entire shape of a building into a smaller space, where the task will be to estimate a path that allows us to maintain this shape. To navigate the shape of the building, it is desirable to know the contours of buildings and their major corner points. Current state-of-the-art segmentation networks are able to identify the locations of buildings and their segmentation mask in a given image. With this information, it is easier to get the contour of a building. However, it may not maintain the rigid geometrical structure in terms of the shape and area of the final output.

Allowing the neural network to predict major keypoints is a considerable design factor because it can provide directional constraints for algorithms to search for a suitable building shape. This rationale is the motivation to propose a snapping al-

gorithm, which is capable of snapping a contour into the appropriate shape of the building. The algorithm uses its contour points and predicted keypoints to navigate through the 2D space, while creating an approximate shape of the building. This section describes the working of CRF and the snapping algorithm. It also describes the necessity to modify the snapping algorithm to include keypoints, such as the corner points of a building.

3.5.1 Using Conditional Random Fields

In this graphical model, a vertex V can be viewed as an image pixel and an edge E can be viewed as a node pair for which the pairwise cost is defined. This means that we can now combine the unary and pairwise cost to calculate the total energy cost of the model, as follows:

$$E(x, I) = \sum_{u \in V} \phi_u(X_u = x_u | I) + \sum_{u,v \in E} \phi_{u,v}(X_u = x_u, X_v = x_v | I) \tag{3.22}$$

where $\boldsymbol{X}$ = random variable that describes sequential observations

θ_C = angular threshold for corner point

Then, CRF will have to minimise this energy cost for the given configuration of x. After the use of CRF on the segmentation result, the final output was satisfactory. However, the result was missing the structural properties present within the buildings, such as shape and area. Therefore, CRF alone is not sufficient to meet this study's objectives. Therefore, another method to exploit the geometry present in the segmentation is proposed to better structure the detected buildings.

3.5.2 Proposed Snapping Algorithms

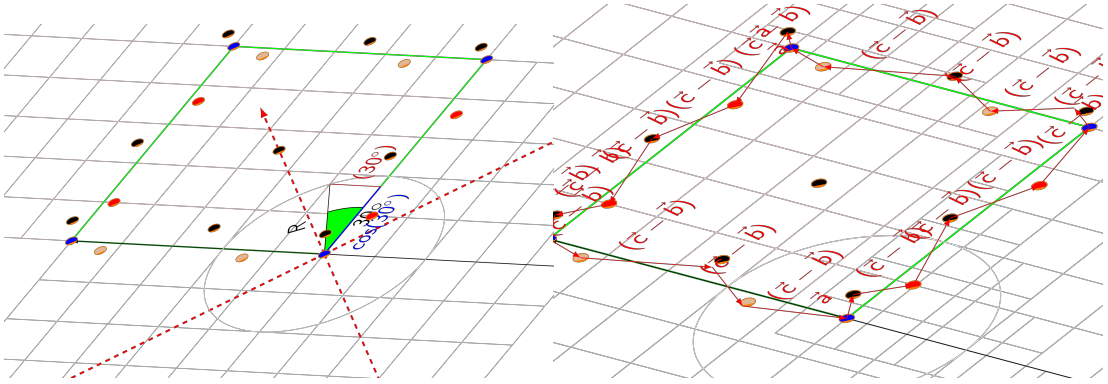
When dealing with structures, their geometry is a vital piece of information because it allows engineers to properly estimate the size and shape of the building. Therefore, it is necessary to improve the results after using CRF and propose a new post-processing algorithm that is able to properly estimate the shape of the buildings. This thesis explains how snapping occurs on a given set of contours, how keypoints in a building can be exploited to infer a proper shape of the building, and how the orientation of the building can be estimated from the azimuth. The snapping algorithm works on a single instance of the building's contour. Therefore, the snapping is specific to the correctly classified building.

Snapping Algorithm, Method 1

The snapping contour algorithm that is used in this thesis exploits the line segments in the building's region to improve the shape of the predicted contour. First, the method selects a reference line that is based on all of the lines that are within the proximity of the contour. After detecting these lines, this method then employs a voting algorithm that selects a reference with maximum amount of support in a particular direction or angle (α_R). These reference lines provide the basis to snap the contour into the proper shape of the building. The motivation to use the reference line is to orient the the shape of contour to the axis of the building. This reference line acts as the bounding condition to the building, which limits the search space for tuning the shape of the building.

This work defines a procedure called `FixContour()` that fixes the shape of the building based on the reference line and a contour. This procedure first sorts the points of the contour such that the initial point will have the lowest value in terms of $y_$coordinate. In the case of a tie, the algorithm breaks the ties by taking the point with lowest $x_$coordinate.

The algorithm then loops through all of the contour points, taking three consecutive points in circular fashion. Considering a, b, and c as these three consecutive points, the algorithm finds the *direction* from a to b, and from b to c. It the proceeds in an anticlockwise direction and checks whether a given point b is a corner point or the a point. As it moves along the anticlockwise direction, if the direction of a, b, or c forms a clockwise angle, then it marks point b as a corner point, and otherwise b is an edge point. It then calls a procedure that snaps the point b and c to a new point based on this information. Once these points are updated using the `snap()` procedure, at the end of every loop it appends the point b into the list *new_ contour*. The algorithm finally returns the list as shown in Figure 3.9.



(a) (b)

Figure 3.9: The algorithm's mechanism plot:(a) Row points surrounding target mask (green square); colours show different point directions.(b) Algorithm procedure using cross-product method to eliminate and accept point with high voting.

Method 1 Algorithm 1. 1: `FixContour()`

Procedure `FixContour(`*contour, line*`):`

Data: *contour*, a list of points representing contour *line*

Result: *new_ contour*, a list of updated contour points

begin

1 Sort the contour so that the first point is the point with lowest value of ycoordinate

2 Create a list new_contour to hold new points

3 **for** $i = 0; i \leq$ (no. of points); i++ **do**

4 take a, b, c as i, $(i+1)$ and $(i+2)$ point in the contour

5 calculate $direction = (c-b) \times (b-a)$

6 **if** $direction > 0$ **then**

7 $b \leftarrow$ *snap(b, a, line)*

8 update $(i+1)$ point $\leftarrow b$

else

9 $c \leftarrow$ *snap(c, b, line)*

10 update $(i+2)$ point $\leftarrow c$

end

11 append b to the *new_ contour*

end

end

The important aspect is how the algorithm in each loop checks for the direction of consecutive lines to snap the line inward or outward based on the direction of consecutive lines. The idea here is to allow the algorithm to form sharp edges and corners because they are responsible for reflecting the structure of the building. The algorithm `FixContour()` fixes the output as first stage to fixed,and refine edges and corner,then in second stage a special algorithm called `snap()` using the as reference line . This method is responsible for altering the points of consecutive

lines to align them in such a way that it either forms a corner or an edge.

The procedure `snap()` snaps point q into the line through point p. To do so, the `snap()` first finds the lines passing through the point p, which is parallel(l_1) and perpendicular(l_2) to the given line(l). It then proceeds to find the line(l_0) passing through pq. Once these lines are found, it computes the slope of each line and aligns the line (l_0) through pq into the line that matches its slope. If (l_0) aligns with (l_1), then the algorithm selects (l_1) as a reference, and it selects (l_2) otherwise. It finds the perpendicular line to a reference line passing through point q, and finds the intersection point (ip) of the perpendicular and reference line. It then checks the distance of ip and q, and moves q to the point that splits the line through ip-q in certain ratio as shown in Figure 3.10.

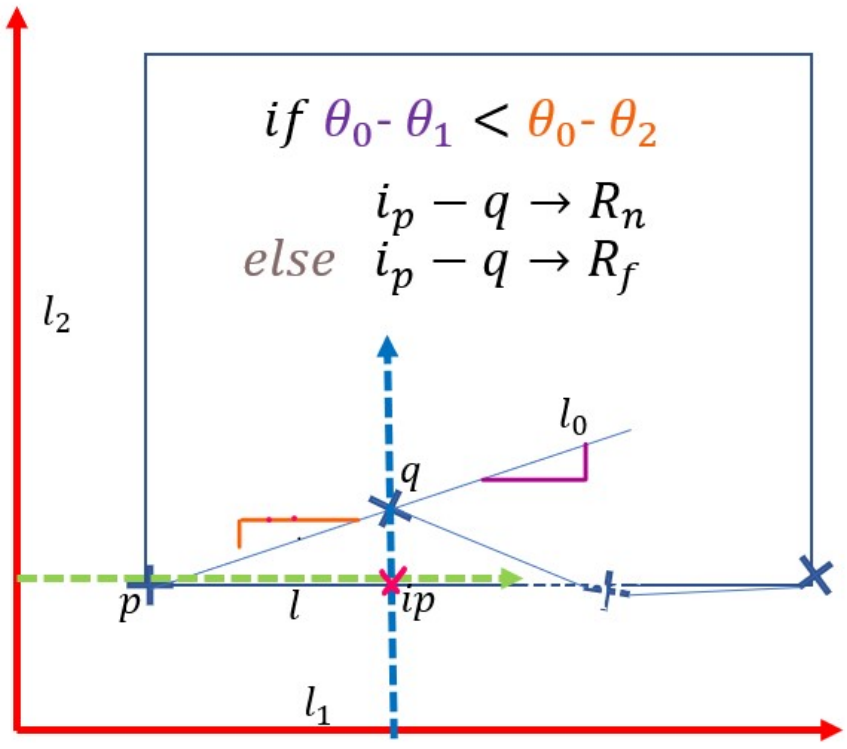


Figure 3.10: The initial stages of the target coordinate points selection using Method 1.

Method 1. Algorithm 1. 2: `snap()`

Procedure `snap(`*q, p, line,* D_{thresh}*,* R_n*,* R_f`)`**:**

begin

Find the line l_1 passing through the point p on the contour which is perpendicular to the given *line*(l)

Find the other line l_2 passing through the point p on the contour which is parallel to the given *line*(l)

Find a line l_0 passing through point p and point q

Calculate the slope of lines l_0, l_1, and l_2 as θ_0, θ_1 and θ_2

if $\|\theta_0 - \theta_1\| < \|\theta_0 - \theta_2\|$ **then**

    Find the line *temp* passing through q and perpendicular to l_1

    Use this line *temp* and l_1 to find the intersection point ip

else

    Find the line *temp* passing through q and perpendicular to l_2

    Use this line *temp* and l_2 to find the intersection point ip

end

Calculate the distance d from point q to point ip

if $d < D_{thresh}$ **then**

    $ip \leftarrow$ point between ip and q that splits the line ip-q in ratio R_n

else

    $ip \leftarrow$ point between ip and q that splits the line ip-q in ratio R_f

end

return ip

end

Even following through this algorithm, the algorithm may not respect the orientation of the given set of three-consecutive points, because three points will support controlling the direction, as two of the points can control the voting direction based on the previous points in a repetitive process. Therefore, this work

proposes another algorithm that provides an improvement over the snapping algorithm, which can be seen in the following section.

Updating Snapping Algorithms, Method 2

As a new updating of Method 1, using the key-points predicted by the Mask R-CNN improves the result from the predicted fixed contour. The new procedure `FixContourV2()` respects the properties of the building. This updated procedure seeks to find the position of the point c, based on the point b. Consequently, it allows the procedure to expand the line in an outward direction with respect to the contour rather than an inward direction. This procedure is able to obtain sharp angles on the corners and straight lines on the edges. For a point to be a corner, the given set of points $\{a, b, c\}$ should form a perpendicular at b; while for b to be an edge, the given set should form co-linear points. The proposed Algorithm `snapV2()` is similar to `snap()`, the only difference is the comparison to the reference line as can be seen in Figuer 3.9. In the updated version, the comparison is omitted because it checks the alignment with a reference using a different procedure.

Method 2. Algorithm 2. 3: `FixContourV2()`

Procedure `FixContourV2(`*contour*`):`

Data: *contour*, a list of points representing contour

Result: *new_ contour*, a list of updated contour points

begin

1 Sort the contour so that the first point is the point with lowest value of ycoordinate

2 Create a list new_contour to hold new points

3 **for** $i = 0; i \leq$ (no. of points)$; i++$ **do**

4 take a, b, c as i, $(i+1)$ and $(i+2)$ point in the contour

5 calculate $\angle ABC$

6 find reference line *ref*

7 find l through b and c, and check it's alignment with reference line *ref*

8 **if** *l aligns with ref* **then**

9 find line l_b parallel to *ref* passing through b

else

10 find line l_b perpendicular to *ref* passing through b

end

11 find *ip* of perpendicular to l_b passing through c.

12 calculate $direction = (c - b) \times (ip - b)$

13 **if** *direction* > 0 *and* $\angle ABC$ *is corner or edge* **then**

14 $b =$ *snapV2(b, c, line)*

15 update $(i+1)$ point $\leftarrow b$

else

16 $c =$ *snapV2(c, b, line)*

17 update $(i+2)$ point $\leftarrow c$

end

18 append b to the *new_ contour*

end

end

.

$$b = \begin{cases} \text{corner}, & \text{if } \|\theta_b - \pi/2\| \leq \theta_C \\ \text{edge}, & \text{if } 0 \leq \theta_b < \theta_E \text{ or} \\ & \pi - \theta_E < \theta_b \leq \pi \end{cases} \tag{3.23}$$

where
$\theta_b = \angle ABC$
$\theta_C =$ Angular threshold for corner point
$\theta_E =$ Angular threshold for edge point
Ideally, it is desirable for θ_C and θ_E to be equal to zero

Method 2. Algorithm 2. 4: `snapV2()`

Procedure *q,p,line,* D_{thresh}, R_n, R_f**:**

begin

1 Find the line l_{prl} passing through the point p on the contour which is parallel to the given *line*(l)

2 Find the other line l_{prp} passing through the point q on the contour which is perpendicular to the given *line*(l)

3 Use l_{prl} and l_{prp} to find the intersection point ip

4 Calculate the distance d from point q to point ip

5 **if** $d < D_{thresh}$ **then**

6 $ip \leftarrow$ point between ip and q that splits the line ip-q in ratio R_n

else

7 $ip \leftarrow$ point between ip and q that splits the line ip-q in ratio R_n

end

8 **return** ip

end

3.5.3 Determining Building Orientation

The snapping algorithm requires a suitable reference to snap the point in the contour to a new location. Therefore, it is necessary for the reference to be as robust as possible. A good reference will be able to tell the orientation of the building and also acts as a single point of reference. By considering the azimuth or vertical to be a reference, the directional property of the building can be inferred. This directional property allows the direction of the building's outlines to be identified. It is only possible to navigate the contour to fix its shape with the proper directional orientation of the building lines. Thus, finding a good reference is a key-part of the solution to estimating the shape of the building.

The `FindReference()` procedure groups the predicted key-points along with the contour points. It then scans through the points and groups them at a certain distance, d_{min}. This ensures that the key-points are not cluttered and are well separated. It then sorts the key-points according to the clockwise direction, and proceeds to find the reference of the building using the lines through the adjacent keypoints.

To find the orientation of the building, the algorithm calculates the inclination (azimuth) of each line joining the adjacent keypoints with respect to North (i.e., vertical axis). Therefore, a single point of reference is obtained that can be used to compare the inclinations of all of the lines. Using these angles, it selects the reference that supports the maximum number of angles. It starts at angle A, and calculates support for this angle under the threshold θ_{Az}. If the angle A supports maximum azimuth angles, it then selects the algorithm and otherwise it proceeds with a new candidate as shown in Figure 3.11.

Method 2: Algorithm 2. 5: `FindReference()`

Procedure `FindReference(`*list of azimuth angles,* θ_{Az}`)`**:**

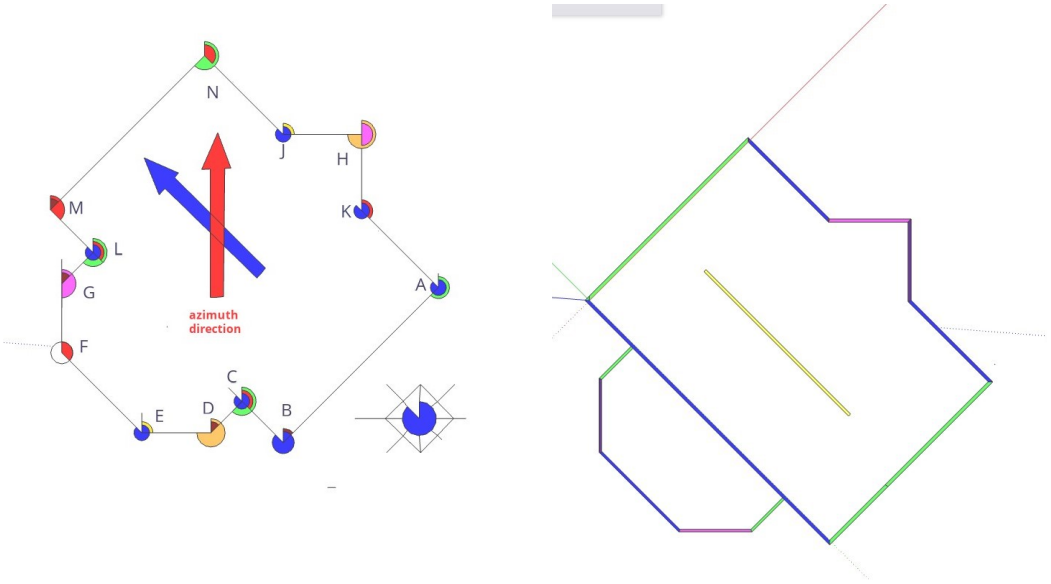
Data: list consisting the azimuth angles of lines in the contour; θ_{Az} angle under which the reference is consider to be valid

Result: A line representing the orientation of the building.

begin

1 create a list of angles from (0 to 90)
2 create a table that holds angle, corresponding support as well as error
3 initialise all supports as well as error in the table to 0
4 **for** θ_a *in list of angles* **do**
5 **for** θ_{az} *in list of azimuths* **do**
6 diff $= \|\theta_{az} - \theta_a\|$
7 **if** $diff \leq \theta_{Az}$ **then**
8 $Error_{\theta_a} \leftarrow Error_{\theta_a} + diff$
9 $Support_{\theta_a} \leftarrow Support_{\theta_a} + 1$
end
end
10 From the table select reference with lowest error. (In case of ties, increase the angle precision of the two lowest estimates and perform operation from 3-9)
end
11 Using the angle estimate the equation of line through origin as reference line

end



(a) (b)

Figure 3.11: Demonstration of the building angles with different colours: (a) similar angles with respect to the north azimuth have the same colour; the high-voting angles represent the orentation of the building. In (b), the similarity represents the parallel outlines of the perimeter.

3.6 Experimental Setup

This section outlines the experimental setup followed in the study. It provides information on the inclusion criteria for issues in the study, what the parameters were, and how they have been tested. This section then describes each model's input variables and parameters, and it explains why they were chosen for the purposes of this study. The tool used for data augmentation is also represented by the following mathematical procedures to carry out post-processing. This section also presents the methods used to analyse the data. Finally, the technical issues observed in the process are also discussed.

3.6.1 Study Area and Data Source

This research uses a dataset of VHR satellite imagery that shows exactly how the images were measured. These images represent an area from the Vaihingen city in Germany and are taken by the German Association of Photogrammetry and Remote Sensing (DGPF). Unlike normal images, these images are Near-Infrared Region (NIR) images. They differ from normal RGB images because instead of the normal blue channel, the images contain a a near-infrared channel; as shown in Figure 3.12. In the dataset, there are 33 patches (of different sizes), which are called tiles. Table 3.1 contains more detail about all of the patches, each consisting of a true orthophoto (TOP) extracted from a larger TOP mosaic. Each tile represents a major area of the city, as shown in Figure 3.12. For this study, the Vaihingen dataset was chosen as a case study area. The urban and suburban regions of Vaihingen consist of three test areas for which reference data for various object classes are available, representing a valid test for classification algorithms. The tiles are roughly 2500×2000 pixels and are taken with ground sampling distance of 9cm. For each tile in the dataset, the challenge community provides digital surface models (DSM) images and also semantic labels for the object in the scene. The dataset has six classes that have been defined as, impervious surfaces (RGB: 255, 255, 255), building (RGB: 0, 0, 255), low vegetation (RGB: 0, 255, 255), tree (RGB: 0, 255, 0), car (RGB: 255, 255, 0), and clutter/background (RGB: 255, 0, 0). This work is completely based on a 2D approach as shown in Figure 3.13. There are three RGB bands in the TOP 8 bit TIFF files, which correspond to the near infrared, red and green bands of the camera. The DSM are TIFF files that have one band, with the grey levels that correspond to the DSM heights encoded as 32 bit float values. As the TOP and the DSM are defined on the same grid, it is not required to consider the geocoding information throughout the processing.

Most of the other prevalent works extract the features from the 3D spatial in-

formation as obtained in the form of DSM images. However, this work relies completely on the tiles and segmentation map to perform the classification and feature extraction for the building class.

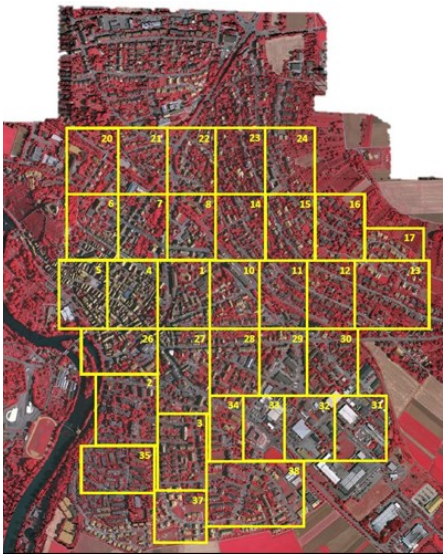
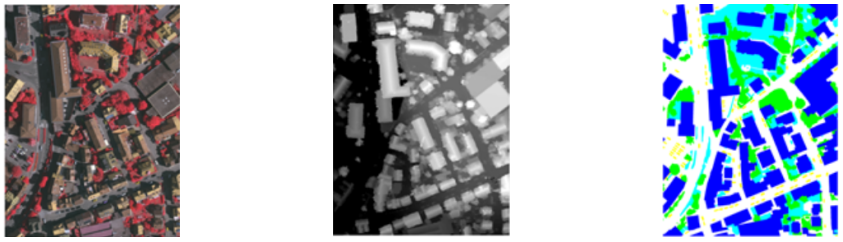


Figure 3.12: Study area of each patch in the Vaihingen dataset[3]

Training a neural network architecture requires a significant amount of data pre-processing and feature engineering. This section will summarise the pre-processing steps that are required to train the network.



(a) Orthophoto (b) DSM (c) Ground truth

Figure 3.13: Sample of semantic object classification contest patches

Table 3.1: Details of patches of the Vaihingen

TOP	DSM	N_{row}	N_{row}	GT
top_mosaic_09cm_area1	dsm_09cm_matching_area1	1919	2569	top_mosaic_09cm_area1
top_mosaic_09cm_area2	dsm_09cm_matching_area2	2428	2767	
top_mosaic_09cm_area3	dsm_09cm_matching_area3	2006	3007	top_mosaic_09cm_area3
top_mosaic_09cm_area4	dsm_09cm_matching_area4	1887	2557	====
top_mosaic_09cm_area5	dsm_09cm_matching_area5	1887	2557	top_mosaic_09cm_area5
top_mosaic_09cm_area6	dsm_09cm_matching_area6	1887	2557	====
top_mosaic_09cm_area7	dsm_09cm_matching_area7	1887	2557	top_mosaic_09cm_area7
top_mosaic_09cm_area8	dsm_09cm_matching_area8	1887	2557	====
top_mosaic_09cm_area10	dsm_09cm_matching_area10	1887	2557	====
top_mosaic_09cm_area11	dsm_09cm_matching_area11	1893	2566	top_mosaic_09cm_area11
top_mosaic_09cm_area12	dsm_09cm_matching_area12	1922	2575	====
top_mosaic_09cm_area13	dsm_09cm_matching_area13	2818	2558	top_mosaic_09cm_area13
top_mosaic_09cm_area14	dsm_09cm_matching_area14	1919	2565	====
top_mosaic_09cm_area15	dsm_09cm_matching_area15	1919	2565	top_mosaic_09cm_area15
top_mosaic_09cm_area16	dsm_09cm_matching_area16	1919	2565	====
top_mosaic_09cm_area17	dsm_09cm_matching_area17	2336	1281	top_mosaic_09cm_area17
top_mosaic_09cm_area20	dsm_09cm_matching_area20	1866	2315	====
top_mosaic_09cm_area21	dsm_09cm_matching_area21	1903	2546	top_mosaic_09cm_area21
top_mosaic_09cm_area22	dsm_09cm_matching_area22	1903	2546	====
top_mosaic_09cm_area23	dsm_09cm_matching_area23	1903	2546	top_mosaic_09cm_area23
top_mosaic_09cm_area24	dsm_09cm_matching_area24	1903	2546	====
top_mosaic_09cm_area26	dsm_09cm_matching_area26	2995	1783	top_mosaic_09cm_area26
top_mosaic_09cm_area27	dsm_09cm_matching_area27	1917	3313	====
top_mosaic_09cm_area28	dsm_09cm_matching_area28	1917	2567	top_mosaic_09cm_area28
top_mosaic_09cm_area29	dsm_09cm_matching_area29	1917	2563	====
top_mosaic_09cm_area30	dsm_09cm_matching_area30	1934	2563	top_mosaic_09cm_area30
top_mosaic_09cm_area31	dsm_09cm_matching_area31	1980	2555	====
top_mosaic_09cm_area32	dsm_09cm_matching_area32	1980	2555	top_mosaic_09cm_area32
top_mosaic_09cm_area33	dsm_09cm_matching_area33	1581	2555	====
top_mosaic_09cm_area34	dsm_09cm_matching_area34	1388	2555	top_mosaic_09cm_area34
top_mosaic_09cm_area35	dsm_09cm_matching_area35	2805	1884	====
top_mosaic_09cm_area37	dsm_09cm_matching_area37	1996	1995	top_mosaic_09cm_area37

These very high resolution satellite images cannot be directly applied to the neural networks without pre-processing the images. Therefore, the pre-processing step will resize the images in a reasonable size of 512x512 pixels. OpenCV provides a resizing function, called *cv2.resize()*, that is able to resize the image matrix to the desirable size. As the resizing operation is performed, it is important to use methods that are able to factor the extra noise that may come from resizing of the data. To progress with the computational part of the problem, this thesis will

[3]source:https://*www2.isprs.org/commissions/comm2/wg4/results/a1*$_d$*etect/*

use different open source frameworks because they provide the necessary functions and modules that will be used in this project. This software, along with a brief description, is listed in the Appendix (under software setup).

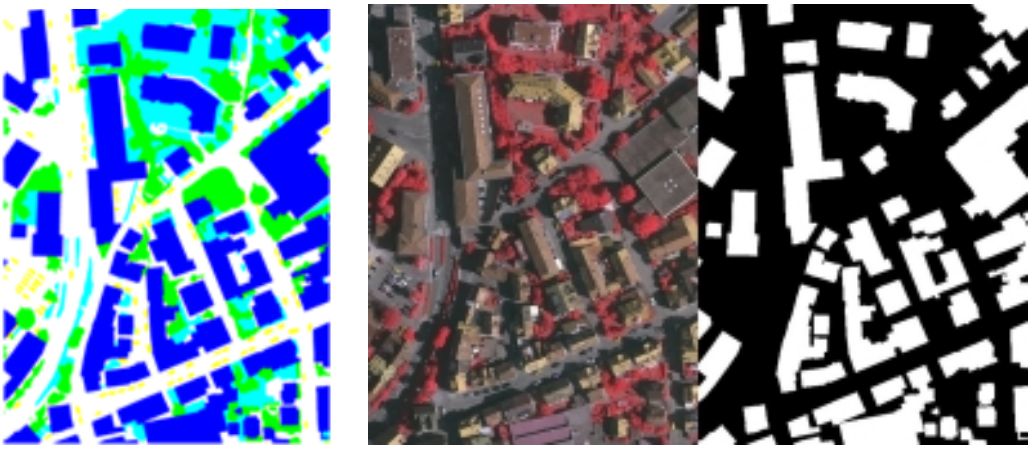
3.6.2 Training Setup and Pre-processing

An Nvidia T4 GPU with 16 GB of memory is used to train the model. However, because of the memory restriction imposed by the deeper models, the batch size has been limited to two. This study could not fit a batch size of eight into the GPU memory to compare both larger and smaller models. Consequently, a batch size of two was used to achieve comparable results for both models. The dataset that is used contains different classes of objects. However, the main objects of interest are the "buildings" class. Therefore, the primary task here is to extract a binary segmentation map that contains only buildings. The image contains different colours to represent the mask for different objects. In the case of the buildings, the default ground truth colour for the building mask is blue, as shown in the Figure 3.14. To extract the buildings from the images, it is desirable to threshold the image within the value of the blue colour. The *cv2.inRange()* function that is provided by the OpenCV can be used to perform the thresholding operation, which is responsible for filtering the pixels in images within the given boundary. Once the buildings are extracted from the ground truth, the entire image is converted into a binary mask, where a pixel value of 255 represents a building and a pixel value of 0 represents a non-building area, as shown in Figure 3.14-(b).

After the segmentation mask has been obtained, it can be used to extract the contours from the segmentation. These extracted contours allow the pre-processing algorithm to extract the keypoints present in the building. The OpenCV library provides a function called *approxPoly()*. Given a well-defined polygon, this function can estimate the smallest polygon that is able to define the shape of the

building. Consequently, the corners from the polygon provide the information for the required keypoints. This keypoint information, along with the segmentation mask, are stored in a JSON file that is later used to process the data and images during the training step.

Because the number of pixels directly correlates to the size of the network, this study had to limit the size of the image training image to 512×512 pixels. It was then able to train the network on a batch size of two images per batch. In addition, several data augmentation techniques were used because of the limited amount of images (e.g., rotation, cropping and flipping). These augmentation techniques are applied within the data processing unit of the training code. However, in case of GANs, the training size is 256×256 pixels because of the memory restriction imparted by the network. This happens because the attention function that is used in the attention-GAN has a quadratic complexity in terms of space.



(a) (b)

Figure 3.14: Illustrating the beginning of the training setup and pre-processing, where: (a) is the actual ground truth mask, and (b), on the right, is the same building target converted to binary mask, and concatenated with the top mosaic image on the left.

3.6.3 Deep Learning Hyper-parameter Setup

To proceed with training the network, this study selected 29 areas, as can be seen in Table 3.1, and some areas were left for testing (i.e., areas 8, 13, 31, and 34). The learning rate of the network was initialised at 0.00025 and the decay rate was set to 0.85 at every 15,000 iterations, up to 100,000 iterations. This study first trained the model on 75000 iterations. Because a satisfactory result was not obtained, this study increased the number of iterations to 100,000. The following section will describe this point in more detail. Table 3.2 below contains more experimental parameters on this study with a brief description about each parameter (see section 3.6.5 for more details).

3.6.4 Model Setup

As mentioned at the beginning of this chapter, the methods used in this research to perform image detection are based on the literature review. Therefore this thesis has reused and developed two models to meet the aims and the objectives for executing image detection. The first model was GAN*pix2pix* [299] (which is one of the latest deep learning methodologies), and it is a variant attention-GANs. The second one was*detectron2* [340], which was used for the part of the Mask R-CNN. Each of those models has different structure requirements for implementation, as demonstrated in the previous section. This section will demonstrate each one individually in more detail.

Setup Training and Test Data of the GAN model

Phillip Isola et al. in their 2016 paper titled "Image-to-Image Translation with Conditional Adversarial Networks" and presented at CVPR in 2017 presented the *Pix2Pix* model, which is a type of conditional GAN where the generation of the output image is conditional on input—in this case, a source image. The generator

model for the *Pix2Pix* GAN is implemented as a U-Net. The discriminator is provided with both a source image and the target image, and must determine whether the target is a believable transformation of the source image. Decent results can be obtained reasonably quickly and on small datasets on some tasks. For example, to learn to generate facades (see, for example, [299]), the model was trained on just 400 images for about 2 hours (on a single Pascal Titan X GPU). However, for more complex puzzles, it may be essential to train on far larger datasets and this can take many hours or even days. This research has implemented the *PyTorch* version model, which is under active development and can produce results comparable to the Torch version.

- **Preprocessing step**: to convert ground truth into ground truth as masked with the building in grayscale.
- **Augmentation step:** Because the case study has a limited number of ground truth images, several augmentation techniques (e.g., rotation, cropping, and flipping) were used.
- **A Python script** is provided to generate training data in the form of couples of images A, B, where A and B are two different depictions files of the same underlying view. For example, these might be pairs ground truth, ortho images or DSM, ortho images. They can then learn to translate A to B or B to A.
- **Create folder path**: Data with subfolders A and B should each have subfolders for training, validation, test, and output. In path to data (A)train, put training images in style ground truth masked images. In path to data(B) train, put the corresponding images in style (orthophoto). Repeat same for other data splits (validation, test, output).
- **Concatenated step:** Corresponding images in a pair (A, B) must be the same size and have the same filename. Once the data is formatted in this

way, the next step was ready (which is called the concatenated step).

- **Extract edges:** Python scripts are used to extract rough edges from image by run scripts HED to compute edges.

- Post-process been used to refine the output edges by using Python scripts, called *PostprocessHED.m* to simplify edges.

- **Calculate loss L1 error:** errL1, errG and errD referring to Equation (3.16, 3.17).

  In this research stage, the results obtained from the GANs are stochastic in nature. This means that they are not suitable for the task of remote sensing, where the building location barely changes. *SAGAN* was proposed to add some localisation to the output to solve these issues.

- *SAGAN* The hypothesis was to add *attention* block to GAN. The assumption is that the network will be able to figure out how buildings are structured and where they are located. Unfortunately, no significant improvement happened (which will be described in the next chapter).

Setup Training and Testing Data of the Mask R-CNN model

The method reused was *Detectron2*, which was introduced by Facebook AI Research (FAIR). This method came with an advanced open source library, giving many object detection and segmentation problems. *Detectron2* is based upon the Mask-RCNN benchmark for the part of the Mask R-CNN to detect key-points, as well as for instance segmentation. The implementation was in *PyTorch*, procedure follows:

- The first step is to install the *Detectron2* and required dependencies libraries (see the appendix).

- *Detectron2* requires the data in a specific format (i.e., JSON files). *Detectron2* over helper functions will input the image directory folder path to convert the dataset to JSON files format. The JSON files are then opened and loaded. Each image is read from the path, and its height, weight, file name, and image ID are stored in a dictionary called *record*. Next, the bounding box details are stored in another dictionary, called *obj*. At the end of each loop, *record* is appended to a list called *dataset_dicts*. Similarly, the bounding box dictionaries are also appended to a list *objs*. This list will be assigned as the value against the *annotations* key in the *record* dictionary. Each of these dictionaries is then appended to a final list, which will be returned. The next step is to register these training and validation datasets using the *DatasetCatalog.register* and the *MetadataCatalog* method. It starts by importing the *DefaultTrainer* from the engine module of *Detectron2*. The dataset and other parameters are defined, such as batch size and classes (building in this study), and then the model is initialized with with pre-trained weights and trained further. The max iterations parameter will vary depending upon the size of the dataset and the complexity of the task.

The pre-trained models are read for prediction and testing up to these steps. In this thesis, one extra step is added along with the segmentation map to detect corners as keypoints. As a result, predicting corner keypoints will allow new algorithms to be developed that can estimate the geometry of the building with the help of FCN to generate a heatmap for each of the keypoint and extract the points from the heatmap. Therefore, the loss on this network is now on each of the sampled ROI, and is given as:

$$\text{Loss}(L) = L_{cls} + L_{box} + L_{mask} + L_{kps}, \tag{3.24}$$

where, L_{cls} is classification loss, L_{box} is bounding-box loss, L_{mask} is segmentation mask loss and L_{kps} is the keypoint loss.

3.6.5 Post-processing Setup

When dealing with building structures, geometry is a vital piece of information because it allows engineers to properly estimate the size and shape of the building. Therefore, it is is necessary to improve the contours detected by Mask R-CNN by applying two step post-processing techniques. During the first step, CRF is applied to provide a smoothing effect to the detected contours. In the second step, a novel contour refinement techniques are proposed. These steps provide a better estimate approach of the shape of the buildings. For post-processing, there are a few parameters for which appropriate values have to be selected. In most of the experiments, the shape of the final contour from the algorithm is dependent on the quality of the reference line and the quality of the mask. When the backbone of the model was upgraded from ResNet50 to ResNet101, there was a 1.2%, 2.6%, 3.0% improvement in overall accuracy, F1-score and IoU, respectively.

To refine the shape of the contour, `FixContour()` requires a proper reference line. Because the reference selection uses voting algorithm to find the reference line, a threshold angle α_R has to be defined under which lines are to be considered as parallel. Ideally, this should be 0° but in this thesis it is 5°. As the experiment is run for different angle ranges, from [0°...10°], it was not possible to get suitable lines for angles $\geq 5°$. Therefore, the minimum angle that can reliably filter the parallel lines was used (i.e., 5°). As mentioned in the previous section, grouping the key-points provides the better estimation of the shape. Therefore, the minimum distance d_{min} between any two key-points has to be defined. In this thesis, values from $10 - 100$ at interval of 10 were used. The distance that can satisfy the minimum number of points that represents a building while maintaining reasonable inter-cluster distance was then selected. In this thesis, $d_{min} = 30$

provides a sufficient amount of key-points to reliably detect the reference lines and corresponding angle.

The algorithm requires the center point in the given triplet points to be classified as a corner point or an edge point. Ideally, the angles at corners should be equal to 90° while at an edge it should be 180°or 0°. However, in most cases there will be error on the classification of the lines. Therefore, this value would have to be limited according to Equation 3.23. The suitable parameter was found by setting θ_C and θ_E in the interval of 5° within the range of [0°...45°]. The angle was bound to 45° because it will mostly be an edge when $ABC < 45°$ and it will be a corner if $ABC > 45°$. Therefore, several values of θ_C and θ_E were tried and the count of the lines that support the building property was estimated. The threshold should be defined such that it can cover the maximum number of key-points as edge and corner, while still preserving the shape of the building. In some cases, some key-points may not contribute to being an edge or a corner. These angles should be avoided. The angles for which there is maximum support of the corners and edges were selected. After trying different angles in the range $[0°, ..., 45°]$, the final result used $\theta_C = 25°$ and $\theta_E = 35°$ because these angles satisfied the property of the corner and edge as defined by Equation 3.23.

To select the appropriate reference line, the defined reference angle should have maximum alignment with the all of the lines formed by connecting the keypoints. Therefore, the alignment under a threshold of θ_{Az} has to be calculated. In case the lines are perfectly aligned with the assumed reference line, θ_{Az} will be zero. But that is not the case, and upon experimenting with different value of θ_{Az} in the range of $[0, ..., 10]$, when the difference of the angle between line and reference is less than value of 2°, the reference lines were more accurate to the actual reference. This value was used for θ_{Az} because a large number of lines were not required to support the reference line. If a large number of lines support the reference, then a tilted reference will ultimately effect the accuracy. A summary

of the experimental setup can be seen in Table 3.2.

Table 3.2: Experimental parameters

Parameter	Brief Description	Value
Learning rate	Defines the speed at which the network learns	0.00025
Decay iteration	Defines the number of steps after which the learning rate starts to decrease	15000
Decay rate	Defines the factor by which the learning rate decreases	0.85
Region selection offset	Extra region outside of the predicted box that needs to be selected	50 pixels
Closing kernel size	Kernel size of morphological closing operation	11 pixels
Opening kernel size	Kernel size of morphological opening operation	11 pixels
Snapping distance threshold (D_{thresh})	Distance to be considered for the keypoint to be snapped	50 pixels
Snapping near ratio R_n	Ratio representing the nearest distance from the intersection point to the keypoint	0.15
Voting algorithm angle (α_R)	Angle used in the Method 1 to calculate reference	5°
Grouping distance Threshold (d_{min})	Grouping the close keypoints	30 pixel
Azimuth angle threshold (θ_{Az})	Maximum deviation for the angles to be consider as azimuth (vertical with reference to screen)	2°
Corner angle Threshold (θ_C)	Maximum angle from the vertical to be considered as corner refer to Equation 3.23	25°
Edge angle threshold (θ_E)	Maximum angle from the horizontal be considered as edge (refer Equation 3.23)	35°

3.7 Performance Metrics

To evaluate the performance, well-known performance metrics (i.e., Precision, Recall, F1-score, Intersection over Union, overall pixel accuracy, and structural similarity index measure) were used to evaluate the pixel- and object-based performance of the developed approaches. They were chosen because they have been applied in most studies in the field of machine learning, such as Aksoy et al. [342], Ozgun et al. [256], and Wang et al. [343]. Each metric measures different parts of the segmentation result. The following subsections illustrated their prin-

cipal parameters, components, and the equations that were used to evaluate the performance of this thesis and compare the state-of-the-art.

3.7.1 Precision

Precision is responsible for measuring the true building segmentation present in the output; meanwhile, recall measures the completeness of the segmented result. Consequently, the area of a building can be evaluated in a way that is accurately covered, as well as the area of the building that is fully covered.Note the following abbreviations in the equation: TP (true positive); FP (false positive); FN (false negative) (also see section 3.7.6). Therefore, mathematically:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.25}$$

3.7.2 Recall

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.26}$$

3.7.3 F1-Score

The F1-score, also known as Dice coefficient, measures the similarity of the predicted segmentation with the actual building segmentation.

$$\text{F1-Score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.27}$$

3.7.4 Intersection of Union

The IoU, which is also known as Jacard coefficient, is similar to F1-score. It quantitatively measures the location and size of the predicted building compared

to ground truth:

$$\text{Intersection of Union} = \frac{A \cap B}{A \cup B} = \frac{TP}{TP + FP + FN} \tag{3.28}$$

3.7.5 Structural Similarity Index Measure

Structural similarity index measure (SSIM) is a method that correctly predicts the perceived quality of digital television and cinematic images, and other types of digital images and videos. SSIM assesses the similarity of two images. The SSIM index is a complete reference metric, which measures or predicts image quality using an initial lossless or deformation image as a reference. SSIM is a perception-based model that treats image degradation as a perceived change in structural information while incorporating critical perceptual phenomena, such as luminance and contrast masking terms. Structural information refers to the idea that pixels have strong interdependencies, incredibly when spatially close. These dependencies contain vital structural information of the object's scene. Colour information masking occurs in image distortions (in this particular instance segmentations) causes them to be less visible in highlights and shadows. Meanwhile, contrast masking is a phenomenon in which deformations become less visible and where there is a significant activity in the texture image. Equation 3.33 can be calculated as the distance between two windows (x , y) of standard size NN. As mentioned earlier, the SSIM formula is based on three comparison components: luminance (l), contrast (c), and structure (s). Each of these components can be calculated individually, as follows [343]:

$$l(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{3.29}$$

where

$$c_1 = (k_1 L)^2$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{3.30}$$

where

$$c_2 = (k_2 L)^2$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x \sigma_y + c_3} \tag{3.31}$$

where

$$c_3 = (c_2 / 2)$$

where:

μ_x is the average of x;

μ_y is the average of y;

σ_x^2 is the variance of x;

σ_y^2 is the variance of y;

σ_{xy} is the covariance of x and y;

c_1 , c_2 two small constants to stabilise the division;

L the dynamic range of the pixel-values ($L = 255$ for 8 bits/pixel gray scale images);

$k_1 = 0.01$ and $k_2 = 0.03$ by default.

SSIM is then a weighted combination of equations (3.29, 3.30 and 3.31):

$$\text{SSIM}(x,y) = \left[\; l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma \; \right] \tag{3.32}$$

where, α, β, γ are parameters specifying the significance of the three components. When setting the weights $\alpha = \beta = \gamma = 1$, the resulting SSIM index is given by:

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.33}$$

3.7.6 Overall Accuracy

The overall accuracy gives a measure of the correctly labelled pixels.

$$\text{Overall Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.34}$$

Here, TP true positive is where the pixels are present in the ground truth and the predicted building, producing a true positive; FP false positive is where the pixels are absent in the ground truth building but present in the predicted building; FN false negative refers to the amount of pixels present in the ground truth but absent in the predicted building; and TN true negative refers to the amount of pixels present in the ground truth but absent in the predicted building.

3.8 Summary

This chapter has reviewed two types of deep learning -GAN and Region-based convolutional neural networks using Mask R-CNN. Each contains several levels of post-processing techniques steps. This section will critique them, and explain the advantages and disadvantages of each type, followed by the operations performed during and after the training stage. The aim is to discover which nomination should be focused on in the subsequent chapters and why this nomination was made.

- The first group of the methodology used GANs.
- The first level was only GANs concatenated with ground truth. When using GANs, there may be a slight fluctuation in the generated images conditioned on the input image's pre-processing.
- The model's structures based on GANs Networks does not contain any localisation information.

- A shortage in ground truth images. These issues were solved in the experimental stage by using the following steps:

- More augmentation of ground truth images was created to address the lack of ground truth samples.

- To obtain localisation information, an Attention block on top of the GAN network was used as SAGAN.

- The second level is based on *SAGAN*. As mentioned in the first level, GAN networks do not contain any localisation information. Therefore, an attention block was used to provide location information to perform the attention. The results afforded better accuracy than GAN-based techniques and proved the hypothesis that self-attention will provide better accuracy. The attention module in the upsampling block is designed to direct the model's attention to the location that it should focus on when shaping the segmentation.

- The third level is based on *SAGAN* with post-processing using CRF. This level improves the accuracy of the building reconstruction by focusing on improving extracted rooftop masks by assigning a higher cost to the cluster of interrelated parts using CRF, which aims to minimise the unary cost with the pairwise cost. The CRF improved segmented areas of the rooftop are further processed to extract rectangular contours, and then line detection follows.

- The fourth level is based on *SAGAN* with post-processing using CRF and fitting algorithm Method 1. This level focuses on using another inherent feature, the direction of the lines, to select the best lines representing the rooftop boundaries. Method 1 provides better results compared to the previous levels. Up to this point, the proposed method and post-processing techniques rely heavily on improvement based on the detection of optimal

lines and contours to improve the accuracy of predicted models. Consequently, there was a need to explore another method by which the proposed research might produce competitive results compared with state-of-the-art methods in remote sensing (as described in the following section).

- A description of the methodology used in second group follows:

- The first level used Mask R-CNN architectures. The new model relies on the meta-architecture as given by Mask-RCNN. In this new concept, the methodology used a neural network to detect the keypoints. There is very little work in this field to perform segmentation by detecting the keypoints. At this level, the search will rely on the Mask R-CNN architecture. The architecture will use the ResNet50 backbone, with Feature Pyramid Network (FPN) as head of the network to detect the keypoints. This custom head will use the "attention block" from the previous work. Because keypoints are non-local parameters, the attention blocks will be allowed to learn these parameters. However, they were not defined because the loss function did not work. This idea inspired the search and was used to perform building segmentation and keypoint extractions using two head Mask-RCNN and modified snap-contour algorithms. The *SAGAN* based segmentation neural network was replaced with a mask R-CNN network to preserve and extract finer details at the segmentation stage and provide it to the second mark-RCNN network. A modified snap-contour algorithm was proposed to help improve the accuracy by looking at the keypoints, control points, and the prediction model provided by a two head Mask R-CNN.

- The second level was Mask R-CNN using two backbones (i.e., ResNet-50 and ResNet-101). In general, the result of this result was similar to that of the previous method. The contour is extracted, and the close polygon is estimated.

- The third level was based on Mask R-CNN two backbones (ResNet-50,ResNet-

101) and post-processing using CRF. Mask R-CNN uses two backbones (ResNet-50 and ResNet-101) and two post-processing methods (CRF and method 1). using the snap-contour algorithm. After applying the mask R-CNN, the resultant contours are further processed using a snap-contour algorithm (details of the algorithm provided Section 3.6.1).

- The fourth level was based on Mask R-CNN using two backbones (ResNet-50 and ResNet-101) and two post-processing methods (CRF and method 2). This level proposes to modify the snap-contour algorithm method 1 to include the directional information. The algorithm is divided into two parts (details of the algorithm are provided Section 3.6.2). The procedure snap (q, p, line, buffer) is responsible for snapping the point 'q' to the line passing through 'p'. The buffer controls where the new point should lie. The centre of the contour is then used to sort the points based on their polar angles. Once they are sorted, the horizontal reference angle and vertical reference angle are found. The `FixContour()` algorithm performs remarkably well when the reference lines detect support in the orientation of the building.

Finally, after considering many factors to setup the experimental parameters, as can be seen in Table 3.2, the second group of algorithms in method 2 addressed the research aim. The improvements can be seen when referring to the matrix results in terms of IoU, where it improved by more than 2%; also, precision improved by more than four percent. Note that the focus is on IoU and precision because they reflect how accurate the prediction of the feature is compared with the ground truth; in particular IoU with regard to direction. The findings are comparable with state-of-the-art remote sensing, which will be explained next in Chapter Four.

Chapter 4

Results

4.1 Introduction

This chapter describes the results from the deep learning architectures, referring to the methodology used, and is divided into two main groups. Each group has different levels of outputs related to the methods that have been used for each level. For simplicity, these findings are as follows:

- The first group obtains the results of the outcome of the GAN *pix2pix* model and ASGAN, followed by two stages of post-processing:
  1 Output based on GAN *pix2pix* model.
  2 Output based on SAGAN .
  3 Output based on SAGAN with Post-processing using CRF.
  4 Output based on SAGAN with post-processing using CRF and fitting algorithm Method 1 .
- The second group obtain the results of the outcome based on Mask R-CNN architectures:
  1 Output based on Mask R-CNN architecture.
  2 Output based on Mask R-CNN using two backbones (ResNet-50 and ResNet-101).
  3 Output based on Mask R-CNN two backbone (ResNet-50 and ResNet-101) and post-processing using CRF .

4 Output based on Mask R-CNN using two backbone (ResNet-50 and ResNet-101) and two post-processing (CRF, method 1).

5 Output based on Mask R-CNN using two backbone (ResNet-50 and ResNet-101) and two post-processing (CRF, method 2).

To illustrate the results across various cases, area 31 was selected as an example for visualisation purposes. However, the model provides similar results in every area that is present in the test set (which is attached in the Appendix). As can be seen in Figure 4.1, ground truth as per the source, provides the measurement taken from the ground, and in some cases, it does not match exactly with the rooftop of the building (as seen in a & b above); in addition, the guttering, as an extension, can affect the picture. However, the matrix used for evaluation is not affected by these issues with regard to the extra information, including the dataset, such as directions.

4.2 Results of the GAN-based Approaches

As mentioned in the introduction of this chapter, the first group obtained the results of the outcome of the GAN-based approaches, which will be demonstrated and followed by a brief description. To compare the results across various cases, area 31 was selected. However, the model provides similar results in every area which is present in the test set attached in the appendix.

1 **GANs *pix2pix*-based model output**

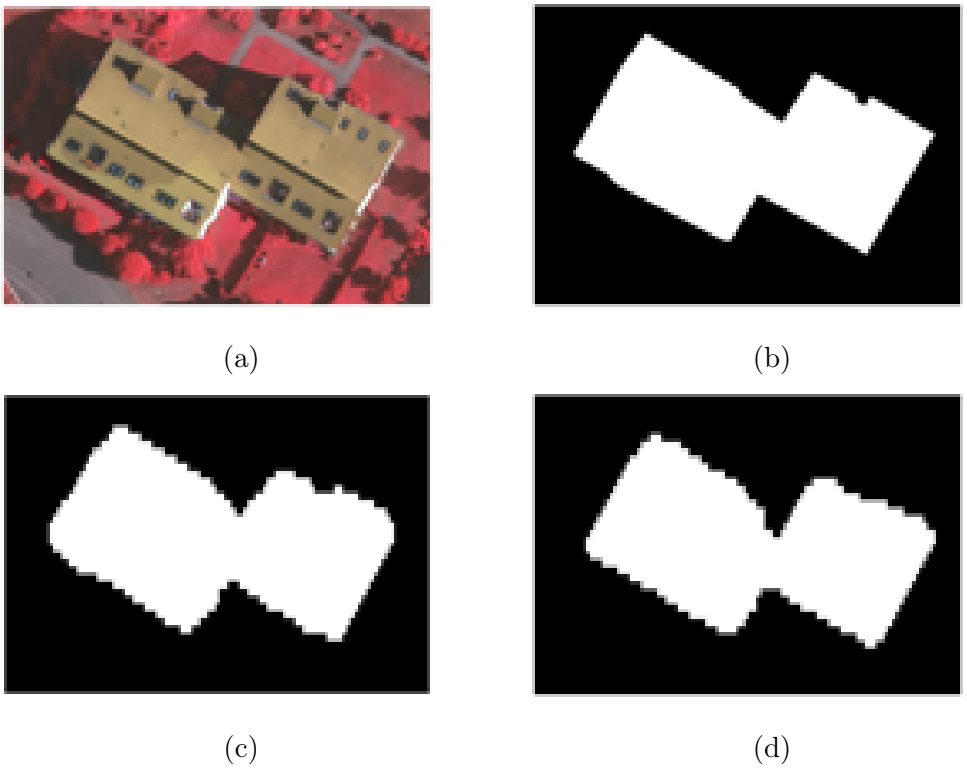
The results of the output based on GAN networks without any post-processing are shown in Figure 4.1.

(a) (b) (c)

Figure 4.1: Outputs obtained from the GAN *pix2pix* model, where (a) is the original image from area 34, (b) is the ground truth, and (c) is the output of *pix2pix* GANs.

2 **Output-based on SAGAN**

Figure 4.2 shows the output based on GANs and SAGAN.



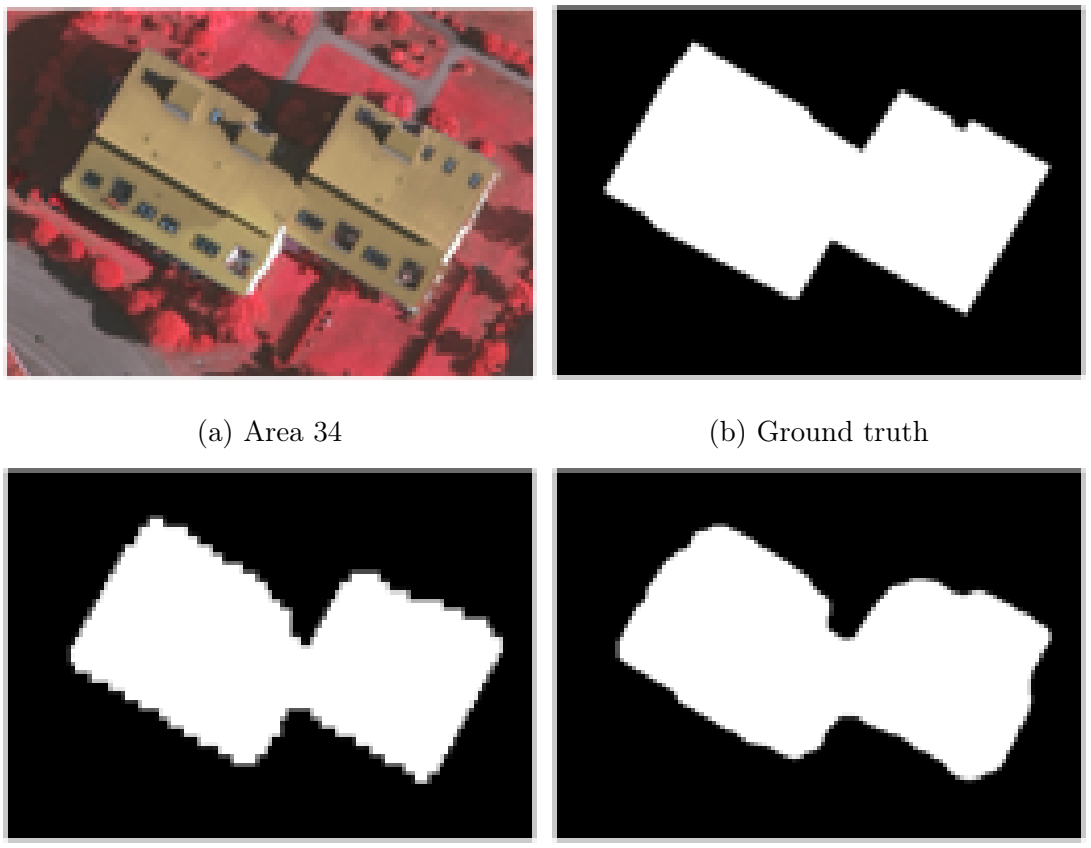
(a) (b)

(c) (d)

Figure 4.2: Outputs obtained from SAGAN, where (a) is the original image from area 34, (b) is the ground truth, (c) is the output of the GANs, and (d) is the output based on SAGAN.

3 **Output based on SAGAN with post-processing using CRF**

Figures 4.3 show the output based on SAGAN with post-processing using CRF.



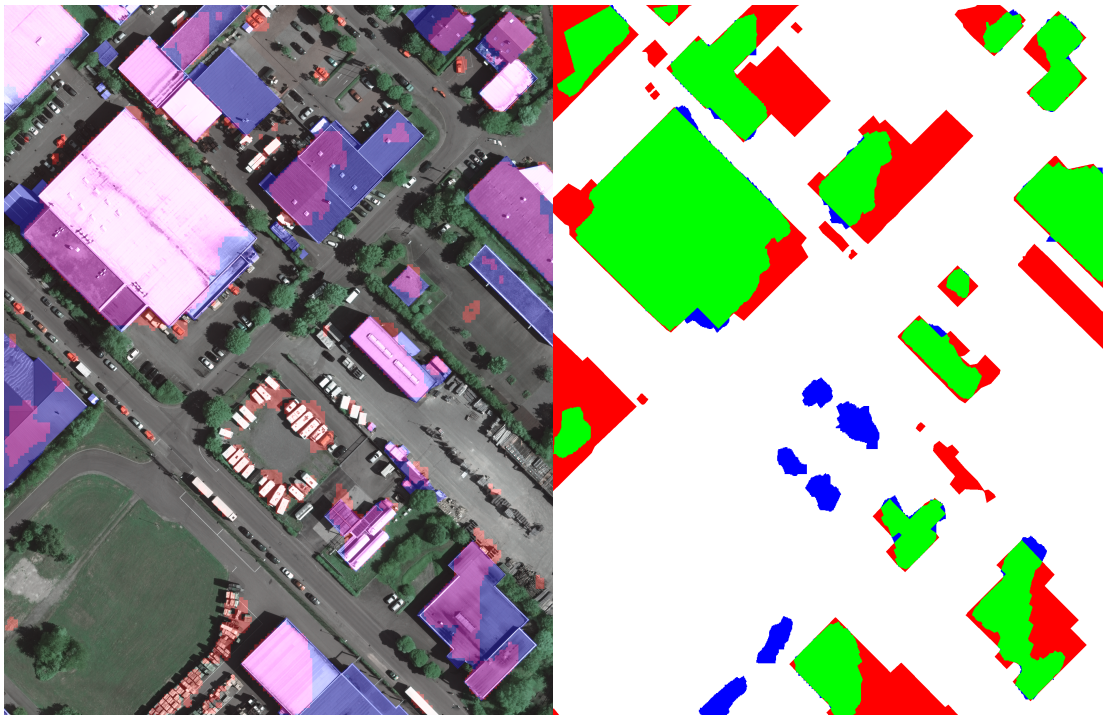
(a) Area 34 (b) Ground truth

(c) SAGAN output (d) CRF

Figure 4.3: The output of the GANs, where (a) is the original image from area 34, (b) is the ground truth, (c) is based on SAGAN, and (d) after applying CRF.

The previous example showed the output obtained from GANs and SAGAN, followed by an application of CRF. There is a stark difference between the two outputs in terms of the building shape position. Just by application of CRFs alone, it was possible to remove some of the false positives; as can be seen in Figures 4.4. However, this does not provide a clue or direction

of how to proceed in finding the proper shape of the building in the search space. The complexity of the problem also increases. Therefore, to ease this task, this research proposes to predict the keypoints along with the contours from the output of the detection boundaries.

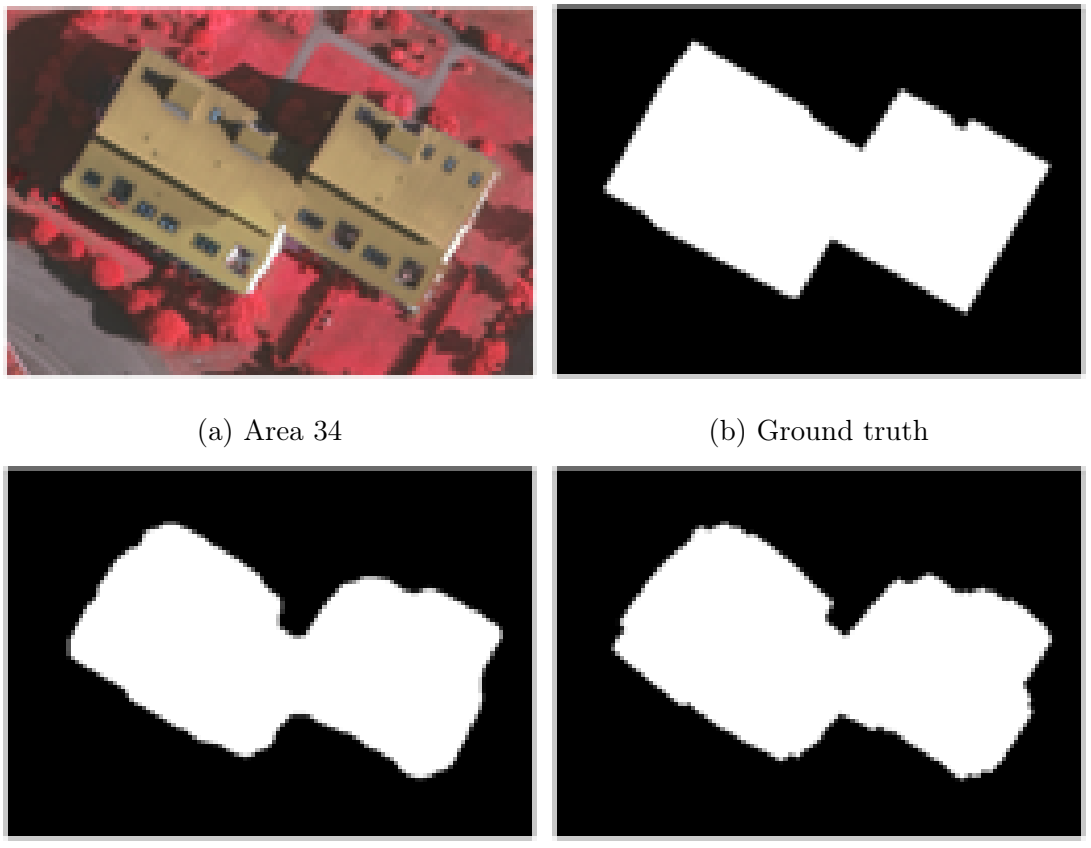


(a) (b)

Figure 4.4: Output obtained from GANs, where (a) is the output from SAGAN, and (b) is the output after application of CRF where green color represents a true positive (TP), blue color represents false positive (FP), and red color represents a false negative (FN) compared with the ground truth.

4 **Output based on SAGAN with post-processing**

The following results shown the output based on SAGAN after applying CRF and post-processing fitting algorithm (Method 1), as shown in Figure 4.5



(a) Area 34 (b) Ground truth

(c) Using CRF (d) Using Method 1

Figure 4.5: Outputs obtained from SAGAN with CRF post-processing and a line fitting algorithm Method 1.

As described in the methodology, different techniques were used on the results obtained by using GANs. Given that the output from the GANs does not have keypoints, it is not possible to apply the snapping algorithm with Method 2.

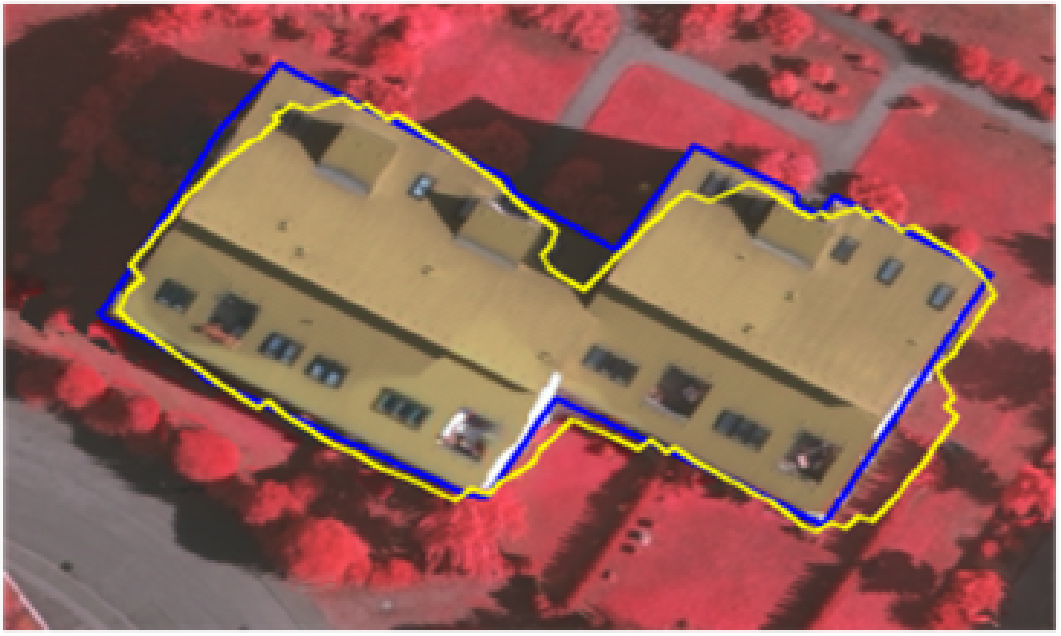
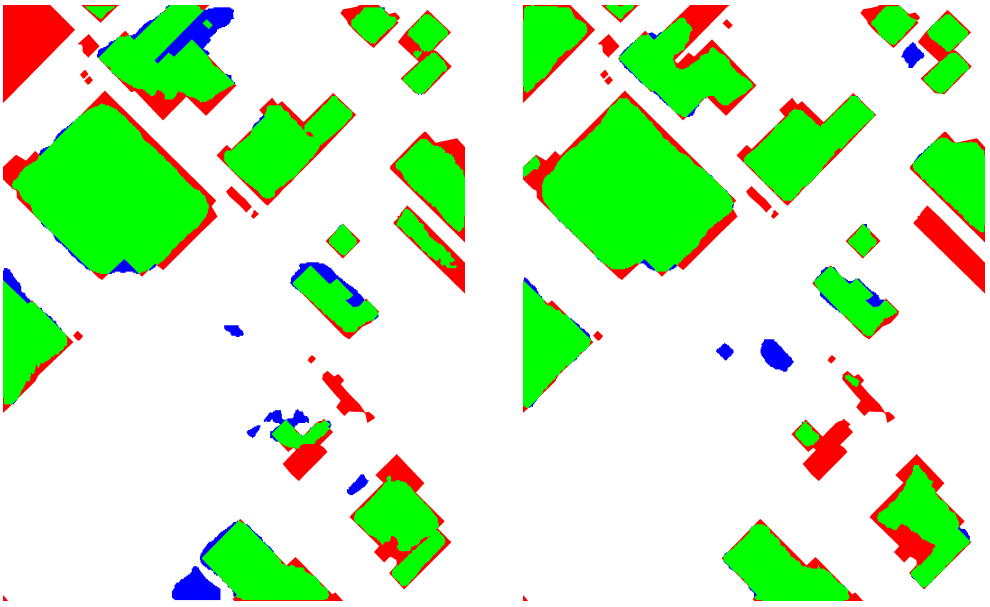


Figure 4.6: Previous result: the white line is post-processed contour (Method 1), yellow is predicted contour (SAGAN), and blue is ground truth.

4.3 Output Based on the Mask R-CNN Architecture

This research has been used pre-trained *detectron2* model and added two different structure sizes (i.e., ResNet-50 and ResNet-101) as a backbone of Mask R-CNN. As the size of the model increased, so did the performance of the model. This can be seen in Figure 4.8, which shows the effect of this increase on the final output based on Mask R-CNN without any post-processing.the Figure4.8-(a) demonstrates the closer view of the building, showing the true positive (b)shows the same areas and the increase in true positive, which reflects how a bigger model can produce more accurate detection in terms of TP.

Figure 4.7: Output image based on Mask R-CNN area 8.



(a) Backbone using ResNet-50 (b) Backbone using ResNet-101

Figure 4.8: Outputs obtained from Mask R-CNN with different backbone sizes in area 31. Green color represents true positive, blue represents false positives and red represents false negatives.

The larger model is able to detect the important regions of the map and the building structure, which are on par with the original image. The accuracy of the model also improves comparing with GANs models, as can be seen in Table 4.1. The reason for the improvement of the segmentation comes from the design of Mask R-CNN, which is able to focus on the regions and extract the required mask from that particular region. Although most of the refinement comes from the further post-processing, Mask R-CNN cannot address the issue of the geometrical structures present within the building architectures.

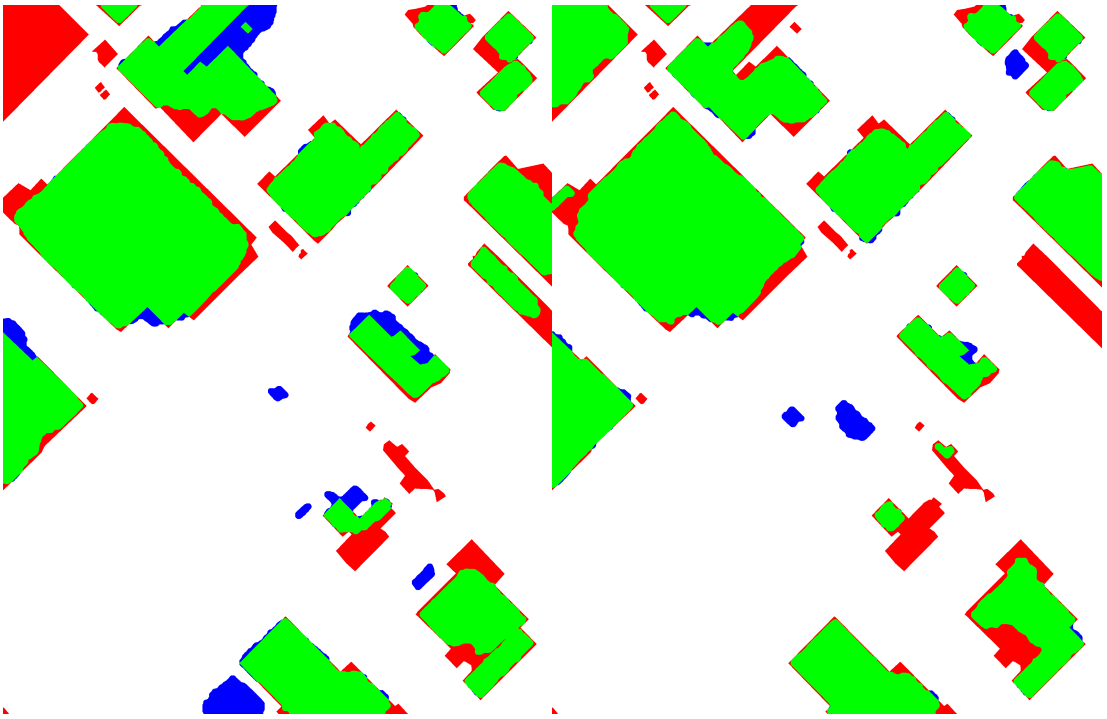
Table 4.1: Comparison of Mask R-CNN and GAN

Method	Precision	Recall	F1-Score	IoU	SSIM	Overall Accuracy
GAN	0.669	0.833	0.742	0.590	0.832	0.87
SAGAN+CRF	0.646	0.912	0.757	0.608	0.912	0.88
SAGAN+Method 1	0.628	0.908	0.743	0.590	0.911	0.88
Mask R-CNN+Method 1 ResNet-50	0.807	0.936	0.867	0.765	0.953	0.936
Mask R-CNN+Method 1 ResNet-101	0.823	0.958	0.885	0.794	0.913	0.945

The structures are very important because they provide information on the buildings. The following stages of post-processing improve the issues:

1 **Output based on Mask R-CNN with CRF**

For further refinement of the results, CRF is used as the first post-processing step. The following results show the output based on Mask R-CNN using two backbones (ResNet-50 and ResNet-101) and post-processing using CRF.



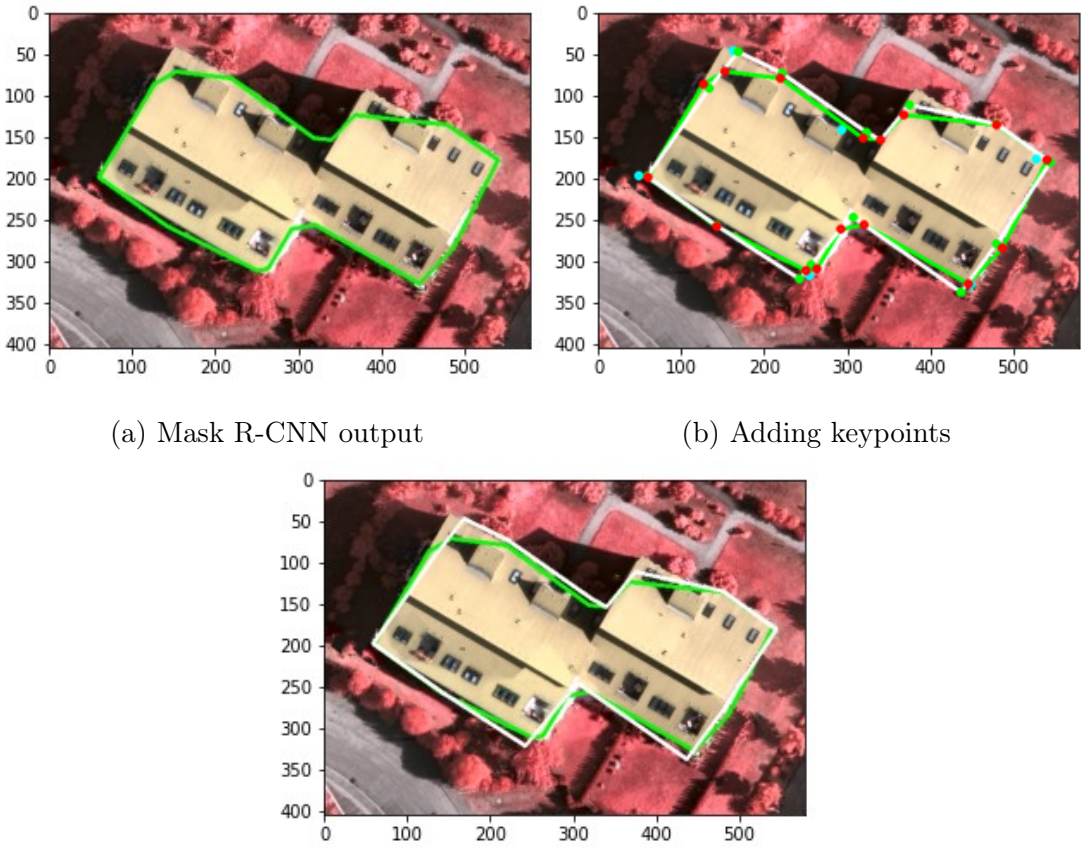
(a) Backbone using ResNet-50 (b) Backbone using ResNet-101

Figure 4.9: Outputs obtained from Mask R-CNN after application of CRF. Green color represents true positive, blue represents false positives and red represents false negatives.

After the application of CRF, there is some rigidity in the shape of the building. In addition, in some cases there is fragmentation of minute segments and the removal of these segments; as can be seen in Figure 4.9.

2 **Proposed Method 1 and Method 2**

Because the CRF could not deliver the structural rigidity required in the building, this research has developed two novel post-processing methods in two different stages. In the first stage, the algorithm (Method 1) snaps the segmentation contour into a proper geometrical shape. The output of this algorithm is shown in Figure 4.9-(a), and Figure 4.10.



(a) Mask R-CNN output

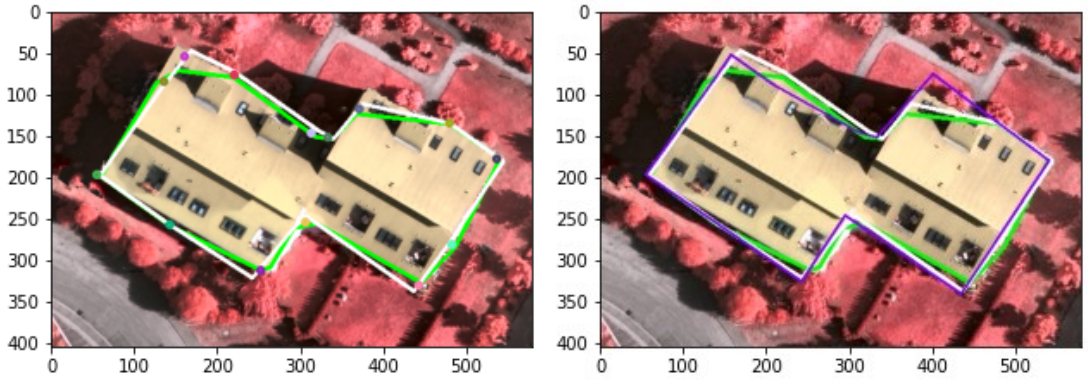
(b) Adding keypoints

(c) The modification compared with R-CNN output (a)

Figure 4.10: Outputs obtained from Mask R-CNN after application Method 1;where red points represent the points predicted by Mask R-CNN contour, green points represent points on fixed contours and light-blue represents the keypoints predicted by the neural network.

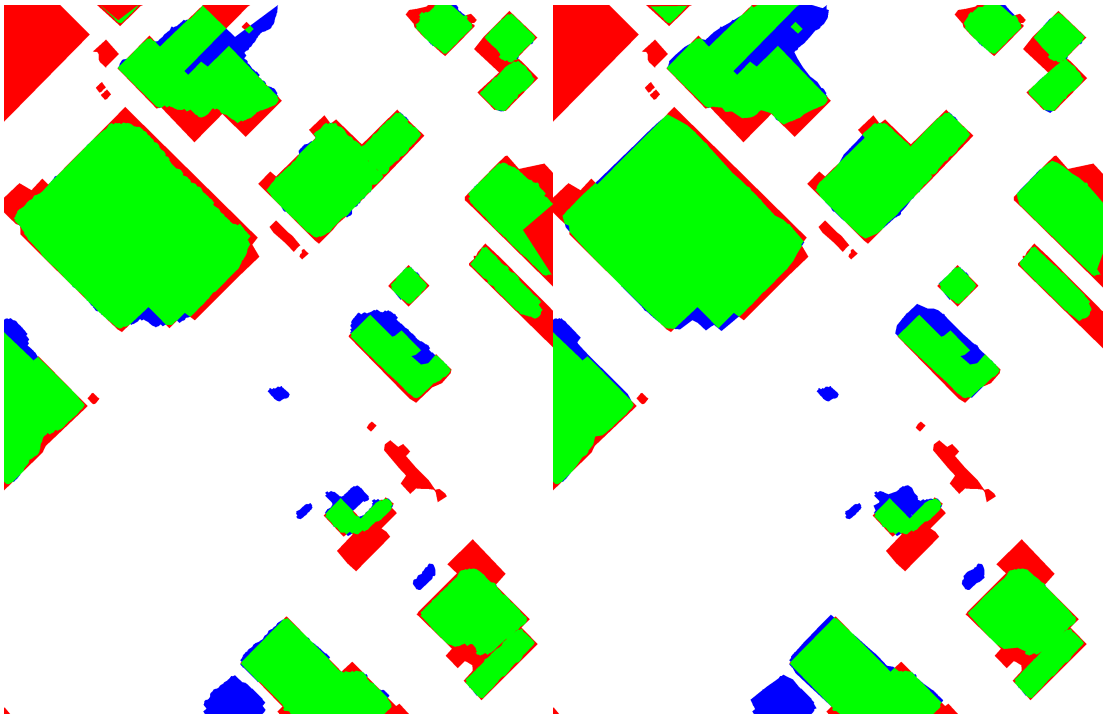
As can be seen in Figures 4.12, the coloured points represent different groups, where the red groups represent the intersection between ground truth and the predicted mask. In this way, the key-points correct those groups by selecting the voting of points, which either have the same orientation with respect to the azimuth direction, or are within the buffer of each threshold. All of these processes were completed in Methods 1 and 2,

and you can see in Figure 4.11 (b) how the purple contour represents the modified mask.



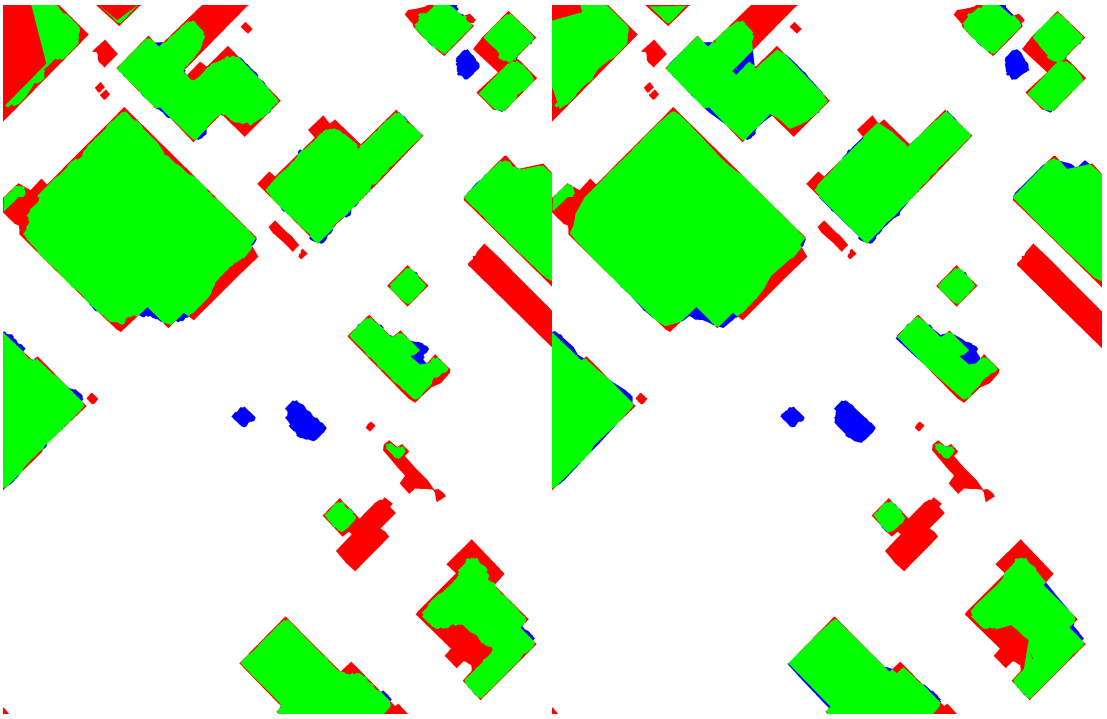
(a) Ground Truth points (b) purple contour represents reference lines

Figure 4.11: the second stage of post-processing;where the green contour represent by Mask R-CNN the withe contour ground truth, and purple contour represents the output obtained Method 2.



(a) Mask R-CNN + Method 1 (b) Mask R-CNN + Method 2

Figure 4.12: ResNet-50 backbone Mask R-CNN based, with application of CRF and proposed Methods 1 and 2.



(a) Mask R-CNN + Method 1 (b) Mask R-CNN + Method 2

Figure 4.13: ResNet-101 backbone Mask R-CNN based, with application of CRF and proposed Methods 1 and 2.

The second stage, which uses Method 2, proposed the final refinement by using snapping with keypoints to improve geometrical structure when compared to the normal snapping in Method 1, as can be seen in Figures (4.11-(b), 4.12-(b), and 4.13-(b)). The outputs from both methods are provided to show a side-by-side comparison to recognise the differences more easily. Additionally, the snapping algorithm Method 2 finds buildings' references, using the lines connecting the adjacent keypoints. As a result, the building orientations are determined; as illustrated in Figure 4.14.

Figure 4.14: Resultant output image based on Method 2 finding building references, where a yellow line represents one of the buildings' orientations.

As can be seen in the figure, the blue lines represent the orientation directions for each mask predicted individually. Here, the yellow lines are parallel to the objects, in this case buildings, which are facing the same direction and are in the same relative class; whereas other classes such as roads and green areas are either parallel or perpendicular.

4.4 Comparison of the Outputs

This section will compare examples from different areas and provide the resultant output images with Mask R-CNN and post-processing techniques. It also compares the resultant images of the proposed refinement technique with the GAN-based results achieved in the previous sections. In addition the results of two different AI backbones of model size and post-processing techniques are analysed to exhibit the most compelling aspects of the final output.

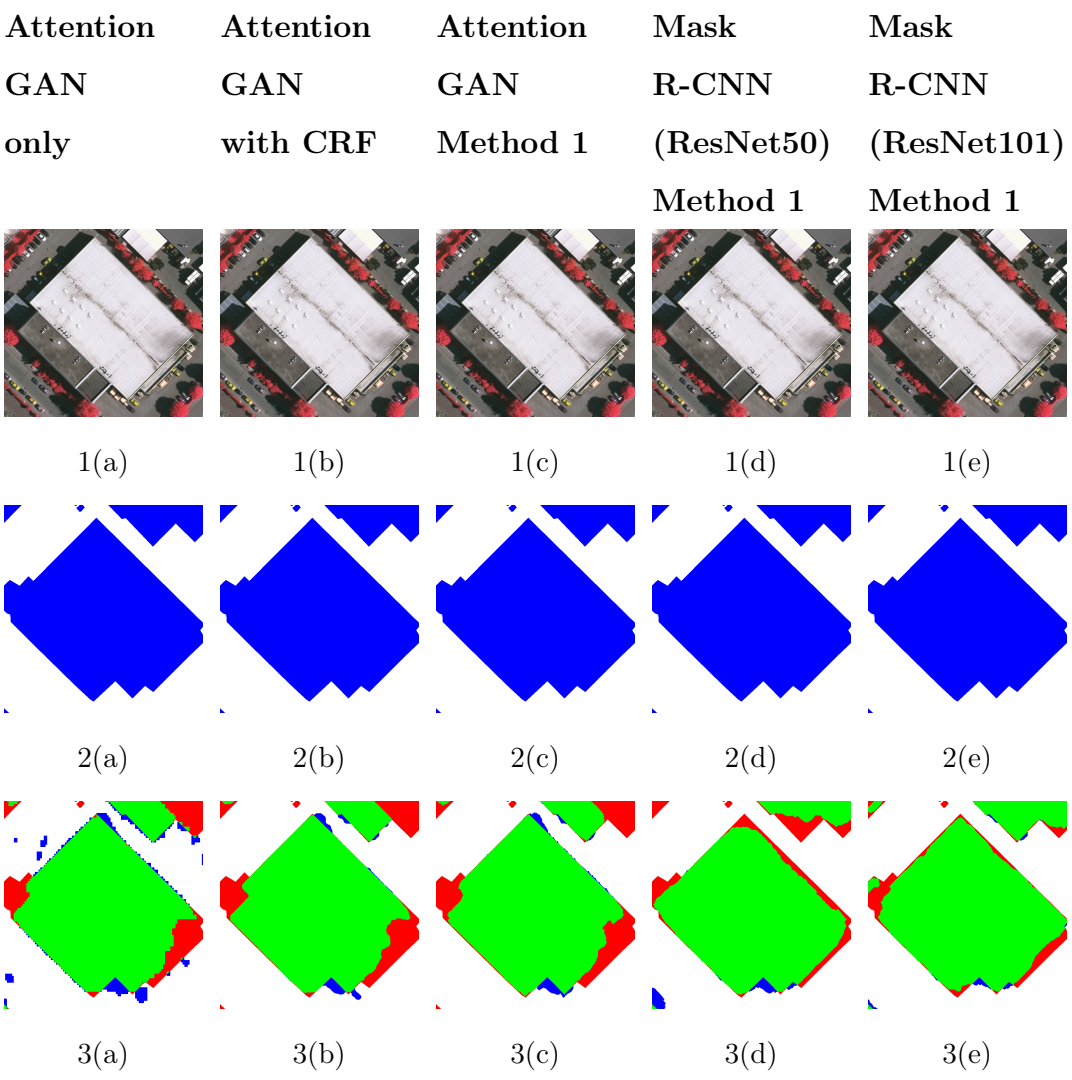


Figure 4.15: Comparison of an output from Attention GAN with Mask R-CNN.

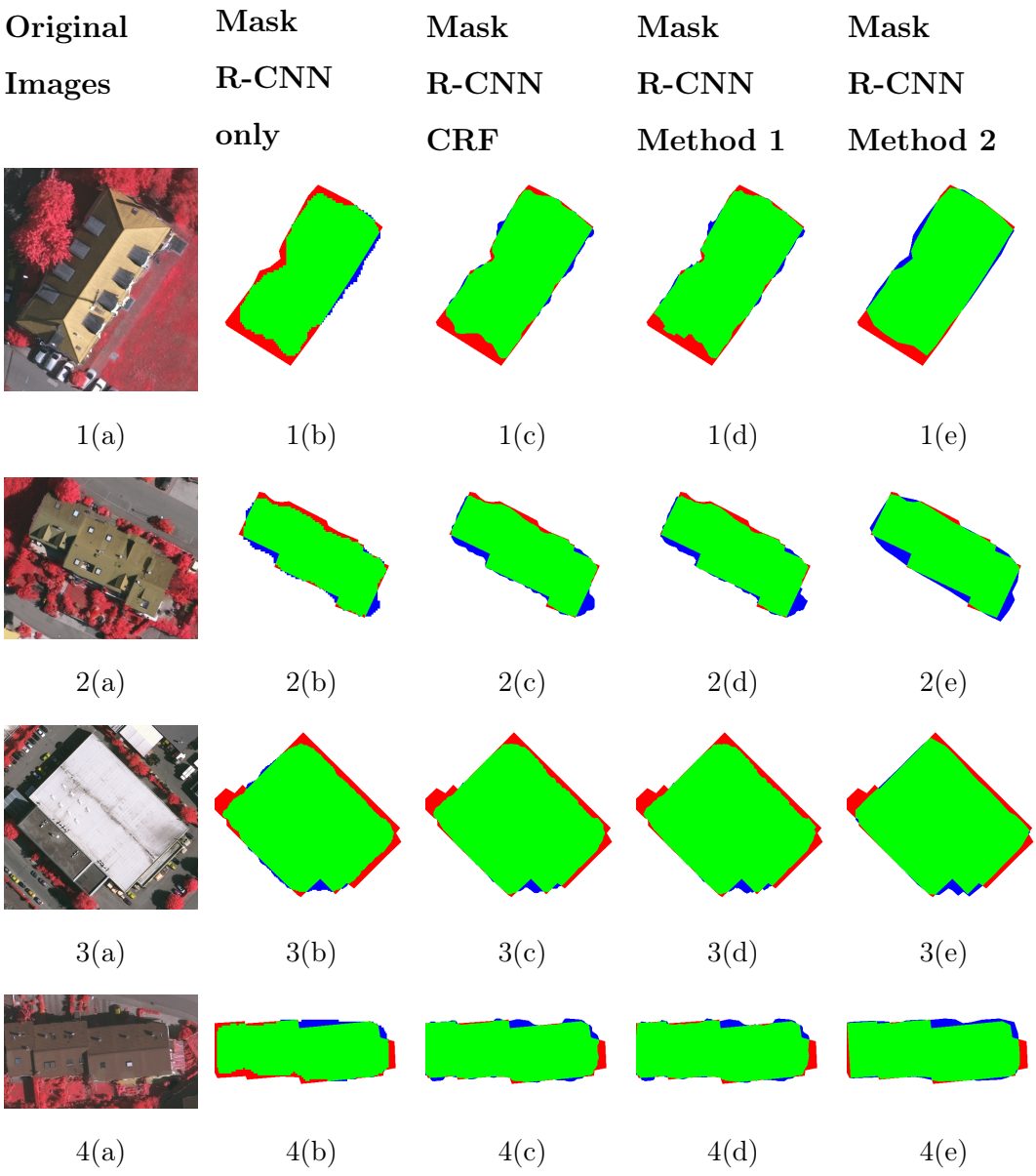


Figure 4.16: Results when using ResNet-50 as backbone. Green color represents a true positive, blue color represents false positive, and red color represents a false negative compared with the ground truth. Each building is selected from different areas.

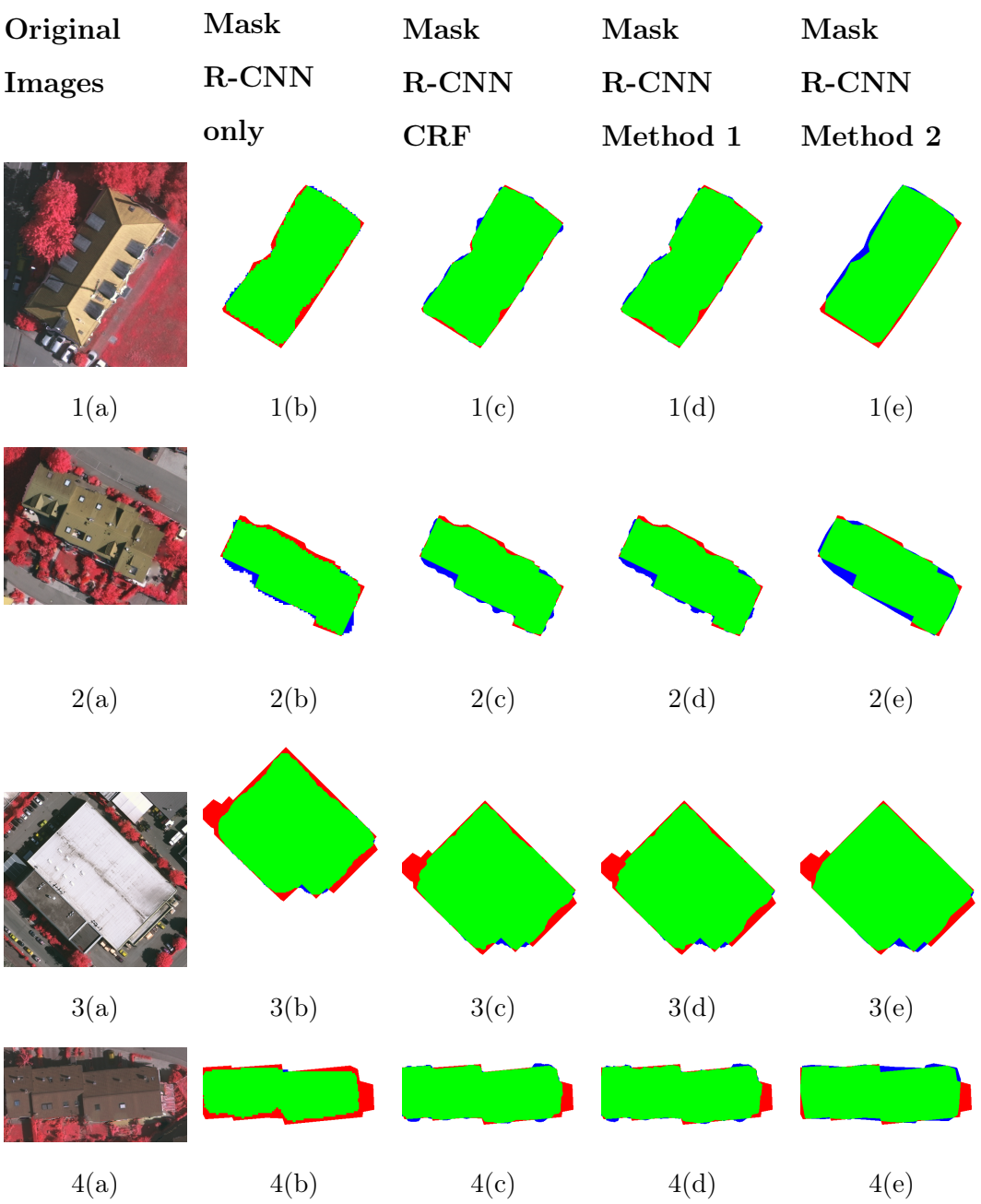


Figure 4.17: Results when using ResNet-101 as a backbone. Green colour represents a true positive, blue colour represents a false positive, and red color represents a false negative compared with the ground truth. Each building is selected from different areas.

As can be seen in the Figures 4.17-4(e) and Tables above, there is an improvement

from using deeper Resnet 101, compared to deep Resnet 50, in terms of the performance matrix used, especially IoU, precision and overall accuracy by 2.9%, 1.6%, 0.9% respectively.

4.5 Performance Metrics

This section presents the results from different areas but for a single building. The results are provided for a single building because this allows us to reflect on the gradual improvement of the segmentation mask and geometry of the building. The results are summarised in quantitative form in Table 4.2.

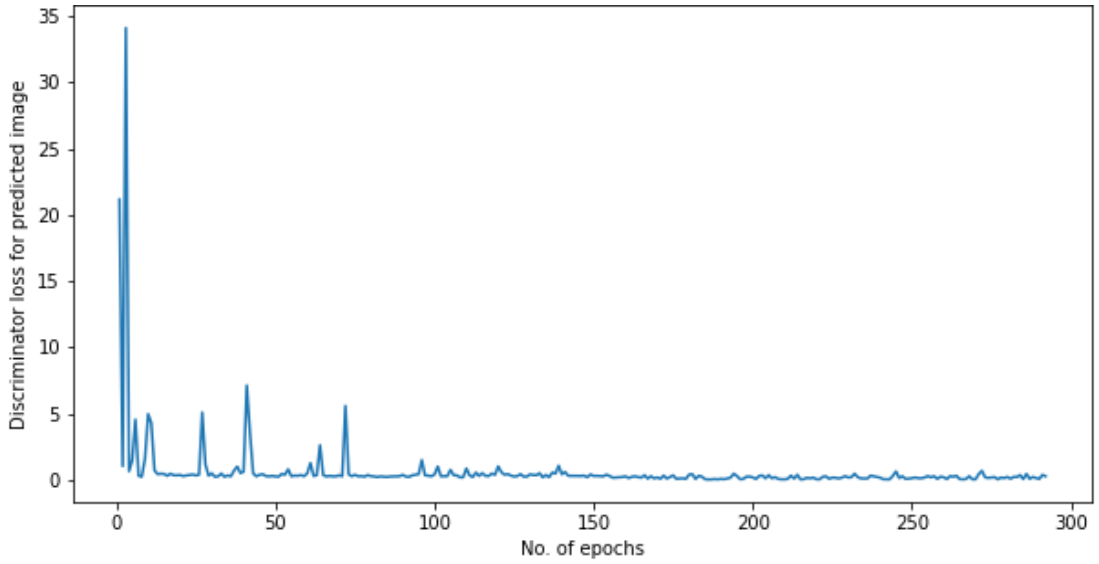
Table 4.2: Performance of the proposed method

Backbone Architecture	Method	Precision	Recall	F1 Score	IoU	Overall Accuracy
ResNet-50	Mask R-CNN	0.787	0.934	0.854	0.746	0.931
	Mask R-CNN+CRF	0.821	0.933	0.873	0.775	0.939
	Mask R-CNN+Method 1	0.807	0.936	0.867	0.765	0.936
	Mask R-CNN+Method 2	0.843	0.915	0.878	0.782	0.940
ResNet-101	Mask R-CNN	0.806	0.961	0.877	0.781	0.942
	Mask R-CNN+CRF	0.843	0.958	0.897	0.813	0.951
	Mask R-CNN+Method 1	0.823	0.958	0.885	0.794	0.945
	Mask R-CNN+Method 2	0.864	0.941	0.901	0.819	0.952

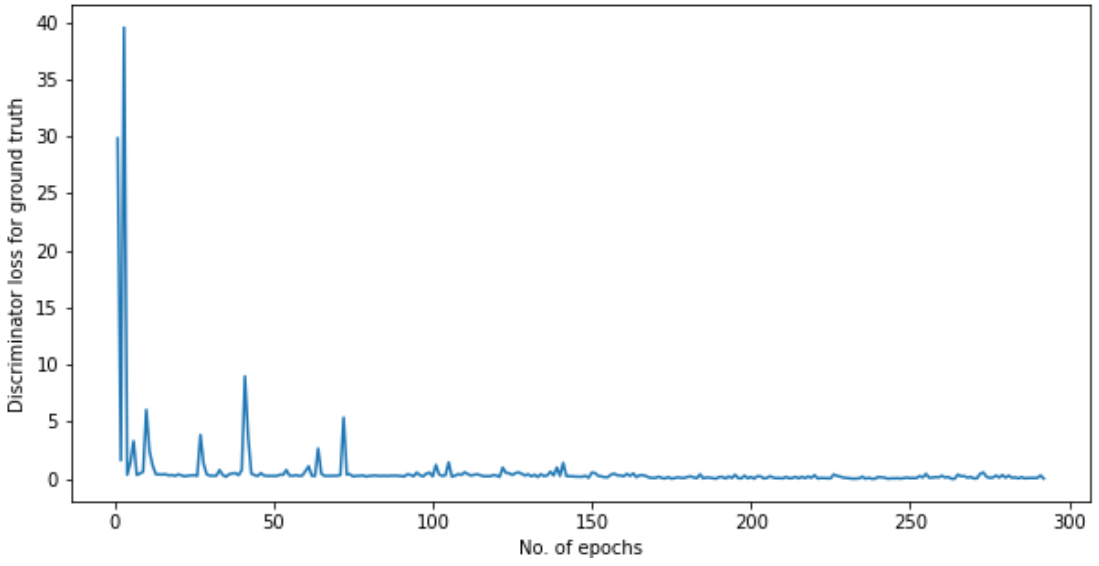
4.5.1 Training Details

Because this work involves training different neural networks, this section will provide the training loss of the neural networks for different results. Figures (4.18, 4.19 4.22 and 4.26) show the loss value as the training progresses.

Figures 4.18-(b) and 4.18-(a) show the discriminator loss, while Figures 4.19-(a) and 4.19-(b) show the loss for the generator over the course of 300 epochs. Each



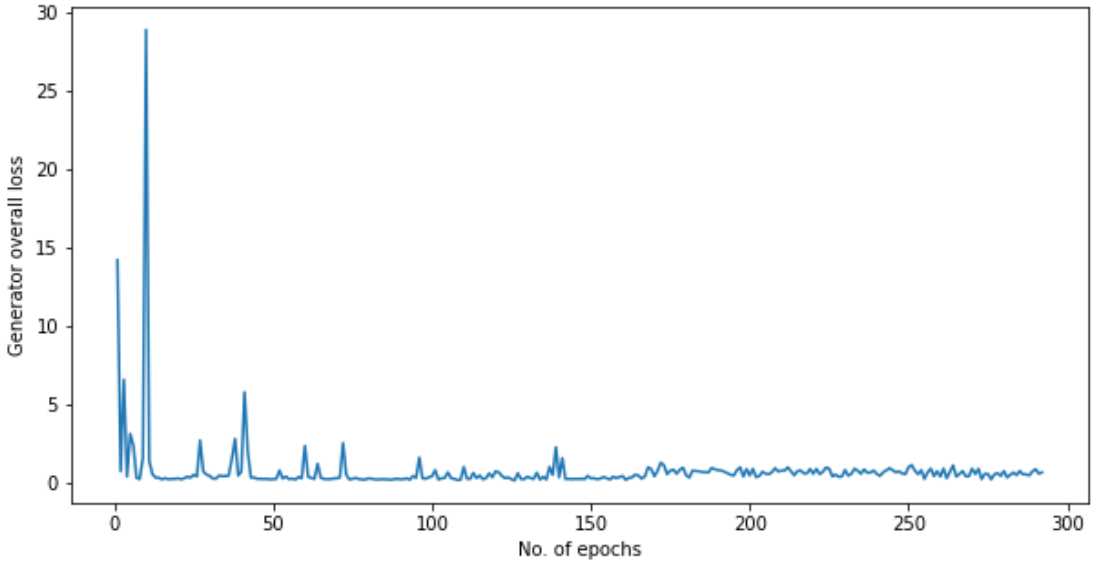
(a) Discriminator loss for (fake) image



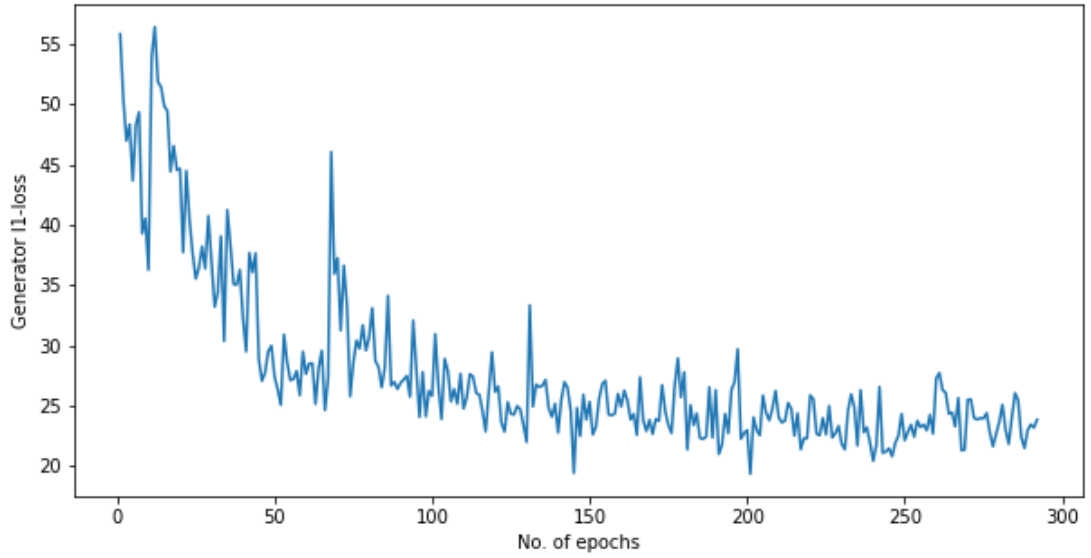
(b) Discriminator loss for (real) image

Figure 4.18: Discriminator loss function of GANs

epoch has 20 iterations to go through the entire dataset. The batch size for this training was set to 2. It is noticeable that the entire GAN has collapsed after 200 epochs. The only factor that is decreasing is the L1-Loss function, which happens because it is responsible for training the generator to refine the segmentation map.



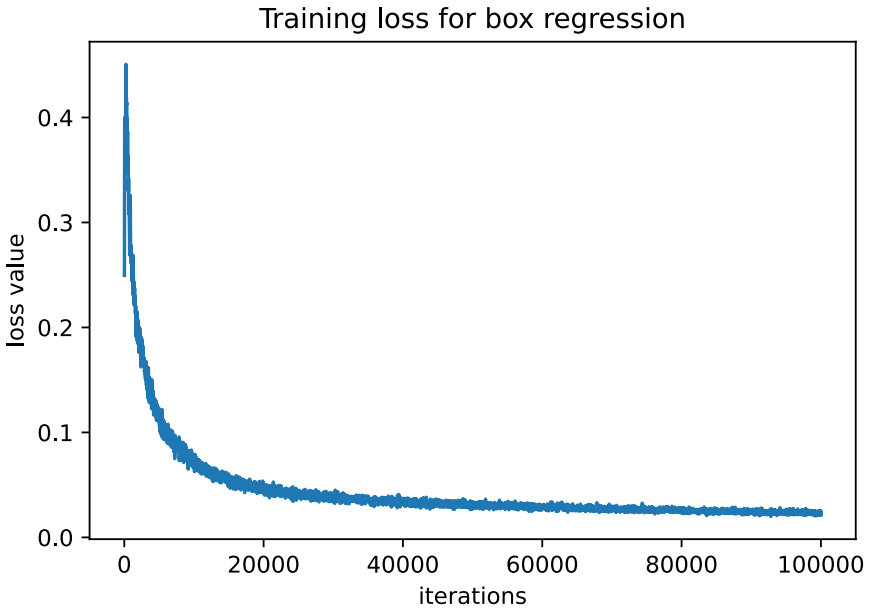
(a) Generator overall loss



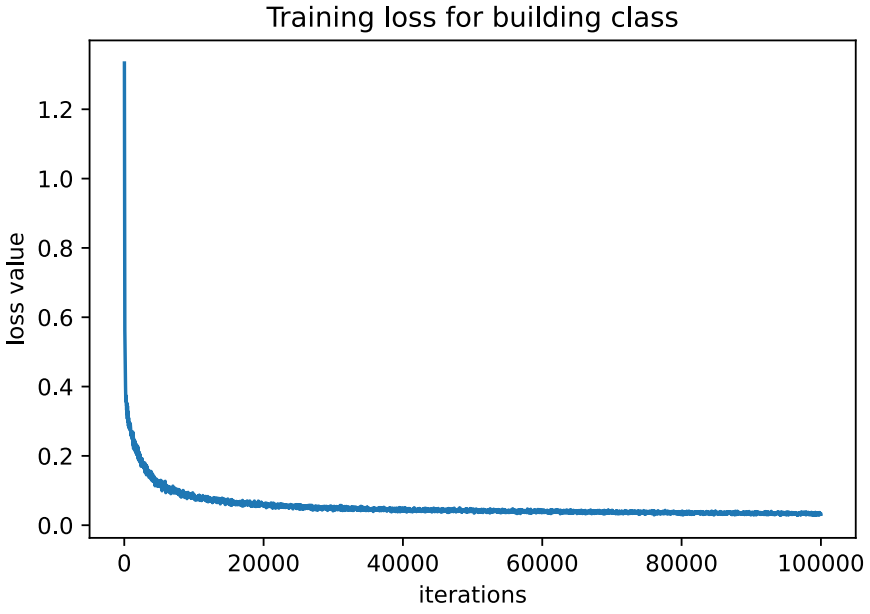
(b) Generator L1-loss

Figure 4.19: Generator loss function of GANs

As can be seen in Figures (4.22, 4.21 and 4.20), the loss value is for training losses and is used for classification, boundary prediction, region prediction, region location, key point prediction, and segmentation mask predictions. This figure summarises the training results for the Mask R-CNN with ResNet50 backbone. Here, it is possible to observe that the loss value is asymptotic to the x-axis as

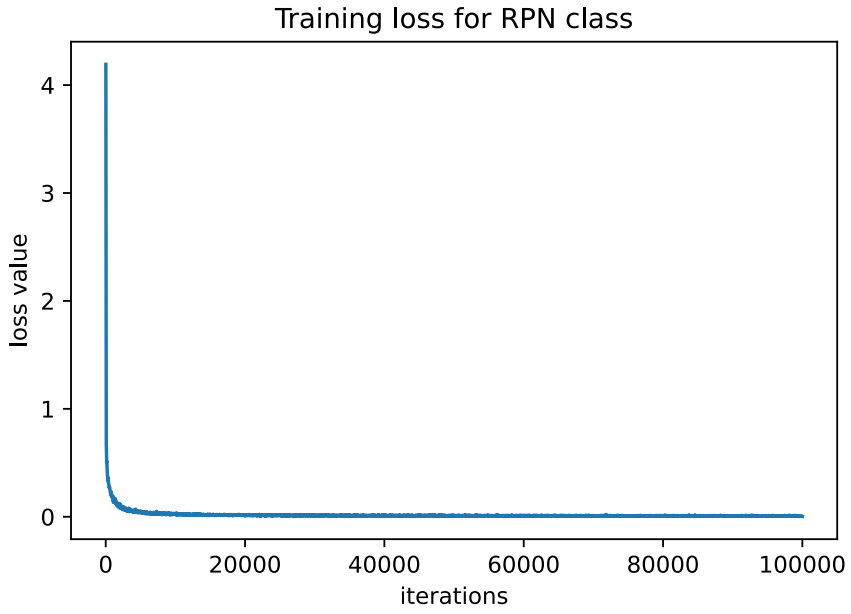


(a) Training loss for box regression

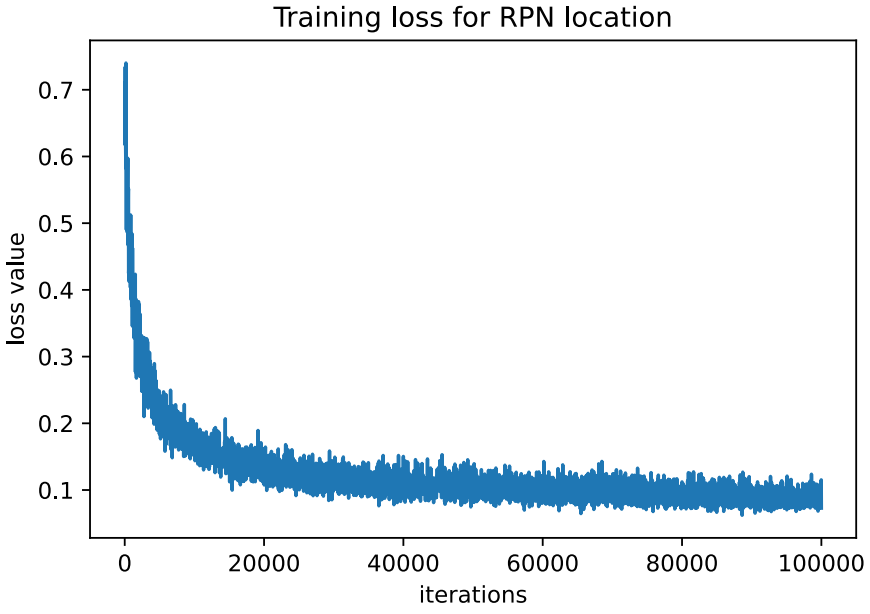


(b) Training loss for building class

Figure 4.20: Training Loss function for Mask R-CNN with ResNet-50 backbone.

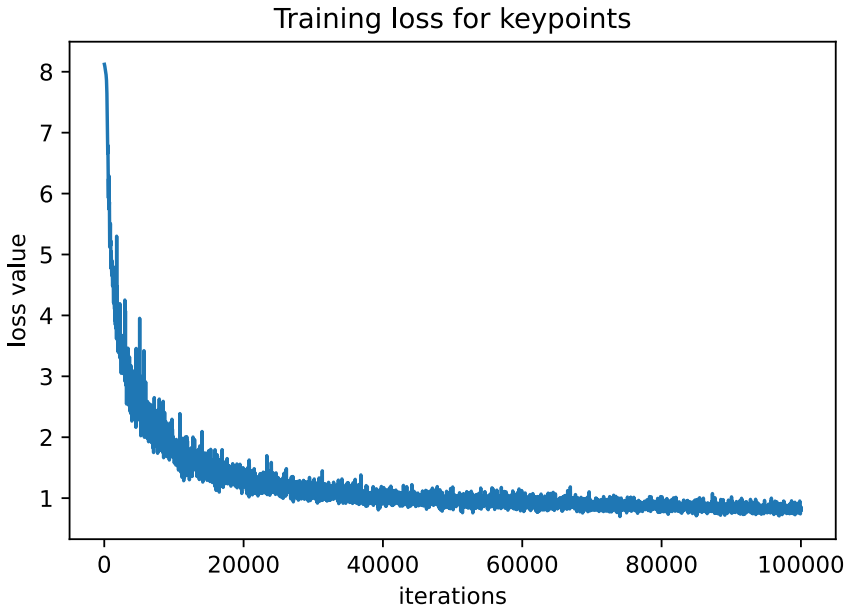


(a) Training loss for RPN class

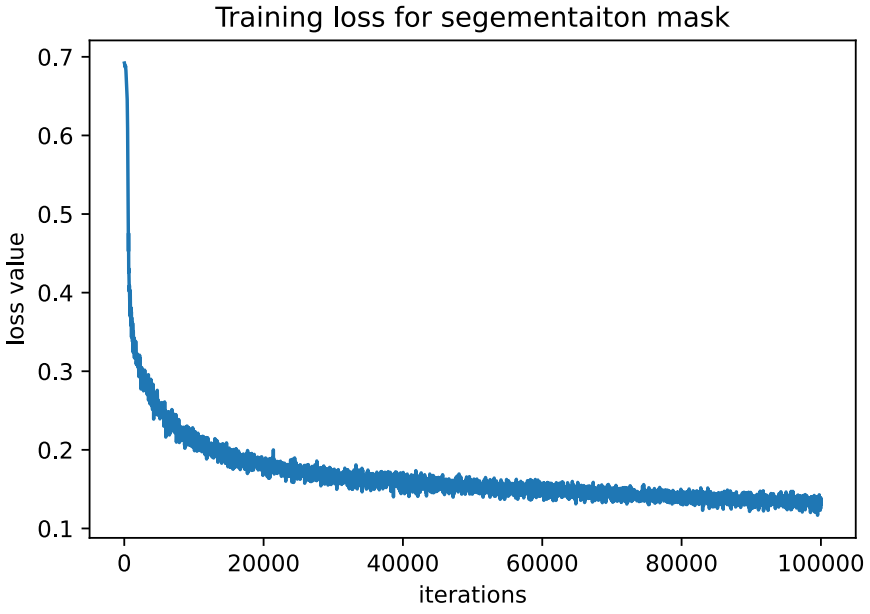


(b) Training loss for RPN location

Figure 4.21: RPN Loss function for Mask R-CNN with ResNet-50 backbone.



(a) Training loss for keypoints



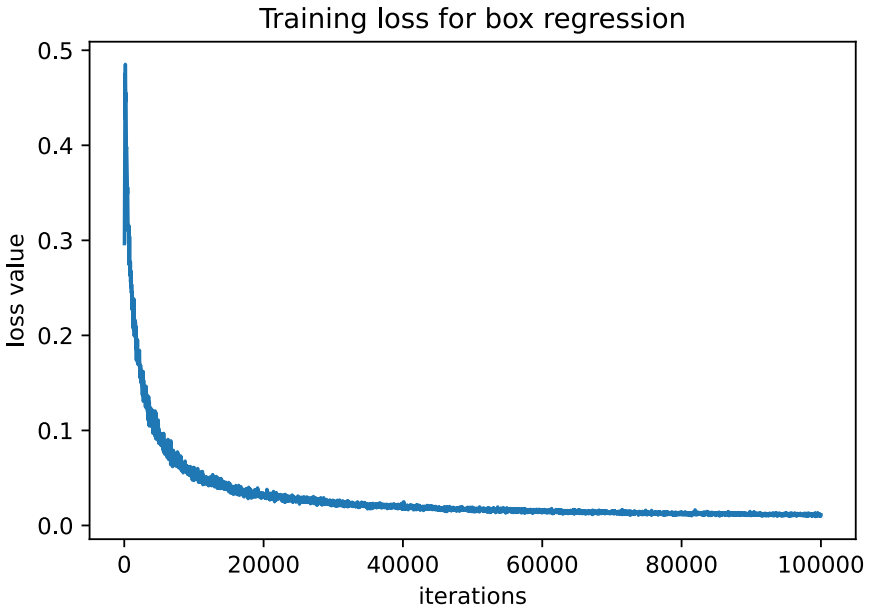
(b) Training loss for segmenation mask

Figure 4.22: keypoints Loss function for Mask R-CNN with ResNet-50 backbone.

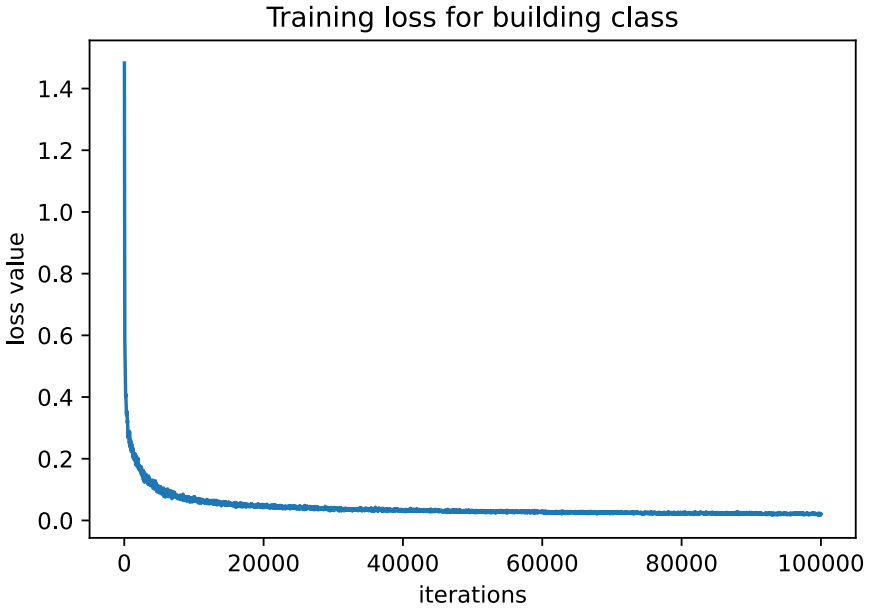
the training approaches $100,000$ iterations. Consequently, it was necessary to stop the training. In addition, according to Figure 4.22-(a), it can be seen that there are ups and downs in the graph. This happens because the network is not properly regularised, and the loss is swinging up and down. Different factors affect and produce such unstable training, and the learning rate is one such factor.

Classification, boundary prediction, region prediction, region location, key point prediction, and segmentation mask predictions all use the loss value to evaluate the accuracy of the model's prediction; as can be seen in Figures 4.22, 4.20 and 4.21. This figure summarises the training results for the Mask R-CNN with ResNet50 backbone. Here, it is possible to observe that as the training approaches $100,000$ iterations, the loss value is asymptotic to the x-axis. Consequently, it was necessary to stop the training. In addition, there are ups and downs in the graph in Figure 4.22-(a). This happens because the network is not properly regularised, and the loss is swinging up and down. The learning rate is one of the factors can affect and produce unstable training.

Figures 4.23, 4.24 and 4.25 shows a loss value similar to Figure 4.22 but for the Mask R-CNN with ResNet-101 backbone. Unlike the case of the ResNet-50 backbone, the deeper ResNet1010 is able to provide a stable training under the same parameters. This stability can be seen more clearly in Figure 4.26, which reflects the total loss for each network. The loss function for the deeper net is smoother, which means that the training is smoother. Consequently, it was possible to see stark differences during the testing stage.

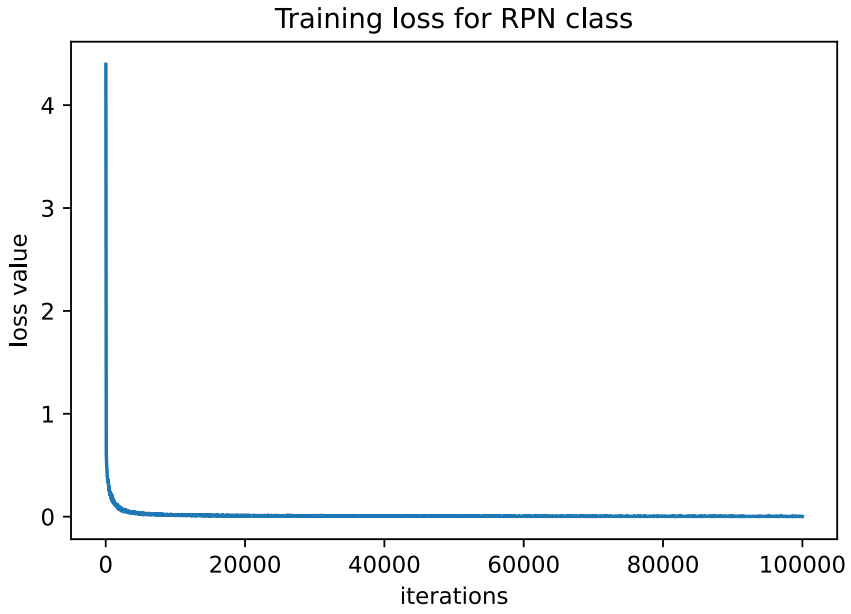


(a) Training loss for box regression

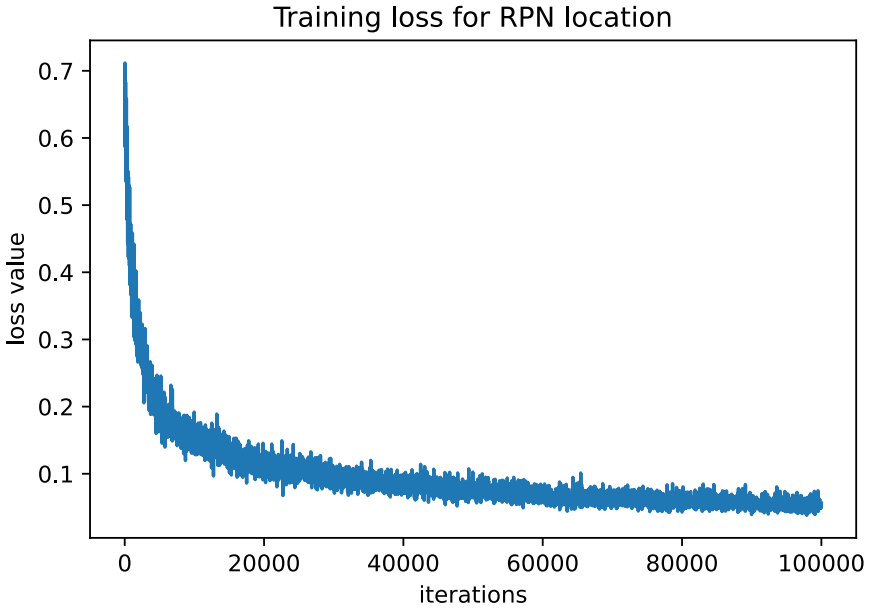


(b) Training loss for building class

Figure 4.23: Loss function for Mask R-CNN with ResNet-101 backbone.

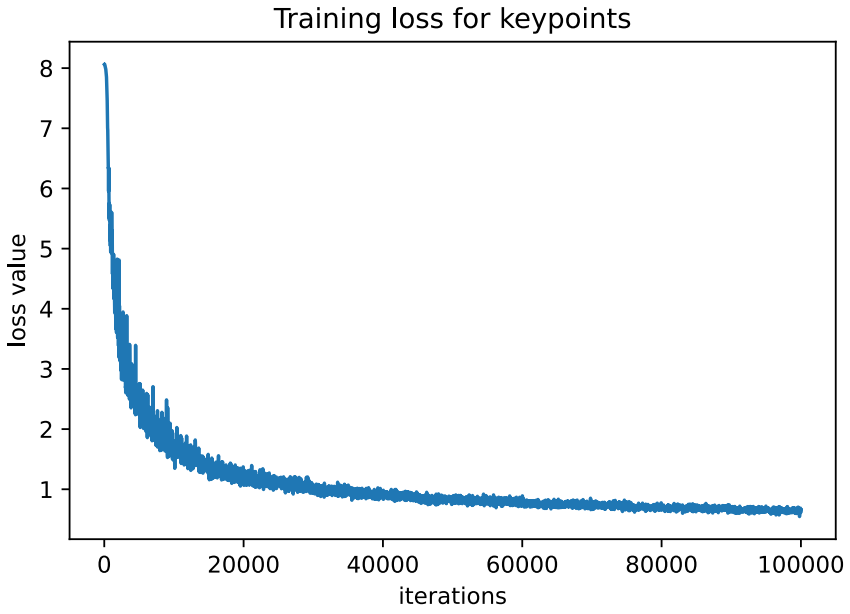


(a) Training loss for RPN class

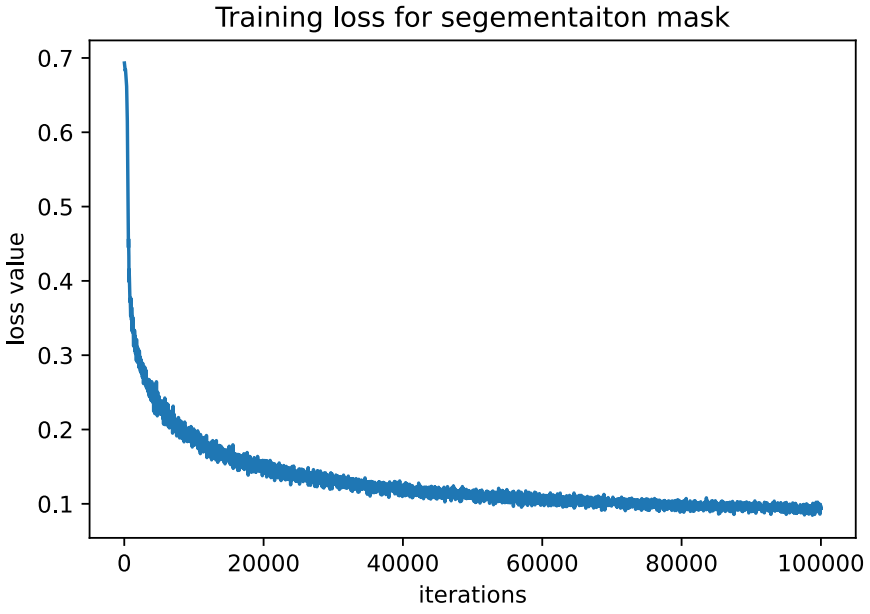


(b) Training loss for RPN location

Figure 4.24: RPN training Loss function for Mask R-CNN with ResNet-101 backbone.

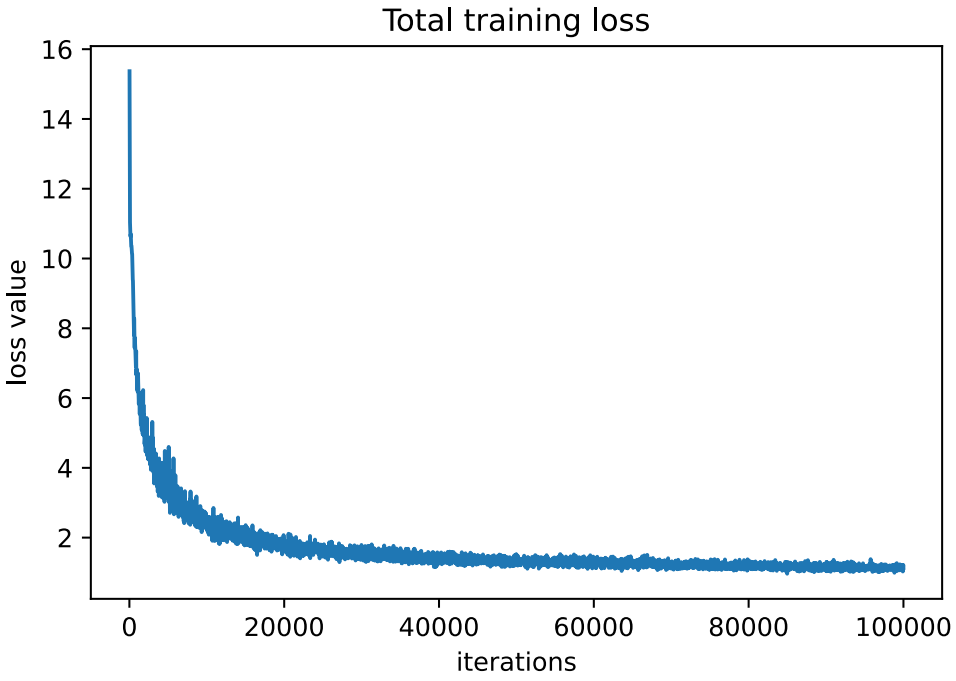


(a) Training loss for keypoints

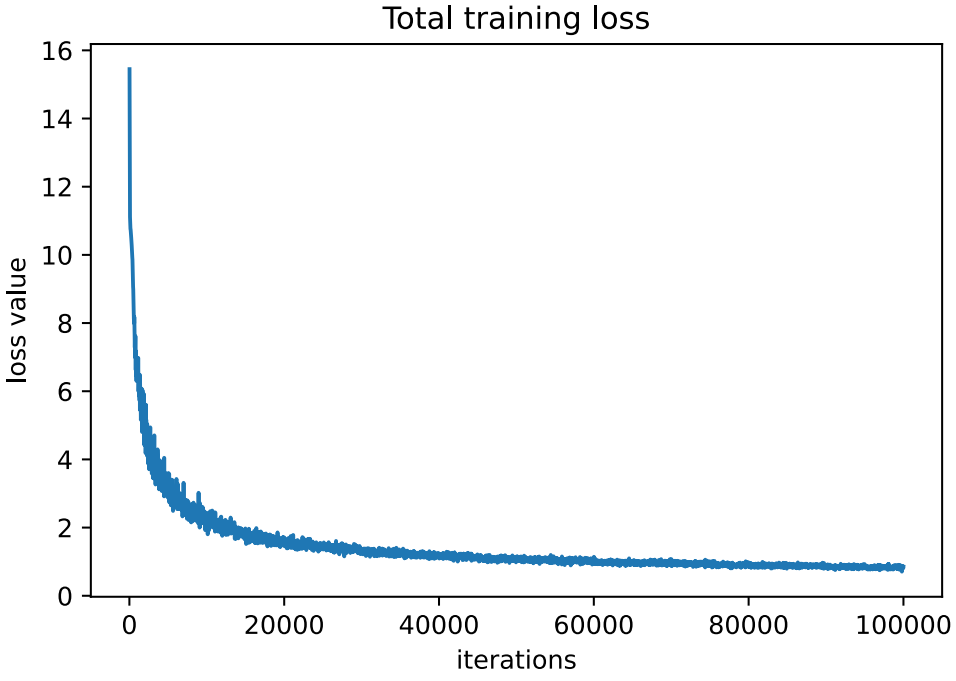


(b) Training loss for segmenation mask

Figure 4.25: keypoints & segmenation mask loss function for Mask R-CNN with ResNet-101 backbone.



(a) ResNet-50 backbone



(b) ResNet-101 backbone

Figure 4.26: Total training loss for Mask-RCNN

4.5.2 Computational Time

The most difficult aspect when using machine learning techniques is training times, because at this stage, the model attempts to regulate the neural weights if the P-trained model is used, the model regulates the weight; for example,using Image.net Models. However, the trained weight is dependent on different classes using CNN techniques. A new innovation in 2015 led to major organisations, such as Google and Facebook, producing Backbones, for example VGG16 and VGG19; Resnet34; Resnet50, and Resnet101, as shown in Figure 4.26 The main purpose for using backbones is to increase the output accuracy and decrease the computational time needed for training such big architectures. Practically, in this research, Resnet101, Resnet50 and Resnet34 were chosen because of the simplicity of their architecture, training time, and loss of function compared with other backbones. The performance of the network should be tested. As mentioned in the experimental setup, the testing set of this research work consists of four images, which are Areas (8, 13, 31, and 34). When running the entire post-processing algorithm on these images, the results were achieved within around 29 ± 1.34seconds, on average. The time taken is the average time on all four areas. This time is also the average from both of the networks. The post-processing time is not dependent on the network but is dependent on the buildings predicted by the network. Therefore, the larger model and smaller model have similar average times. When pre-processing the data to extract the keypoints for training, the total time taken for each image was 109 milliseconds on average. When training, the total training time for the model with ResNet-50 backbone was 29 hours and 10 minutes, while the total training time for the model with ResNet-101 backbone was 32 hours and 19 minutes.

4.6 Observations

This section presents the results of the observations as follows:

- Compared to the improvement of the results with the normal prediction from the Mask R-CNN-only network, the proposed method is able to increase the F1-score by 2.3%, precision by 2.1% IOU by 3.7% and overall accuracy by 1.2%. In every case, the proposed algorithm has a significant improvement on the precision, while trading off some values on recall. Consequently, this affected the value of F1-score.

- The effect of model size on the final output shows that the model's performance increased as the size of the model increased. This can be seen in Figure 4.17. The larger model is able to detect the important regions of the map and the building structures that are detected are on par with the original image. The accuracy of the model also improves, as can be seen in Table 4.2. The improved segmentation comes from the design of Mask R-CNN, which is able to propose the regions and extract the required mask from that particular region. Mask R-CNN is an architecture that fits into the idea of this problem. Therefore, most of the refinement comes from the further post-processing with the help of Mask R-CNN to address the issue of geometrical structures that are present within the building architectures. Geometrical structures are very important because they provide information about the building.

- CRF was used as one of the first post-processing steps for further refinement of the results. After the application of CRF, some rigidity in the shape of the building can be seen. In addition, the fragmentation of minute segments and in some cases the removal of these segments can be observed, as can be seen in Figure 4.9.

- The effects of post-processing on the final output show that the developed novel snapping algorithm is able to snap the segmentation contour into a proper geometrical shape. The outputs from both of the proposed methods are shown in Figures (4.12, and 4.13) to compare them side by side. Method 2, snapping with keypoints, can get a better geometrical structure compared to the snapping in Method 1. These results provide better structural details than a CRF because it was unable to deliver the required structural rigidity in the building. This is clearly visible in Figure 4.12.

- The algorithm performs remarkably well when the reference lines detected support the in the orientation of the building which is clearly visible shown in Figure 4.11-(b).

Table 4.1 summarises the results when comparing Mask R-CNN with SAGAN. A drastic improvement can be seen with Mask R-CNN because of stable training, robust architecture and generally well-defined loss functions. This table also provides a reason to develop the Method 2 because it clearly demonstrates that the proposed method is clearly not sufficient because of the CRF. This limitation was imposed because the proposed Method 1 was unable to exploit the geometry present within the building. Note that the table does not show the results from method 2 because GAN and SAGAN do not provide any resultant keypoints.

Chapter 5

Discussion

This chapter discusses the results of the research and the impact of the techniques applied to the machine learning approaches used. As a reminder, the approaches are briefly presented, along with a discussion of the results for each technique and the deep learning approaches utilised. An attempt has been made in this thesis to explain how different building images have been tested, and the success of the post-processing techniques that have been used. The case study research has been matched with a grounded theory research approach through the literature review on archived and open source materials accessed online. In addition, descriptive methodologies have been used to document the potential of, and difficulties in, using AI, cloud computing, ML, and DL with remote sensing, by developing nations. The section on the techniques used has presented the main types of methods used, followed by comparing and distinguishing between contemporary state-of-the-art methods. Furthermore, quantitative and qualitative research methodologies have been explored to examine their applicability to the current research and, therefore, other similar research. It has been shown that there are benefits from using post-processing algorithms during and after training ML models. Furthermore, the limitations of the research, including the ML data collection and an analysis of the framework's limitations, are set out. Finally, a brief recap demonstrating the rationale for the particular research approach chosen is provided.

5.1 Interpretation of the Results

In this study, the algorithms proposed have been successfully tested on hundreds of building images, yet to show accurate results, four images are reported in Figures 4.17 and 4.16. The AI architecture and post-processing techniques proposed can accurately detect the building boundaries, as can be seen in Figures 4.16 and 4.17.

These figures show the results for four different buildings that have bushes covering the roof boundaries; different roof geometries; multiple edges, length wise and width wise; regular changes, and so on. As can be seen in Figure 4.17-1(e) and Figure 4.17-2(e), the buildings' boundaries have been clearly identified and the geometric shapes have been refined. The algorithm worked perfectly, even when the images had different contrast distributions such that one side of the building was light while the other side was in shade as shown in the Figure 4.16.

The most useful aspect of the algorithms proposed is that they can provide accurate and useful information about the orientation and direction of building elements as part of the process. This helps to find the orientation of the building, based on which, more semantic information can be extracted according to the North azimuth direction obtained by `FixContour()` algorithm Method 2. For example, referring to Figure 4.17-4(e), the algorithm provides the major or principal direction from left to right, using one of these can divide the building into upper and lower parts, and can trace the upper and lower boundaries, find the two major building sectional changes, correlate them and can then provide very useful semantic information that the building has three sections. Further semantic information can also be derived. For example, the building's left-hand sectional part is smaller compared to the right-hand sectional part. Figure 4.16 reports the results for all four buildings using the same architecture, with ResNet-50 being used along with the proposed post-processing techniques.

5.1.1 Major Findings of this Research

The findings from the current study confirm that 2D city models at LOD1 can be generated by utilising VHR satellite images. Furthermore, employing RGB data from VHR satellite images to assist in the automated detection of urban areas improves the segmentation of buildings within their surroundings in a single image space, which shows that this data could be beneficial for the automated detection of buildings with various geometries. Reliable output results were achieved when applying the algorithm to extract footprints. This demonstrates that the algorithm developed can create accurate 2D models from single-satellite imagery and can achieve up to 95% accuracy.

The algorithms act on challenges such as the complexity of large cities, where the decision can be obtained more quickly. Furthermore, 2D building models can be predicted and reconstructed from VHR satellite imagery by considering the complex urban topography; the characteristics of the target objects, and the similarities between the objects regions and their backgrounds in the same image, as well as the illumination conditions and imaging time. Additionally, the developed algorithms can sharpen the geometry boundaries with the help of deep learning techniques, which is a significant contribution, by producing reference points and specifying the buildings' orientation.The orientation of the building's surrounding directions in large regions can be defined by azimuth through simply following the lines of the algorithm's code. In terms of sustainability, the orientation associated with massing, could be critical in providing a building with passive visual comfort. Orientation and overall design must be addressed together at an early design stage, since neither can be fully maximised without the other. A suitable orientation drives the building to reduce power loads and maximise free solar energy and wind for new infrastructure development. Furthermore, determining the best orientation could even support other environmental conditions, such as existing districts or remote countryside areas, as it can be used to

assess the suitability and potential of sites for solar panels; for example, on the rooftops of specific buildings based on their orientation and the area calculated as being most exposed to sunlight. Doing so should reduce the cost of services and energy expenses for nearby populations, which is especially beneficial for economically challenged towns and cities. The main approaches can be used to refine depth map predictions CRFs. However, several previous works have used CRFs' post-processing potential to refine depth image predictions. These works define pixel-wise and pair-wise loss in terms of pixels and their neighbours using an intermediate predicted guidance signal, such as geometric features, ground truth, and keypoints. An initially predicted depth map is then refined by inference with the CRF. While post-processing algorithms help enhance the initial depth predictions and produce qualitatively better results. Moreover, another potential advantage of using the proposed post-processing algorithms for depth refinement is to use them as image enhancement methods. There are a lot of different purposes that post-processing algorithms could be used for in various fields, such as for medical purposes, converting raster to vector data, or different types of clustering in multiple formats; also, as denoising methods that maintain the correct image contours, and for different mapping multiplication processes. In addition, post-processing algorithms can take advantage of the keypoints obtained by the ML models, and the simplicity of vector data multiplication, allowing fully differential and much faster computation processes.

5.2 Discussion and Explanation of the Results

Comparison with Contemporary State-of-the-Art Methods

The proposed algorithm has been compared with state-of-the-art methods, and the results are provided in Table 5.2. This comparison is based on the Vaihigen dataset for building class. As can be seen, the overall accuracy of the proposed

method is approximately 1% higher compared to another previously proposed method [6], which is based on a contextual graph model that preserves contours. However, the algorithm that is proposed in this paper provides accurate contour points, along with directional information, as compared to a previously recorded method [6]. The technique that is proposed in this paper achieves a similar recall percentage to another paper [7]; however, the architectures and computational time of SEG.H-sc1, HED.H + SEG.H-sc1 are more complex. In fact, there is no direct comparison with the proposed method and the method of[7] because the method in[7] uses DEM along with VHR images for training and testing. One research study [292] used a combination of FCN and Mask-R-CNN; however, their methods provide approximately 3% and 1% better results in the precision and F1-score, respectively. Recent work [291] provides better precision and an F1-score almost equal to the proposed method. Even so, it provides lower recall results when compared to the method that is proposed here. The method proposed in[291] uses different sizes of patches during the first stage of semantic-free segmentation MRS. Achieving a proper patch size is a crucial task for this method. Consequently, the authors used Segmentation Scale Parameter (SSP) in the MRS. The research work tested various SSP parameters on Vaihingen and Potsdam datasets, and found experimental values of 20 to 50 for Vaihingen images, and 40 to 100 for the Potsdam images. Furthermore, the algorithm that is proposed in this thesis has been compared with the method proposed previously [292] based on EaNet. The F1-score achieved by EaNet is approximately 6% higher than the proposed method; however, the proposed technique achieves a similar IoU percentage. The ResNet101 backbone was used for both EaNet and the proposed methods. While the EaNet utilised 40K iterations on the Vaihingen dataset, the proposed technique used 100k. The proposed method took 32 hours and 19 minutes of training time, and provided results or inference in just 29 seconds. It should be mentioned that the previous matrix, with regard to evaluating the accuracy of the techniques, has facilitated a comparison of ground truth with

the predicted mask to see how accurate (or not) a model is regarding matching shape and orientation.

Table 5.1: Comparison of different an output progress

Method	Precision progress	Recall progress	F1-progress	IoU-progress	Overall-progress
Mask R-CNN (Deeper backbone progress)	1.90%	2.70%	2.30%	3.50%	1.10%
Mask R-CNN+CRF	2.20%	2.50%	2.40%	3.80%	1.20%
Mask R-CNN+Method 1	1.60%	2.20%	1.80%	2.90%	0.90%
Mask R-CNN+Method 2	2.10%	2.60%	2.50%	3.7%	1.20%

Table 5.2: Quantitative comparison results on Vaihingen validation datasets (units: %).

Authors	Year	Methods	Precision	Recall	F1 Score	IOU	Overall Accuracy
Zhao et. al [6]	2017	Non-CRF Model	-	-	-	-	92.96
		CRF Model	-	-	-	-	94.54
Marmanis et. al [7]	2018	SEG·H-sc1	-	93.8	-	-	-
		HED·H+SEG·H-sc1	-	95.6	-	-	-
Zhou et. al [325]	2019	FCN	84.3	94.7	89.2	-	-
		Mask R-CNN-L	83.3	97.9	90.1	-	-
Zhen et. al [292]	2020	EaNet (ResNet101)	-	-	96.4	81.7	-
Zhang et. al [291]	2020	MLCG-OCNN	94.7	86.3	90.3	-	-
Proposed Method 1 (ResNet101)	2020	Mask-RCNN CRF+Method 1	82.3	95.8	88.5	79.4	94.5
Proposed Method 2 (ResNet101)	2020	Mask-RCNN CRF+Method 2	86.5	94.0	90.1	81.19	95.2

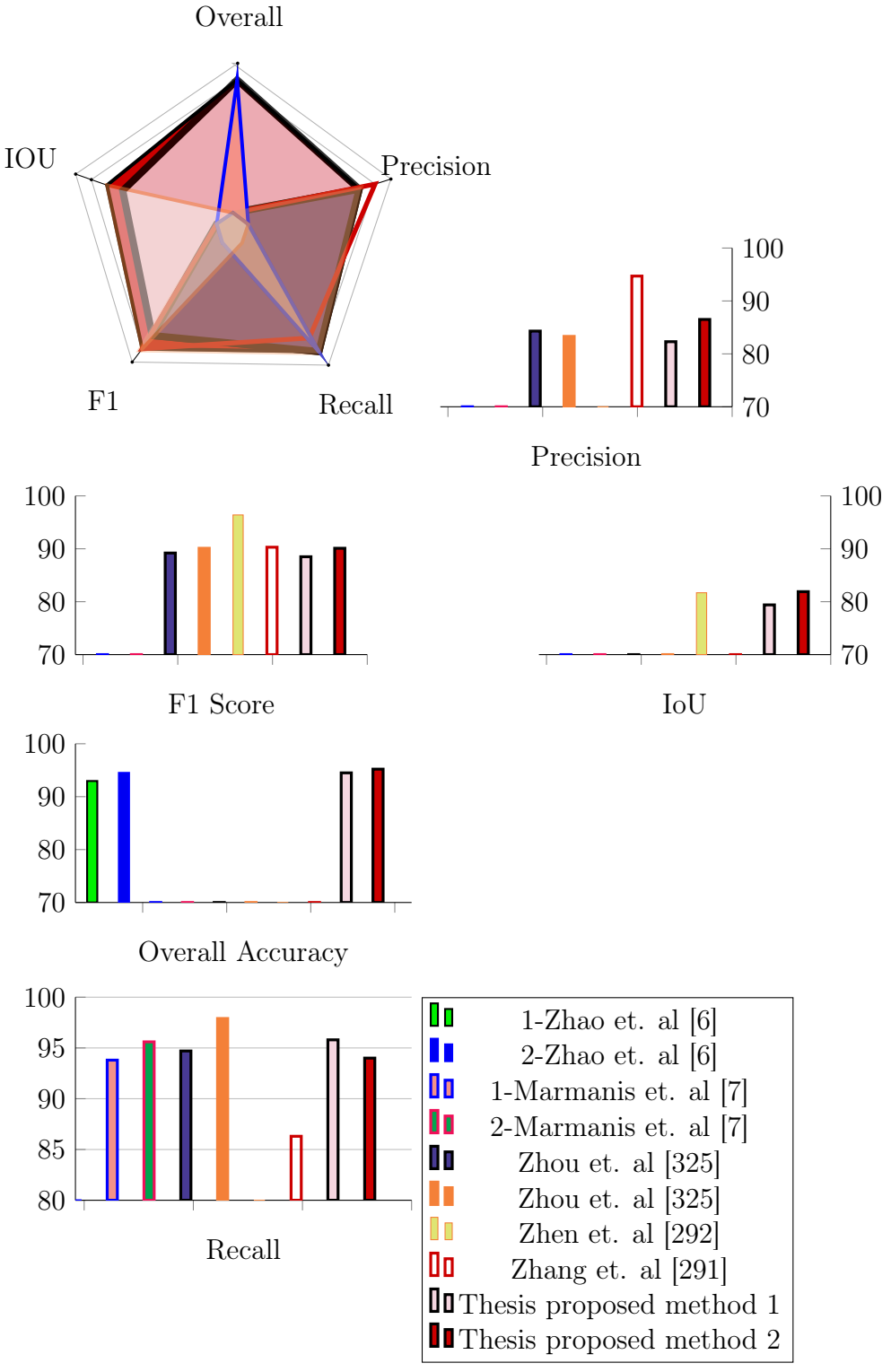


Figure 5.1: Spiderweb and charts diagrams are shown 5 Dimensions of performance metrics, quantitative comparison the results, with state-of-the-art Methods..

5.3 Implications and Some Practical Applications

This study's findings have revealed several advances in the field of urban modelling, and the discoveries herein should assist in advancing the application of remote sensing data obtained from photogrammetric satellite images, aerial photography, and topographical frameworks. The results from the automated extraction of 2D building models shows that the newly developed method may be used to clarify the geometry of urban objects, especially the characteristics of buildings. Some examples of these applications are as follows:

- **Structural analysis of the building:**

  A statement of the importance of the output results, the analysis of the findings validates the research purposes, hypotheses in the literature concerning the reconstruction of 2D building models. The results of the newly developed approach have revealed some critical points that can assist in developing an appropriate methodology with regard to the strengths, outcomes, and deficiencies of this research in comparison to previous studies. According to the literature, the findings confirm that satellite imagery and remote sensing technology can be used in urban planning. The advantages of these kinds of data include higher accuracy and precision in object measurements. In addition, various details can be discerned that are either unobtainable or may even be of limited effectiveness with other data sources. This happens because rectification of VHR satellite images is consistent in different spectrum bands, which allows for the extraction of the actual distances of the features to be measured for the conception of their respective 2D and or 3D building models. Furthermore, the high spatial resolution makes it easier to understand and identify the shape of the building's footprint, which is essential in detecting 2D to reconstruct 3D models of urban spaces on a large scale using an unsupervised method with AI assistance.

- **Orientation detection:**

  Using an unsupervised method to detect objects can be helpful in automated analysis because this method can cover large areas and include many types of data sources, such as forecasting and cloud data, to provide a significant portion of an image. Furthermore, the results from determining the building's orientation can be used to assess the potential for solar energy and how it can benefit buildings by determining the most suitable areas available on individual rooftops, as well as the suitable positions that can significantly influence the amount of solar energy available for energy performance and consumption. According to Peng and Lu [191], the first step in assessing a rooftop's PV potential is to determine the available rooftop area to install a PV system. The extracted 2D geometry of a building is used to calculate the percentage of the gross outside wall with gross floor area, which can help to determine the building's energy performance by forecasting the optimal aspect ratio between them. This will consequently reduce the requirement for heating systems by calculating the buildings that receive more heat radiation in the winter and shadows in the summer. The results can then be used to predict the impact caused by building structure, shape, and orientation, which is helpful for urban planning in new construction projects. Given that this is an unsupervised method, the most remarkable advantage of the findings is that this method can track a building's geometry to create accurate 2D building geometry to produce 3D models without any additional data, and at low cost and time, when compared to traditional methods.

5.4 Limitations of This Research

- **Data for training:**

  Many of the deep learning techniques (e.g., GAN and Mask R-CNN) that have been used in this research need a large number of images to achieve a trained model often many times the number of images required to achieve better accuracies, in terms of thousands of images. Uploading terra bytes of images for processing on a cloud-based GPU platform requires a considerable amount of computer resources, which are expensive. Therefore, this research has been limited by the amount of computational power available.

- **Computational power:**

  Machine learning requires a powerful GPU (e.g., this thesis utilised 12 GPUs). For large cities, and for high resolution images, a very large number of GPUs would be required. Although GPUs are available on cloud computing services (e.g., Amazon, Google, Microsoft), the cost of using these GPUs is very high. Therefore, this research has been limited by the amount of computational power that was required.

- **Training time:**

  Some of the neural network models, particularly those using combined neural networks, also need a significant amount of training time because terra bytes of high resolution satellite images need to be processed.Therefore, this research has been limited by the amount of training time that was required.

- **Backbone architectures of the models required:**

  A large size of backbones are available (e.g., ResNet-152, ResNet-164 or ResNet-1202). However, the limitation of using a large number of backbones is that they need a great amount of computational power and training time. In addition, increasing the number of backbones only leads to a small increase in accuracy, especially with small datasets. Therefore, this research

has been limited by the amount of computational power and training time required by the backbones.

- **Experimental parameters setup:**
  The majority of ML algorithms include user-defined parameters that might influence classification accuracy[344]. As an example, the activation function, the number of hidden layers, the number of nodes in each layer, the learning rate, momentum factor, weight initialisation, the number of iterations, and so on. Although the default settings for these parameters are often provided, verification to find their optimal values is required to guarantee that the best possible classifier is achieved. The relative difficulty of performing parameter optimisation for different classifiers is often a key factor in algorithm selection. Moreover, customising the platforms is one of the most complex parts during experimental setup, especially when using pre-trained models from different datasets. In particular, customising data to be a competitor with the pre-trained model datasets can be difficult. For example, each dataset has a different data format, or the data must be categorised in a specific format. Furthermore, some parameters may be unsuitable for the searcher's objectives. Adjusting hyper-parameters to be capable of performing correctly can be another obstacle. In some cases, poor adjustment can lead to failure.

5.5 A summary of the Results

The deep learning architectures proposed, and the post-processing techniques, can accurately detect building boundaries, as shown in Figure 4.15, and Figure 4.16. The figures present four buildings that have different roof geometries, with bushes covering the roof boundaries; in addition, the roofs have multiple hips, and vary in length and width, and so on. Figures 4.16-1(e), and 4.17-1(e)

show that even small changes in the building's boundaries can be identified, these appear as small v-shaped notches. One of the most important aspects of the technique is the provision of useful directional information to assist in exposing major sections of the building. Moreover, based on that, it is possible to extract more semantic information. Take, for example, Figure 4.17-4(e), the algorithm provides the principal direction from left to right, which can be used to divide the building into an upper and lower part; trace the upper and lower boundaries; discover the two major building sectional changes and correlate them, as well as revealing important semantic informaion which shows the building has three sections. It is also possible to glean additional semantic information, that is, the building's left-hand section is smaller than the right-hand sectional part. Figure 4.16 shows the results for all four buildings using the same architecture, with ResNet-50 used alongside the proposed post-processing techniques.

A key primary contribution is that the snapping algorithm in Method 2 can find building references, using the lines connecting adjacent keypoints. Therefore, the building's orientation can be determined, as shown in Figures (4.11-(b) and 4.14). This helps to find the meaningful orientation direction of the building, based on which more semantic information can be extracted based on North azimuth direction obtained by `FixContour()` algorithm Method 2 . Further semantic information can also be derived; for example, that the building's left-most sectional part is smallest compared to the right-most sectional part. Employing RGB data from VHR satellite images to assist in the automated detection of urban areas improves the segmentation of buildings within their surroundings in a single image space. Therefore, this data could be beneficial for the automated detection of buildings with various geometries. The algorithm is simultaneous, acting on challenges such as the complexity of large cities. This enables the decision to be obtained more quickly. Furthermore, this study has been able to predict and reconstruct 2D building models from VHR imagery by considering the complex urban topography, characteristics of the target objects, the similarity between the

objects regions and their backgrounds in the same image, and the illumination conditions at imaging time, which are attached as an index. Additionally, the developed algorithm can sharpen the geometry boundaries with the help of deep learning techniques, producing references points and specifying the buildings orientations, which is a significant contribution. Several of the buildings' images have been used to properly evaluate the algorithms. To show accurate results, four images are given in Figures 4.16 and 4.17. The proposed algorithm was also compared with state-of-the-art methods, and the results are given in Table 5.2.

The results of the developed approach propose some critical points that can assist in developing an appropriate methodology. The strengths and consequences of this research, and its deficiencies in comparison to previous studies, including the general limitations of AI, follow:

- Data for training: Many of the deep learning techniques that are used in this research, such as GAN and Mask R-CNN, need a large number of images to achieve a trained model (e.g., thousands of images are required to train and achieve better accuracies).

- Computational power: Machine learning needs a large number of GPUs. Computational power is another limitation (e.g., uploading terra bytes of images for processing onto a cloud-based GPU platform requires a lot of time and cost).

- Experimental parameters setup issues: The model parameters define or express a model in ML or DL. Nevertheless, training a model entails selecting the optimal hyperparameters that the training set will use to learn the optimal parameters that correctly patch the input features (independent variables) to the labels or objectives (dependent variables), resulting in a source of learning, and the choice of an optimisation algorithm (e.g., gradient descent, stochastic gradient descent).

Chapter 6

Conclusions and Recommendations for Future Work

This chapter concentrates on the significance of this research and it makes some recommendations for future work to add to the findings. This study has developed a post-processing method using deep learning techniques to extract 2D building footprints, which can be used worldwide. This study is based on VHR satellite imagery data. The tool used in the case study (the Vaihingen/Germany dataset) has produced relevant qualitative and quantitative evaluations and is comparable to state-of-the-art remote sensing. The proposed deep learning method scored 95% in overall accuracy, while the algorithms have been shown to successfully reduce data loss.

Furthermore, the proposed method can improve the detection of the final output and extract complex 2D objects from VHR satellites (e.g. buildings, roads, vegetation etc.).The following section presents the significance of the research findings, followed by the conclusions of the research, and finally, the future research recommendations are set out.

6.1 Significance of the Research Findings

This section presents the study's results and findings concerning the research questions and hypotheses, which can be found in Section 1.2.

- **The first hypothesis of this research work states:** "Satellite imagery

data is sufficient to provide accurate LoD information for roof top detection." This study has found that the fusion of CNN-based methods is an effective solution to provide accurate LoD0 information for rooftop detection using VHR remotely sensed imagery. The complementarity attained by the CNN for deep spatial feature representation can improve the blurred boundaries, and the loss of fine spatial details can be reduced through the convolutional process. This research has revealed an effective solution to balance the trade-off between feature recognition and localisation, and it opens the way for detecting complex rooftop tasks through the use of VHR imagery.

Furthermore, modern commercial satellites can collect data based on to very specific customer requests. This means that, unlike government satellite missions whose satellites follow consistent paths, commercial satellites can be tasked to capture a particular location at a specific time. It guarantees the availability of satellite data, which otherwise could be missing or delayed due to changes in the satellite's route. This ability has radically changed disaster response and mitigation: whenever a natural or man-made disaster occurs, high-resolution satellites are the first to provide a detailed remote view of damaged territories inaccessible from the ground. This feature is coming soon to LandViewer, therefore it will send commercial satellites to the target area of interest to facilitate taking highly detailed images. However, some disadvantages include the high-cost of commercial satellites carrying the most advanced technology sensors, which will require significant investment. That is why satellite image data of high resolution is not cheap. To make it more affordable, reselling platforms like LandViewer would allow customers to pay only for the part of an image that covers the selected area. This is an excellent value for money option considering the price of an entire image. Another drawback of the HVR is that it always

involves small area coverage; the better the resolution, the less total ground area can be seen in an image. That is why high-resolution satellite data is more suitable for small-scale monitoring or analysis. It would take a few images from Pleiades-1, Kompsat-3 or SuperView-1 satellites to cover an area the size of Cardiff. A single Landsat 8 image, in turn, can capture a territory equal to many Cardiff cities. To sum up, the first matter is to figure out what exactly satellite data is needed for. For example, if the aim is to fulfil curiosity about how the Eiffel Tower or cities look from space, it is suitable to choose Google Maps to explore the still world in fine detail. On the other hand, if the goal is to study, map and monitor objects and changes in time, it is necessary to go to LandViewer and obtain a satellite image. A low- or medium-resolution image will be enough for free-of-charge observations of territories on a large scale (country or city scale). If there is a need to zoom in close to objects the size of a house or a car, the user should consider spending some money on a high-resolution image. In addition, because high resolution images are costly, it is important to use appropriate resolutions for machine learning models. Moreover, the long procedures needed for pre-processing if the resolution is not appropriate, in order to make the images suitable, defeats the object of unsupervised learning methods.

- **The second hypothesis of this research work states:** "An architecture method that is based on deep learning techniques can be used to classify and extract localised contours around building boundaries using VHR remotely-sensed images." A CNN was proposed for rooftop detection in an urban area using VHR images. CNN networks were applied to characterise objects and their spatial context by using within-object and between-object information. Specifically, Mask R-CNN networks with two different model structures were developed to predict linearly shaped objects (e.g. boundaries of the building rooftops). Multiple small window size

CNNs were sampled along each object based on geometric characteristics, and integrated through statistical majority voting; whereas the large window size CNN was used only once per object for prediction using a broad spatial context. A rule-based decision fusion was designed to integrate the class-specific distribution results conditional upon these two CNN models, in which the prediction of a linearly shaped object from the small window size CNNs was given priority; whereas the large window size CNN was trusted in alternative cases.

The effectiveness of the proposed CNN method was tested on large urban areas with complex roofs, such as the buildings in Vaihingen, Germany. CNN combined with large and small window sizes achieved excellent detection accuracy and computational efficiency. This study concludes that the object detection-based CNN method can effectively solve complex building footprint extraction 2D using VHR imagery. The proposed CNN method with two CNN networks model sizes is designed to sample specific locations defined by the size and geometry of image objects. It then integrates them in a class-specific manner to obtain an effective and efficient urban output. This method with large and small window size CNNs produced the most detailed class results compared to the sub-modules, and other contextual-based and object-based benchmark methods. Moreover, high computational efficiency was achieved with much more acceptable time requirements. Therefore, the proposed CNN method is effective and efficient for large urban areas using VHR imagery, and has great potential in many applications.

This study proposes a novel boundary regulated network for accurate roof segmentation and outline extraction from VHR aerial images. The Mask RCNN model can perform automatic segmentation and outline extraction from RGB images. Its performance has been verified through several experiments on a VHR dataset covering the data set containing 33 patches (of different sizes). With its unique boundary restriction and regulation design,

the proposed method has achieved significant results as well as producing keypoints obtained by the models used to classify and extract localised contours around building boundaries using VHR remotely-sensed images. In comparison, backbones RestNet 50 and 101 achieved gains of (0.941, 0.901, 0.819, 0.952) in F1 score, Recall, and IoU and overall accuracy, respectively.

- **The third hypothesis of this research work states:** "Post-processing techniques can significantly improve the output for detection accuracy." A novel post-processing algorithm has been proposed to enhance the output of deep CNNs. The proposed method illustrates how to create contours by iteratively applying the algorithms. The proposed post-processing algorithms and the CNN approach were evaluated by testing them on large urban datasets from Vaihingen, Germany. The findings show a progressively increasing trend in the error of identified buildings, with an average overall accuracy ranging from 94%. Complex land surface classes cast by shadows, which are exceedingly difficult to manage, were recognised correctly, as were complex land patterns (e.g. parking, large trucks, and overlapping objects). This study concludes that post-processing steps are necessary for unsupervised and automatic VHR remotely sensed imagery to efficiently solve complicated challenges, such as older, poorly landscaped areas. Additionally, the post-processing approaches and patch-based CNN models were commonly supportive for pixel-level and neighbourhood features, improving each other during classifier iteration, and accurately extracting additional information from detected buildings. Sharpness calculations have demonstrated that adding post-processing algorithms improves model performance by 2.5%, 4.86%, 1.2% in F1 score, IoU and overall accuracy respectively, as can be seen in Table 4.2 and Table 5.1.

- **The fourth hypothesis of this research work states:** "Detection of the vectors performed by deep learning architecture-based methods can

provide vital information to help post-processing and determine useful geometrical information for building detection." While vectors are utilised in a wide range of industries, their usage in machine learning is particularly notable. From a ML perspective, the advantage of vectors is that they can be converted from raster data to vectors at various stages of an ML project; this can be accomplished through the mathematical definition of vectors, which transfers all the information required to work with and understand the final ML outputs. For instance, Mask R-CNN keypoints could be employed as unlabelled vectors to compute and extract rooftop orientations of buildings. The case studies have revealed the utility of the algorithms described in the third hypothesis of this research. The results show that the proposed algorithms generate suitable vectors when conducted using Mask R-CNN. Therefore, they can be used as an alternative method to extract additional information from detected buildings, such as the orientation. It should be noted that there is remarkable rigidity in the shape of the building, and the fragmentation of minute segments is removed. As a result, the extension makes it much easier to display a specific object's keypoints as a single hot mask or ground-truth mask, and then use the Mask R-CNN framework to execute this objective. In other words, this approach is a single framework that can support bounding box detection, mask segmentation, and keypoint detection. Furthermore, a vector is a mathematical object that is represented in space by an array of numbers with an origin, direction, and magnitude length. In addition, to represent a location, or even a change in a mathematical framework or space conceptually, vector space elements are a collection of objects that are closed by adding rules, and they support a method for scalar multiplication that can be used in many fields, such as medical imagery and advanced scientific multiplication.

6.2 Conclusions

In line with the research aim and objectives, this thesis has presented the methods used for the re-purposing of pre-trained ML models, and the development of novel post-processing algorithms for extracting 2D building footprints using deep learning techniques. The newly developed novel approach demonstrates the automatic generation of 2D building models from a single VHR satellite imagery. The computation discussed may be applied to a wide range of research and experimentation. Moreover, despite the limitations of machine learning approaches, particularly with regard to small projects such as this and the restrictions faced, positive outcomes have been achieved. However, if the results are applied to larger, government-scale projects, these limitations will not be faced.The conclusions drawn from this current research are as follows:

- Deep learning algorithms are beneficial for remote sensing as semi-supervised or unsupervised applications to solve problems whose representations are complex (e.g., nonlinear or even undefined classes), and therefore cannot be generalised.

- Deep learning approaches should be followed by custom post-processing algorithms (depending on the application's aims and types) to solve technical challenges, including parameter differences.

- Joining more DL architectures in one application was not an easy task. Although the results may be distinctive, these methods require experts who have experience in DL architecture networks, also require high computational power, more time and data for training.

- The effect of using different backbones:
  In terms of post-processing, there are a few parameters for which we must select appropriate values. In the majority of the experiments, the shape of

the final contour from the proposed algorithm depends on the quality of the reference line and the quality of the mask. When the model's backbone was upgraded from ResNet50 to ResNet101, precision, IoU, and overall accuracy improved by 14 %, 12 %, and 14 %, respectively.

6.3 Recommendations for Future Research

The research presented has developed deep learning algorithms for object detection using VHR remotely sensed imagery. However, additional investigation is required because several aspects of the proposed methodologies have not been thoroughly addressed or examined. The critical limitation of this research is the lack of testing for the employability of created methodologies. This research examined transferability using the Vaihingen semantic segmentation datasets, where training was done on some annotated tiles, and testing was done on the image tiles that were not used for training. However, most studies in this area used the same training and testing data, and the methods were verified only in specific regions without transferring to new unexplored regions. Thus, future work should evaluate the applicability of these approaches to a larger geographical area to verify that these methods are robust, adaptable, and scalable. Apart from the previously described adaptable methods, the proposed deep learning methods could be developed from various angles, including data sources, approaches, and implementations. The following detailed suggestions are proposed:

- Using multi-source geospatial data

  This research has focused on building detection using VHR remotely sensed imagery. However, many other data sources exist in the field of remote sensing, such as hyperspectral, SAR, and LiDAR, which have not been used in this study. The fusion of these multiple data sources by designing novel deep architectures would be another direction for further research. In addition,

the object detection in this thesis was undertaken at a generalised spatial and semantic level (e.g., building class), without identifying other classes (e.g., trees, roads, cars etc.). This subject could be addressed further by incorporating multi-source geospatial data (e.g. classified geospatial data in large areas might be further distinguished). Therefore, future research should use semantic information from other data sources, such as SAR and LiDAR, to generalise the spatial and semantic classification of the feature map classes. In particular, the fusion of deep learning models is a potential approach in the field of remote sensing. Thus, images produced by remote sensing directly result from physical operations, such as light reflection and microwave scattering. Hence, a combination of physics-based models that describe the process underlying the images and newly created artificial intelligence technology in their field could be used.

- Approaches

  This thesis proposes two different approaches to detect urban 2D objects utilising fixed input patch sizes. However, the features retrieved in urban areas are essentially scale-dependent and appear at various scales. Therefore, it is recommended that further research should adapt a multitude of CNNs with different input patch sizes to the variable sizes and shapes of urban objects via weighted decision fusion. Additionally, a 2D object detection can be expanded to a 3D object detection, reconstruction, and classification framework. These multi-layer representations necessitate further development of the deep learning framework to incorporate additional deep learning models into the iterative process, which may support one another through collaborative remote sensing assemblies and an optimisation framework.

- Implementations

  Numerous potential applications could be used to show the impact of the

methodologies that have been developed in this research. One option is to investigate change detection in rapidly increasing metropolitan areas, where organisational structures and processes, including local governments, and small and medium-sized businesses, require assistance in making decisions. Further deep learning approaches, such as recurrent neural networks (RNN) and Long short-term memory (LSTM), can be built to simulate land cover and land use patterns through time-series analysis. Additionally, this research has focused on the detection of buildings in urban environments. Apart from addressing technology issues, deep learning in remote sensing enables new applications, such as monitoring global changes and analysing protected areas. In this environment, deep learning provides a practical framework that enables remote sensing researchers to expand the scope of the field.

Finally, a database should be built to contain all of the remote sensing features to help develop deep learning approaches in the remote sensing field. This database should be open source and include a variety of data sources, scales, pre-trained models, meta-architectures, and meta-algorithms, similar to machine learning dataset platforms (e.g., COCO, ImageNet, Zoo etc.). This platform database would help to develop competitive pre-trained models for further research, which would help the remote sensing field develop many frameworks and standards for a range of applications and aspects. Furthermore, some other aspects that are significant for further research in remote sensing are as follows:

- Building a foundation of frameworks and standards for open-source data set formatting, especially for remote sensing, as this will help to encourage further research.
- Pre-trained models, meta-architectures, and meta-algorithms would diminish many ML drawbacks such as computational power, amount of data

needed for training, massive backbone architectures, and experimental parameter setup.

Finally, it is hoped that this research will inspire advanced applications that will extend the study of remote sensing, such as research supported by government departments, or large-scale national and international institutions. Furthermore, the post-processing could be used in alternative fields, with its benefits extending to other disciplines.

Bibliography

[1] Frederik Tack, Gurcan Buyuksalih, and Rudi Goossens. 3d building reconstruction based on given ground plan information and surface models extracted from spaceborne imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 67:52–64, 2012.

[2] Gong Cheng and Junwei Han. A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:11–28, 2016.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[5] Yilei Shi, Qingyu Li, and Xiao Xiang Zhu. Building footprint generation using improved generative adversarial networks. *IEEE Geoscience and Remote Sensing Letters*, 16(4):603–607, 2018.

[6] Wenzhi Zhao, Shihong Du, Qiao Wang, and William J Emery. Contextually guided very-high-resolution imagery classification with semantic segments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 132:48–60, 2017.

[7] Dimitrios Marmanis, Konrad Schindler, Jan Dirk Wegner, Silvano Galliani, Mihai Datcu, and Uwe Stilla. Classification with an edge: Improving semantic image segmentation with boundary detection. *ISPRS Journal of Photogrammetry and Remote Sensing*, 135:158–172, 2018.

[8] Serkan Kiranyaz, Miguel Ferreira, and Moncef Gabbouj. Automatic object extraction over multiscale edge field for multimedia retrieval. *IEEE Transactions on Image Processing*, 15(12):3759–3772, 2006.

[9] Leo Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (11):1768–1783, 2006.

[10] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

[11] Dzung L Pham, Chenyang Xu, and Jerry L Prince. Current methods in medical image segmentation. *Annual review of biomedical engineering*, 2(1):315–337, 2000.

[12] Kenneth I Laws. Textured image segmentation. Technical report, University of Southern California Los Angeles Image Processing INST, 1980.

[13] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):884–900, 1996.

[14] Luminita A Vese and Tony F Chan. A multiphase level set framework for image segmentation using the mumford and shah model. *International journal of computer vision*, 50(3):271–293, 2002.

[15] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[16] Theodosios Pavlidis and Y-T Liow. Integrating region growing and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233, 1990.

[17] Larry S Davis. A survey of edge detection techniques. *Computer graphics and image processing*, 4(3):248–270, 1975.

[18] Luis Alvarez, Pierre-Louis Lions, and Jean-Michel Morel. Image selective smoothing and edge detection by nonlinear diffusion. ii. *SIAM Journal on numerical analysis*, 29(3):845–866, 1992.

[19] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 465–470. IEEE, 1996.

[20] Azriel Rosenfeld and Mark Thurston. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers*, (5):562–569, 1971.

[21] Peter Meer and Bogdan Georgescu. Edge detection with embedded confidence. *IEEE Transactions on pattern analysis and machine intelligence*, 23(12):1351–1365, 2001.

[22] Markus Stricker and Michael Swain. The capacity of color histogram indexing. In *CVPR*, volume 94, pages 704–708, 1994.

[23] Jifeng Ning, Lei Zhang, David Zhang, and Chengke Wu. Robust object tracking using joint color-texture histogram. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(07):1245–1263, 2009.

[24] A Müfit Ferman, A Murat Tekalp, and Rajiv Mehrotra. Robust color histogram descriptors for video segment retrieval and identification. *IEEE Transactions on image processing*, 11(5):497–508, 2002.

[25] Carol L Novak and Steven A Shafer. Anatomy of a color histogram. In *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 599–605. IEEE, 1992.

[26] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[27] Jianping Fan, David KY Yau, Ahmed K Elmagarmid, and Walid G Aref. Automatic image segmentation by integrating color-edge extraction and seeded region growing. *IEEE transactions on image processing*, 10(10):1454–1466, 2001.

[28] Le Wang, Jianru Xue, Nanning Zheng, and Gang Hua. Automatic salient object extraction with contextual cue. In *2011 international conference on computer vision*, pages 105–112. IEEE, 2011.

[29] Jianping Fan, Xingquan Zhu, and Lide Wu. Automatic model-based semantic object extraction algorithm. *IEEE Transactions on circuits and systems for video technology*, 11(10):1073–1084, 2001.

[30] Çağlar Aytekin, Serkan Kiranyaz, and Moncef Gabbouj. Quantum mechanics in computer vision: automatic object extraction. In *2013 IEEE International Conference on Image Processing*, pages 2489–2493. IEEE, 2013.

[31] Haonan Yu, Jia Li, Yonghong Tian, and Tiejun Huang. Automatic interesting object extraction from images using complementary saliency maps. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 891–894. ACM, 2010.

[32] Wei Zeng, Wen Gao, and Debin Zhao. Automatic moving object extraction in mpeg video. In *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS'03.*, volume 2, pages II–II. IEEE, 2003.

[33] Stefan Hinz and Albert Baumgartner. Automatic extraction of urban road networks from multi-view aerial imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 58(1-2):83–98, 2003.

[34] Ye Lu and Ze-Nian Li. Automatic object extraction and reconstruction in active video. *Pattern Recognition*, 41(3):1159–1172, 2008.

[35] Imane Sebari and Dong-Chen He. Automatic fuzzy object-based analysis of vhsr images for urban objects extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 79:171–184, 2013.

[36] Taejung Kim and Jan-Peter Muller. Development of a graph-based approach for building detection. *Image and Vision Computing*, 17(1):3–14, 1999.

[37] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[38] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geomet-

ric flows. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2290–2297, 2009.

[39] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *European conference on computer vision*, pages 352–365. Springer, 2010.

[40] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *European conference on computer vision*, pages 13–26. Springer, 2012.

[41] Zhenguo Li, Xiao-Ming Wu, and Shih-Fu Chang. Segmentation using superpixels: A bipartite graph partitioning approach. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 789–796. IEEE, 2012.

[42] Jason Chang, Donglai Wei, and John W Fisher. A video representation using temporal superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2058, 2013.

[43] Fabio Galasso, Roberto Cipolla, and Bernt Schiele. Video segmentation with superpixels. In *Asian Conference on Computer Vision*, pages 760–774. Springer, 2012.

[44] Yuhang Zhang, Richard Hartley, John Mashford, and Stewart Burn. Superpixels via pseudo-boolean optimization. In *2011 International Conference on Computer Vision*, pages 1387–1394. IEEE, 2011.

[45] Peng Wang, Gang Zeng, Rui Gan, Jingdong Wang, and Hongbin Zha. Structure-sensitive superpixels via geodesic distance. *International journal of computer vision*, 103(1):1–21, 2013.

[46] Na Tong, Huchuan Lu, Lihe Zhang, and Xiang Ruan. Saliency detection with multi-scale superpixels. *IEEE Signal Processing Letters*, 21(9):1035–1039, 2014.

[47] Matthias Reso, Jorn Jachalsky, Bodo Rosenhahn, and Jorn Ostermann. Temporally consistent superpixels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 385–392, 2013.

[48] Michael Van den Bergh, Xavier Boix, Gemma Roig, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. *International Journal of Computer Vision*, 111(3):298–314, 2015.

[49] Junjie Yan, Yinan Yu, Xiangyu Zhu, Zhen Lei, and Stan Z Li. Object detection by labeling superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5107–5116, 2015.

[50] Alastair P Moore, Simon JD Prince, and Jonathan Warrell. "lattice cut"-constructing superpixels using layer constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2117–2124. IEEE, 2010.

[51] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, 2018.

[52] Guang Shu, Afshin Dehghan, and Mubarak Shah. Improving an object detector and extracting regions using superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3721–3727, 2013.

[53] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[54] Heng-Da Cheng, X_ H_ Jiang, Ying Sun, and Jingli Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[55] Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):1026–1038, 2002.

[56] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639, 1990.

[57] Francine Catté, Pierre-Louis Lions, Jean-Michel Morel, and Tomeu Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical analysis*, 29(1):182–193, 1992.

[58] VINCENT TORRE and TA Poggio. On edge detection ieee transaction on pattern analysis and machine intelligence, vol, 1986.

[59] James Lee, R Haralick, and Linda Shapiro. Morphologic edge detection. *IEEE Journal on Robotics and Automation*, 3(2):142–156, 1987.

[60] Leila Shafarenko, H Petrou, and Josef Kittler. Histogram-based segmentation in a perceptually uniform color space. *IEEE transactions on image processing*, 7(9):1354–1358, 1998.

[61] K Konstantinidis, A Gasteratos, and I Andreadis. Image retrieval based on fuzzy color histogram processing. *Optics Communications*, 248(4-6):375–386, 2005.

[62] Greg Pass and Ramin Zabih. Histogram refinement for content-based image retrieval. In *Proceedings Third IEEE Workshop on Applications of Computer Vision. WACV'96*, pages 96–102. IEEE, 1996.

[63] Markus Andreas Stricker and Markus Orengo. Similarity of color images. In *Storage and retrieval for image and video databases III*, volume 2420, pages 381–392. International Society for Optics and Photonics, 1995.

[64] Alan L Yuille, Peter W Hallinan, and David S Cohen. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2):99–111, 1992.

[65] Øivind Due Trier, Anil K Jain, and Torfinn Taxt. Feature extraction methods for character recognition-a survey. *Pattern recognition*, 29(4):641–662, 1996.

[66] Nalini K Ratha, Shaoyun Chen, and Anil K Jain. Adaptive flow orientation-based feature extraction in fingerprint images. *Pattern Recognition*, 28(11):1657–1672, 1995.

[67] Fei Chen, Huimin Yu, Roland Hu, and Xunxun Zeng. Deep learning shape priors for object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1870–1877, 2013.

[68] Huazhu Fu, Yanwu Xu, Stephen Lin, Damon Wing Kee Wong, and Jiang Liu. Deepvessel: Retinal vessel segmentation via deep learning and conditional random field. In *International conference on medical image computing and computer-assisted intervention*, pages 132–139. Springer, 2016.

[69] Yohhan Pao. Adaptive pattern recognition and neural networks. 1989.

[70] Haifeng Xu, Akmal A Younis, and Mansur R Kabuka. Automatic moving object extraction for content-based applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(6):796–812, 2004.

[71] Shih-Chia Huang and Bo-Hao Chen. Automatic moving object extraction through a real-world variable-bandwidth network for traffic monitoring systems. *IEEE Transactions on Industrial Electronics*, 61(4):2099–2112, 2013.

[72] Athena Stassopoulou and Terry Caelli. Building detection using bayesian networks. *International journal of pattern recognition and artificial intelligence*, 14(06):715–733, 2000.

[73] Nikhil R Pal and Sankar K Pal. A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294, 1993.

[74] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[75] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. *Computer vision, graphics, and image processing*, 29(1):100–132, 1985.

[76] King-Sun Fu and JK Mui. A survey on image segmentation. *Pattern recognition*, 13(1):3–16, 1981.

[77] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[78] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International journal of computer vision*, 43(1):7–27, 2001.

[79] John Canny. A computational approach to edge detection. In *Readings in computer vision*, pages 184–203. Elsevier, 1987.

[80] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[81] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *International journal of computer vision*, 30(2):117–156, 1998.

[82] Paul Bao, Lei Zhang, and Xiaolin Wu. Canny edge detection enhancement by scale multiplication. *IEEE transactions on pattern analysis and machine intelligence*, 27(9):1485–1490, 2005.

[83] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

[84] Todd R Reed and JM Hans Dubuf. A review of recent texture segmentation and feature extraction techniques. *CVGIP: Image understanding*, 57(3):359–372, 1993.

[85] Jianchang Mao and Anil K Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE transactions on neural networks*, 6(2):296–317, 1995.

[86] Mingqiang Yang, Kidiyo Kpalma, and Joseph Ronsin. A survey of shape feature extraction techniques, 2008.

[87] Zi-Quan Hong. Algebraic feature extraction of image for recognition. *Pattern recognition*, 24(3):211–219, 1991.

[88] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[89] Shu Liao, Yaozong Gao, Aytekin Oto, and Dinggang Shen. Representation learning: a unified deep learning framework for automatic prostate mr segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 254–261. Springer, 2013.

[90] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750, 2015.

[91] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[92] Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. Neocognitron: A neural network model for a mechanism of visual pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (5):826–834, 1983.

[93] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[94] Chungan Lin and Ramakant Nevatia. Building detection and description from a single intensity image. *Computer vision and image understanding*, 72(2):101–121, 1998.

[95] Wei-Te Li, Haw-Shiuan Chang, Kuo-Chin Lien, Hui-Tang Chang, and Yu-Chiang Frank Wang. Exploring visual and motion saliency for automatic video object extraction. *IEEE Transactions on image processing*, 22(7):2600–2610, 2013.

[96] Caglar Aytekin, Serkan Kiranyaz, and Moncef Gabbouj. Automatic object segmentation by quantum cuts. In *2014 22nd International Conference on Pattern Recognition*, pages 112–117. IEEE, 2014.

[97] Maria Vakalopoulou, Konstantinos Karantzalos, Nikos Komodakis, and Nikos Paragios. Building detection in very high resolution multispectral data with deep learning features. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1873–1876. IEEE, 2015.

[98] Zoran Zivkovic and Ben Krose. An em-like algorithm for color-histogram-based object tracking. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. IEEE, 2004.

[99] Patrick Pérez, Carine Hue, Jaco Vermaak, and Michel Gangnet. Color-based probabilistic tracking. In *European Conference on Computer Vision*, pages 661–675. Springer, 2002.

[100] Harpreet S Sawhney and James L Hafner. Efficient color histogram indexing. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 66–70. IEEE, 1994.

[101] Michael J Swain and Dana H Ballard. Indexing via color histograms. In *Active perception and robot vision*, pages 261–273. Springer, 1992.

[102] James Hafner, Harpreet S. Sawhney, William Equitz, Myron Flickner, and Wayne Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE transactions on pattern analysis and machine intelligence*, 17(7):729–736, 1995.

[103] Shamik Sural, Gang Qian, and Sakti Pramanik. Segmentation and histogram generation using the hsv color space for image retrieval. In *Proceedings. International Conference on Image Processing*, volume 2, pages II–II. IEEE, 2002.

[104] Ju Han and Kai-Kuang Ma. Fuzzy color histogram and its use in color image retrieval. *IEEE Transactions on image Processing*, 11(8):944–952, 2002.

[105] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress, 2011.

[106] Joost Van De Weijer and Cordelia Schmid. Coloring local feature extraction. In *European conference on computer vision*, pages 334–348. Springer, 2006.

[107] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[108] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990.

[109] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[110] Bernard Widrow, Rodney G Winter, and Robert A Baxter. Layered neural nets for pattern recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1109–1118, 1988.

[111] Rosa Lasaponara and Nicola Masini. Detection of archaeological crop marks by using satellite quickbird multispectral imagery. *Journal of archaeological science*, 34(2):214–221, 2007.

[112] Jing Peng, Dong Zhang, and Yuncai Liu. An improved snake model for building detection from urban aerial images. *Pattern Recognition Letters*, 26(5):587–595, 2005.

[113] Limor Shashua-Bar and Michael E Hoffman. Vegetation as a climatic component in the design of an urban street: An empirical model for predicting the cooling effect of urban green areas with trees. *Energy and buildings*, 31(3):221–235, 2000.

[114] Zheng Tan, Kevin Ka-Lun Lau, and Edward Ng. Urban tree design approaches for mitigating daytime urban heat island effects in a high-density urban environment. *Energy and Buildings*, 114:265–274, 2016.

[115] Laura E Jackson. The relationship of urban design to human health and condition. *Landscape and urban planning*, 64(4):191–200, 2003.

[116] Ni-Bin Chang, YL Wei, CC Tseng, and C-YJ Kao. The design of a gis-based decision support system for chemical emergency preparedness and response in an urban environment. *Computers, Environment and Urban Systems*, 21(1):67–94, 1997.

[117] Gideon S Golany. Urban design morphology and thermal performance. *Atmospheric Environment*, 30(3):455–465, 1996.

[118] Hannah Badland and Grant Schofield. Transport, urban design, and physical activity: an evidence-based update. *Transportation Research Part D: Transport and Environment*, 10(3):177–196, 2005.

[119] Lawrence D Frank, Martin A Andresen, and Thomas L Schmid. Obesity relationships with community design, physical activity, and time spent in cars. *American journal of preventive medicine*, 27(2):87–96, 2004.

[120] Sam CM Hui. Low energy building design in high density urban cities. *Renewable energy*, 24(3-4):627–640, 2001.

[121] Luciano Alparone, Stefano Baronti, Andrea Garzelli, and Filippo Nencini. A global quality measurement of pan-sharpened multispectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 1(4):313–317, 2004.

[122] P Ready and P Wintz. Information extraction, snr improvement, and data compression in multispectral imagery. *IEEE Transactions on communications*, 21(10):1123–1131, 1973.

[123] Paul G Lucey, David T Blewett, and B Ray Hawke. Mapping the feo and tio2 content of the lunar surface with multispectral imagery. *Journal of Geophysical Research: Planets*, 103(E2):3679–3699, 1998.

[124] Juan B Mena. State of the art on automatic road extraction for gis update: a novel classification. *Pattern recognition letters*, 24(16):3037–3058, 2003.

[125] William D Philpot. Bathymetric mapping with passive multispectral imagery. *Applied optics*, 28(8):1569–1578, 1989.

[126] Fabrice Rossi, Amaury Lendasse, Damien François, Vincent Wertz, and Michel Verleysen. Mutual information for the selection of relevant variables in spectrometric nonlinear modelling. *Chemometrics and intelligent laboratory systems*, 80(2):215–226, 2006.

[127] Yun Zhang. Optimisation of building detection in satellite images by combining multispectral classification and texture filtering. *ISPRS journal of photogrammetry and remote sensing*, 54(1):50–60, 1999.

[128] Petter Weibring, Thomas Johansson, Hans Edner, Sune Svanberg, Barbro Sundnér, Valentina Raimondi, Giovanna Cecchi, and Luca Pantani. Fluorescence lidar imaging of historical monuments. *Applied Optics*, 40(33):6111–6120, 2001.

[129] D Lognoli, G Cecchi, I Mochi, L Pantani, V Raimondi, R Chiari, Thomas Johansson, Petter Weibring, Hans Edner, and Sune Svanberg. Fluorescence lidar imaging of the cathedral and baptistery of parma. *Applied Physics B*, 76(4):457–465, 2003.

[130] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[131] Fumiki Hosoi and Kenji Omasa. Estimating vertical plant area density profile and growth parameters of a wheat canopy at different growth stages using three-dimensional portable lidar imaging. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(2):151–158, 2009.

[132] John A Saghri, Andrew G Tescher, and John T Reagan. Practical transform coding of multispectral imagery. *IEEE signal processing magazine*, 12(1):32–43, 1995.

[133] Sandra J Winterbottom and David J Gilvear. Quantification of channel bed morphology in gravel-bed rivers using airborne multispectral imagery and aerial photography. *Regulated Rivers: Research & Management: An International Journal Devoted to River Research and Management*, 13(6):489–499, 1997.

[134] Ildiko Suveg and George Vosselman. Reconstruction of 3d building models from aerial images and maps. *ISPRS Journal of Photogrammetry and remote sensing*, 58(3-4):202–224, 2004.

[135] Beril Sirmacek and Cem Unsalan. Building detection from aerial images using invariant color features and shadow information. In *2008 23rd International Symposium on Computer and Information Sciences*, pages 1–5. IEEE, 2008.

[136] Duo-Min He and Gerald GL Seet. Divergent-beam lidar imaging in turbid water. *Optics and lasers in engineering*, 41(1):217–231, 2004.

[137] Franz Rottensteiner, Gunho Sohn, Markus Gerke, Jan Dirk Wegner, Uwe Breitkopf, and Jaewook Jung. Results of the isprs benchmark on urban

object detection and 3d building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:256–271, 2014.

[138] You Hongjian and Zhang Shiqiang. 3d building reconstruction from aerial ccd image and sparse laser sample data. *Optics and Lasers in Engineering*, 44(6):555–566, 2006.

[139] Txomin Hermosilla, Luis A Ruiz, Jorge A Recio, and Javier Estornell. Evaluation of automatic building detection approaches combining high resolution images and lidar data. *Remote Sensing*, 3(6):1188–1210, 2011.

[140] F Mark Danson, David Hetherington, Felix Morsdorf, Benjamin Koetz, and Britta Allgower. Forest canopy gap fraction from terrestrial laser scanning. *IEEE Geoscience and remote sensing letters*, 4(1):157–160, 2007.

[141] Mohamar Moussa Ouédraogo, Aurore Degré, Charles Debouche, and Jonathan Lisein. The evaluation of unmanned aerial system-based photogrammetry and terrestrial laser scanning to generate dems of agricultural watersheds. *Geomorphology*, 214:339–355, 2014.

[142] Liang Cheng, Lihua Tong, Yanming Chen, Wen Zhang, Jie Shan, Yongxue Liu, and Manchun Li. Integration of lidar data and optical multi-view images for 3d reconstruction of building roofs. *Optics and Lasers in Engineering*, 51(4):493–502, 2013.

[143] Ramakant Nevatia and K Ramesh Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13(3):257–269, 1980.

[144] James H Churnside and James J Wilson. Airborne lidar imaging of salmon. *Applied optics*, 43(6):1416–1424, 2004.

[145] Alexander Prokop. Assessing the applicability of terrestrial laser scanning for spatial snow depth measurements. *Cold Regions Science and Technology*, 54(3):155–163, 2008.

[146] Farhad Samadzadegan, Ali Azizi, Michael Hahn, and Curo Lucas. Automatic 3d object recognition and reconstruction based on neuro-fuzzy modelling. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(5):255–277, 2005.

[147] Fumiki Hosoi and Kenji Omasa. Factors contributing to accuracy in the estimation of the woody canopy leaf area density profile using 3d portable lidar imaging. *Journal of experimental botany*, 58(12):3463–3473, 2007.

[148] Mingyue Lu and Yongjian He. Organization and indexing method for 3d points cloud data. *Geo-Information Science*, 2, 2008.

[149] Jan Elseberg, Dorit Borrmann, and Andreas Nüchter. One billion points in the cloud–an octree for efficient processing of 3d laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 76:76–88, 2013.

[150] Frédéric Bosché and Emeline Guenet. Automating surface flatness control using terrestrial laser scanning and building information models. *Automation in construction*, 44:212–226, 2014.

[151] Timothy L Haithcoat, Wenbo Song, and James D Hipple. Building footprint extraction and 3-d reconstruction from lidar data. In *IEEE/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas (Cat. No. 01EX482)*, pages 74–78. IEEE, 2001.

[152] José Luis Lerma, Santiago Navarro, Miriam Cabrelles, and Valentín Villaverde. Terrestrial laser scanning and close range photogrammetry for 3d archaeological documentation: the upper palaeolithic cave of parpalló as a case study. *Journal of Archaeological Science*, 37(3):499–507, 2010.

[153] B Riveiro, P Morer, P Arias, and I De Arteaga. Terrestrial laser scanning and limit analysis of masonry arch bridges. *Construction and building materials*, 25(4):1726–1735, 2011.

[154] Weixing Wang, Weisen Zhao, Lingxiao Huang, Vivian Vimarlund, and Zhiwei Wang. Applications of terrestrial laser scanning for tunnels: a review. *Journal of Traffic and Transportation Engineering (English Edition)*, 1(5):325–337, 2014.

[155] George L Heritage and David J Milan. Terrestrial laser scanning of grain roughness in a gravel-bed river. *Geomorphology*, 113(1-2):4–11, 2009.

[156] Stephanie Fekete. *Geotechnical applications of LiDAR for geomechanical characterization in drill and blast tunnels and representative 3-dimensional discontinuum modelling.* PhD thesis, 2010.

[157] Hao Yang, Xiangyang Xu, and Ingo Neumann. Optimal finite element model with response surface methodology for concrete structures based on terrestrial laser scanning technology. *Composite Structures*, 183:2–6, 2018.

[158] M Sturzenegger and D Stead. Close-range terrestrial digital photogrammetry and terrestrial laser scanning for discontinuity characterization on rock cuts. *Engineering Geology*, 106(3-4):163–182, 2009.

[159] Hao Yang, Mohammad Omidalizarandi, Xiangyang Xu, and Ingo Neumann. Terrestrial laser scanning technology for deformation monitoring and surface modeling of arch structures. *Composite Structures*, 169:173–179, 2017.

[160] J Armesto, Javier Roca-Pardiñas, H Lorenzo, and P Arias. Modelling masonry arches shape using terrestrial laser scanning data and nonparametric methods. *Engineering Structures*, 32(2):607–615, 2010.

[161] H González-Jorge, B Riveiro, J Armesto, and P Arias. Standard artifact for the geometric verification of terrestrial laser scanning systems. *Optics & Laser Technology*, 43(7):1249–1256, 2011.

[162] Mathieu Brédif, Didier Boldo, Marc Pierrot-Deseilligny, and Henri Maître. 3d building reconstruction with parametric roof superstructures. In *2007 IEEE International Conference on Image Processing*, volume 2, pages II–537. IEEE, 2007.

[163] Lu Wang and Chee-hung Henry Chu. 3d building reconstruction from lidar data. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 3054–3059. IEEE, 2009.

[164] Yusuf Arayici. An approach for real world data modelling with the 3d terrestrial laser scanner for built environment. *Automation in construction*, 16(6):816–829, 2007.

[165] Florent Lafarge, Xavier Descombes, Josiane Zerubia, and Marc Pierrot-Deseilligny. Building reconstruction from a single dem. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[166] Carlos A Vanegas, Daniel G Aliaga, and Bedrich Benes. Automatic extraction of manhattan-world building masses from 3d laser range scans.

IEEE transactions on visualization and computer graphics, 18(10):1627–1637, 2012.

[167] Vivek Verma, Rakesh Kumar, and Stephen Hsu. 3d building detection and modeling from aerial lidar data. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2213–2220. IEEE, 2006.

[168] John W McLean and Jonathan D Freeman. Effects of ocean waves on airborne lidar imaging. *Applied optics*, 35(18):3261–3269, 1996.

[169] John James Degnan III and David Nelson Wells. Scanner/optical system for three-dimensional lidar imaging and polarimetry, July 23 2013. US Patent 8,493,445.

[170] Alexander Prokop, Peter Schön, Florian Singer, Gaëtan Pulfer, Mohamed Naaim, Emmanuel Thibert, and Alvaro Soruco. Merging terrestrial laser scanning technology with photogrammetric and total station data for the determination of avalanche modeling parameters. *Cold Regions Science and Technology*, 110:223–230, 2015.

[171] Susan L Handy, Marlon G Boarnet, Reid Ewing, and Richard E Killingsworth. How the built environment affects physical activity: views from urban planning. *American journal of preventive medicine*, 23(2):64–73, 2002.

[172] Xin Huang, Liangpei Zhang, and Pingxiang Li. Classification and extraction of spatial features in urban areas using high-resolution multispectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 4(2):260–264, 2007.

[173] Kostas Daniilidis, Petros Maragos, and Nikos Paragios. *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings*, volume 6315. Springer, 2010.

[174] David Weikersdorfer, David Gossow, and Michael Beetz. Depth-adaptive superpixels. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 2087–2090. IEEE, 2012.

[175] Alexander Schick, Mika Fischer, and Rainer Stiefelhagen. Measuring and evaluating the compactness of superpixels. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 930–934. IEEE, 2012.

[176] Hyunggi Cho, Young-Woo Seo, BVK Vijaya Kumar, and Ragunathan Raj Rajkumar. A multi-sensor fusion system for moving object detection and tracking in urban driving environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1836–1843. IEEE, 2014.

[177] Helmut Mayer. Automatic object extraction from aerial imagery—a survey focusing on buildings. *Computer vision and image understanding*, 74(2):138–149, 1999.

[178] Naci Yastikli. Documentation of cultural heritage using digital photogrammetry and laser scanning. *Journal of Cultural Heritage*, 8(4):423–427, 2007.

[179] Mostafa Kabolizade, Hamid Ebadi, and Ali Mohammadzadeh. Design and implementation of an algorithm for automatic 3d reconstruction of building models using genetic algorithm. *International Journal of Applied Earth Observation and Geoinformation*, 19:104–114, 2012.

[180] Chengqiang Zhao, Wenlin Gong, Mingliang Chen, Enrong Li, Hui Wang, Wendong Xu, and Shensheng Han. Ghost imaging lidar via sparsity constraints. *Applied Physics Letters*, 101(14):141123, 2012.

[181] Thomas H Painter, Daniel F Berisford, Joseph W Boardman, Kathryn J Bormann, Jeffrey S Deems, Frank Gehrke, Andrew Hedrick, Michael Joyce, Ross Laidlaw, Danny Marks, et al. The airborne snow observatory: Fusion of scanning lidar, imaging spectrometer, and physically-based modeling for mapping snow water equivalent and snow albedo. *Remote Sensing of Environment*, 184:139–152, 2016.

[182] PN Bierwirth, TJ Lee, and RV Burne. Shallow sea-floor reflectance and water depth derived by unmixing multispectral imagery. *Photogrammetric Engineering and Remote Sensing;(United States)*, 59(3), 1993.

[183] Franz Rottensteiner, John Trinder, Simon Clode, and Kurt Kubik. Using the dempster–shafer method for the fusion of lidar data and multi-spectral images for building detection. *Information fusion*, 6(4):283–300, 2005.

[184] Xuelian Meng, Le Wang, and Nate Currit. Morphology-based building detection from airborne lidar data. *Photogrammetric Engineering & Remote Sensing*, 75(4):437–442, 2009.

[185] Gregory P Asner, David E Knapp, Joseph Boardman, Robert O Green, Ty Kennedy-Bowdoin, Michael Eastwood, Roberta E Martin, Christopher Anderson, and Christopher B Field. Carnegie airborne observatory-2: Increasing science data dimensionality via high-fidelity multi-sensor fusion. *Remote Sensing of Environment*, 124:454–465, 2012.

[186] Franz Rottensteiner, John Trinder, Simon Clode, and Kurt Kubik. Building detection by fusion of airborne laser scanner data and multi-spectral images: Performance evaluation and sensitivity analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(2):135–149, 2007.

[187] Yi Hui Lu, John C Trinder, and Kurt Kubik. Automatic building detection using the dempster-shafer algorithm. *Photogrammetric Engineering & Remote Sensing*, 72(4):395–403, 2006.

[188] Franz Rottensteiner, John Trinder, Simon Clode, Kurt Kubik, and Brian Lovell. Building detection by dempster-shafer fusion of lidar data and multispectral aerial imagery. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 339–342. IEEE, 2004.

[189] S Lagüela, J Martínez, J Armesto, and P Arias. Energy efficiency studies through 3d laser scanning and thermographic technologies. *Energy and Buildings*, 43(6):1216–1221, 2011.

[190] Shaohui Sun and Carl Salvaggio. Aerial 3d building detection and modeling from airborne lidar point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(3):1440–1449, 2013.

[191] Gregory P Asner, David E Knapp, Ty Kennedy-Bowdoin, Matthew O Jones, Roberta E Martin, Joseph Boardman, and R Flint Hughes. Invasive species detection in hawaiian rainforests using airborne imaging spectroscopy and lidar. *Remote Sensing of Environment*, 112(5):1942–1955, 2008.

[192] Don Leckie, François Gougeon, David Hill, Rick Quinn, Lynne Armstrong, and Roger Shreenan. Combined high-density lidar and multispectral imagery for individual tree crown analysis. *Canadian Journal of Remote Sensing*, 29(5):633–649, 2003.

[193] Yinghai Ke, Lindi J Quackenbush, and Jungho Im. Synergistic use of quickbird multispectral imagery and lidar data for object-based forest species classification. *Remote Sensing of Environment*, 114(6):1141–1154, 2010.

[194] Mohammad Awrangjeb, Chunsun Zhang, and Clive S Fraser. Automatic extraction of building roofs using lidar data and multispectral imagery. *ISPRS journal of photogrammetry and remote sensing*, 83:1–18, 2013.

[195] John C Price. Combining panchromatic and multispectral imagery from dual resolution satellite instruments. *Remote sensing of environment*, 21(2):119–128, 1987.

[196] Bruno Vallet, Marc Pierrot-Deseilligny, Didier Boldo, and Mathieu Brédif. Building footprint database improvement for 3d reconstruction: A split and merge approach and its evaluation. *ISPRS journal of photogrammetry and remote sensing*, 66(5):732–742, 2011.

[197] Florent Lafarge, Xavier Descombes, Josiane Zerubia, and Marc Pierrot-Deseilligny. Automatic building extraction from dems using an object approach and application to the 3d-city modeling. *ISPRS Journal of photogrammetry and remote sensing*, 63(3):365–381, 2008.

[198] Martin Huber, Wolfgang Schickler, Stefan Hinz, and Albert Baumgartner. Fusion of lidar data and aerial imagery for automatic reconstruction of building surfaces. In *2003 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas*, pages 82–86. IEEE, 2003.

[199] Dimitri Bulatov, Gisela Häufel, Jochen Meidow, Melanie Pohl, Peter Solbrig, and Peter Wernerus. Context-based automatic reconstruction and texturing of 3d urban terrain for quick-response tasks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 93:157–170, 2014.

[200] Jan Kybic. High-dimensional mutual information estimation for image registration. In *2004 International Conference on Image Processing, 2004. ICIP'04.*, volume 3, pages 1779–1782. IEEE, 2004.

[201] TMKG Fernando, HR Maier, and GC Dandy. Selection of input variables for data driven models: An average shifted histogram partial mutual information estimator approach. *Journal of Hydrology*, 367(3-4):165–176, 2009.

[202] Helene Sportouche, Florence Tupin, and Leonard Denise. Building extraction and 3d reconstruction in urban areas from high-resolution optical and sar imagery. In *2009 Joint Urban Remote Sensing Event*, pages 1–11. IEEE, 2009.

[203] Kenji Omasa, Fumiki Hosoi, and Atsumi Konishi. 3d lidar imaging for detecting and understanding plant responses and canopy structure. *Journal of experimental botany*, 58(4):881–898, 2006.

[204] David Frere, Jan Vandekerckhove, Theo Moons, and Luc Van Gool. Automatic modelling and 3d reconstruction of urban buildings from aerial imagery. In *IGARSS'98. Sensing and Managing the Environment. 1998 IEEE International Geoscience and Remote Sensing. Symposium Proceedings.(Cat. No. 98CH36174)*, volume 5, pages 2593–2596. IEEE, 1998.

[205] Caroline Baillard and Andrew Zisserman. Automatic reconstruction of piecewise planar models from multiple views. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 559–565. IEEE, 1999.

[206] Anestis Koutsoudis, Blaž Vidmar, George Ioannakis, Fotis Arnaoutoglou, George Pavlidis, and Christodoulos Chamzas. Multi-image 3d reconstruction data evaluation. *Journal of Cultural Heritage*, 15(1):73–79, 2014.

[207] Jay K Hackett and Mubarak Shah. Multi-sensor fusion: a perspective. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 1324–1330. IEEE, 1990.

[208] Simon Lynen, Markus W Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 3923–3929. IEEE, 2013.

[209] Ren C Luo, M-H Lin, and Ralph S Scherp. Dynamic multi-sensor data fusion system for intelligent robots. *IEEE Journal on Robotics and Automation*, 4(4):386–396, 1988.

[210] Dorota A Grejner-Brzezinska, Charles K Toth, Hongxing Sun, Xiankun Wang, and Chris Rizos. A robust solution to high-accuracy geolocation: Quadruple integration of gps, imu, pseudolite, and terrestrial laser scanning. *IEEE Transactions on instrumentation and measurement*, 60(11):3694–3708, 2011.

[211] Konstantinos Karantzalos and Nikos Paragios. Recognition-driven two-dimensional competing priors toward automatic and accurate building detection. *IEEE Transactions on Geoscience and Remote Sensing*, 47(1):133–144, 2008.

[212] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[213] Bernard Widrow and Rodney Winter. Neural nets for adaptive filtering and adaptive pattern recognition. *Computer*, 21(3):25–39, 1988.

[214] Miguel A Ferrer, Jesus B Alonso, and Carlos M Travieso. Offline geometric parameters for automatic signature verification using fixed-point arithmetic. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):993–997, 2005.

[215] Florence Tupin and Michel Roux. Detection of building outlines based on the fusion of sar and optical features. *ISPRS Journal of Photogrammetry and Remote Sensing*, 58(1-2):71–82, 2003.

[216] Yanfeng Wei, Zhongming Zhao, and Jianghong Song. Urban building extraction from high-resolution satellite panchromatic image using clustering and edge detection. In *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, volume 3, pages 2008–2010. Ieee, 2004.

[217] Jing Lin and Liangsheng Qu. Feature extraction based on morlet wavelet and its application for mechanical fault diagnosis. *Journal of sound and vibration*, 234(1):135–148, 2000.

[218] Gary G Yen and K-C Lin. Wavelet packet feature extraction for vibration monitoring. *IEEE transactions on industrial electronics*, 47(3):650–667, 2000.

[219] Kenneth E Hild, Deniz Erdogmus, and José Príncipe. Blind source separation using renyi's mutual information. *IEEE Signal Processing Letters*, 8(6):174–176, 2001.

[220] Tim R Oke. Street design and urban canopy layer climate. *Energy and buildings*, 11(1-3):103–113, 1988.

[221] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[222] Hua-mei Chen and Pramod K Varshney. Mutual information-based ct-mr brain image registration using generalized partial volume joint histogram estimation. *IEEE Transactions on medical imaging*, 22(9):1111–1119, 2003.

[223] Adhish Prasoon, Kersten Petersen, Christian Igel, François Lauze, Erik Dam, and Mads Nielsen. Deep feature learning for knee cartilage segmentation using a triplanar convolutional neural network. In *International conference on medical image computing and computer-assisted intervention*, pages 246–253. Springer, 2013.

[224] Michiel Kallenberg, Kersten Petersen, Mads Nielsen, Andrew Y Ng, Pengfei Diao, Christian Igel, Celine M Vachon, Katharina Holland, Rikke Rass Winkel, Nico Karssemeijer, et al. Unsupervised deep learning applied to breast density segmentation and mammographic risk scoring. *IEEE transactions on medical imaging*, 35(5):1322–1331, 2016.

[225] Zeynettin Akkus, Alfiia Galimzianova, Assaf Hoogi, Daniel L Rubin, and Bradley J Erickson. Deep learning for brain mri segmentation: state of the art and future directions. *Journal of digital imaging*, 30(4):449–459, 2017.

[226] MR Avendi, Arash Kheradvar, and Hamid Jafarkhani. A combined deep-learning and deformable-model approach to fully automatic segmentation of the left ventricle in cardiac mri. *Medical image analysis*, 30:108–119, 2016.

[227] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from

sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016.

[228] Kenny H Cha, Lubomir Hadjiiski, Ravi K Samala, Heang-Ping Chan, Elaine M Caoili, and Richard H Cohan. Urinary bladder segmentation in ct urography using deep-learning convolutional neural network and level sets. *Medical physics*, 43(4):1882–1896, 2016.

[229] Tommy WS Chow and Di Huang. Estimating optimal feature subsets using efficient estimation of high-dimensional mutual information. *IEEE Transactions on Neural networks*, 16(1):213–224, 2005.

[230] Shu-Li Sun and Zi-Li Deng. Multi-sensor optimal information fusion kalman filter. *Automatica*, 40(6):1017–1023, 2004.

[231] Raffaele Gravina, Parastoo Alinia, Hassan Ghasemzadeh, and Giancarlo Fortino. Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges. *Information Fusion*, 35:68–80, 2017.

[232] Otman Basir and Xiaohong Yuan. Engine fault diagnosis based on multi-sensor information fusion using dempster–shafer evidence theory. *Information Fusion*, 8(4):379–386, 2007.

[233] G Rigatos and S Tzafestas. Extended kalman filtering for fuzzy modelling and multi-sensor fusion. *Mathematical and computer modelling of dynamical systems*, 13(3):251–266, 2007.

[234] Ning Xiong and Per Svensson. Multi-sensor management for information fusion: issues and approaches. *Information fusion*, 3(2):163–186, 2002.

[235] James Llinas and David L Hall. An introduction to multi-sensor data fusion. In *ISCAS'98. Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (Cat. No. 98CH36187)*, volume 6, pages 537–540. IEEE, 1998.

[236] Chun Zhu and Weihua Sheng. Human daily activity recognition in robot-assisted living using multi-sensor fusion. In *2009 IEEE International Conference on Robotics and Automation*, pages 2154–2159. IEEE, 2009.

[237] JAP Coutinho, F Mirante, JC Ribeiro, JM Sansot, and JL Daridon. Cloud and pour points in fuel blends. *Fuel*, 81(7):963–967, 2002.

[238] Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

[239] Yves Normandin, Regis Cardin, and Renato De Mori. High-performance connected digit recognition using maximum mutual information estimation. *IEEE Transactions on speech and audio processing*, 2(2):299–311, 1994.

[240] Rudy Moddemeijer. On estimation of entropy and mutual information of continuous distributions. *Signal processing*, 16(3):233–248, 1989.

[241] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994.

[242] Mattias Nilsson, Harald Gustaftson, Søren Vang Andersen, and W Bastiaan Kleijn. Gaussian mixture model based mutual information estimation between frequency bands in speech. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–525. IEEE, 2002.

[243] Y-L Chow. Maximum mutual information estimation of hmm parameters for continuous speech recognition using the n-best algorithm. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 701–704. IEEE, 1990.

[244] Christopher Kennedy, Stephanie Pincetl, and Paul Bunje. The study of urban metabolism and its applications to urban planning and design. *Environmental pollution*, 159(8-9):1965–1973, 2011.

[245] John Whaley, Michael C Martin, and Monica S Lam. Automatic extraction of object-oriented component interfaces. In *ACM SIGSOFT Software Engineering Notes*, volume 27, pages 218–228. ACM, 2002.

[246] Wenke Lee, Salvatore J Stolfo, and Kui W Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344)*, pages 120–132. IEEE, 1999.

[247] David D Lewis. Feature selection and feature extraction for text categorization. In *Proceedings of the workshop on Speech and Natural Language*, pages 212–217. Association for Computational Linguistics, 1992.

[248] Hynek Hermansky, Daniel PW Ellis, and Sangita Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, volume 3, pages 1635–1638. IEEE, 2000.

[249] Gail A Carpenter. Neural network models for pattern recognition and associative memory. *Neural networks*, 2(4):243–257, 1989.

[250] Kunihiko Fukushima. A neural network for visual pattern recognition. *Computer*, (3):65–75, 1988.

[251] Teuvo Kohonen, Gyorgy Barna, and Ronald Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *IEEE International Conference on Neural Networks*, volume 1, pages 61–68, 1988.

[252] Robert J Schalkoff. Pattern recognition. *Wiley Encyclopedia of Computer Science and Engineering*, 2007.

[253] Ken-ichi Kamijo and Tetsuji Tanigawa. Stock price pattern recognition-a recurrent neural network approach. In *1990 IJCNN International Joint Conference on Neural Networks*, pages 215–221. IEEE, 1990.

[254] M Berthod, L Gabet, G Giraudon, and JL Lotti. High-resolution stereo for the detection of buildings. In *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, pages 135–144. Springer, 1995.

[255] Salar Ghaffarian. Automatic building detection based on supervised classification using high resolution google earth images. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(3):101, 2014.

[256] Ali Ozgun Ok. Automated extraction of buildings and roads in a graph partitioning framework. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:79–84, 2013.

[257] Chuiqing Zeng, Jinfei Wang, Wenfeng Zhan, Peijun Shi, and Autumn Gambles. An elevation difference model for building height extraction from stereo-image-derived dsms. *International journal of remote sensing*, 35(22):7614–7630, 2014.

[258] Yanming Chen, Liang Cheng, Manchun Li, Jiechen Wang, Lihua Tong, and Kang Yang. Multiscale grid method for detection and reconstruction of building roofs from airborne lidar data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(10):4081–4094, 2014.

[259] Xiaoying Jin and Curt H Davis. Automated building extraction from high-resolution satellite imagery in urban areas using structural, contextual, and spectral information. *EURASIP Journal on Advances in Signal Processing*, 2005(14):745309, 2005.

[260] Gunho Sohn and Ian Dowman. Data fusion of high-resolution satellite imagery and lidar data for automatic building extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(1):43–63, 2007.

[261] E Grilli, F Menna, and F Remondino. A review of point clouds segmentation and classification algorithms. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:339, 2017.

[262] Bailang Yu, Hongxing Liu, Jianping Wu, Yingjie Hu, and Li Zhang. Automated derivation of urban building density information using airborne lidar data and object-based method. *Landscape and Urban Planning*, 98(3-4):210–219, 2010.

[263] Ansgar Brunn, E Gulch, F Lang, and W Forstner. A multi-layer strategy for 3d building acquisition. In *in Proceedings IAPR-TC7 Workshop*. Citeseer, 1996.

[264] Shraddha Singhal and Sudha Radhika. Automatic detection of buildings from aerial images using color invariant features and canny edge detection. *International Journal of Engineering Trends and Technology (IJETT)*, 11(8):393–396, 2014.

[265] Caroline Baillard and Andrew Zisserman. A plane-sweep strategy for the 3d reconstruction of buildings from multiple images. *International Archives of Photogrammetry and Remote Sensing*, 33(B2; PART 2):56–62, 2000.

[266] M Cord, N Paparoditis, and M Jordan. Dense, reliable and depth discontinuity preserving dem computation from hrv urban stereopairs. *International Archives of Photogrammetry and Remote Sensing*, 32:49–56, 1997.

[267] Wolfgang Eckstein and Olaf Muenkelt. Extracting objects from digital terrain models. In *Remote sensing and reconstruction for three-dimensional objects and scenes*, volume 2572, pages 43–51. International Society for Optics and Photonics, 1995.

[268] Makoto Nagao and Takashi Matsuyama. Edge preserving smoothing. *Computer graphics and image processing*, 9(4):394–407, 1979.

[269] Antonis Katartzis and Hichem Sahli. A stochastic framework for the identification of building rooftops using a single remote sensing image. *IEEE Transactions on Geoscience and Remote Sensing*, 46(1):259–271, 2007.

[270] Stephane Girard, Philippe Guerin, Henri Maitre, and Michel Roux. Building detection from high-resolution color images. In *Image and Signal Processing for Remote Sensing IV*, volume 3500, pages 278–289. International Society for Optics and Photonics, 1998.

[271] Dong Chen, Liqiang Zhang, P Takis Mathiopoulos, and Xianfeng Huang. A methodology for automated segmentation and reconstruction of urban 3-d buildings from als point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(10):4199–4217, 2014.

[272] Eberhard Gülch. Application of semi-automatic building acquisition. In *Automatic extraction of man-made objects from aerial and space images (II)*, pages 129–138. Springer, 1997.

[273] Christoph Hug. Extracting artificial surface objects from airborne laser scanner data. In *Automatic extraction of man-made objects from aerial and space images (II)*, pages 203–212. Springer, 1997.

[274] Florent Lafarge, Xavier Descombes, Josiane Zerubia, and Marc Pierrot-Deseilligny. Structural approach for building reconstruction from a single dsm. *IEEE Transactions on pattern analysis and machine intelligence*, 32(1):135–147, 2008.

[275] Claus Brenner. Building reconstruction from images and laser scanning. *International Journal of Applied Earth Observation and Geoinformation*, 6(3-4):187–198, 2005.

[276] Franz Rottensteiner and Christian Briese. A new method for building extraction in urban areas from high-resolution lidar data. In *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, volume 34, pages 295–301. Natural Resources Canada, 2002.

[277] Emmanuel Baltsavias, Scott Mason, and Dirk Stallmann. Use of dtms/dsms and orthoimages to support building extraction. In *Automatic extraction of man-made objects from aerial and space images*, pages 199–210. Springer, 1995.

[278] Richard Hoffman and Anil K Jain. Segmentation and classification of range images. *IEEE transactions on pattern analysis and machine intelligence*, (5):608–620, 1987.

[279] Liang Cheng, Jianya Gong, Manchun Li, and Yongxue Liu. 3d building model reconstruction from multi-view aerial imagery and lidar data. *Photogrammetric Engineering & Remote Sensing*, 77(2):125–139, 2011.

[280] Yusheng Xu, Wei Yao, Sebastian Tuttas, Ludwig Hoegner, and Uwe Stilla. Unsupervised segmentation of point clouds from buildings using hierarchical clustering based on gestalt principles. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(11):4270–4286, 2018.

[281] Christopher Jaynes, Robert Collins, Frank Stolle, Allen Hanson, Edward Riseman, and Howard Schultz. Three-dimensional grouping for site modeling from aerial images. *Radius: Image Understanding for Imagery Intelligence*, page 223, 1997.

[282] Peng-hui Tian and Li-chun Sui. Building contours extraction from light detect and ranging data. In *2011 Symposium on Photonics and Optoelectronics (SOPO)*, pages 1–3. IEEE, 2011.

[283] Nan Su, Ye Zhang, Shu Tian, Yiming Yan, and Xinyuan Miao. Shadow detection and removal for occluded object information recovery in urban high-resolution panchromatic satellite images. *IEEE Journal of Selected*

Topics in Applied Earth Observations and Remote Sensing, 9(6):2568–2582, 2016.

[284] Y Li, L Zhu, H Shimamura, and K Tachibanab. An integrated system on large scale building extraction from dsm. *Int Arch Photogramm Remote Sens Spat Inf Sci*, 38:35–39, 2010.

[285] Yan Li, Lin Zhu, Peng Gong, and Hideki Shimamura. A refined marker controlled watershed for building extraction from dsm and imagery. *International Journal of Remote Sensing*, 31(6):1441–1452, 2010.

[286] Hanns-F Schuster. Segmentation of lidar data using the tensor voting framework. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(B3):1073–1078, 2004.

[287] Chun Liua, Beiqi Shia, Xuan Yanga, and Nan Lia. Legion segmentation for building extraction from lidar based dsm data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39:B3, 2012.

[288] Ansgar Brunn and Uwe Weidner. Extracting buildings from digital surface models. *International Archives of Photogrammetry and Remote Sensing*, 32(3 SECT 4W2):27–34, 1997.

[289] Chun Liu, Beiqi Shi, Xuan Yang, Nan Li, and Hangbin Wu. Automatic buildings extraction from lidar data in urban area by neural oscillator network of visual cortex. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(4):2008–2019, 2013.

[290] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[291] Chenxiao Zhang, Peng Yue, Deodato Tapete, Boyi Shangguan, Mi Wang, and Zhaoyan Wu. A multi-level context-guided classification method with object-based convolutional neural network for land cover classification using very high resolution remote sensing images. *International Journal of Applied Earth Observation and Geoinformation*, 88:102086, 2020.

[292] Xianwei Zheng, Linxi Huan, Gui-Song Xia, and Jianya Gong. Parsing very high resolution urban scene images by learning deep convnets with edge-aware loss. *ISPRS Journal of Photogrammetry and Remote Sensing*, 170:15–28, 2020.

[293] Yuchu Qin, Yunchao Wu, Bin Li, Shuai Gao, Miao Liu, and Yulin Zhan. Semantic segmentation of building roof in dense urban environment with deep convolutional neural network: A case study using gf2 vhr imagery in china. *Sensors*, 19(5):1164, 2019.

[294] So-Young Park, Dae Geon Lee, Eun Jin Yoo, and Dong-Cheon Lee. Segmentation of lidar data using multilevel cube code. *Journal of Sensors*, 2019, 2019.

[295] Xiangyu Zhuo, Friedrich Fraundorfer, Franz Kurz, and Peter Reinartz. Building detection and segmentation using a cnn with automatically generated training data. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 3461–3464. IEEE, 2018.

[296] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European conference on computer vision*, pages 354–370. Springer, 2016.

[297] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[298] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[299] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[300] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[301] Quang-Hieu Pham, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. Jsis3d: Joint semantic-instance segmentation of 3d point clouds

with multi-task pointwise networks and multi-value conditional random fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[302] Teerapong Panboonyuen, Kulsawasd Jitkajornwanich, Siam Lawawirojwong, Panu Srestasathiern, and Peerapon Vateekul. Road segmentation of remotely-sensed images using deep convolutional neural networks with landscape metrics and conditional random fields. *Remote Sensing*, 9(7):680, 2017.

[303] Ian J Dowman, Karsten Jacobsen, Gottfried Konecny, and Rainer Sandau. *High resolution optical satellite imagery*. Whittles Publishing Scotland, UK, 2012.

[304] John Duncan, Jadunandan Dash, and Peter M Atkinson. The potential of satellite-observed crop phenology to enhance yield gap assessments in smallholder landscapes. *Frontiers in Environmental Science*, 3:56, 2015.

[305] Sergio M Vicente-Serrano, X Pons-Fernández, and JM Cuadrat-Prats. Mapping soil moisture in the central ebro river valley (northeast spain) with landsat and noaa satellite imagery: a comparison with meteorological data. *International Journal of Remote Sensing*, 25(20):4325–4350, 2004.

[306] Clement Atzberger. Advances in remote sensing of agriculture: Context description, existing operational monitoring systems and major information needs. *Remote sensing*, 5(2):949–981, 2013.

[307] Huang Yao, Rongjun Qin, and Xiaoyu Chen. Unmanned aerial vehicle for remote sensing applications—a review. *Remote Sensing*, 11(12):1443, 2019.

[308] James B Campbell and Randolph H Wynne. *Introduction to remote sensing*. Guilford Press, 2011.

[309] Yannick Rio, M Salome Rodriguez-Morgade, and Tomas Torres. Modulating the electronic properties of porphyrinoids: a voyage from the violet to the infrared regions of the electromagnetic spectrum. *Organic & biomolecular chemistry*, 6(11):1877–1894, 2008.

[310] Soe W Myint, Patricia Gober, Anthony Brazel, Susanne Grossman-Clarke, and Qihao Weng. Per-pixel vs. object-based classification of urban land

cover extraction using high spatial resolution imagery. *Remote sensing of environment*, 115(5):1145–1161, 2011.

[311] Paul M Dare. Shadow analysis in high-resolution satellite imagery of urban areas. *Photogrammetric Engineering & Remote Sensing*, 71(2):169–177, 2005.

[312] Norbert Haala and Claus Brenner. Extraction of buildings and trees in urban environments. *ISPRS journal of photogrammetry and remote sensing*, 54(2-3):130–137, 1999.

[313] Paul VC Hough. Method and means for recognizing complex patterns, December 18 1962. US Patent 3,069,654.

[314] Rajiv Kumar, M Arthanari, and M Sivakumar. Image segmentation using discontinuity-based approach. *Int. J. Multimedia Image Process*, 1:72–78, 2011.

[315] George Stockman. Object recognition and localization via pose clustering. *Computer vision, graphics, and image processing*, 40(3):361–387, 1987.

[316] Chi-ho Chan and Grantham KH Pang. Fabric defect detection by fourier analysis. *IEEE transactions on Industry Applications*, 36(5):1267–1276, 2000.

[317] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[318] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[319] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[320] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[321] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[322] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[323] Avinash Hindupur. Avinash: Cumulative numbers of published papers (gans), 2021.

[324] Xuran Pan, Fan Yang, Lianru Gao, Zhengchao Chen, Bing Zhang, Hairui Fan, and Jinchang Ren. Building extraction from high-resolution aerial imagery using a generative adversarial network with spatial and channel attention mechanisms. *Remote Sensing*, 11(8):917, 2019.

[325] Kaixuan Zhou, Y Chen, Ihor Smal, and Roderik Lindenbergh. Building segmentation from airborne vhr images using mask r-cnn. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2019.

[326] Yanning Zhou, Omer Fahri Onder, Qi Dou, Efstratios Tsougenis, Hao Chen, and Pheng-Ann Heng. Cia-net: Robust nuclei instance segmentation with contour-aware information aggregation. In *International Conference on Information Processing in Medical Imaging*, pages 682–693. Springer, 2019.

[327] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[328] Yanjun Wang, Qi Chen, Qing Zhu, Lin Liu, Chaokui Li, and Dunyong Zheng. A survey of mobile laser scanning applications and key techniques over urban areas. *Remote Sensing*, 11(13):1540, 2019.

[329] Devis Tuia, Michele Volpi, Loris Copa, Mikhail Kanevski, and Jordi Munoz-Mari. A survey of active learning algorithms for supervised remote sensing

image classification. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):606–617, 2011.

[330] Yanbiao Sun, Liang Zhao, Shoudong Huang, Lei Yan, and Gamini Dissanayake. L2-sift: Sift feature extraction and matching for large images in large-scale aerial photogrammetry. *ISPRS Journal of Photogrammetry and Remote Sensing*, 91:1–16, 2014.

[331] Cong Chen, Guohui Zhang, Zhen Qian, Rafiqul A Tarefder, and Zong Tian. Investigating driver injury severity patterns in rollover crashes using support vector machine models. *Accident Analysis & Prevention*, 90:128–139, 2016.

[332] Thomas J Pingel, Keith C Clarke, and William A McBride. An improved simple morphological filter for the terrain classification of airborne lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 77:21–30, 2013.

[333] Selçuk Reis and Kadim Taşdemir. Identification of hazelnut fields using spectral and gabor textural features. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(5):652–661, 2011.

[334] Gong Cheng, Junwei Han, Lei Guo, Xiaoliang Qian, Peicheng Zhou, Xiwen Yao, and Xintao Hu. Object detection in remote sensing imagery using a discriminatively trained mixture model. *ISPRS Journal of Photogrammetry and Remote Sensing*, 85:32–43, 2013.

[335] Shixia Liu, Xiting Wang, Mengchen Liu, and Jun Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 1(1):48–56, 2017.

[336] Peter M Atkinson and Adrian RL Tatnall. Introduction neural networks in remote sensing. *International Journal of remote sensing*, 18(4):699–709, 1997.

[337] Lizhe Tan and Jean Jiang. *Digital signal processing: fundamentals and applications*. Academic Press, 2018.

[338] Sakrapee Paisitkriangkrai, Jamie Sherrah, Pranam Janney, and Anton Van Den Hengel. Semantic labeling of aerial and satellite imagery. *IEEE Jour-*

nal of Selected Topics in Applied Earth Observations and Remote Sensing, 9(7):2868–2881, 2016.

[339] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[340] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[341] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[342] Selim Aksoy, Ismet Zeki Yalniz, and Kadim Tasdemir. Automatic detection and segmentation of orchards using very high resolution imagery. *IEEE Transactions on geoscience and remote sensing*, 50(8):3117–3131, 2012.

[343] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[344] Aaron E Maxwell, Timothy A Warner, and Fang Fang. Implementation of machine-learning classification in remote sensing: An applied review. *International Journal of Remote Sensing*, 39(9):2784–2817, 2018.

Appendix

A. Software Setup

To progress with the computational part of the problem, different open source frameworks are used because they provide the necessary functions and modules that are to be used in this project. These software, along with a brief description, are listed below:

- Python
- OpenCV
- NumPy
- SciPy
- Matplotlib
- PyTorch
- Detectron2

Python

Python is an interpreted, high level programming language that allows convenient access to general programming practices and approaches. The language features rich set of libraries and frameworks, that allows easier implementation of the ideas and solution in terms of image processing and computer vision.

OpenCV

OpenCV is a popular library for computer vision and image processing, which features a wide set of algorithms that are primarily used in the state-of-the-art technologies. The algorithms included within OpenCV allows quick and real time processing of the images.

NumPy

NumPy is a numerical Python library that allows matrices and a high-dimensional array top be created, along with set of operations that can be performed on these data structures.

SciPy

SciPy is a scientific Python library that features different computational functions across several scientific domains. These functions are based on the solutions in ODEs, Linear Algebra, optimisation, integration, signal analysis, image processing and so on. SciPy inter-operates with Numpy.

Matplotlib

Matplotlib is a plotting library in Python that allows graphs and figures to be created for different data sources. It allows easy operation with NumPy and SciPy. Pytorch is an optimized tensor processing library for deep learning using CPUs and GPUs. The library allows inter-operation with Python syntax. It also allows automatic differentiation, meaning that it allows the gradients to be calculated on the fly without having to worry about backward calculation. Pytorch also seamlessly operates with NumPy.

B. *pix2pix* GAN Model [341] & *Detectron2* [340] code

section{pix2pix-code}

```
from __future__ import absolute_import

from __future__ import division

from __future__ import print_function

import tensorflow as tf

print(tf.__version__)

import numpy as np

import argparse

import os

import json

import glob

import random

import collections

import math

import time

parser = argparse.ArgumentParser()

parser.add_argument("--input_dir", default= "maps/train" , help="path to folder containing
images")

parser.add_argument("--mode",  default= "train" ,choices=["train", "test", "export"])

parser.add_argument("--output_dir", default= "facades_train" ,help="where to put output
files")

parser.add_argument("--seed", type=int)

parser.add_argument("--checkpoint", default=None, help="directory with checkpoint to
resume training from or use for testing")
```

```
parser.add_argument("--max_steps", type=int, help="number of training steps (0 to disable)")

parser.add_argument("--max_epochs", type=int, help="number of training epochs")

parser.add_argument("--summary_freq", type=int, default=100, help="update summaries
every summary_freq steps")

parser.add_argument("--progress_freq", type=int, default=50, help="display progress every
progress_freq steps")

parser.add_argument("--trace_freq", type=int, default=0, help="trace execution every
trace_freq steps")

parser.add_argument("--display_freq", type=int, default=0, help="write current training
images every display_freq steps")

parser.add_argument("--save_freq", type=int, default=5000, help="save model every
save_freq steps, 0 to disable")

parser.add_argument("--separable_conv", action="store_true", help="use separable
convolutions in the generator")

parser.add_argument("--aspect_ratio", type=float, default=1.0, help="aspect ratio of output
images (width/height)")

parser.add_argument("--lab_colorization", action="store_true", help="split input image into
brightness (A) and color (B)")

parser.add_argument("--batch_size", type=int, default=1, help="number of images in batch")

parser.add_argument("--which_direction", type=str, default="AtoB", choices=["AtoB",
"BtoA"])

parser.add_argument("--ngf", type=int, default=64, help="number of generator filters in first
conv layer")

parser.add_argument("--ndf", type=int, default=64, help="number of discriminator filters in
first conv layer")

parser.add_argument("--scale_size", type=int, default=286, help="scale images to this size
before cropping to 256x256")
```

```
parser.add_argument("--flip", dest="flip", action="store_true", help="flip images
horizontally")

parser.add_argument("--no_flip", dest="flip", action="store_false", help="don't flip images
horizontally")

parser.set_defaults(flip=True)

parser.add_argument("--lr", type=float, default=0.0002, help="initial learning rate for adam")

parser.add_argument("--beta1", type=float, default=0.5, help="momentum term of adam")

parser.add_argument("--l1_weight", type=float, default=100.0, help="weight on L1 term for
generator gradient")

parser.add_argument("--gan_weight", type=float, default=1.0, help="weight on GAN term
for generator gradient")



# export options

parser.add_argument("--output_filetype", default="png", choices=["png", "jpeg"])

a = parser.parse_args()


EPS = 1e-12

CROP_SIZE = 256


Examples = collections.namedtuple("Examples", "paths, inputs, targets, count,
steps_per_epoch")

Model = collections.namedtuple("Model", "outputs, predict_real, predict_fake, discrim_loss,
discrim_grads_and_vars, gen_loss_GAN, gen_loss_L1, gen_grads_and_vars, train")

def preprocess(image):

   with tf.name_scope("preprocess"):
```

```
        # [0, 1] => [-1, 1]
        return image * 2 - 1
def deprocess(image):
    with tf.name_scope("deprocess"):
        # [-1, 1] => [0, 1]
        return (image + 1) / 2
def preprocess_lab(lab):
    with tf.name_scope("preprocess_lab"):
        L_chan, a_chan, b_chan = tf.unstack(lab, axis=2)
        # L_chan: black and white with input range [0, 100]
        # a_chan/b_chan: color channels with input range ~[-110, 110], not exact
        # [0, 100] => [-1, 1],  ~[-110, 110] => [-1, 1]
        return [L_chan / 50 - 1, a_chan / 110, b_chan / 110]
def deprocess_lab(L_chan, a_chan, b_chan):
    with tf.name_scope("deprocess_lab"):
        # this is axis=3 instead of axis=2 because we process individual images but deprocess
batches
        return tf.stack([(L_chan + 1) / 2 * 100, a_chan * 110, b_chan * 110], axis=3)
```

```
def augment(image, brightness):
    # (a, b) color channels, combine with L channel and convert to rgb
    a_chan, b_chan = tf.unstack(image, axis=3)
```

```
    L_chan = tf.squeeze(brightness, axis=3)
    lab = deprocess_lab(L_chan, a_chan, b_chan)
    rgb = lab_to_rgb(lab)
    return rgb


def discrim_conv(batch_input, out_channels, stride):
    padded_input = tf.pad(batch_input, [[0, 0], [1, 1], [1, 1], [0, 0]], mode="CONSTANT")
    return tf.layers.conv2d(padded_input, out_channels, kernel_size=4, strides=(stride, stride),
padding="valid", kernel_initializer=tf.random_normal_initializer(0, 0.02))


def gen_conv(batch_input, out_channels):
    # [batch, in_height, in_width, in_channels] => [batch, out_height, out_width,
out_channels]
    initializer = tf.random_normal_initializer(0, 0.02)
    if a.separable_conv:
        return tf.layers.separable_conv2d(batch_input, out_channels, kernel_size=4, strides=(2,
2), padding="same", depthwise_initializer=initializer, pointwise_initializer=initializer)
    else:
        return tf.layers.conv2d(batch_input, out_channels, kernel_size=4, strides=(2, 2),
padding="same", kernel_initializer=initializer)


def gen_deconv(batch_input, out_channels):
    # [batch, in_height, in_width, in_channels] => [batch, out_height, out_width,
out_channels]
    initializer = tf.random_normal_initializer(0, 0.02)
    if a.separable_conv:
```

```
        _b, h, w, _c = batch_input.shape
        resized_input = tf.image.resize_images(batch_input, [h * 2, w * 2],
method=tf.image.ResizeMethod.NEAREST_NEIGHBOR)
        return tf.layers.separable_conv2d(resized_input, out_channels, kernel_size=4,
strides=(1, 1), padding="same", depthwise_initializer=initializer,
pointwise_initializer=initializer)
    else:
        return tf.layers.conv2d_transpose(batch_input, out_channels, kernel_size=4, strides=(2,
2), padding="same", kernel_initializer=initializer)


def lrelu(x, a):
    with tf.name_scope("lrelu"):
        # adding these together creates the leak part and linear part
        # then cancels them out by subtracting/adding an absolute value term
        # leak: a*x/2 - a*abs(x)/2
        # linear: x/2 + abs(x)/2
        # this block looks like it has 2 inputs on the graph unless we do this
        x = tf.identity(x)
        return (0.5 * (1 + a)) * x + (0.5 * (1 - a)) * tf.abs(x)


def batchnorm(inputs):
    return tf.layers.batch_normalization(inputs, axis=3, epsilon=1e-5, momentum=0.1,
training=True, gamma_initializer=tf.random_normal_initializer(1.0, 0.02))


def check_image(image):
```

```
    assertion = tf.assert_equal(tf.shape(image)[-1], 3, message="image must have 3 color
channels")
    with tf.control_dependencies([assertion]):
        image = tf.identity(image)

    if image.get_shape().ndims not in (3, 4):
        raise ValueError("image must be either 3 or 4 dimensions")



    # make the last dimension 3 so that you can unstack the colors
    shape = list(image.get_shape())
    shape[-1] = 3
    image.set_shape(shape)
    return image


# based on
https://github.com/torch/image/blob/9f65c30167b2048ecbe8b7befdc6b2d6d12baee9/generic/
image.c
def rgb_to_lab(srgb):
    with tf.name_scope("rgb_to_lab"):
        srgb = check_image(srgb)
        srgb_pixels = tf.reshape(srgb, [-1, 3])

        with tf.name_scope("srgb_to_xyz"):
            linear_mask = tf.cast(srgb_pixels <= 0.04045, dtype=tf.float32)
```

```
        exponential_mask = tf.cast(srgb_pixels > 0.04045, dtype=tf.float32)
        rgb_pixels = (srgb_pixels / 12.92 * linear_mask) + (((srgb_pixels + 0.055) / 1.055) **
2.4) * exponential_mask
        rgb_to_xyz = tf.constant([
          #    X        Y          Z
          [0.412453, 0.212671, 0.019334], # R
          [0.357580, 0.715160, 0.119193], # G
          [0.180423, 0.072169, 0.950227], # B
        ])
        xyz_pixels = tf.matmul(rgb_pixels, rgb_to_xyz)
      # https://en.wikipedia.org/wiki/Lab_color_space#CIELAB-CIEXYZ_conversions
      with tf.name_scope("xyz_to_cielab"):
        # convert to fx = f(X/Xn), fy = f(Y/Yn), fz = f(Z/Zn)
        # normalize for D65 white point
        xyz_normalized_pixels = tf.multiply(xyz_pixels, [1/0.950456, 1.0, 1/1.088754])
        epsilon = 6/29
        linear_mask = tf.cast(xyz_normalized_pixels <= (epsilon**3), dtype=tf.float32)
        exponential_mask = tf.cast(xyz_normalized_pixels > (epsilon**3), dtype=tf.float32)
        fxfyfz_pixels = (xyz_normalized_pixels / (3 * epsilon**2) + 4/29) * linear_mask +
(xyz_normalized_pixels ** (1/3)) * exponential_mask
        # convert to lab
```

```
        fxfyfz_to_lab = tf.constant([
            #  l       a       b
            [  0.0,  500.0,    0.0], # fx
            [116.0, -500.0,  200.0], # fy
            [  0.0,    0.0, -200.0], # fz
        ])
        lab_pixels = tf.matmul(fxfyfz_pixels, fxfyfz_to_lab) + tf.constant([-16.0, 0.0, 0.0])

    return tf.reshape(lab_pixels, tf.shape(srgb))


def lab_to_rgb(lab):
    with tf.name_scope("lab_to_rgb"):
        lab = check_image(lab)
        lab_pixels = tf.reshape(lab, [-1, 3])

        # https://en.wikipedia.org/wiki/Lab_color_space#CIELAB-CIEXYZ_conversions
        with tf.name_scope("cielab_to_xyz"):
            # convert to fxfyfz
            lab_to_fxfyfz = tf.constant([
                #   fx      fy        fz
                [1/116.0, 1/116.0,  1/116.0], # l
                [1/500.0,     0.0,      0.0], # a
                [    0.0,     0.0, -1/200.0], # b
```

```
        ])
        fxfyfz_pixels = tf.matmul(lab_pixels + tf.constant([16.0, 0.0, 0.0]), lab_to_fxfyfz)
        # convert to xyz
        epsilon = 6/29
        linear_mask = tf.cast(fxfyfz_pixels <= epsilon, dtype=tf.float32)
        exponential_mask = tf.cast(fxfyfz_pixels > epsilon, dtype=tf.float32)
        xyz_pixels = (3 * epsilon**2 * (fxfyfz_pixels - 4/29)) * linear_mask + (fxfyfz_pixels
** 3) * exponential_mask



        # denormalize for D65 white point
        xyz_pixels = tf.multiply(xyz_pixels, [0.950456, 1.0, 1.088754])



    with tf.name_scope("xyz_to_srgb"):
        xyz_to_rgb = tf.constant([
            #     r           g          b
            [ 3.2404542, -0.9692660,  0.0556434], # x
            [-1.5371385,  1.8760108, -0.2040259], # y
            [-0.4985314,  0.0415560,  1.0572252], # z
        ])
        rgb_pixels = tf.matmul(xyz_pixels, xyz_to_rgb)
        # avoid a slightly negative number messing up the conversion
        rgb_pixels = tf.clip_by_value(rgb_pixels, 0.0, 1.0)
        linear_mask = tf.cast(rgb_pixels <= 0.0031308, dtype=tf.float32)
```

```
        exponential_mask = tf.cast(rgb_pixels > 0.0031308, dtype=tf.float32)
        srgb_pixels = (rgb_pixels * 12.92 * linear_mask) + ((rgb_pixels ** (1/2.4) * 1.055) -
0.055) * exponential_mask


    return tf.reshape(srgb_pixels, tf.shape(lab))


def load_examples():
  if a.input_dir is None or not os.path.exists(a.input_dir):
    raise Exception("input_dir does not exist")


  input_paths = glob.glob(os.path.join(a.input_dir, "*.jpg"))
  decode = tf.image.decode_jpeg
  if len(input_paths) == 0:
    input_paths = glob.glob(os.path.join(a.input_dir, "*.png"))
    decode = tf.image.decode_png


  if len(input_paths) == 0:
    raise Exception("input_dir contains no image files")


  def get_name(path):
    name, _ = os.path.splitext(os.path.basename(path))
    return name


  # if the image names are numbers, sort by the value rather than asciibetically
```

```
    # having sorted inputs means that the outputs are sorted in test mode
    if all(get_name(path).isdigit() for path in input_paths):
        input_paths = sorted(input_paths, key=lambda path: int(get_name(path)))
    else:
        input_paths = sorted(input_paths)

    with tf.name_scope("load_images"):
        path_queue = tf.train.string_input_producer(input_paths, shuffle=a.mode == "train")
        reader = tf.WholeFileReader()
        paths, contents = reader.read(path_queue)
        raw_input = decode(contents)
        raw_input = tf.image.convert_image_dtype(raw_input, dtype=tf.float32)


        assertion = tf.assert_equal(tf.shape(raw_input)[2], 3, message="image does not have 3
channels")
        with tf.control_dependencies([assertion]):
            raw_input = tf.identity(raw_input)


        raw_input.set_shape([None, None, 3])


        if a.lab_colorization:
            # load color and brightness from image, no B image exists here
```

```
        lab = rgb_to_lab(raw_input)
        L_chan, a_chan, b_chan = preprocess_lab(lab)
        a_images = tf.expand_dims(L_chan, axis=2)
        b_images = tf.stack([a_chan, b_chan], axis=2)
    else:
        # break apart image pair and move to range [-1, 1]
        width = tf.shape(raw_input)[1] # [height, width, channels]
        a_images = preprocess(raw_input[:,:width//2,:])
        b_images = preprocess(raw_input[:,width//2:,:])


if a.which_direction == "AtoB":
    inputs, targets = [a_images, b_images]
elif a.which_direction == "BtoA":
    inputs, targets = [b_images, a_images]
else:
    raise Exception("invalid direction")


# synchronize seed for image operations so that we do the same operations to both
# input and output images
seed = random.randint(0, 2**31 - 1)
def transform(image):
    r = image
```

```
        if a.flip:
            r = tf.image.random_flip_left_right(r, seed=seed)



        # area produces a nice downscaling, but does nearest neighbor for upscaling
        # assume we're going to be doing downscaling here
        r = tf.image.resize_images(r, [a.scale_size, a.scale_size],
method=tf.image.ResizeMethod.AREA)



        offset = tf.cast(tf.floor(tf.random_uniform([2], 0, a.scale_size - CROP_SIZE + 1,
seed=seed)), dtype=tf.int32)
        if a.scale_size > CROP_SIZE:
            r = tf.image.crop_to_bounding_box(r, offset[0], offset[1], CROP_SIZE,
CROP_SIZE)
        elif a.scale_size < CROP_SIZE:
            raise Exception("scale size cannot be less than crop size")
        return r
    with tf.name_scope("input_images"):
        input_images = transform(inputs)



    with tf.name_scope("target_images"):
        target_images = transform(targets)
```

```
    paths_batch, inputs_batch, targets_batch = tf.train.batch([paths, input_images,
target_images], batch_size=a.batch_size)
    steps_per_epoch = int(math.ceil(len(input_paths) / a.batch_size))

    return Examples(
        paths=paths_batch,
        inputs=inputs_batch,
        targets=targets_batch,
        count=len(input_paths),
        steps_per_epoch=steps_per_epoch,
    )


def create_generator(generator_inputs, generator_outputs_channels):
    layers = [ ]

    # encoder_1: [batch, 256, 256, in_channels] => [batch, 128, 128, ngf]
    with tf.variable_scope("encoder_1"):
        output = gen_conv(generator_inputs, a.ngf)
        layers.append(output)

    layer_specs = [
        a.ngf * 2, # encoder_2: [batch, 128, 128, ngf] => [batch, 64, 64, ngf * 2]
        a.ngf * 4, # encoder_3: [batch, 64, 64, ngf * 2] => [batch, 32, 32, ngf * 4]
        a.ngf * 8, # encoder_4: [batch, 32, 32, ngf * 4] => [batch, 16, 16, ngf * 8]
```

```
        a.ngf * 8, # encoder_5: [batch, 16, 16, ngf * 8] => [batch, 8, 8, ngf * 8]
        a.ngf * 8, # encoder_6: [batch, 8, 8, ngf * 8] => [batch, 4, 4, ngf * 8]
        a.ngf * 8, # encoder_7: [batch, 4, 4, ngf * 8] => [batch, 2, 2, ngf * 8]
        a.ngf * 8, # encoder_8: [batch, 2, 2, ngf * 8] => [batch, 1, 1, ngf * 8]
    ]

    for out_channels in layer_specs:
        with tf.variable_scope("encoder_%d" % (len(layers) + 1)):
            rectified = lrelu(layers[-1], 0.2)
            # [batch, in_height, in_width, in_channels] => [batch, in_height/2, in_width/2,
out_channels]
            convolved = gen_conv(rectified, out_channels)
            output = batchnorm(convolved)
            layers.append(output)

    layer_specs = [
        (a.ngf * 8, 0.5),   # decoder_8: [batch, 1, 1, ngf * 8] => [batch, 2, 2, ngf * 8 * 2]
        (a.ngf * 8, 0.5),   # decoder_7: [batch, 2, 2, ngf * 8 * 2] => [batch, 4, 4, ngf * 8 * 2]
        (a.ngf * 8, 0.5),   # decoder_6: [batch, 4, 4, ngf * 8 * 2] => [batch, 8, 8, ngf * 8 * 2]
        (a.ngf * 8, 0.0),   # decoder_5: [batch, 8, 8, ngf * 8 * 2] => [batch, 16, 16, ngf * 8 * 2]
        (a.ngf * 4, 0.0),   # decoder_4: [batch, 16, 16, ngf * 8 * 2] => [batch, 32, 32, ngf * 4 * 2]
        (a.ngf * 2, 0.0),   # decoder_3: [batch, 32, 32, ngf * 4 * 2] => [batch, 64, 64, ngf * 2 * 2]
```

```
        (a.ngf, 0.0),       # decoder_2: [batch, 64, 64, ngf * 2 * 2] => [batch, 128, 128, ngf * 2]
    ]



    num_encoder_layers = len(layers)
    for decoder_layer, (out_channels, dropout) in enumerate(layer_specs):
        skip_layer = num_encoder_layers - decoder_layer - 1
        with tf.variable_scope("decoder_%d" % (skip_layer + 1)):
            if decoder_layer == 0:
                # first decoder layer doesn't have skip connections
                # since it is directly connected to the skip_layer
                input = layers[-1]
            else:
                input = tf.concat([layers[-1], layers[skip_layer]], axis=3)



            rectified = tf.nn.relu(input)
            # [batch, in_height, in_width, in_channels] => [batch, in_height*2, in_width*2,
out_channels]
            output = gen_deconv(rectified, out_channels)
            output = batchnorm(output)
```

```
            if dropout > 0.0:
                output = tf.nn.dropout(output, keep_prob=1 - dropout)

            layers.append(output)

    # decoder_1: [batch, 128, 128, ngf * 2] => [batch, 256, 256, generator_outputs_channels]
    with tf.variable_scope("decoder_1"):
        input = tf.concat([layers[-1], layers[0]], axis=3)
        rectified = tf.nn.relu(input)
        output = gen_deconv(rectified, generator_outputs_channels)
        output = tf.tanh(output)
        layers.append(output)

    return layers[-1]

def create_model(inputs, targets):
```

```
def create_discriminator(discrim_inputs, discrim_targets):
    n_layers = 3
    layers = []



    # 2x [batch, height, width, in_channels] => [batch, height, width, in_channels * 2]
    input = tf.concat([discrim_inputs, discrim_targets], axis=3)



    # layer_1: [batch, 256, 256, in_channels * 2] => [batch, 128, 128, ndf]
    with tf.variable_scope("layer_1"):
        convolved = discrim_conv(input, a.ndf, stride=2)
        rectified = lrelu(convolved, 0.2)
        layers.append(rectified)



    # layer_2: [batch, 128, 128, ndf] => [batch, 64, 64, ndf * 2]
    # layer_3: [batch, 64, 64, ndf * 2] => [batch, 32, 32, ndf * 4]
    # layer_4: [batch, 32, 32, ndf * 4] => [batch, 31, 31, ndf * 8]
    for i in range(n_layers):
        with tf.variable_scope("layer_%d" % (len(layers) + 1)):
            out_channels = a.ndf * min(2**(i+1), 8)
            stride = 1 if i == n_layers - 1 else 2  # last layer here has stride 1
            convolved = discrim_conv(layers[-1], out_channels, stride=stride)
```

```
            normalized = batchnorm(convolved)
            rectified = lrelu(normalized, 0.2)
            layers.append(rectified)


    # layer_5: [batch, 31, 31, ndf * 8] => [batch, 30, 30, 1]
    with tf.variable_scope("layer_%d" % (len(layers) + 1)):
        convolved = discrim_conv(rectified, out_channels=1, stride=1)
        output = tf.sigmoid(convolved)
        layers.append(output)


    return layers[-1]


with tf.variable_scope("generator"):
    out_channels = int(targets.get_shape()[-1])
    outputs = create_generator(inputs, out_channels)

# create two copies of discriminator, one for real pairs and one for fake pairs
# they share the same underlying variables
with tf.name_scope("real_discriminator"):
    with tf.variable_scope("discriminator"):
        # 2x [batch, height, width, channels] => [batch, 30, 30, 1]
```

```
        predict_real = create_discriminator(inputs, targets)


    with tf.name_scope("fake_discriminator"):
        with tf.variable_scope("discriminator", reuse=True):
            # 2x [batch, height, width, channels] => [batch, 30, 30, 1]
            predict_fake = create_discriminator(inputs, outputs)


    with tf.name_scope("discriminator_loss"):
        # minimizing -tf.log will try to get inputs to 1
        # predict_real => 1
        # predict_fake => 0
        discrim_loss = tf.reduce_mean(-(tf.log(predict_real + EPS) + tf.log(1 - predict_fake +
EPS)))


    with tf.name_scope("generator_loss"):
        # predict_fake => 1
        # abs(targets - outputs) => 0
        gen_loss_GAN = tf.reduce_mean(-tf.log(predict_fake + EPS))
        gen_loss_L1 = tf.reduce_mean(tf.abs(targets - outputs))
        gen_loss = gen_loss_GAN * a.gan_weight + gen_loss_L1 * a.l1_weight
```

```
    with tf.name_scope("discriminator_train"):
        discrim_tvars = [var for var in tf.trainable_variables() if
var.name.startswith("discriminator")]
        discrim_optim = tf.train.AdamOptimizer(a.lr, a.beta1)
        discrim_grads_and_vars = discrim_optim.compute_gradients(discrim_loss,
var_list=discrim_tvars)
        discrim_train = discrim_optim.apply_gradients(discrim_grads_and_vars)



    with tf.name_scope("generator_train"):
        with tf.control_dependencies([discrim_train]):
            gen_tvars = [var for var in tf.trainable_variables() if var.name.startswith("generator")]
            gen_optim = tf.train.AdamOptimizer(a.lr, a.beta1)
            gen_grads_and_vars = gen_optim.compute_gradients(gen_loss, var_list=gen_tvars)
            gen_train = gen_optim.apply_gradients(gen_grads_and_vars)



    ema = tf.train.ExponentialMovingAverage(decay=0.99)
    update_losses = ema.apply([discrim_loss, gen_loss_GAN, gen_loss_L1])



    global_step = tf.train.get_or_create_global_step()
    incr_global_step = tf.assign(global_step, global_step+1)
```

```
    return Model(
        predict_real=predict_real,
        predict_fake=predict_fake,
        discrim_loss=ema.average(discrim_loss),
        discrim_grads_and_vars=discrim_grads_and_vars,
        gen_loss_GAN=ema.average(gen_loss_GAN),
        gen_loss_L1=ema.average(gen_loss_L1),
        gen_grads_and_vars=gen_grads_and_vars,
        outputs=outputs,
        train=tf.group(update_losses, incr_global_step, gen_train),
    )




def save_images(fetches, step=None):
    image_dir = os.path.join(a.output_dir, "images")
    if not os.path.exists(image_dir):
        os.makedirs(image_dir)



    filesets = []
    for i, in_path in enumerate(fetches["paths"]):
        name, _ = os.path.splitext(os.path.basename(in_path.decode("utf8")))
```

```
        fileset = {"name": name, "step": step}
        for kind in ["inputs", "outputs", "targets"]:
            filename = name + "-" + kind + ".png"
            if step is not None:
                filename = "%08d-%s" % (step, filename)
            fileset[kind] = filename
            out_path = os.path.join(image_dir, filename)
            contents = fetches[kind][i]
            with open(out_path, "wb") as f:
                f.write(contents)
        filesets.append(fileset)
    return filesets




def append_index(filesets, step=False):
    index_path = os.path.join(a.output_dir, "index.html")



    if os.path.exists(index_path):
        index = open(index_path, "a")
    else:
        index = open(index_path, "w")
```

```
        index.write(" ")
        if step:
            index.write("  ")
        index.write(" ")


    for fileset in filesets:
        index.write("    ")


        if step:
            index.write("<td>%d</td>" % fileset["step"])
        index.write("<td>%s</td>" % fileset["name"])


        for kind in ["inputs", "outputs", "targets"]:
            index.write("<td><img src='images/%s'></td>" % fileset[kind])


        index.write("</tr>")
    return index_path
def main():
    if a.seed is None:
        a.seed = random.randint(0, 2**31 - 1)
```

```
tf.set_random_seed(a.seed)
np.random.seed(a.seed)
random.seed(a.seed)

if not os.path.exists(a.output_dir):
    os.makedirs(a.output_dir)

if a.mode == "test" or a.mode == "export":
    if a.checkpoint is None:
        raise Exception("checkpoint required for test mode")
    # load some options from the checkpoint
    options = {"which_direction", "ngf", "ndf", "lab_colorization"}
    with open(os.path.join(a.checkpoint, "options.json")) as f:
        for key, val in json.loads(f.read()).items():
            if key in options:
                print("loaded", key, "=", val)
                setattr(a, key, val)
    # disable these features in test mode
    a.scale_size = CROP_SIZE
    a.flip = False
```

```
    for k, v in a._get_kwargs():
        print(k, "=", v)


    with open(os.path.join(a.output_dir, "options.json"), "w") as f:
        f.write(json.dumps(vars(a), sort_keys=True, indent=4))


    if a.mode == "export":
        # export the generator to a meta graph that can be imported later for standalone
generation
        if a.lab_colorization:
            raise Exception("export not supported for lab_colorization")


        input = tf.placeholder(tf.string, shape=[1])
        input_data = tf.decode_base64(input[0])
        input_image = tf.image.decode_png(input_data)


        # remove alpha channel if present
        input_image = tf.cond(tf.equal(tf.shape(input_image)[2], 4), lambda: input_image[:,:,:3],
lambda: input_image)
        # convert grayscale to RGB
```

```
    input_image = tf.cond(tf.equal(tf.shape(input_image)[2], 1), lambda:
tf.image.grayscale_to_rgb(input_image), lambda: input_image)


    input_image = tf.image.convert_image_dtype(input_image, dtype=tf.float32)
    input_image.set_shape([CROP_SIZE, CROP_SIZE, 3])
    batch_input = tf.expand_dims(input_image, axis=0)


    with tf.variable_scope("generator"):
        batch_output = deprocess(create_generator(preprocess(batch_input), 3))


    output_image = tf.image.convert_image_dtype(batch_output, dtype=tf.uint8)[0]
    if a.output_filetype == "png":
        output_data = tf.image.encode_png(output_image)
    elif a.output_filetype == "jpeg":
        output_data = tf.image.encode_jpeg(output_image, quality=80)
    else:
        raise Exception("invalid filetype")
    output = tf.convert_to_tensor([tf.encode_base64(output_data)])

    key = tf.placeholder(tf.string, shape=[1])
    inputs = {
        "key": key.name,
```

```
    "input": input.name
}
tf.add_to_collection("inputs", json.dumps(inputs))
outputs = {
    "key":  tf.identity(key).name,
    "output": output.name,
}
tf.add_to_collection("outputs", json.dumps(outputs))



init_op = tf.global_variables_initializer()
restore_saver = tf.train.Saver()
export_saver = tf.train.Saver()



with tf.Session() as sess:
    sess.run(init_op)
    print("loading model from checkpoint")
    checkpoint = tf.train.latest_checkpoint(a.checkpoint)
    restore_saver.restore(sess, checkpoint)
    print("exporting model")
    export_saver.export_meta_graph(filename=os.path.join(a.output_dir, "export.meta"))
    export_saver.save(sess, os.path.join(a.output_dir, "export"), write_meta_graph=False)
```

```
    return



examples = load_examples()
print("examples count = %d" % examples.count)



# inputs and targets are [batch_size, height, width, channels]
model = create_model(examples.inputs, examples.targets)



# undo colorization splitting on images that we use for display/output
if a.lab_colorization:
    if a.which_direction == "AtoB":
        # inputs is brightness, this will be handled fine as a grayscale image
        # need to augment targets and outputs with brightness
        targets = augment(examples.targets, examples.inputs)
        outputs = augment(model.outputs, examples.inputs)
        # inputs can be deprocessed normally and handled as if they are single channel
        # grayscale images
        inputs = deprocess(examples.inputs)
    elif a.which_direction == "BtoA":
        # inputs will be color channels only, get brightness from targets
```

```
            inputs = augment(examples.inputs, examples.targets)
            targets = deprocess(examples.targets)
            outputs = deprocess(model.outputs)
        else:
            raise Exception("invalid direction")
    else:
        inputs = deprocess(examples.inputs)
        targets = deprocess(examples.targets)
        outputs = deprocess(model.outputs)

    def convert(image):
        if a.aspect_ratio != 1.0:
            # upscale to correct aspect ratio
            size = [CROP_SIZE, int(round(CROP_SIZE * a.aspect_ratio))]
            image = tf.image.resize_images(image, size=size,
method=tf.image.ResizeMethod.BICUBIC)

        return tf.image.convert_image_dtype(image, dtype=tf.uint8, saturate=True)

    # reverse any processing on images so they can be written to disk or displayed to user
    with tf.name_scope("convert_inputs"):
        converted_inputs = convert(inputs)

    with tf.name_scope("convert_targets"):
        converted_targets = convert(targets)
```

```
    with tf.name_scope("convert_outputs"):
        converted_outputs = convert(outputs)

    with tf.name_scope("encode_images"):
        display_fetches = {
            "paths": examples.paths,
            "inputs": tf.map_fn(tf.image.encode_png, converted_inputs, dtype=tf.string,
name="input_pngs"),
            "targets": tf.map_fn(tf.image.encode_png, converted_targets, dtype=tf.string,
name="target_pngs"),
            "outputs": tf.map_fn(tf.image.encode_png, converted_outputs, dtype=tf.string,
name="output_pngs"),
        }

    # summaries
    with tf.name_scope("inputs_summary"):
        tf.summary.image("inputs", converted_inputs)

    with tf.name_scope("targets_summary"):
        tf.summary.image("targets", converted_targets)
```

```
    with tf.name_scope("outputs_summary"):
        tf.summary.image("outputs", converted_outputs)


    with tf.name_scope("predict_real_summary"):
        tf.summary.image("predict_real", tf.image.convert_image_dtype(model.predict_real,
dtype=tf.uint8))


    with tf.name_scope("predict_fake_summary"):
        tf.summary.image("predict_fake", tf.image.convert_image_dtype(model.predict_fake,
dtype=tf.uint8))


    tf.summary.scalar("discriminator_loss", model.discrim_loss)
    tf.summary.scalar("generator_loss_GAN", model.gen_loss_GAN)
    tf.summary.scalar("generator_loss_L1", model.gen_loss_L1)

    for var in tf.trainable_variables():
        tf.summary.histogram(var.op.name + "/values", var)


    for grad, var in model.discrim_grads_and_vars + model.gen_grads_and_vars:
        tf.summary.histogram(var.op.name + "/gradients", grad)
```

```
    with tf.name_scope("parameter_count"):
        parameter_count = tf.reduce_sum([tf.reduce_prod(tf.shape(v)) for v in
tf.trainable_variables()])

    saver = tf.train.Saver(max_to_keep=1)

    logdir = a.output_dir if (a.trace_freq > 0 or a.summary_freq > 0) else None
    sv = tf.train.Supervisor(logdir=logdir, save_summaries_secs=0, saver=None)
    with sv.managed_session() as sess:
        print("parameter_count =", sess.run(parameter_count))

        if a.checkpoint is not None:
            print("loading model from checkpoint")
            checkpoint = tf.train.latest_checkpoint(a.checkpoint)
            saver.restore(sess, checkpoint)

        max_steps = 2**32
        if a.max_epochs is not None:
            max_steps = examples.steps_per_epoch * a.max_epochs
        if a.max_steps is not None:
            max_steps = a.max_steps

        if a.mode == "test":
            # testing
```

```
# at most, process the test data once
        start = time.time()
        max_steps = min(examples.steps_per_epoch, max_steps)
        for step in range(max_steps):
            results = sess.run(display_fetches)
            filesets = save_images(results)
            for i, f in enumerate(filesets):
                print("evaluated image", f["name"])
            index_path = append_index(filesets)
        print("wrote index at", index_path)
        print("rate", (time.time() - start) / max_steps)
    else:
        # training
        start = time.time()



        for step in range(max_steps):
            def should(freq):
                return freq > 0 and ((step + 1) % freq == 0 or step == max_steps - 1)


            options = None
```

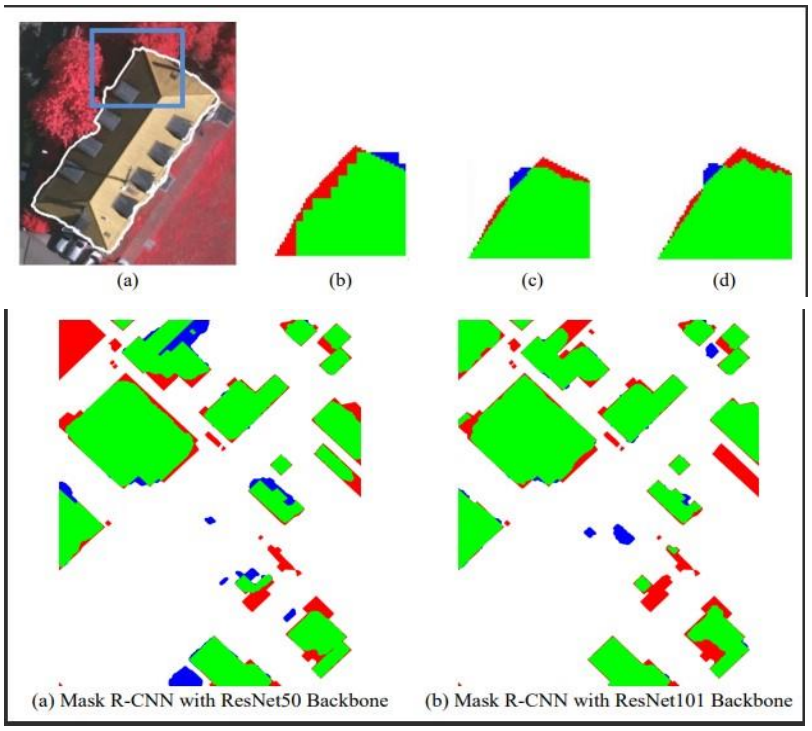
```
run_metadata = None
if should(a.trace_freq):
    options = tf.RunOptions(trace_level=tf.RunOptions.FULL_TRACE)
    run_metadata = tf.RunMetadata()

fetches = {
    "train": model.train,
    "global_step": sv.global_step,
}

if should(a.progress_freq):
    fetches["discrim_loss"] = model.discrim_loss
    fetches["gen_loss_GAN"] = model.gen_loss_GAN
    fetches["gen_loss_L1"] = model.gen_loss_L1

if should(a.summary_freq):
    fetches["summary"] = sv.summary_op

if should(a.display_freq):
    fetches["display"] = display_fetches

results = sess.run(fetches, options=options, run_metadata=run_metadata)

if should(a.summary_freq):
```

```
            print("recording summary")
            sv.summary_writer.add_summary(results["summary"], results["global_step"])
        if should(a.display_freq):
            print("saving display images")
            filesets = save_images(results["display"], step=results["global_step"])
            append_index(filesets, step=True)

        if should(a.trace_freq):
            print("recording trace")
            sv.summary_writer.add_run_metadata(run_metadata, "step_%d" %
results["global_step"])

        if should(a.progress_freq):
            # global_step will have the correct step count if we resume from a checkpoint
            train_epoch = math.ceil(results["global_step"] / examples.steps_per_epoch)
            train_step = (results["global_step"] - 1) % examples.steps_per_epoch + 1
            rate = (step + 1) * a.batch_size / (time.time() - start)
            remaining = (max_steps - step) * a.batch_size / rate
            print("progress  epoch %d  step %d  image/sec %0.1f  remaining %dm" %
(train_epoch, train_step, rate, remaining / 60))
            print("discrim_loss", results["discrim_loss"])
            print("gen_loss_GAN", results["gen_loss_GAN"])
            print("gen_loss_L1", results["gen_loss_L1"])

        if should(a.save_freq):
```

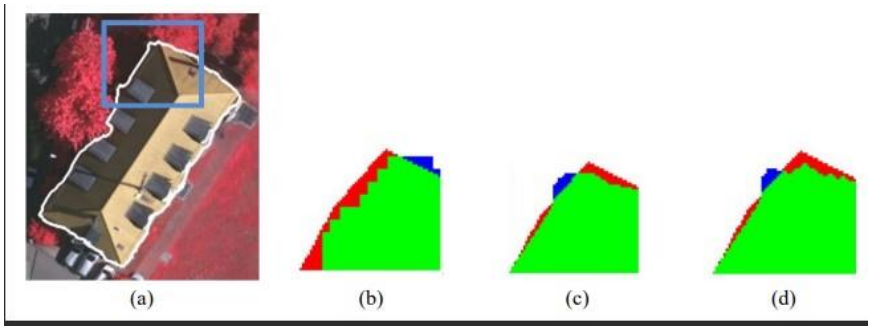
```
            print("saving model")
            saver.save(sess, os.path.join(a.output_dir, "model"), global_step=sv.global_step)

        if sv.should_stop():
            break

main()
```

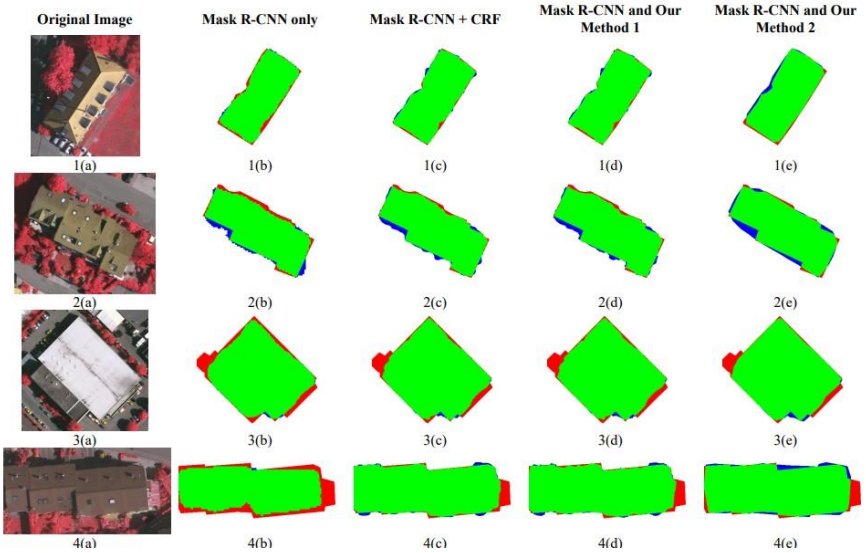
extra output Output Method 1

(a) Raw output from GAN on original image (b) Output after application of CRF



extra output Output Method 2


(a)
(b)
(c)
(d)


Original Image
Mask R-CNN only
Mask R-CNN + CRF
Mask R-CNN and Our Method 1
Mask R-CNN and Our Method 2
1(a)
1(b)
1(c)
1(d)
1(e)
2(a)
2(b)
2(c)
2(d)
2(e)
3(a)
3(b)
3(c)
3(d)
3(e)
4(a)
4(b)
4(c)
4(d)
4(e)