DOCTORAL THESIS NO. 31

Computational and Spatial Analyses of Rooftops for Urban Solar Energy Planning

Mohammad Aslani

UNIVERSITY OF GÄVLE

Gävle University Press

Dissertation for the Degree of Doctor of Philosophy in Geospatial Information Science to be publicly defended on Friday 18 November 2022 at 09:00 in Room 23213, University of Gävle.

External reviewer: Professor Alison Heppenstall, University of Glasgow

Cover illustration: Mohammad Aslani

Gävle University Press
ISBN 978-91-88145-93-2
ISBN 978-91-88145-94-9 (pdf)
urn:nbn:se:hig:diva-39741

Distribution:
University of Gävle
Faculty of Engineering and Sustainable Development
Department of Computer and Geospatial Sciences
SE-801 76 Gävle, Sweden
+46 26 64 85 00
www.hig.se

When nothing seems to help, I go and look at a stonecutter hammering away at his rock perhaps a hundred times without as much as a crack showing in it. Yet at the hundred and first blow it will split in two, and I know that it was not that blow that did it, but all that had gone before.

Jacob Riis

To my parents

Without them, none of my success would be possible.

Abstract

In cities where land availability is limited, rooftop photovoltaic panels (RPVs) offer high potential for satisfying concentrated urban energy demand by using only rooftop areas. However, accurate estimation of RPVs potential in relation to their spatial distribution is indispensable for successful energy planning. Classification, plane segmentation, and spatial analysis are three important aspects in this context. Classification enables extracting rooftops and allows for estimating solar energy potential based on existing training samples. Plane segmentation helps to characterize rooftops by extracting their planar patches. Additionally, spatial analyses enable the identification of rooftop utilizable areas for placing RPVs. This dissertation aims to address some issues associated with these three aspects, particularly (a) *training support vector machines (SVMs) in large datasets*, (b) *plane segmentation of rooftops*, and (c) *identification of utilizable areas for RPVs*. *SVMs* are among the most potent classifiers and have a solid theoretical foundation. However, they have high time complexity in their training phase, making them inapplicable in large datasets. Two new instance selection methods were proposed to accelerate the training phase of SVMs. The methods are based on locality-sensitive hashing and are capable of handling large datasets. As an application, they were incorporated into a rooftop extraction procedure, followed by plane segmentation. *Plane segmentation of rooftops* for the purpose of solar energy potential estimation should have a low risk of overlooking superstructures, which play an essential role in the placement of RPVs. Two new methods for plane segmentation in high-resolution digital surface models were thus developed. They have an acceptable level of accuracy and can successfully extract planar segments by considering superstructures. Not all areas of planar segments are utilizable for mounting RPVs, and some factors may further limit their useability. Two spatial methods for identifying *RPV-utilizable areas* were developed in this realm. They scrutinize extracted planar segments by considering panel installation regulations, solar irradiation, roof geometry, and occlusion, which are necessary for a realistic assessment of RPVs potential. All *six* proposed methods in this thesis were thoroughly evaluated, and the experimental results show that they can successfully achieve the objectives for which they were designed.

Keywords: machine learning, classification, segmentation, support vector machines, instance selection, rooftop plane segmentation, photovoltaic panels, utilizable rooftop areas, geoinformatics

Sammanfattning

I städer där marktillgången är begränsad erbjuder takmonterade solpaneler (eng. rooftop photovoltaic panels) ett attraktivt alternativ för att tillfredsställa höga energibehov. Noggrann värdering av deras potential i förhållande till spatial utbredning och variation är dock oumbärligt för framgångsrik energiplanering. För detta krävs klassificering och segmentering av plana ytor samt spatial analys. Klassificering möjliggör extrahering av hustak och uppskattning av solenergipotentialen baserat på befintliga träningsprov. Segmentering i plan hjälper till att karakterisera hustaken genom extrahering av deras plana segment och spatial analys möjliggör identifiering av användbara takytor för placering av takmonterade solpaneler. Denna avhandlings syfte är att adressera olika problem associerade med dessa; särskilt: (a) *träning av stödvektormaskiner (eng. support vector machines) för stora datamängder*, (b) *segmentering i plan av hustakspunkter* och (c) *identifiering av lämpliga ytor för placering av takmonterade solpaneler*. Stödvektormaskiner tillhör de mest kraftfulla klassificeringsmetoderna och vilar på en solid teoretisk grund. Men på grund av hög komplexitet under träningsfasen är de tidskrävande, vilket gör dem olämpliga för stora datamängder. Två nya initiala urvalsmetoder (eng. instance selection methods) för data föreslås för att påskynda träningsfasen i stödvektormaskiner. Metoderna är baserade på lokalitetskänslig hashning och kan hantera stora datamängder. De inkorporeras i en applikation i form av extrahering av takyta följt av segmentering i plan. *Segmentering av hustak* för uppskattning av solenergipotential bör inkludera låg risk att förbise överbyggnader, som spelar en viktig roll vid placeringen av takmonterade solpaneler. Två nya metoder för segmentering i plan för högupplösta digitala ytmodeller har därför utvecklats. De har en acceptabel nivå av noggrannhet och kan framgångsrikt extrahera plana segment genom att ta hänsyn till överbyggnader. Alla ytor med extraherade plana segment är dock inte användbara för montering av takmonterade solpaneler, samtidigt som andra faktorer ytterligare kan begränsa ytornas användbarhet. Två spatiala metoder för att identifiera användbara takmonterade solpanelytor har utvecklats för detta ändamål. De granskar extraherade plana segment genom att ta hänsyn till regler för panelinstallationer, solinstrålning, takgeometri och ocklusion, vilket är nödvändigt för en realistisk bedömning av potentialen av takmonterade solpaneler. Samtliga *sex* föreslagna metoder i denna studie har utvärderats noggrant och de experimentella resultaten visar att de framgångsrikt kan uppnå de mål som de utformades för.

Nyckelord: maskininlärning, klassificering, segmentering, stödvektormaskiner, urval av träningsdata, segmentering av takytor, solcellspaneler, utnyttjande av takytor, geoinformatik

Acknowledgments

This is the most difficult part of the thesis for me to write as I am afraid I will inevitably fall short of doing it justice, but I will do my best. I would like to express my sincere gratitude to those who have directly or indirectly made this Ph.D. thesis possible and this period of my life transformative.

First and foremost, I must thank my main advisor Stefan Seipel for believing in my potential and helping me design my own research path. I was extremely fortunate to work with Stefan as my advisor. His enthusiasm, alacrity, supportiveness, and patience have led me to impactful, engaging initiatives. He is an incredibly generous collaborator, friend, and insightful mentor; cheers to Stefan for much more than I can express in acknowledgments. I am also grateful to my co-advisors, Anders Brandt and Julia Åhlén. They have been helpful to talk to, and their simple-hearted support and friendliness have been unconditional during the Ph.D. I truly value our discussions on research and teaching.

I would like to thank Gunilla Mårtensson, the head of the engineering and sustainable development faculty, for her support. I would like to express my gratitude to Jonas Boustedt, the head of the computer and geospatial sciences department, for swiftly handling my concerns and offering me an awesome workplace. I also thank staff members of the department and those on the fifth and sixth floors of Building 99, including Goran Milutinovic and Anders Jackson, for their camaraderie and understanding. I am delighted to start working with all of them this year as a colleague. I am thankful to Björn O Karlsson for his patience in answering my questions regarding solar panels, even though he was retired. Additionally, I thank Uppsala municipality and Lantmäteriet for providing data for this study.

Saving the best for last, I owe a sincere debt of gratitude to my parents for providing me with a privileged childhood and instilling the value of education in me. My heartfelt appreciation to my father, who sparked my interest in math and engineering and taught me the value of honesty. And my genuine appreciation to my mother for encouraging me to pursue my passion in whichever career I desire and for teaching me the value of hard work. Their selfless, unending, unwavering, and uncomplicated love and support have laid the foundations for me to build my professional and academic careers. I am eternally indebted to them for my progress in life, and I hope they know how much I look up to them.

List of Papers

This thesis is based on the following papers, which are referred to in the text by Roman numerals.

Paper I
Aslani, M., Seipel, S. (2020). A fast instance selection method for support vector machines in building extraction. *Applied Soft Computing Journal*, 97 Part B: 106716. https://doi.org/10.1016/j.asoc.2020.106716

Paper II
Aslani, M., Seipel, S. (2021). Efficient and decision boundary aware instance selection for support vector machines. *Information Sciences*, 577: 579–598. https://doi.org/10.1016/j.ins.2021.07.015

Paper III
Aslani, M., Seipel, S. (2022). Automatic identification of utilizable rooftop areas in digital surface models for photovoltaics potential assessment. *Applied Energy*, 306: 118033. https://doi.org/10.1016/j.apenergy.2021.118033

Paper IV
Aslani, M., Seipel, S. (2022). A Spatially Detailed Approach to the Assessment of Rooftop Solar Energy Potential based on LiDAR Data. In: *Proceedings of the 8th International Conference on Geographical Information Systems Theory, Applications and Management*: 56–63. https://doi.org/10.5220/0011108300003185

Paper V
Aslani, M., Seipel, S. (Under review, Second round). Rooftop segmentation and optimization of photovoltaic panel layouts in digital surface models.

List of Abbreviations

BPLSH	Border Point extraction based on Locality-Sensitive Hashing
CART	Classification And Regression Trees
CBCH	Clustering-Based Convex Hull
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DRLSH	Data Reduction based on Locality-Sensitive Hashing
DSMs	Digital Surface Models
GA	Genetic Algorithm
GPUs	Graphic Processing Units
LiDAR	Light Detection and Ranging
LSH-IS-S	Locality-Sensitive Hashing Instance Selection by one-pasS
MDDC	Minimum Density Divisive Clustering
NASA	National Aeronautics and Space Administration
nDSM	Normalized Digital Surface Model
NDVI	Normalized Difference Vegetation Index
PCA	Principal Component Analysis
PSDSP	Prototype Selection based on Dense Spatial Partitions
RANSAC	RANdom SAmple Consensus
RPVs	Rooftop Photovoltaic Panels
SVMKM	Support Vector Machines K-Means
SVMs	Support Vector Machines

Table of Contents

Part I
Comprehensive summary

1 Introduction

1.1 Background

Solar energy has been acknowledged as a critical component in achieving energy sustainability owing to its high availability and capacity. It has the potential to minimize greenhouse gas emissions while meeting a significant amount of energy demand (Joshi et al., 2021, Yu et al., 2022). In urban areas, solar energy can be extensively applied on-site using rooftop photovoltaic panels (RPVs), which are scalable and technologically robust (Bódis et al., 2019, Li et al., 2020). In cities where land availability is limited, RPVs can convert buildings to active power generators using only rooftops (Sánchez-Aparicio et al., 2021, Walch et al., 2020). They can alleviate congestion on local urban networks and assist in transitions to nearly zero-energy buildings.

Efficient deployment of RPVs and estimating their energy potential can be done accurately by the inherent capabilities of geoinformatics. Numerous challenges that emerge in RPVs planning and estimating their cost-effectiveness are spatial in nature and require spatial information, which can be properly handled in geoinformatics. Geoinformatics offers a wide range of spatial methodologies to acquire detailed knowledge about solar energy potential. Spatial data management, analysis, and modeling are among salient features of geoinformatics, which can benefit different solar-relevant projects (Gassar and Cha, 2021, Gawley and McKenzie, 2022, Wu and Biljecki, 2021, Zhong et al., 2022). In particular, geoinformatics enables modeling and analyzing rooftops and their surrounding environment, which are necessary for making informed decisions about RPVs deployment (Szabó et al., 2016).

The rapid growth of computational approaches has fueled the development of geoinformatics. Advances in machine learning have brought about more accurate and robust spatial analyses and modeling. Machine learning methods provide the immense ability to extract underlying patterns and segment desirable objects in spatial data without requiring human intelligence (Assouline et al., 2018, Mohajeri et al., 2018, Sun et al., 2022). They can play a pivotal role in the accurate recognition and modeling of rooftops that are necessary for RPVs placement. In addition, their combination with optimization methods allows for an efficient design of RPVs in a spatial context.

1.2 Problem statement

Segmentation and semantic segmentation are active research fields with different engineering and non-engineering use cases. Semantic segmentation aims to identify pixels or points with specific semantic information, whereas segmentation aims to group together pixels with comparable characteristics regardless of their semantic content (Dougherty, 2013). In recent decades, advances in machine learning have led to substantial developments in both fields. Machine learning approaches can effectively deal with a wide range of use cases by continuously training or adjusting their abilities. Supervised and unsupervised

learning are two broad classes of machine learning (Alpaydin, 2020). In supervised learning, often known as a classification problem, the desired class labels are provided in advance, and the learning procedure seeks a decision boundary that best separates classes. In unsupervised learning, often referred to as a clustering problem, no predefined labels are provided, and the learning procedure seeks a set of homogenous clusters that are the best representatives of the dataset (Kubat, 2017). Supervised and unsupervised learning are quite effective for doing both segmentation and semantic segmentation tasks.

Machine learning has facilitated the recognition and modeling of different objects in geoinformatics, particularly in urban contexts. Rooftop modeling has benefited greatly from machine learning techniques (Dixit et al., 2021, Maulik and Chakraborty, 2017, Ren et al., 2022, Turker and Koc-San, 2015, Zhong et al., 2021). They allow for accurate recognition of rooftops and their forming planar patches, which have important practical values in a myriad of geoinformatics applications, such as rooftop solar energy potential estimation.

1.2.1 Selecting instances to train support vector machines

Support vector machines (SVMs) have been widely recognized as one of the most powerful classifiers in machine learning due to their structural risk minimization (Cortes and Vapnik, 1995, Vapnik, 1998). However, their training phase involves solving a quadratic programming optimization problem that poses a computational complexity of $O(n^3)$ (Christmann and Steinwart, 2008). This computational cost hinders SVMs from being used in classification problems that require handling large datasets. Indeed, SVMs training decelerates significantly when dealing with large datasets. This is particularly problematic for pixel-based object extraction (e.g., rooftops) from high-resolution spatial data, such as aerial images and digital surface models (DSMs). The accuracy of SVMs for object extraction and their generalization ability for predicting unseen pixels depend on training samples. Due to the high correlation of neighboring pixels in spatial data, the produced training datasets are usually large and contain many redundant samples. Although using all training samples may reduce the misclassification risk on unseen data points, it may drastically increase the training time of SVMs and even preclude the possibility of training.

Numerous techniques have been proposed to circumvent the computational burden of the training phase, which may be classified into two categories. Methods in the first category aim to accelerate the training computations by reducing the complexity of the underlying optimization problem (Abe, 2010, Kaufmann, 1999, Pavlov et al., 2000, Platt, 1999). However, most of these methods are still computationally demanding for processing large datasets and require substantial memory (Guo and Boukir, 2015, Liu et al., 2017). Methods in the second category, termed instance selection, are aimed at selecting a small subset of training instances to reduce the computational complexity of training (Akinyelu and Ezugwu, 2019, Wang and Shi, 2008). These methods are based on the fact that a training dataset is not used to characterize classes but rather to aid in accurate separation (Foody and Mathur, 2006). Instance selection

methods strategically choose samples representative of the original training set and capable of preserving the original classification power of SVMs. Indeed, instance selection methods accelerate the training of SVMs by filtering out instances that have a negligible impact on classification accuracy (Cervantes et al., 2015). The advantage of the second category is that its methods are usually more general and can be beneficial for classifiers other than SVMs (Garcia et al., 2012, Olvera-López et al., 2010b).

As a result of the rising number of records in datasets, a variety of instance selection methods have been developed. Most existing methods either have high time complexity or ineffectively balance classification accuracy and reduction rate. An efficient instance selection method should be applicable to large datasets and satisfactorily maintain the original classification accuracy while significantly eliminating superfluous instances. Efficient instance selection methods can be incorporated into automatic feature extraction from spatial data, particularly rooftop extraction in this thesis.

1.2.2 Extraction of planar segments of rooftops

Rooftop extraction is a necessary process for various spatial applications. However, accurately characterizing rooftops and reliable analyses of them require identifying their planar segments (Biljecki et al., 2015). This is particularly true in designing RPVs layouts and assessing rooftop solar energy potential. RPVs placement is practically performed at the roof-plane level, and solar suitability of rooftops relies on the shape of rooftops emerged by their constituent planar segments (Sánchez-Aparicio et al., 2021, Thebault et al., 2020). Point clouds and DSMs acquired by airborne laser scanning—also referred to as light detection and ranging (LiDAR)—or stereophotogrammetry enable automatic roof plane segmentation. They provide 3D and 2.5D georeferenced spatial data of the landscape (Jochem et al., 2012).

Plane segmentation divides a rooftop into homogeneous non-overlapping planar patches, allowing direction, slope, size, area, and border of the forming roof faces to be determined. Two of the most common strategies for extracting rooftop planar segments are model-driven and data-driven. In the model-driven approach, planar segments are extracted based on a predefined library of roof shapes in a top-down manner (Zheng and Weng, 2015). The approach works by first defining a library of roof shapes and then choosing a shape that best matches the corresponding area of the DSM or LiDAR. This approach ensures regularized planar patches, but its performance highly depends on the defined library. That is, planar segments of a roof whose shape is not included in the library might not be correctly identified (Wang et al., 2018). Additionally, this approach is likely to result in under-segmentation and overlooking superstructures. In the data-driven approach, on the other hand, each planar segment is derived independently of the overall roof shape (Chen et al., 2014). This can be quite beneficial in the adherence of planar segments to their underlying surface. While this approach is more noise-sensitive than model-driven, it is not limited to a set of predetermined shapes. Therefore, it is capable of extracting all planar segments

of any arbitrary polyhedral rooftop to the extent the spatial resolution allows it (Benciolini et al., 2018, Gilani et al., 2018). Clustering, region growing, and random sample consensus (RANSAC) are among the methods widely employed in the data-driven approach (please refer to Section 1.3). Each of these methods has distinct benefits and shortcomings, necessitating the development of new plane segmentation methods that capitalize on the advantages of each.

1.2.3 Identification of utilizable areas for RPVs

All rooftop areas are not usually available for reasonably installing RPVs, and many factors impose limitations. Available areas for mounting RPVs—referred to as utilizable areas—are critical for a realistic evaluation of rooftop solar potential and, consequently, for informed planning of the RPVs contribution to local networks (Gassar and Cha, 2021, Nelson and Grubesic, 2020). Rooftop superstructures (e.g., chimneys), shadow effects, and panel installation regulations are among the constraints that limit utilizable areas. Installing RPVs on occluded segments or small segments may lower their cost-effectiveness. Indeed, superstructures, multiplanarity of rooftops, and uneven distributions of solar irradiation over rooftops make the identification of utilizable areas challenging, especially in large urban areas.

Numerous studies have addressed the issue of determining rooftop areas utilizable for RPVs when rooftop models are available. A typical approach to determining these areas is to apply a set of loss coefficients showing the average reduction of rooftop areas (Romero Rodríguez et al., 2017). These loss coefficients are estimated based on a series of simplified rule-of-thumb assumptions about rooftops, such as a proportion of rooftop areas mainly in shadow and used for rooftop components (e.g., air conditioning) and service areas (i.e., reserved areas for accessibility and safety purposes). However, this approach provides only a rough estimate of the overall utilizable areas and does not provide any details about the utilizable parts of each planar segment. Additionally, adapting coefficients to new datasets is not straightforward, and incompatible loss coefficients may result in overlooking variations of rooftops.

Recently, more advanced methods have been proposed in which utilizable areas are identified in a spatially detailed manner, in particular at the panel level. More specifically, a feasible layout of RPVs is identified and considered utilizable areas. However, most of the existing methods overlook shadow effects (Mainzer et al., 2017), roof components (Jung et al., 2021), or even installation regulations (Sánchez-Aparicio et al., 2021) in determining the layout of RPVs.

1.3 Related work

In this section, a foundation on the topic is provided, and the current state of knowledge is briefly explained. Specifically, the most relevant methods in instance selection, roof plane segmentation, and utilizable area identification, along with their characteristics, are reviewed.

1.3.1 Instance selection for SVMs

Instance selection methods aim to lower the number of instances while preserving the original classification accuracy by finding instances that contribute to the delineation of classes. Manifold instance selection methods have been proposed in the context of classification, and they can be categorized into five groups: random-based, clustering-based, distance-based, neighborhood-based, and tree-based methods. In random-based methods, a small subset of instances is chosen randomly regardless of the pattern of the data. The most important features of these methods are their simplicity, data size independence, and low computational cost, which allow them to be used with a wide range of datasets (Schmidt et al., 2008). A large standard deviation in classification accuracy, however, may result from using this group of methods.

In clustering-based methods, critical samples are identified by analyzing partitions obtained via clustering. They mainly consist of the following steps: 1) *clustering the dataset*, 2) *identification of potential clusters*, and 3) *selecting representative samples* from potential clusters. *Clustering* methods such as k-means (Barros de Almeida et al., 2000, Shen et al., 2016), adaptive clustering (Boley and Cao, 2004), fuzzy clustering (Cervantes et al., 2006), minimum enclosing ball (Cervantes et al., 2008), hierarchical micro-clustering (Yu et al., 2003), and Ward-linkage clustering (Wang and Shi, 2008) have been employed in previous studies. *Identifying potential clusters* has been primarily done based on the heterogeneity of clusters. The methods proposed by Barros de Almeida et al. (2000) and Olvera-López et al. (2010a) rely on the notion that critical instances significantly contributing to classification appear only in heterogeneous clusters (mixed-class clusters), and homogeneous clusters do not play any role in classification. In SVMKM developed by Barros de Almeida et al. (2000), for example, after clustering the data using k-means, all instances of homogeneous clusters except the centroids are eliminated. However, Cervantes et al. (2008), Koggalage and Halgamuge (2004) and Shen et al. (2016) suggested that homogeneous clusters may contain critical instances, and ignoring them can impair the original classification accuracy. Numerous methods for *selecting representative samples* from potential clusters have been applied, including the safety region (Koggalage and Halgamuge, 2004, Zeng et al., 2008), the convex-concave hull (López Chau et al., 2013), and Fisher's discriminate analysis (Shen et al., 2016). For example, Birzhandi and Youn (2019) developed an instance selection method based on k-means and convex hull algorithms. The method, known as clustering-based convex hull (CBCH), first clusters the data using k-means. Then, it preserves all instances of heterogeneous clusters and exterior instances of homogeneous clusters extracted by a convex hull algorithm. This reduces the chance of missing key examples appearing in homogenous clusters. One of the limitations associated with clustering-based methods is determining the appropriate clustering parameters (e.g., the number of clusters and the stopping criteria), which directly impacts the instance selection process.

In distance-based methods, critical instances are identified by measuring the distance between each instance and its opposite-class instances. Euclidean,

Mahalanobis, and Hausdorff distances have been used in this group of methods (Abe and Inoue, 2001, Liu et al., 2003, Wang et al., 2007). Distance-based methods suffer from high computational and memory complexity due to the necessity to compute distances between samples.

Neighborhood-based methods identify critical instances by inspecting their neighborhood property. In this context, Shin and Cho (2002) proposed a method called NPPS whose underlying idea is that critical samples tend to have heterogeneous neighbors, whereas insignificant samples do not. Two heuristics—*proximity* and *correctness*—were proposed to evaluate the homogeneity and consistency of each instance neighborhood. The *proximity* of each instance measures the homogeneity of its neighbors, and a positive proximity value indicates that the instance has heterogeneous neighbors and hence should be preserved. *Correctness* quantifies the consistency of an instance with its neighbors and is used to identify noisy instances. NPPS requires computing k-nearest neighbors for all samples, which is computationally demanding in dealing with large datasets. To overcome this limitation, NPPS was enhanced by Shin and Cho (2007) so that it does not require calculating k-nearest neighbors for all samples. Indeed, the enhanced method computes the k-nearest neighbors for only a small subset of samples located near decision boundaries rather than the entire dataset. Wang and Kwong (2010) used the *proximity* heuristic to develop a new active learning method for selecting critical instances. Zhu et al. (2020) proposed a novel heuristic for selecting instances named *cited count*. Cited count measures the importance of an instance by counting the number of times it is selected as the nearest neighbor of other-class instances. Instances far from other classes have cited count close to zero and thus are discarded. In the mentioned methods, the number of neighbors is an influential parameter.

In the same context, Carbonera and Abel (2018) developed a method called prototype selection based on dense spatial partitions (PSDSP) that has a linear time complexity to the number of samples. PSDSP extracts representative samples by partitioning the data space and analyzing the density of each partition. The main steps of the algorithm are: 1) dividing the space into non-overlapping partitions using an n-dimensional grid, 2) determining the density of each partition based on the number of samples, and 3) picking critical samples from the first k densest partitions, with k indicating the desired number of samples. Its linear time complexity enables it to handle large datasets. Another instance selection method with linear time complexity was proposed by Arnaiz-González et al. (2016). This method, called locality-sensitive hashing instance selection by one-pass (LSH-IS-S), aims to decrease the number of instances by removing similar samples. It first hashes instances into buckets using locality-sensitive hashing (please see Section 2.2) so that similar instances fall into the same bucket with a high likelihood. Then, it goes through each instance and preserves only one instance in each bucket. In simple terms, an instance is preserved if another instance did not previously occupy its bucket. Therefore, only one instance in a set of similar instances is preserved. Both PSDSP and LSH-IS-S are prone to losing some border instances.

In tree-based methods, indispensable instances are selected based on the output of decision trees. The method proposed by Chang et al. (2010) partitions the data space into subregions using a binary C4.5 decision tree algorithm and discards instances in homogeneous regions. Guo et al. (2010) proposed an instance selection method using ensemble learning based on classification and regression trees (CART) algorithms. In their proposed method, multiple trees are first trained on randomly selected subsets of instances, and the instances labeled differently by the trained trees are considered indispensable and preserved. Their proposed method might be inefficient in handling datasets consisting of a large number of records and features. To address this problem, Guo and Boukir (2015) showed that using a low sampling ratio and ensemble size can accelerate instance selection without sacrificing performance.

1.3.2 Extraction of planar segments of rooftops

Plane segmentation is an essential step in describing the geometry of rooftops. As explained in Section 1.2, model-driven and data-driven are two approaches that might be used for plane segmentation. This section reviews only the data-driven methods, as the proposed plane segmentation methods in the thesis come within this category. Data-driven methods may be classified as edge-based, region-growing-based, model-fitting-based, or clustering-based.

Edge-based methods first use edge detection to delineate boundaries of planar segments and then group the points enclosed by the boundaries (Rabbani et al., 2006). Boundaries are defined by points that have a major change in their local surface properties, such as normal vectors, gradients, curvatures, or higher-order derivatives (Nguyen and Le, 2013). The majority of these methods are directly derived from 2D image processing. They enable fast segmentation, but they may provide inaccurate results caused by delivering discontinuous edges, especially when handling point clouds with noise or uneven density (Castillo et al., 2013, Grilli et al., 2017).

Region growing is a classical method that uses neighborhood attributes to merge close points or regions with similar properties. It begins with *selecting a number of seeds* and continues by *growing the seeds* according to some coplanarity criteria (Vo et al., 2015). A common way for *seed selection* is based on curvature, in which a plane is fitted to every point and its neighbors, and points with low-fitting residuals are picked as seeds. Common coplanarity criteria for the *growing stage* are normal consistency (the angle between normal vectors of neighboring points and the adjusting plane) and point-to-plane distance (Xie et al., 2020). Owing to the simplicity of region growing, it has been frequently used for plane segmentation (Dong et al., 2018, Xiao et al., 2013). Huang et al. (2015) and Jochem et al. (2009) used region growing to extract rooftop planar segments to assess solar potential. To improve the computational efficiency of region growing, Deschaud and Goulette (2010) presented a voxel-based method that replaces points with voxels. Along the same line of research, Araújo and Oliveira (2020) proposed an adaptive octree-based region growing for fast plane segmentation of point clouds. The performance of region growing methods

highly depends on the arrangement and order of seeds as well as normal and curvature estimations of points.

Model-fitting-based methods use primitive shapes (e.g., planes) for segmentation. Several primitive shapes are fitted to the point cloud dataset, and the shape that best matches the dataset is chosen. The points that fit the chosen shape are then grouped together as a segment. In this class of methods, RANSAC, as a popular robust estimator (Fischler and Bolles, 1981), has been widely employed (Bauer et al., 2005, Chen et al., 2014). The two key processes of RANSAC-based methods are *hypotheses generation* and *hypotheses verification*, both done in an iterative manner (Grilli et al., 2017). The *hypotheses generation* step produces a set of plane parameters by randomly choosing a minimum number of required data points. The *hypotheses verification* step chooses the most plausible hypothesis from all the estimated parameter sets. Despite the simplicity of classical RANSAC, using it for plane segmentation in point clouds may lead to the detection of spurious planes. Different variants and modifications of RANSAC have been proposed to address this issue (Raguram et al., 2013, Xie et al., 2020). In Schnabel et al. (2007), an enhanced RANSAC algorithm was proposed that draws only adjacent samples using spatial information and leverages normal vectors of points to improve the inlier function. Li et al. (2017) proposed to partition the point cloud into planar and non-planar cells and apply RANSAC only on planar cells, as non-planar cells usually cause the creation of spurious planes. However, the mentioned methods might still result in the creation of spurious planes.

In clustering-based methods, planar segments are formed by grouping points with similar *features*. Feature definition plays a significant role in this group of methods and should provide the potential for distinctly delineating planar segments. Features should be defined such that points on the same planar segment are mapped close together in the feature space, which is required to simplify the identification of planar clusters and enhance the speed and accuracy of clustering. Point position and locally estimated surface normal vectors are two features that have been widely used (Xie et al., 2020, Xu and Stilla, 2021). Sampath and Shan (2010) identified planar segments by clustering local normal vectors using a fuzzy k-means clustering algorithm (Jain et al., 1999). They incorporated planarity analysis, which separates planar from non-planar points, to enhance the clustering performance. Lukač et al. (2020) extracted planar segments of rooftops using density-based spatial clustering of applications with noise (DBSCAN) (Ester et al., 1996). Position, slope, azimuth, and shadow are among the employed clustering features. In clustering-based methods, the employed clustering algorithm highly affects the results, and thus, choosing a suitable clustering algorithm is crucial. Algorithms with high time complexity might not be able to handle high-resolution point clouds. Additionally, unadaptable algorithms such as k-means and DBSCAN that require manually tuning input parameters for the area or the dataset are not trivial to be applied.

1.3.3 Identification of utilizable areas for RPVs

All parts of rooftop planar segments are not utilizable for RPVs. Utilizable areas should provide sufficient space and solar irradiation for mounting RPVs. Planar segments that are small, surrounded by other objects, or north-facing (in the northern hemisphere) might not be practicable for RPVs. Accurately estimating utilizable areas is necessary for a realistic rooftop solar energy potential estimation. In what follows, some recently developed methods to identify utilizable areas are briefly reviewed.

Romero Rodríguez et al. (2017) determined utilizable areas by applying some loss coefficients to a 3D rooftop model containing only major planar segments. The employed loss coefficients quantitatively model different factors that affect utilizable areas, such as shadow effects, service areas, and superstructures. Adapting the loss coefficients to other areas and datasets is not always trivial. Szabó et al. (2016) compared LiDAR point cloud and image-derived point cloud datasets in extracting planar segments and utilizable areas. To determine the utilizable areas, extracted planar segments are scrutinized in terms of azimuth, tilt (slope), area, and compactness. The study indicates that the LiDAR-based point cloud dataset leads to superior results for plane segmentation and, accordingly, identifying utilizable areas. In Huang et al. (2015), a solar irradiation model was developed for simulating direct and indirect solar irradiation based on DSMs and by utilizing graphic processing units (GPUs). The model considers position, orientation, atmospheric conditions, and occlusion. Rooftops and their forming planar segments are also automatically extracted. Rooftops are extracted by analyzing the height and vegetation index of pixels, and planar segments of rooftops are identified using region growing. The results of the solar irradiation model and rooftop plane segmentation are used to identify utilizable areas. Unsuitable planar segments are filtered out by defining some thresholds for area, tilt, aspect, average solar irradiation, and average sunlight duration.

Applying thresholds at the level of planar segments does not provide sufficient flexibility for the identification of utilizable areas. A partly utilizable segment is either entirely rejected or accepted, i.e., the utilizable area cannot be identified as an independent subsegment. One potential solution to this problem is to identify utilizable areas by *placing* RPVs, which is the same way they are recognized in practice.

Zhong and Tong (2020) developed a two-step procedure to identify utilizable areas in DSMs. Firstly, initial suitable rooftop areas are obtained by performing tilt, azimuth, and solar irradiation analyses. Then, utilizable areas are identified by designing a layout of RPVs that has a maximum coverage of the initial suitable areas. The RPVs layout design problem is conceptualized as a maximal covering problem in which a number of *facilities* are located such that they serve the maximum amount of *demand*. RPVs are considered *facilities*, and the initial suitable rooftop areas are considered *demand*. RPVs location and orientation (portrait or landscape) are determined so that RPVs (*facilities*) provide the maximum coverage of the initial suitable areas (*demands*). Their method disregards planar segments; thus, RPVs may be mounted over ridge

lines, which does not happen in practice. Udell and Toole (2019) proposed a method to find a viable layout of RPVs over planar segments. Their method is based on mixed-integer linear programming to ensure that the designed RPVs arrays meet a specific desired energy while minimizing the installation cost. It limits the placement of RPVs parallel to roof faces and prevents placing them over rooftop edges.

Sánchez-Aparicio et al. (2021) developed a web-based tool for automatically placing RPVs over rooftops. The tool enables designing RPVs in different orientations and directions as well as estimating their energy production. Major planar segments of rooftops are reconstructed by processing LiDAR and aerial images. Despite the simplicity of the tool, the placement procedure considers only the geometry of planar segments, and it overlooks shadow effects that are critical to the efficiency of RPVs. In Yildirim et al. (2021), software for designing RPVs layouts and simulating their electricity production over 3D building models was developed. The software supports various configurations of RPVs and enables placing RPVs in both manual and automatic modes. However, in placing RPVs, it does not consider service areas and rooftop superstructures.

In Mainzer et al. (2017), the proposed methodology considers the shapes and superstructures of rooftops in placing RPVs. Rooftop shapes and superstructures are identified by processing aerial images obtained from publicly available datasets. More specifically, a method based on edge detection is used for the purpose of segmentation. RPVs are placed by using an algorithm that incrementally iterates over the usable roof face areas and fits as many panels as possible within each rooftop. However, the methodology does not consider shadow effects in placing RPVs and cannot accurately determine the tilt of roof faces as no elevation data is used. Another methodology for placing RPVs was proposed by de Vries et al. (2020). It places RPVs using 3D roof faces obtained from aerial images and LiDAR datasets. The placement procedure is based on trial-and-error; that is, it tries different layouts and selects the one with the highest number of RPVs. Although it considers service areas, it ignores solar irradiation in placing RPVs, and thus, the placement is done only based on the geometry of rooftops. Lukač et al. (2020) proposed a methodology that uses an evolutionary algorithm for finding the optimal layout of RPVs. It considers solar irradiation when placing RPVs, but service areas and the alignment of RPVs with the edges of rooftop segments are ignored.

1.4 Research objectives and questions

To address some of the research gaps identified, analyzed, and synthesized in Sections 1.2 and 1.3, three objectives are specified in the dissertation. To achieve these objectives, a few critical questions are highlighted. The objectives and their corresponding questions are as follows:

Objective I: *Developing new instance selection methods that are applicable to large datasets and that can adequately balance the reduction rate and classification accuracy.* The following questions are addressed to achieve this

objective:

- **RQ1:** How can similar samples be identified with linear time complexity?
- **RQ2:** How can border data points (potential support vectors) and interior data points be quickly distinguished?

Objective II: *Developing new plane segmentation methods that are less sensitive to noise and able to handle high-resolution DSMs with a suitable level of accuracy*. The following questions are addressed in connection with this objective:

- **RQ3:** How can clustering-based plane segmentation be enhanced to better identify roof faces?
- **RQ4:** How can the impact of seed order in region growing be alleviated?
- **RQ5:** How can a planarity test be made less sensitive to noise?
- **RQ6:** How can the generation of spurious planes be avoided in RANSAC-based plane segmentation?

Objective III: *Developing new spatially detailed methods for automatic identification of utilizable areas of rooftops that consider installation regulations (service areas), solar irradiation, occlusion, and shape (geometry) of rooftops*. The following questions are answered in relation to objective III:

- **RQ7:** How can areas of planar segments that cannot accommodate an RPV be identified?
- **RQ8:** How can a feasible layout of RPVs that avoids low irradiated areas and maximizes total energy production be identified?

1.5 Research contributions

Six new methods were proposed, two for each of the mentioned objectives. These new methods were published in five scientific papers listed at the beginning of the thesis. In what follows, the six contributions that include *the proposed methods* are presented. Figure 1 indicates the relationships between the scientific articles, research objectives, contributions, and questions.

- **Contribution 1:** A novel instance selection method for SVMs with linear time complexity and low memory consumption, named data reduction based on locality-sensitive hashing (DRLSH). It quickly finds and excludes samples that do not contribute to the description of each class using local sensitivity concepts. Simplicity, integer-based calculations, and the reliance of the method on the property of approximate distance-preserving mapping make it well-suited for dealing with large datasets.
- **Contribution 2:** A novel instance selection method for SVMs with suitable time complexity applicable to large datasets, named border point extraction based on locality-sensitive hashing (BPLSH). It preserves only border patterns and a few interior data points without significantly degrading the original classification accuracy. It does not require computing

the distance between instances; thus, it is appropriate for handling large datasets.

- **Contribution 3:** A new plane segmentation method based on clustering and segment growing integration. Unlike the existing clustering-based methods, the clustering step does not require any prior knowledge regarding the dataset (e.g., the optimal number of clusters) and has an optimized computational speed (linear time complexity), making it suited for high-resolution DSMs. A modified segment growing algorithm is incorporated to avoid any possible over-segmentation.

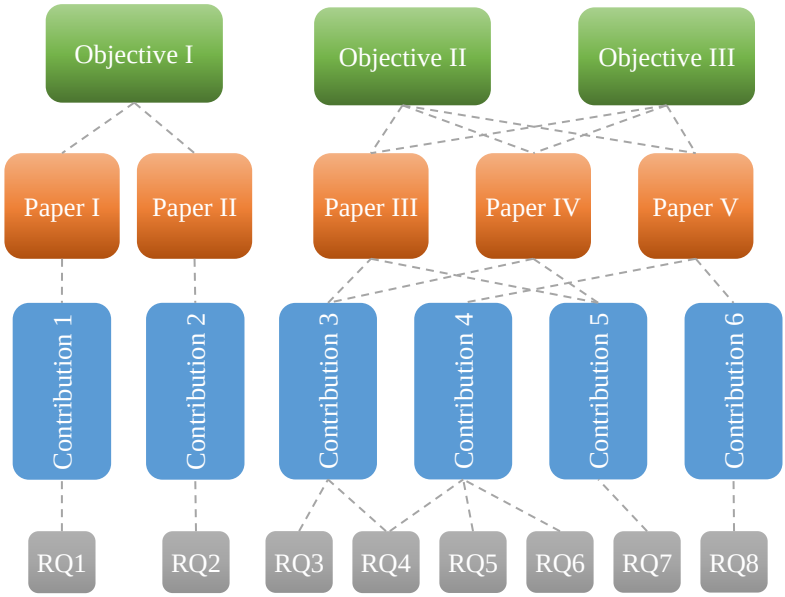

Figure 1. Mapping between the scientific articles, research objectives, contributions (*proposed methods*), and questions.

- **Contribution 4:** A new plane segmentation method that integrates model fitting and segment growing. It is less sensitive to noise and less likely to produce spurious planes. It includes a new planarity analysis method to robustly estimate normal vectors and accurately exclude non-planar pixels, preventing the creation of spurious planes. Modified region growing is used to accurately assign the excluded non-planar pixels to the initially identified planes and avoid the problem of over-segmentation.
- **Contribution 5:** A new method based on morphological operations for the identification of RPV-utilizable rooftop areas. It enables the elimination of service areas and geometrically unsuitable areas from roof faces in a spatial manner, and it facilitates a realistic assessment of rooftop solar energy potential.

- **Contribution 6:** A new method based on metaheuristic optimization for automatically identifying a layout of RPVs that leads to efficient energy production. It considers the shape of roof segments, solar irradiation, occlusion, and installation constraints for placing RPVs. It allows for a more accurate assessment of rooftop solar potential as the number of RPVs is estimated instead of just raw areas.

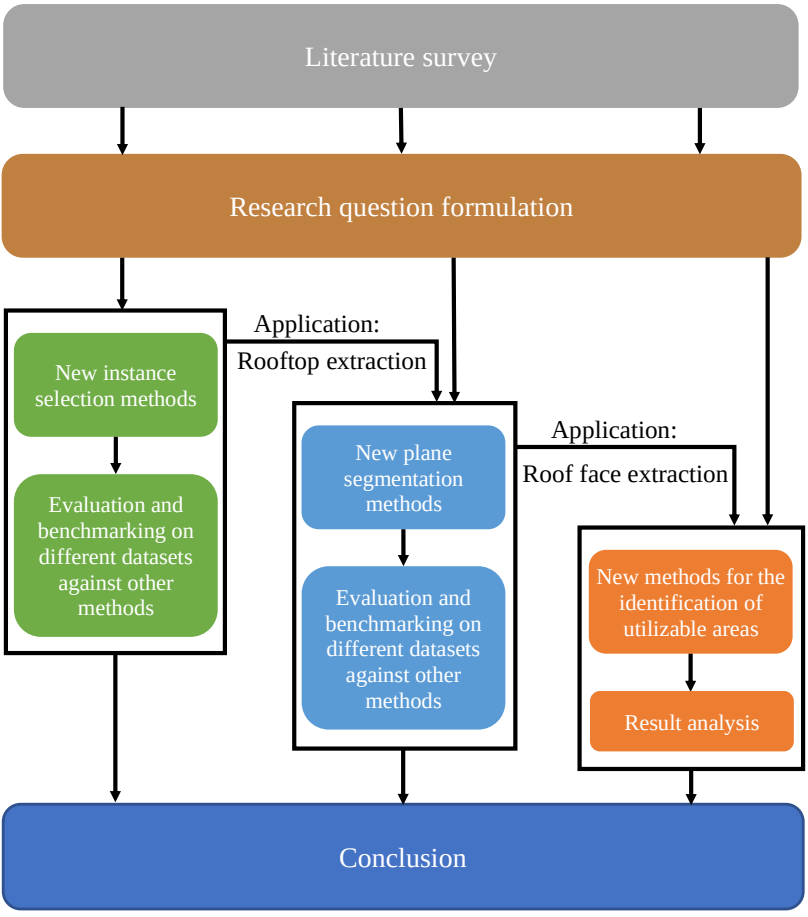

Figure 2. Research strategy of the thesis.

1.6 Research methodology

The flowchart in Figure 2 gives an overview of the research strategy followed in this thesis. After conducting a literature review and formulating research questions, the main core of the thesis, which is developing new methods, was carried out. Firstly, two new instance selection methods using the concept of locality-sensitive hashing were developed. The methods are general and can

be used in any spatial or non-spatial domain. To verify their effectiveness, they were applied to different datasets, and in one of the experimental studies, the methods were incorporated into the procedure of automatic rooftop extraction using SVMs. They were also benchmarked against some state-of-the-art instance selection techniques. Secondly, two new plane segmentation methods were developed based on the integration of clustering, region growing, RANSAC, and planarity analysis to capitalize on the advantages of each method. As an application, they were used to extract planar patches of rooftops. They were compared with some other techniques on two test sites with different roof morphologies to validate their performance. Following that, new methods for identifying RPV-utilizable areas from the extracted planar segments were developed. The results of the methods were compared, and their energy production was computed. Finally, the major findings of the study were concluded.

1.7 Dissertation outline

This dissertation is composed of two main parts. Part I summarizes the study, in which the remainder is organized as follows. Chapter 2 briefly overviews the concepts and algorithms that underpin the study. It explains SVMs, locality-sensitive hashing, k-means clustering, Euclidean clustering, minimum density divisive clustering, and genetic algorithms. Chapters 3–5 explain the developed methods and their performance—presented as five scientific publications. More specifically, Chapter 3 describes the instance selection methods, Chapter 4 describes the plane segmentation methods, and Chapter 5 describes the methods for the identification of RPV-utilizable areas. Figure 3 demonstrates the connection between Chapters 3–5 and the scientific papers. Finally, Chapter 6 concludes the thesis and suggests directions for future work. Part II is the collection of the five publications—four journal articles and one conference paper.

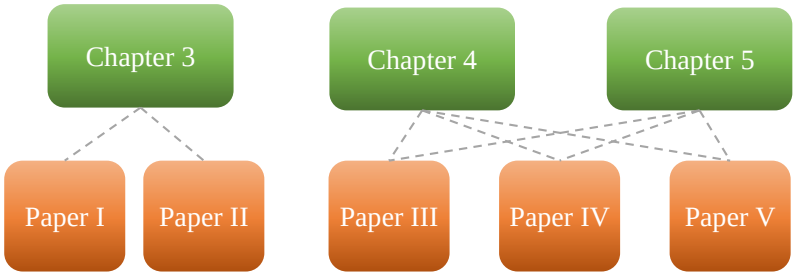

Figure 3. Relationship between the scientific articles and Chapters 3–5.

1.8 Authorship contribution statement

The contributions of the authors of *all papers* are as follows. Mohammad Aslani conceptualized the study with valuable inputs from Stefan Seipel. Mohammad

Aslani developed methodologies and did programming, formal analysis, investigation, writing, and revision. Stefan Seipel contributed to the conceptualization of the methods and with comments and revisions during all research steps and the writing of the papers. Mohammad Aslani is the corresponding author of all papers.

2 Theoretical background

This chapter reviews the methods and concepts necessary for comprehending the remainder of the thesis. First, the concepts of SVMs and locality-sensitive hashing that lay the foundations of instance selection are presented. Then the outlines of some clustering methods and a typical genetic algorithm—used to do plane segmentation and RPVs placement optimization—are briefly explained.

2.1 Support vector machines

SVMs are among the best supervised machine learning models that have a solid mathematical basis (Cortes and Vapnik, 1995, Vapnik, 1998) and are capable of performing linear or nonlinear classification (Murphy, 2012). SVMs were originally designed for binary classification; however, they can be extended to handle multiclass classification by decomposing the original problem into a series of binary problems (Hsu and Lin, 2002). This section explains only the concepts of binary SVMs to show their potential for accurate classification.

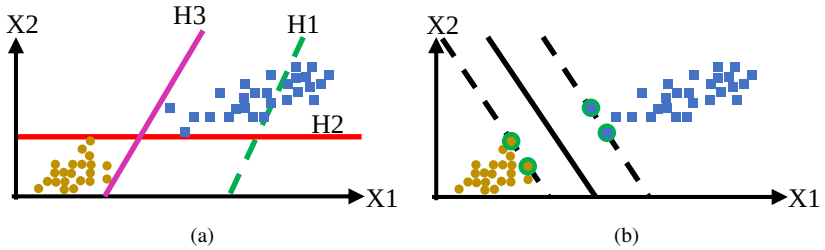

Figure 4. Basis of SVM classification for a two-class dataset. (a) Three different linear classifiers—*H1*, *H2*, and *H3*. The first decision boundary, *H1*, does not correctly classify all training instances. *H2* and *H3* correctly classify all training instances, but they are close to the border instances. (b) Solid line represents a decision boundary generated by an SVM. The line separates the classes with the maximal margin from each class. Circled samples indicate *support vectors*. The area between two dashed lines is called the *margin*.

The fundamental idea behind SVMs is illustrated in Figure 4. Figure 4a shows some decision boundaries for classifying a two-class dataset. The model *H1* cannot correctly delineate the classes, and the other two decision boundaries correctly classify the dataset but with a small margin from each class. In contrast, the decision boundary generated by an SVM and shown as a solid line in Figure 4b has a maximal distance from the nearest training instances of each class. It generalizes better than the other possible decision boundaries and has the minimum risk of misclassification on unseen samples as it has a maximal distance from each class. SVMs determine optimal decision boundaries based on *support vectors*, which are training instances that lie at the edge of each class. Other instances are discarded as they do not contribute to the estimation

of decision boundaries. Indeed, SVMs generate decision boundaries with a high degree of generalizability by using a small set of training instances.

Consider $T = \{(x_i, y_i) \,|\, x_i \in R^d, y_i \in \{-1, +1\}\}, i = 1, 2, \ldots, t$ a set of training instances, where x_i is a d-dimensional input feature of the ith instance, y_i is the corresponding class labels, and t is the number of instances. There might be many decision boundaries that can correctly classify the instances, but only one of them has the maximal margin. Finding a decision boundary with the maximal margin, called the *optimal decision boundary*, is the main purpose of SVMs. Depending on the data, there are different strategies for finding an optimal decision boundary, explained in the following sections.

2.1.1 Hard-margin linear SVMs

A hard-margin linear SVM looks for a *hyperplane* that accurately separates instances and has the maximum margin from the edge of each class. This hyperplane, called the *maximum-margin hyperplane*, lies halfway between *two parallel hyperplanes* with the greatest possible distances between them (Figure 5). Any hyperplane in a d-dimensional space can be defined as $W^T X + b = 0$, where $W \in R^d$ denotes the hyperplane normal vector, and b denotes the bias that controls the position of the hyperplane from the origin. Given a *normalized dataset*, the *two parallel hyperplanes* passing along the edge of each class can be defined using $W^T X + b = +1$ and $W^T X + b = -1$. A hard-margin linear SVM finds W and b so that the distance between these two parallel hyperplanes becomes maximum.

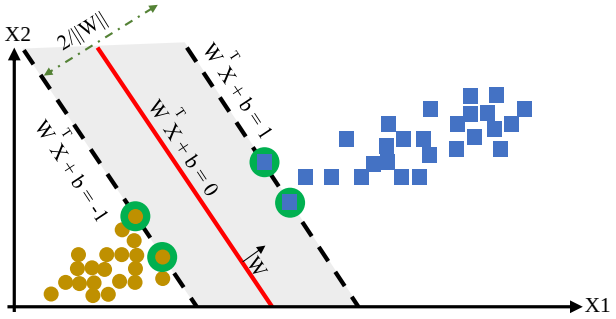

Figure 5. Maximum-margin hyperplane.

Finding the maximum-margin hyperplane requires preventing instances from falling into the margin (the highlighted region in Figure 5). Therefore, the following constraints can be defined according to Equation 1. The function must return -1 or less for negative instances and +1 or greater for positive instances.

$$\begin{cases} W^T x_i + b \geq +1, \text{ if } y_i = +1 \\ W^T x_i + b \leq -1, \text{ if } y_i = -1 \end{cases} \tag{1}$$

The constraints can be combined as follows:

$$y_i\left(W^T x_i + b\right) - 1 \geq 0 \tag{2}$$

The objective function that should be maximized is the distance between the hyperplanes $W^T X + b = +1$ and $W^T X + b = -1$. The distance between these two hyperplanes is $2/\|W\|$, and its maximization corresponds to the minimization of $\|W\|$. The objective function can be defined in a quadratic form to simplify the calculations. Equation 3 shows the constrained optimization problem solved in hard-margin linear SVMs. The solution to this optimization problem, W and b, produces the maximum-margin hyperplane.

$$\min \frac{1}{2}\|W\|^2, \text{ subject to } y_i\left(W^T x_i + b\right) - 1 \geq 0 \; \forall i = 1, \ldots, t \tag{3}$$

This formulation of the problem is called the *primal form*. As the cost function is quadratic, there is only a single global minimum. Solving the problem in its present form is difficult because the parameter b plays no role in the cost function. A method of *Lagrange multipliers*, a strategy for finding the optimal points of a function with constraints, is used to solve this problem. Equation 4 defines the *Lagrangian function* of the problem, which may be optimized without further considering the constraints.

$$L_p(W, b, \alpha) = \frac{1}{2}\|W\|^2 - \sum_{i=1}^{t} \alpha_i \left[y_i\left(W^T x_i + b\right) - 1\right] \tag{4}$$

In this equation, $\alpha_i \geq 0$ are Lagrangian multipliers. The solution to the original constrained optimization problem always corresponds to a *saddle* point of the Lagrangian function. More specifically, L_p should be minimized with respect to W and b and should be maximized with respect to α while keeping $\alpha_i \geq 0$. L_p may be minimized by differentiating it with respect to W and b and setting the derivatives to zero (Equations 5 and 6).

$$\frac{\partial L_p}{\partial W} = 0 \Rightarrow W = \sum_{i=1}^{t} \alpha_i y_i x_i \tag{5}$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^{t} \alpha_i y_i = 0 \tag{6}$$

Substituting Equations 5 and 6 into Equation 4 yields a new formulation referred to as the *dual* form of the primary L_p.

$$L_D = \sum_{i=1}^{t} \alpha_i - \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t} \alpha_i \alpha_j y_i y_j x_i^T x_j \tag{7}$$

$$L_D = \sum_{i=1}^{t} \alpha_i - \frac{1}{2}\sum_{i=1}^{t}\sum_{j=1}^{t} \alpha_i H_{ij} \alpha_j \text{ where } H_{ij} = y_i y_j x_i^T x_j \tag{8}$$

By rewriting Equation 8, the following optimization problem is obtained, which is dependent on α_i, and should be maximized with respect to α_i.

$$\max_{\alpha} \left[\sum_{i=1}^{t} \alpha_i - \frac{1}{2} \alpha^T \mathbf{H} \alpha \right] \text{ s.t. } \alpha_i \geq 0 \, \forall i \text{ and } \sum_{i=1}^{t} \alpha_i y_i = 0 \tag{9}$$

This is a convex quadratic optimization problem and can be solved by quadratic programming. After determining the optimized values of α_i, W is computed using Equation 5. The last step is to compute b. According to Karush-Kuhn-Tucker condition, $\alpha_i \left[y_i \left(W^T x_i + b \right) - 1 \right] = 0$. When $\alpha_i > 0$ (i.e., $\alpha_i \neq 0$), there will be:

$$\alpha_i \left[y_i \left(W^T x_i + b \right) - 1 \right] = 0 \Rightarrow y_i \left(W^T x_i + b \right) = 1 \Rightarrow W^T x_i + b = y_i \tag{10}$$

As $y_i \in \{-1, +1\}$, those instances whose $\alpha > 0$ are on the hyperplanes ($W^T X + b = \pm 1$), and they are *support vectors*. The parameter b can be computed using the set of support vectors as follows:

$$b = \frac{1}{|S|} \sum_{s \in S} \left(y_s - W^T x_s \right) \tag{11}$$

In this equation, S denotes the set of indices of the support vectors, and it is determined by finding the indices i, where $\alpha_i > 0$. The optimal values of W and b define the maximum-margin hyperplane. Each new point x' is classified by evaluating $y' = sgn(W^T x' + b)$. Considering Equations 5 and 11 and knowing that α values of only support vectors are greater than zero, it can be inferred that only support vectors are used to predict the class of new instances, and non-support vectors play no role in computing the output. Consequently, it is not essential to incorporate all instances when classifying using SVMs; only instances close to decision boundaries may be enough for accurate classification.

2.1.2 Soft-margin linear SVMs

Hard-margin linear SVMs suffer from two drawbacks. First, they cannot handle data that are not linearly separable. For instance, no decision boundary can be determined for the dataset in Figure 6a as it is not linearly separable. Second, hard-margin SVMs are susceptible to overfitting and are sensitive to outliers. For instance, the existence of only one outlier in Figure 6b has significantly changed the decision boundary compared with Figure 4b.

To deal with these problems, it is necessary to relax the constraints to allow for misclassification slightly. This is done by introducing a set of positive slack variables ξ_i, $i = 1, \ldots, t$ to Equation 1 for each sample:

$$\begin{cases} W^T x_i + b \geq +1 - \xi_i, \text{ if } y_i = +1 \\ W^T x_i + b \leq -1 + \xi_i, \text{ if } y_i = -1 \end{cases} \text{ where } \xi_i \geq 0 \, \forall i \tag{12}$$

ξ_i denotes the distance between the sample and the hyperplane passing through the support vectors of the sample class (Figure 7). If a point is on the correct side of the margin boundary, its corresponding ξ is zero. The two constraints can be combined to give:

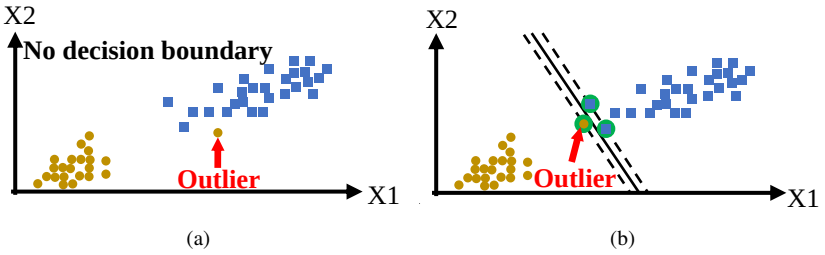

Figure 6. (a) Example that a hard-margin linear SVM is unable to find a decision boundary, and (b) Sensitivity of hard-margin linear SVMs to outliers.

$$y_i \left(W^T x_i + b\right) - 1 + \xi_i \geq 0 \text{ where } \xi_i \geq 0 \forall i \tag{13}$$

This constraint will be true for any misclassified sample by setting their corresponding ξ to a sufficiently large value. However, as the classification objective is to reduce misclassification errors, a regularization term $c\sum_{i=1}^{t} \xi_i$ is added to the previous objective function (Equation 3) to penalize solutions for which ξ_i are large. The parameter c regulates the compromise between the margin width and the slack variable penalty. The larger the value of c is chosen, the higher the penalty will be for samples that lie on the wrong side of the margin boundary. Moreover, the tolerance for misclassification will decrease, and the likelihood of overfitting will increase. On the other hand, setting c to a small value may lead to unnecessarily high numbers of support vectors. It is noteworthy that the optimization problem given by Equation 3 is a specific case when $c \to \infty$.

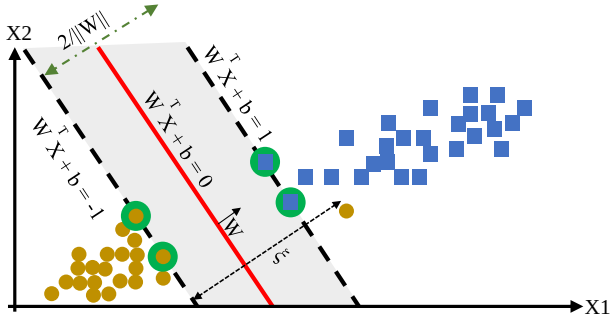

Figure 7. Soft-margin decision boundary. Misclassification of points is allowed.

$$\min_{W,b,\xi} \left[\frac{1}{2} ||W||^2 + c \sum_{i=1}^{t} \xi_i \right] \text{ s.t. } y_i \left(W^T x_i + b\right) - 1 + \xi_i \geq 0 \text{ and } \xi_i \geq 0 \forall i \tag{14}$$

Reformulating the optimization problem as a Lagrangian gives Equation 15, where $\alpha_i \geq 0$ and $\mu_i \geq 0$ are the Lagrangian multipliers.

$$L_p(W,b,\xi,\alpha,\mu) = \frac{1}{2}||W||^2 + c\sum_{i=1}^{t}\xi_i - \sum_{i=1}^{t}\alpha_i\left[y_i\left(W^T x_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{t}\mu_i\xi_i \tag{15}$$

L_p should be minimized with respect to W, b, and ξ and maximized with respect to α and μ. After setting the derivatives with respect to W, b, and ξ to zero, substituting them back in, and simplification the results, the following dual form of the optimization problem is obtained, which can be solved by quadratic programming.

$$\max_{\alpha}\left[\sum_{i=1}^{t}\alpha_i - \frac{1}{2}\alpha^T\mathbf{H}\alpha\right] \text{ s.t. } 0 \le \alpha_i \le c\ \forall i \text{ and } \sum_{i=1}^{t}\alpha_i y_i = 0 \tag{16}$$

Comparing Equations 16 and 9 shows that adding the regularization term $c\sum_{i=1}^{t}\xi_i$ has made only one change that is converting $0 \le \alpha_i$ to $0 \le \alpha_i \le c$. The parameter W is determined by solving the dual optimization problem and substituting the optimized values of α into Equation 5. The parameter b is computed in the same manner as in Equation 11, but the set of support vectors used to calculate b is obtained using the indices i whose $0 < \alpha_i < c$.

2.1.3 Soft-margin nonlinear SVMs

The produced decision boundaries of the explained SVMs are linear and may not perform well in classifying nonlinearly separable data. Therefore, it is necessary to extend them to allow for nonlinear decision surfaces. For this purpose, the data are mapped into a high dimensional space, so the mapping facilitates separating the data using a linear decision boundary. A function $\Phi : R^d \to F$ that maps any vector from the input space into a new space F (sometimes called feature space) is used. To apply this mapping, it is necessary to replace every x with $\Phi(x)$ in the SVM algorithm. As the algorithm can be entirely expressed in terms of the product of x_i and x_j (i.e., $x_i^T x_j$), *only the products* of the mapped inputs (i.e., $\Phi(x_i)^T\Phi(x_j)$) need to be calculated without requiring calculating $\Phi(x_i)$ and $\Phi(x_j)$ separately. Therefore, a kernel function is substituted for their product as follows:

$$k(x_i,x_j) = \Phi(x_i)^T\Phi(x_j) \tag{17}$$

This is referred to as the *kernel trick* (Boser et al., 1992), and its advantage is that computing $k(x_i,x_j)$ is much less expensive than computing $\Phi(x_i)^T\Phi(x_j)$, especially when the mapping is to a high-dimensional space. There are several different kernel functions, and one that is widely used is the radial basis kernel defined using Equation 18, in which σ is the parameter controlling the width of the Gaussian kernel.

$$k(x_i,x_j) = \exp\left(-\frac{\left|x_i - x_j\right|^2}{2\sigma^2}\right) \tag{18}$$

Soft-margin SVMs using Gaussian kernels have two hyperparameters—c and σ—affecting classification accuracy. With large values of the hyperparameters, SVMs tend to overfit, and with small values of the hyperparameters, there is a tendency for SVMs to underfit the training data. Therefore, it is necessary to tune the hyperparameters for the task at hand carefully. In this thesis, nonlinear soft-margin SVMs were used, and their hyperparameters were optimized using Bayesian optimization through a cross-validation scheme (Shahriari et al., 2016) to ensure proper classification accuracy.

Solving the optimization problems in Equations 9 and 16 involves quadratic programming, which poses a computational complexity of $O(t^3)$ and memory complexity of $O(t^2)$, where t denotes the number of training samples. These high memory and computational complexities inhibit the applicability of SVMs on large training samples. As non-support vectors do not play any role in classification, one immediate remedy to the computational burden of training SVMs is to eliminate non-support vectors from the training dataset using an instance selection method with low computational and memory complexities.

2.2 Locality-sensitive hashing

Locality-sensitive hashing is an effective and fast approach for checking the similarity among items (Leskovec et al., 2014). It relies on the concept of *locality-sensitive hash functions* that aim to assign similar items to the same bucket with a high probability while minimizing the likelihood of allocating dissimilar items to the same bucket. In addition to having a suitable run time, locality-sensitive hashing scales well with the data dimension (Indyk and Motwani, 1998). Let X be an n-dimensional space, D be a distance measure, and $H = \{h : X \to U\}$ be a set of hash functions that maps the original space X to some universe U. The set H is called (d_1, d_2, p_1, p_2)-sensitive if the following conditions remain valid for every h in H between any two points x and y:

- If $D(x,y) \leq d_1$, then the probability that $h(x) = h(y)$ is at least p_1.
- If $D(x,y) \geq d_2$, then the probability that $h(x) = h(y)$ is at most p_2.

The property of hash functions drawn from H is that close points have a high chance of being hashed to the same value, whereas faraway points have a low probability of being hashed to the same value. p_1 *should be high* (close to 1) to prevent false negatives, and p_2 *should be low* (close to 0) to prevent false positives. Nothing in these statements relates to what occurs when the distance of the objects is between d_1 and d_2. The distances d_1 and d_2 can be as close as possible, but the penalty is that p_1 and p_2 also get closer. However, it is feasible to combine hash functions such that p_1 and p_2 are separated without affecting the distances d_1 and d_2.

To decrease the probability of false positive p_2, a set of k independently selected hash functions of H are concatenated for a given integer M to form a family of hash functions $g(X) = (h_1(X), h_2(X), \ldots, h_k(X))$, where $h_i \in H$. As a result, the probability of true positive p_1 decreases as well. To compensate for the decrease in p_1—increase the collision probability of close samples in

buckets—a set of hash function families $G = \{g_1, g_2, \ldots, g_l\}$ is constructed.

Datar et al. (2004) proposed a hash function h for the Euclidean metric according to Equation 19. In this equation, $\vec{a}$ denotes an n-dimensional vector whose components are selected independently from a Gaussian distribution with a mean of 0 and a standard deviation of 1. The parameter b denotes a random real value drawn uniformly from $[0, r]$, and the parameter r controls the width and number of buckets. A small value of w leads to a large number of buckets with a small width. The hash function indicates in what slice of each hyperplane the point (object) has fallen.

$$h_{\vec{a},b}(\vec{x}) = \left\lfloor \frac{\vec{a} \cdot \vec{x} + b}{r} \right\rfloor \tag{19}$$

2.3 Clustering

Clustering is the most important group of unsupervised techniques. It divides data points into a set of partitions based on a specific pre-defined objective function, aiming to decrease inter-partition similarity and increase intra-partition similarity. Indeed, data points are divided so that points within a group are similar to one another and are dissimilar to points of other groups. Both the similarity and the dissimilarity should be measurable in a clear and meaningful manner. From the pattern recognition perspective, clustering should lead to a representation that best reflects the population being sampled.

Given a set of input patterns $T = \{x_i \mid x_i \in R^d\}$, where t is the number of instances and x_i is the ith d-dimensional sample. In this thesis, clustering is referred to as techniques that seek a k-partition of T, $C = \{C_1, C_2, \ldots, C_k\}$ ($k \leq t$), such that

- $C_i \neq \varnothing$, $i = 1, \ldots, k$;
- $\cup_{i=1}^{k} C_i = T$;
- $C_i \cap C_j = \varnothing$, $i, j = 1, \ldots, k$ and $i \neq j$

There are many well-known clustering methods, and they are used in different applications (Xu and Wunsch, 2005). This study employed k-means++, Euclidean clustering, and minimum density divisive clustering (MDDC), which are further explained in the following sections.

2.3.1 k-means and k-means++

k-means is one of the oldest clustering methods, and its concept is still extensively used in different domains for cluster analysis purposes. k-means partitions the dataset by finding k cluster centroids such that the sum of squared errors between the centroids and their cluster members is minimized. Formally, the goal of k-means is to find a set of k clusters (in the dataset T) whose centers are $\mu = \{\mu_1, \mu_2, \ldots, \mu_k\}$ such that the following cost function is minimized:

$$\Phi = \sum_{j=1}^{k} \sum_{x_i \in c_j} \left| x_i - \mu_j \right|^2 \tag{20}$$

A typical way to minimize the cost function is Lloyd's algorithm (Algorithm 1). It operates in the following steps: First, a set of k data points that serve as k cluster centers is randomly chosen. Second, data points are assigned to their nearest cluster center based on a similarity measure, e.g., Euclidean distance. Third, cluster centers are recalculated as the means of assigned data points (MacQueen, 1967). The second and third steps are repeated until some convergence criteria are fulfilled, e.g., the maximum number of iterations or minimum change in cluster centers. According to research, this algorithm is likely to converge to the global optimum when the clusters are well-separated (Meila, 2006).

Algorithm 1 Pseudo-code of Lloyd's algorithm.

Input: A set of data points $T = \{x_i \,|\, x_i \in R^d\}$, $i = 1, 2, \ldots, t$
Number of desired clusters k
Output: A set of k clusters $C = \{c_j, j = 1, 2, \ldots, k\}$

1: Arbitrarily choose k data points from T as initial centroids
2: **repeat**
3: Assign each data point x_i to the cluster that has the closest centroid
4: Calculate a new mean for each cluster
5: **until** convergence criteria are met

Lloyd's algorithm has a problem of initialization sensitivity. It may lead to different solutions for a given k and different initial values of centroids. Indeed, it may not always lead to the global optimum solution for any initial values of centroids. One way to overcome this problem is to run k-means multiple times for the same value of k but with different initializations and choose the clustering results with the smallest squared error (Equation 20). However, this idea is computationally demanding, especially in large datasets. To circumvent this problem, Arthur and Vassilvitskii (2007) developed k-means++ that features a smart centroid initialization phase. It avoids poor solutions that standard k-means might produce due to unsuitably chosen centroids. The idea of k-means++ is that spreading out k initial cluster centroids enhances convergence speed and decreases the possibility of converging to local minimum solutions. In k-means++, the following steps are taken to choose initial centroids:

1. Pick the first centroid among the data points randomly.
2. Compute the distance d between each data point and the nearest centroid.
3. Pick the next centroid based on a weighted probability corresponding to squared distances d^2.
4. Repeat steps 2 and 3 until all k centroids are selected.

In both k-means and k-means++, the optimal number of clusters has to be predefined. If k is less than the optimal value, the result does not reflect the essence of the underlying data. In contrast, if k is greater than the optimal value,

the resulting model may represent unnecessary relationships between the data points. To determine the optimal number of clusters, some methods have been developed. In these methods, cluster purity (Manning et al., 2008) is measured for different values of k, and the value that results in a suitable cluster purity is chosen. Most of these methods are computationally inefficient, especially when the dataset is large. It restricts applications of k-means and k-means++ in problems where automatic clustering is required and no domain knowledge exists.

2.3.2 Euclidean clustering

Euclidean clustering essentially partitions the data points based on the closeness of points, defined by a distance threshold r in Euclidean space (Klasing et al., 2008). Clusters are created such that each point of a cluster lies in an r-radius neighborhood of at least one point in the same cluster. The shape and number of clusters are determined by the clustering process using the distance threshold rather than being specified as input parameters to the algorithm. Therefore, this algorithm is useful when the neighborhood distance is already known, but the number of clusters is unknown. The major steps of the algorithm are as follows:

1. Go through all points.
2. If the current point has been assigned to a cluster, go to the next point.
3. For the current point
   (a) Find all neighbors located within distance r.
   (b) If any of these neighbors have been clustered, assign the current point and all unassigned neighbors to the same cluster.
   (c) If the current point has been clustered, and there are neighbors assigned to different clusters, merge all these clusters.
   (d) If the current point and its neighbors have not been assigned to any cluster, define a new cluster and assign the points to it.

Algorithm 2 shows a detailed pseudo-code description of the clustering method. The nearest neighbor queries account for the bulk of the computational demand of clustering. To enhance the computational performance of the algorithm, nearest neighbor queries are performed only for the points that have not been clustered (lines 2–4). Moreover, a *k-d tree* can be used to accelerate the nearest neighbor searches (Bentley, 1975).

2.3.3 Minimum density divisive clustering

MDDC is a density-based hierarchical clustering algorithm, which assumes clusters are contiguous regions of high-probability density separated by contiguous regions of low-probability density. It starts with a single, all-inclusive cluster and generates a nested series of partitions by iteratively splitting clusters until no clusters can be divided further (Algorithm 3). Partitions are formed by a set of hyperplanes called minimum density hyperplanes (MDHs). MDHs are determined to pass through regions with low-probability density and avoid

Algorithm 2 Pseudo-code of Euclidean clustering algorithm.

Input: A set of data points $T = \{x_i \,|\, x_i \in R^d\}$, $i = 1, 2, \ldots, t$
Distance threshold r
Output: A set of clusters C

for each $x_i \in T$ **do**
  if $hasCluster(x_i)$ **then**
    go to the next point
  end if
  $NN \leftarrow$ neighbors of x_i in radius r
  for each $x_j \in NN$ **do**
    if $hasCluster(x_i)$ & $hasCluster(x_j)$ **then**
      if $clusterOf(x_i) \neq clusterOf(x_j)$ **then**
        $mergeClusters(clusterOf(x_i), clusterOf(x_j))$
      end if
    else
      if $hasCluster(x_j)$ **then**
        $clusterOf(x_i) \leftarrow clusterOf(x_j)$
      else
        if $hasCluster(x_i)$ **then**
          $clusterOf(x_j) \leftarrow clusterOf(x_i)$
        end if
      end if
    end if
  end for
  if $\neg hasCluster(x_i)$ **then**
    $clusterOf(x_i) \leftarrow makeNewCluster$
    for $x_j \in NN$ **do**
      $clusterOf(x_j) \leftarrow clusterOf(x_i)$
    end for
  end if
end for

intersections with high-density areas (Pavlidis et al., 2016). In what follows, the determination of an MDH is explained.

The density on a hyperplane H is defined as the integral of the probability density function p along the hyperplane:

$$I(v,b) = \int_{H(v,b)} p(x)dx \tag{21}$$

In this equation, v and b denote the unit normal vector of the hyperplane and the displacement of the hyperplane from the origin, respectively. As the density function p is unknown in practical applications, a continuous density estimator

Algorithm 3 Pseudo-code of MDDC.

Input: A set of data points $T = \{x_i \,|\, x_i \in R^d\}, i = 1, 2, \ldots, t$
Output: A set of clusters C

1: Set $c \leftarrow T$
2: **while** there is at least one cluster in C that can split further **do**
3: Select the largest cluster c among the clusters that can split
4: Split c into two subsets c_1 and c_2 using a minimum density hyperplane
5: Remove c from C and set $C \leftarrow C \cup \{c_1, c_2\}$
6: **end while**

with isotropic Gaussian kernels is used:

$$\hat{I}(v,b) = \int_{H(v,b)} \frac{1}{t\left(2\pi h^2\right)^{\frac{d}{2}}} \sum_{i=1}^{t} \exp\left\{-\frac{\|x - x_i\|^2}{2h^2}\right\} dx \tag{22}$$

In this equation, h denotes the bandwidth for the kernel density estimator, t denotes the number of instances, and d denotes the dimension of the dataset. The advantage of this class of kernel density estimator is that $\hat{I}(v,b)$ can be exactly evaluated through a one-dimensional kernel density estimator, constructed by the projections of the data points onto v and evaluating the density at b:

$$\hat{I}(v,b) = \int_{H(v,b)} \frac{1}{t\left(2\pi h^2\right)^{\frac{1}{2}}} \sum_{i=1}^{t} \exp\left\{-\frac{(b - v \cdot x_i)^2}{2h^2}\right\} dx \tag{23}$$

The MDH for clustering is the solution to the minimization problem stated in Equations 24–26. In this minimization, a projected vector v and intercept b should be found such that the density distribution has a minimum at some point between the projected data.

$$\min_{v} \varphi_{CL}(v) \tag{24}$$

$$\varphi_{CL}(v) = \min_{b} f_{CL}(v,b) \tag{25}$$

$$f_{CL}(v,b) = \hat{I}(v,b) + \frac{\left(e^{1/2} h^2 2\pi\right)^{-1}}{\eta^{\varepsilon}} \max\left\{0, \mu_v - \alpha\sigma_v - \mathrm{b}, \mathrm{b} - \mu_v - \alpha\sigma_v\right\}^{1+\varepsilon} \tag{26}$$

The parameters μ_v and σ_v denote the mean and standard deviation of the projections $\{v \cdot x_i\}_{i=1}^{t}$. The second term in Equation 26, $\frac{\left(e^{1/2} h^2 2\pi\right)^{-1}}{\eta^{\varepsilon}} \max\left\{0, \mu_v - \alpha\sigma_v - \mathrm{b}, \mathrm{b} - \mu_v - \alpha\sigma_v\right\}^{1+\varepsilon}$, is a penalty function that ensures the optimal MDH does not have a large value of $|b|$. This constraint is necessary to

prevent hyperplanes from being placed far from the center of the data as the density is always close to zero at the tails of density functions. The parameter $\alpha > 0$ specifies the trade-off between a balanced bi-partition and the ability to identify hyperplanes with a lower density. Smaller values of α allow more balanced data partitions at the expense of eliminating low-density hyperplanes that effectively separate clusters. Increasing α, on the other hand, raises the probability of separating only a few outlying observations. The parameter $\varepsilon \in (0,1)$ ensures globally continuous differentiability of the penalty function. The parameter $\eta \in (0,1)$ controls the distance between the minimizers of $\arg\min_b f_{CL}(v,b)$ and $\arg\min_b \hat{l}(v,b)$.

According to Pavlidis et al. (2016), $\eta = 10^{-2}$ and $\varepsilon = 1 - 10^{-6}$ are suitable values and avoid numerical instability. The bandwidth parameter h is automatically set using Silverman rule (Silverman, 1986). The optimal value of α is adaptively determined by progressively increasing its value, solving the optimization problem, and assessing outcomes. The time complexity of estimating an MDH is *linear* with respect to the number of samples, making MDDC suitable to handle large datasets. For more information, readers may refer to Pavlidis et al. (2016).

Each resulting cluster from bi-partitioning needs to be evaluated to see if it can split further in the next iterations. This evaluation is done by checking the form of the density distribution of the cluster. If the distribution is unimodal, that is, the detected MDH falls at the endpoints of the search interval $[\mu_v - \alpha\sigma_v - \mathrm{b}, \mathrm{b} - \mu_v - \alpha\sigma_v]$, the cluster is deemed indivisible.

As is evident from Algorithm 3, MDDC *does not require any prior knowledge* regarding the dataset as input parameters, such as the optimal number of clusters or distance between points, and it adaptively determines the shape and number of clusters inherent within the data. This property makes MDDC suitable for the tasks that require automatic clustering, such as plane segmentation employed in this thesis.

2.4 Genetic algorithm

Genetic algorithms (GAs) are well-known metaheuristic optimization techniques inspired by the process of natural adaptation to evolve solutions to problems (Holland, 1992). They find the fittest solution to a given optimization problem by repeatedly applying stochastic operators, emulating natural ways of evolution, to a set of possible solutions (Haupt and Haupt, 2004). They are more efficient than random and exhaustive search algorithms yet need no derivative information or other auxiliary knowledge regarding the problem; only the corresponding fitness levels affect the search directions.

Although there are different variants of GAs, they typically share the following structure: A typical GA works by iteratively updating a pool of potential solutions, known as a population. Each solution encoded into a chromosome-like data structure is scored at each iteration using the fitness function, and some

fit solutions are selected to seed a new population. Some selected solutions are passed on intact to the next population, and others serve as the basis for generating new offspring via genetic operations, including crossover and mutation. The procedure forms a generate-and-test beam-search of solutions, where the fittest variants of solutions are most likely to be considered next. It enables GAs to effectively explore spaces of solutions with complex interdependent components whose effects on the overall solution fitness values may be difficult to model.

The pseudo-code shown in Algorithm 4 gives the outline of a prototypical GA. The algorithm begins with an initial set of random solutions P. The generated solutions are evaluated using a fitness function defined based on the optimization problem. Then, the next population of solutions P_s is formed by probabilistically *selecting* solutions from the current population according to their fitness as well as by producing *new solutions*. A part of the *new solutions* is produced by applying a *crossover* operator to pairs of solutions in P, and the remainder is generated by *mutating* the resultant solution generation. This procedure is repeated until a sufficiently suitable solution is discovered. In what follows, the main components of a GA are elucidated.

Algorithm 4 Pseudo-code of a prototypical genetic algorithm.

Input: A function that evaluates a solution and assigns a score to it, *Fitness*
The number of solutions in each population, p
The proportion of the population to be replaced by crossover at each step, r
The mutation rate, m
Output: The most fit solution

1: $P \leftarrow$ a population of p randomly generated solutions encoded to chromosomes
2: Compute $Fitness(i)$ for each $i \in P$
3: **while** fitness of the most fit solution in P is not sufficient **do**
4: **[Selection]**: Probabilistically select $(1-r) \times p$ solutions of P to add to a new generation P_s.
5: **[Crossover]**: Probabilistically select $r \times p$ solutions of P, pair them up, produce offspring solutions by applying the crossover operator and insert the offspring solutions into P_s.
6: **[Mutation]**: Choose $m \times p$ members of P_s. For each, invert a random bit in its representation.
7: $P \leftarrow P_s$
8: Compute $Fitness(i)$ for each $i \in P$
9: **end while**
10: Return the solution with the highest fitness from P

2.4.1 Encoding

Encoding is a process of representing solutions so that they can be easily manipulated by genetic operators (e.g., crossover). All feasible solutions should relate to at least one possible chromosome in encoding; otherwise, the entire search space cannot be explored. The most common encoding scheme is *binary encoding*, in which each solution is represented as a string of 0 and 1 bits (Katoch et al., 2021), and each bit shows some characteristics of the solution. This type of encoding allows for faster implementation of genetic operators; nevertheless, modifications of the resulting solutions after applying genetic operators are occasionally required. *Value encoding* is another common encoding scheme. In this encoding, every solution is represented by a string of some type of value, ranging from integers to complex objects. It is particularly beneficial when the problem values are difficult to be encoded in a binary form. In this thesis, value encoding was used. Other types of encoding are octal, hexadecimal, permutation, and tree. Readers can refer to Sivanandam and Deepa (2008) for more information.

2.4.2 Selection

Selection is a crucial step in GAs that determines whether a particular solution can participate in the generation of the next population. It is usually stochastically designed to maximize the likelihood of selecting solutions with a high fitness level while retaining the diversity of the population large enough to avoid convergence to suboptimal solutions. The most common selection techniques are *roulette wheel*, *rank*, and *tournament* (Sivanandam and Deepa, 2008). In *roulette wheel*, each solution is assigned to a wheel section whose size is proportional to its fitness. Then the wheel is rotated, and the solution occupying the section where the wheel marker stopped is chosen. In this manner, solutions are selected with a probability proportional to their fitness. In the roulette wheel method, solutions are sometimes chosen by using multiple markers equally placed around the wheel. It eliminates the need to turn the wheel multiple times as several solutions can be selected with a *single* spin of the wheel. *Rank selection* is a modified form of roulette wheel selection. It uses ranks instead of fitness values to determine the selection probability. Solutions are first sorted ascendingly by their fitness values (from worst to best), and the likelihood of selecting a solution is proportionate to its position in the sorted population. In *tournament selection*, several tournaments consisting of a few numbers of randomly selected solutions are conducted, and the fittest solution from each tournament is chosen. In this method, the probability of a solution being selected depends much less on its fitness value. In this thesis, rank selection was used.

2.4.3 Crossover

Crossover is a basic genetic operator that produces offspring by fusing more than one parent (Sivanandam and Deepa, 2008). As a way to stochastically generate

new solutions from existing solutions, it resembles the biological process of gene recombination between chromosomes. From the optimization standpoint, the purpose of crossover is to ease the exploitation of the search space and efficiently evolve populations to optimal points via recombining solutions (Mitchell, 1996). The most common crossover operators are *single-point*, *multi-point*, *uniform*, *shuffle*, and *intermediate*, each with its own way of exploring and exploiting the search space. In *single-point crossover*, parent bit strings are cut at a random point, called the pivot point or crossover point, and the sections after the cuts are swapped. Figure 8 illustrates single-point crossover, in which the bits after the pivot point are swapped to generate offspring.

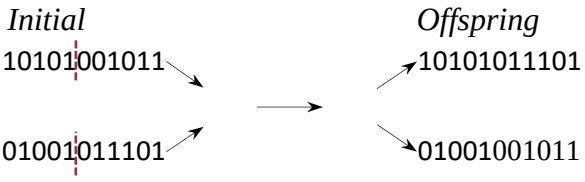

Figure 8. Example of single-point crossover.

Multi-point crossover is similar to single-point crossover, but it works with more than one pivot point. In *uniform crossover*, solution bits are chosen randomly for swapping, which may increase the level of disruption of the parents but allows for better exploration of the search space. In *shuffle crossover*, bits are first randomly shuffled in both parents—but in the same manner. Then a single-point crossover operator by choosing a random pivot point is applied. Bits in the generated offspring are finally unshuffled in the same manner as they were initially shuffled. The effect of this crossover is comparable to that of uniform crossover. *Intermediate crossover*, which is applicable to real variables, generates offspring by taking a weighted average of the parents. It is controlled by a scaling factor selected over a specified interval. This crossover can be adapted to provide sufficient exploration of the search space. In this thesis, intermediate crossover was employed.

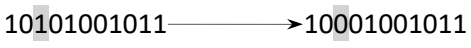

Figure 9. Example of mutation.

2.4.4 Mutation

After crossover, the strings may be subjected to mutation. It is a random process, analogous to biological mutation, where some of the bits of strings in the population are flipped. More specifically, in a mutation step, a fraction of the new population is first selected with uniform probability (not based on fitness), and in each of the selected strings, a random bit is then picked, and its value is inverted (Figure 9). While crossover is meant to leverage the current solutions to find better ones, mutation is intended to aid in exploring the entire search space.

Indeed, the purpose of mutation is to maintain diversity from one population to the next population and to prevent trapping at a local minimum.

3 Instance selection methods for SVMs

This chapter summarizes the contributions of Papers I and II (Aslani and Seipel, 2020, 2021). It accomplishes *research objective I* and addresses *research questions 1* and *2* (*RQ1* and *RQ2*), introduced in Section 1.4.

3.1 Introduction

As datasets grow in size, many systems and algorithms have difficulty analyzing them to generate exploitable knowledge. If a supervised learning algorithm with high time and memory complexity, such as SVMs, is used, this issue can be momentous and even hinder obtaining results (Nalepa and Kawulok, 2018). Reducing the size of datasets by picking a representative subset, called instance selection, is a potential solution that offers two advantages—saving memory and accelerating classification algorithms (Cervantes et al., 2015, Olvera-López et al., 2010b). Samples that are very close to one another (similar samples) or samples far from classes boundaries—called inner samples—often do not contribute to the classification accuracy and can be discarded. In this study, *two new instance selection methods*, named *DRLSH* and *BPLSH*, were proposed. They are inspired by locality-sensitive hashing and have suitable time complexity to handle large datasets. In what follows, each method is explained in detail.

3.2 Instance selection method: DRLSH

3.2.1 Algorithm

DRLSH, the proposed instance selection method, searches for similar instances based on partitioning the space into many buckets in several layers. It consists of two major steps: 1) hashing data points into buckets and 2) measuring similarity and removing similar instances. The first step is to identify the buckets that each instance belongs to using a set of hash function families. Let $G = \{g_1, g_2, \ldots, g_l\}$ be a set of hash function families such that $g(x \in T) = (h_1(x), h_2(x), \ldots, h_k(x))$, where h_i denotes a hash function $h : R^d \rightarrow N$ (Equation 19). $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ is a training dataset in which x denotes a d-dimensional input feature, y denotes the corresponding class, and n denotes the number of training data. Each hash function family g_i splits the input space into a set of smaller regions known as buckets. Specifically, a hash function family produces a layer of non-overlapping d-dimensional buckets. The output of a set of hash function families G with l members can be perceived as l layers of buckets (Figure 10). Indeed, G projects samples from a d-dimensional decimal space to an l-dimensional integer space. Each data point in the new space corresponds to a bucket id in the relevant layer. In the second step, instances that share a bucket with a given data point are retrieved, and their similarity is measured. The similarity index between two instances is defined as the number of buckets they share in all l layers. Closer instances are likely to be hashed in the same buckets in different layers, resulting in a high similarity index

value. Next, instances whose similarity index values to a given instance exceed a predetermined threshold *ST* are eliminated. DRLSH is fast as it does not investigate any possible pairs of samples for similarity measurements.

Figure 10 illustrates the meaning of the similarity index. It contains five layers of hash function families g_i and four data points *A*, *B*, *C*, and *D*. The similarity index between *A* and *B* is four out of five, whereas that between *A* and *C* is two out of 5. This is because the number of identical buckets for *A* and *B* is four and that for *A* and *C* is two. Therefore, *B* is more similar to *A* in comparison to *C* and *D*, and it can be considered dispensable.

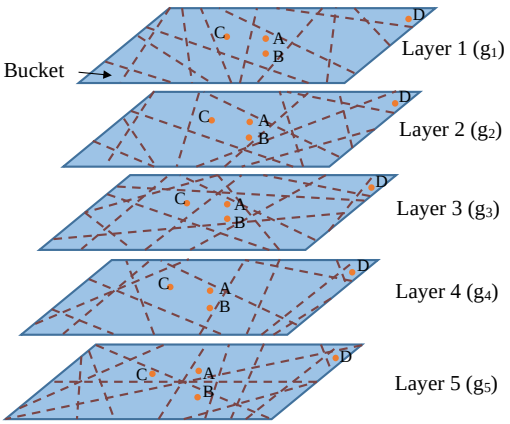

Figure 10. Example of similarity index concept. Different hyperplanes in each layer are representations of h_i.

Algorithm 5 presents the pseudo-code of DRLSH. The inputs to the algorithm are a training dataset *T*, a set of hash function families *G*, and a similarity threshold *ST*. The algorithm is composed of two major loops. The first loop (lines 2–7) computes the bucket id of each instance in all layers, and its time complexity is linear with respect to the number of instances *n* as every instance is included only once. In the second loop (lines 8–26), the samples of class *y* whose number of shared buckets with the current sample *x* is greater than *ST* are regarded as extraneous instances and removed. The time complexity of the second loop is also linear with respect to the number of instances *n*. The number of layers *l*, the number of hash functions *k*, and the similarity threshold *ST* are the parameters that directly contribute to the performance of DRLSH. The source code of DRLSH can be found at `https://github.com/mohaslani/DR.LSH`.

3.2.2 Parameter analyses

This section provides an empirical comparison to demonstrate the effect of input parameters on the performance of DRLSH. This comparison uses a 2D synthetic dataset comprised of 9000 instances with two classes (Figure 11a). Table 1 summarizes the number of preserved samples with respect to *k* and *ST*. For simplicity, the number of layers *l* is set to 10. The preservation rate

Algorithm 5 Pseudo-code of DRLSH, the proposed instance selection method.

Input: A training dataset $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
A set of hash function families $G = \{g_1, g_2, \ldots, g_l\}$ such that $g(x \in T) = (h_1(x), h_2(x), \ldots, h_k(x))$ and h_i is a hash function $(h : R^d \rightarrow N)$
ST denotes a similarity threshold and should not be greater than l
Output: A set of selected instances $(SI \subseteq T)$

```
SI ← ∅
for each x ∈ T do
    for each function family g ∈ G do
        bid ← bucket id assigned to x by g
        Add bid to b_x^g
    end for
end for
for each class y ∈ {y_1, y_2, ..., y_m} do
    S ← {x_i | y_i = y}; % S contains all instances in class y
    for each x ∈ S do
        I ← ∅
        for each function family g ∈ G do
            bid ← b_x^g
            Neighbors ← all instances of class y in bid except x
            Add Neighbors to I
        end for
        for each unique z ∈ I do
            SimilarityIndex ← calculate the frequency of z in I
            if SimilarityIndex >= ST then
                Remove z from S
                Remove z from all buckets
            end if
        end for
    end for
    Add S to SI
end for
```

decreases as buckets become coarser, and ST becomes smaller. It can be seen that high values of k and ST yields high preservation rates. This is because DRLSH becomes more rigorous in identifying similar samples when the values of the input parameters increase. This property enables users to decide the extent to which instances are preserved. Additionally, Table 2 compares the execution time of DRLSH for different values of k and ST when l is set to 10. The computation time increases as the parameters k and ST increase. This is because the execution time of DRLSH is mostly dependent on the number of

preserved samples. The more samples are preserved, the more data points must be passed through in the algorithm, and the longer the execution time becomes. Figure 11b shows the selected instances for $k = 20$, $ST = 4$, and $l = 10$. It is clear that DRLSH successfully preserves the extent of each class.

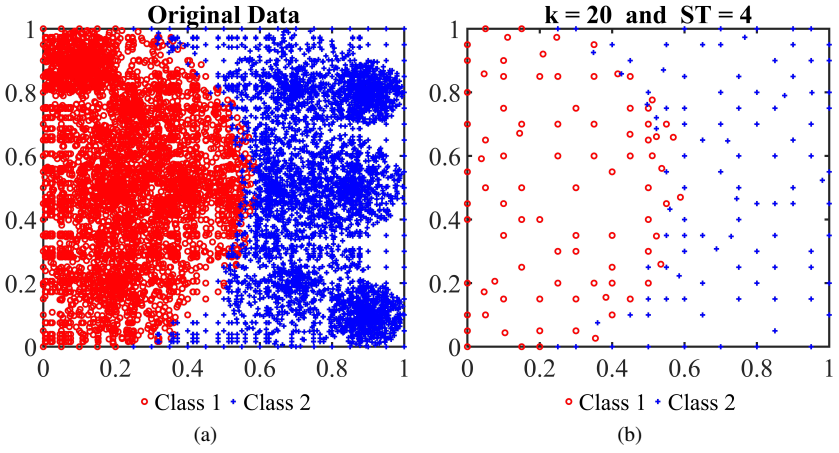

Figure 11. (a) Original dataset, and (b) selected instances for $k = 20$, $ST = 4$, and $l = 10$.

Table 1. Number of instances selected for different values of k and ST ($l = 10$).

Similarity threshold (ST)	Number of hash functions in each layer (k)					
	30	25	20	15	10	5
8	2084	1661	1216	810	432	146
6	851	646	462	289	151	52
4	368	277	191	120	64	25
2	146	111	79	51	29	12

Table 2. Execution time (sec.) for different values of k and ST ($l = 10$).

Similarity threshold (ST)	Number of hash functions in each layer (k)					
	30	25	20	15	10	5
8	0.176	0.140	0.112	0.086	0.064	0.043
6	0.106	0.091	0.080	0.068	0.052	0.036
4	0.092	0.081	0.071	0.060	0.044	0.032
2	0.082	0.073	0.062	0.052	0.040	0.029

3.2.3 Rooftop extraction

The proposed instance selection method is general and can be useful in any data mining application that requires handling large datasets. As an application, it was incorporated into a procedure of pixel-based rooftop extraction using SVMs. The proposed method has also been used in other applications by other

researchers, and readers can refer to (Aydin, 2022, Baldini and Hernandez-Ramos, 2021) to see its performance results.

SVMs as robust classifiers have been widely utilized for rooftop extraction (Gao et al., 2018, Huang and Zhang, 2013, Turker and Koc-San, 2015, Turlapaty et al., 2012, Zhang and Guo, 2007). In the pixel-based rooftop extraction procedure using SVMs, a training dataset based on features of pixels and their labels is first produced. Features should be able to distinguish rooftops from other objects; height and vegetation index are typical features for this purpose. Then, an SVM is trained using the training dataset to predict the labels of other pixels in the area.

The performance of a trained SVM is mainly a function of the instances used for training. The generalization ability and accuracy of SVMs in unseen pixels in test areas are attributable to the employed training instances. The training dataset should be of a suitable size and properly cover the extent of each known class (e.g., roof and non-roof). Otherwise, the training step becomes slow, or the trained SVMs do not become sufficiently accurate. Therefore, selecting a manageable subset of training pixels that provides a representative description of each class and leads to suitable classification accuracy on unseen data is necessary. A prevalent way is to manually select a few small patches of pixels with a suitable distribution as training instances. However, it is time-intensive and might not lead to desirable results. Another commonly used method is randomly selecting a specific number of pixels from each class. Although this strategy is fast, it might not satisfactorily characterize the spectral responses of classes in the feature space. Another way is to choose some pixels systematically. In this context, Jin et al. (2014) evaluated the performance of four conventional sampling methods, including two methods based on *class* stratification and two methods based on *spatial* stratification. The first group of methods considers only the class of samples, and the second group considers both the spatial distribution and class of samples. Although the class and spatial distribution of samples are regarded, their distribution in the *feature space*—critical in training—is overlooked in these methods. To address the mentioned issues, DRLSH was used to choose informative instances (pixels).

3.2.4 Dataset preparation

The performance of DRLSH was evaluated on a test site located in downtown Gothenburg. The employed spatial datasets include a true-orthophoto, an image-derived DSM, LiDAR point clouds, and ground truth data of rooftops. The spatial resolution of both the DSM and true-orthophoto is 10 cm, and the point density of the LiDAR data is 3 points per square meter. All datasets were obtained from the Swedish mapping, cadastral, and land registration authority[1]. Noisy points in the LiDAR data were removed based on the difference between the last and first returns. Moreover, the image-derived DSM was enhanced by

[1] www.lantmateriet.se

the first return of the LiDAR data to ensure no gaps in the surface model. All the collected data were georeferenced in SWEREF99. Figure 12 shows the true-orthophoto of the test site.

To correctly segment rooftops using SVMs, different features for each pixel of the area (as explanatory variables) were defined. These features are as follows: 1) a normalized difference vegetation index (NDVI) estimating vegetation growth and biomass, 2) a normalized DSM (nDSM) (Zhang et al., 2003) showing the height of each pixel, 3) a gradient magnitude image of the DSM (Gonzalez et al., 2020) showing local height changes, 4) a second spatial derivative image of the DSM (Gonzalez et al., 2020) showing enhanced sharp changes of the height values, and 5) a terrain roughness image (Riley et al., 1999) characterizing the heterogeneity in the DSM.

A training dataset was created based on the above features and the labels taken from the ground truth data of rooftops. The dataset consists of 23,750,000 samples with five input features and one output feature. The output feature shows if the sample belongs to a rooftop. Each output class occupies a distinct area in the feature space based on its inherent characteristic and environmental context.

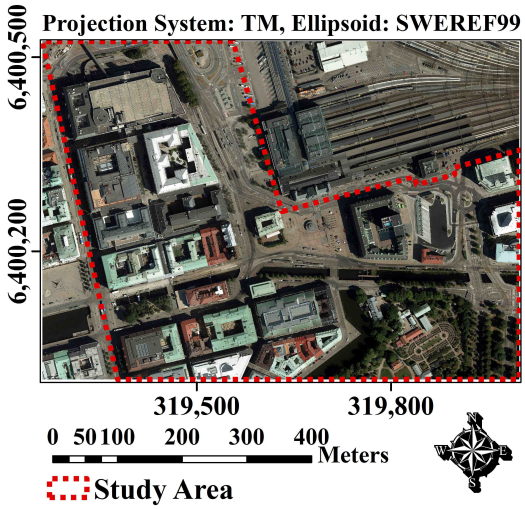

Figure 12. True-orthophoto of the study area.

3.2.5 Results of evaluation and discussion

To evaluate the performance of the proposed method, it was benchmarked against three other methods—random sampling, LSH-IS-S (Arnaiz-González et al., 2016), and PSDSP (Carbonera and Abel, 2018). LSH-IS-S and PSDSP have linear time complexity, making them well-suited for the prepared large dataset. Preservation rate and classification loss are two quantitative metrics used to evaluate the performance of the instance selection methods. These

metrics were estimated using a repeated stratified p-fold cross-validation scheme (Ishibuchi and Nojima, 2013). In this scheme, the dataset is partitioned into p folds, and each time $p-1$ folds are considered a training set—utilized for training SVMs after an instance selection method is applied. The remaining fold is used as a test set to assess the prediction ability of SVMs. The entire process is repeated r times to reduce the influence of partitioning the dataset. The final classification loss and preservation rate are computed by averaging the folds and repetitions. In this study, the parameters p and r were set to 10 and 7, respectively.

Preservation rate and classification loss metrics are calculated using Equations 27 and 28. These two metrics were simultaneously employed because an ideal instance selection method should be able to minimize the preservation rate while maintaining the original classification performance. In these equations, N_{TR} denotes the total number of instances in the training set, N_{STR} denotes the number of selected instances from the training set, x_j denotes the jth sample of the test set, m denotes the number of samples in the test set, f denotes the decision boundary function obtained by the SVM trained on the selected dataset, and $f(x_j)$ denotes the classification score for x_j. Additionally, $c_j \in \{-1,+1\}$ denotes the observed class label, where -1 indicates the negative class (non-roof) and +1 indicates the positive class (roof) (Hastie et al., 2009).

$$Preservation\ Rate = \frac{N_{STR}}{N_{TR}} \times 100 \tag{27}$$

$$Classification\ Loss = \sum_{j=1}^{m} log(1+exp(-f(x_j)c_j)) \tag{28}$$

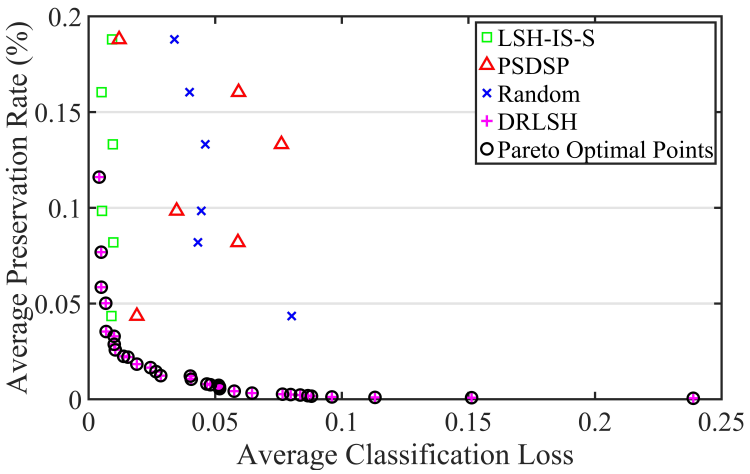

Figure 13. Comparing the performance of LSH-IS-S, PSDSP, Random, and DRLSH.

Figure 13 compares the performance of DRLSH with the other methods regarding the classification loss and preservation rate. The methods were executed

for different input configurations (i.e., different values for input parameters) to make the comparisons fair. Pareto optimality theory (Ehrgott, 2005) was used to choose the best configuration (and method) as there are two conflicting performance criteria. According to the Pareto optimality theory, optimal configurations are the ones that are not dominated by others. Black circles show Pareto points. As can be seen from the figure, DRLSH outperforms the other methods as all the Pareto points are from the group of DRLSH. The impacts of DRLSH on the preservation rate, classification accuracy, and execution time of the SVMs are shown in Table 3. It offers that DRLSH can decrease the number of instances and execution time significantly without considerably impairing the original classification accuracy.

Table 3. Comparing DRLSH with the case that no instance selection method is used.

	Preservation rate (%)	Classification accuracy (%)	Execution time (sec.)
SVM	100	99.99	26915
DRLSH + SVM	0.026	99.98	492

3.3 Instance selection method: BPLSH

3.3.1 Algorithm

BPLSH, the proposed instance selection method, seeks border instances as they are the samples significantly contributing to the construction of decision boundaries. It is based on the idea that a border instance has heterogeneous *neighbors* and is the closest to a sample from an opposite class. The pseudo-code of BPLSH is presented in Algorithm 6. The inputs are a training dataset T and a set of hash function families G, and the output is a subset of selected instances. BPLSH consists of two phases: 1) identifying the buckets of each sample (lines 2–9) and 2) finding border samples and removing non-essential instances (lines 10–42). In the first phase, each instance is assigned to a set of buckets by using a group of hash function families. The time complexity of the first phase is $O(n \times d \times M \times L)$, where n is the number of instances, d is the number of features, M is the number of hash functions, and L is the number of hash function families. The following definitions are first presented to help clarify the second phase:

Definition 1. The *similarity index* between two instances (*SimIndex*) is defined as the number of buckets they share in all layers.

Definition 2. *Neighbors* of a given instance x_i are the instances that share at least one bucket with x_i.

Definition 3. Two instances are *quite close* if their *similarity index* is equal to L, where L is the number of layers.

Definition 4. The *nearest neighbors* of an instance x_i are the *neighbors* that have the maximum *similarity index* with x_i.

The second phase consists of the following six steps:

I Extract the *neighbors* of sample x_i.

II Calculate the *similarity index* of the *neighbors*.

III If x_i and its *neighbors* are homogeneous, save x_i, remove the *neighbors* with *SimIndex*$>= 2$ from the training dataset, and go to step VI.

IV If x_i and its *neighbors* are heterogeneous and the *nearest neighbors* of x_i from opposite classes are *quite close*, save x_i and the *nearest neighbors* from each opposite class and go to step VI.

V If x_i and its *neighbors* are heterogeneous and the *nearest neighbors* of x_i from opposite classes are not *quite close* to it, save the *nearest neighbors* from each opposite class, remove the *quite close neighbors* to x_i with the same class from the dataset, and go to step VI.

VI Repeat steps I to V until all instances in the dataset are either investigated or deleted.

Algorithm 6 Pseudo-code of BPLSH, the proposed instance selection method.

Input: A training dataset $T = \{(x_i, y_i) \,|\, x_i \in R^d, y_i \in \{1,2,\ldots,p\}\}, i = 1,\ldots,t$
A set of hash function families $G = \{g_1, g_2, \ldots, g_L\}$, such that
$g(x \in T) = (h_1(x), h_2(x), \ldots, h_M(x))$, and h_i is a hash function
$(h : R^d \rightarrow N)$
Output: A set of selected instances $(SI \subseteq T)$

$SI, b^g_x, b^g_{bid} \leftarrow \varnothing$
for each $(x, y) \in T$ **do**
  $i \leftarrow$ index of (x, y)
  for each function family $g \in G$ **do**
    $bid \leftarrow$ bucket id assigned to x by g
    Add bid to b^g_i
    Add (x, y) to b^g_{bid}
  end for
end for
for each $(x, y) \in T$ **do**
  $i \leftarrow$ index of (x, y)
  $Neighbors \leftarrow \varnothing$
  for each function family $g \in G$ **do**
    $bid \leftarrow b^g_i$
    $I \leftarrow$ all instances in b^g_{bid} except x
    Add I to $Neighbors$
  end for
  $UNeighbors \leftarrow$ unique elements of $Neighbors$
  $SimIndex \leftarrow$ similarity index of $UNeighbors$ to x
  $C_N \leftarrow$ Classes of $UNeighbors$
  $C_A \leftarrow \{y, C_N\}$
  if C_A is homogeneous **then**

```
        Remove UNeighbors with SimIndex >= 2 from T
        Add (x, y) to SI
    else if C_A is heterogeneous then
        MS ← 0
        for each C ∈ C_N except y do
            N_C ← UNeighbors whose class = C
            SimIndex_C ← SimIndex of N_C
            M ← max(SimIndex_c)
            SN ← N_C with SimIndex_C = M
            Add SN to SI
            MS ← max(MS, M)
        end for
        if MS is equal to L then
            Add (x, y) to SI
        else
            SR ← UNeighbors whose class = y & SimIndex = L
            Remove SR from T
        end if
    end if
end for
```

In step III, only one instance from a homogeneous region is preserved. These regions are not close to the decision boundaries and do not play an important role in classification. Additionally, the *neighbors* of x_i are removed from the dataset to prevent exploring them in subsequent iterations, accelerating the process. To ensure that border instances are preserved, *neighbors* with *SimIndex*>=2 are removed, as *neighbors* with *SimIndex*>=1 may contain some border instances. Steps IV and V process heterogeneous regions to identify border instances. Both steps save the *nearest neighbors* from each opposite class as border instances. The only difference between steps IV and V is the closeness of x_i to border instances. If x_i is *quite close* to border instances, it is preserved as well (step IV). Otherwise, *quite close neighbors* are regarded as interior-redundant instances and removed (step V). The upper bound of the time complexity of all the mentioned steps is $O(n \times \overline{N})$, where $\overline{N}$ is the average number of *neighbors* for each instance. The source code of BPLSH can be found at `https://github.com/mohaslani/BPLSH`.

3.3.2 Parameter analyses

It is illustrated how the input parameters—M and L—affect the resulting subsets. To this end, BPLSH was executed for different values of M and L on a synthetic two-dimensional dataset comprising 30,141 instances with three classes indicated in Figure 14a. Tables 4 and 5 summarize the preservation rate and execution time for $M \in \{20, 60, 100\}$ and $L \in \{10, 20, 30\}$. It is observed that the preservation rate decreases when M or L increases. Furthermore, the execution time of BPLSH on the dataset lowers when M increases. This is

because as the parameter M grows, the bucket sizes reduce, the number of instances with the homogeneous proximity region increases, and fewer samples thus need to be assessed. The selected instances of the synthetic dataset for M = 60 and L = 30 are shown in Figure 14b. It is self-evident that BPLSH removes most interior samples while retaining the patterns placed around the decision borders.

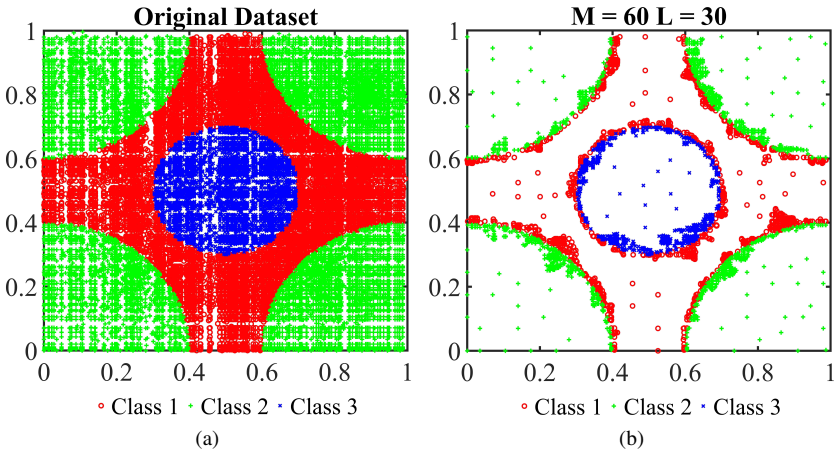

Figure 14. (a) Original dataset, and (b) obtained dataset for M = 60 and L = 30.

Table 4. Preservation rate of BPLSH (%).

Number of hash function families (L)	Number of hash functions (M)		
	20	60	100
10	40.52	18.38	13.61
20	37.16	17.04	11.65
30	33.56	16.13	10.78

Table 5. Execution time of BPLSH (sec.).

Number of hash function families (L)	Number of hash functions (M)		
	20	60	100
10	2.12	1.64	1.56
20	5.52	3.20	2.71
30	9.48	4.76	3.91

3.3.3 Rooftop extraction and data

The effectiveness of BPLSH was verified by applying it to different datasets. However, in this dissertation, only the experiment of rooftop extraction is discussed, and readers can refer to Paper II (Aslani and Seipel, 2021) for the

other experiments. The employed rooftop extraction procedure is similar to that of Section 3.2. The same spatial dataset—including a true-orthophoto, an image-derived DSM, LiDAR point clouds, and ground truth data of rooftops—was employed but in a different area in downtown Gothenburg shown in Figure 15.

A training dataset based on the fusion of the true-orthophoto and the DSM was prepared that consists of about 1,100,000 instances with five input features and one output feature. The input features were formed based on an nDSM, a gradient magnitude of the DSM, a second spatial derivative of the DSM, terrain roughness, and NDVI. The output feature showing the label of points (roof and non-roof) was obtained from the rooftop ground truth data. The high number of training instances necessitates applying instance selection to accelerate the training phase of SVMs.

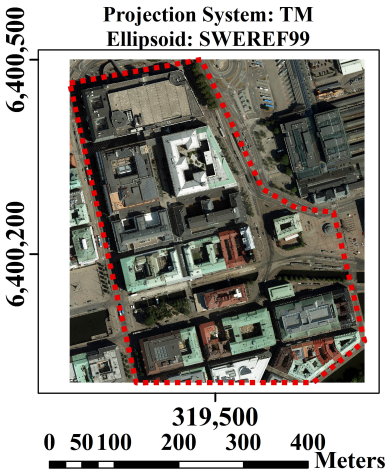

Figure 15. True-orthophoto and boundary of the study area.

3.3.4 Results of evaluation and discussion

BPLSH was benchmarked against three representative instance selection methods, namely NPPS (Shin and Cho, 2007), SVMKM (Barros de Almeida et al., 2000), and CBCH (Birzhandi and Youn, 2019). These methods were chosen owing to their effectiveness and ease of application. All instance selection methods were performed for different input configurations to ensure fair comparisons. The performance of the SVM on the original training set—denoted by NoIS—is also presented to better appreciate the effect of the instance selection methods. Like the previous experiment, all methods were evaluated through a repeated stratified p-fold cross-validation scheme. Moreover, classification errors on test sets and preservation rates were used to evaluate the performance of the instance selection methods. The preservation rate and classification error are calculated according to Equations 27 and 29.

$$Classification\ Error = 100 - \frac{N_{CTE}}{N_{TE}} \times 100 \tag{29}$$

In Equation 29, N_{TE} and N_{CTE} denote the number of instances and the number of correctly classified instances of the test set, respectively. Figure 16a compares the performance of the methods for different values of input parameters in terms of preservation rate and classification error. NPPS leads to the highest classification error, and SVMKM yields the highest preservation rate, but BPLSH adequately balances the two metrics. The Pareto optimality theory was used to identify which method has the best performance. The Pareto set in Figure 16b indicates that BPLSH dominates SVMKM, CBCH, and NoIS. Table 6 also lists the effects of BPLSH on the preservation rate, classification error, and execution time. It is evident that BPLSH significantly reduces the number of instances and execution time by maintaining the original classification accuracy.

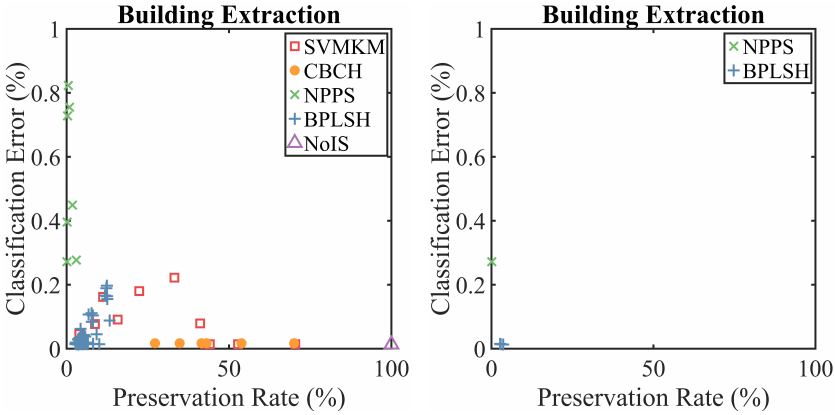

Figure 16. (a) Comparing SVMKM, CBCH, NPPS, BPLSH, and NoIS, (b) Pareto-optimal sets.

Table 6. Effect of BPLSH on preservation rate, classification error, and execution time.

	Preservation rate (%)	Classification error (%)	Execution time (sec.)
SVM	100	0.014	910
BPLSH + SVM	2.834	0.014	509

4 Plane segmentation methods

This chapter summarizes parts of Papers III, IV, and V contributions (Aslani and Seipel, 2022b,a,c). It accomplishes *research objective II* and addresses *research questions 3–6* (*RQ3–RQ6*), introduced in Section 1.4.

4.1 Introduction

In automatic rooftop plane segmentation using DSMs, rooftop pixels are grouped into non-overlapping planar patches. It is a key process for a wide variety of spatial use cases and is particularly critical for determining ideal places for RPVs installation. *Two methods for plane segmentation were proposed*—the first of which is based on clustering, and the second one is based on model fitting. In this chapter, these methods are described, and their performance is assessed.

4.2 Plane segmentation method I

4.2.1 Algorithm

In the first plane segmentation method, initial planar patches are obtained by clustering. To be able to identify planar segments by clustering, three issues should be addressed: 1) defining a suitable feature space, 2) managing unclear cluster boundaries, and 3) maintaining adaptability and computational efficiency of clustering. A suitable feature space should provide the potential for distinctly outlining planar segments. Indeed, the feature space should be defined such that pixels belonging to the same planar segments are mapped close to each other while remaining distant from pixels belonging to other segments. In this study, the feature space was defined by the normal vector of each pixel (pixels have x, y, and z components). This feature space definition is viable as pixels on the same planar segment usually have similar normal vectors. However, mapping *all* pixels of a rooftop in the feature space may make the cluster boundaries of the planar segments obscure (second issue) and may disturb clustering. This is because rooftops can have some pixels with ambiguous normal vectors that are randomly scattered in the feature space and do not match those of nearby pixels. These pixels are usually located in the vicinity of more than one plane or noisy pixels, and they are called non-planar pixels. To address this issue, excluding non-planar pixels from the clustering process is necessary.

Being located in the proximity of multiple planes, the error of plane fitting to the neighborhood of non-planar pixels is substantial. Therefore, non-planar pixels may be identified by fitting planes and measuring the fitting errors. Principal component analysis (PCA) is used to find the best fitting plane. In PCA, a 3D covariance matrix for each pixel p and its neighboring pixels is first calculated. Next, eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ and their corresponding eigenvectors v_1, v_2, and v_3 of the covariance matrix are computed. The eigenvector v_3 approximates the normal vector of a fitted plane whose perpendicular distances

to the points are minimal. The eigenvalue λ_3 contains the error of plane fitting, and the larger it is, the more likely the pixel p and its neighbors are to be non-planar. As a result, pixels whose λ_3 is larger than a predefined threshold are selected as non-planar pixels.

By having planar pixels, initial segments are formed by clustering their normal vectors. Given the possibly large size of rooftops, as well as the fact that the shape and number of clusters vary across them, it is necessary to utilize a clustering algorithm that can handle large datasets and adaptively determine the shape and number of clusters (third issue). MDDC (Pavlidis et al., 2016) introduced in Chapter 2 satisfies these requirements. It is a statistically well-founded clustering algorithm with an optimized computational speed. It has a linear time complexity with respect to the number of points and can automatically determine the shape and number of clusters, making it ideally suited for rooftop plane segmentation.

Each resulting planar patch obtained by clustering may contain multiple parallel or coplanar segments that are not spatially connected. This is because pixels are grouped only based on their normal vectors, and their spatial connectivity is not considered in clustering. Therefore, such planar patches are separated in the original spatial space using Euclidean clustering, introduced in Chapter 2. Finally, the initially excluded non-planar pixels are assigned to the best segments using a new segment growing procedure. It employs the clustering-derived segments as robust seeds, obviating the need for seed selection in traditional segment growing algorithms. In what follows, the steps of the designed segment growing procedure are described. First, patches are sorted by their size, and a plane is fitted to the largest patch. Then, if the fitted plane accurately estimates adjacent non-planar pixels and segments, they are merged with the patch. The growing criteria are based on the point-to-plane distance and the angle between normal vectors. In particular, a point-to-plane distance threshold and an angle threshold are used to check whether adjacent pixels or segments can be added to the current patch. These thresholds should be determined based on the accuracy of the DSM. Next, the neighbors of the expanded patch are evaluated, and the procedure is repeated until no further segments or non-planar pixels are merged. Once the growth of the largest patch is complete, its initial seed and all added segments are excluded from the list. Afterward, the next largest patch is selected, and the above procedure is repeated until no patch remains on the list.

Non-planar pixels are allowed to be assigned to several patches in segment growing, and the best segments for those pixels are determined separately in the end. The distances of each non-planar pixel to the corresponding patches are first calculated and then assigned to the closest segments. This strategy alleviates the effects of the growing order on the assignment of non-planar pixels.

4.2.2 Test sites and data

The proposed plane segmentation method was evaluated in two test sites. The first test site is located in Gothenburg, and it features a heterogeneous mix of buildings with variations in shape and orientation. Its DSM has a spatial resolution of 10 cm and has been generated using image matching by Swedish mapping, cadastral, and land registration authority. The second test site is located in Uppsala and contains a residential neighborhood with detached buildings. Its DSM has a spatial resolution of 15 cm and has been produced from LiDAR point clouds with an average density of 67 pts/m^2. The LiDAR point cloud was kindly provided by Uppsala municipality[2] for non-commercial use. Varied urban morphologies of the test sites along with different dataset properties make the evaluation of plane segmentation more thorough and trustworthy. For more details regarding the test sites, please refer to Paper V (Aslani and Seipel, 2022c).

4.2.3 Results of evaluation and discussion

To do rooftop plane segmentation, rooftop boundaries should be extracted first. The procedures explained in Chapter 3 may be used for this purpose. However, other methods were employed for the sake of variation. The rooftops of the first test site were extracted using an analytical method. Properties such as height, greatest width, and area were used to separate rooftops from other above-ground objects. For more details regarding the method employed for rooftop extraction in the first test site, please refer to Paper III (Aslani and Seipel, 2022b). The rooftops of the second test site were extracted using PointNet++, a deep learning model (Qi et al., 2017). It is a popular neural network for semantic segmentation of unorganized LiDAR point clouds. It allows for multiscale point feature learning and can be trained without requiring parameters specific to objects in point clouds. It captures both the local and global features. Please refer to Paper IV (Aslani and Seipel, 2022a) for further information on extracting rooftops using PointNet++.

Planar segments were identified by applying the proposed method on the extracted rooftops. Figure 17 illustrates some results of plane segmentation. Visual inspection shows that the proposed plane segmentation method is successful in capturing planar segments. Holes in some roof faces occur due to noise or small rooftop superstructures, including vents and small chimneys, that cannot be recognized as independent planar segments.

$$Completeness = \frac{TP}{TP+FN} \tag{30}$$

$$Quality = \frac{TP}{TP+FP+FN} \tag{31}$$

[2] www.uppsala.se/

Figure 17. Plane segmentation results over some rooftops. The true-orthophoto is used only for visualization purposes.

Completeness and quality at the pixel-based level are two metrics used to quantitatively assess the reliability of the proposed plane segmentation method. These two metrics are calculated using Equations 30 and 31. In the equations, *TP*, *FP*, and *FN* denote true positive, false positive, and false negative, respectively, and they are computed by comparing the extracted planar segments with the manually prepared ground truth data (Cai et al., 2018). RANSAC and region growing—extensively employed in plane segmentation—were also applied to benchmark the performance of the proposed method (Grilli et al., 2017, López-Fernández et al., 2015, Xie et al., 2020), but only the results of RANSAC are presented in the thesis. Table 7 summarizes the assessment results for the test sites. According to Rottensteiner et al. (2012, 2014), the proposed plane segmentation method is practically effective and relevant as completeness and quality are greater than 70% in both test sites. Moreover, the proposed method outperforms RANSAC in terms of completeness and quality. It is also noticed that the methods perform better in the Uppsala test site that is ascribed to the rooftop complexity and the DSM accuracy of the Uppsala test site. More specifically, this is because the rooftops are more intricate in the Gothenburg test site, and the height accuracy of the DSM is lower as well.

Table 7. Assessment results in the test sites.

Method	Test site	Completeness (%)	Quality (%)
Proposed method	Gothenburg	84.74	73.90
	Uppsala	98.69	98.22
RANSAC	Gothenburg	50.75	42.16
	Uppsala	96.93	94.43

4.3 Plane segmentation method II

4.3.1 Algorithm

The second method is based on multi-model fitting. The initial segments are identified using a RANSAC-like strategy. Using RANSAC to find planar patches of rooftops is a multi-model fitting problem in the sense that RANSAC must be applied multiple times. In these problems, RANSAC often produces spurious planes across the boundaries of roof faces. This is because RANSAC may use non-planar pixels to generate hypotheses. To avoid this problem, non-planar pixels must first be excluded from the set of pixels used for hypotheses generation in RANSAC. For this purpose PCA can be employed, similar to Section 4.2. However, it is sensitive to small changes and may produce inconsistent normal vectors. To address this limitation, a simple but effective planarity analysis method shown in Algorithm 7 was proposed. This method identifies non-planar pixels by re-estimating PCA normal vectors and comparing them with the original ones. The median estimator is used to re-estimate PCA normal vectors, and as it is a robust estimator, the resulting normal vectors become more consistent and less affected by local changes. A pixel is considered non-planar if one of the angles between its normal vector and those of its neighbors exceeds a predefined threshold.

Algorithm 7 Planarity test.

Input: Pixels of a rooftop PX
PCA normal vectors L
Angle threshold θ
Output: Planar pixels F
Non-planar pixels G

for each pixel $p \in PX$ **do**
  $L^p \leftarrow$ Normal vectors of p from L
  $N \leftarrow [\text{median}(L_x^p), \text{median}(L_y^p), \text{median}(L_z^p)]$
  $N \leftarrow N/\|N\|$
  $\phi \leftarrow$ angles between N and L^p
  if $\max(\phi) < \theta$ **then**
    Add p to F
  else
    Add p to G
  end if
end for

The identified planar pixels are segmented into non-overlapping segments using RANSAC. Locally optimized RANSAC is used instead of classical RANSAC for this purpose. This is because classical RANSAC employs the very minimum sample size, three points, and may not accurately group all pixels in a segment even when the samples are free of outliers. However, locally optimized

RANSAC improves classical RANSAC by performing a local optimization step on the so-far-the-best plane (Lebeda et al., 2012). In the locally optimized RANSAC, in addition to the point-plane distance, the angle between the point normal vector and the plane normal vector is considered in picking inliers to prevent the creation of meaningless segments.

In this step, multiple planar patches that are coplanar yet spatially separated may be grouped into a single segment as spatial connectivity of pixels is not considered in RANSAC. Euclidean clustering is then used to split multi-part patches. Therefore, in each iteration, locally optimized RANSAC followed by Euclidean clustering is applied, and the segmented pixels are excluded from the list of pixels. The procedure is repeated on the remainder of the pixels until no segment can be found. Finally, the non-planar pixels, initially excluded, are assigned back to the best segment using the segment growing procedure explained in Section 4.2.1.

4.3.2 Results of evaluation and discussion

The two test sites introduced in Section 4.2.2, Gothenburg and Uppsala, were used to evaluate the performance of the second proposed plane segmentation method. A visualization of the plane segmentation results on some rooftops is shown in Figure 18. As is seen in the figure, most planar segments are successfully identified. The performance of the method was evaluated in terms of completeness and quality (Equations 30 and 31) and was compared with that of RANSAC. Table 8 summarizes the performance of the methods for both test sites. It suggests that the proposed method has suitable performance and outperforms RANSAC.

Figure 18. Some samples of the plane segmentation results.

Similar to the experiments in Section 4.2, the methods have better performance in the Uppsala test site owing to the DSM accuracy and intricacy of rooftops. Moreover, comparing Tables 7 and 8 shows that the second plane segmentation method has slightly better performance than the first method. However, the second method failed in some cases, one of which is seen in

Figure 19. In this example, the method cannot completely identify a roof face because tree canopies have occluded parts of it.

Table 8. Evaluation results in the test sites.

Method	Test site	Completeness (%)	Quality (%)
Proposed method	Gothenburg	89.75	80.53
	Uppsala	100.00	99.52
RANSAC	Gothenburg	50.75	42.16
	Uppsala	96.93	94.43

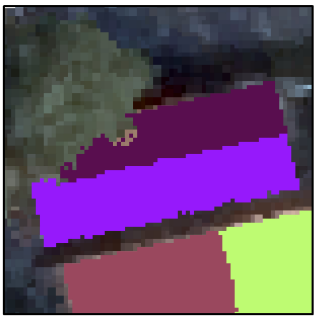

Figure 19. Example of the method failure.

5 Spatially detailed methods for automatic identification of RPV-utilizable areas

This chapter summarizes parts of Papers III, IV, and V contributions (Aslani and Seipel, 2022b,a,c). It accomplishes *research objective III* and addresses *research questions 7* and *8* (*RQ7* and *RQ8*), introduced in Section 1.4.

5.1 Introduction

RPVs harvest sunrays and turn them into energy, and they have been recognized as a potent technology for satisfying part of the energy demand of cities. However, their economic feasibility is contingent on some factors that should be evaluated before deployment. The first factor is the *shadow* cast by nearby objects. Shadow decreases the amount of irradiation reaching rooftop surfaces and can considerably impair the performance of RPVs. The second factor is the *slope* and *aspect* of the underlying surface over which RPVs are mounted. They have a significant impact on how solar panels are exposed to the sun. The third factor is the *size* of the planar segment that RPVs are installed over. Installing RPVs across small segments may require special structures, which increases installation costs. These factors together render rooftops heterogeneous in terms of suitability for placing RPVs as they often consist of multiple planar segments with different *slopes*, *aspects*, and *sizes*. They also contain rooftop superstructures (e.g., chimneys) that cast *shadows* on adjacent segments and cause discontinuity in major planar segments. Therefore, only certain areas of rooftops are cost-effective for installing RPVs. These areas become even more limited when local installation regulations are considered. Determining these areas, referred to as *utilizable* areas, is necessary for a reliable estimate of RPVs energy production. *Two new methods for identifying utilizable areas were proposed.* The methods scrutinize planar segments for their suitability in a spatially detailed manner. Each method contains a new way of analyzing planar segments to find utilizable areas.

5.2 Identification of areas based on morphological operations

This method identifies utilizable areas of each planar segment by applying three constraints—*technical*, *geometric*, and *solar*. As an installation requirement, the *technical constraint* removes service areas—margins between the edges of RPVs and edges of segments—to ease accessibility. The *geometric constraint* excludes parts of segments that cannot accommodate an RPV in terms of dimensions. The *solar constraint* eliminates segments that are insufficiently exposed to the sun due to occlusion or their orientation. Figure 20 illustrates the effects of the constraints. The steps for applying these constraints are further explained in the following sections.

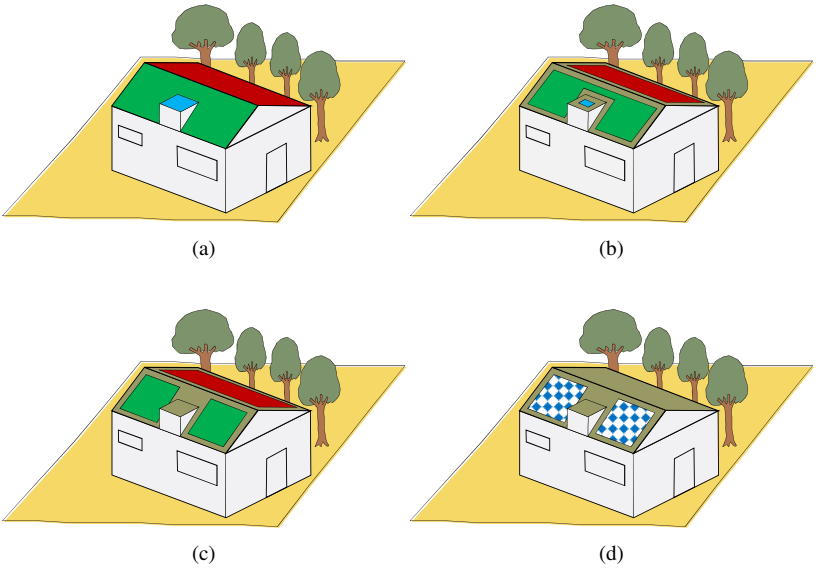

Figure 20. Schematic presentation of the utilizable area extraction procedure. (a) Rooftop planar segments are shown in red, green, and blue. (b) Rooftop areas obtained by applying the *technical constraint*. It removes buffer zones around the edges of segments dedicated to accessibility and safety. (c) Impact of the *geometric constraint*. Areas that cannot accommodate an RPV are excluded. For example, a small area between the dormer and the ridge. (d) Final utilizable areas for RPVs installation after eliminating the segments with a low average amount of solar irradiation (e.g., mainly shadowed). All the steps are fully automated.

5.2.1 3D-2D conversion

Applying these constraints requires further spatial analysis of planar segments. While planar segments generally have three-dimensional geometry in space, they can be considered two-dimensional as they consist of planar points. Analyzing them in a 3D manner adds unnecessary complexities and makes the process computationally demanding. To avoid these problems, 3D planar segments are first converted into 2D. This is done using a coordinate system transformation such that the new z-axis becomes parallel to the normal vector of the planar segment. The transformation consists of rotations around the z and y axes, and it preserves the original shape of 3D segments, such as inner angles, length, and width. Figure 21b indicates a 2D planar segment transformed from 3D.

5.2.2 Technical constraint

Depending on the local installation regulations, there should be a margin between the segment and the RPVs as a service area. These areas are excluded from utilizable areas by applying the technical constraint. Most studies applied the technical constraint in a non-spatial manner, for example, by employing a

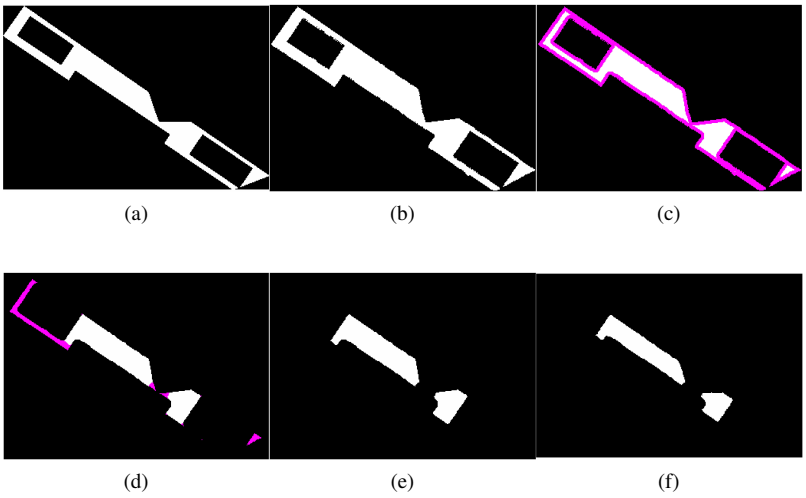

Figure 21. (a) Top view of a 3D planar segment, (b) Segment converted into 2D, (c) Service areas highlighted in pink, (d) Geometrically unsuitable areas in pink, (e) Utilizable areas of the segment, and (f) Top view of the utilizable areas.

constant coefficient. However, this study applied the technical constraint in a spatial manner, i.e., in terms of regions derived from roof faces. The proposed way of removing service areas is by utilizing a morphological erosion operation, one of the fundamental operations in image processing. It is based on a structuring element for probing the segment (Sundararajan, 2017), and it eliminates areas that cannot contain the structuring element completely. The employed structuring element is circular with a radius equal to the width of the requested service area W_{ez}. By applying the erosion operator, the segment is shrunk by the radius of the structuring element (Figure 21c).

5.2.3 Geometric constraint

Some parts of the shrunken segments may not yet provide sufficient room for RPVs, and thus, including them as utilizable areas may lead to overestimating solar energy potential. In most studies, the geometric constraint is either ignored or applied in a naïve manner. A typical way in the literature is to remove segments whose areas are below a specific threshold (Groppi et al., 2018, Hong et al., 2017, Huang et al., 2015). However, this method lacks the flexibility necessary for detailed spatial analysis to find potential subareas of segments. For example, the roof face shown in Figure 22 is quite likely to satisfy any area condition and thus be deemed completely utilizable. However, a portion of it—the cross-hatched area—is too small to accommodate an RPV with a size of 1.7 m $\times$ 1.0 m, and that part must thus be considered non-utilizable.

A new algorithm based on morphological opening operations was proposed to remove geometrically unsuitable parts. An opening operation identifies

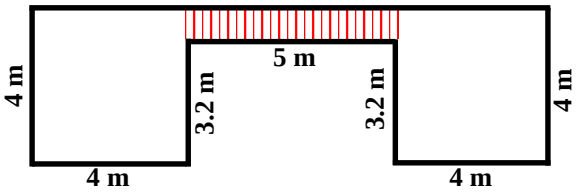

Figure 22. Sample segment. The highlighted part does not offer enough space for installing an RPV with a size of 1.7 m $\times$ 1.0 m.

Algorithm 8 Pseudo-code of the proposed algorithm for applying the geometric constraint.

Input: A shrunken segment RF_T
A solar panel SP with a size of RPV_{size}
A set of rotation angles $\Delta = \{0°, 10°, 20°, \cdots, 170°\}$
Output: A segment without geometrically unsuitable parts RF_{TG}

$RF_{TG} \leftarrow$ a zero matrix with a size of RF_T
for each $\theta \in \Delta$ **do**
  $SP_\theta \leftarrow$ rotate SP with an angle of θ
  $I_\theta \leftarrow O_{SP_\theta}[RF_T]$ $\%O_{SP_\theta}$ is an opening operation with structuring element SP_θ
  $RF_{TG} \leftarrow union(RF_{TG}, I_\theta)$
end for
$RF^c_{TG} \leftarrow connected\ component\ labeling(RF_{TG})$
for each $RF^r_{TG} \in RF^c_{TG}$ **do**
  if $area(RF^r_{TG}) > area(SP)$ **then**
    Preserve RF^r_{TG}
  end if
end for

regions of the segment that may be covered by the structuring element when it is completely enclosed inside the segment. The pseudo-code of the proposed algorithm is presented in Algorithm 8. The algorithm inputs are a segment RF_T obtained from the previous step, a structuring element representing a solar panel SP, and a set of angles Δ for rotating SP. The algorithm iteratively applies a number of opening operations with varying orientations of the structuring element. Indeed, the employed solar panel SP is rotated around its center by $\theta \in \Delta$, and it is used as the structuring element in each iteration. This is because solar panels can be installed with different orientations in practice. By applying the opening operation, the areas of the segment that can offer enough room for a rotated solar panel SP_θ are preserved. The output RF_{TG} is the merging of all suitable regions obtained over the iterations. Finally, segments with an area smaller than that of the solar panel are excluded. Figures 21d and 21e illustrate

geometrically unsuitable regions and obtained utilizable areas.

5.2.4 Solar constraint

In this step, the corresponding 3D forms of the resulting segments from the previous steps are first determined (Figure 21f). Then the segments are evaluated in terms of average solar irradiation level. Utilizable areas must be exposed to an adequate amount of solar irradiation; otherwise, they might not be economically viable. Segments that receive low irradiation are eliminated by selecting an appropriate threshold value SoI_T. Indeed, segments whose average solar irradiation is below SoI_T are excluded from utilizable areas.

5.2.5 Results and discussion

The proposed method was applied to the two test sites introduced in Section 4.2.2, and the results are presented and discussed. The identification of utilizable areas involves estimating solar irradiation maps over rooftops, and the solar model of ArcGIS Desktop was used for this purpose. The solar model estimates the *global* solar irradiation of an area at a given time by considering *direct* and *diffuse* components. The direct component contains solar radiation that reaches the earth's surface directly from the sun without being scattered or absorbed by the atmosphere. The diffuse component comprises sunrays dispersed by molecules and particles in the atmosphere that yet still reach the earth's surface.

To accurately model the direct and diffuse components of solar irradiation, three different maps are computed: a *viewshed map*, a *sun map*, and a *sky map*. A *viewshed map* represents the sky occlusion caused by surrounding objects when looking at the sky from a particular point. It is used to incorporate *shadow effects* necessary for more realistic estimates of solar irradiation. A *sun map* represents discretized sun positions in the sky during a specific period. It is computed depending on the time configuration, temporal granularity (hour interval and day interval), and location of the area. The intersection of the *viewshed* and *sun maps*, showing visible sun positions for a specific location, is used to compute the direct component. A *sky map* represents a subdivided view of the entire sky-dome discretized into a series of sectors along zenith and azimuth. The diffuse component is computed based on the intersection of the *sky map* and the *viewshed map*.

Atmospheric effects are also taken into account in computing the direct and diffuse components. *Transmittance* and *diffuse proportion* are two input parameters of the model describing the atmospheric effects. They have a substantial impact on estimates of solar irradiation. *Transmittance* estimates the average fraction of incoming solar radiation that passes through the atmosphere. It typically ranges from 0.4 for cloudy sky conditions to 0.7 for very clear sky conditions. *Diffuse proportion* estimates the fraction of global normal radiation flux that reaches the surface after being scattered. Its typical values are between 0.2 for very clear skies and 0.7 for dense clouds. These parameters

were calibrated using NASA surface meteorology and solar energy databases[3]. The estimated global solar irradiation in a sample scene is shown in Figure 23. As is evident, shadow effects are considered in the solar irradiation estimation. Additionally, as expected, south-facing roof faces receive the highest solar irradiation.

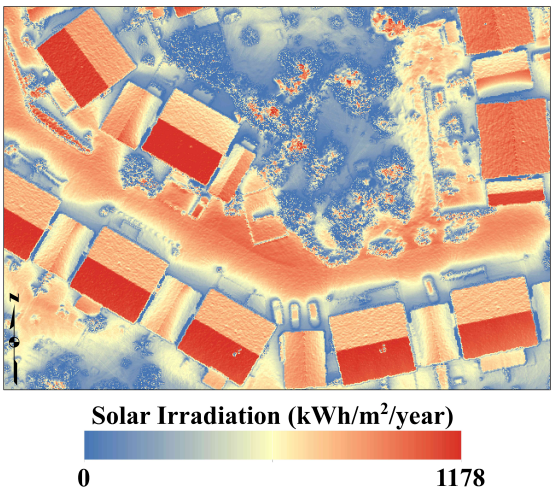

Figure 23. Annual global solar irradiation incident in a sample scene.

The utilizable areas were derived by applying the technical, geometric, and solar constraints. The width of service areas W_{ez}, solar irradiation threshold SoI_T, and size of an RPV were set to 0.3 (m), 1000 (kWh/m^2/year), and 1.7 (m) $\times$ 1.0 (m), respectively. These values can vary depending on different factors such as local regulations, technological-economic advancements of RPVs, location, and anticipated energy needs. Figure 24 shows the resulting utilizable areas in a sample scene. As is seen in the figure, the impacts of minor superstructures, highlighted by white circles, are considered in identifying utilizable areas. Additionally, the proposed method does not choose utilizable areas from north-facing segments, and mostly south-facing rooftop areas are recognized as utilizable. Some large south-facing planar segments might not be considered utilizable—e.g., the rooftop in the bottom right corner in Figure 24—which is owing to shadow effects. It can be inferred that the method explicitly considers the geometry, superstructures, and orientations of rooftops along with shadow effects in identifying utilizable areas.

Table 9 lists the total area (in m^2) and annual electricity yields (in kWh) for rooftops and their utilizable parts in the test sites. The electricity yields are estimated using Equation 32. In this equation, I_V denotes the total amount of global irradiation (in kWh/m^2) that is incident on the segment in a given period,

[3] https://power.larc.nasa.gov/data-access-viewer/

Figure 24. Top view of the identified utilizable rooftop areas of some buildings.

R denotes the spatial resolution of the DSM (in m), ψ denotes the tilt (slope) angle of the segment, α_e denotes the module efficiency, and α_{pr} denotes the performance ratio. Module efficiency shows the percentage of the received solar energy that can be converted into electricity, and it depends on the employed technology of the RPV (Green et al., 2021). The performance ratio describes the relationship between the actual and theoretical energy output of an RPV, and it is a function of environmental and engineering-related factors (e.g., dirt and temperature) (Romero Rodríguez et al., 2017). The module efficiency and performance ratio were set to 0.16 and 0.75, respectively.

$$E = I_V \cdot R^2 \cdot \frac{1}{\cos\psi} \cdot \alpha_e \cdot \alpha_{pr} \tag{32}$$

Table 9. Comparing the rooftop regions with utilizable regions over the test sites.

		Gothenburg test site	Uppsala test site
Area	Entire rooftop regions (m^2)	64417	4181
	Utilizable regions (m^2)	17385	627
	Ratio (%)	27	15
Annual Energy Yield	Entire rooftop regions (kWh)	7085719	402895
	Utilizable regions (kWh)	2319862	80766
	Ratio (%)	33	20

According to Table 9, 27% and 15% of rooftops at the Gothenburg and Uppsala test sites are utilizable, a considerable proportion when it comes to urban scales. The energy yields of utilizable regions are one-third and one-fifth of those of the entire rooftop regions in the test sites, which are significant decreases. It reveals the prominent role of orientation, occlusions, and service areas in rooftop solar energy potential estimates. In other words, it indicates the amount of overestimated energy yield that may result from ignoring the mentioned considerations. It has to be noted that the accuracy of the utilizable

area estimation is attributed to the accuracy of plane segmentation, which is—in turn—a function of the accuracy of the employed data and plane segmentation method. The more accurate the DSM and plane segmentation methods are, the more accurate the estimation of utilizable areas will be.

5.3 Identification of areas considering optimal placement of RPVs

The previous method does not consider the layout of RPVs, and thus, it may lead to some inaccuracies in determining utilizable areas. To address this limitation, a new method based on the placement of RPVs was proposed. It identifies a viable layout of RPVs that maximizes energy production of rooftops while considering the shape, solar irradiation, occlusion, and service areas of rooftop segments. It uses metaheuristic optimization to find the optimal layouts with the highest number of RPVs in highly irradiated areas.

5.3.1 Placement of RPVs

The method consists of two major steps: 1) initial processes and 2) RPVs placement. The first step involves obtaining the required information for designing a layout of RPVs. The boundaries of planar segments are first identified and converted into 2D (Aslani and Seipel, 2022b). Next, service areas are excluded, and solar irradiation distributions over rooftops are computed (Fu and Rich, 2000, Rich et al., 1994). The second step includes determining the optimal layouts of RPVs and computing their energy yields. The optimal layouts are defined as the ones that produce the highest amount of energy while avoiding the placement of RPVs in areas with low irradiation. Determining the optimal layouts requires parametric modeling of RPVs placement and defining a fitness function. A layout of RPVs over a planar segment is defined using the following six variables:

- $O(O_x, O_y)$: initial point for placing RPVs.
- j: orientation of RPVs.
- d: distance between rows of RPVs.
- β: direction of RPVs rows.
- k: distance between adjacent RPVs in one row.

To produce a layout of RPVs over a segment, the bounding box of its polygon derived from the previous step, RF, is first computed. Then, RPVs are placed inside the bounding box based on the chosen values of $O(O_x, O_y)$, β, j, k, and d. Afterward, RPVs that are not entirely inside the polygon RF are eliminated. Finally, the average solar irradiation incident on each RPV is estimated, and the RPVs whose average solar irradiation values are below SoI_T are considered inefficient and removed.

A fitness function evaluates the quality of a produced layout. Different fitness functions may be defined depending on the purpose of the RPVs deployment. In this study, the fitness function is defined based on solar irradiation incident

on RPVs. Particularly, the sum of the average solar irradiation of the RPVs in a layout is regarded as the fitness function. The layout with the highest fitness function value should be chosen as optimal. A genetic algorithm (GA) is used to solve this optimization problem. It is a metaheuristic search algorithm based on Darwin's theory of evolution (Holland, 1992). It iteratively evolves populations of potential solutions using biologically inspired operators, such as mutation and crossover, to converge to the optimal solution. For more details on GAs, please refer to Chapter 2. It should be noted that there are numerous alternative optimization techniques to GAs for this problem (Kennedy and Eberhart, 1995, Mirjalili and Lewis, 2016, Mirjalili et al., 2014, Chu et al., 2006). However, determining which optimization algorithm is best suited to the problem is not within the scope of this study, and the GA is utilized only as a proof of concept. After optimization with the GA, the corresponding 3D coordinates and energy production of the obtained RPVs are computed.

5.3.2 Results and discussion

The method was applied to the test sites described in Section 4.2.2, and the results are briefly presented and discussed. Similar to Section 5.2.5, the solar model of ArcGIS Desktop was used to estimate solar irradiation over rooftops, and the diffuse proportion and transmittance parameters were calibrated using NASA surface meteorology and solar energy databases. The parameters W_{ez} and SoI_T, were respectively set to 0.5 (m) and 1000 (kWh/m^2/year), and the size of an RPV was set to 1.7 (m) $\times$ 1.0 (m). The layout variables (O, j, β, d, k) were determined to maximize the fitness function. The output of the layout optimization in a sample scene is shown in Figure 25. The figure shows that roof geometry, superstructures, and solar irradiation are considered in placing RPVs. For instance, no RPVs are deployed over north-facing segments as expected. Additionally, the lower part of the rooftop in the bottom right corner of the figure is not covered by RPVs. This is owing to the shadow cast by the trees on the south of the rooftop. Due to the objective of the optimization, which is to put more RPVs in highly irradiated regions, it follows that RPVs are placed on rooftops without any gaps between them (i.e., $d = 0$ and $k = 0$).

Table 10. Summary statistics of the rooftop regions and resulting RPVs in the test sites.

		Gothenburg test site	Uppsala test site
Area	Entire rooftop regions (m^2)	64417	4181
	RPV-covered regions (m^2)	11415	532
	Ratio (%)	18	13
Annual Energy Yield	Entire rooftop regions (kWh)	7085719	402895
	RPV-covered regions (kWh)	1527178	68734
	Ratio (%)	22	17

Table 10 compares the entire rooftop and RPV-covered regions in both test sites regarding the area and annual energy yields. The electricity yields

are computed using Equation 32, in which the efficiency α_e and performance ratio α_{pr} of RPVs were set to 0.16 and 0.75, respectively. Regarding the area, RPVs cover about 18% and 13% of the rooftops in the test sites. Comparing the energy yields of rooftops and their RPV-covered regions shows that the proposed method plays a critical role in preventing the overestimation of solar energy potential by 4–6 times. Comparing Tables 9 and 10 also reveals that the first method is overestimating RPVs potential. This is because it computes utilizable areas at the segment level, whereas the second method computes them at the RPV level, allowing for a more thorough and realistic analysis of rooftop areas.

Figure 25. Optimized placement of RPVs in a sample scene.

6 Conclusion and future work

This thesis addressed three issues in classification, segmentation, and spatial analysis in the way of arriving at an accurate estimate of RPVs potential. The first issue was the training of SVMs for classification tasks in large datasets. Despite being one of the most powerful classifiers, SVMs have $O(n^3)$ time complexity, making them unsuitable for large datasets. This is particularly problematic in pixel-based rooftop extraction, where datasets produced from training areas mostly contain large numbers of instances. One potential solution is to select a small subset of training data reflective of the entire set. Most existing methods for this purpose are either prohibitively time-consuming or unsuccessful at balancing the classification accuracy and reduction rate. The second issue was the extraction of planar segments of rooftops, as RPVs are installed based on planar patches. Plane segmentation should be able to extract all planar patches and have a low risk of under-segmenting roof superstructures. This is because rooftop superstructures restrict the placement of RPVs to a considerable extent, and under-segmentation of rooftop superstructures may lead to overestimating RPVs potential. The third issue was finding utilizable areas of each planar segment for RPVs. Accurate identification of utilizable areas is indispensable for a realistic estimate of rooftop solar energy potential. Utilizable areas should meet certain requirements, namely reasonable size for accommodating RPVs, receiving sufficient solar irradiation, and complying with local installation regulations. The task of identification of utilizable areas is inherently complex due to rooftop occlusions, uneven distribution of solar irradiation, and the mentioned requirements for placing RPVs. Therefore, new spatial analyses that can thoroughly scrutinize planar segments by considering the mentioned factors are necessary.

Some methods were proposed for (a) instance selection, (b) plane segmentation, and (c) identification of RPV-utilizable areas. They can assist in the extraction and modeling of rooftops as well as a realistic assessment of RPVs potential. Two new instance selection methods, DRLSH and BPLSH, were proposed on the basis of locality-sensitive hashing. They both identify indispensable samples by analyzing the similarity of data points and their labels. DRLSH aims to identify similar instances of each class, whereas BPLSH aims to identify both similar instances and border instances. DRLSH reduces the number of samples by eliminating similar data points from each class separately. However, in BPLSH, samples from all classes are analyzed together, and only similar samples far from the decision boundaries are eliminated. The time complexity of DRLSH is less than that of BPLSH. However, BPLSH is more likely to preserve the original decision boundaries of datasets as it aims to preserve border instances rather than only eliminate similar instances. To verify their performance, they were benchmarked against some state-of-the-art methods. In one experimental study, they were incorporated into a procedure of pixel-based rooftop extraction from the integration of DSMs and aerial images. The results showed that they outperform the other methods, and thus, they are suitable for

integration into the rooftop extraction procedure.

Two new data-driven methods for plane segmentation in DSMs were proposed, both of which have a suitable level of accuracy. The first method relies mostly on clustering, whereas the second relies primarily on model fitting. In both methods, planarity analysis is incorporated to enhance segmentation. In the first method, initial planar patches are identified by clustering the normal vectors of planar pixels obtained by planarity analysis. Unlike the existing clustering-based methods—requiring non-trivial parameter tuning or even prior knowledge of the number of clusters—the clustering step is adaptive, making it straightforward to apply to any dataset. Additionally, it has optimized computational efficiency, allowing for the handling of high-resolution DSMs. To precisely segment the transition between two planar segments and prevent over-segmentation, modified region growing is applied to the initial planar patches. In the second plane segmentation method that is less sensitive to noise, planar segments are identified by locally optimized RANSAC, followed by modified region growing. It uses a new planarity analysis method to exclude non-planar pixels from RANSAC to circumvent the creation of spurious planes. Unlike PCA, the new planarity analysis method is less sensitive to slight local changes and noise in estimating normal vectors and identifying non-planar pixels. The two proposed methods were benchmarked against some commonly used methods, and the results indicated the superiority of the proposed methods.

Two new spatially detailed methods for the identification of utilizable areas were proposed. Both methods consider installation regulations (service areas), solar irradiation, roof shape, and occlusions in computing the utilizable areas. The first method is based on morphological operations and identifies utilizable areas by excluding service areas, geometrically unsuitable areas, and roof faces with low irradiation. However, the second method identifies utilizable areas by directly placing RPVs on planar segments. It searches for a layout of RPVs that best uses areas of planar segments by utilizing a metaheuristic optimization algorithm. The results showed that the second method provides more flexibility in scrutinizing planar segments and thus leads to more realistic outputs.

Outliers in datasets drastically decrease classification accuracy and lead to poor prediction results. Outliers are data points whose predictive attributes or classes are incorrect. The proposed methods for instance selection are likely to preserve outliers. One potential direction for future work is to design some specific mechanisms/heuristics in DRLSH and BPLSH so that they can recognize outliers while eliminating indispensable samples. Indeed, some outlier-filter stages can be added to make them robust to outliers. In this thesis, the proposed instance selection methods were evaluated on SVMs. k-nearest neighbors is another well-known classifier that—despite its simplicity—has high time complexity. As DRLSH and BPLSH have suitable time complexities, they can be potential instance selection methods for k-nearest neighbors. Therefore, another suggestion for future study is to evaluate the performance of DRLSH and BPLSH on k-nearest neighbors and determine how well they can balance the classification accuracy and reduction rate.

Due to inaccuracy in the input data and plane segmentation methods, the resulting planar patches inevitably have irregular boundaries, causing straight lines to seem jagged. Additionally, this irregularity in boundaries may impair further analysis. To improve the results, planar segments can be transferred into structured polygons using a regularization method that considers the global topology and structure of the rooftop. It can also help with greater transferability across multiple platforms in geoinformatics. In this study, the plane segmentation methods are limited to only DSMs as input, and in the case of having LiDAR point clouds, they should first be converted to DSMs. Although this conversion decreases the amount of input data and simplifies neighborhood definition, it may introduce smoothing effects at sharp objects. Therefore, it would be beneficial to develop the plane segmentation methods so they can also work with LiDAR point clouds.

The estimation of utilizable areas is a function of the spatial resolution of the DSMs. If their resolution decreases, even large rooftop features suffer increasingly from under-segmentation; hence, less importance is given to rooftop features in identifying utilizable areas. In this thesis, only two spatial resolutions, 10 and 15 cm, were evaluated. Another suggestion for future work is to measure the sensitivity of the utilizable area estimation to the DSM spatial resolution and evaluate how much utilizable areas may vary as the spatial resolution of the DSM gradually becomes coarser. Moreover, in this study, RPVs were placed to maximize the energy production of a rooftop while avoiding low-irradiated areas regardless of the energy demand of buildings. The mismatch between the energy production of RPVs and the energy demand of a building leads to unwanted power flow between the household and the grid. If the daily RPVs generation and electricity consumption profiles are different, the building needs to either export a portion of the generated energy back to the grid or import energy from the grid. These situations contribute to financial loss for the owner when the price of power imported is greater than that of export. Therefore, it is important to appropriately balance energy production and energy demand. It would be interesting to develop a method that searches for a layout with the fewest possible RPVs that maximizes the self-consumption of the building. This can result in a reduction in operational energy costs. Another limitation regarding placing RPVs in the developed methods is that they can be placed over any planar patches with suitable solar irradiation and geometry regardless of their roofing material. For instance, RPVs may be placed over glass roofs or large skylight windows, which is not common in practice. To address this limitation, the material composition of each planar segment should also be recognized and considered in the placement procedure. However, the identification of rooftop materials using DSMs is almost impossible as they do not provide any spectral information. One alternative is to use (multispectral)-aerial images and machine learning for the classification of rooftop materials.

References

Abe, S. (2010). *Support vector machines for pattern classification*. Springer, London.

Abe, S. and Inoue, T. (2001). Fast training of support vector machines by extracting boundary data. In Dorffner, G., Bischof, H., and Hornik, K., editors, *International Conference on Artificial Neural Networks*, pages 308–313, Berlin, Heidelberg. Springer.

Akinyelu, A. A. and Ezugwu, A. E. (2019). Nature inspired instance selection techniques for Support Vector Machine speed optimization. *IEEE Access*, 7:154581–154599.

Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT Press, Cambridge, Massachusetts, 4 edition.

Araújo, A. M. and Oliveira, M. M. (2020). A robust statistics approach for plane detection in unorganized point clouds. *Pattern Recognit.*, 100:107115.

Arnaiz-González, Á., Díez-Pastor, J.-F., Rodríguez, J. J., and García-Osorio, C. (2016). Instance selection of linear complexity for big data. *Knowl-Based Syst.*, 107:83–95.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*, pages 1027–1035, New Orleans, Louisiana. Society for Industrial and Applied Mathematics.

Aslani, M. and Seipel, S. (2020). A fast instance selection method for support vector machines in building extraction. *Appl. Soft Comput.*, 97:106716.

Aslani, M. and Seipel, S. (2021). Efficient and decision boundary aware instance selection for support vector machines. *Inf. Sci.*, 577:579–598.

Aslani, M. and Seipel, S. (2022a). A spatially detailed approach to the assessment of rooftop solar energy potential based on LiDAR data. In *The 8th International Conference on Geographical Information Systems Theory, Applications and Management*, pages 56–63. SCITEPRESS.

Aslani, M. and Seipel, S. (2022b). Automatic identification of utilizable rooftop areas in digital surface models for photovoltaics potential assessment. *Appl. Energy*, 306, Part A:118033.

Aslani, M. and Seipel, S. (2022c). Rooftop segmentation and optimization of photovoltaic panel layouts in digital surface models. *Under Review*.

Assouline, D., Mohajeri, N., and Scartezzini, J.-L. (2018). Large-scale rooftop solar photovoltaic technical potential estimation using Random Forests. *Appl. Energy*, 217:189–211.

Aydin, F. (2022). A new instance selection method for enlarging margins between classes. *J. Intell. Syst. Theory Appl.*, 5(2):119–126.

Baldini, G. and Hernandez-Ramos, J. L. (2021). An intrusion detection system implemented with instance selection based on locality sensitive hashing for data reduction. In *26th European Wireless Conference*, pages 1–6, Verona, Italy. IEEE.

Barros de Almeida, M., de Padua Braga, A., and Braga, J. P. (2000). SVM-KM: speeding SVMs learning with a priori cluster selection and k-means. In *Proceedings. Vol.1. Sixth Brazilian Symposium on Neural Networks*, pages 162–167, Rio de Janeiro, Brazil.

Bauer, J., Karner, K., Schindler, K., Klaus, A., and Zach, C. (2005). Segmentation of building from dense 3D point-clouds. In *Proceedings of the ISPRS*, pages 12–14, Enschede, Holland.

Benciolini, B., Ruggiero, V., Vitti, A., and Zanetti, M. (2018). Roof planes detection via a second-order variational model. *ISPRS J. Photogramm. Remote Sens.*, 138:101–120.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D city models: State of the art review. *ISPRS Int. J. Geoinf.*, 4(4):2842–2889.

Birzhandi, P. and Youn, H. Y. (2019). CBCH (clustering-based convex hull) for reducing training time of support vector machine. *J. Supercomput.*, 75(8):5261–5279.

Bódis, K., Kougias, I., Jäger-Waldau, A., Taylor, N., and Szabó, S. (2019). A high-resolution geospatial assessment of the rooftop solar photovoltaic potential in the European Union. *Renew. Sust. Energ. Rev.*, 114:109309.

Boley, D. and Cao, D. (2004). Training support vector machine using adaptive clustering. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, page 126–137, Florida, USA.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, Pennsylvania. ACM.

Cai, L., Shi, W., Miao, Z., and Hao, M. (2018). Accuracy assessment measures for object extraction from remote sensing images. *Remote Sens.*, 10(2):303.

Carbonera, J. L. and Abel, M. A. (2018). An efficient prototype selection algorithm based on dense spatial partitions. In Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., and Zurada, J. M., editors, *Artificial Intelligence and Soft Computing*, pages 288–300. Springer, Cham.

Castillo, E., Liang, J., and Zhao, H. (2013). Point cloud segmentation and denoising via constrained nonlinear least squares normal estimates. In Breuß, M., Bruckstein, A., and Maragos, P., editors, *Innovations for Shape Analysis: Models and Algorithms*, pages 283–299. Springer, Heidelberg.

Cervantes, J., García Lamont, F., López-Chau, A., Rodríguez Mazahua, L., and Sergio Ruíz, J. (2015). Data selection based on decision tree for SVM classification on large data sets. *Appl. Soft Comput.*, 37:787–798.

Cervantes, J., Li, X., and Yu, W. (2006). Support vector machine classification based on fuzzy clustering for large data sets. In Gelbukh, A. and Reyes-Garcia, C. A., editors, *MICAI 2006: Advances in Artificial Intelligence*, pages 572–582, Berlin, Heidelberg. Springer.

Cervantes, J., Li, X., Yu, W., and Li, K. (2008). Support vector machine classification for large data sets via minimum enclosing ball clustering. *Neurocomputing*, 71(4):611–619.

Chang, F., Guo, C.-Y., Lin, X.-R., Liu, C.-C., and Lu, C.-J. (2010). Tree decomposition for large-scale SVM problems. In *2010 International Conference on Technologies and Applications of Artificial Intelligence*, pages 233–240, Hsinchu, Taiwan. IEEE.

Chen, D., Zhang, L., Mathiopoulos, P. T., and Huang, X. (2014). A methodology for automated segmentation and reconstruction of urban 3-D buildings from ALS point clouds. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 7(10):4199–4217.

Christmann, A. and Steinwart, I. (2008). *Support Vector Machines*. Springer, New York, NY.

Chu, S.-C., Tsai, P.-W., and Pan, J.-S. (2006). Cat swarm optimization. In *Proceedings of the 9th Pacific Rim International Conference on Artificial Intelligence*, PRICAI'06, page 854–858, Guilin, China. Springer.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Mach. Learn.*, 20(3):273–297.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 253–262, New York, USA. ACM.

de Vries, T. N. C., Bronkhorst, J., Vermeer, M., Donker, J. C. B., Briels, S. A., Ziar, H., Zeman, M., and Isabella, O. (2020). A quick-scan method to assess photovoltaic rooftop potential based on aerial imagery and LiDAR. *Sol. Energy*, 209:96–107.

Deschaud, J.-E. and Goulette, F. (2010). A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing. In *3DPVT*, Paris, France.

Dixit, M., Chaurasia, K., and Kumar Mishra, V. (2021). Dilated-ResUnet: A novel deep learning architecture for building extraction from medium resolution multi-spectral satellite imagery. *Expert Syst. Appl.*, 184:115530.

Dong, Z., Yang, B., Hu, P., and Scherer, S. (2018). An efficient global energy optimization approach for robust 3D plane segmentation of point clouds. *ISPRS J. Photogramm. Remote Sens.*, 137:112–133.

Dougherty, G. (2013). *Pattern Recognition and Classification an Introduction*. Springer, New York.

Ehrgott, M. (2005). *Multicriteria Optimization*. Springer, Berlin Heidelberg.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *the Second International Conference on Knowledge Discovery in Databases and Data Mining*, pages 226–231, Portland. AAAI Press.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Assoc. Comput. Machinery*, 24(6):381–395.

Foody, G. M. and Mathur, A. (2006). The use of small training sets containing mixed pixels for accurate hard image classification: Training on mixed spectral responses for classification by a SVM. *Remote Sens. Environ.*, 103(2):179–189.

Fu, P. and Rich, P. M. (2000). The Solar Analyst 1.0 Manual. Technical report, Helios Environmental Modeling Institute (HEMI), USA.

Gao, X., Wang, M., Yang, Y., and Li, G. (2018). Building extraction from RGB VHR images using Shifted Shadow algorithm. *IEEE Access*, 6:22034–22045.

Garcia, S., Derrac, J., Cano, J., and Herrera, F. (2012). Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(3):417–435.

Gassar, A. A. A. and Cha, S. H. (2021). Review of geographic information systems-based rooftop solar photovoltaic potential estimation approaches at urban scales. *Appl. Energy*, 291:116817.

Gawley, D. and McKenzie, P. (2022). Investigating the suitability of GIS and remotely-sensed datasets for photovoltaic modelling on building rooftops. *Energy Build.*, 265:112083.

Gilani, S. A. N., Awrangjeb, M., and Lu, G. (2018). Segmentation of airborne point cloud data for automatic building roof extraction. *GISci. Remote Sens.*, 55(1):63–89.

Gonzalez, R. C., Woods, R. E., and Eddins, S. L. (2020). *Digital Image Processing Using MATLAB*. Pearson Prentice Hall, New Jersey, 3rd edition.

Green, M., Dunlop, E., Hohl-Ebinger, J., Yoshita, M., Kopidakis, N., and Hao, X. (2021). Solar cell efficiency tables (Version 58). *Prog. Photovolt. Res. Appl.*, 29(7):657– 667.

Grilli, E., Menna, F., and Remondino, F. (2017). A review of point clouds segmentation and classification algorithms. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLII-2/W3:339–344.

Groppi, D., de Santoli, L., Cumo, F., and Astiaso Garcia, D. (2018). A GIS-based model to assess buildings energy consumption and usable solar energy potential in urban areas. *Sustain. Cities Soc.*, 40:546–558.

Guo, L. and Boukir, S. (2015). Fast data selection for SVM training using ensemble margin. *Pattern Recognit. Lett.*, 51:112–119.

Guo, L., Boukir, S., and Chehata, N. (2010). Support vectors selection for supervised learning using an ensemble approach. In *20th International Conference on Pattern Recognition*, pages 37–40, Istanbul, Turkey. IEEE.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning. Data mining, inference, and prediction.* Springer, New York.

Haupt, R. L. and Haupt, S. E. (2004). *Practical Genetic Algorithms*. John Wiley and Sons, Inc, New Jersey, second edition.

Holland, J. H. (1992). Genetic Algorithms. *Sci. Am.*, 267(1):66–73.

Hong, T., Lee, M., Koo, C., Jeong, K., and Kim, J. (2017). Development of a method for estimating the rooftop solar photovoltaic (PV) potential by analyzing the available rooftop area using Hillshade analysis. *Appl. Energy*, 194:320–332.

Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural. Netw.*, 13(2):415–425.

Huang, X. and Zhang, L. (2013). An SVM ensemble approach combining spectral, structural, and semantic features for the classification of high-resolution remotely sensed imagery. *IEEE Trans. Geosci. Remote. Sens.*, 51(1):257–272.

Huang, Y., Chen, Z., Wu, B., Chen, L., Mao, W., Zhao, F., Wu, J., Wu, J., and Yu, B. (2015). Estimating roof solar energy potential in the downtown area using a GPU-accelerated solar radiation model and airborne LiDAR data. *Remote Sens.*, 7(12):17212–17233.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC 98: Proceedings of the 30th annual ACM symposium on theory of computing*, pages 604–613, Texas, USA. ACM.

Ishibuchi, H. and Nojima, Y. (2013). Repeated double cross-validation for choosing a single solution in evolutionary multi-objective fuzzy classifier design. *Knowl-Based Syst.*, 54:22–31.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323.

Jin, H., Stehman, S. V., and Mountrakis, G. (2014). Assessing the impact of training sample selection on accuracy of an urban classification: A case study in Denver, Colorado. *Int. J. Remote Sens.*, 35(6):2067–2081.

Jochem, A., Höfle, B., Rutzinger, M., and Pfeifer, N. (2009). Automatic roof plane detection and analysis in airborne LiDAR point clouds for solar potential assessment. *Sensors*, 9(7):5241–5262.

Jochem, A., Höfle, B., Wichmann, V., Rutzinger, M., and Zipf, A. (2012). Area-wide roof plane segmentation in airborne LiDAR point clouds. *Comput. Environ. Urban Syst.*, 36(1):54–64.

Joshi, S., Mittal, S., Holloway, P., Shukla, P. R., Ó Gallachóir, B., and Glynn, J. (2021). High resolution global spatiotemporal assessment of rooftop solar photovoltaics potential for renewable electricity generation. *Nat. Commun.*, 12(1):5738.

Jung, S., Jeoung, J., Kang, H., and Hong, T. (2021). Optimal planning of a rooftop PV system using GIS-based reinforcement learning. *Appl. Energy*, 298:117239.

Katoch, S., Chauhan, S. S., and Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimed. Tools. Appl.*, 80(5):8091–8126.

Kaufmann, L. (1999). Solving the Quadratic Programming problem arising in support vector classification. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods: Support Vector Learning*, pages 147–168. MIT Press, Cambridge.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948.

Klasing, K., Wollherr, D., and Buss, M. (2008). A clustering method for efficient segmentation of 3D laser data. In *2008 IEEE International Conference on Robotics and Automation*, pages 4043–4048, Pasadena, California, USA.

Koggalage, R. and Halgamuge, S. (2004). Reducing the number of training samples for fast support vector machine classification. *Neural Inform. Process. Lett. Rev.*, 2:57–65.

Kubat, M. (2017). *An Introduction to Machine Learning*. Springer, Cham, 2 edition.

Lebeda, K., Matas, J., and Chum, O. (2012). Fixing the locally optimized RANSAC. In *British Machine Vision Conference*, Guildford, United Kingdom.

Leskovec, J., Rajaraman, A., and Ullman, J. (2014). *Mining of Massive Datasets*. Cambridge University Press, Cambridge, England, third edition.

Li, G., Xuan, Q., Akram, M. W., Golizadeh Akhlaghi, Y., Liu, H., and Shittu, S. (2020). Building integrated solar concentrating systems: A review. *Appl. Energy*, 260:114288.

Li, L., Yang, F., Zhu, H., Li, D., Li, Y., and Tang, L. (2017). An improved RANSAC for 3D point cloud plane segmentation based on normal distribution transformation cells. *Remote Sens.*, 9(5):433.

Liu, C., Wang, W., Wang, M., Lv, F., and Konan, M. (2017). An efficient instance selection algorithm to reconstruct training set for support vector machine. *Knowl-Based Syst.*, 116:58–73.

Liu, Y.-G., Chen, Q., and Yu, R.-Z. (2003). Extract candidates of support vector from training set. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics*, volume 5, pages 3199–3202, Xi'an, China. IEEE.

López Chau, A., Li, X., and Yu, W. (2013). Convex and concave hulls for classification with support vector machine. *Neurocomputing*, 122:198–209.

López-Fernández, L., Lagüela, S., Picón, I., and González-Aguilera, D. (2015). Large scale automatic analysis and classification of roof surfaces for the installation of solar panels using a multi-sensor aerial platform. *Remote Sens.*, 7(9):1226–11248.

Lukač, N., Špelič, D., Štumberger, G., and Žalik, B. (2020). Optimisation for large-scale photovoltaic arrays' placement based on Light Detection And Ranging data. *Appl. Energy*, 263:114592.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, California, USA.

Mainzer, K., Killinger, S., McKenna, R., and Fichtner, W. (2017). Assessment of rooftop photovoltaic potentials at the urban level using publicly available geodata and image recognition techniques. *Sol. Energy*, 155:561–573.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England.

Maulik, U. and Chakraborty, D. (2017). Remote sensing image classification: A survey of support-vector-machine-based advanced techniques. *IEEE Geosci. Remote. Sens. Mag.*, 5(1):33–52.

Meila, M. (2006). The uniqueness of a good optimum for k-means. In *Proceedings of the 23 rd International Conference on Machine Learning*, pages 625–632, Pittsburgh, Pennsylvania.

Mirjalili, S. and Lewis, A. (2016). The whale optimization algorithm. *Adv. Eng. Softw.*, 95:51–67.

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61.

Mitchell, M. (1996). *An introduction to genetic algorithms*. MIT Press, Cambridge.

Mohajeri, N., Assouline, D., Guiboud, B., Bill, A., Gudmundsson, A., and Scartezzini, J.-L. (2018). A city-scale roof shape classification using machine learning for solar energy applications. *Renew. Energy*, 121:81–93.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, Massachusetts.

Nalepa, J. and Kawulok, M. (2018). Selecting training sets for support vector machines: A review. *Artif. Intell. Rev.*, 52(2):857–900.

Nelson, J. R. and Grubesic, T. H. (2020). The use of LiDAR versus unmanned aerial systems (UAS) to assess rooftop solar energy potential. *Sustain. Cities Soc.*, 61:102353.

Nguyen, A. and Le, B. (2013). 3D point cloud segmentation: A survey. In *6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 225–230, Manila, Philippines. IEEE.

Olvera-López, J. A., Carrasco-Ochoa, J. A., and Martínez-Trinidad, J. F. (2010a). A new fast prototype selection method based on clustering. *Pattern Anal. Appl.*, 13(2):131–141.

Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., and Kittler, J. (2010b). A review of instance selection methods. *Artif. Intell. Rev.*, 34(2):133–143.

Pavlidis, N. G., Hofmeyr, D. P., and Tasoulis, S. K. (2016). Minimum density hyperplanes. *J. Mach. Learn. Res.*, 17(156):1–33.

Pavlov, D., Mao, J., and Dom, B. (2000). Scaling-up support vector machines using boosting algorithm. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 219–222, Barcelona. IEEE.

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods: Support Vector Learning*, pages 185–208. MIT Press, Cambridge.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, pages 5099–5108, California.

Rabbani, T., van den Heuvel, F. A., and Vosselmann, G. (2006). Segmentation of point clouds using smoothness constraint. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, 36:248–253.

Raguram, R., Chum, O., Pollefeys, M., Matas, J., and Frahm, J.-M. (2013). USAC: A Universal Framework for Random Sample Consensus. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):2022–2038.

Ren, H., Xu, C., Ma, Z., and Sun, Y. (2022). A novel 3D-geographic information system and deep learning integrated approach for high-accuracy building rooftop solar energy potential characterization of high-density cities. *Appl. Energy*, 306:117985.

Rich, P. M., Dubayah, R. O., Hetrick, W. A., and Saving, S. C. (1994). Using Viewshed models to calculate intercepted solar radiation: Applications in ecology. *Am. Soc. Photogramm. Remote Sens. Tech. Pap.*, pages 524–529.

Riley, S. J., Degloria, S. D., and Elliot, R. (1999). A terrain ruggedness index that quantifies topographic heterogeneity. *Int. J. Sci.*, 5:23–27.

Romero Rodríguez, L., Duminil, E., Sánchez Ramos, J., and Eicker, U. (2017). Assessment of the photovoltaic potential at urban level based on 3D city models: A case study and new methodological approach. *Sol. Energy*, 146:264–275.

Rottensteiner, F., Sohn, G., Gerke, M., Wegner, J. D., Breitkopf, U., and Jung, J. (2014). Results of the ISPRS benchmark on urban object detection and 3D building reconstruction. *ISPRS J. Photogramm. Remote Sens.*, 93:256–271.

Rottensteiner, F., Sohn, G., Jung, J., Gerke, M., Baillard, C., Bénitez, S., and Breitkopf, U. (2012). The ISPRS benchmark on urban object classification

and 3D building reconstruction. In *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 293–298, Melbourne, Australia.

Sampath, A. and Shan, J. (2010). Segmentation and reconstruction of polyhedral building roofs from aerial LiDAR point clouds. *IEEE Trans. Geosci. Remote. Sens.*, 48(3):1554–1567.

Sánchez-Aparicio, M., Martín-Jiménez, J., Del Pozo, S., González-González, E., and Lagüela, S. (2021). Ener3DMap-SolarWeb roofs: A geospatial web-based platform to compute photovoltaic potential. *Renew. Sust. Energ. Rev.*, 135:110203.

Schmidt, K., Behrens, T., and Scholten, T. (2008). Instance selection and classification tree analysis for large spatial datasets in digital soil mapping. *Geoderma*, 146(1):138–146.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for point-cloud shape detection. *Comput. Graph. Forum*, 26:214–226.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2016). Taking the human Out of the loop: A review of Bayesian optimization. *Proc. IEEE*, 104(1):148–175.

Shen, X.-J., Mu, L., Li, Z., Wu, H.-X., Gou, J.-P., and Chen, X. (2016). Large-scale support vector machine classification with redundant data reduction. *Neurocomputing*, 172:189–197.

Shin, H. and Cho, S. (2002). Pattern selection for support vector classifiers. In Yin, H., Allinson, N., Freeman, R., Keane, J., and Hubbard, S., editors, *International Conference on Intelligent Data Engineering and Automated Learning*, pages 469–474, Berlin, Heidelberg. Springer.

Shin, H. and Cho, S. (2007). Neighborhood property-based pattern selection for support vector machines. *Neural Comput.*, 19(3):816–855.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. CRC press, London.

Sivanandam, S. and Deepa, S. (2008). *Introduction to Genetic Algorithms*. Springer, Berlin, Heidelberg, 1st edition.

Sun, T., Shan, M., Rong, X., and Yang, X. (2022). Estimating the spatial distribution of solar photovoltaic power generation potential on different types of rural rooftops using a deep learning network applied to satellite images. *Appl. Energy*, 315:119025.

Sundararajan, D. (2017). *Digital Image Processing A Signal Processing and Algorithmic Approach*. Springer, Singapore.

Szabó, S., Enyedi, P., Horváth, M., Kovács, Z., Burai, P., Csoknyai, T., and Szabó, G. (2016). Automated registration of potential locations for solar energy production with Light Detection And Ranging (LiDAR) and small format photogrammetry. *J. Clean. Prod.*, 112:3820–3829.

Thebault, M., Clivillé, V., Berrah, L., and Desthieux, G. (2020). Multicriteria roof sorting for the integration of photovoltaic systems in urban environments. *Sustain. Cities Soc.*, 60:102259.

Turker, M. and Koc-San, D. (2015). Building extraction from high-resolution optical spaceborne images using the integration of support vector machine (SVM) classification, Hough transformation and perceptual grouping. *Int. J. Appl. Earth Obs.*, 34:58–69.

Turlapaty, A., Gokaraju, B., Du, Q., Younan, N. H., and Aanstoos, J. V. (2012). A hybrid approach for building extraction from spaceborne multi-angular optical imagery. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 5(1):89–100.

Udell, M. and Toole, O. (2019). Optimal design of efficient rooftop photovoltaic arrays. *INFORMS J. Appl. Anal.*, 49(4):281–294.

Vapnik, V. N. (1998). *Statistical learning theory*. John Wiley and Sons, New York.

Vo, A.-V., Truong-Hong, L., Laefer, D. F., and Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS J. Photogramm. Remote Sens.*, 104:88–100.

Walch, A., Castello, R., Mohajeri, N., and Scartezzini, J.-L. (2020). Big data mining for the estimation of hourly rooftop photovoltaic potential and its uncertainty. *Appl. Energy*, 262:114404.

Wang, D. and Shi, L. (2008). Selecting valuable training samples for SVMs via data structure analysis. *Neurocomputing*, 71(13):2772–2781.

Wang, J., Neskovic, P., and Cooper, L. N. (2007). Selecting data for fast support vector machines training. In Chen, K. and Wang, L., editors, *Trends in Neural Computation*, pages 61–84. Springer, Berlin, Heidelberg.

Wang, R. and Kwong, S. (2010). Sample selection based on maximum entropy for support vector machines. In *2010 International Conference on Machine Learning and Cybernetics*, volume 3, pages 1390–1395, Qingdao, China. IEEE.

Wang, R., Peethambaran, J., and Chen, D. (2018). LiDAR point clouds to 3-D urban models: A review. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 11(2):606–627.

Wu, A. N. and Biljecki, F. (2021). Roofpedia: Automatic mapping of green and solar roofs for an open roofscape registry and evaluation of urban sustainability. *Landsc. Urban Plan.*, 214:104167.

Xiao, J., Zhang, J., Adler, B., Zhang, H., and Zhang, J. (2013). Three-dimensional point cloud plane segmentation in both structured and unstructured environments. *Rob. Auton. Syst.*, 61(12):1641–1652.

Xie, Y., Tian, J., and Zhu, X. X. (2020). Linking points with labels in 3D: A review of point cloud semantic segmentation. *IEEE Geosci. Remote. Sens. Mag.*, 8(4):38–59.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Trans. Neural. Netw.*, 16(3):645–678.

Xu, Y. and Stilla, U. (2021). Toward building and civil infrastructure reconstruction from point clouds: A review on data and key techniques. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, 14:2857–2885.

Yildirim, D., Büyüksalih, G., and Şahin, A. D. (2021). Rooftop photovoltaic potential in Istanbul: Calculations based on LiDAR data, measurements and verifications. *Appl. Energy*, 304:117743.

Yu, H., Yang, J., and Han, J. (2003). Classifying large data sets using SVMs with hierarchical clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 306–315, Washington. ACM.

Yu, J., Tang, Y. M., Chau, K. Y., Nazar, R., Ali, S., and Iqbal, W. (2022). Role of solar-based renewable energy in mitigating CO2 emissions: Evidence from quantile-on-quantile estimation. *Renew. Energy*, 182:216–226.

Zeng, Z.-Q., Xu, H.-R., Xie, Y.-Q., and Gao, J. (2008). A geometric approach to train SVM on very large data sets. In *3rd International Conference on Intelligent System and Knowledge Engineering*, volume 1, pages 991–996, Xiamen. IEEE.

Zhang, K., Chen, S.-C., Whitman, D., Shyu, M.-L., Yan, J., and Zhang, C. (2003). A progressive morphological filter for removing nonground measurements from airborne LiDAR data. *IEEE Trans. Geosci. Remote. Sens.*, 41(4):872–882.

Zhang, Q.-J. and Guo, L. (2007). Self-enhanced SVM extraction of building objects from high resolution satellite images. In *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, pages 13–16, Kumamoto, Japan. IEEE.

Zheng, Y. and Weng, Q. (2015). Model-driven reconstruction of 3-D buildings using LiDAR data. *IEEE Geosci. Remote Sens. Lett.*, 12(7):1541–1545.

Zhong, Q., Nelson, J. R., Tong, D., and Grubesic, T. H. (2022). A spatial optimization approach to increase the accuracy of rooftop solar energy assessments. *Appl. Energy*, 316:119128.

Zhong, Q. and Tong, D. (2020). Spatial layout optimization for solar photovoltaic (PV) panel installation. *Renew. Energy*, 150:1–11.

Zhong, T., Zhang, Z., Chen, M., Zhang, K., Zhou, Z., Zhu, R., Wang, Y., Lü, G., and Yan, J. (2021). A city-scale estimation of rooftop solar photovoltaic potential based on deep learning. *Appl. Energy*, 298:117132.

Zhu, Z., Wang, Z., Li, D., and Du, W. (2020). NearCount: Selecting critical instances based on the cited counts of nearest neighbors. *Knowl-Based Syst.*, 190:105196.

Part II
Papers reprints

Papers
Associated papers have been removed in the electronic version of this thesis.
For more details about the papers see:
http://urn.kb.se/resolve?urn=urn:nbn:se:hig:diva-39741