OBJECT DETECTION USING OPTICAL AND LIDAR DATA FUSION WITH GRAPH-CUTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Onur Taşar
March 2017

OBJECT DETECTION USING OPTICAL AND LIDAR DATA FUSION WITH GRAPH-CUTS

By Onur Taşar

March 2017

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Selim Aksoy(Advisor)

Aydın Alatan

Ramazan Gökberk Cinbiş

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

OBJECT DETECTION USING OPTICAL AND LIDAR DATA FUSION WITH GRAPH-CUTS

Onur Taşar
M.S. in Computer Engineering
Advisor: Selim Aksoy
March 2017

Object detection in remotely sensed data has been a popular problem and is commonly used in a wide range of applications in domains such as agriculture, navigation, environmental management, urban monitoring and mapping. However, using only one type of data source may not be sufficient to solve this problem. Fusion of aerial optical and LiDAR data has been a promising approach in remote sensing as they carry complementary information for object detection. We propose frameworks that partition the data in multiple levels and detect objects with minimal supervision in the partitioned data. Our methodology involves thresholding the data according to height, and dividing the data into smaller components to process it efficiently in the preprocessing step. For the classification task, we propose two graph cut based procedures that detect objects in each component using height information from LiDAR, spectral information from aerial data, and spatial information from adjacency maps. The first procedure provides a binary classification, whereas the second one performs a multi-class classification. We use the first framework to separate buildings from trees in the high pixels, and roads from grass areas in the low pixels. The second procedure is used to detect all of the classes in each component at once.

The only supervision our proposed methodology requires consists of samples that are used to estimate the weights of the edges in the graph for the graph-cut procedures. Experiments using a benchmark data set show that the performance of the proposed methodology that uses small amount of supervision is compatible with the ones in the literature.

Keywords: Object detection, data fusion, graph cut.

ÖZET

ÇİZGĖ KESİT İLE OPTİK VE LIDAR VERİ VERİ FÜZYONU KULLANARAK NESNE TESPİTİ

Onur Taşar
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Selim Aksoy
Mart 2017

Uzaktan algılama verisinde nesne tespiti, oldukça popüler bir problem olmuştur ve tarım, navigasyon, çevre düzenlemesi, kentsel görüntüleme ve haritacılık gibi çok geniş bir yelpazedeki uygulama alanlarında yaygın olarak kullanılmaktadır. Fakat bu problemi çözmek için sadece tek bir veri kaynağını kullanmak yeterli olmayabilir. Havadan çekilmiş optik ve LiDAR verilerinin füzyonu, birbirini tamamlayıcı bilgiler taşıdığı için uzaktan algılama verisinde nesne tespitinde umut verici bir yaklaşım olmuştur. Biz verinin çoklu düzey bölünmesini gerçekleştiren ve bölünmüş veriyi minimum düzeyde öğrenme kullanarak nesneleri tespit eden sistemler öneriyoruz. Yöntemimiz ön işleme aşamasında veriyi yükseklik değerine göre eşikleme ve daha etkili bir biçimde işlemek için onu küçük bileşenlere bölme adımları içeriyor. Sınıflandırma görevi için ise, LiDAR verisinden yükseklik, havadan çekilmiş veriden spektral ve komşuluk haritasından konumsal bilgileri kullanarak, her bir bileşende nesneleri tespit eden iki farklı çizge kesme tabanlı yöntem öneriyoruz. İlk yöntem ikili sınıflandırma sağlarken, ikincisi ise çoklu sınıflandırma gerçekleştiriyor. İlk yöntemi kullanarak yüksek piksellerde binaları ağaçlardan, alçak piksellerde ise yolları çimenlik alanlardan ayırt ediyoruz. İkinci yöntem ise her bir bileşendeki bütün nesneleri tek seferde tespit etmek için kullanılıyor.

Bizim önerdiğimiz yöntemin ihtiyaç duyduğu tek öğrenme çizge kesme yöntemindeki çizgenin kenarlarının ağırlıklarını belirlemek için kullanılan örneklerdir. Bir kıyas veri seti üzerinde gerçekleştirilen deneyler gösteriyor ki önerdiğimiz az miktarda öğrenme kullanan yöntem literatürdeki sonuçlarla uyumludur.

Anahtar sözcükler: Nesne tespiti, veri füzyonu, çizge kesit.

Acknowledgement

First of all, I would like to express my gratitude to my supervisor Assoc. Prof. Selim Aksoy for his motivation and encouragement throughout my master's education. Without his guidance and assistance, this thesis would not have been possible.

I also appreciate Asst. Prof. Ramazan Gökberk Cinbiş and Prof. Aydın Alatan for their feedback and constructive comments on my thesis.

I thank to my friends at Bilkent; Ali Burak Ünal, Barış Geçer, Bülent Sarıyıldız, Can Fahrettin Koyuncu, Caner Mercan, Gencer Sümbül, Iman Deznaby, Noushin Salek Faramarzi, Troya Çağıl Köylü and others for all the enjoyable moments.

I also would like to thank to my best friends Can Altuğer and Semo for their amazing companionship and valuable emotional support when I need most. I will never forget the enjoyable and unforgettable time we have had together.

Most importantly, I am so lucky to have a wonderful, caring and supportive family. I am grateful to my beloved father Süleyman Taşar, mother Fadime Taşar, sister Özlem Taşar and brother Özgür Taşar for their endless support and love. None of my achievements would have been possible without them.

Finally, as a Turkish prospective scientist, my sincere gratitude goes to Mustafa Kemal Atatürk, who showed us that science is the most reliable guide for civilization, for life, for success in the world. I dedicate this scientific contribution to my father Süleyman Taşar, my mother Fadime Taşar and the great leader Mustafa Kemal Atatürk!

This work was supported in part by the BAGEP Program of the Science Academy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Problem Definition

With the continuous advancements in satellite sensors, it is now possible to capture massive amount of remote sensing data having very high spatial resolution, as well as rich spectral information. Having such a huge, very high resolution data has opened the door to various remote sensing applications but at the same time it has made the processing operation harder. Object detection has been one of the most popular problems in remote sensing applications, and has been studied extensively for several decades because a good solution for this problem is crucial for a wide range of very important applications in domains such as agriculture, navigation, environmental management, urban monitoring and mapping. A good methodology for this problem has to detect the objects effectively and efficiently.

In traditional classification methods, each pixel is classified by using only spectral information, where the spatial information is ignored. However, in [3, 4, 5, 6, 7] it is proved that using object-based classification methods, where spatially similar pixels are grouped into an object, achieve more satisfactory results than using traditional pixel-based classification algorithms. The main reason why pixel-wise techniques perform worse is that since each pixel is processed

independently, the resulting label map becomes spatially incoherent in most of the cases. On the contrary, using spatial information alone is also not suitable for classification problems because semantic information of the pixels is lost if the spectral information is totally ignored. For these reasons, both spectral and spatial information definitely need to be used.

There are a lot of published papers that are related to spatial-spectral image classification. To give a few examples, in [8] extended multi-attribute profiles, which represent effectively the spatial and spectral information of hyper-spectral images are used as the features. A kernel is created using multiple kernel learning approach and given to an SVM classifier to generate a semantic labeling from the given image. In [9] in order to respect spatial coherency, the neighboring pixels of each pixel in a $w \times w$ window is found and a box of size $w \times w \times n$, where n is number of channels, is created for each pixel. 1-D vectors that have been obtained by flattening the boxes and the spectrum vectors of the hyper-spectral image are fed into a Deep Belief Network (DBN) structure to classify the image. In [10] instead of converting spatial features to 1-D vectors as in most of the papers, they propose a matrix representation for the features to preserve the spatial coherency. Pixels are classified with the SVM that takes the important features that are selected from the matrix representation with their proposed matrix-based discriminant analysis method. Spatial-spectral hybrid features are extracted by applying spectral-spatial cross-correlation constraint method to both spectral values and spatial adjacency, and the data are classified using a random forest classifier in [11].

Due to the increasing level of detail in the spectral bands of the optical images, using only color information as features for object detection may not be sufficiently discriminative because colors of pixels belonging to the same object may have very high variations. In addition, shadows falling onto objects make the detection task even more challenging. Since the works described above use only color information, they may fail to detect the objects that are in shadowed areas. In addition to the high resolution optical satellite and aerial imagery, more recently, developments in the Light Detection and Ranging (LiDAR) technology and LiDAR's capability of storing 3D shapes in the real world as point clouds

have enabled grid based digital surface models (DSM) that are generated from the point clouds as additional data sources. Consequently, exploiting both color and height information together by fusing aerial optical and DSM data has been a popular problem in remote sensing image analysis.

1.2 Contribution

In this thesis, we study the object detection problem for remote sensing data by using spatial, spectral and height information. We propose hierarchical frameworks to solve this problem. We use both aerial data, which provide spectral information, and LiDAR based DSMs, where height information of each pixel is stored. We first remove all objects from the DSMs by using morphological operations to estimate the bare ground. The resulting data are called the digital terrain model (DTM). Then, we subtract the DTMs from the DSMs to generate another data, which store relative height information of each pixel above the estimated ground, and called the normalized digital surface model (NDSM). Our hierarchical framework starts by partitioning the data set into two as high and low by thresholding the NDSMs. We then find the connected components in both high and low binary masks that represents this partitioning. This reduces the computational complexity because the connected components that are processed separately are much more smaller than the whole very high resolution input data. The main computational tool in this thesis is a graph-cut framework that fuses the spectral information from aerial data, height information from DSMs, and spatial information from neighborhood maps. Minimal supervision, where training data are used for estimating class-conditional probabilities to determine weight of the edges in the neighborhood maps, is used. Graph-cut procedure performs a pixel-wise classification but it forces the neighboring pixels to have the same label so that the resulting classification map becomes spatially coherent. We propose to use two frameworks, where the first framework is based on a max-flow/min cut algorithm [12, 13] and the second one is based on multi-label optimization [14, 15, 16]. In the first framework, we apply the max-flow/min-cut algorithm to high pixels to detect buildings and trees, and to low pixels to find roads and grass

areas. Then, we merge the classification maps coming from the low and the high pixels to get only one classification map. In the second framework, we apply the multi-label optimization algorithm to each connected component to detect all of the classes in the given data. The main contribution of this thesis is the use of these graph cut frameworks for fusion of spectral, height and spatial information for object detection with minimal supervision. Finally, extensive quantitative and qualitative experiments, where different types of areas are used in a benchmark data set, are done to evaluate performance of our frameworks. Although we use only small amount of supervision, our results are compatible with the ones in the literature.

Some part of this work was published as [17].

1.3 Outline

In Chapter 2, we introduce the related works that use spatial, spectral and height information in order to classify the remote sensing data.

In Chapter 3, we first describe the data set used in this thesis in great detail. We then provide an overview about the two proposed frameworks. Then, we explain how we generate the relative height information of each pixel from the raw elevation data and how we divide the data into smaller components to process it efficiently. Finally, we explain the features we extract from both aerial and LiDAR data, and the graph cut procedures in detail.

In Chapter 4, firstly, we describe the experimental setup, then, explain evaluation criteria and how we optimize the parameters. We provide both numeric and visual results, and give a detailed discussion at the end of this chapter.

Finally, in Chapter 5, we conclude the thesis.

Chapter 2

Related Works

Since the most common object class in remote sensing applications is building, there are lots of papers trying to solve the building detection problem using spatial, spectral and height information. Almost all of the approaches start with dividing the data into two groups as high and low by thresholding the LiDAR data. Then, the low pixels are ignored and buildings are detected in the high pixels. In [18], a threshold is applied to the high pixels to eliminate the trees covering low area. Then, buildings are separated from the remaining trees by taking advantage of edge histograms. However, all of the buildings are assumed to a have rectangular shape, which may not be always the case. A probabilistic framework that is based on conditional random fields (CRF) is proposed in [19]. In [20, 21], after dividing the data into two in the preprocessing step, buildings are separated from the remaining trees by using traditional machine learning methods. The method described in [21] can also detect trees. A stratified morphology based approach is described in [22]. Firstly, potential buildings are extracted using the proposed morphological building index (MBI) method and they are post-processed by morphological operations. However, obviously, the biggest disadvantage of these algorithms is that they have been designed to detect only one class. In real life problems, there are more than one class in remote sensing data.

Among the methods that are able to detect more than one class, rule based

approaches are very popular. Most of the published works using rule based approaches use height and NDVI as the features because NDVI is a distinctive feature for the healthy vegetation, and height is a good feature to separate the high objects such as buildings and trees from the low ones such as grass areas and roads. Objects are detected by setting several thresholds for these features. The input data are segmented by applying a region growing method to the LiDAR data in [23]. Small regions are assumed to belong to tree class because the pixels belonging to trees tend to show high height variance. The remaining classes are detected by setting thresholds for certain features such as NDVI. In [24, 25, 26], lots of thresholds are set for height and NDVI features to detect objects. In [26], additional features such as the Enhanced Shadow Index (ESI) are also used. In [24], different thresholds are also determined to detect the objects belonging to the same class but located in shadowed and open areas. Finally borders of buildings are straightened by Hough transform. An object based classification approach, which generates a segmentation first and classifies each region of the segmentation map later, is proposed in [27, 28]. The segmentation map is generated by watershed in the former paper, and it is generated by the quad-tree algorithm in combination of region growing in the latter paper. In [27], the input image is classified two different ways that are based on Gaussian Mixture Models and class specific rules. In [28], apart from the NDVI and height, a lot of additional features such as brightness and slope are also used. In [29], roads and buildings are detected with the rules defined in the paper, and the rest of the classes are found by an SVM classifier. A hierarchical rule tree is defined in [30] to detect the objects. Classification of the hyper-spectral and LiDAR data is performed separately in [31], and these two classification maps are combined with the defined rule chains. However, the rule chains of these methods might be data specific and may fail to detect objects when new data comes.

Deep neural network based approaches are also very popular in detecting objects by using spatial, spectral and height information. In [32], various extinction profiles are extracted from both the hyper-spectral and LiDAR data and used as the features. Then, these features are combined together with several feature

fusion methods such as vector stacking and graph based methods. The fused features are given to a deep convolutional structure with logistic regression to classify the data. In [33], two different convolutional neural networks are proposed to extract features from the hyper-spectral and LiDAR data. Spatial-spectral features are extracted with a 3D CNN from the hyper-spectral data, and elevation related features are extracted with a 2D CNN from the LiDAR data. The extracted features are fused with another CNN. Finally, objects are detected with a logistic regression classifier that takes the fused features as inputs. In [34], the fusion operation is performed at the data level. The LiDAR data are appended to the hyper-spectral data. The fused data are fed to the CNN structure described in [35] for classification. In [36], a new feature representation method that is based on a CNN trained from both the LiDAR and hyper-spectral data is proposed. In [37], classification is performed with various methods such as SVM, CNN and performance comparisons are done for these classifiers. In same the paper, it is claimed that the CNN based methods outperform the other methods with the used data set. From the remote sensing labeling contests and lots of published works we learned that with the deep convolutional based methods, better results are obtained in most of the cases. However, deep CNNs require high computational power and it is very slow to train them if a very good GPU is not available. In addition, they need a huge training data.

There are also lots of published papers, which explain methods that fuse spatial, spectral and height information at the feature level. In most of these works, a lot of features are extracted first, then certain dimension reduction techniques are applied to get rid of the redundant information. Finally, classification is performed by giving the selected features as inputs to the well-known classifiers such as SVM. In [38, 39, 40], morphological features are extracted from both hyper-spectral and LiDAR data. Then, a graph-based feature fusion method is applied to the features in order to project them onto a subspace. The data are classified using the features in the subspace. Authors of [41] also use the morphological features as inputs. The classification is performed with the subspace multinomial logistic regression classifier. In [42], features are extracted from both hyper-spectral and LiDAR data using deep convolutional networks.

The extracted features are ranked according to how well they represent the input data using the approach that is based on random forests described in [43]. After extracting the features, the most important ones are selected with a random forest in [44]. Then, the data are classified with an object based method, where the data are segmented and each segment is classified after that. Finally, the data are classified using the features that represent the data the best. Firstly, both hyper-spectral and LiDAR data are fused in [45]. Then, the features are extracted and dimension of the features are reduced with Principal Component Analysis (PCA). Finally, the data are classified with an SVM and a Maximum Likelihood Classifier (MLC). In [46], the features are extracted and the most important ones are selected using Particle Swarm Optimization (PSO). Then, two fuzzy classification algorithms, which are fuzzy *k-nearest* neighbor (FKNN) and fuzzy maximum likelihood (FML) are applied. A final decision template is designed to combine these two fuzzy classification maps. In [47, 48], morphological features are extracted and dimension reduction approaches are applied. In the former paper random forest is used whereas in the latter SVM is used for the classification. However, all of the methods described above use a dimension reduction method and each of them has its own advantages and disadvantages.

Classifying the data by extracting the features from both the hyper-spectral and LiDAR data, and feeding them to a classification system consisting of the well-known machine learning classifiers such as SVM, random forests, logistics regression is also very common. Three different SVM based classification methods are described in [49]. In the first method, after extracting the features from the hyper-spectral and LiDAR data, the features are given to an SVM classifier directly. In the second and third approaches, the hyper-spectral data are classified with an SVM. In the second approach, the obtained classification map as well as features of the LiDAR are given to another SVM. In the final approach, errors are reduced in the classification map using the height information at the post-processing step. In [50], the hyper-spectral and LiDAR data are classified separately with SVM classifiers. The resulting classification maps are combined with a Naive Bayes classifier. A pixel-wise classification is performed with Maximum Likelihood Classification (MLC) and SVM in [51]. A classical object based

classification is also done. Finally, classification maps that are obtained at the end of the pixel-based and object-based classifications are combined. In [52], a two step classification procedure is proposed. Firstly, data are classified using SVM. Then, the classification map is transformed to another map, where Euclidean distances between each pixel and its neighboring pixels are stored. The most common label in the neighborhood of each pixel is also found and using this information and the distance map, the classification map obtained by the SVM is reclassified. In [53], textural features are extracted from the hyper-spectral data and height features are extracted from the LiDAR, and the data are classified with the Multinomial Logistic Regression (MLR) classifier that takes the extracted features as inputs. The authors of [54], apply Dempster-Shafer algorithm, which generates two outputs; one is the initial classification map and the other is a set of training pixels. An SVM is trained from the training pixels and class conditional probabilities are calculated for each pixel using the trained SVM. The final classification map is generated with the Markov Random Field (MRF) model that uses the probabilities. Geometric and spectral features are extracted from LiDAR and hyper-spectral data respectively, and the data are classified with a Bayesian Network (BS) model in [55]. In [56], using the spectral and height information, a histogram for each class is generated. It is claimed that histograms belonging to different classes have different characteristic. Data are classified using the histograms with a Kullback-Leible divergence based method. In [57], Extended Attribute Profiles (EAPs) are extracted from both data and fused together. Then, objects are detected by feeding the fused features to random forest and SVM classifiers. An object based methodology is followed in [58]. Data are segmented first. Then, dual morphological top hat profile features are extracted and each region is classified using a random forest classifier.

Binary [12, 13] and multi-way [14, 15, 16] graph cut algorithms are commonly used tools in image segmentation and classification problems because of their effectiveness and efficiency. There are a few published papers, which solve object detection problems by fusing LiDAR and hyper-spectral data with graph-cut frameworks. A hierarchical building detection method, which consists of three steps is proposed in [59]. In the fist step, the shadows are detected by using

a morphological index. In the second step, the top-hat reconstruction method is applied to the DSM, and combining its output with the NDVI based features coming from the hyper-spectral data, an initial building mask is extracted. In the final step, the initial building mask is consolidated with the graph-cut procedure. The obvious downside of this step is that it is designed to detect only buildings. In [60], the LiDAR data are represented by Fisher vectors. The classification is performed by using the hyper-spectral data, as well as the Fisher vectors with the multi-way graph-cut procedure. To the best of our knowledge, the closest paper to our work is the one described in [61], which proposes an object based approach that is based on the multi-way graph cut procedure. Firstly, the LiDAR data are segmented according to the normal vector of each point using a region growing method. Then, the wrongly segmented regions are corrected with the spectral information. After extracting various features from both the hyper-spectral and LiDAR data, for each region in the segmentation map, average of each feature of the pixels belonging to a region is taken and the new features are assigned to that region. By doing this, each region is represented by only one feature vector. Finally, each region is classified using a multi-way graph-cut procedure. The first drawback of this method is that a lot of rule based features that have lots of thresholds are extracted. The determined thresholds for these features may be specific to the used data set, it may not be generalized to other real world problems. In addition, one of the used data sets is the Vaihingen data set that is provided by the International Society for Photogrammetry and Remote Sensing (ISPRS). We use the same data set in our experiments. However, results for only 3 areas are provided although in the original data set there 16 areas. In our experiments, we test our method on 10 different areas.

The Vaihingen data set, which we use in this thesis and provided by the International Society for Photogrammetry and Remote Sensing (ISPRS), is a contest data set where some part of the data set is annotated. The participants are expected to train their models from the annotated part of the data set and submit classification maps for other parts of the data set, which do not have a ground-truth. There are some published papers and technical reports whose results are

ranked by the ISPRS. In [62], an Adaboost-based classifier is trained from a subset of images, and is combined with a Conditional Random Field (CRF) model to classify the data. In [63], three different classification systems are proposed where the first one uses a CNN model, the second one combines the CNN with a Random Forest classifier, and the last one post-process the hybrid model with a CRF model. A two step classification system, where the features are extracted using a CNN model and then fed to an SVM classifier is proposed in [64]. A fully convolutional model with no down-sampling is proposed in [65]. A deep fully convolutional neural network models, which fuses both aerial and LiDAR data are described in [66] and [67]. In [68], a deep convolutional neural network model with boundary detection is proposed. A CNN framework, which extracts features from multiple resolutions and combines them is proposed in [69]. A random forest based and a fully convolutional neural network (FCN) based classification system are described, and their performances are compared in [70]. In [71], a structured CNN, which includes a radial basis function (RBF) kernel approximation and a linear classifier layers is proposed. A dual-stream deep neural network model, which learns from both local and global visual information is proposed in [72]. Two CNN-based methods, which are explained in [73] and [74] are tested on the PASCAL VOC, NYUDv2, SHIFT-flow data sets in the original papers but their scores are also available on the results list for the Vaihingen data set.

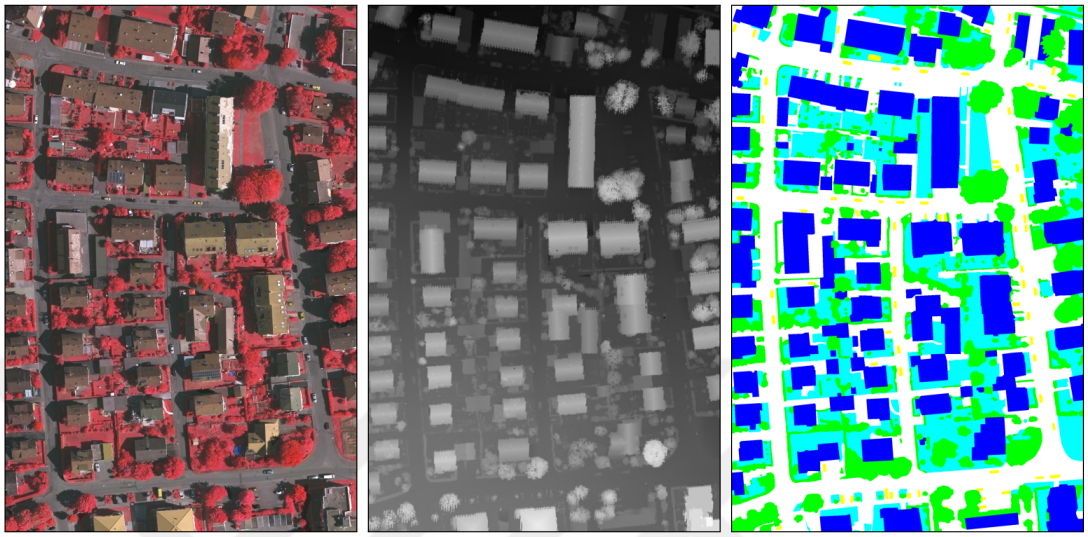
Chapter 3

Methodology

In this chapter firstly the data set is explained in great detail. Then, we give an overview of the two proposed frameworks. The flowchart for the proposed methodology can be seen in Figures 3.2 and 3.3. After that, we give more detail about each step of the methodology. In the Preprocessing Section we explain how we generate the relative height information from the given raw elevation data using four different methods, and how we divide the data set into smaller components to reduce the computational complexity. In the Features Section, we explain the extracted features from both the spectral and the height data. Finally, we describe the proposed frameworks that are based on max-flow/min-cut and multi-label optimization procedures, respectively.

3.1 Data set

We use the 2D Semantic Labeling Contest data set that was acquired over Vaihingen, Germany and was provided by the International Society for Photogrammetry and Remote Sensing (ISPRS). The data set contains pan-sharpened color infrared ortho photo images, having infrared, red and green channels, with a ground sampling distance of 8 cm and a radiometric resolution of 11 bits. It also contains



(a) (b) (c)

Figure 3.1: An example area from the data set. (a) Ortho photo. (b) DSM. (c) Reference data with class labels: impervious surfaces (white), building (blue), low vegetation (cyan), high vegetation (green), car (yellow).

point cloud data that were acquired by an airborne laser scanner [75]. The digital surface model (DSM) data, where height information of each pixel in the infrared images is stored, are provided as well. In the data set, there are two different types of DSMs that were generated using two different methods. The ones in the first group were generated from the original ortho photo images by dense matching using the Match-T software [76]. The DSMs in the second ground were generated by interpolating the point cloud with a grid width of 25 cm. Although its resolution is lower, we use the DSMs that were generated from the point clouds because they are more accurate. Since the Vaihingen area is huge, it is hard to process the raw infrared and DSM data. For this reason, the whole area has been divided into 33 tiles by the data set providers. Among these 33 tiles, only 16 of them have ground-truth. In addition, DSMs for 6 out of 16 regions are not usable. As a result, we use only area 1, 3, 5, 7, 11, 15, 29, 30, 34 and 37. In the data set there are 6 different classes; impervious surfaces (road), building, low vegetation (grass), high vegetation (tree), car and background. Only a few tiles have background objects. Infrared ortho photo image, DSM and ground-truth for area 3 can be seen in Figure 3.1.

3.2 Overview

We propose a method that is able to detect the classes in the ISPRS Vaihingen data set in a mostly unsupervised way by using spatial, spectral and height information of each pixel. Although there are 5 major classes and a background class in the ground-truth, we detect 4 of the classes; impervious surfaces, building, low vegetation, and high vegetation. The reason why our system is not able to detect cars is that the DSMs and infrared ortho photo images have been taken on different dates. For this reason, position of some of the cars in the 3 band images and the DSMs do not match.

We begin with grouping the pixels in the data set as high and low by thresholding the DSMs according to height. We then extract binary masks for the high and low pixels, and find the connected components in these masks. The high pixels constitute buildings and trees, whereas the low ones consist of roads and grass areas. However, at the end of this thresholding step, a small portion of the pixels can be classified incorrectly; high pixels may contain grass and road, and low pixels may have building and tree. We use two different graph cut procedures to detect classes in all of the connected components. The first procedure separates buildings from trees in each component in the high binary mask, and roads from grass in each component in the binary mask for the low pixels using the max-flow/min-cut algorithm described in [12, 13]. The flowchart of this framework can be found in Figure 3.2. In the second graph cut procedure, instead of running max-flow/min-cut procedure in each connected component of the high and low binary masks separately, we apply the multi-label optimization [14, 15, 16] to each connected component in order to detect all of the classes directly. The flowchart of the proposed method that is based on the multi-label optimization can be found in Figure 3.3. Max-flow/min-cut algorithm provides an optimal solution [13] but multi-label optimization does not always guarantee an optimal solution [15].

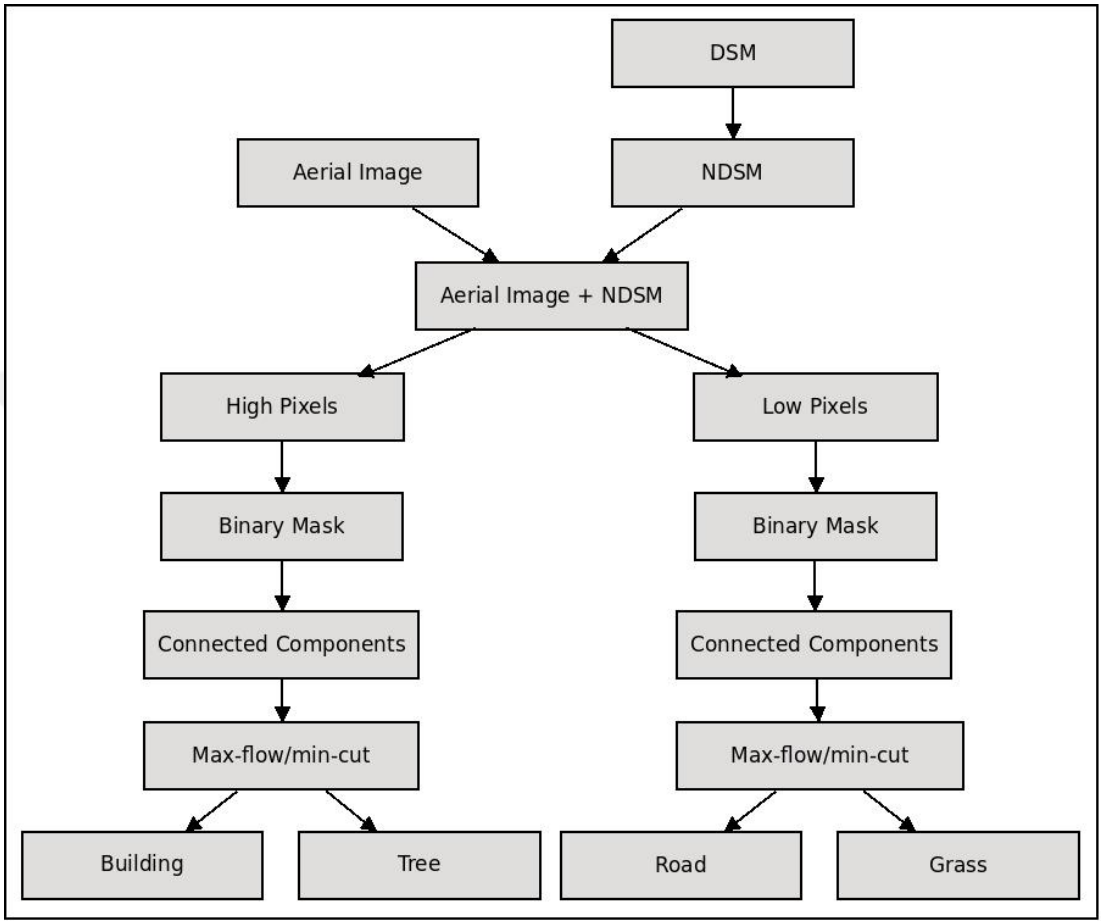


Figure 3.2: Flowchart of the proposed method that is based max-flow/min-cut.

3.3 Preprocessing

In the preprocessing step, we generate binary masks for the high and low pixels by thresholding each DSM data and find the connected components in both of the masks. We use four different methods in order to divide the pixels in the DSMs as high and low. These are gray-scale reconstruction based on geodesic dilation, progressive morphological filtering, area opening and opening. Among these four methods, the progressive morphological filtering finds the high and low pixels directly. Others take a DSM as an input and generate another data, containing height information of the bare earth without objects on it. The resulting data are called the digital surface model (DTM). After generating the DTM, we subtract it from the DSM in order to generate the normalized digital surface model (NDSM),

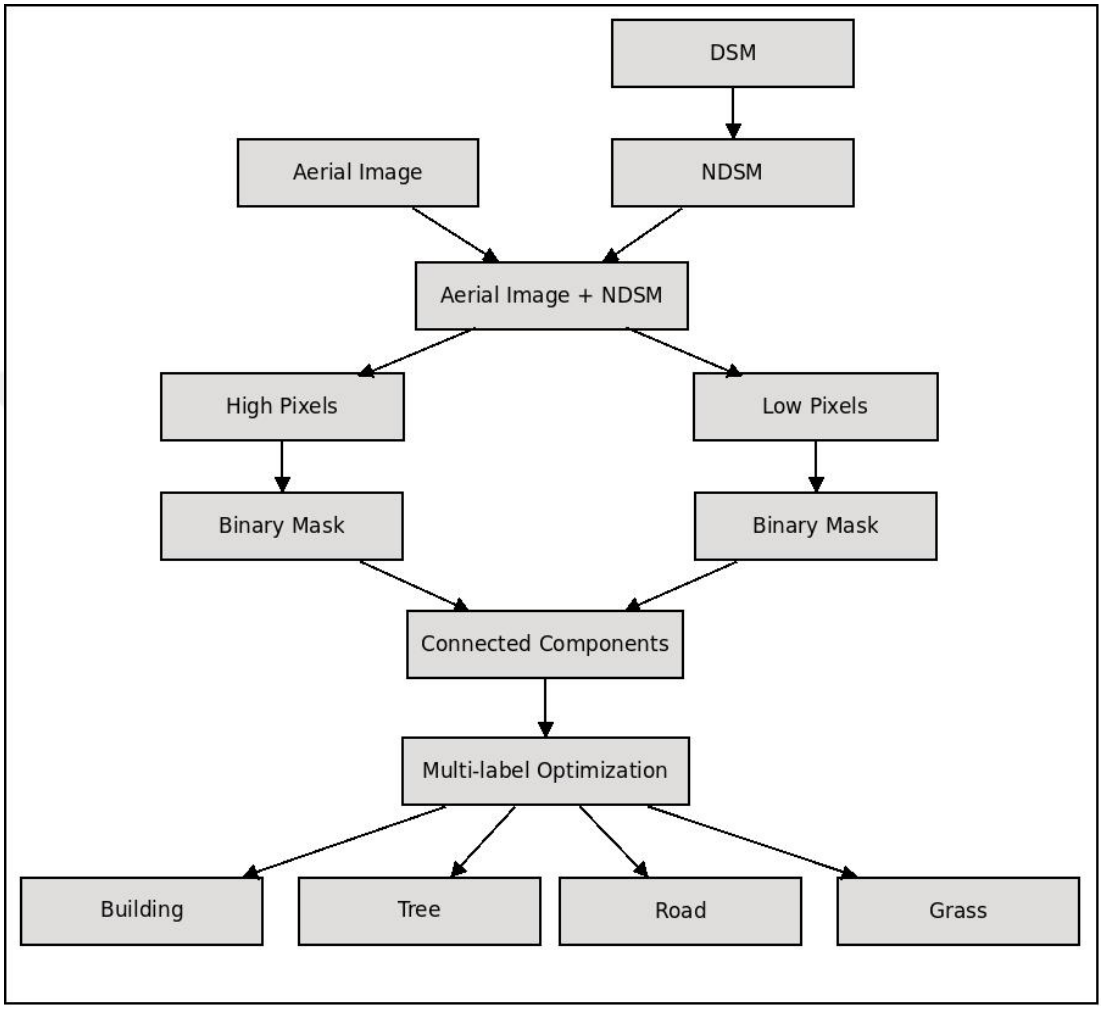


Figure 3.3: Flowchart of the proposed method that is based on multi-label optimization.

which is another data that contains relative height information of each pixel above the estimated ground. We finally group the pixels as high and low by thresholding the NDSMs.

3.3.1 Grayscale Reconstruction Based on Geodesic Dilation

Gray-scale reconstruction takes two input gray-scale images with the same size. These images are called mask and marker. The marker image must be below the

mask image. In this algorithm, the marker image is dilated as long as it remains below the mask image. In other words, the mask image limits dilation of the marker image.

The marker and mask images are denoted by J and I respectively, and the structuring element is denoted by B. The gray-scale dilation of J and the structuring element B is

$$\delta(J) = J \oplus B \tag{3.1}$$

where $\oplus$ denotes the gray-scale dilation.

The geodesic dilation of J limited by I and dilated by the structuring element B sized 1 is

$$\delta_I^1(J) = (J \oplus B) \wedge I. \tag{3.2}$$

Gray-scale reconstruction is obtained by applying the geodesic dilation successively n times until the marker image fits under the mask image as

$$\delta_I^n(J) = \underbrace{\delta_I^1(J) \circ \delta_I^1(J) \circ \ldots \circ \delta_I^1(J)}_{\text{n times}}. \tag{3.3}$$

The DSM data corresponds to I in the equations above. When we apply the gray-scale reconstruction method to a DSM data, the resulting reconstructed image corresponds to DTM. More information about the gray-scale reconstruction and geodesic dilation can be found in [77], and applications of these algorithms to the DSM data can be found in [1].

The gray-scale reconstruction procedure is illustrated in Figure 3.4.

3.3.2 Progressive Morphological Filtering

The progressive morphological filtering algorithm [2] applies the morphological opening operation to the given DSM progressively. It first opens the DSM with an initial structuring element. After that, the opened image is substracted from the given DSM. Among the pixels in the resulting data obtained at the end of the substraction operation, the ones which are higher than the predefined threshold

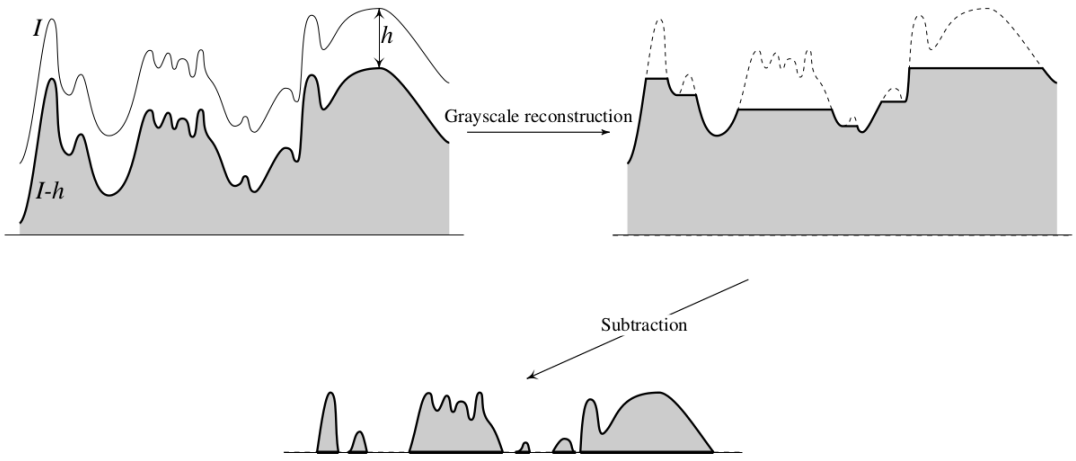


Figure 3.4: Flowchart of the Gray-scale Reconstruction. Top-left image: I is the mask image (DSM), I-h corresponds to J, which is the marker image, Top-right image: reconstructed image (DTM), bottom image: difference between the DSM and DTM (NDSM). This figure was taken from [1].

value are labeled as non-ground whereas the others are labeled as ground. In the subsequent iterations, size of the structuring element is increased according to

$$w_k = 2kb + 1 \tag{3.4}$$

where w_k is the size of the current structuring element, k is the iteration number and b is the size of the structuring element. In addition to this, height threshold is also increased in each iteration as

$$dh_{T,k} = \begin{cases} dh_0, & \text{if } w_k \leq 3 \\ s(w_k - w_{k-1}) + dh_0, & \text{if } w_k > 3 \\ dh_{max}, & \text{if } dh_{T,k} > dh_{max} \end{cases} \tag{3.5}$$

where $dh_{T,k}$ is the current threshold, dh_0 is the initial height threshold, dh_{max} is the maximum height threshold and s is the slope of the area.

At the end of this algorithm, pixels of the given DSM image are labeled as ground and non-ground.

An illustration for the progressive morphological filtering algorithm can be found in Figure 3.5. As it can be seen from the Figure 3.5 that the height

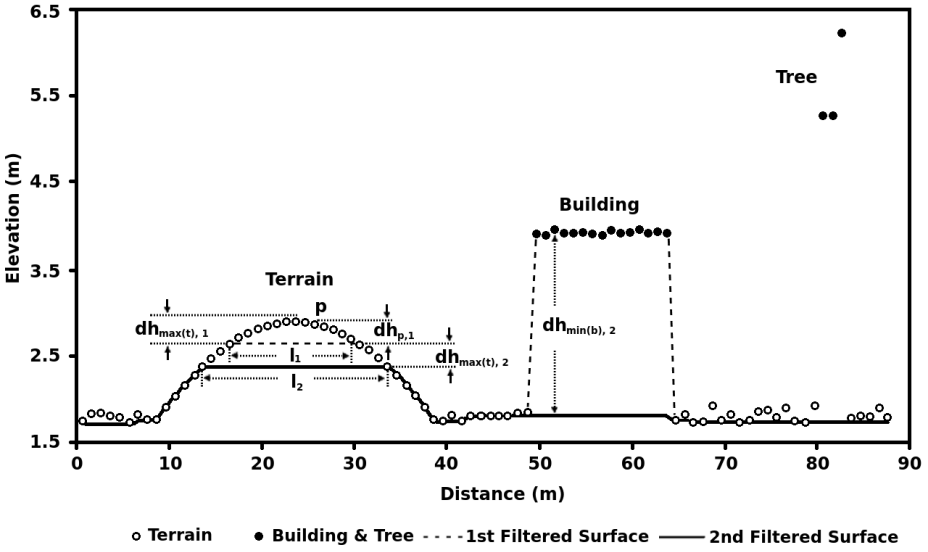
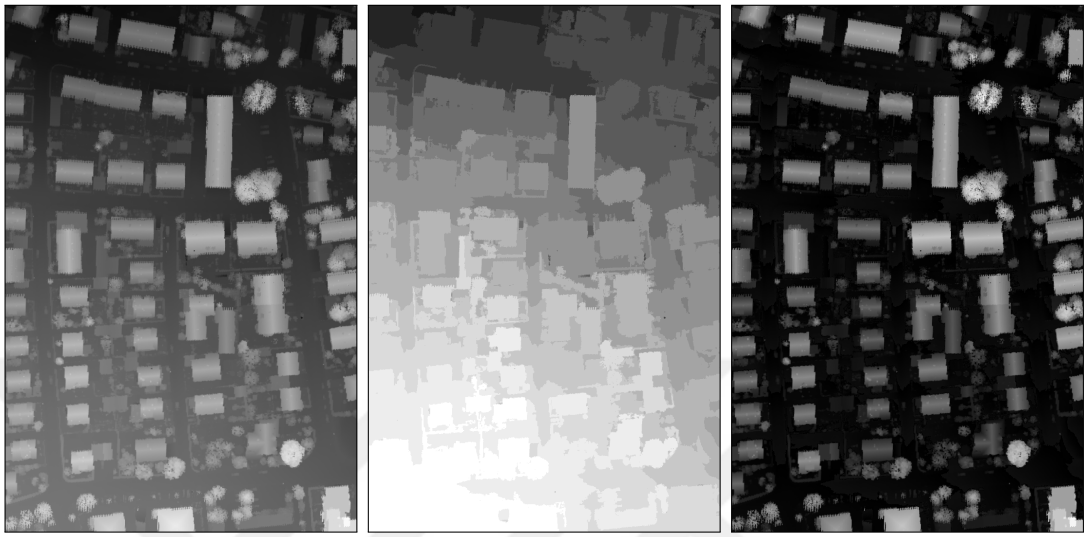


Figure 3.5: Illustration for the progressive morphological filtering algorithm. I_1 and I_2 are sizes of the structuring elements, $dh_{max(t),1}$ and $dh_{max(t),2}$ are the height thresholds in the first and second iterations, and $dh_{p,1}$ is difference between the height of the pixel P in the original DSM image and the height of the same pixel in the opened image at the end of the first iteration. This figure was taken from [2].

difference of the pixel P in the DSM and opened image is smaller than the height threshold. For this reason, this pixel is labeled as ground. On the other hand, the height difference of a pixel, which belongs to the building in the DSM and opened image, is bigger than the height threshold. Therefore, it is labeled as non-ground.

3.3.3 Area Opening

In the binary area opening method, connected components are found in a binary image and the ones having an area smaller than the given threshold values are removed. The binary area opening method later has been extended to gray-scale area opening [78], where the given image I is decomposed into several gray-scale images by defining different gray levels and area opening of the image I is the



(a) (b) (c)

Figure 3.6: Illustration of NDSM generation using area opening. (a) DSM, (b) DTM obtained by using area opening, (c) NDSM generated by subtracting DTM from DSM. Each image in the figure has been normalized between 0-255 for better visualization.

supremum of the gray-scale images that are smaller than I and having larger area than the area threshold.

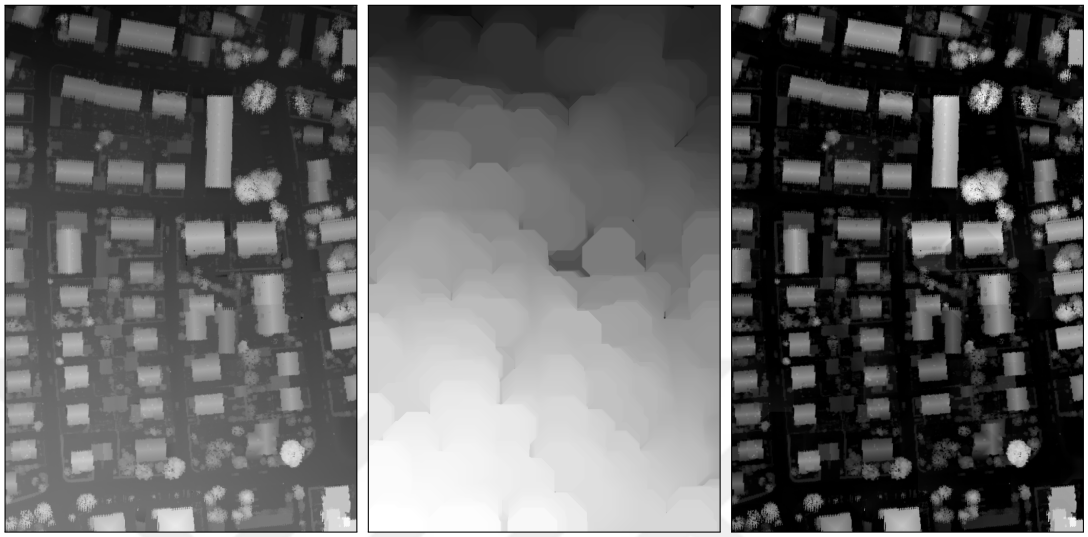
3.3.4 Opening

In order to generate DTMs from DSMs, we apply gray-scale morphological opening method successively. The opening operation starts with opening the input image by the structuring element whose shape is a disk with a radius of 3. In each iteration radius of the structuring element is increased according to

$$r_{current} = 2r_{previous}k + 1 \tag{3.6}$$

where k is the iteration number.

In the first iterations, since a small structuring element is used, the objects with a small width such as traffic lamps and small bushes are removed. As the size of the structuring element increases, the algorithm starts removing larger



(a) (b) (c)

Figure 3.7: Illustration of DSM generation using opening. (a) DSM, (b) DTM obtained by using opening, (c) NDSM generated by subtracting DTM from DSM. For the sake of visualization purposes, each image in this figure is normalized between 0 and 255.

objects. In the last iterations, the method removes the big buildings.

We separate the data set as high and low pixels using these four methods with different settings. To understand which method separates the best, we consider trees and building as one (high), roads and grass areas as another class (low), and calculate an accuracy for each method. Among these four methods classical gray-scale opening works the best. For this reason, in our experiments we use this method.

3.3.5 Generation of the Connected Components

After dividing the data set as high and low pixels, we extract binary masks for the separated pixels and find connected components in these masks. The main reason why we do not use the whole images is the issue of computational complexity. Since building very large graphs for the graph-cut procedures require extensive computational power, it is not feasible to use the images directly. Hence, we run

the graph-cut method in each component that is much more smaller than the whole image.

Example binary masks and their corresponding connected components can be found in Figure 3.8.

3.4 Features

We use several features that are based on spectral and height information. The features are explained in great detail in Sections 3.4.1 and 3.4.2.

3.4.1 Spectral Features

The first feature that is related to spectral information is color, which is self-explanatory.

The second feature in this category is normalized difference vegetation index (NDVI), which is one of the most common features in remote sensing applications because it is a very distinctive feature for the healthy vegetation. One condition to extract this feature is that the image must have infrared and red color bands. Intrinsically, the healthy vegetation reflects a large portion of the infrared light but absorbs most of the red light. However, for any other objects this situation does not apply.

NDVI is calculated as

$$NDVI = \frac{NIR - RED}{NIR + RED} \tag{3.7}$$

where NIR is the Near Infrared band. The values that NDVI can take range from -1 to 1.

An example infrared image and the corresponding NDVI map can be seen in Figure 3.9.

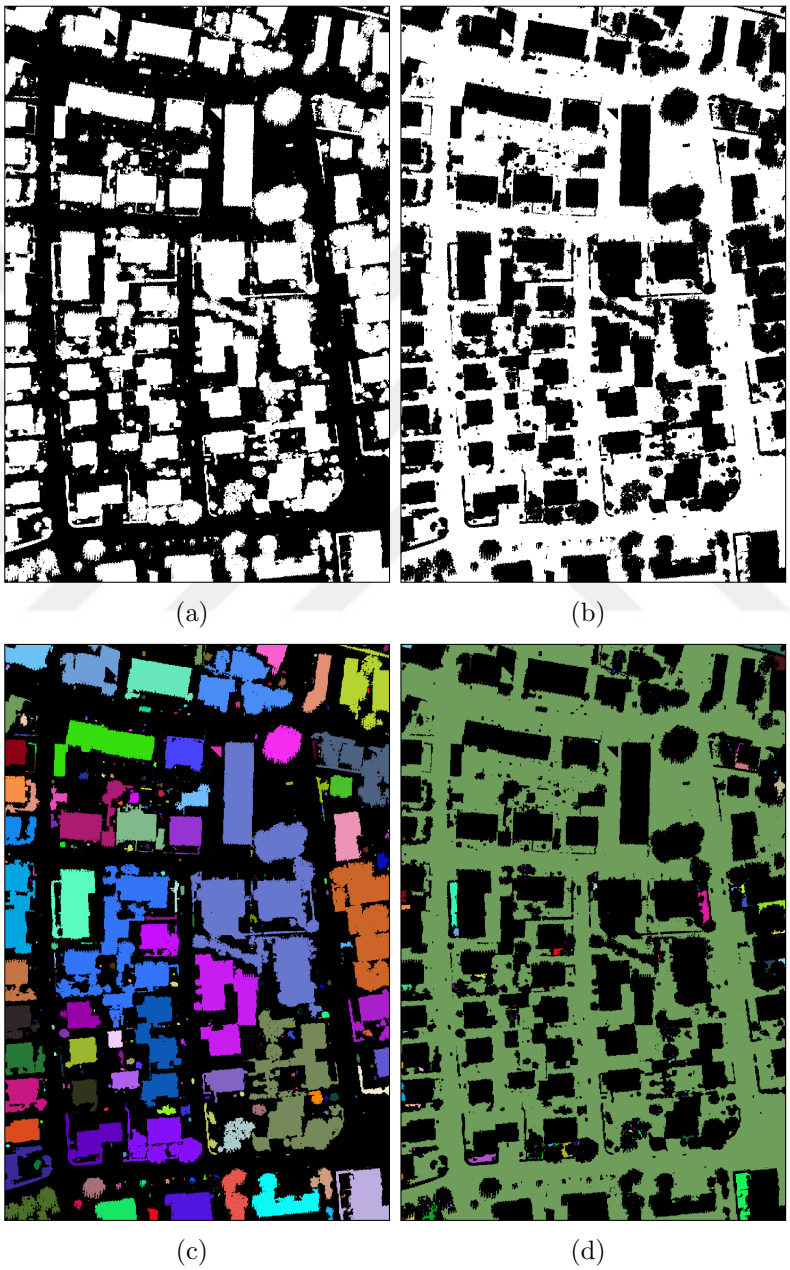


Figure 3.8: Example binary masks, and their connected components. (a) Mask for the high pixels, (b) Mask for the low pixels, (c) Connected components of the high mask, (d) Connected components of the low mask.

As can be seen from Figure 3.9, the pixels belonging to trees and grass have very high NDVI value than the others.

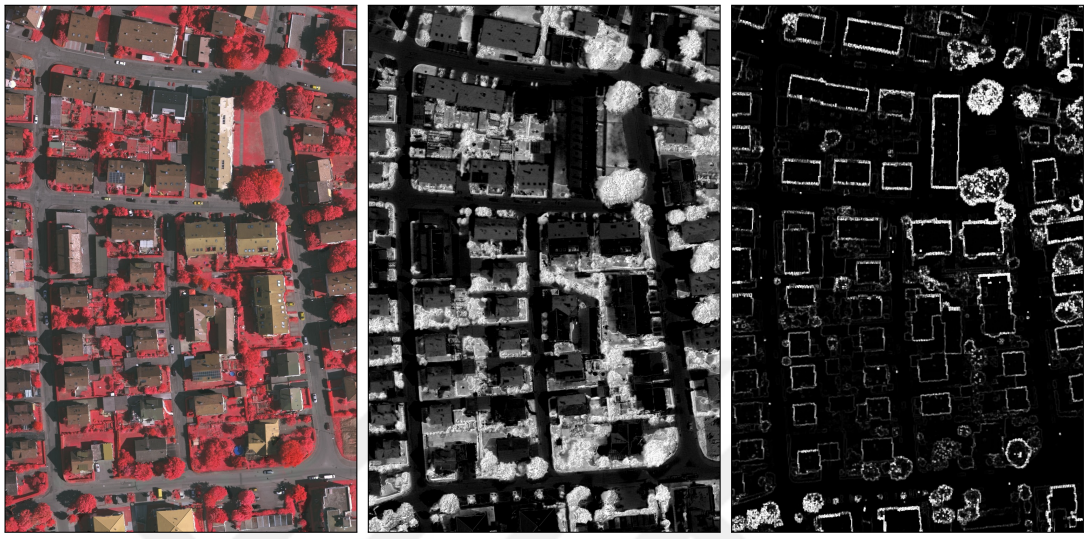
3.4.2 Height Features

The features falling into this group is just the relative height information, which is stored in the DSMs, and irregularity that needs detailed explanation.

The pixels belonging to buildings, roads and grass mostly belong to planar shapes, whereas the ones belonging to trees have irregular shapes. For this reason, irregularity is a potentially distinctive feature to separate trees from the other classes.

Although the DSM is two dimensional data, since height information of each pixel is available in the DSMs, they can be considered as if they are three dimensional data. We slide a 7×7 pixel window on the DSM data. Using the x, y coordinates and the height information of the pixels in the window, we calculate a 3×3 covariance matrix. From this covariance matrix, we find the smallest eigenvalue. The resulting value is assigned as the irregularity value of the pixel that lies in the middle of the window. By repeating this operation, we find the irregularity value of each pixel in the DSM image. Since the the pixels belonging to roads, buildings and grass are parts of planar shapes, their the smallest eigenvalues are very close to zero. On the other hand, the smallest eigenvalue of the pixels belonging to trees are larger.

An example area and corresponding irregularity map can be seen in Figure 3.9. As it can be observed from the figure that irregularity value of the pixels belonging to trees are high, whereas most of the pixels that belong to the other classes have lower irregularity. However, there are some pixels causing problems as well. The pixels, which lie at the borders of buildings have a relatively high value for the irregularity because when calculating the smallest eigenvalue for these pixels, some of the pixels falling into the sliding window are on the building, whereas the others are on the ground. A similar problem also applies to some of



(a) (b) (c)

Figure 3.9: An example infrared ortho photo image and its corresponding feature intensity maps. (a) Infrared image, (b) NDVI map, (c) Irregularity map.

the trees. Some small vegetation patches such as bushes have planar shapes. As a result, the pixels belonging to these sort of vegetation have small irregularity values.

These problems are solved by the graph cut procedures explained below.

3.5 Graph Cuts

3.5.1 Max-flow/Min-cut Procedure

We detect buildings and trees by applying the max-flow/min-cut framework described in [12, 13] to the connected components of the binary mask for the high pixels. We repeat the same operation in order to separate roads from grass in each connected component of the binary mask for the low pixels.

In this graph cut framework, each node in the graph corresponds to a pixel in the connected component. There are also two specially designed terminal nodes

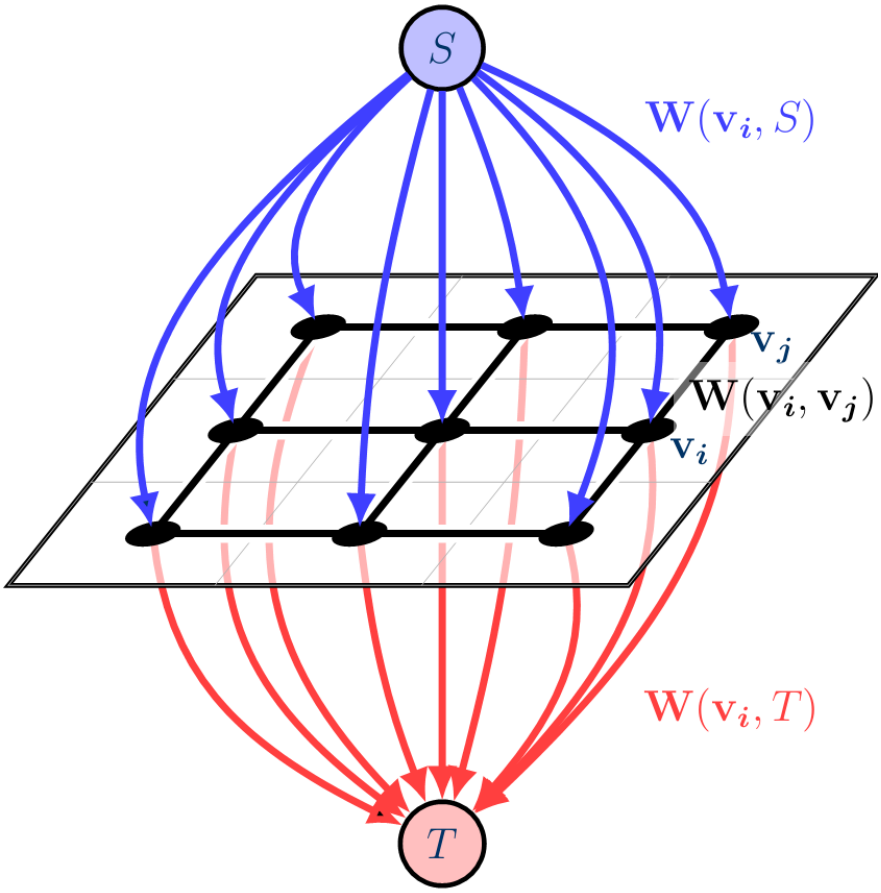


Figure 3.10: An example graph model for the max-flow/min-cut procedure.

named source S and sink T. In our problem, the source corresponds to one class of interest (e.g., building or grass) and the sink corresponds to another (e.g., tree or road). All of the nodes in the graph are connected to both the source and the sink nodes, and all of the neighboring nodes are connected to each other with weighted edges. The weight between two neighboring nodes is the cost of cutting the edge between these nodes. The weight between a node corresponding to a pixel in the image and a terminal node, which represents a class of interest is the cost of assigning the node to the terminal node. The goal of the cut procedure is to assign each of the nodes to one and only one of the terminal nodes by cutting some edges. An example graph model can be seen in Figure 3.10.

Let $\mathcal{C}$ be the class set and f be a particular feature value of a pixel. The probability of assigning the pixel to class c using the feature value f is calculated

as

$$p(c \mid f) = \frac{p(f \mid c) \cdot p(c)}{\sum_{c \in \mathcal{C}} p(f \mid c) \cdot p(c)} \tag{3.8}$$

where $p(c \mid f)$ is the posterior probability and $p(f \mid c)$ is the class-conditional probability distribution function (pdf) for feature f. $p(c)$ is the prior probability for class c. The classes are assumed to be equally probable in the rest of the thesis.

In order to calculate the posterior probabilities, probability distributions that model the likelihood of each pixel in each class for each feature need to be estimated. We estimate the probability distributions of all of the classes for the NDVI feature by Gaussian mixture models with four components and for the irregularity feature by exponential functions. For the height feature, we estimate the distributions of road, grass and tree by exponential functions, and distribution of building by a Gaussian mixture model. The models are selected based on visual inspection.

Histograms for the irregularity, height and NDVI values, and the fitted models can be seen in Figures 3.11, 3.12 and 3.13. The pixel posterior probability map and the posterior probability distribution of each class for each feature can be seen in Figures 3.15, 3.16 and 3.17 and 3.18, respectively.

Let V correspond to the set containing all pixels and N be the set consisting of all pixel pairs in a connected component. We define the vector $A = (A_1, \ldots, A_{|V|})$ as the binary vector whose elements A_v are the labels for the pixels.

In order for the max-flow/min-cut framework to detect the classes in each connected component, we define a cost function. The max-flow/min-cut procedure cuts some of the edges for minimizing the cost function. At the end of the cut operation, each node in the graph is assigned to either the source or the sink terminal node. The cost function to separate buildings from trees in the connected components for the high masks is defined as

$$E(A) = \underbrace{\lambda_1 \cdot R^{\text{NDVI}}(A) + \lambda_2 \cdot R^{\text{Irregularity}}(A)}_{\text{Data Cost}} + \underbrace{\lambda_3 \cdot B^{\text{Spatial}}(A)}_{\text{Smoothness Cost}} \tag{3.9}$$

where

$$R^i(A) = \sum_{v \in V} R^i_v(A_v), \quad i \in \{\text{NDVI}, \text{Irregularity}\} \tag{3.10}$$

accumulates the individual penalties for assigning pixel v to a particular class A_v based on the NDVI and the irregularity features. For the smoothness cost in (3.9), we use two different definitions. We run our experiments for these two smoothness cost definitions, and compare and discuss the results that are obtained using both definitions. The first definition for the $B^{\text{Spatial}}(A)$ term is

$$B^{\text{spatial}}(A) = \sum_{\{u,v\} \in N} \delta_{A_u \neq A_v} \tag{3.11}$$

which penalizes the cases where neighboring pixel pairs have different class labels ($\delta_{A_u \neq A_v} = 1$ if $A_u \neq A_v$, and 0 otherwise).

The second definition is

$$B^{\text{Spatial}}(A) = \sum_{\{u,v\} \in N} B_{u,v} \cdot \delta_{A_u \neq A_v} \tag{3.12}$$

where

$$B_{u,v} = p\left(D\left(u, v\right)\right) \tag{3.13}$$

and

$$D(u, v) = \left\| C(u) - C(v) \right\|. \tag{3.14}$$

The $C\left(\cdot\right)$ in (3.14) stands for color information of the given pixel.

Both (3.11) and (3.12) enforce spatial regularization, where neighboring pixels have similar labels as much as possible. The only difference between these two definitions is that rather than assigning a constant to the cost of cutting the edge between two neighboring pixels as described in (3.11), we give a higher weight to the edge between two neighboring pixels if their color is similar to each other, and give a lower weight if their color difference is high in (3.12). To accomplish this task, we estimate the probability distribution that models the likelihood of color distance between two neighboring pixels. Histogram of the distances between all pairs of neighboring pixels and the fitted exponential model can be seen in Figure 3.14.

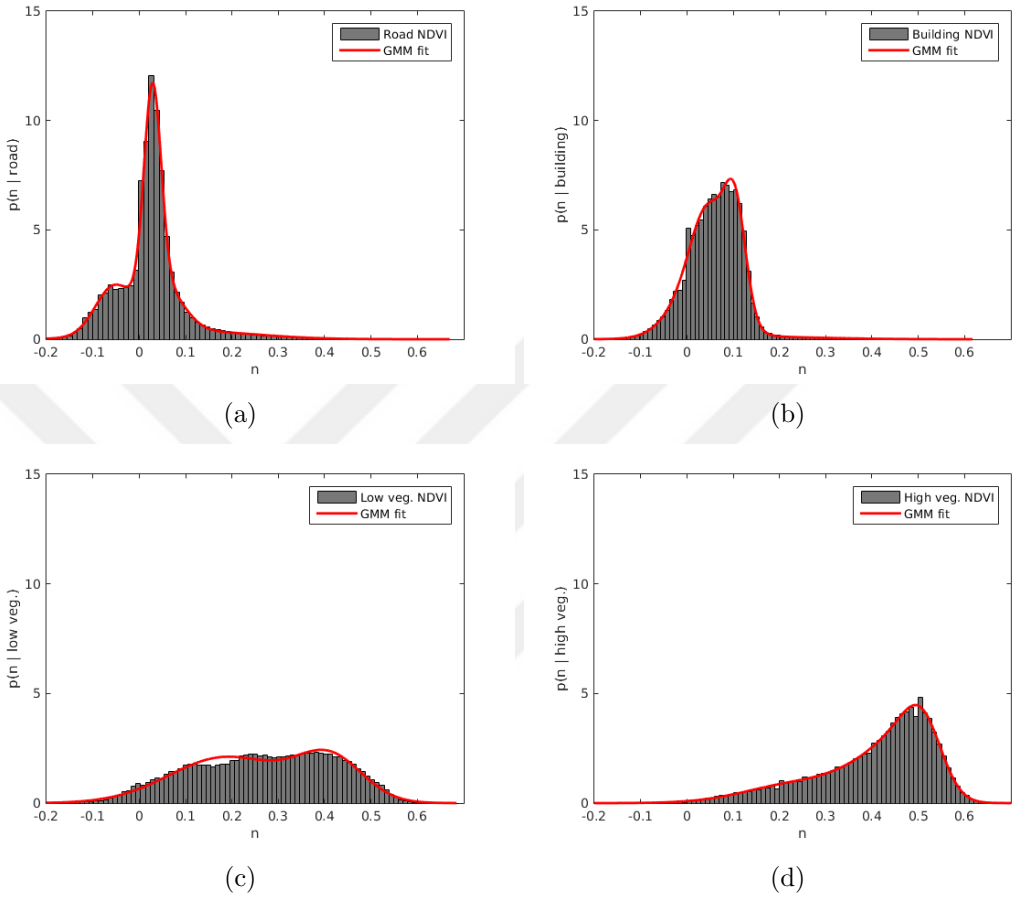


Figure 3.11: Histograms and the estimated probability distributions of each class for the NDVI feature. NDVI values are modeled using Gaussian mixture models with 4 components. (a) road, (b) building, (c) grass and (d) tree.

$R_v^i(A_v)$ is defined according to the classes of interest. Since the cut procedure minimizes a cost function by removing (i.e., cutting) some of the edges, the weight of a pixel that is connected to the terminal node that corresponds to the correct class should be low as the penalty for assigning that pixel to the correct class should introduce a low cost. We use the posterior probabilities to determine the costs. The posterior probabilities are calculated using (3.8), where $\mathcal{C} = \{\text{building}, \text{high veg.}\}$.

In particular for the separation of buildings from high vegetation the $R_v^i(A_v)$

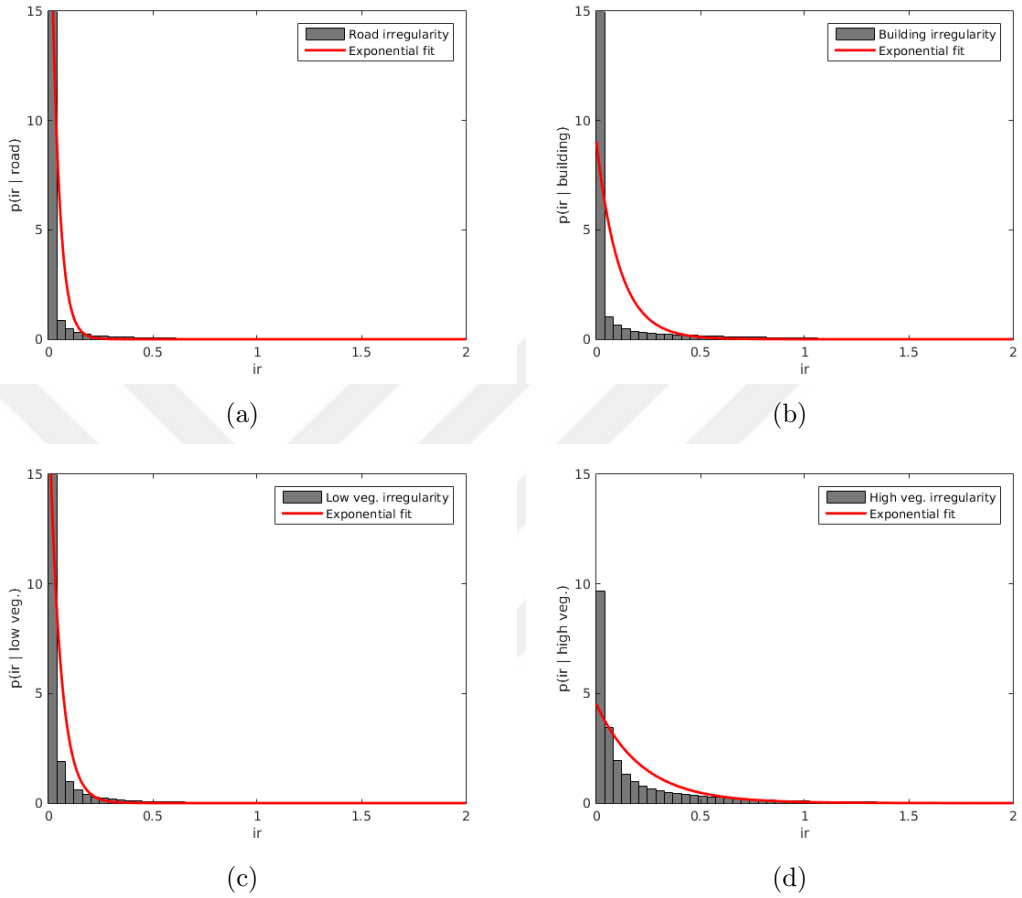


Figure 3.12: Histograms and the estimated probability distributions of each class for the irregularity feature. The irregularity values are modeled using Exponential models. (a) road, (b) building, (c) grass and (d) tree.

is defined as

$$R_v^{\text{NDVI}}(A_v = \text{building}) = -p(\text{building} \mid n), \tag{3.15}$$

$$R_v^{\text{NDVI}}(A_v = \text{high veg.}) = -p(\text{high veg.} \mid n), \tag{3.16}$$

$$R_v^{\text{Irregularity}}(A_v = \text{building}) = -p(\text{building} \mid ir), \tag{3.17}$$

$$R_v^{\text{Irregularity}}(A_v = \text{high veg.}) = -p(\text{high veg.} \mid ir). \tag{3.18}$$

where n is the NDVI and ir is the irregularity value of pixel (node) v. λ_1, λ_2 and λ_3 in (3.9) determine the relative importance of each term.

We define another cost function to discriminate road and grass in the connected components of the low binary mask. However, as it can easily be observed from

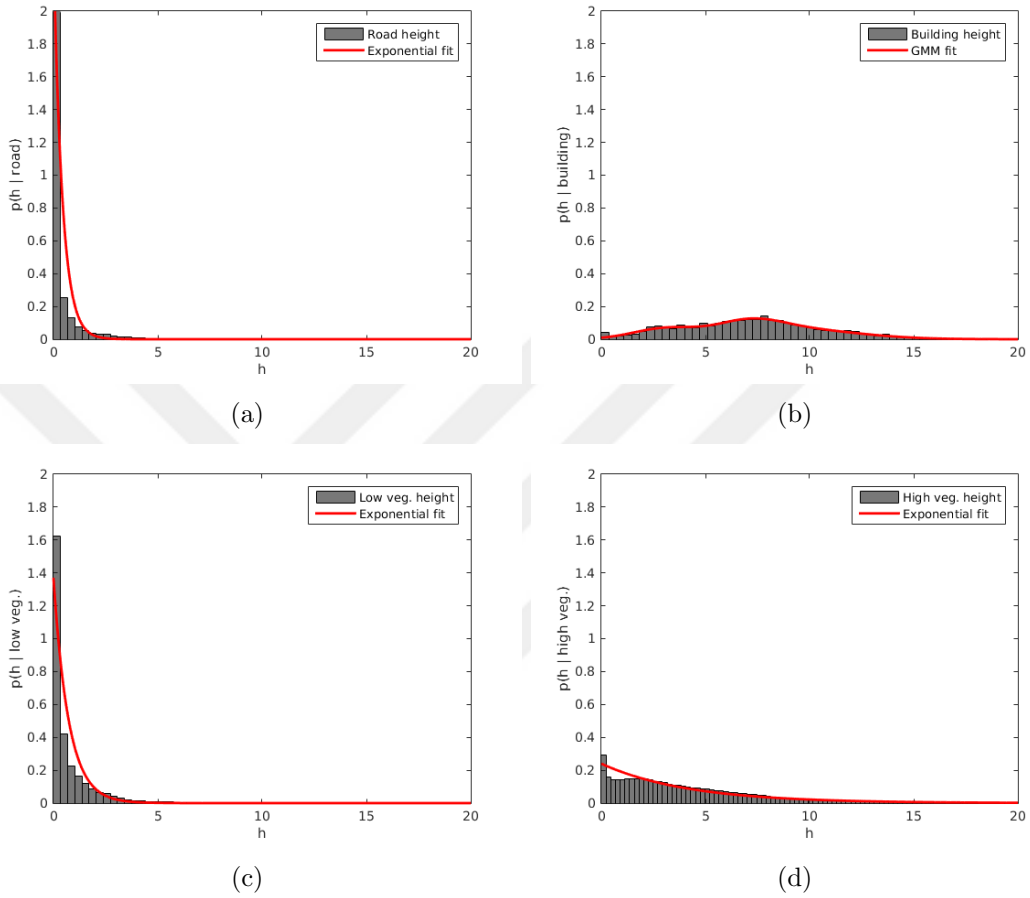


Figure 3.13: Histograms and the estimated probability distributions of each class for the height feature. Height values of roads, grass and trees are modeled using Exponential models, and height values of buildings are modeled by a Gaussian mixture model with 4 components. (a) road, (b) building, (c) grass and (d) tree.

Figure 3.9 that the irregularity is not a convenient feature to separate roads from grass because height variance of the points belonging to either of these two classes is low. Therefore, in this part of the method, we do not use the irregularity feature in the cost function. The cost function is

$$E(A) = \underbrace{\lambda_1 \cdot R^{\text{NDVI}}(A)}_{\text{Data Cost}} + \underbrace{\lambda_2 \cdot B^{\text{spatial}}(A)}_{\text{Smoothness Cost}}. \tag{3.19}$$

For the smoothness cost in (3.19), we follow the same steps we do in (3.11) and (3.12). The posterior probabilities are calculated using (3.8), where $\mathcal{C} =$

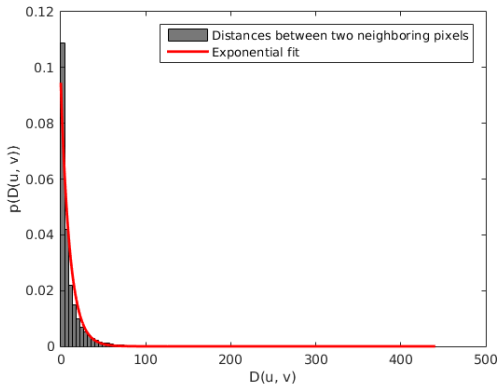


Figure 3.14: Histogram and the estimated probability distribution for the distances between two neighboring pixels. The distances are modeled using Exponential fit.

$\{\text{road}, \text{low veg.}\}$. The terms in the data cost are defined as

$$R_v^{\text{NDVI}}(A_v = \text{road}) = -p(\text{road} \mid n), \tag{3.20}$$

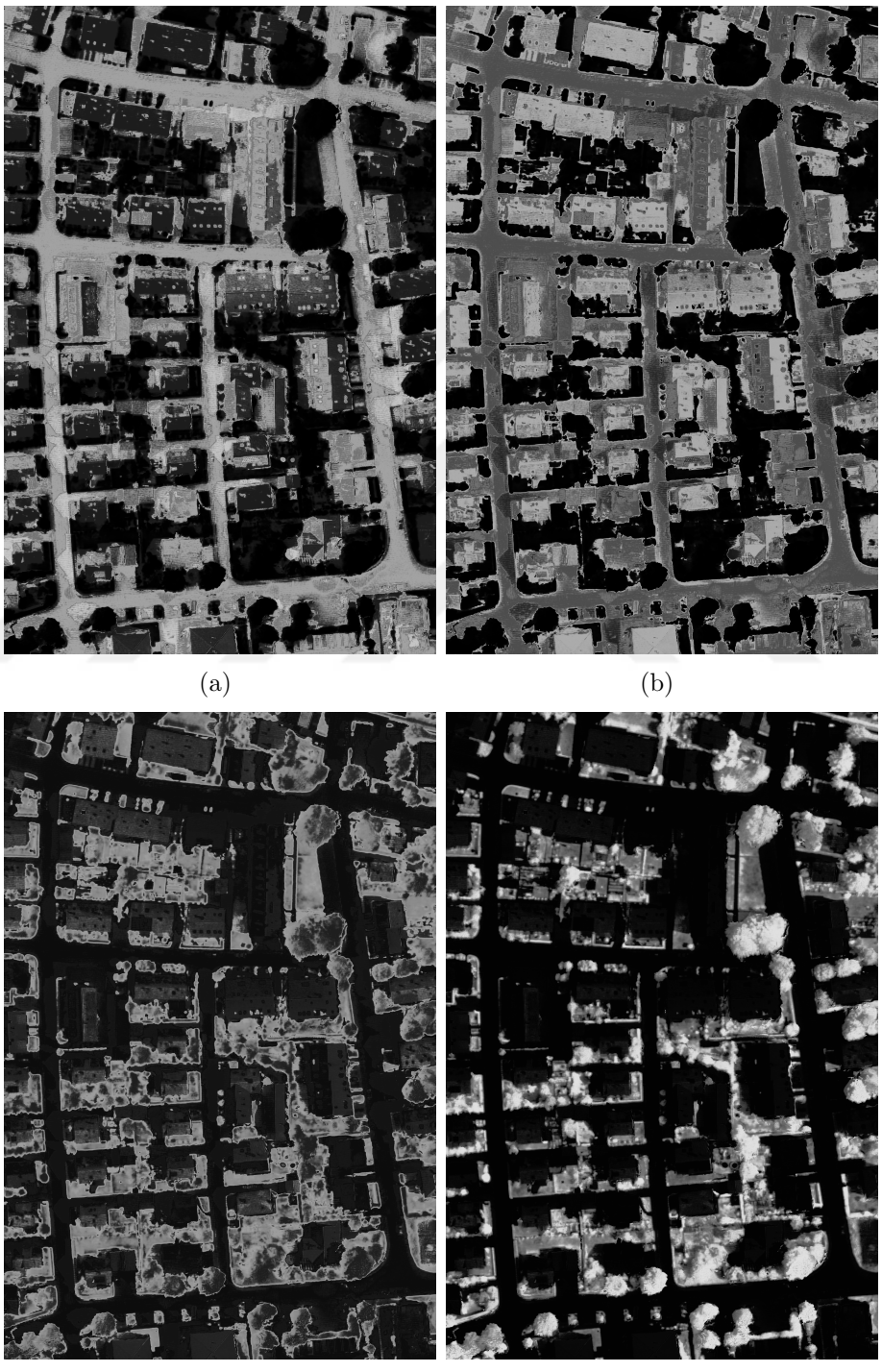
$$R_v^{\text{NDVI}}(A_v = \text{low veg.}) = -p(\text{low veg.} \mid n). \tag{3.21}$$

Finally, we merge the classification maps coming from the low and high pixels to get only one classification map.

3.5.2 Multi-label Optimization Procedure

The graph model for the multi-label optimization procedure [14, 15, 16] is quite similar to the one for the max-flow/min-cut framework. The main difference is that instead of two terminal nodes as in the max-flow/min-cut procedure, there can be any number of terminal nodes in this framework. In our case, there are four terminal nodes because we have four different classes. Another important point is that although there are four terminal nodes, the cut procedure does not have to assign at least one pixel to each of the classes. If none of the pixels fits to a class, the cut procedure may not assign any pixels to that class.

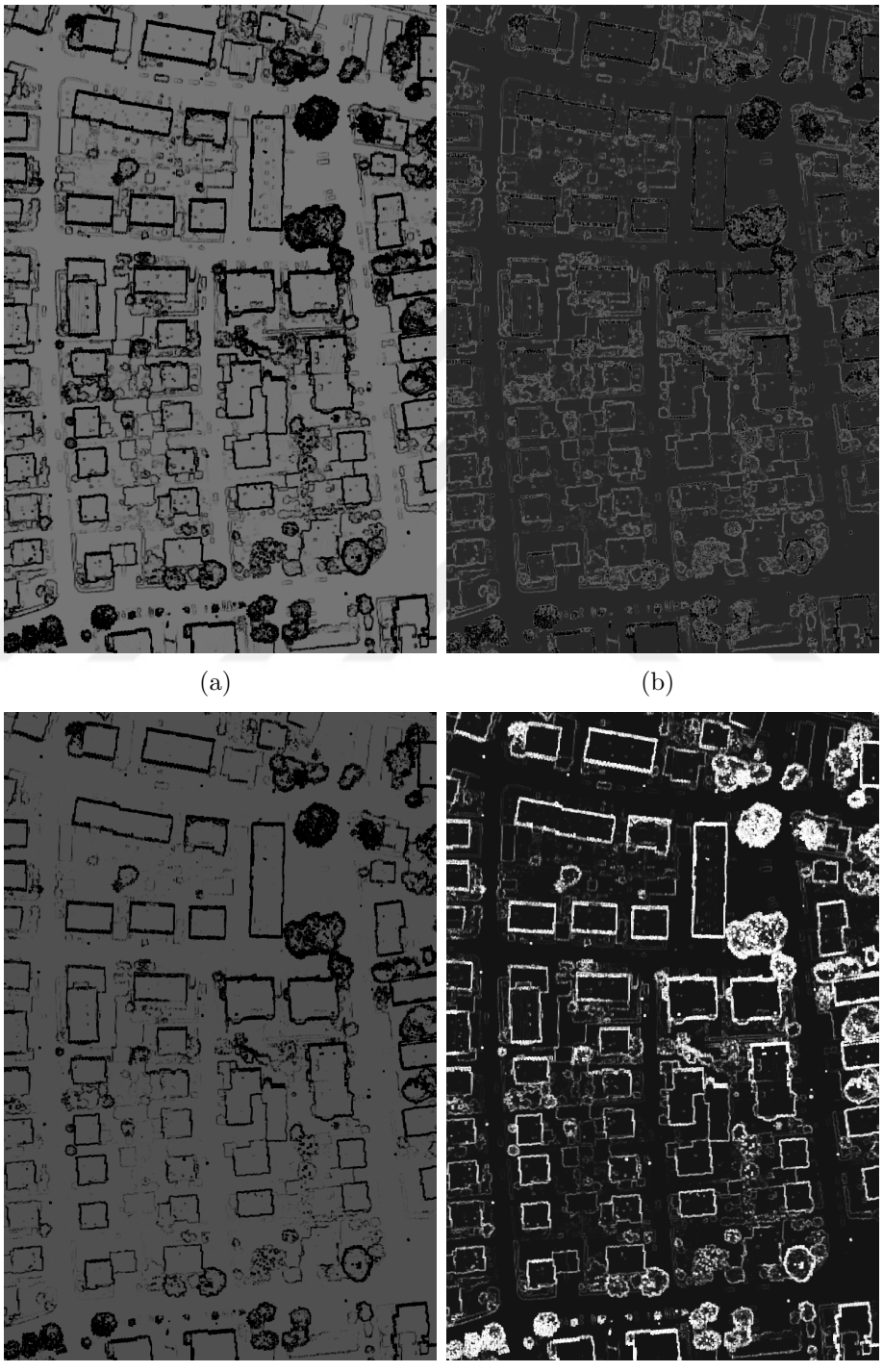
The irregularity is a distinctive feature to detect trees, and NDVI is a convenient feature to separate healthy vegetation (tress and grass) from roads and



(a) (b)

(c) (d)

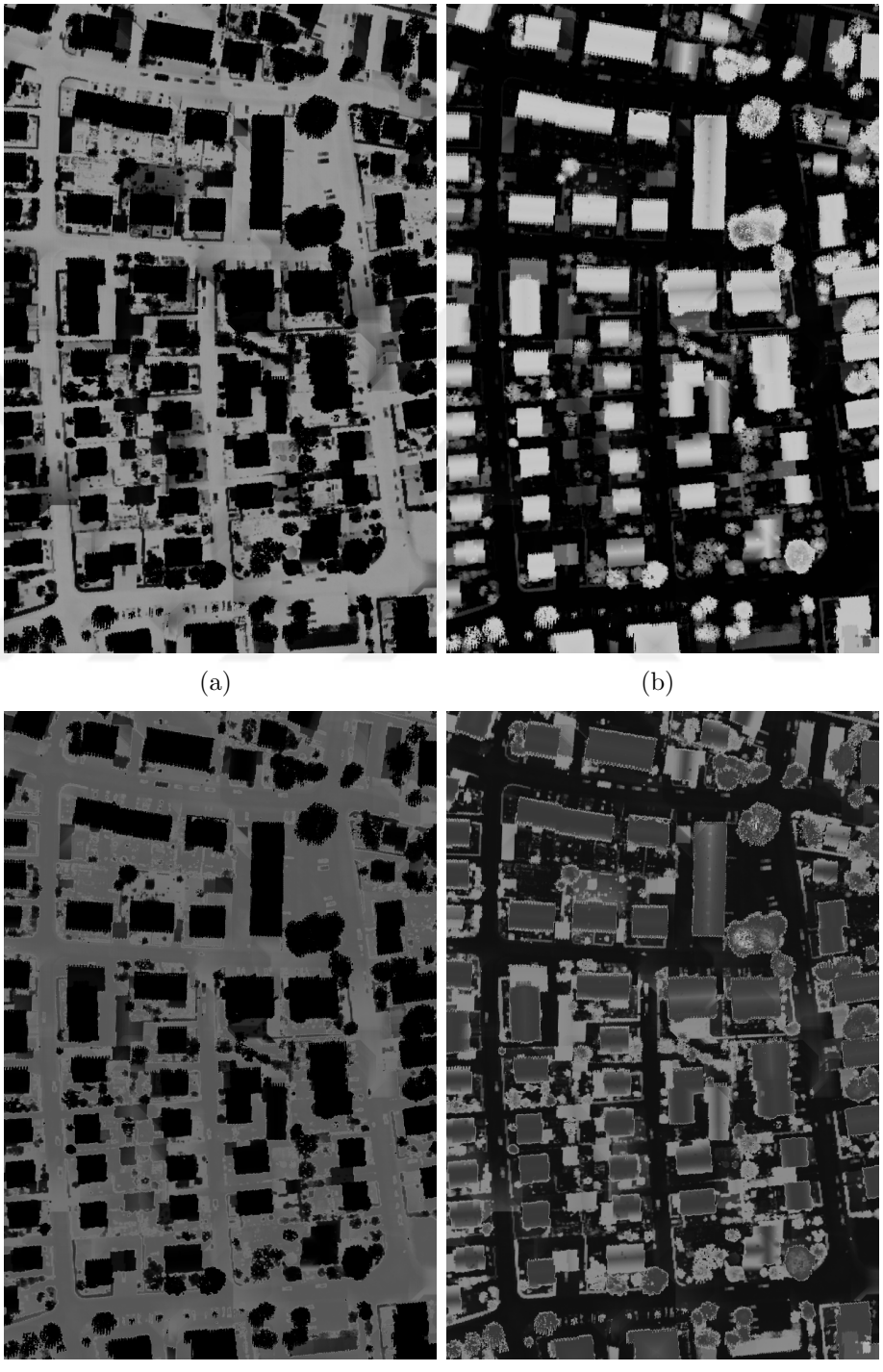
Figure 3.15: Posterior probability map of each class for the ndvi feature. (a) road, (b) building, (c) grass and (d) tree.



(a) (b)

(c) (d)

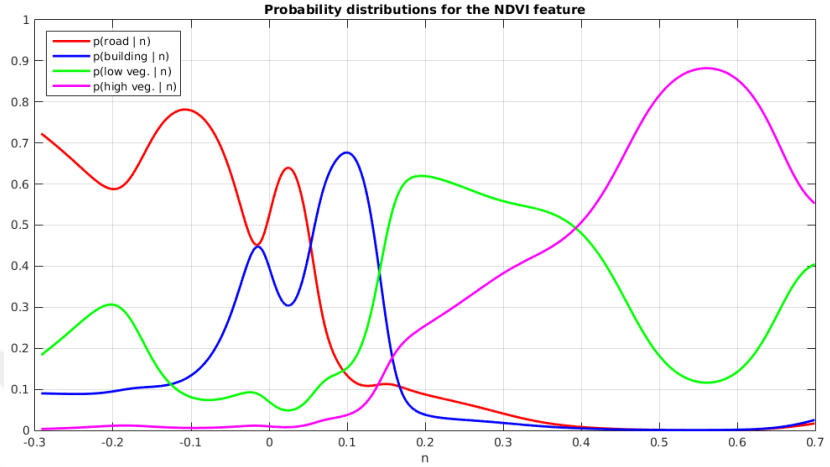
Figure 3.16: Posterior probability map of each class for the irregularity feature. (a) road, (b) building, (c) grass and (d) tree.



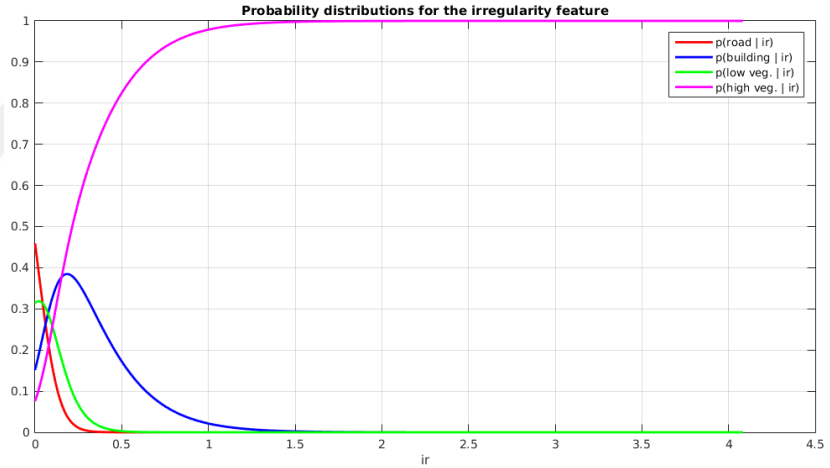
(a) (b)

(c) (d)

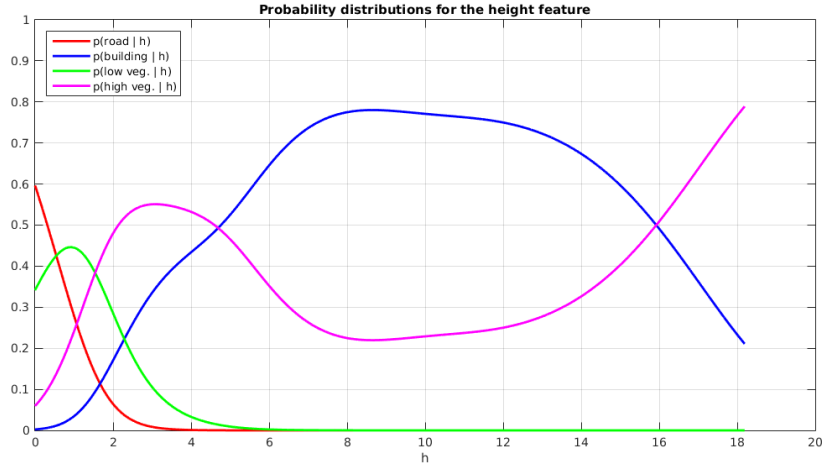
Figure 3.17: Posterior probability map of each class for the height feature. (a) road, (b) building, (c) grass and (d) tree.



(a)



(b)



(c)

Figure 3.18: Posterior probability distributions of each class for (a) NDVI, (b) irregularity, (c) height.

buildings as described in the previous sections. However, since the multi-label optimization framework assigns each pixel in all of the connected components to one of the four classes, an additional feature is required to distinguish roads from buildings because the pixels belonging to either of these classes have similar irregularity and NDVI values. Therefore, we use the relative height information as another feature.

The cost function for the multi-label optimization is

$$E(A) = \underbrace{\lambda_1 \cdot R^{\text{NDVI}}(A) + \lambda_2 \cdot R^{\text{Irregularity}}(A) + \lambda_3 \cdot R^{\text{Height}}(A)}_{\text{Data Cost}} + \underbrace{\lambda_4 \cdot B^{\text{Spatial}}(A)}_{\text{Smoothness Cost}} \tag{3.22}$$

where

$$R^i(A) = \sum_{v \in V} R_v^i(A_v), \quad i \in \{\text{NDVI}, \text{Irregularity}, \text{Height}\}. \tag{3.23}$$

For the smoothness cost in (3.22), we use the same constant and adaptive smoothness cost definitions described in (3.11) and (3.12), respectively. The posterior probabilities are calculated using (3.8), where $\mathcal{C} = \{\text{road}, \text{building}, \text{low veg.}, \text{high veg.}\}$.

Data costs for the NDVI feature are calculated using the equations (3.20), (3.15), (3.21) and (3.16). Costs of building and tree classes for the irregularity feature are determined by (3.17) and (3.18), and the costs of the other classes are defined as

$$R_v^{\text{Irregularity}}(A_v = \text{road}) = -p(\text{road} \mid ir), \tag{3.24}$$

$$R_v^{\text{Irregularity}}(A_v = \text{low veg.}) = -p(\text{low veg.} \mid ir). \tag{3.25}$$

Finally, cost of each class for the height feature is calculated as

$$R_v^{\text{height}}(A_v = \text{road}) = -p(\text{road} \mid h), \tag{3.26}$$

$$R_v^{\text{height}}(A_v = \text{building}) = -p(\text{building} \mid h), \tag{3.27}$$

$$R_v^{\text{height}}(A_v = \text{low veg.}) = -p(\text{low veg.} \mid h), \tag{3.28}$$

$$R_v^{\text{height}}(A_v = \text{high veg.}) = -p(\text{high veg.} \mid h). \tag{3.29}$$

where h is the height value of pixel (node) v.

Chapter 4

Experiments

As described in the Section 3.1, the whole Vaihingen region has been divided into smaller areas to process the data more efficiently by the data set providers. In the experiments, we use the areas having spectral data, DSM data that have been generated from the LiDAR data, and ground-truth. These areas are 1, 3, 5, 7, 11, 15, 28, 30, 34 and 37. The spectral and DSM data of the whole Vaihingen region and the selected areas are illustrated in Figures 4.1 and 4.2.

In this chapter, firstly, we explain the experimental setup. Then, the evaluation criteria to measure how well our frameworks perform are provided. We explain how we optimize the parameters. Finally, we provide both numeric and visual results, and discuss these results at the end of the chapter.

4.1 Experimental Setup

In our experiments, we use 3-fold cross validation. How we divide the areas into three groups is shown in Table 4.1.

There are several parameters that need to be optimized in this thesis. The first two parameters are the ones used for generating the DTMs from the DSMs,

and dividing the data into two as high and low. We use the gray-scale opening method that is explained in detail in Section 3.3.4. We use this method because it works better than other three DTM generation algorithms as described in the same section. The first parameter value we determine is the one for the number of iterations in (3.6). The second one is the height threshold to separate the data as high and low. The remaining parameters are the λ values that are used in the cost functions of max-flow/min-cut for the high pixels, max-flow/min-cut for the low pixels, and multi-label optimization. These λ parameters adjust the relative importance of each term in (3.9), (3.19) and (3.22), respectively. We determine different values for the λs in these equations for constant and adaptive smoothness costs. In order to optimize the parameters explained above, we search the best values for the parameters in certain ranges.

We provide visual results for 10 test areas obtained by using 4 different methods; max-flow/min-cut with constant smoothness cost, max-flow/min-cut with adaptive smoothness cost, multi-label optimization with constant smoothness cost and multi-label optimization with adaptive smoothness cost. We also provide numeric results for each group, as well as overall results, which are calculated from the results for each group.

In addition to extracting histograms for each feature by using all the training data as shown in Figures 3.11, 3.12, 3.13, we also extract histograms for each class and feature by using only 5% of the training data. We do not observe a big difference when we use the whole training data or 5% of the training data; the histograms and the fitted models are almost identical.

4.2 Evaluation Criteria

In order to evaluate the performance of our frameworks, we use the pixel-based and the object-based metrics defined in [79] and [80]. For the pixel-based evaluation, the metrics described in the papers are *correctness* and *completeness*; *completeness* is the fraction of the pixels in the ground-truth that were detected and

Table 4.1: Data division for the 3-Fold cross validation.

Group 1		**Group 2**		**Group3**	
Train	**Test**	**Train**	**Test**	**Train**	**Test**
area 1	area 5	area 1	area 3	area 3	area 1
area 7	area 15	area 7	area 11	area 11	area 7
area 28	area 37	area 28	area 30	area 30	area 28
area 3		area 5	area 34	area 34	
area 11		area 15		area 5	
area 30		area 37		area 15	
area 34				area 37	

correctness is the percentage of the detected pixels that match the ground-truth. These values are calculated from pixel-based confusion matrices. *Correctness* and *completeness* of a class that is located at the ith row of the confusion matrix is calculated as

$$Correctness_i = \frac{Conf_{ii}}{\sum_{j=1}^{\#\text{ of Classes}} Conf_{ji}} \tag{4.1}$$

and

$$Completeness_i = \frac{Conf_{ii}}{\sum_{j=1}^{\#\text{ of Classes}} Conf_{ij}} \tag{4.2}$$

where $Conf$ is the confusion matrix and # of classes is 4 in our case. *Correctness* and *completeness* are also known as precision and recall, respectively, in the literature.

According to the object-based metric described in the papers, firstly, connected components in the predicted classification map are found. Then, the components having smaller area than $50m^2$ are removed. If at least 50% of a component matches with the correct reference object, that component is counted as true positive. Object based correctness and completeness are denoted by $correctness_{50}$ and $completeness_{50}$, respectively, and calculated from object-based confusion matrices using (4.1) and (4.2).

F1-score is the evaluation metric, which is used by the International Society for Photogrammetry and Remote Sensing (ISPRS) to rank the methods according to how well they perform on the competition data sets. We also calculate F1-scores

from both pixel-based and object-based confusion matrices. F1-score for a class is defined as

$$\text{F1-score} = \frac{2 \cdot \mathit{Correctness} \cdot \mathit{Completeness}}{\mathit{Correctness} + \mathit{Completeness}}. \tag{4.3}$$

Final evaluation metric we use is the accuracy, which is calculated from confusion matrices.

4.3 Parameter Selection

We search the best values for the number of iterations parameters in the range $1, 2, \cdots, 30$ and for the height thresholds in the range $0.3, 0.4, \cdots, 5$. When determining the best parameters for each group, we consider buildings and trees as one class (high), and grass and roads as another class (low). Then, in each iteration we calculate confusion matrices for all of the train images in each group and sum them. We calculate an accuracy from the overall confusion matrix for each group. Finally, we record the parameters when the accuracy for each group reaches the highest value. The resulting values for parameters can be seen in Table 4.2.

We optimize the λ parameters in the cost functions of 4 methods for each group separately. In order to compute an accuracy for a group, we generate confusion matrices from the training areas of that group and sum them. We then calculate an accuracy from this matrix. We search the best parameters for the λs in the cost functions for the max-flow/min-cut frameworks with smoothness cost in the range $1, 2, \cdots, 15$. If the highest accuracy is achieved when one of the λs is 15, we extend the search space from 1-15 to 1-30. We keep extending the search spaces by 15 until the highest accuracy is obtained when all of the λ values are not equal to boundary values of the search spaces. As explained in Section 3.5, we set 1 to the cost of separating two nodes for the constant smoothness cost and use the probability distribution function that is illustrated in Figure 3.14 to determine the cost of cutting the edge between two nodes for the adaptive smoothness cost. However, as it can be seen from Figure 3.14 that values the pdf

can take are much more smaller than 1. For this reason, when determining the λ parameters for the smoothness cost in (3.9), (3.19) and (3.22) with adaptive smoothness, we multiply the search space by 10. We follow the same strategy explained above to find the λ values for the data costs in the same equations. For the max-flow/min-cut [12, 13] and multi-label optimization [14, 15, 16] algorithms we use the implementations given at [81]. However, the multi-label optimization tool is designed for the edge between two neighboring nodes, and a node and a terminal node in the graph to take only integer values. We use the same strategy explained above to optimize parameters for the multi-label optimization methods but when optimizing the λs parameters in (3.22), we multiply the search range by 1000 to use 3-digit precision. Tables 4.3, 4.4 and 4.5 show the values, which are divided by 1000 to get rid of lots of zeros, for λ parameters.

Table 4.2: Parameters for dividing the data into two groups as high and low.

Parameter	**Group 1**	**Group 2**	**Group 3**
# of Iterations	22	22	19
Height	1.5	1.5	1.1

Table 4.3: The best values for λs in the cost function of the max-flow/min-cut for the high pixels.

Smoothness Cost	**Parameter**	**Group 1**	**Group 2**	**Group 3**
Constant	λ_1	5	10	5
	λ_2	1	4	2
	λ_3	7	17	22
Adaptive	λ_1	4	4	4
	λ_2	1	1	1
	λ_3	340	270	340

Table 4.4: The best values for λs in the cost function of the max-flow/min-cut for the low pixels.

Smoothness Cost	**Parameter**	**Group 1**	**Group 2**	**Group 3**
Constant	λ_1	7	5	10
	λ_2	12	1	16
Adaptive	λ_1	9	14	1
	λ_2	140	410	60

Table 4.5: The best values for λs in the cost function of the multi-label optimization. The parameters are divided by 1000.

Smoothness Cost	Parameter	Group 1	Group 2	Group 3
Constant	λ_1	14	10	12
	λ_2	9	6	9
	λ_3	12	8	11
	λ_4	24	12	20
Adaptive	λ_1	8	2	2
	λ_2	2	1	1
	λ_3	7	2	2
	λ_4	490	90	160

4.4 Results

Aerial data, ground-truth, DSM, and DTM, NDSM and connected components using the parameters shown in Table 4.2 for all of the test areas in each group can be found in Figures 4.3 to 4.12.

We calculate pixel-based confusion matrices from the results obtained by using 4 methods for each group. Since we have 4 different methods, we provide 4 different pixel-based confusion matrices for each group. In order to generate a confusion matrix for each method and group, we calculate confusion matrices from all of the test areas in each group, and sum them. We also calculate an accuracy from each confusion matrix. Pixel-based confusion matrices with *correctness*, *completeness* and accuracy values for each method and group can be seen in Tables A.1, A.2 and A.3. Following exactly the same steps, we calculate object-based confusion matrices, which are shown in Tables A.4, A.5 and A.6.

We calculate F1-scores for each group, as well as each area in all 3 groups. Pixel-based F1-scores can be found in Tables A.7, A.8, A.9 and A.10, and object-based F1-scores can be seen in Tables A.11, A.12, A.13 and A.14. We also calculate an overall pixel-based F1-score for each class and method by taking average of the F1-scores of that class obtained by that method in all 3 groups in Tables A.7 to A.10. Repeating this operation, we calculate an overall object-based F1-score for each class from Tables A.11 to A.14. We also calculate standard

deviation of F1-score for each class. Pixel-based and object based overall F1-scores are shown in Tables 4.6 and 4.7. We calculate an overall recall value and a standard deviation for each class, as well. The overall pixel based and object based recall values can be found in Tables 4.8 and 4.9. Finally, we calculate overall accuracy for each method, by taking average of accuracies calculated from confusion matrices obtained by that method in all 3 groups. We also calculate standard deviation of the accuracy for each method. Pixel-based and object-based overall accuracies can be found in Tables 4.10 and 4.11.

Visual results for all of the test areas are illustrated in Figures 4.13 to 4.22. Classification maps generated by the two cut frameworks with constant and adaptive smoothness cost, and the ground-truth can be seen in these figures.

Figure 4.1: Aerial data of the whole Vaihingen region and the selected areas.

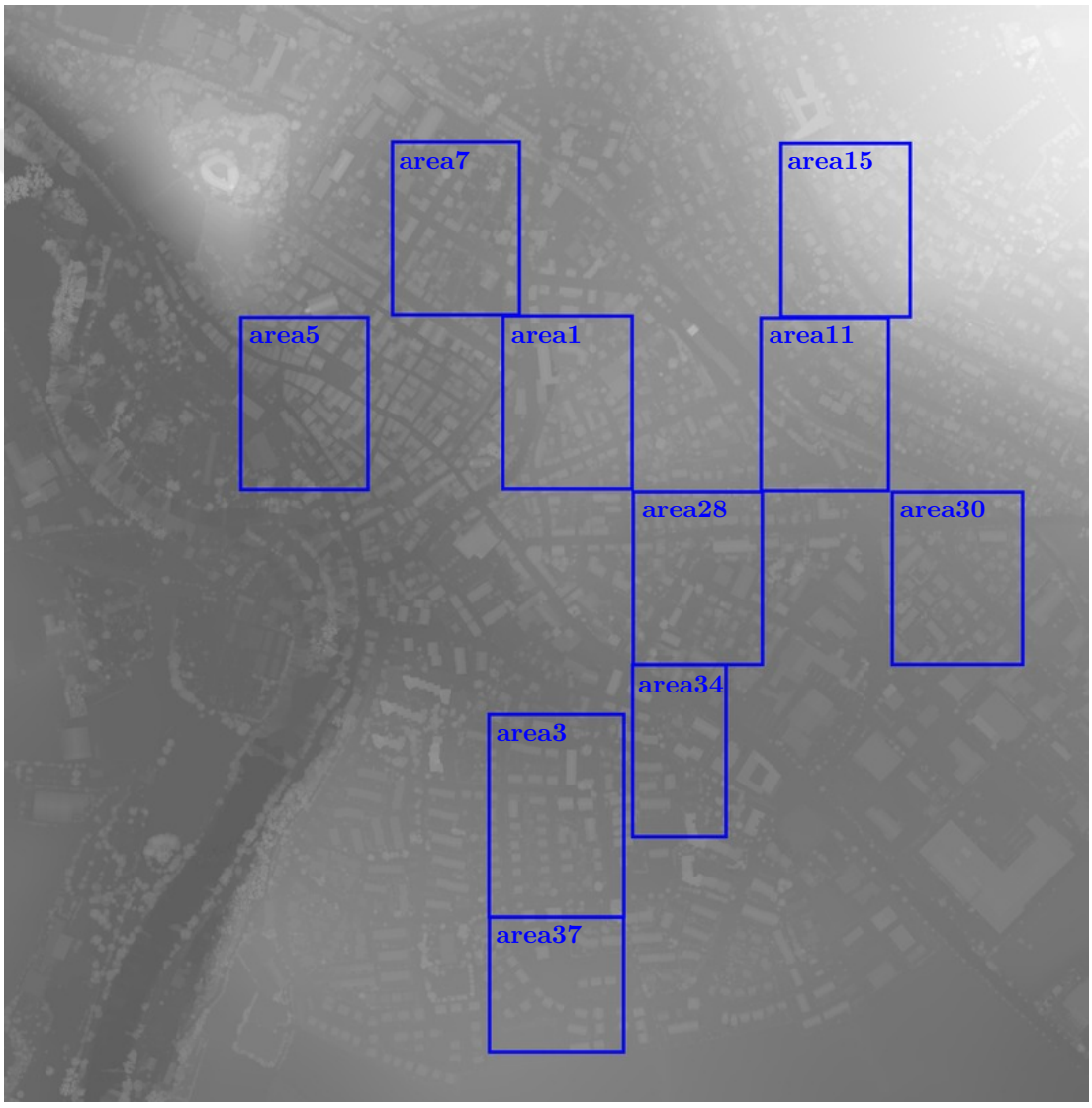


Figure 4.2: DSM data of the whole Vaihingen region and the selected areas.

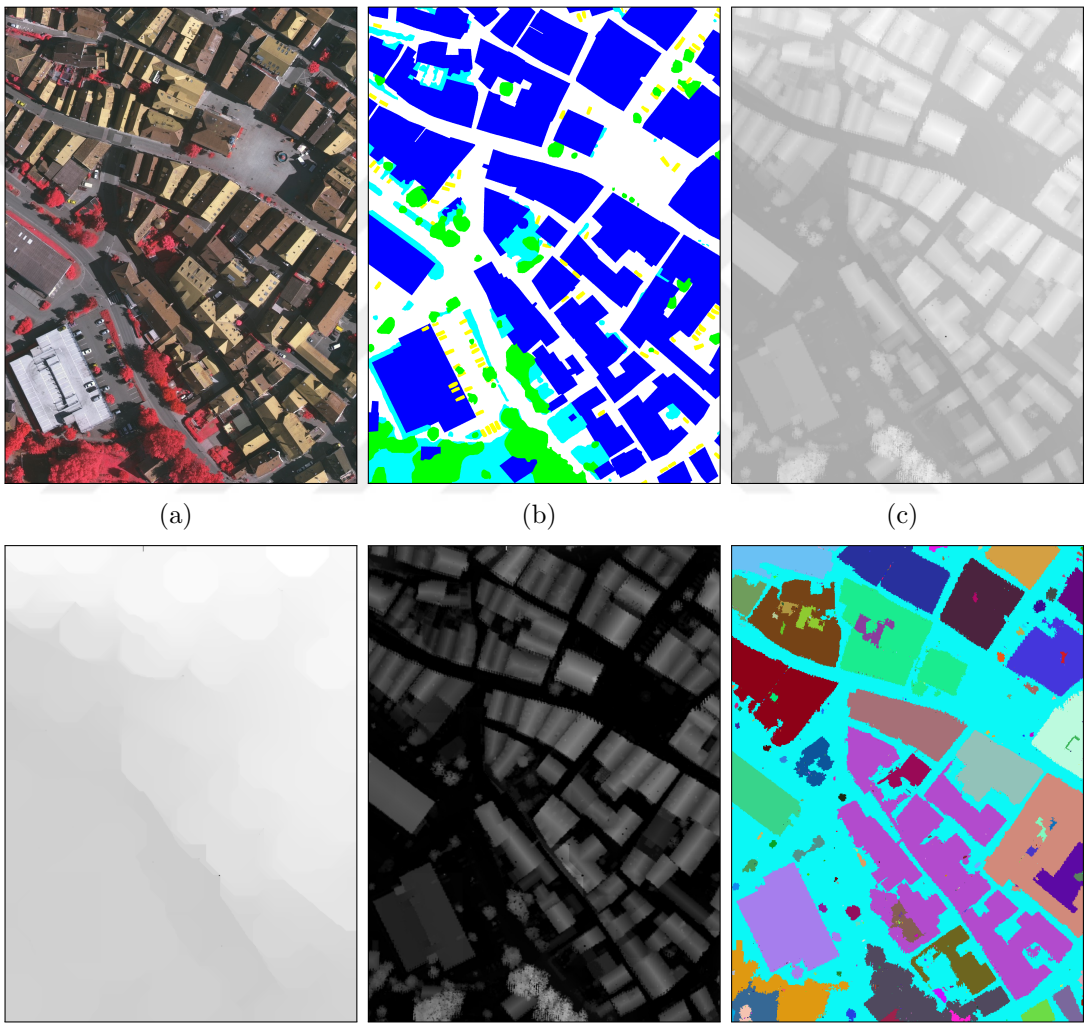
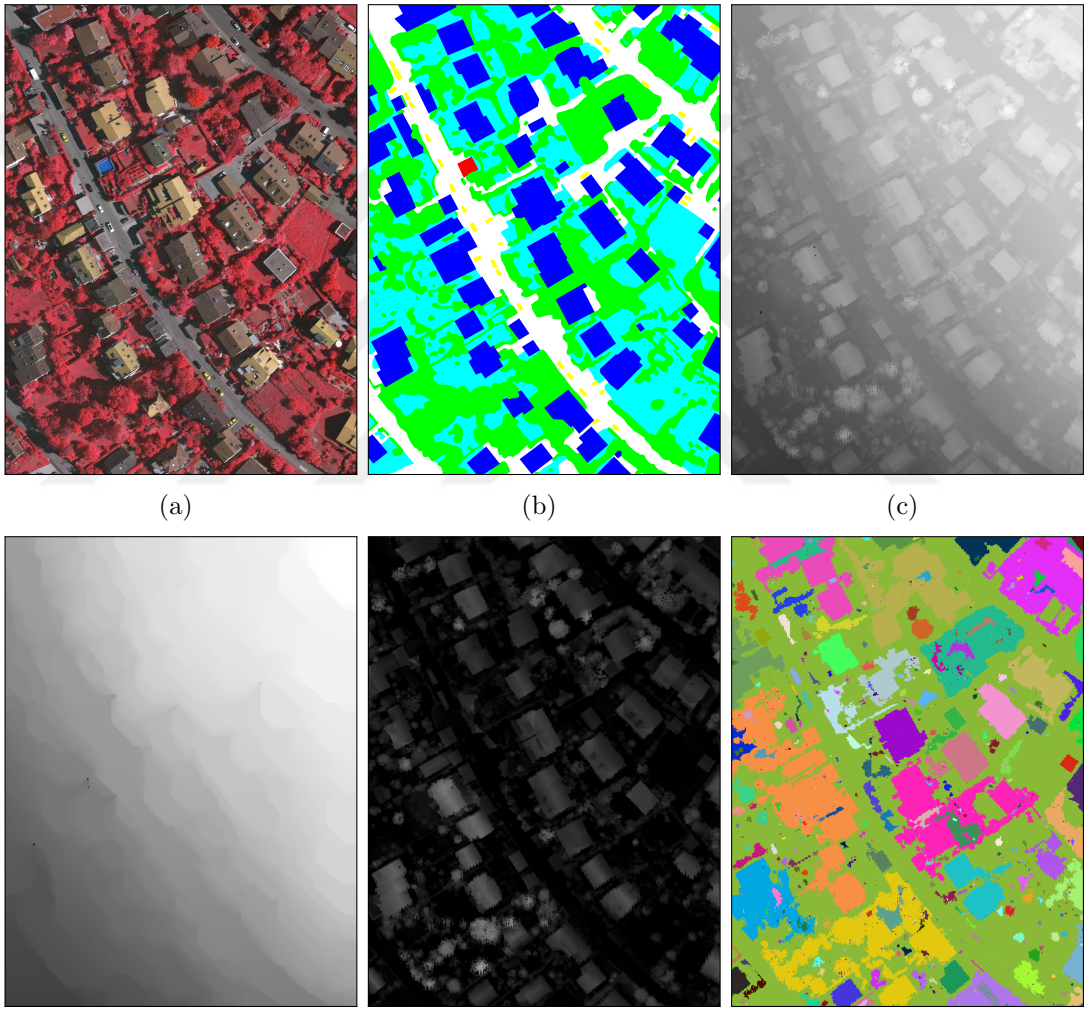


Figure 4.3: Group 1, area 5. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.



(a) (b) (c)

(d) (e) (f)

Figure 4.4: Group 1, area 15. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

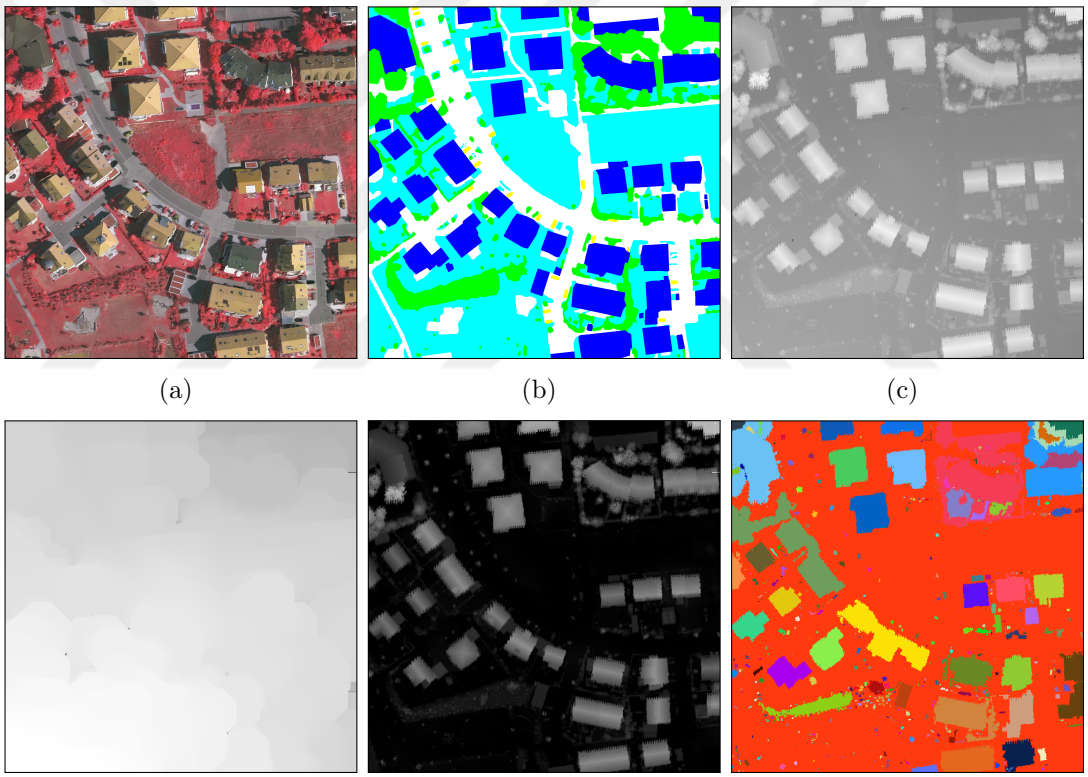


Figure 4.5: Group 1, area 37. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

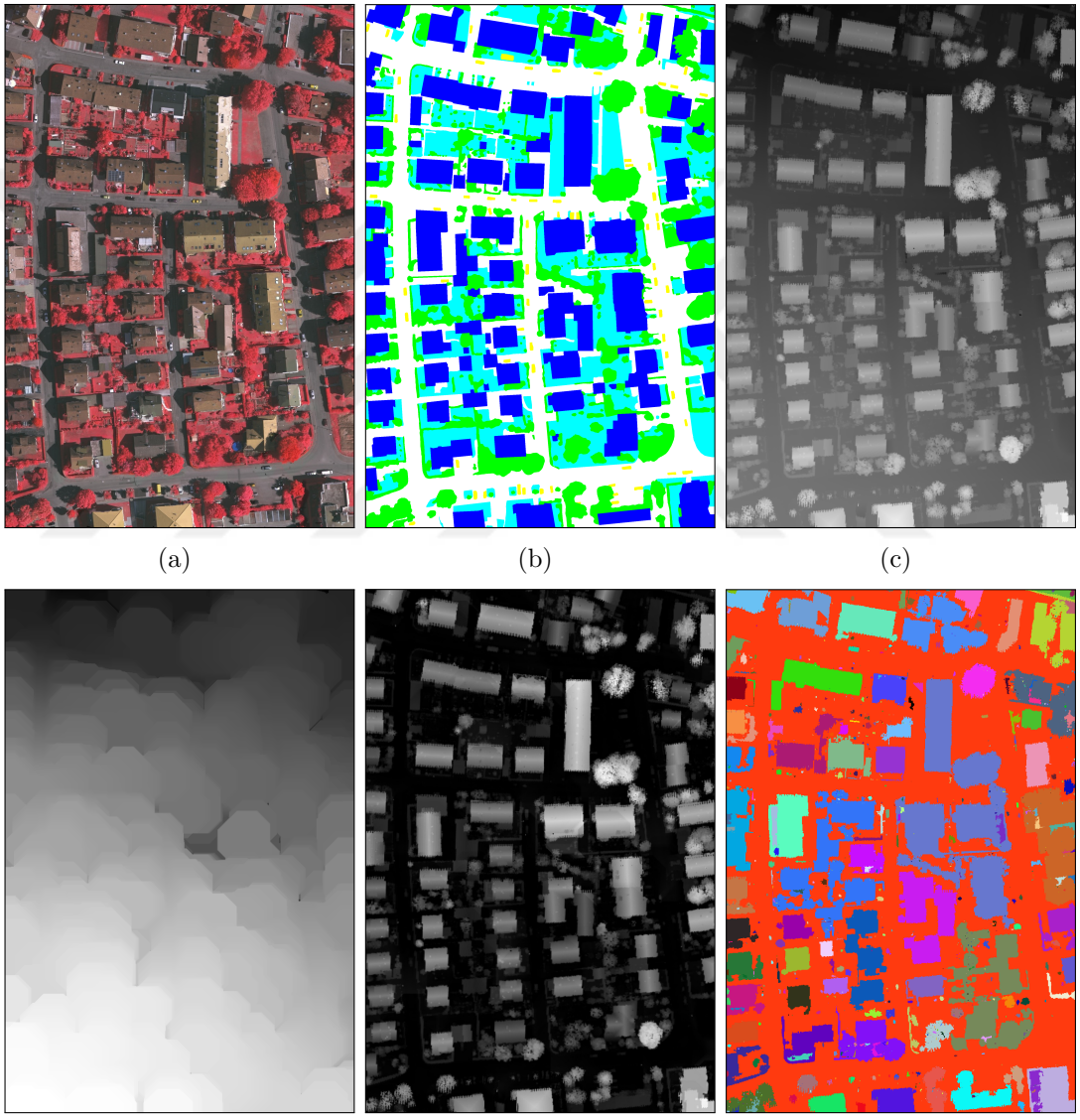


Figure 4.6: Group 2, area 3. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

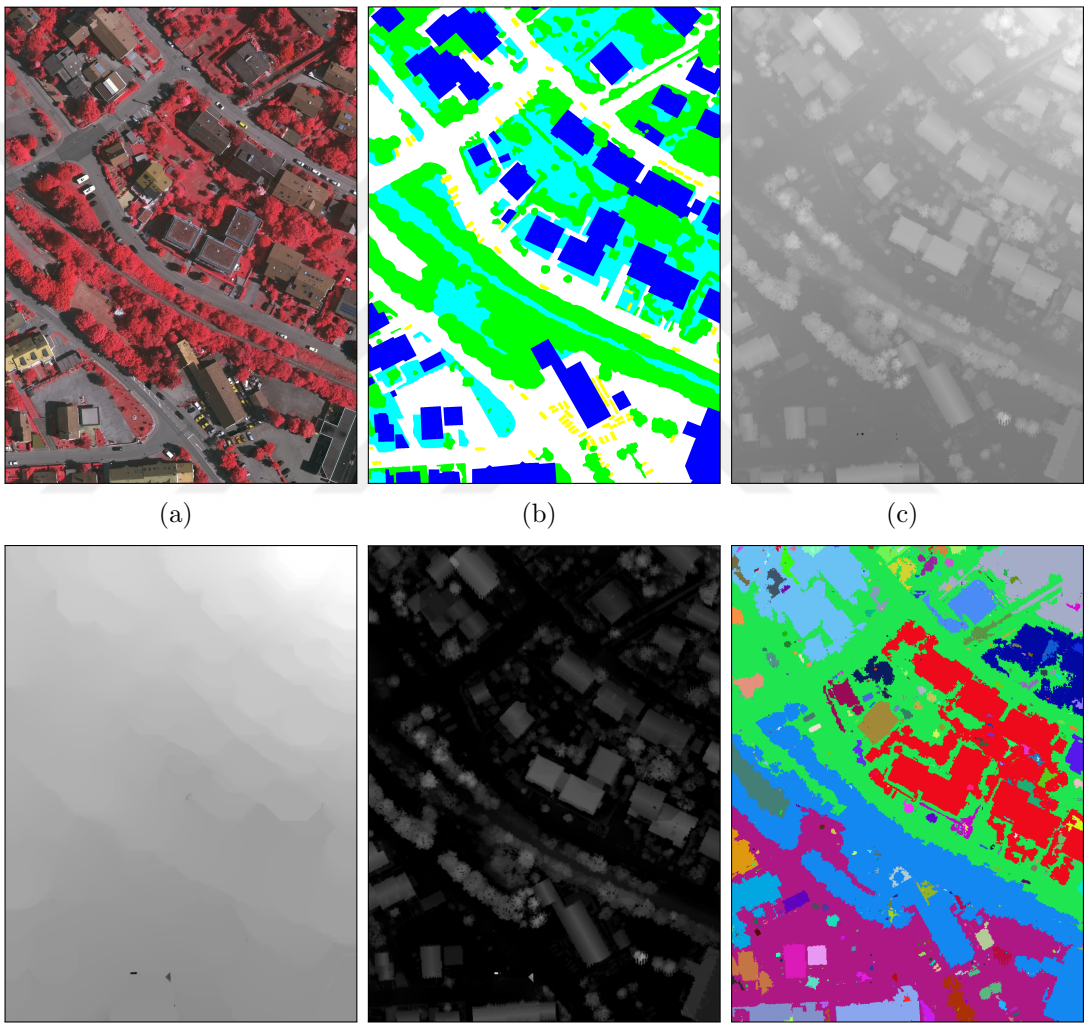


Figure 4.7: Group 2, area 11. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

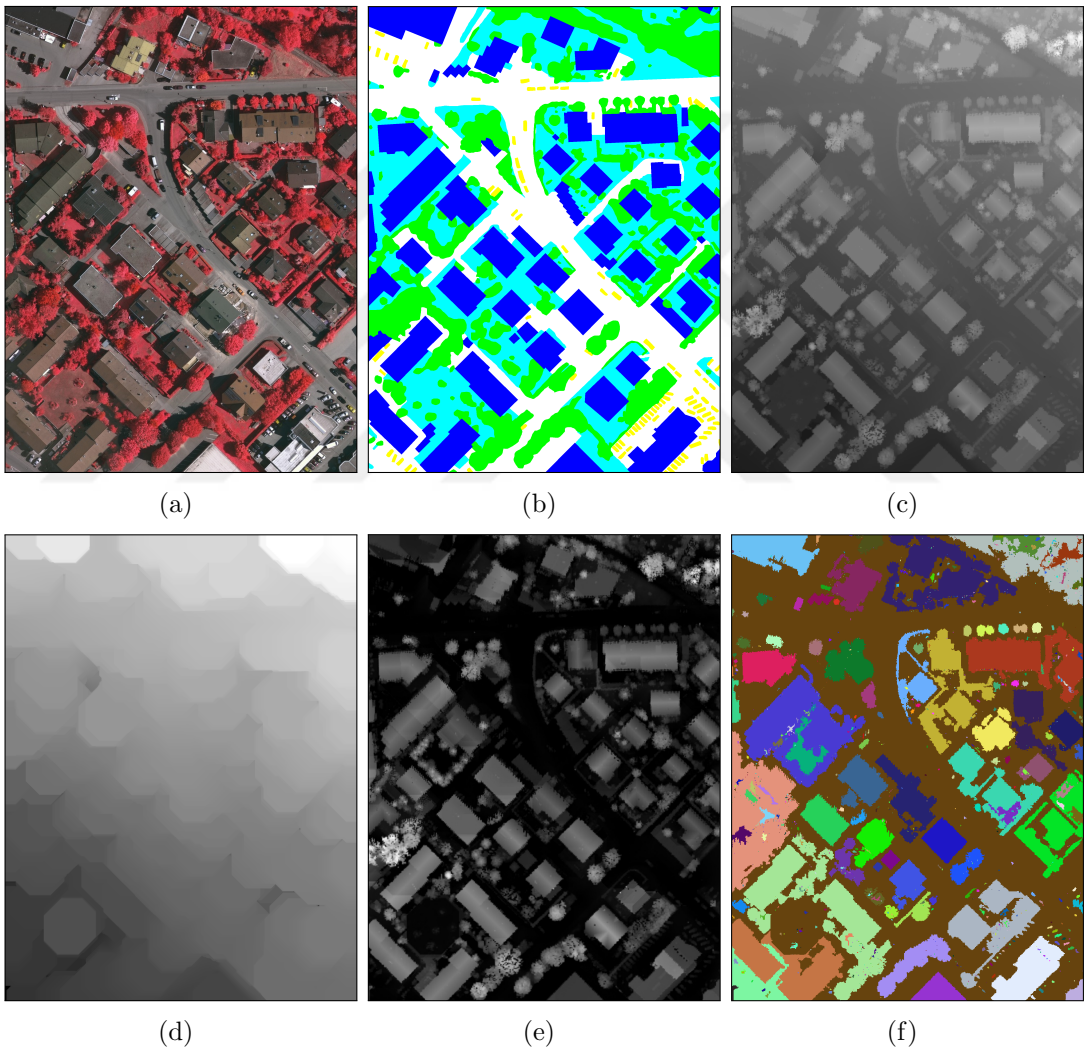


Figure 4.8: Group 2, area 30. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

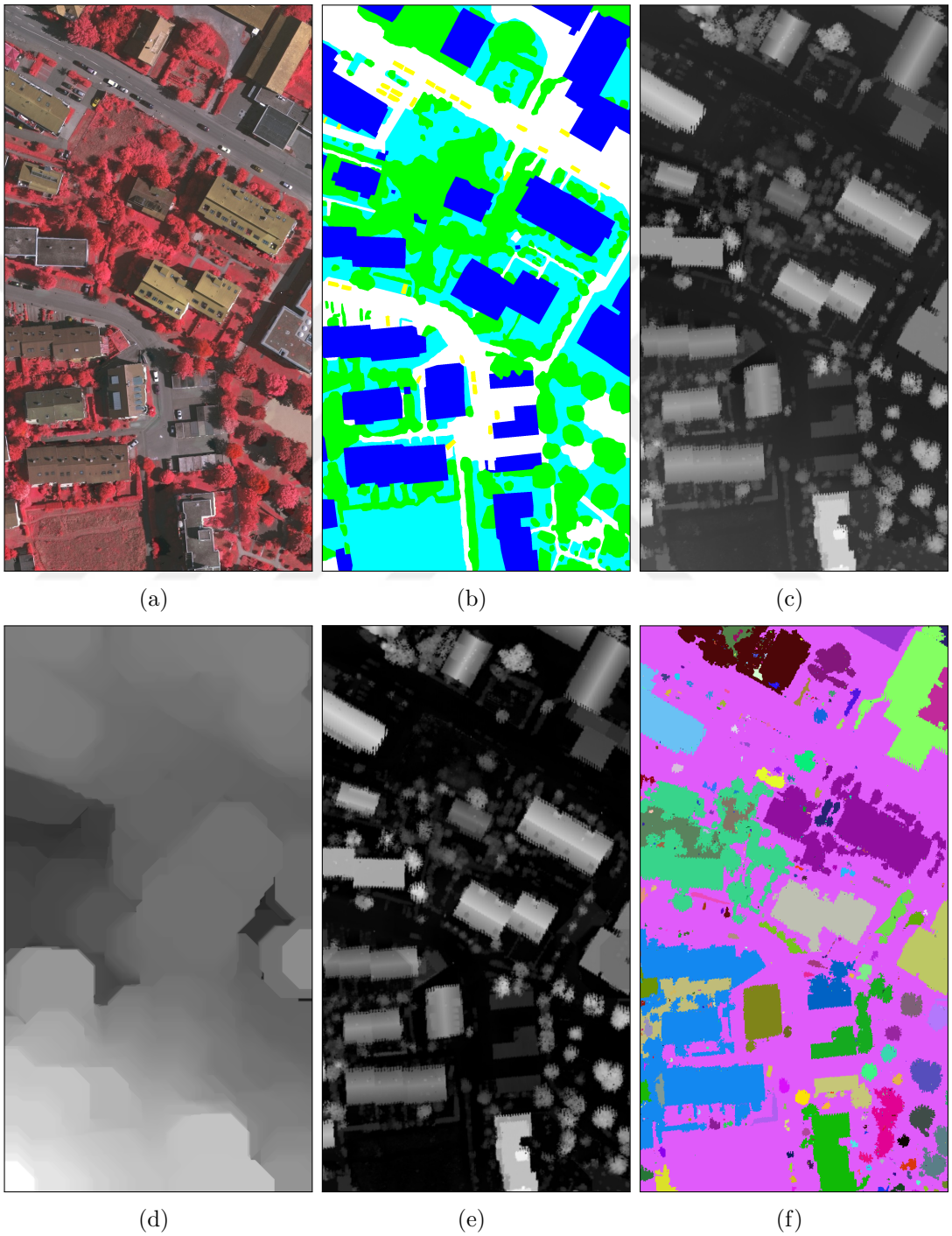


Figure 4.9: Group 2, area 34. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

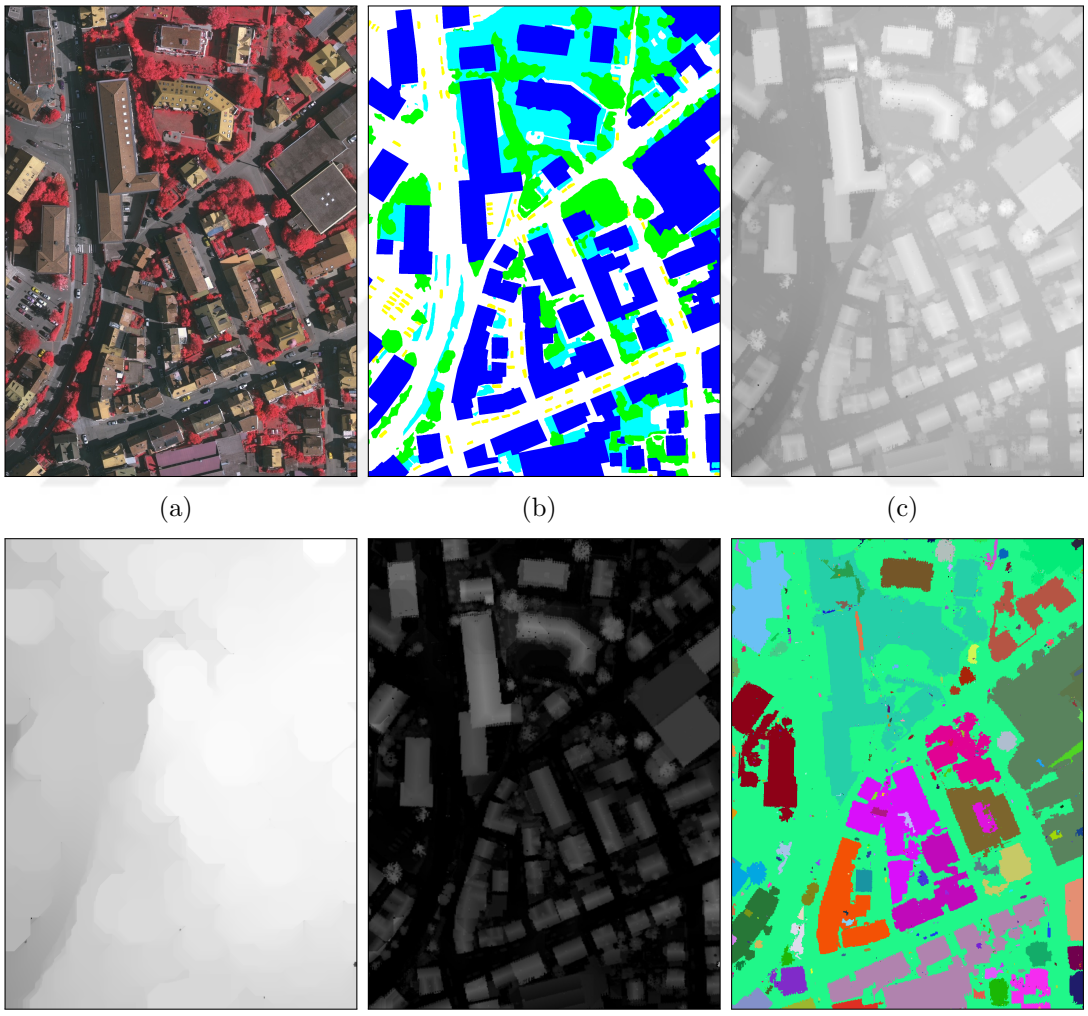


Figure 4.10: Group 3, area 1. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

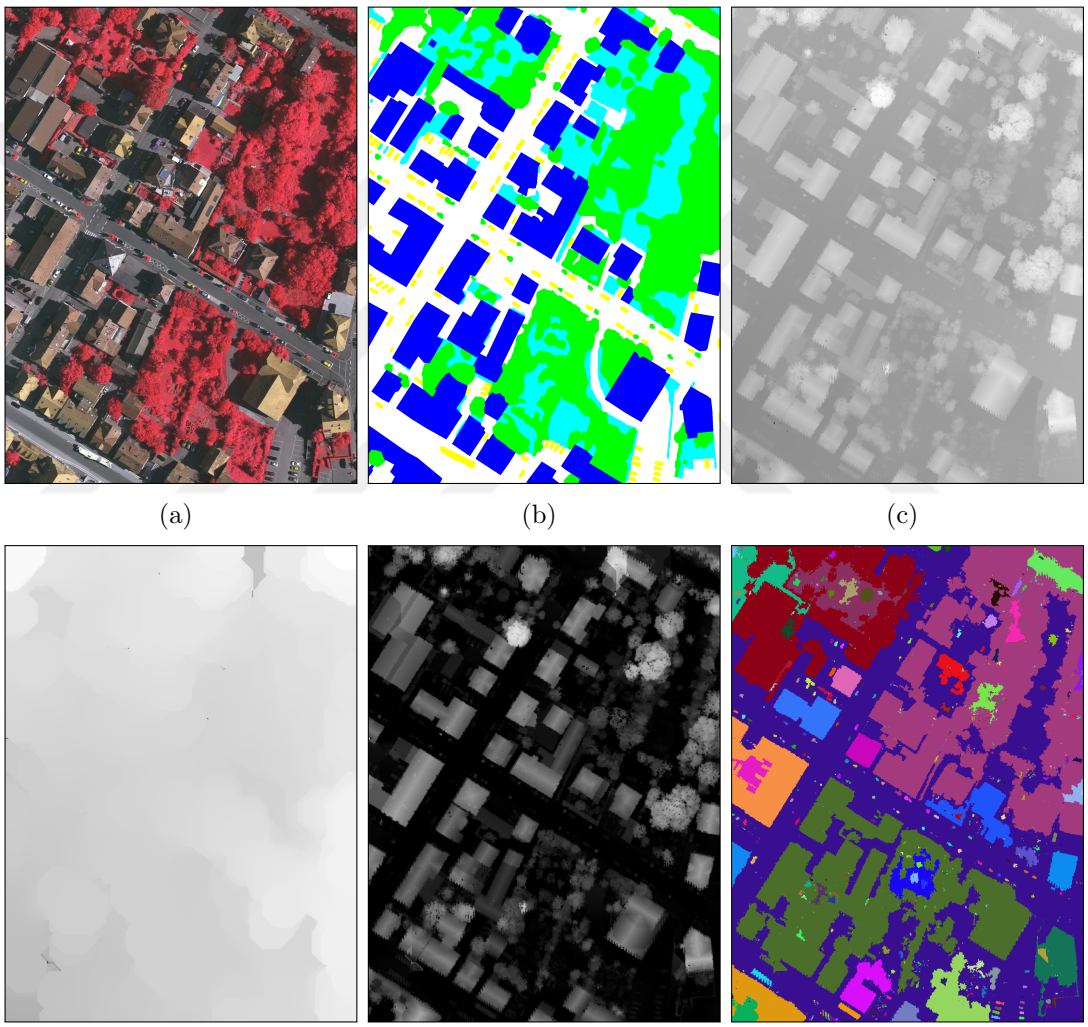


Figure 4.11: Group 3, area 7. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

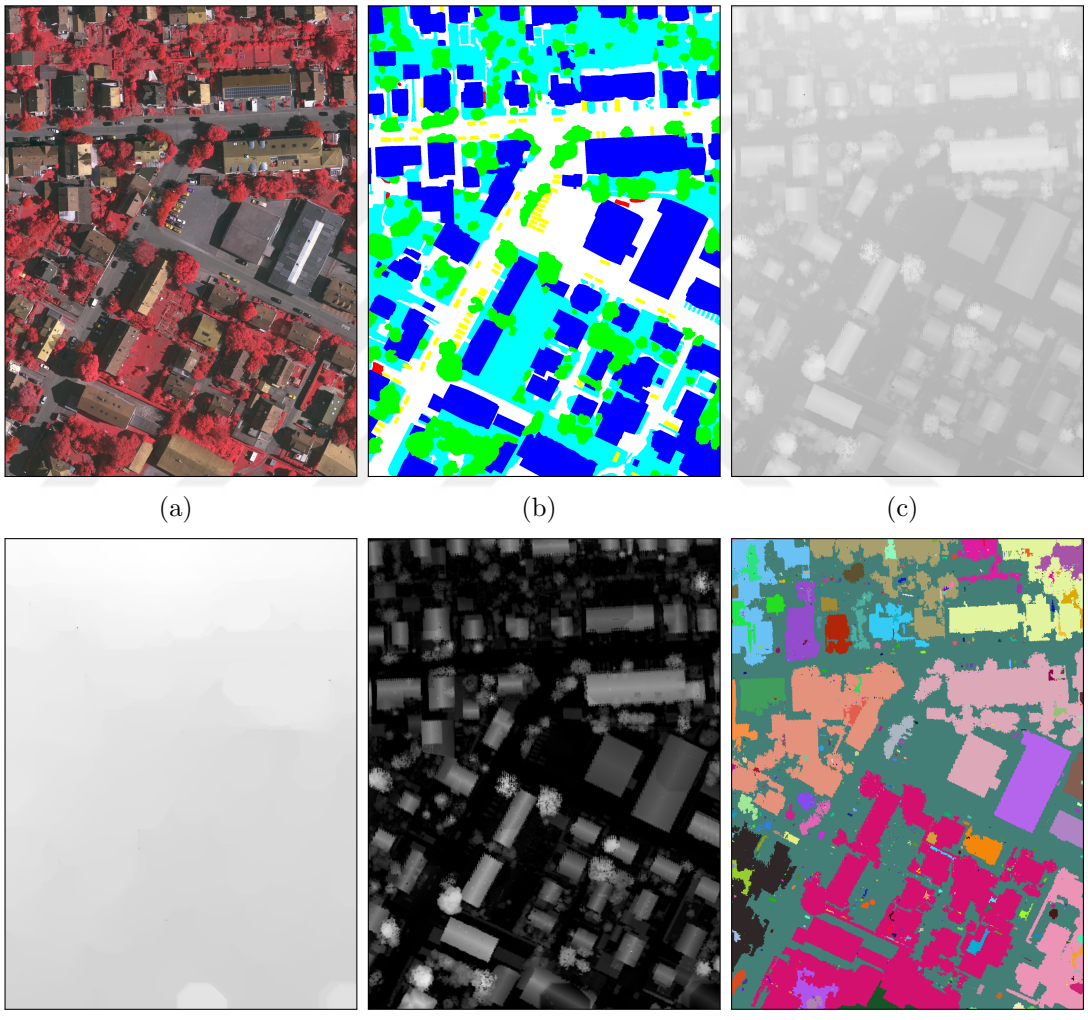


Figure 4.12: Group 3, area 28. (a) Aerial, (b) Ground-truth, (c) DSM, (d) DTM, (e) NDSM, (f) Connected components.

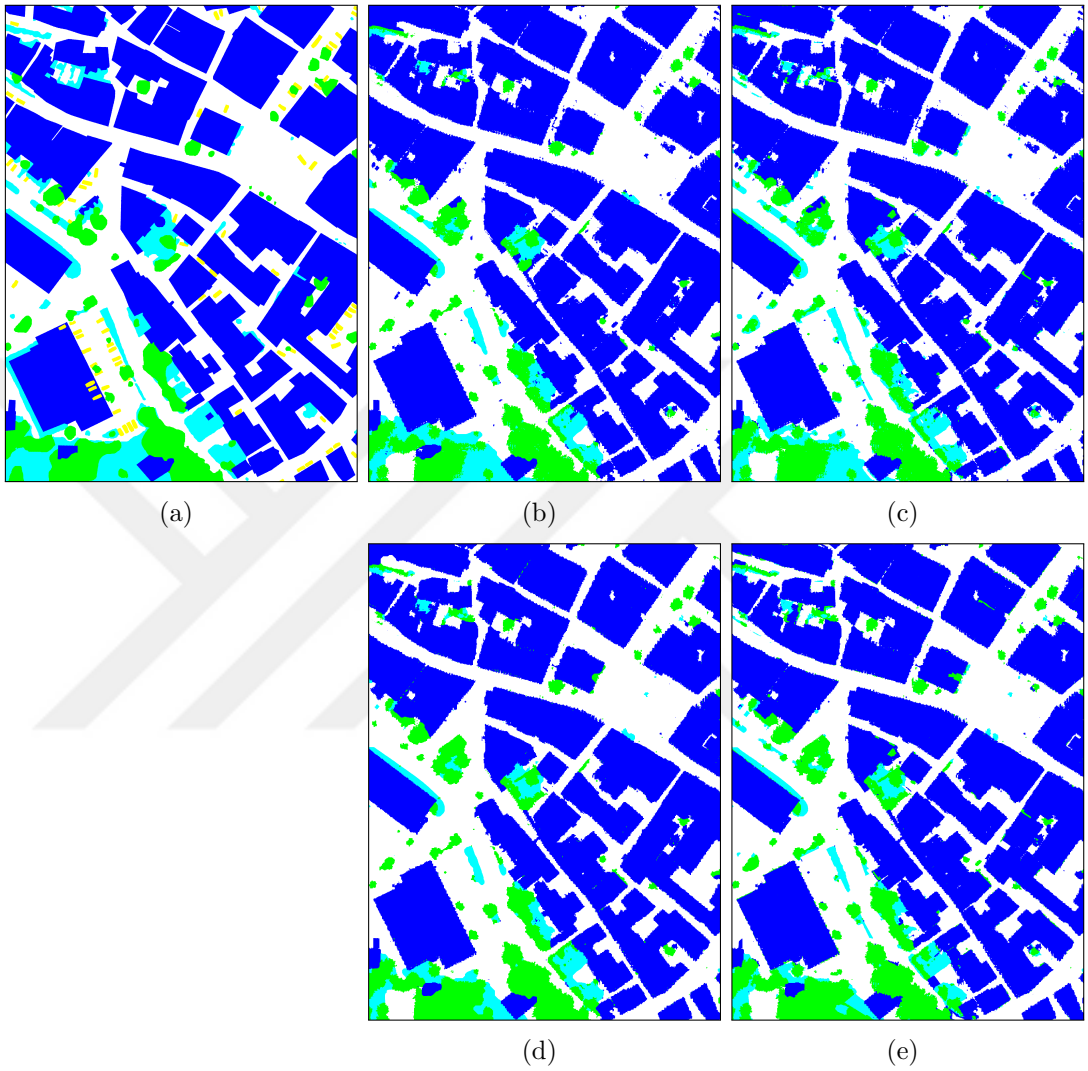


Figure 4.13: Results for group 1, area 5. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

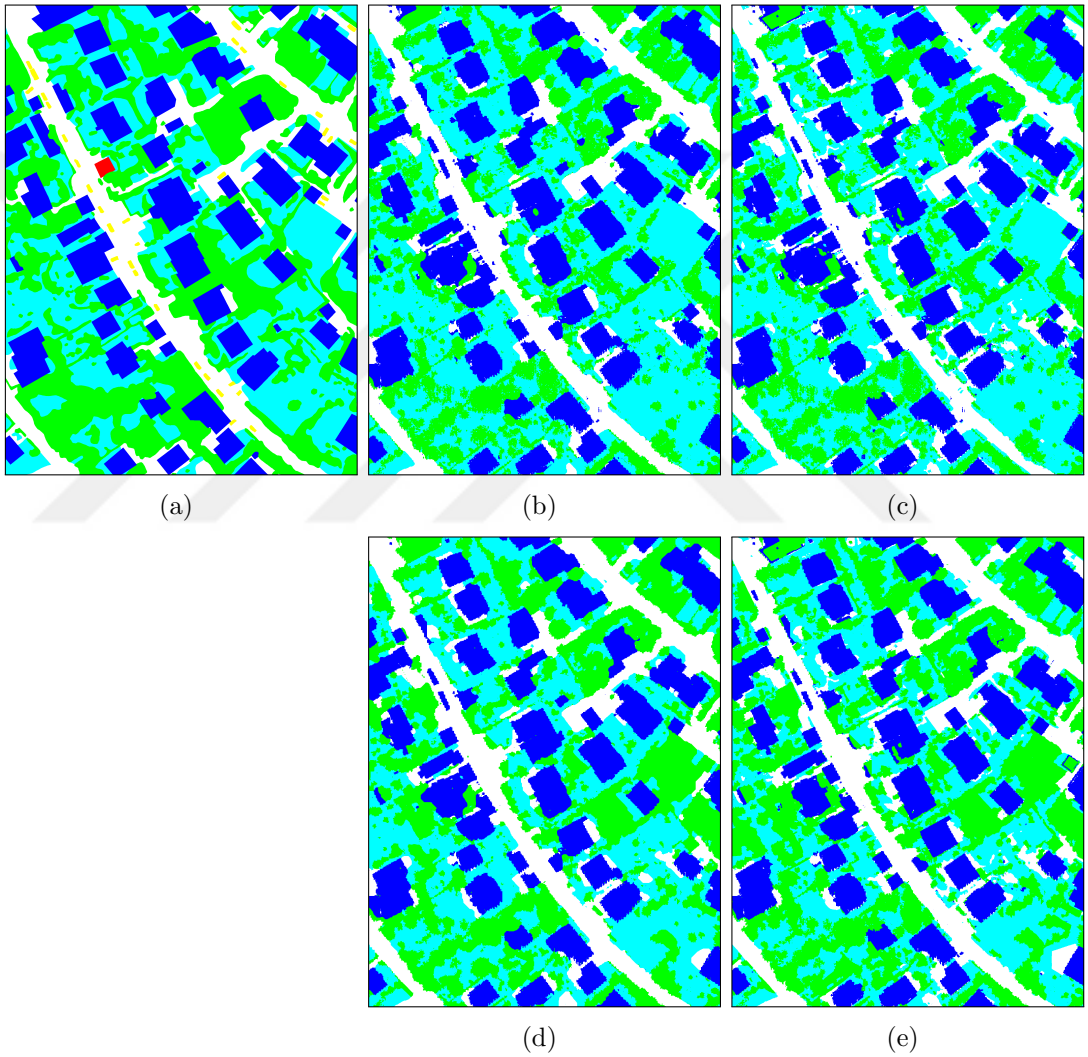


Figure 4.14: Results for group 1, area 15. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

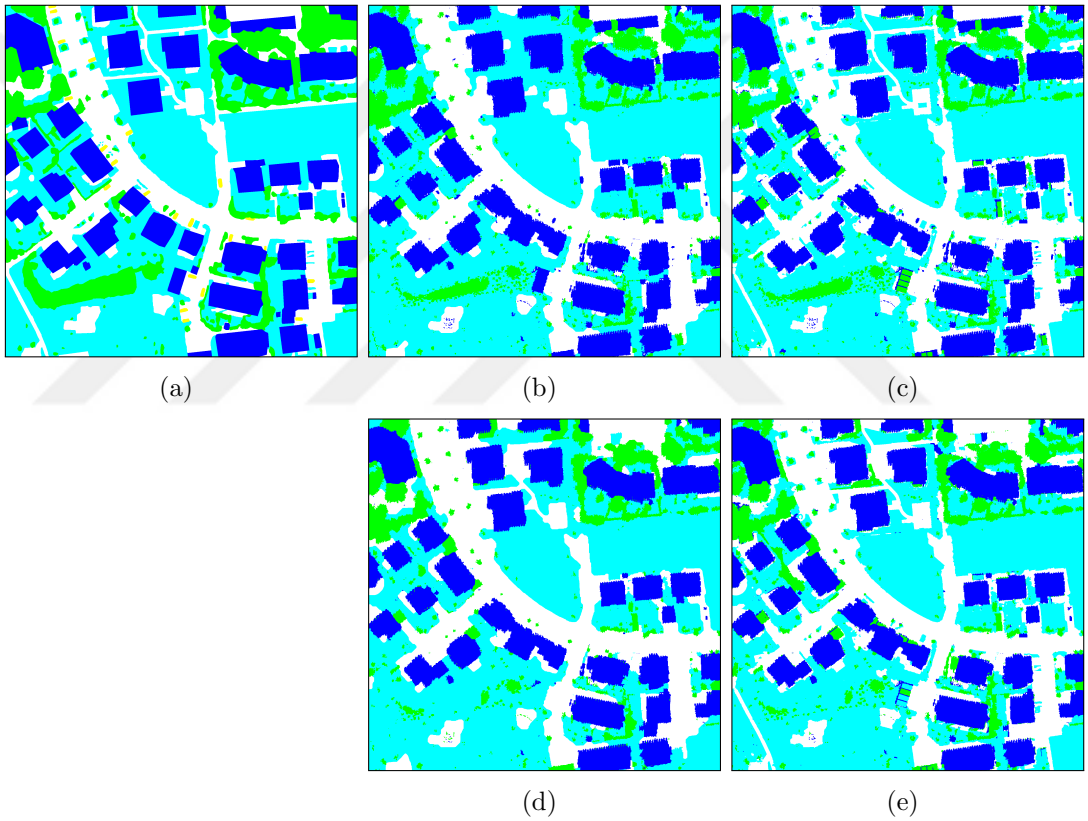


Figure 4.15: Results for group 1, area 37. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

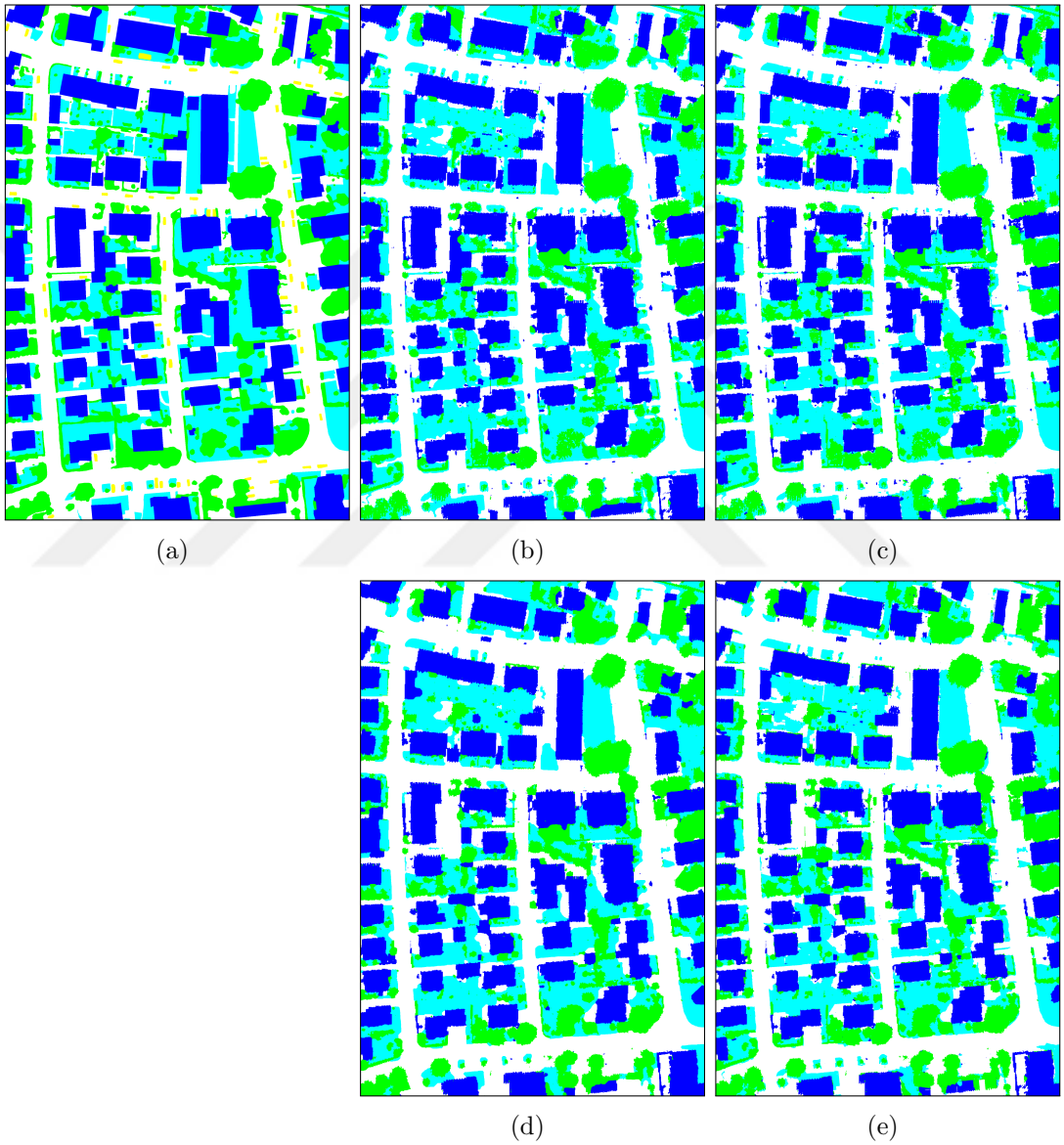


Figure 4.16: Results for group 2, area 3. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

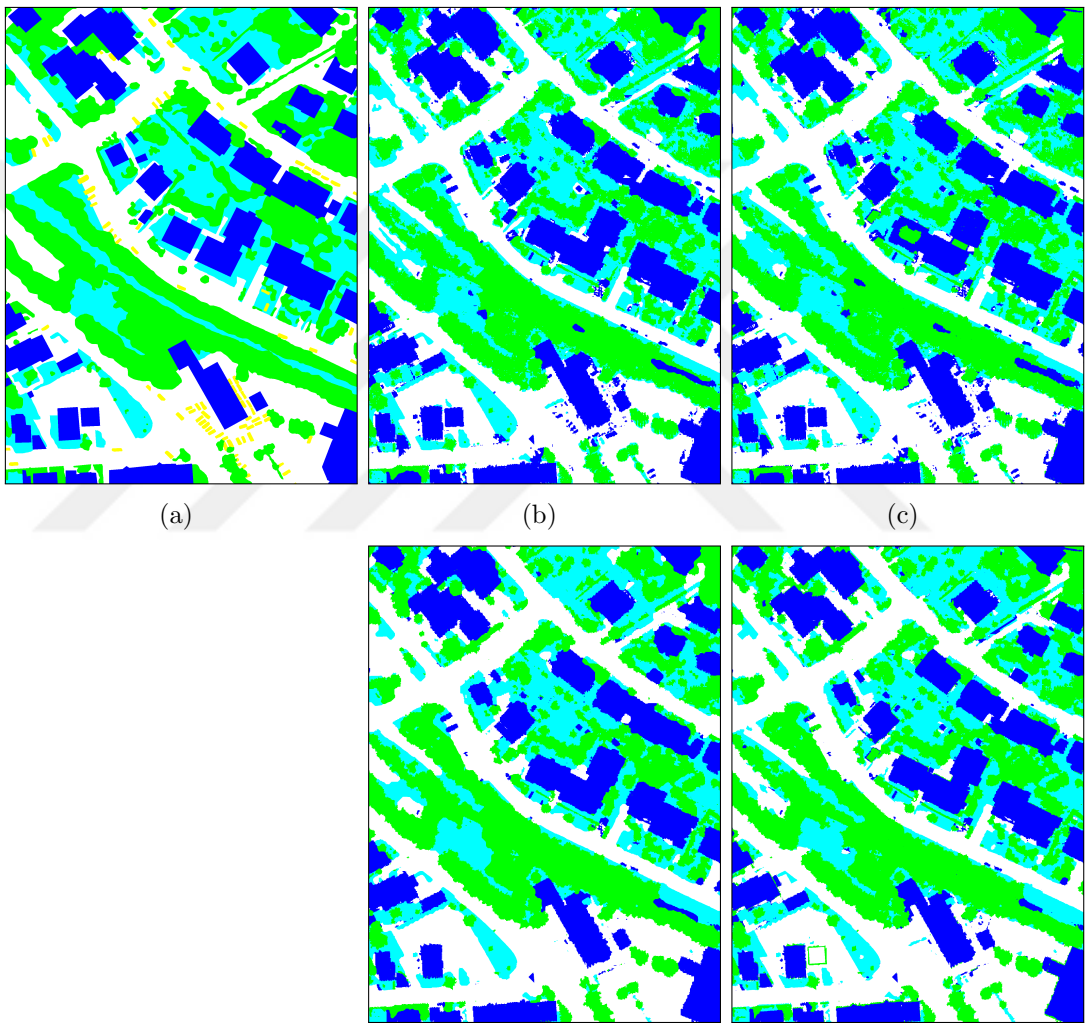


Figure 4.17: Results for group 2, area 11. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

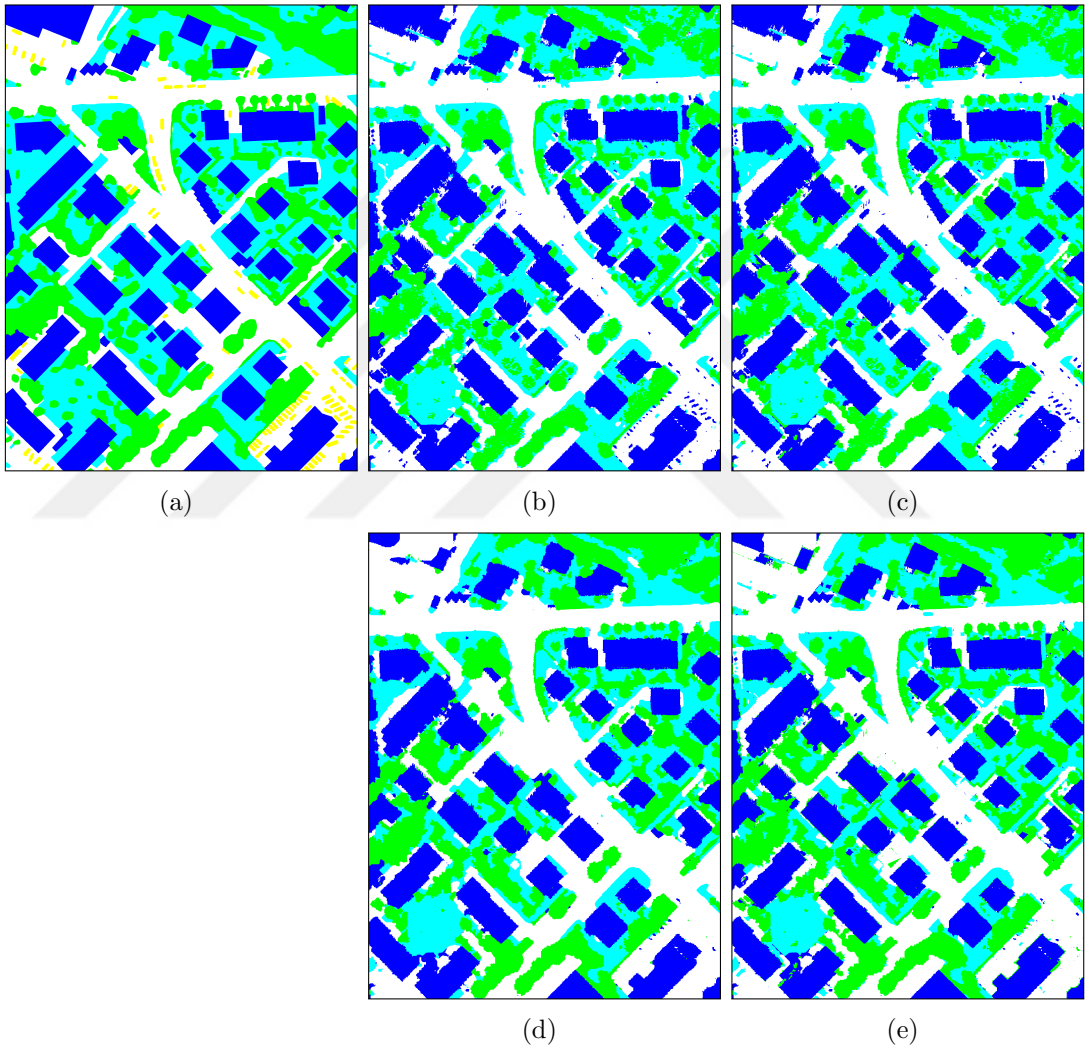


Figure 4.18: Results for group 2, area 30. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

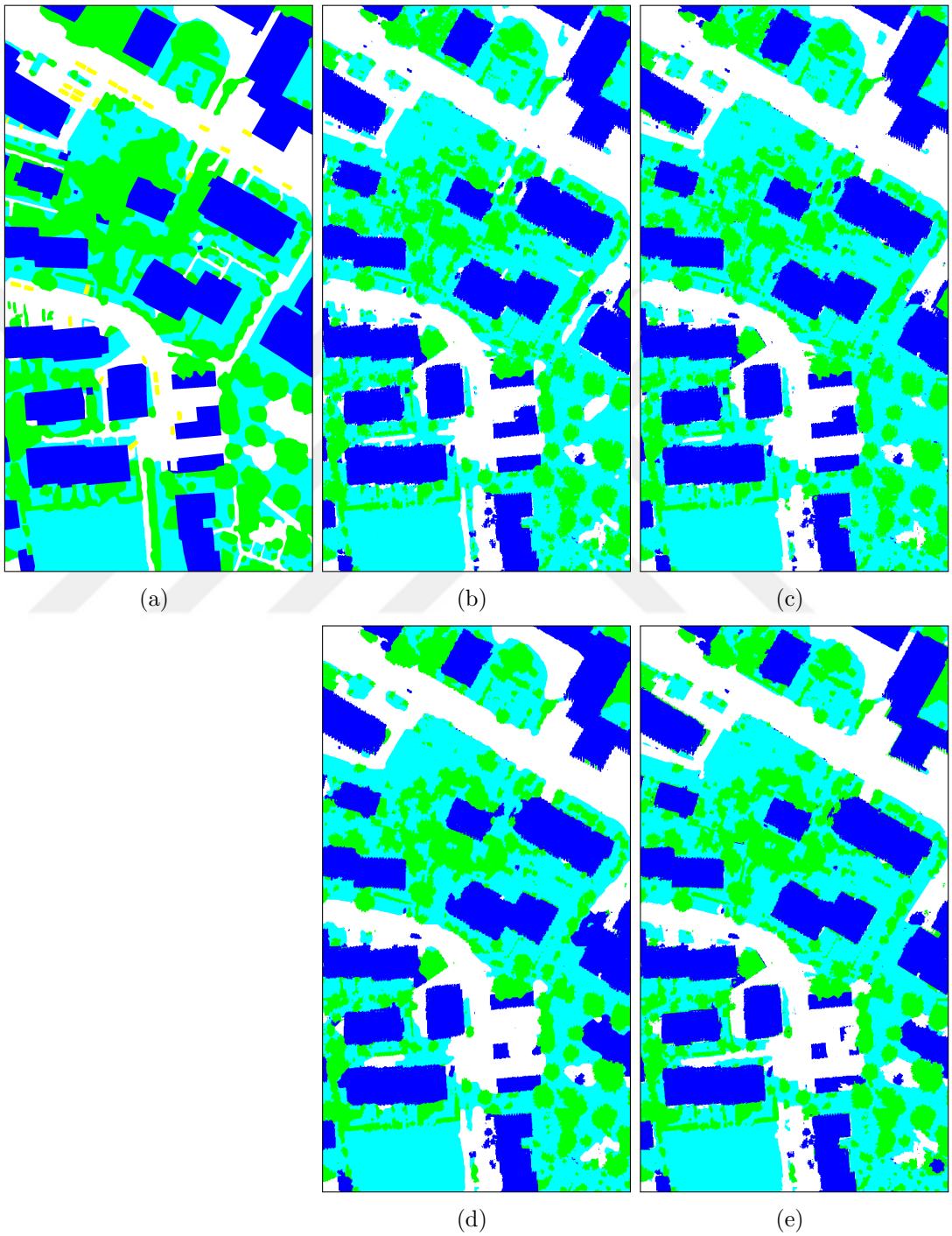


Figure 4.19: Results for group 2, area 34. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

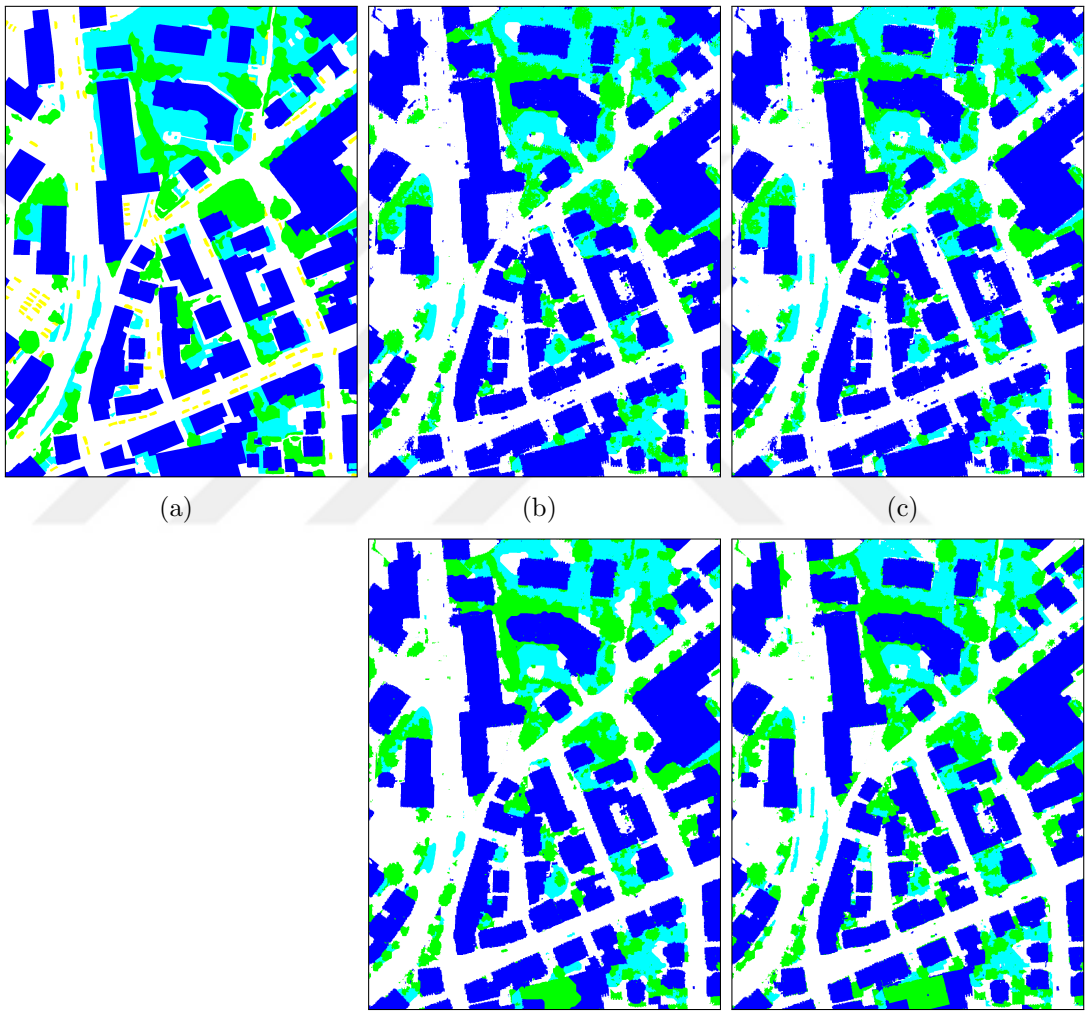


Figure 4.20: Results for group 3, area 1. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

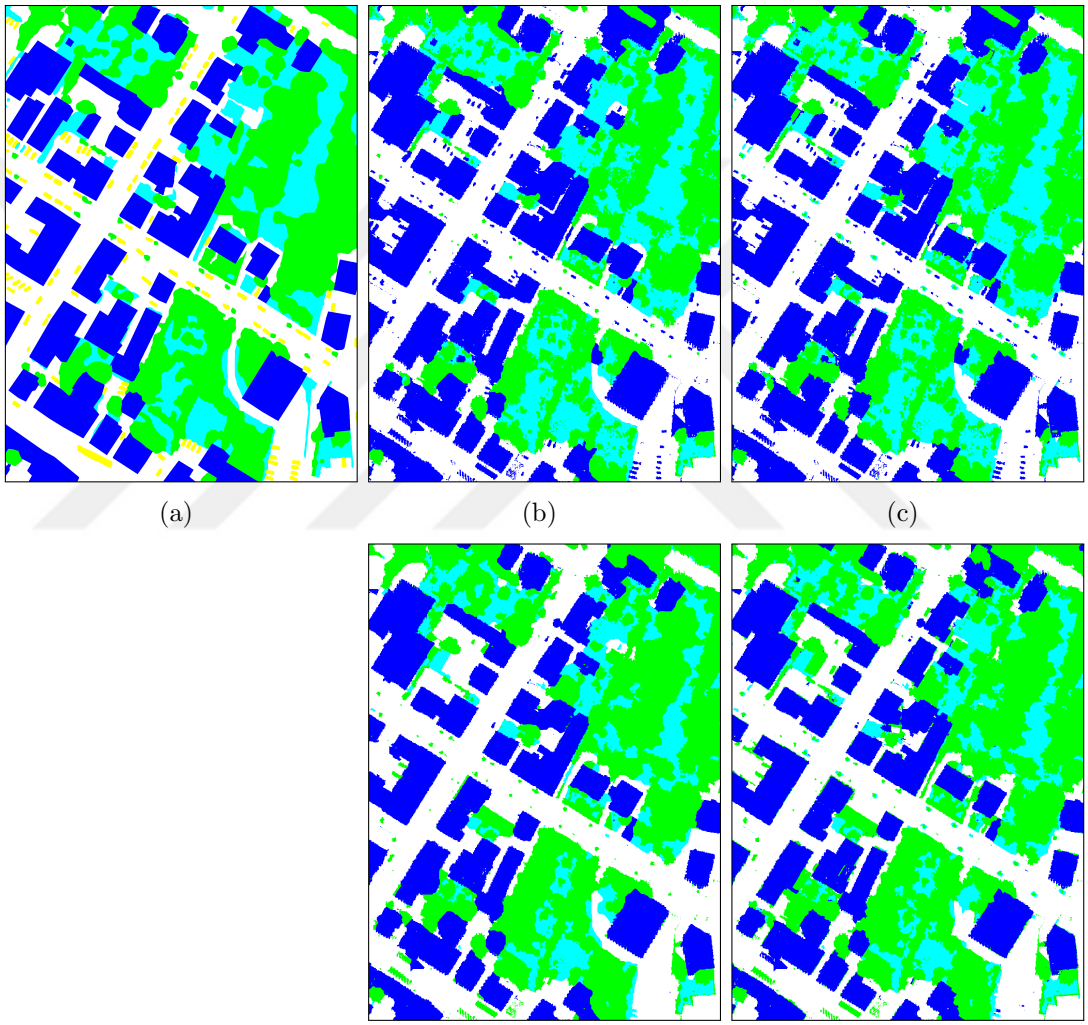


Figure 4.21: Results for group 3, area 7. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

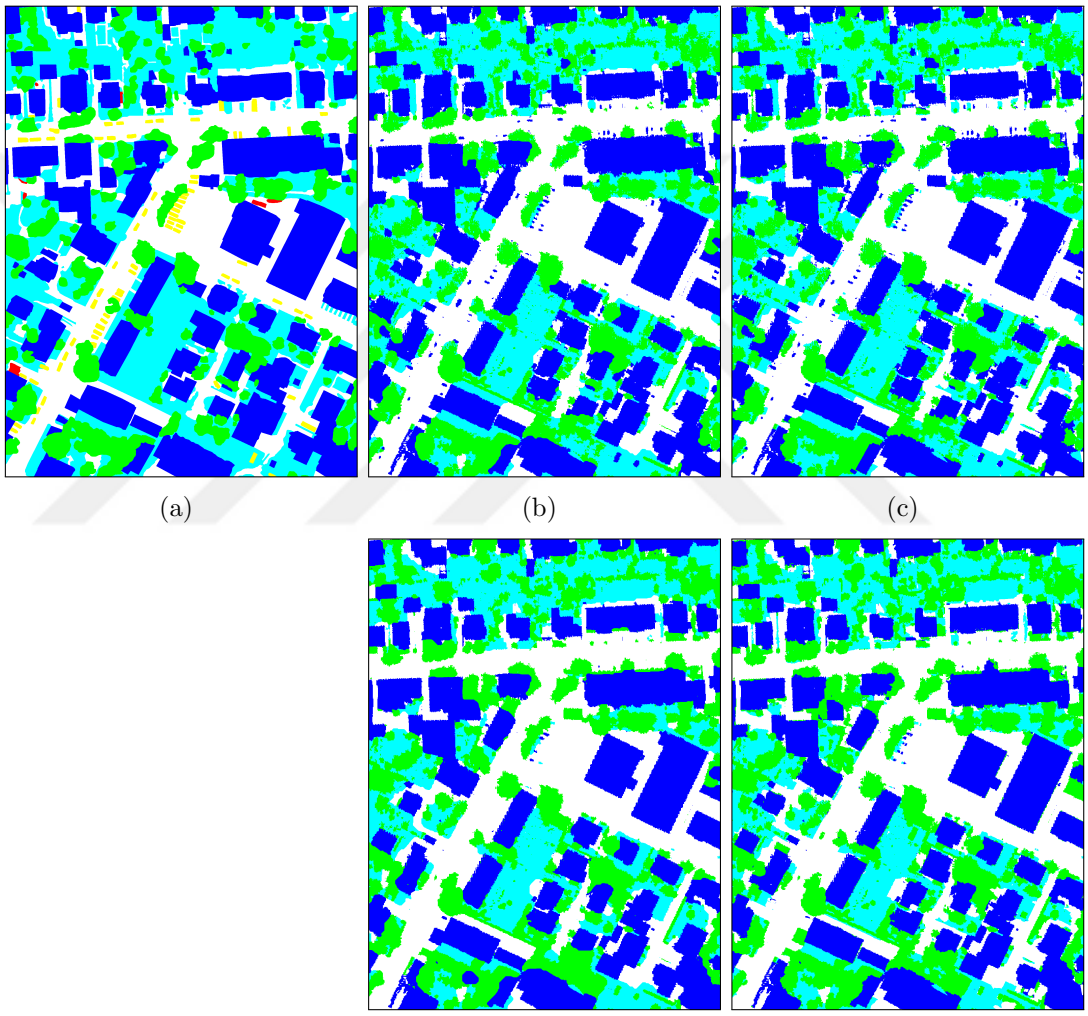


Figure 4.22: Results for group 3, area 28. (a) Ground-truth, (b) Max-flow/min-cut, constant smoothness cost, (c) Max-flow/min-cut, adaptive smoothness cost, (d) Multi-label optimization, constant smoothness cost, (e) Multi-label optimization, adaptive smoothness cost.

Table 4.6: Pixel-based overall F1-scores calculated from Tables A.7, A.8, A.9 and A.10. Method 1 and method 2 are max-flow/min-cut with constant and adaptive smoothness costs, and method 3 and method 4 are multi-label optimization with constant and adaptive smoothness costs.

	Method	Road	Building	Grass	Tree
Train	**method 1**	0.853 ± 0.007	0.915 ± 0.005	0.703 ± 0.012	0.777 ± 0.021
	method 2	0.855 ± 0.007	0.916 ± 0.003	0.702 ± 0.015	0.771 ± 0.021
	method 3	0.861 ± 0.006	0.917 ± 0.003	0.691 ± 0.015	0.768 ± 0.018
	method 4	0.859 ± 0.006	0.918 ± 0.001	0.680 ± 0.023	0.755 ± 0.024
Test	**method 1**	0.852 ± 0.012	0.919 ± 0.008	0.698 ± 0.035	0.769 ± 0.043
	method 2	0.855 ± 0.011	0.919 ± 0.007	0.697 ± 0.030	0.764 ± 0.044
	method 3	0.855 ± 0.013	0.915 ± 0.012	0.678 ± 0.026	0.753 ± 0.042
	method 4	0.856 ± 0.012	0.916 ± 0.011	0.670 ± 0.028	0.743 ± 0.040

Table 4.7: Object-based overall F1-scores calculated from A.11, A.12, A.13 and A.14. Method 1 and method 2 are max-flow/min-cut with constant and adaptive smoothness costs, and method 3 and method 4 are multi-label optimization with constant and adaptive smoothness costs.

	Method	Road	Building	Grass	Tree
Train	**method 1**	0.816 ± 0.098	0.988 ± 0.007	0.839 ± 0.007	0.910 ± 0.022
	method 2	0.787 ± 0.057	0.989 ± 0.006	0.820 ± 0.003	0.911 ± 0.008
	method 3	0.848 ± 0.062	0.990 ± 0.005	0.852 ± 0.007	0.900 ± 0.020
	method 4	0.765 ± 0.063	0.983 ± 0.010	0.841 ± 0.033	0.901 ± 0.016
Test	**method 1**	0.822 ± 0.101	0.991 ± 0.006	0.837 ± 0.014	0.912 ± 0.021
	method 2	0.797 ± 0.155	0.993 ± 0.003	0.822 ± 0.028	0.903 ± 0.045
	method 3	0.826 ± 0.144	0.993 ± 0.004	0.847 ± 0.014	0.896 ± 0.041
	method 4	0.776 ± 0.089	0.988 ± 0.010	0.853 ± 0.054	0.903 ± 0.046

Table 4.8: Pixel-based overall recall values calculated from pixel-based confusion matrices in Tables A.1, A.2 and A.3. Method 1 and method 2 are max-flow/min-cut with constant and varying smoothness costs, and method 3 and method 4 are multi-label optimization with constant and varying smoothness costs.

	Method	Road	Building	Grass	Tree
Train	**method 1**	0.832 ± 0.012	0.951 ± 0.003	0.742 ± 0.012	0.719 ± 0.027
	method 2	0.837 ± 0.009	0.949 ± 0.002	0.737 ± 0.016	0.718 ± 0.028
	method 3	0.870 ± 0.008	0.924 ± 0.011	0.678 ± 0.021	0.763 ± 0.017
	method 4	0.869 ± 0.004	0.919 ± 0.016	0.652 ± 0.028	0.772 ± 0.017
Test	**method 1**	0.831 ± 0.016	0.951 ± 0.003	0.735 ± 0.070	0.714 ± 0.082
	method 2	0.840 ± 0.018	0.948 ± 0.003	0.729 ± 0.058	0.713 ± 0.083
	method 3	0.862 ± 0.013	0.921 ± 0.013	0.665 ± 0.082	0.753 ± 0.090
	method 4	0.867 ± 0.008	0.913 ± 0.019	0.641 ± 0.077	0.768 ± 0.077

Table 4.9: Object-based overall recall values calculated from object-based confusion matrices in Tables A.4, A.5 and A.6. Method 1 and method 2 are max-flow/min-cut with constant and varying smoothness costs, and method 3 and method 4 are multi-label optimization with constant and varying smoothness costs.

	Method	Road	Building	Grass	Tree
Train	**method 1**	0.802 ± 0.107	0.988 ± 0.011	0.888 ± 0.024	0.877 ± 0.030
	method 2	0.776 ± 0.020	0.991 ± 0.009	0.858 ± 0.023	0.882 ± 0.005
	method 3	0.903 ± 0.039	0.982 ± 0.012	0.867 ± 0.022	0.886 ± 0.020
	method 4	0.880 ± 0.009	0.976 ± 0.018	0.803 ± 0.051	0.914 ± 0.009
Test	**method 1**	0.794 ± 0.103	0.990 ± 0.018	0.905 ± 0.027	0.876 ± 0.044
	method 2	0.796 ± 0.187	0.993 ± 0.012	0.882 ± 0.039	0.856 ± 0.089
	method 3	0.841 ± 0.159	0.990 ± 0.010	0.860 ± 0.015	0.884 ± 0.049
	method 4	0.874 ± 0.131	0.980 ± 0.019	0.823 ± 0.064	0.917 ± 0.052

Table 4.10: Overall accuracies calculated from pixel-based confusion matrices in Tables A.1, A.2 and A.3.

	Method 1	Method 2	Method 3	Method 4
Train	0.826 ± 0.005	0.826 ± 0.004	0.825 ± 0.005	0.820 ± 0.005
Test	0.823 ± 0.013	0.825 ± 0.012	0.819 ± 0.008	0.816 ± 0.004

Table 4.11: Overall accuracies calculated from object-based confusion matrices in Tables A.4, A.5 and A.6.

	Method 1	Method 2	Method 3	Method 4
Train	0.922 ± 0.012	0.915 ± 0.010	0.926 ± 0.008	0.912 ± 0.019
Test	0.926 ± 0.017	0.917 ± 0.025	0.924 ± 0.009	0.919 ± 0.025

4.5 Discussion

Pixel-based evaluation criteria are usually criticized because it is not possible to decide if an object is a true positive or not with this way, and it is possible in object-based evaluation metrics. However, we can criticize object-based metrics, as well. The main problem is that as described in Section 4.2, the objects with an area smaller than $50m^2$ area removed. This threshold directly effects the results. As we increase the threshold, we get better results but fewer objects are included in the confusion matrix.

Looking at the F1-scores and the accuracies from confusion matrices, we do not observe big differences among 4 methods; the results we get are almost identical. The height threshold, which is determined in the preprocessing step is very important for max-flow/min-cut methods because when some of the high pixels are classified as low, or vice versa, there is no chance for these pixels to be classified correctly. This is the biggest disadvantage of the binary classification setting used in our max-flow/min-cut frameworks. On the other hand, in multi-label optimization frameworks, we do not encounter the same problem because we do not set a height threshold; each component is classified directly. However, multi-label optimization performs multi-class classification, whereas max-flow/min-cut frameworks do a binary classification. This can be considered as a disadvantage of multi-label optimization over max-flow/min-cut frameworks because multi-class classification is a harder task than binary classification.

The Vaihingen data set, which we use in this thesis, is a benchmark contest dataset provided by the International Society for Photogrammetry and Remote Sensing (ISPRS). The whole Vaihingen region has been divided into 33 areas. Among these 33 areas, 16 of them are annotated and the remaining 17 areas do not have publicly available ground-truth. The participants for this contest are expected to train their models from the annotated 16 areas, and submit the classification maps for the other 17 areas, which are generated by their models to the ISPRS. The ISPRS computes confusion matrices from all 17 areas and add them up. Then, pixel-based F1-score for each class is calculated from the overall

confusion matrix. Another important point is that the ISPRS emphasizes that the ground-truth for 17 areas are eroded by a disc shaped structuring element with a radius of 3. The eroded areas are ignored during the evaluation. We do extensive experiments using the 16 areas having ground-truth. Moreover, we do not erode the ground-truth; we use them directly. Therefore, we do not know exactly how our frameworks are going to perform for the areas whose ground-truth are not provided. Hence, comparing our method with the ones, which are tested on the 17 areas would not be fair. However, we think that comparing our methodology with the other methods can give a rough idea about where we stand. The results for the contest can be found in Table 4.13. We take average and calculate standard deviation of the F1-scores for the methods that belong to the same participants. Abbreviations for some of the methods in Table 4.13 can be seen in Table 4.12. The ones that are not listed in Table 4.12 are not described in a published paper or a well organized technical report.

Table 4.12: Abbreviations of the methods whose results are submitted to the ISPRS, and their corresponding references.

Abbreviation	**Reference**
SVL	[62]
ADL	[63]
ONE	[37], [66]
DST	[65]
DLR	[67], [68]
VOA	[74]
INR	[69]
RIT	[70]
ETH	[71]
UCal	[73]
UPB	[72]

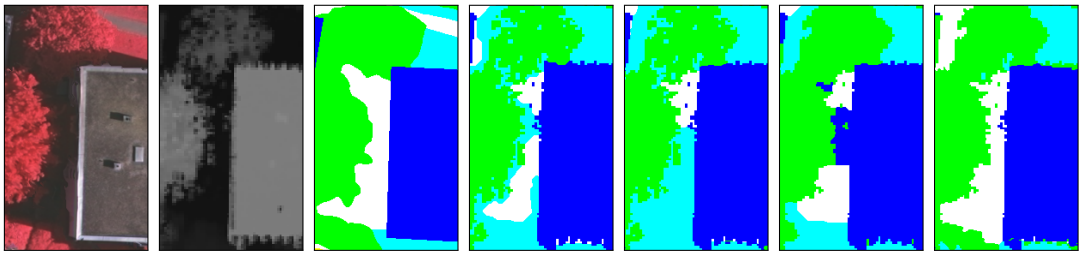
Except for learning weight of the edges in the graphs, and determining thresholds for height and number of iterations in the preprocessing step, our frameworks are completely unsupervised. Although we use only small amount of supervision, from the overall pixel-based F1-scores, which are shown in Table 4.6 and the results belonging to the other methods shown in Table 4.13, we conclude that our results for road and building classifications reach state of the art.

Table 4.13: ISPRS F1-score results table.

	Road	**Building**	**Grass**	**Tree**	**Car**	**Overall**
SVL	85.5 ± 1.7	89.3 ± 3.2	75.9 ± 2.4	83.8 ± 1.5	52.6 ± 6.0	83.5 ± 2.1
ADL	88.9 ± 0.7	92.7 ± 0.6	80.8 ± 1.7	87.5 ± 0.9	60.6 ± 2.4	87.1 ± 1.0
UT_Mev	84.3	88.7	74.5	82.0	9.9	81.8
HUST	86.9	92.0	78.3	86.9	29.0	85.9
ONE	87.7 ± 2.9	90.6 ± 3.5	79.5 ± 3.1	87.1 ± 1.8	59.4 ± 16.2	85.9 ± 3.0
DST	90.4 ± 0.1	93.6 ± 0.1	83.0 ± 0.6	89.0 ± 0.3	73.2 ± 0.9	88.9 ± 0.3
UZ	89.2	92.5	81.6	86.9	57.3	87.3
DLR	90.9 ± 1.0	93.6 ± 1.0	82.7 ± 1.2	89.3 ± 0.6	78.7 ± 2.5	89.0 ± 0.7
UOA	89.8	92.1	80.4	88.2	82.0	87.6
IVFL	88.2	92.4	79.8	86.7	50.7	86.5
INR	91.1	94.7	83.4	89.3	71.2	89.5
RIT	89.3 ± 1.6	93.5 ± 1.2	81.1 ± 1.6	88.2 ± 0.8	63.2 ± 16.4	87.7 ± 1.4
ETH_C	87.2	92.0	77.5	87.1	54.5	85.9
Ano	90.2 ± 0.2	93.1 ± 0.1	81.6 ± 0.2	88.6 ± 0.0	74.1 ± 0.6	88.3 ± 0.1
UCal	91.2 ± 1.7	93.7 ± 1.3	81.3 ± 3.2	88.5 ± 1.6	79.0 ± 14.0	88.9 ± 1.9
CASIA	93.0 ± 0.4	95.7 ± 0.5	84.5 ± 0.3	89.8 ± 0.2	83.8 ± 4.2	90.8 ± 0.4
UPB	87.5	89.3	77.3	85.8	77.1	85.1
NLPR	90.4 ± 3.0	93.8 ± 2.1	82.5 ± 3.2	88.8 ± 1.7	67.7 ± 18.9	88.8 ± 2.7
GSN	92.5 ± 0.1	94.0 ± 0.1	83.3 ± 0.5	89.6 ± 0.0	84.4 ± 1.2	89.9 ± 0.1

Since we use the height information from DSMs in addition to the spatial and spectral information from the aerial data, our frameworks are able to detect the objects which are located under shadows. This is the most powerful point of our methodology. Figure 4.23 provides a zoomed example. As it can be seen from the figure that the building casts a shadow onto the tree, which is located at top-left of the building. Our frameworks can detect big part of this tree successfully.

However, from Table 4.6 we also see that our results are slightly lower than state of the art for grass and tree classification. We believe that the problem stems from the LiDAR data because the LiDAR data are not as dense as very high resolution aerial data. The problem is illustrated in Figure 4.24. From the figure, we see that there are lots of holes among the pixels belonging to trees in the DSM data. These holes are very common among tree pixels in the LiDAR data. Almost every tree has this kind of holes. When we threshold the data according to the height, the pixels that are located at these holes are classified as low pixels. For this reason, there is no way for this kind of pixels to be classified as a tree



(a) (b) (c) (d) (e) (f) (g)

Figure 4.23: Zoomed example 1. (a) Aerial data, (b) DSM, (c) Ground-truth, (d) Max-flow/min-cut with constant smoothness output, (e) Max-flow/min-cut with adaptive smoothness output, (f) Multi-label optimization with constant smoothness output, (g) Multi-label optimization with adaptive smoothness output.

when the max-flow/min-cut frameworks are used. Because their NDVI value is high, they are usually classified as grass. Figure 4.24, and both pixel-based and object-based confusion matrices show that a lot of pixels belonging to tree class are classified as grass, and lots of pixels belonging to grass are classified as tree. If the LiDAR data was denser and these holes did not exist, we would have got better results for tree and grass classification. A more detailed processing of the original LiDAR point clouds instead of using the provided DSM directly may also improve the results. From Figure 4.24, we observe that the multi-label optimization frameworks can detect these problematic pixels correctly in some cases because we do not threshold the data in this framework. In addition, the cut procedure forces the neighboring pixels to have the same label even if these pixels have very low height value. For this reason, very small holes are classified as tree. However, as described in the second paragraph multi-label optimization frameworks also have their own disadvantages.

Another reason why we get slightly lower results for grass class is that we use only NDVI, which is vulnerable to shadows, as the feature. In the middle of the zoomed example illustrated in Figure 4.25, there is a big black area, which is marked as grass in the ground-truth. Since this area is very dark and its height is very low, this area is classified as road. Moreover, this zoomed example is a part of area 5, and from the pixel-based F1-score tables we see that F1 scores of grass class for area 5 are always the lowest compared with F1-scores of the same class

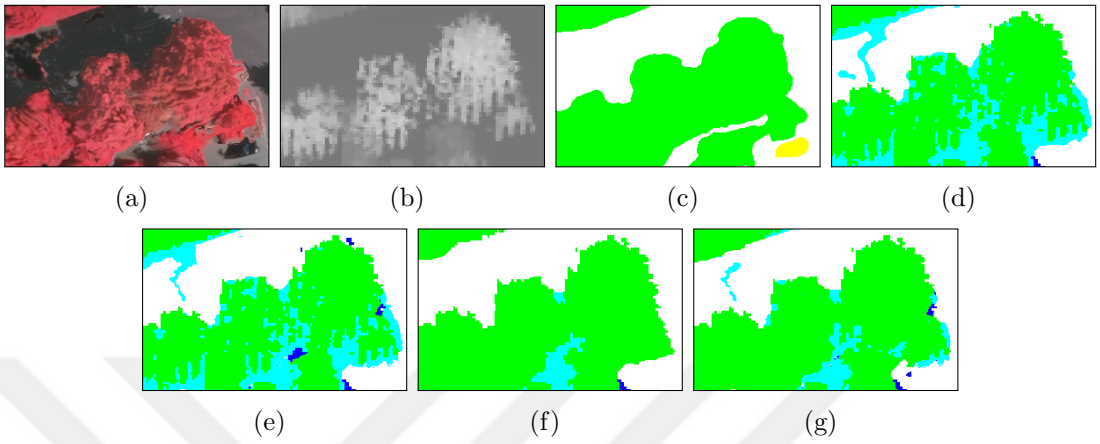


Figure 4.24: Zoomed example 2. (a) Aerial data, (b) DSM, (c) Ground-truth, (d) Max-flow/min-cut with constant smoothness output, (e) Max-flow/min-cut with adaptive smoothness output, (f) Multi-label optimization with constant smoothness output, (g) Multi-label optimization with adaptive smoothness output.

for other areas. We believe that this big dark area causes this problem. From Figure 4.3, we see that grass areas constitute only small part of area 5. Among the pixels belonging to grass, big dark area is misclassified. For this reason, a low F1-score is obtained.

Although we reach state of the art for building classification task, irregularity of the points, which are located at the borders of some buildings in the LiDAR data reduces detection rate a little. An example building whose borders are irregular in the LiDAR data is shown in Figure 4.26. Eroding the ground truth as described earlier will ignore these problems in the evaluation.

We also think that some part of the ground-truth does not represent the data very well. Two zoomed examples about this problem are provided in Figures 4.27 and 4.28. Looking at the aerial image and the DSM, we believe that some part of the pixels, which are marked as tree in the ground-truth seem more like grass than tree. As it can be seen from the same image, our frameworks classify some of such pixels as grass.

Another important point is that as we describe in Section 3, the cut frameworks can not detect cars because location of the cars in the LiDAR and the aerial

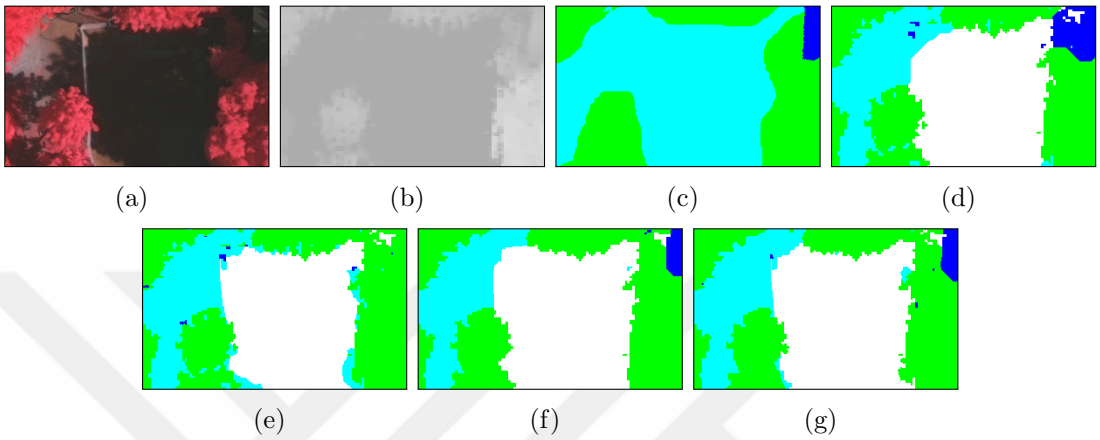


Figure 4.25: Zoomed example 3. (a) Aerial data, (b) DSM, (c) Ground-truth, (d) Max-flow/min-cut with constant smoothness output, (e) Max-flow/min-cut with adaptive smoothness output, (f) Multi-label optimization with constant smoothness output, (g) Multi-label optimization with adaptive smoothness output.

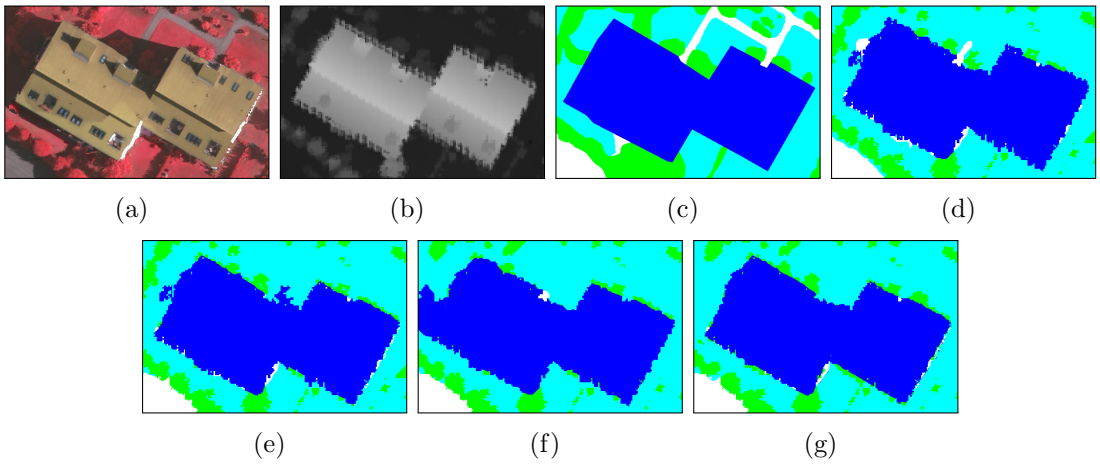


Figure 4.26: Zoomed example 4. (a) Aerial data, (b) DSM, (c) Ground-truth, (d) Max-flow/min-cut with constant smoothness output, (e) Max-flow/min-cut with adaptive smoothness output, (f) Multi-label optimization with constant smoothness output, (g) Multi-label optimization with adaptive smoothness output.

data do not match. Examples for this case are shown in Figure 4.29 and 4.30. Locations of cars in aerial data and ground-truth match. When we generate confusion matrices we ignore the pixels that are marked as car in the ground-truth. However, we do not ignore the pixels, which seem like a car in the LiDAR data. They are mostly classified as building. For this reason, they are counted as wrongly classified pixels and reduce the performance of the frameworks slightly.

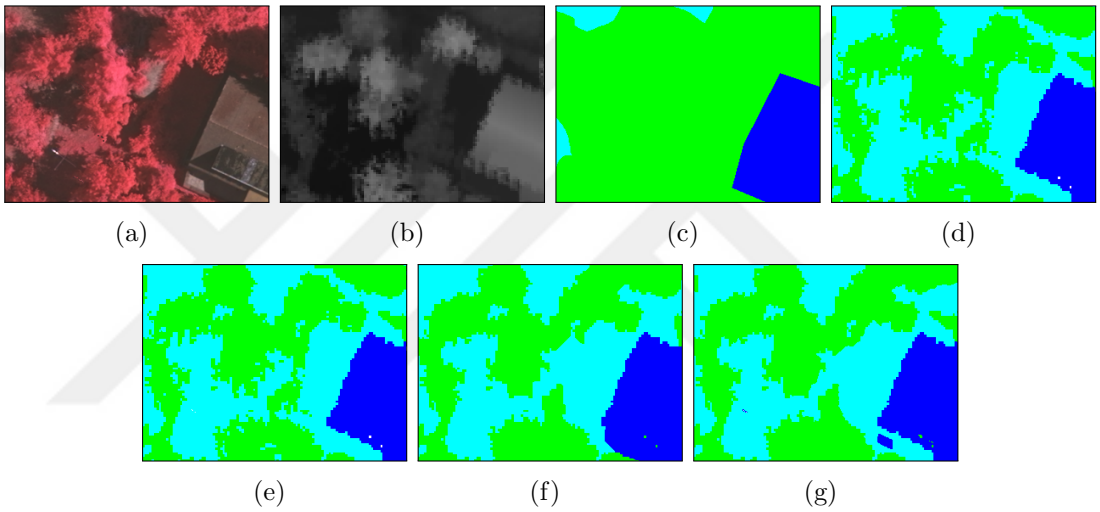


Figure 4.27: Zoomed example 5. (a) Aerial data, (b) DSM, (c) Ground-truth, (d) Max-flow/min-cut with constant smoothness output, (e) Max-flow/min-cut with adaptive smoothness output, (f) Multi-label optimization with constant smoothness output, (g) Multi-label optimization with adaptive smoothness output.

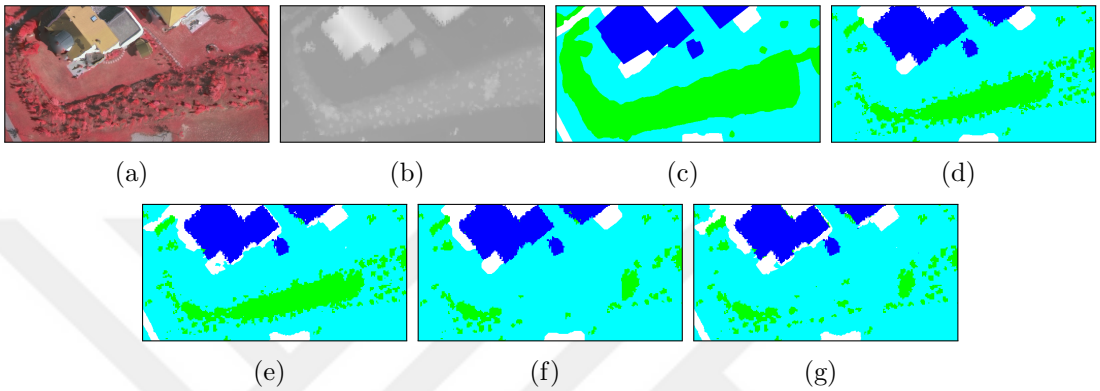


Figure 4.28: Zoomed example 6. (a) Aerial data, (b) DSM, (c) Ground-truth, (d) Max-flow/min-cut with constant smoothness output, (e) Max-flow/min-cut with adaptive smoothness output, (f) Multi-label optimization with constant smoothness output, (g) Multi-label optimization with adaptive smoothness output.

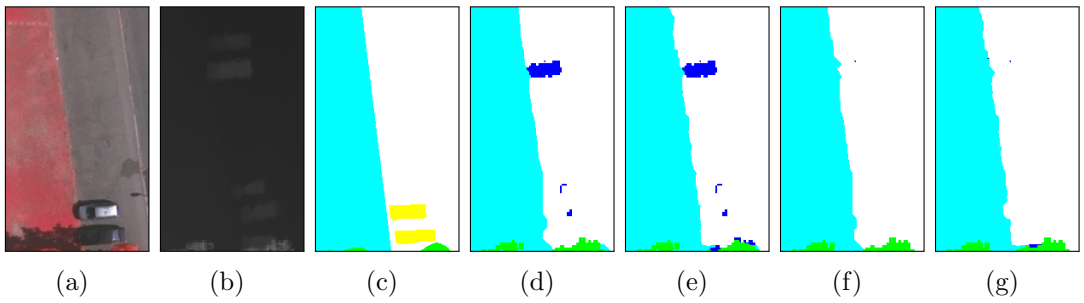


Figure 4.29: Zoomed example 7. (a) Aerial data, (b) DSM, (c) Ground-truth, (d) Max-flow/min-cut with constant smoothness output, (e) Max-flow/min-cut with adaptive smoothness output, (f) Multi-label optimization with constant smoothness output, (g) Multi-label optimization with adaptive smoothness output.

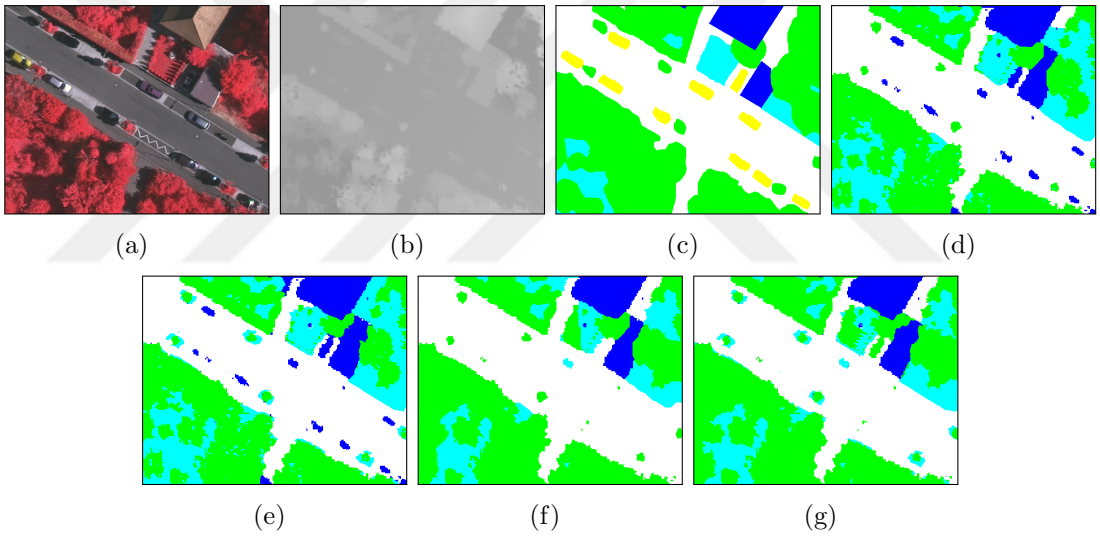


Figure 4.30: Zoomed example 8. (a) Aerial data, (b) DSM, (c) Ground-truth, (d) Max-flow/min-cut with constant smoothness output, (e) Max-flow/min-cut with adaptive smoothness output, (f) Multi-label optimization with constant smoothness output, (g) Multi-label optimization with adaptive smoothness output.

Chapter 5

Conclusion

In this thesis, we proposed a hierarchical methodology that detects roads, buildings, grass areas and trees in remote sensing data. In the preprocessing step of the methodology, we estimated bare earth from the given raw elevation data and subtracted it from the elevation data in order to generate another data, which store relative height information of each pixel above the estimated ground called the normalized digital surface model (NDSM). We also thresholded NDSM to separate the data into two groups as high and low. We then found connected components in both high and low pixels. Since we used very high resolution data, processing the data directly was not suitable. Dividing the data into smaller components enabled us to process the data more efficiently.

For the classification problem, we proposed two graph-cut based frameworks that use spectral information from very high resolution aerial images, height information from NDSM, and spatial information from the graph, which is built for the cut frameworks. The first framework performed a binary classification; it detected buildings and trees in each component for the high pixels, and separated roads from grass areas in components of the low pixels. The second framework detected objects in each components directly by doing a multi-class classification. We also used two different smoothness costs for these two frameworks, where we assigned a constant cost of separating two neighboring pixels from each other

in the former, and we gave a weight, which is inversely proportional to the color difference between pixels, to the edge between two neighboring pixels in the latter case. In our frameworks, calculating the edge weights in the graphs was the only step where supervision was used.

We did experiments on a benchmark data set, which is provided by the International Society for Photogrammetry and Remote Sensing (ISPRS). In our experiments we proved that our frameworks detected some of objects successfully even if they are located under shadow where methods that use only one data source often fail. We also showed that although we used only small amount of supervision in our methodology, our results for building and tree detection problem are compatible with the ones, which are considered as state of the art. Finally, we showed that our results for road and grass detection are slightly lower than state of the art due to the sparsity of the LiDAR data and the problem that stems from location difference of cars in the aerial and the LiDAR sets.

In the future work, more features could be extracted to increase the performance of our frameworks. In addition, our methodology is not able to detect cars because of the mismatch of locations of cars in the LiDAR and the aerial data. In the processing step, a registration operation between the LiDAR and the aerial data can be performed so that locations of cars in both data could match. Besides, holes among the pixels belonging to trees in the LiDAR data reduced the accuracy slightly. In the preprocessing step, these holes can be tried to be eliminated.

Bibliography

[1] H. Arefi and M. Hahn, "A morphological reconstruction algorithm for separating off-terrain points from terrain points in laser scanning data," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 3/W19, pp. 120–125, 2005.

[2] K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, and C. Zhang, "A progressive morphological filter for removing nonground measurements from airborne lidar data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 4, pp. 872–882, 2003.

[3] D. C. Duro, S. E. Franklin, and M. G. Dubé, "A comparison of pixel-based and object-based image analysis with selected machine learning algorithms for the classification of agricultural landscapes using spot-5 hrg imagery," *Remote Sensing of Environment*, vol. 118, pp. 259–272, 2012.

[4] S. W. Myint, P. Gober, A. Brazel, S. Grossman-Clarke, and Q. Weng, "Per-pixel vs. object-based classification of urban land cover extraction using high spatial resolution imagery," *Remote Sensing of Environment*, vol. 115, no. 5, pp. 1145–1161, 2011.

[5] T. G. Whiteside, G. S. Boggs, and S. W. Maier, "Comparing object-based and pixel-based classifications for mapping savannas," *International Journal of Applied Earth Observation and Geoinformation*, vol. 13, no. 6, pp. 884–893, 2011.

[6] S. Bhaskaran, S. Paramananda, and M. Ramnarayan, "Per-pixel and object-oriented classification methods for mapping urban features using ikonos satellite data," *Applied Geography*, vol. 30, no. 4, pp. 650–665, 2010.

[7] R. C. Weih and N. D. Riggan, "Object-based classification vs. pixel-based classification: comparative importance of multi-resolution imagery," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 4, p. C7, 2010.

[8] T. Liu, Y. Gu, X. Jia, J. A. Benediktsson, and J. Chanussot, "Class-specific sparse multiple kernel learning for spectral–spatial hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 12, pp. 7351–7365, 2016.

[9] Y. Chen, X. Zhao, and X. Jia, "Spectral–spatial classification of hyperspectral data based on deep belief network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2381–2392, 2015.

[10] R. Hang, Q. Liu, H. Song, and Y. Sun, "Matrix-based discriminant subspace ensemble for hyperspectral image spatial–spectral feature fusion," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 2, pp. 783–794, 2016.

[11] Z. Liu, B. Tang, X. He, Q. Qiu, and F. Liu, "Class-specific random forest with cross-correlation constraints for spectral-spatial hyperspectral image classification," *IEEE Geoscience and Remote Sensing Letters*, 2017.

[12] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[13] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient nd image segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, 2006.

[14] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[15] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004.

[16] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," *International Journal of Computer Vision*, vol. 96, no. 1, pp. 1–27, 2012.

[17] O. Taşar and S. Aksoy, "Object detection using optical and lidar data fusion," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 7204–7207, IEEE, 2016.

[18] M. Awrangjeb, C. Zhang, and C. S. Fraser, "Building detection in complex scenes thorough effective separation of buildings from trees," *Photogrammetric Engineering & Remote Sensing*, vol. 78, no. 7, pp. 729–745, 2012.

[19] D. Chai, "A probabilistic framework for building extraction from airborne color image and dsm," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2016.

[20] S. Gilani, M. Awrangjeb, and G. Lu, "Fusion of lidar data and multispectral imagery for effective building detection based on graph and connected component analysis," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 3, p. 65, 2015.

[21] A. Zarea and A. Mohammadzadeh, "A novel building and tree detection method from lidar data and aerial images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 5, pp. 1864–1875, 2016.

[22] Q. Zhang, X. Huang, and G. Zhang, "A morphological building detection framework for high-resolution optical imagery over urban areas," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 9, pp. 1388–1392, 2016.

[23] A. Moussa and N. El-Sheimy, "A new object based method for automated extraction of urban objects from airborne sensors data," in *Proceedings of: XXII ISPRS Congress, Melbourne, Australia*, 2012.

[24] D. Grigillo and U. Kanjir, "Urban object extraction from digital surface model and digital aerial images," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, no. 3, pp. 215–220, 2012.

[25] W. Zhou, "An object-based approach for urban land cover classification: integrating lidar height and intensity data," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 4, pp. 928–931, 2013.

[26] S. Talebi, A. Zarea, S. Sadeghian, and H. Arefi, "Detection of tree crowns based on reclassification using aerial images and lidar data," *ISPRS Archives*, vol. 40, pp. 415–420, 2013.

[27] A. Mehta, O. Dikshit, and K. Venkataramani, "Integration of high-resolution imagery and lidar data for object-based classification of urban area," *Geocarto International*, vol. 29, no. 4, pp. 418–432, 2014.

[28] C. Berger, M. Voltersen, S. Hese, I. Walde, and C. Schmullius, "Robust extraction of urban land cover information from hsr multi-spectral and lidar data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 6, pp. 1–16, 2013.

[29] Y. Kim and Y. Kim, "Improved classification accuracy based on the output-level fusion of high-resolution satellite images and airborne lidar data in urban area," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 3, pp. 636–640, 2014.

[30] H. Li, L. Jing, Z. Sun, J. Li, R. Xu, Y. Tang, and F. Chen, "A novel image-fusion method based on the un-mixing of mixed ms sub-pixels regarding high-resolution dsm," *International Journal of Digital Earth*, vol. 9, no. 6, pp. 606–628, 2016.

[31] H. Guan, Z. Ji, L. Zhong, J. Li, and Q. Ren, "Partially supervised hierarchical classification for urban features from lidar data with aerial imagery," *International Journal of Remote Sensing*, vol. 34, no. 1, pp. 190–210, 2013.

[32] P. Ghamisi, B. Höfle, and X. X. Zhu, "Hyperspectral and lidar data fusion using extinction profiles and deep convolutional neural network," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2016.

[33] Y. Chen, C. Li, P. Ghamisi, C. Shi, and Y. Gu, "Deep fusion of hyperspectral and lidar data for thematic classification," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3591–3594, IEEE, 2016.

[34] S. Morchhale, V. P. Pauca, R. J. Plemmons, and T. C. Torgersen, "Classification of pixel-level fused hyperspectral and lidar data using deep convolutional neural networks," 2016.

[35] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, 2015.

[36] M. Campos-Taberner, A. Romero, C. Gatta, and G. Camps-Valls, "Shared feature representations of lidar and optical images: Trading sparsity for semantic discrimination," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 4169–4172, IEEE, 2015.

[37] A. Lagrange, B. Le Saux, A. Beaupere, A. Boulch, A. Chan-Hon-Tong, S. Herbin, H. Randrianarivo, and M. Ferecatu, "Benchmarking classification of earth-observation data: From learning explicit features to convolutional networks," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 4173–4176, IEEE, 2015.

[38] W. Liao, R. Bellens, A. Pižurica, S. Gautama, and W. Philips, "Combining feature fusion and decision fusion for classification of hyperspectral and lidar data," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 1241–1244, IEEE, 2014.

[39] W. Liao, A. Pižurica, R. Bellens, S. Gautama, and W. Philips, "Generalized graph-based fusion of hyperspectral and lidar data using morphological features," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 3, pp. 552–556, 2015.

[40] W. Liao, M. Dalla Mura, J. Chanussot, and A. Pižurica, "Fusion of spectral and spatial information for classification of hyperspectral remote-sensed imagery by local graph," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 2, pp. 583–594, 2016.

[41] M. Khodadadzadeh, J. Li, S. Prasad, and A. Plaza, "Fusion of hyperspectral and lidar remote sensing data using multiple feature learning," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 6, pp. 2971–2983, 2015.

[42] A. Merentitis and C. Debes, "Automatic fusion and classification using random forests and features extracted with deep learning," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 2943–2946, IEEE, 2015.

[43] A. Merentitis, C. Debes, R. Heremans, and N. Frangiadakis, "Automatic fusion and classification of hyperspectral and lidar data using random forests," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 1245–1248, IEEE, 2014.

[44] H. Guan, J. Li, M. Chapman, F. Deng, Z. Ji, and X. Yang, "Integration of orthoimagery and lidar data for object-based urban thematic mapping using random forests," *International Journal of Remote Sensing*, vol. 34, no. 14, pp. 5166–5186, 2013.

[45] S. Luo, C. Wang, X. Xi, H. Zeng, D. Li, S. Xia, and P. Wang, "Fusion of airborne discrete-return lidar and hyperspectral data for land cover classification," *Remote Sensing*, vol. 8, no. 1, p. 3, 2015.

[46] B. Bigdeli, F. Samadzadegan, and P. Reinartz, "Fusion of hyperspectral and lidar data using decision template-based fuzzy multiple classifier system," *International Journal of Applied Earth Observation and Geoinformation*, vol. 38, pp. 309–320, 2015.

[47] P. Ghamisi, J. A. Benediktsson, and S. Phinn, "Fusion of hyperspectral and lidar data in classification of urban areas," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 181–184, IEEE, 2014.

[48] W. Liao, J. Xia, P. Du, and W. Philips, "Semi-supervised graph fusion of hyperspectral and lidar data for classification," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 53–56, IEEE, 2015.

[49] X. Huang, L. Zhang, and W. Gong, "Information fusion of aerial images and lidar data in urban areas: vector-stacking, re-classification and post-processing approaches," *International Journal of Remote Sensing*, vol. 32, no. 1, pp. 69–84, 2011.

[50] B. Bigdeli, F. Samadzadegan, and P. Reinartz, "A decision fusion method based on multiple support vector machine system for fusion of hyperspectral and lidar data," *International Journal of Image and Data Fusion*, vol. 5, no. 3, pp. 196–209, 2014.

[51] Q. Man, P. Dong, and H. Guo, "Pixel-and feature-level fusion of hyperspectral and lidar data for urban land-use classification," *International Journal of Remote Sensing*, vol. 36, no. 6, pp. 1618–1644, 2015.

[52] J. García-Gutiérrez, D. Mateos-García, M. Garcia, and J. C. Riquelme-Santos, "An evolutionary-weighted majority voting and support vector machines applied to contextual classification of lidar and imagery data fusion," *Neurocomputing*, vol. 163, pp. 17–24, 2015.

[53] X. Xu, J. Li, and A. Plaza, "Fusion of hyperspectral and lidar data using morphological component analysis," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3575–3578, IEEE, 2016.

[54] Y. Cao, H. Wei, H. Zhao, and N. Li, "An effective approach for land-cover classification from airborne lidar fused with co-registered data," *International Journal of Remote Sensing*, vol. 33, no. 18, pp. 5927–5953, 2012.

[55] Z. Kang, J. Yang, and R. Zhong, "A bayesian-network-based classification method integrating airborne lidar data with optical images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2016.

[56] Y. Zhou and F. Qiu, "Fusion of high spatial resolution worldview-2 imagery and lidar pseudo-waveform for object-based image analysis," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 101, pp. 221–232, 2015.

[57] M. Pedergnana, P. R. Marpu, M. Dalla Mura, J. A. Benediktsson, and L. Bruzzone, "Classification of remote sensing optical and lidar data using extended attribute profiles," *IEEE Journal of Selected Topics in Signal Processing*, vol. 6, no. 7, pp. 856–865, 2012.

[58] Q. Zhang, R. Qin, X. Huang, Y. Fang, and L. Liu, "Classification of ultra-high resolution orthophotos combined with dsm using a dual morphological top hat profile," *Remote Sensing*, vol. 7, no. 12, pp. 16422–16440, 2015.

[59] R. Qin and W. Fang, "A hierarchical building detection method for very high resolution remotely sensed images combined with dsm using graph cut optimization," *Photogrammetric Engineering & Remote Sensing*, vol. 80, no. 9, pp. 873–883, 2014.

[60] H. Aytaylan and S. E. Yuksel, "Semantic segmentation of hyperspectral images with the fusion of lidar data," in *IEEE Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 2522–2525, IEEE, 2016.

[61] M. Gerke and J. Xiao, "Fusion of airborne laserscanning point clouds and images for supervised and unsupervised scene classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 87, pp. 78–92, 2014.

[62] I. Markus Gerke, "Use of the stair vision library within the isprs 2d semantic labeling benchmark (vaihingen),"

[63] S. Paisitkriangkrai, J. Sherrah, P. Janney, V.-D. Hengel, *et al.*, "Effective semantic pixel labelling with convolutional networks and conditional random fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 36–43, 2015.

[64] A. Lagrange and B. Le Saux, "Convolutional neural networks for semantic labeling," tech. rep., Tech. Rep., Onera–The French Aerospace Lab, 2015.

[65] J. Sherrah, "Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery," *arXiv preprint arXiv:1606.02585*, 2016.

[66] N. Audebert, B. L. Saux, and S. Lefèvre, "Semantic segmentation of earth observation data using multimodal and multi-scale deep networks," *arXiv preprint arXiv:1609.06846*, 2016.

[67] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla, "Semantic segmentation of aerial images with an ensemble of cnss," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016*, vol. 3, pp. 473–480, 2016.

[68] D. Marmanis, K. Schindler, J. D. Wegner, S. Galliani, M. Datcu, and U. Stilla, "Classification with an edge: improving semantic image segmentation with boundary detection," *arXiv preprint arXiv:1612.01337*, 2016.

[69] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez, "High-resolution semantic labeling with convolutional neural networks," *arXiv preprint arXiv:1611.01962*, 2016.

[70] S. Piramanayagam, W. Schwartzkopf, F. Koehler, and E. Saber, "Classification of remote sensed images using random forests and deep learning framework," in *SPIE Remote Sensing*, pp. 100040L–100040L, International Society for Optics and Photonics, 2016.

[71] M. Tschannen, L. Cavigelli, F. Mentzer, T. Wiatowski, and L. Benini, "Deep structured features for semantic segmentation," *arXiv preprint arXiv:1609.07916*, 2016.

[72] A. Marcu and M. Leordeanu, "Dual local-global contextual pathways for recognition in aerial imagery," *arXiv preprint arXiv:1605.05462*, 2016.

[73] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[74] G. Lin, C. Shen, A. van den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, pp. 3194–3203, 2016.

[75] N. Haala, H. Hastedt, K. Wolf, C. Ressl, and S. Baltrusch, "Digital photogrammetric camera evaluation–generation of digital elevation models," *Photogrammetrie-Fernerkundung-Geoinformation*, vol. 2010, no. 2, pp. 99–115, 2010.

[76] C. Lemaire, "Aspects of the dsm production with high resolution images," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, no. B4, pp. 1143–1146, 2008.

[77] L. Vincent, "Morphological grayscale reconstruction in image analysis: applications and efficient algorithms," *IEEE Transactions on Image Processing*, vol. 2, no. 2, pp. 176–201, 1993.

[78] L. Vincent, "Grayscale area openings and closings, their efficient implementation and applications," in *First Workshop on Mathematical Morphology and its Applications to Signal Processing*, pp. 22–27, 1993.

[79] M. Rutzinger, F. Rottensteiner, and N. Pfeifer, "A comparison of evaluation techniques for building extraction from airborne laser scanning," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 2, no. 1, pp. 11–20, 2009.

[80] F. Rottensteiner, G. Sohn, M. Gerke, J. D. Wegner, U. Breitkopf, and J. Jung, "Results of the isprs benchmark on urban object detection and 3d building reconstruction," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 93, pp. 256–271, 2014.

[81] Y. Boykov, "Max-flow/min-cut & multi-label optimization." `"http://vision.csd.uwo.ca/code"`.

Appendix A

Confusion Matrices and F1-score Tables

Both pixel based and object based confusion matrices for each group, F1-scores, which are calculated from the confusion matrices for all of the areas and each group can be found in this chapter. Pixel based and object based confusion matrices can be seen in Tables A.1 to A.6. F1-scores are shown in Tables A.7 to A.14.

Table A.1: Pixel-based confusion matrices for group 1. (a) Max-flow/min-cut with constant smoothness cost, (b) Max-flow/min-cut with adaptive smoothness cost, (c) Multi-label optimization with constant smoothness cost, (d) Multi-label optimization with adaptive smoothness cost.

(a)

		Predicted				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	2870445	239304	304284	47106	0.829
	Building	108285	4207154	59533	49747	0.951
	Grass	283691	154102	2551935	193294	0.802
	Tree	79871	45301	823310	1587099	0.626
	Correctness	0.859	0.906	0.683	0.845	**0.824**

(b)

		Predicted				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	2956195	239493	218534	46917	0.854
	Building	132408	4188747	35410	68154	0.947
	Grass	367835	141750	2467791	205646	0.775
	Tree	56908	50979	846273	1581421	0.624
	Correctness	0.841	0.906	0.692	0.831	**0.823**

(c)

		Predicted				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	3000367	198494	191535	70743	0.867
	Building	131367	4142943	62566	87843	0.936
	Grass	416655	128952	2262994	374421	0.711
	Tree	94633	31171	726389	1683388	0.664
	Correctness	0.824	0.920	0.698	0.760	**0.815**

(d)

		Predicted				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	3023410	196793	159860	81076	0.874
	Building	147989	4133261	44123	99346	0.934
	Grass	429370	111825	2143174	498653	0.673
	Tree	88426	32362	657685	1757108	0.693
	Correctness	0.820	0.924	0.713	0.721	**0.813**

Table A.2: Pixel-based confusion matrices for group 2. (a) Max-flow/min-cut with constant smoothness cost, (b) Max-flow/min-cut with adaptive smoothness cost, (c) Multi-label optimization with constant smoothness cost, (d) Multi-label optimization with adaptive smoothness cost.

(a)

		Predicted				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	4768083	279086	672087	120653	0.816
	Building	98863	4513164	57920	59651	0.954
	Grass	468409	234470	2894903	305765	0.742
	Tree	77089	90233	1112868	3407949	0.727
	Correctness	0.881	0.882	0.611	0.875	**0.813**

(b)

		Predicted				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	4787237	274932	652933	124807	0.820
	Building	96045	4498309	60738	74506	0.951
	Grass	449213	202319	2914099	337916	0.747
	Tree	110446	95776	1079511	3402406	0.726
	Correctness	0.880	0.887	0.619	0.864	**0.814**

(c)

		Predicted				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	4950650	246588	511524	131147	0.848
	Building	230916	4324546	68297	105839	0.914
	Grass	478440	192028	2781395	451684	0.713
	Tree	145253	75944	941901	3525041	0.752
	Correctness	0.853	0.894	0.646	0.837	**0.813**

(d)

		Confusion Matrix				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	5010942	205837	444932	178198	0.858
	Building	206463	4304460	53499	165176	0.910
	Grass	544639	142845	2715132	500931	0.696
	Tree	161221	64217	883313	3579388	0.763
	Correctness	0.846	0.912	0.663	0.809	**0.815**

Table A.3: Pixel-based confusion matrices for group 3. (a) Max-flow/min-cut with constant smoothness cost, (b) Max-flow/min-cut with adaptive smoothness cost, (c) Multi-label optimization with constant smoothness cost, (d) Multi-label optimization with adaptive smoothness cost.

(a)

	Confusion Matrix				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	3808806	421024	208526	54060	0.848
	Building	151276	4508230	45082	50911	0.948
	Grass	251648	144730	1635530	435402	0.663
	Tree	97727	57266	414225	2127316	0.789
Correctness		0.884	0.879	0.710	0.797	**0.838**

(b)

	Confusion Matrix				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	3805501	408665	211831	66419	0.847
	Building	153181	4498743	43177	60398	0.946
	Grass	248830	142871	1638348	437261	0.664
	Tree	76452	59340	435500	2125242	0.788
Correctness		0.888	0.880	0.703	0.790	**0.837**

(c)

	Confusion Matrix				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	3917859	253790	145127	175640	0.872
	Building	181716	4340695	36528	196560	0.913
	Grass	323112	92346	1407233	644619	0.570
	Tree	98387	59432	264889	2273826	0.843
Correctness		0.867	0.915	0.759	0.691	**0.828**

(d)

	Confusion Matrix				
		Road	**Building**	**Grass**	**Tree**	**Completeness**
Actual	**Road**	3909292	207544	142987	232593	0.870
	Building	175283	4259708	29580	290928	0.896
	Grass	322844	87604	1364154	692708	0.553
	Tree	91038	62539	261498	2281459	0.846
Correctness		0.869	0.923	0.759	0.652	**0.820**

Table A.4: Object-based confusion matrices for group 1. (a) Max-flow/min-cut with constant smoothness cost, (b) Max-flow/min-cut with adaptive smoothness cost, (c) Multi-label optimization with constant smoothness cost, (d) Multi-label optimization with adaptive smoothness cost.

(a)

		Correctness				
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	15	0	2	0	0.882
	Building	1	95	0	2	0.969
	Grass	2	0	29	0	0.935
	Tree	0	0	8	40	0.833
	Correctness$_{50}$	0.833	1.000	0.744	0.952	**0.923**

(b)

		Confusion Matrix				
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	17	0	1	0	0.944
	Building	1	92	0	1	0.979
	Grass	3	0	37	0	0.925
	Tree	0	0	13	40	0.755
	Correctness$_{50}$	0.810	1.000	0.725	0.976	**0.907**

(c)

		Confusion Matrix				
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	18	0	0	0	1.000
	Building	0	94	0	2	0.979
	Grass	2	0	37	4	0.860
	Tree	0	0	9	43	0.827
	Correctness$_{50}$	0.900	1.000	0.804	0.878	**0.919**

(d)

		Confusion Matrix				
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	17	0	0	0	1.000
	Building	1	93	0	1	0.979
	Grass	8	0	45	7	0.750
	Tree	0	0	7	44	0.863
	Correctness$_{50}$	0.654	1.000	0.865	0.846	**0.892**

Table A.5: Object-based confusion matrices for group 2. (a) Max-flow/min-cut with constant smoothness cost, (b) Max-flow/min-cut with adaptive smoothness cost, (c) Multi-label optimization with constant smoothness cost, (d) Multi-label optimization with adaptive smoothness cost.

(a)

	Confusion Matrix					
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	17	1	7	0	0.680
	Building	0	141	0	0	1.000
	Grass	6	1	75	3	0.882
	Tree	0	0	14	95	0.872
	Correctness$_{50}$	0.739	0.986	0.781	0.969	**0.911**

(b)

	Confusion Matrix					
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	17	1	11	0	0.586
	Building	0	143	0	0	1.000
	Grass	9	1	72	3	0.847
	Tree	0	0	12	96	0.889
	Correctness$_{50}$	0.654	0.986	0.758	0.970	**0.899**

(c)

	Confusion Matrix					
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	15	1	6	0	0.682
	Building	0	137	0	0	1.000
	Grass	8	0	83	4	0.874
	Tree	0	0	10	96	0.906
	Correctness$_{50}$	0.652	0.993	0.838	0.960	**0.919**

(d)

	Confusion Matrix					
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	17	1	5	0	0.739
	Building	0	141	0	0	1.000
	Grass	10	0	71	3	0.845
	Tree	0	0	8	95	0.922
	Correctness$_{50}$	0.630	0.993	0.845	0.969	**0.923**

Table A.6: Object-based confusion matrices for group 3. (a) Max-flow/min-cut with constant smoothness cost, (b) Max-flow/min-cut with adaptive smoothness cost, (c) Multi-label optimization with constant smoothness cost, (d) Multi-label optimization with adaptive smoothness cost.

(a)

	Confusion Matrix					
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	18	1	3	0	0.818
	Building	0	113	0	0	1.000
	Grass	0	0	35	4	0.897
	Tree	0	0	5	59	0.922
	Correctness$_{50}$	1.000	0.991	0.814	0.937	**0.945**

(b)

	Confusion Matrix					
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	18	1	2	0	0.857
	Building	0	112	0	0	1.000
	Grass	1	0	35	4	0.875
	Tree	0	0	5	61	0.924
	Correctness$_{50}$	0.947	0.991	0.833	0.938	**0.946**

(c)

	Confusion Matrix					
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	16	1	1	1	0.842
	Building	0	121	0	1	0.992
	Grass	2	0	38	5	0.844
	Tree	0	0	5	56	0.918
	Correctness$_{50}$	0.889	0.992	0.864	0.889	**0.935**

(d)

	Confusion Matrix					
		Road	**Building**	**Grass**	**Tree**	**Completeness$_{50}$**
Actual	**Road**	15	1	0	1	0.882
	Building	1	128	0	4	0.962
	Grass	2	0	41	4	0.872
	Tree	0	0	2	56	0.966
	Correctness$_{50}$	0.833	0.992	0.953	0.862	**0.941**

Table A.7: Pixel-based F1 scores for the results obtained by max-flow/min-cut with constant smoothness cost.

Group No	Area Info		Road	Building	Grass	Tree
Group 1	Train	area 1	0.865	0.913	0.624	0.753
		area 7	0.888	0.907	0.693	0.825
		area 28	0.832	0.919	0.730	0.779
		area 3	0.844	0.915	0.676	0.753
		area 11	0.860	0.887	0.635	0.843
		area 30	0.875	0.923	0.728	0.821
		area 34	0.817	0.940	0.698	0.731
	Test	area 5	0.893	0.951	0.570	0.804
		area 15	0.797	0.891	0.688	0.726
		area 37	0.796	0.910	0.809	0.638
	Overall Train		0.857	0.915	0.690	0.794
	Overall Test		0.844	0.928	0.737	0.719
Group 2	Train	area 1	0.870	0.914	0.610	0.752
		area 7	0.900	0.921	0.665	0.800
		area 28	0.838	0.919	0.737	0.779
		area 5	0.886	0.942	0.583	0.797
		area 15	0.760	0.892	0.657	0.726
		area 37	0.828	0.910	0.821	0.642
	Test	area 3	0.841	0.917	0.658	0.752
		area 11	0.853	0.888	0.620	0.842
		area 30	0.857	0.922	0.694	0.822
		area 34	0.835	0.939	0.702	0.730
	Overall Train		0.857	0.920	0.710	0.753
	Overall Test		0.847	0.917	0.670	0.794
Group 3	Train	area 3	0.846	0.907	0.687	0.778
		area 11	0.861	0.898	0.637	0.814
		area 30	0.867	0.907	0.708	0.824
		area 34	0.820	0.936	0.719	0.769
		area 5	0.873	0.923	0.470	0.789
		area 15	0.794	0.887	0.699	0.761
		area 37	0.799	0.906	0.813	0.695
	Test	area 1	0.873	0.919	0.642	0.747
		area 7	0.886	0.902	0.694	0.837
		area 28	0.831	0.912	0.704	0.764
	Overall Train		0.845	0.911	0.711	0.782
	Overall Test		0.865	0.912	0.686	0.793

Table A.8: Pixel-based F1 scores for the results obtained by max-flow/min-cut with adaptive smoothness cost.

Group No	Area Info		Road	Building	Grass	Tree
Group 1	Train	area 1	0.870	0.914	0.612	0.748
		area 7	0.891	0.908	0.680	0.819
		area 28	0.842	0.918	0.739	0.774
		area 3	0.849	0.921	0.666	0.750
		area 11	0.863	0.892	0.632	0.838
		area 30	0.865	0.930	0.704	0.812
		area 34	0.855	0.938	0.716	0.727
	Test	area 5	0.898	0.949	0.592	0.780
		area 15	0.769	0.891	0.662	0.724
		area 37	0.829	0.906	0.820	0.630
	Overall Train		0.862	0.917	0.685	0.789
	Overall Test		0.848	0.926	0.731	0.713
Group 2	Train	area 1	0.869	0.913	0.618	0.749
		area 7	0.898	0.919	0.669	0.799
		area 28	0.832	0.918	0.732	0.775
		area 5	0.884	0.940	0.563	0.781
		area 15	0.777	0.892	0.674	0.724
		area 37	0.818	0.905	0.816	0.628
	Test	area 3	0.840	0.919	0.659	0.748
		area 11	0.861	0.881	0.639	0.836
		area 30	0.862	0.930	0.707	0.812
		area 34	0.822	0.937	0.698	0.727
	Overall Train		0.856	0.919	0.713	0.749
	Overall Test		0.849	0.918	0.677	0.789
Group 3	Train	area 3	0.844	0.913	0.671	0.775
		area 11	0.867	0.899	0.646	0.813
		area 30	0.862	0.915	0.699	0.815
		area 34	0.843	0.936	0.734	0.766
		area 5	0.875	0.922	0.487	0.765
		area 15	0.786	0.890	0.693	0.759
		area 37	0.814	0.898	0.818	0.682
	Test	area 1	0.874	0.919	0.636	0.749
		area 7	0.886	0.902	0.688	0.833
		area 28	0.837	0.912	0.706	0.756
	Overall Train		0.848	0.912	0.710	0.777
	Overall Test		0.867	0.912	0.683	0.789

Table A.9: Pixel-based F1 scores for the results obtained by multi-label optimization with constant smoothness cost.

Group No	Area Info		Road	Building	Grass	Tree
Group 1	Train	area 1	0.869	0.922	0.609	0.760
		area 7	0.896	0.920	0.595	0.817
		area 28	0.844	0.915	0.720	0.758
		area 3	0.849	0.915	0.677	0.752
		area 11	0.883	0.912	0.658	0.838
		area 30	0.858	0.898	0.687	0.806
		area 34	0.844	0.932	0.710	0.730
	Test	area 5	0.897	0.955	0.500	0.780
		area 15	0.789	0.896	0.629	0.723
		area 37	0.804	0.895	0.799	0.605
	Overall Train		0.864	0.916	0.675	0.788
	Overall Test		0.845	0.928	0.704	0.709
Group 2	Train	area 1	0.883	0.923	0.632	0.758
		area 7	0.909	0.926	0.601	0.819
		area 28	0.829	0.905	0.724	0.760
		area 5	0.898	0.953	0.545	0.780
		area 15	0.783	0.891	0.634	0.729
		area 37	0.813	0.888	0.809	0.602
	Test	area 3	0.841	0.905	0.670	0.754
		area 11	0.875	0.905	0.643	0.838
		area 30	0.852	0.900	0.689	0.816
		area 34	0.824	0.907	0.706	0.730
	Overall Train		0.864	0.921	0.695	0.753
	Overall Test		0.850	0.904	0.678	0.792
Group 3	Train	area 3	0.850	0.898	0.689	0.751
		area 11	0.876	0.898	0.654	0.822
		area 30	0.854	0.891	0.689	0.800
		area 34	0.850	0.930	0.731	0.761
		area 5	0.896	0.953	0.487	0.755
		area 15	0.791	0.901	0.644	0.741
		area 37	0.816	0.889	0.816	0.632
	Test	area 1	0.867	0.909	0.619	0.725
		area 7	0.895	0.931	0.592	0.804
		area 28	0.842	0.904	0.700	0.720
	Overall Train		0.854	0.914	0.703	0.764
	Overall Test		0.869	0.914	0.651	0.760

Table A.10: Pixel-based F1 scores for the results obtained by multi-label optimization with adaptive smoothness cost.

Group No	Area Info		Road	Building	Grass	Tree
Group 1	Train	area 1	0.873	0.922	0.623	0.748
		area 7	0.894	0.917	0.575	0.814
		area 28	0.835	0.910	0.683	0.730
		area 3	0.846	0.914	0.643	0.752
		area 11	0.887	0.922	0.643	0.836
		area 30	0.860	0.924	0.647	0.797
		area 34	0.828	0.922	0.711	0.735
	Test	area 5	0.899	0.954	0.491	0.752
		area 15	0.780	0.900	0.592	0.723
		area 37	0.814	0.895	0.806	0.617
	Overall Train		0.863	0.918	0.653	0.781
	Overall Test		0.846	0.929	0.693	0.707
Group 2	Train	area 1	0.874	0.914	0.638	0.707
		area 7	0.900	0.919	0.589	0.801
		area 28	0.837	0.901	0.712	0.729
		area 5	0.899	0.952	0.531	0.728
		area 15	0.786	0.898	0.638	0.731
		area 37	0.821	0.894	0.816	0.608
	Test	area 3	0.846	0.909	0.667	0.745
		area 11	0.882	0.922	0.652	0.833
		area 30	0.844	0.906	0.675	0.802
		area 34	0.824	0.913	0.720	0.736
	Overall Train		0.862	0.919	0.694	0.734
	Overall Test		0.852	0.911	0.679	0.786
Group 3	Train	area 3	0.846	0.896	0.663	0.738
		area 11	0.875	0.903	0.647	0.815
		area 30	0.854	0.904	0.650	0.773
		area 34	0.853	0.919	0.735	0.747
		area 5	0.896	0.952	0.481	0.697
		area 15	0.773	0.908	0.628	0.743
		area 37	0.822	0.894	0.825	0.638
	Test	area 1	0.869	0.903	0.637	0.689
		area 7	0.894	0.929	0.573	0.791
		area 28	0.842	0.898	0.678	0.693
	Overall Train		0.852	0.917	0.692	0.751
	Overall Test		0.870	0.909	0.640	0.737

Table A.11: Object-based F1 scores for the results obtained by max-flow/min-cut with constant smoothness cost.

Group No	Area Info		Road	Building	Grass	Tree
Group 1	Train	area 1	0.933	1.000	0.900	0.966
		area 7	1.000	1.000	0.720	0.800
		area 28	0.857	0.989	0.919	0.968
		area 3	0.250	1.000	0.849	0.923
		area 11	0.800	0.982	0.829	0.917
		area 30	0.750	0.987	0.909	0.982
		area 34	0.727	1.000	0.667	0.905
	Test	area 5	0.800	1.000	0.889	1.000
		area 15	0.923	0.963	0.800	0.862
		area 37	0.833	1.000	0.818	0.900
	Overall Train		0.795	0.994	0.843	0.929
	Overall Test		0.857	0.984	0.829	0.889
Group 2	Train	area 1	0.947	1.000	0.900	0.968
		area 7	1.000	1.000	0.741	0.821
		area 28	0.833	0.989	0.914	0.968
		area 5	0.750	0.978	0.857	0.857
		area 15	0.933	0.963	0.842	0.862
		area 37	1.000	1.000	0.813	0.750
	Test	area 3	0.545	1.000	0.838	0.885
		area 11	0.667	0.982	0.789	0.917
		area 30	0.667	0.987	0.917	0.982
		area 34	0.875	1.000	0.667	0.884
	Overall Train		0.923	0.988	0.844	0.886
	Overall Test		0.708	0.993	0.829	0.918
Group 3	Train	area 3	0.444	1.000	0.885	0.969
		area 11	0.667	0.963	0.750	0.865
		area 30	0.667	0.975	0.889	0.984
		area 34	0.800	1.000	0.828	0.923
		area 5	0.750	0.952	0.750	0.857
		area 15	0.923	0.962	0.791	0.836
		area 37	0.833	1.000	0.846	0.909
	Test	area 1	0.923	1.000	0.900	0.968
		area 7	1.000	1.000	0.783	0.839
		area 28	0.769	0.988	0.872	0.954
	Overall Train		0.729	0.981	0.831	0.914
	Overall Test		0.900	0.996	0.854	0.929

Table A.12: Object-based F1 scores for the results obtained by max-flow/min-cut with adaptive smoothness cost.

Group No	Area Info		Road	Building	Grass	Tree
Group 1	Train	area 1	0.947	1.000	0.900	0.970
		area 7	1.000	1.000	0.720	0.800
		area 28	0.833	0.988	0.895	0.951
		area 3	0.429	1.000	0.811	0.906
		area 11	0.800	0.982	0.800	0.898
		area 30	0.714	0.987	0.889	0.964
		area 34	0.875	1.000	0.667	0.905
	Test	area 5	0.889	1.000	0.933	1.000
		area 15	0.824	0.975	0.769	0.867
		area 37	0.923	1.000	0.811	0.750
	Overall Train		0.804	0.994	0.823	0.918
	Overall Test		0.872	0.989	0.813	0.851
Group 2	Train	area 1	0.941	1.000	0.909	0.968
		area 7	0.909	1.000	0.786	0.857
		area 28	0.769	0.988	0.872	0.951
		area 5	0.800	0.977	0.818	0.800
		area 15	0.824	0.976	0.788	0.900
		area 37	0.750	1.000	0.750	0.818
	Test	area 3	0.353	1.000	0.773	0.903
		area 11	0.727	0.982	0.800	0.917
		area 30	0.750	0.987	0.913	0.982
		area 34	0.727	1.000	0.632	0.905
	Overall Train		0.833	0.990	0.818	0.902
	Overall Test		0.618	0.993	0.800	0.928
Group 3	Train	area 3	0.364	1.000	0.828	0.952
		area 11	0.750	0.963	0.769	0.865
		area 30	0.571	0.975	0.894	0.983
		area 34	0.857	1.000	0.800	0.927
		area 5	0.857	0.952	0.778	0.824
		area 15	0.857	0.976	0.792	0.861
		area 37	0.778	1.000	0.824	0.909
	Test	area 1	0.857	1.000	0.857	0.966
		area 7	1.000	1.000	0.762	0.848
		area 28	0.833	0.987	0.900	0.957
	Overall Train		0.723	0.983	0.818	0.913
	Overall Test		0.900	0.996	0.854	0.931

Table A.13: Object-based F1 scores for the results obtained by multi-label optimization with constant smoothness cost.

Group No	Area Info		Road	Building	Grass	Tree
Group 1	Train	area 1	0.857	1.000	0.824	0.968
		area 7	0.933	0.985	0.824	0.733
		area 28	0.800	0.989	0.900	0.952
		area 3	0.444	1.000	0.857	0.921
		area 11	1.000	1.000	0.909	0.943
		area 30	0.727	1.000	0.941	1.000
		area 34	0.824	1.000	0.640	0.857
	Test	area 5	0.889	1.000	0.889	0.923
		area 15	0.933	0.975	0.800	0.848
		area 37	1.000	1.000	0.846	0.818
	Overall Train		0.805	0.996	0.859	0.922
	Overall Test		0.947	0.989	0.831	0.851
Group 2	Train	area 1	0.889	0.988	0.900	0.970
		area 7	1.000	0.986	0.872	0.800
		area 28	0.889	1.000	0.895	0.951
		area 5	0.857	1.000	0.900	0.923
		area 15	0.875	0.964	0.791	0.853
		area 37	1.000	1.000	0.786	0.750
	Test	area 3	0.400	1.000	0.857	0.918
		area 11	0.889	1.000	0.889	0.939
		area 30	0.667	1.000	0.909	0.982
		area 34	0.714	0.971	0.692	0.878
	Overall Train		0.919	0.988	0.851	0.882
	Overall Test		0.667	0.996	0.856	0.932
Group 3	Train	area 3	0.400	1.000	0.897	1.000
		area 11	0.800	0.960	0.865	0.872
		area 30	0.769	1.000	0.927	1.000
		area 34	0.769	1.000	0.720	0.913
		area 5	0.857	1.000	0.824	0.833
		area 15	1.000	0.975	0.760	0.800
		area 37	1.000	0.966	0.867	0.750
	Test	area 1	0.769	0.989	0.737	0.914
		area 7	0.923	1.000	0.875	0.815
		area 28	0.909	0.989	0.895	0.935
	Overall Train		0.819	0.987	0.845	0.896
	Overall Test		0.865	0.992	0.854	0.903

Table A.14: Object-based F1 scores for the results obtained by multi-label optimization with adaptive smoothness cost.

Group No	Area Info		Road	Building	Grass	Tree
Group 1	Train	area 1	0.875	1.000	0.889	0.903
		area 7	1.000	0.985	0.867	0.839
		area 28	0.727	0.989	0.837	0.918
		area 3	0.364	1.000	0.870	0.966
		area 11	0.857	1.000	0.867	0.936
		area 30	0.833	1.000	0.913	0.962
		area 34	0.714	0.974	0.690	0.857
	Test	area 5	0.857	1.000	0.889	0.923
		area 15	0.778	0.975	0.769	0.853
		area 37	0.778	1.000	0.810	0.818
	Overall Train		0.782	0.994	0.854	0.919
	Overall Test		0.791	0.989	0.804	0.854
Group 2	Train	area 1	0.875	0.968	0.957	0.889
		area 7	0.800	0.960	0.895	0.875
		area 28	0.727	0.979	0.878	0.935
		area 5	0.889	1.000	0.889	0.923
		area 15	0.824	0.963	0.800	0.857
		area 37	0.800	1.000	0.842	0.900
	Test	area 3	0.400	1.000	0.870	0.951
		area 11	0.800	1.000	0.839	0.936
		area 30	0.750	1.000	0.909	1.000
		area 34	0.714	0.973	0.667	0.884
	Overall Train		0.818	0.976	0.865	0.893
	Overall Test		0.680	0.996	0.845	0.945
Group 3	Train	area 3	0.333	0.991	0.857	0.983
		area 11	0.769	0.962	0.839	0.900
		area 30	0.667	0.986	0.850	0.936
		area 34	0.833	0.970	0.769	0.898
		area 5	0.800	1.000	0.778	0.769
		area 15	0.700	0.963	0.724	0.831
		area 37	0.778	0.984	0.800	0.783
	Test	area 1	0.857	0.980	0.889	0.941
		area 7	0.923	0.971	0.929	0.815
		area 28	0.750	0.979	0.909	0.935
	Overall Train		0.696	0.980	0.803	0.890
	Overall Test		0.857	0.977	0.911	0.911