Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Towards a Digital Twin of Bologna: Features Extraction and Semantic Classification Using LiDAR

Supervisor:
Prof. Mirko Degli Esposti

Submitted by:
Tommaso Rondini

Academic Year 2023/2024

Upward, not Northward

Edwin A. Abbot
Flatland

Abstract

The development of an Urban Digital Twin has been become a shared objective for many cities around the world. Airborne LiDAR has been one of the most used technologies to reconstruct the urban environment, but an unified univocal method to analyse its point cloud data has not been developed yet. The techniques differ on selected features, algorithms and input data.

Within the bounds of Bologna Digital Twin, the aims of my thesis are:

- analyzing LiDAR data from a physical-geometrical point of view;
- to select features that extrapolate useful information in Bologna LiDAR from all the features found in literature;
- to develop a model to classify point into defined classes, i.e., buildings, cars, grass, rails, roads and trees. It does not have to be the best algorithm or scalable in other contexts, but it must serve as a starting point for future developments.

I discovered how much the three-dimensional distribution of point cloud encompasses information about the analyzed objects, therefore spatial features should be included in the inputs of classification algorithm. The regulation of the feature extraction process was conducted and the importance ranking of all the analyzed features was calculated. A random forest model was developed to classify LiDAR points and it achieves 95% of accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Digital Twin

1.1.1 General information

Digital Twin constitutes one of the most relevant and influential innovations in the field of digital transformation and Industry 4.0 [1]. It is a dynamic model that evolves in parallel with its physical counterpart, representing the functioning of physical systems belonging to the real world. This typology of models has gained increasing importance with the emergence of the Internet of Things (IoT) [2] and cloud computing [3], facilitating integration and interconnection of enormous quantities of data generated by IoT sensors installed in the physical environment, forming an interconnected ecosystem. The Digital Twin allows organizations to make informed decisions through simulations and optimization of decisions in real-time [4].

The characteristic value of the Digital Twin (DT) lies in its ability to connect the physical world to the digital one, providing a simulated environment to test theories, simulate scenarios, predict results and guide decisions without directly influencing the physical element [5]. This is particularly useful in product lifecycle management, from its conception to its disposal [3], as well as in the optimization of complex systems such as energy or water networks, transport and urban infrastructure [6].

The notion of DT was first introduced in the early 2000s by Michael Grieves [1], although its origins can be traced back to earlier concepts of simulation and visualization of complex systems. For example, NASA, as early as the 1960s for space missions, developed models called “twins” specifically for simulating and visualizing operations in critical conditions, avoiding risks to the safety of astronauts during missions [7].

With the evolution of technologies related to IoT, big data and cloud computing, the ability to develop and implement DTs has increased significantly, making this strategic concept central to the transformation in various sectors. DTs have become complex systems that can simulate entire production chains, infrastructures and even cities, opening up unprecedented possibilities for more efficient management of resources, urban planning and mobility in smart cities.

1.1.2 Urban Digital Twin

Among the numerous applications of DT, Urban Digital Twins emerge as a particularly innovative and transformative one. These models digitally simulate all the activities and interactions that take place in real-time within urban environments. These data, which cover aspects such as traffic flow, energy consumption, air quality, infrastructure and public services, are fundamental for creating powerful tools available to administrators, planners and citizens for more efficient urban management. The collection and the basement of all these tools is an Urban Digital Twins

The power of Urban Digital Twins (UDTs) lies in their ability to model and forecast. Like artificial intelligence for data processing and analysis, UDTs can anticipate the consequences of infrastructure changes, policy decisions, extraordinary events, or long-term development trends [8].

From an environmental point of view, UDTs play a fundamental role in sustainability and mitigation of climate impacts. Through precise simulations, they facilitate the optimization of resources and the reduction of CO_2 emissions, in order to increase the capacity of cities to adapt to climatic changes and orienting them towards a green economy [9].

The innovative character of UDTs concerns the strengthening of citizen participation and collaborative governance [10]. By offering interactive and intuitive digital platforms, UDTs promote more direct citizen involvement in urban planning and management, encouraging forms of feedback and suggestions.

In conclusion, UDTs represent powerful fundamental tools for revolutionizing cities, making them more livable, ecological and inclusive. They represent the practical application of the concept of "smart cities", where the strategic integration of technology and data aims to improve the quality of life and build resilient communities, ready to adapt to future challenges [7].

1.2 LiDAR

1.2.1 General

LiDAR (*Light Detection and Ranging*) is a remote sensing instrument for determining the distance to an object using laser pulse [11]. In detail, LiDAR (or LaDAR –*Laser Detection and Ranging*– or laser altimetry) is a optical sensor technology that emits intense, focused, mono-frequency beams and measures the time required for the reflected beams to return [12]. Time, intensity and waveform of the returned signal are used to calculate the interval or distance between objects, together with optical properties such

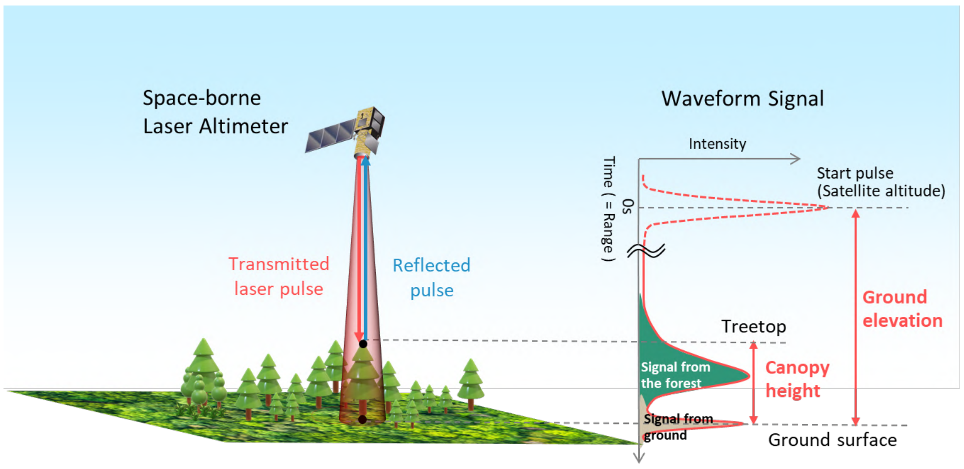


Figure 1.1: An illustration about LiDAR operation. On the left a sketch about emission and catching of the laser beam. On the right a sketch about multiple returns due to the canopy of a tree. Credit by European Space Agency - ESA.

as reflection and absorption. Such technology is similar in operation to RADAR (*Radio Detection and Ranging*), except that shorter wavelength are used than radio waves. Generally, LiDAR devices work in the near-infrared region [13], but there are sensors that work at both lower and higher frequencies [14] from infrared to visible, with the purpose of enabling a multispectral analysis or allowing penetration of water and analyse the seabed [15–17].

Fig. 1.1 shows how LiDAR technology works. The device emits a laser pulse; when the impulse hits the object's surface, the reflected part comes back to the device which calculates the time taken to return and the fraction of energy received compared to the one emitted. Since the pulse can be emitted far from the object's surface, the footprint can no longer be considered a point, but rather a surface. Thus, part of the pulse can be reflected from a surface and another part from a different object below: this happens on edges or on little objects (such as leaves) [18]. Multiple returns can be due to atmospheric phenomena or bidirectional scattering distribution function in thin objects [19].

The deployment of this technology is associated with three categories: satellite, airborne, and terrestrial. The use of satellite is mainly related to the study of celestial bodies, for example in searching for landing site [20–22]. Terrestrial systems are used both on fixed mounts and on vehicles for the reconstruction of the environment. Many companies in the autonomous driving sector use it [23–25]. Currently they are used for augmented reality, too [26, 27].

The employing of planes, helicopter and drones to acquire airborne LiDAR permits the acquisition of detailed information (with respect to satellite acquisition) of wide areas [28]. It is commonly adopted in various fields: reconstruction of cities geography and elements, hood and forest density and biodiversity studies, identification of water basins, and damage quantification after a disaster and so forth.

Tamburlin et al. [29] used LiDAR data of Italian Alps to distinguish three different trees species. They proposed the height variation hypothesis: the higher the variation in tree height, the more complex the overall structure of the forest and the higher the tree species diversity. Hirschmann [30] analysed different metrics –canopy height, vegetation arrangement, canopy cover, structural complexity, and leaf area and density– and their spatial resolution (grain and extension) over which their structural metrics are calculated to study biodiversity.

Tao et al. [31] proposed a method for determining the correct water surface depth from green laser (532 nm) echoes alone in bathymetric LiDAR. Illig et al. [28] analysed airborne blue wavelengths LiDAR data taken in the western zone of San Diego and in water surrounding Iceland. They provided information about mixed layer depth variation and dense plankton layer. The authors highlighted the merits of airborne LiDAR data that allow scanning quickly wide areas with a resolution of the order of meters. The in situ oceanographic measurements can difficultly achieve efficient coverage of wide areas, whereas satellites have a resolution of the order of kilometres.

A quick comparison of LiDAR data before and after a disaster can be done to extract roadways to assess road conditions [32]. Karantanellis et al. [33] used LiDAR and other data to evaluate landslide hazard in the area of Red Beach on Santorini Island, comparing the difference between two datasets acquired seven months apart. The research revealed the accumulation area of failure and flow direction and allowed the quantification of the mass movement.

1.2.2 LiDAR for cities

Since the early 2000s LiDAR data have been widely used for the classification of urban scenarios [34]. The various techniques developed to reconstruct 3D models and digital twins of cities differ according to the type of data available, preprocessing and classification methods. Models usually divide the data into four classes: ground, low vegetation, buildings and trees. Some models manage to distinguish also other categories (e.g. cars, stretches of water, high voltage lines).

Different LiDAR sensors provide different LiDAR data. For each beam, it can be saved information about only the returned signal [35] –so object distance and reflection information–, or multiple-returns too –that allows to consider object complexity–, or the whole waveform of the returned signal [36].

Since 2014 there have been commercial multispectral airborne LiDAR systems that allow obtaining spectral information of the surface. Morsy et al. [14] used a system that operates at three wavelength of 1550 nm, 1064 nm and 532 nm. It was very useful to categorize points because some surfaces are not visible in the whole spectrum. For example, water absorbs infrared radiations and reflects (at least in part) the green light which allows it to well detect swimming pools.

Some studies also use the point cloud and extract features from other data. This happens because urban areas are usually complex and dynamic environments, which makes it difficult to meet all the requirements in a single sensor [37]. Therefore, other data are used to improve classification [38]. Guo et al. [39] combined multi-echoes LiDAR data, full-waveform LiDAR data and multispectral images data to classify dense urban areas. Awrangjeb et al. [40] used colour and texture information to classify point clouds by integrating LiDAR data and multispectral orthophotos.

Many classification techniques and pipelines have only been used on LiDAR data or with other information type. Morsy et al. [14] used a multivariate Gaussian decomposition over their features histograms, after they had distinguished between ground and elevated points through a classification bases on the slope and the difference of height from the lowest point of a neighbourhood. Oliveira and Marçal [41] tested K-mean and DBSCAN clustering methods, showing that the former outperforms the latter. Suárez et al. [42]'s approach contains random forest and Gradient Boosting Machine classifiers and convolution neural networks. Xu Fan [43] converted LiDAR point clouds into 2D raster data and combined them with image data to extract various features and selected a Support Vector Machine as classifier.

A turning point in the use of deep learning techniques was PointNet, which directly uses 3D points as input [44]. The basis of this method consists of a succession of fully connected layers. However it is not able to capture the spatial correlation between points. In order to overcome this issue, alternative point-wise deep learning networks architectures were developed such as SuperPoint Graph [45], PointCNN [46], and DGCNN [47]. Widyaningrum et al. [48] implemented the point-wise deep learning method Dynamic Graph Convolutional Neural Network (DGNCC) trained on an existing 2D base map. In 2017 Ruizhongtai Qi et al. [49] developed PointNet++, a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. By exploiting metric space distances, the network is able to learn local features with increasing contextual scales, therefore spatial correlation is taken in consideration. This neural network was applied to LiDAR data: it is an interpolation method that uses adaptive elevation weight to make full use of the objects in the airborne LiDAR point, which exhibits discrepancies in elevation distributions [50].

1.3 Bologna dataset

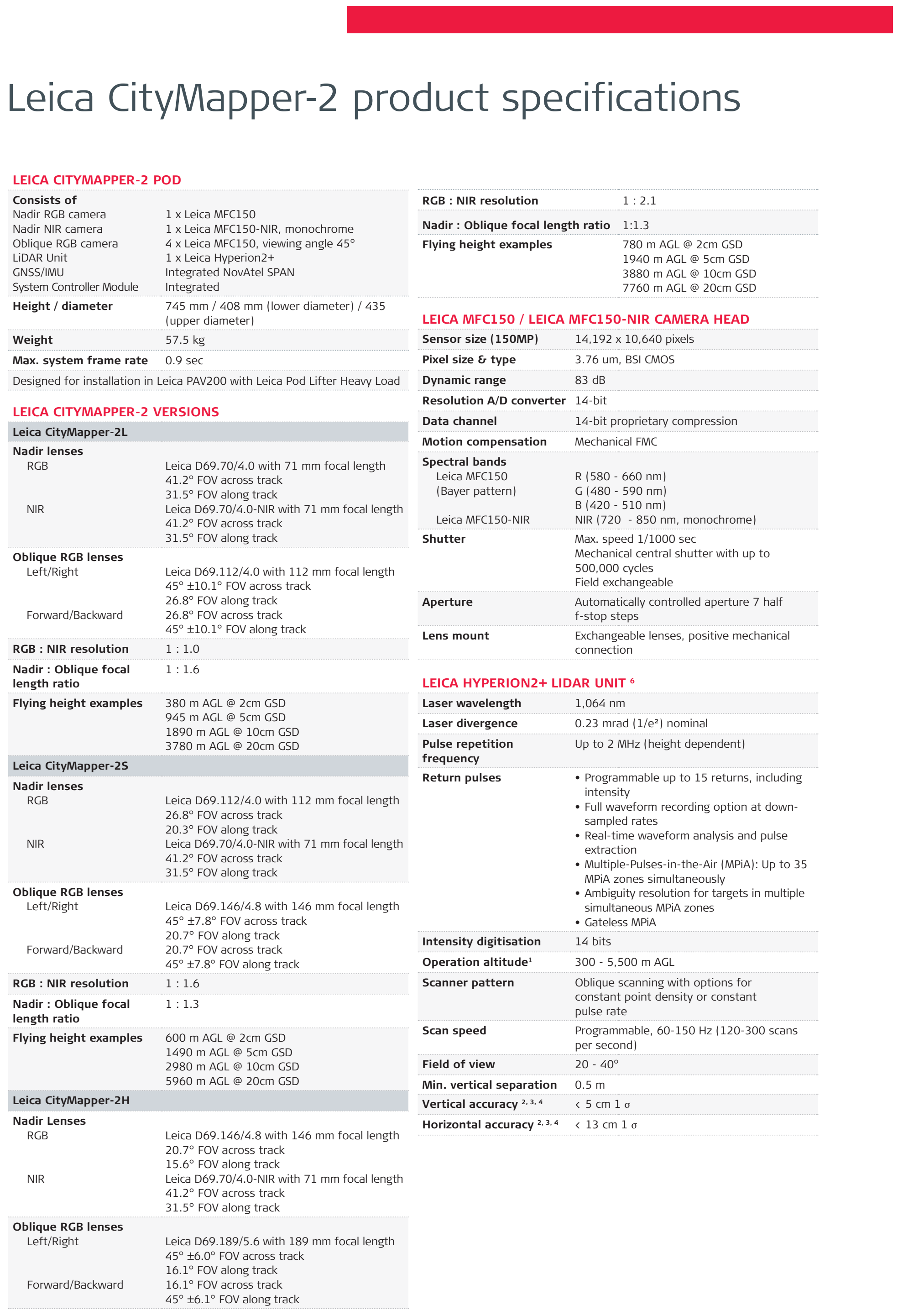
Airborne data were commissioned by Comune di Bologna to *Compagnia Generale Ripreseaeree* (CGR) [51]. It was employed CityMapper-2 sensor manufactured by Leica Geosystem [52]. It is equipped with six digital camera, two of which are nadiral –one RGB and one in near infrared (NIR)– and the rest are inclined of 45°. Image sensor size is $14\,192 \times 10\,640$ pixels and each pixel size is 3.76 µm. CityMapper-2 has a LiDAR acquisition unit with acquire frequency at most 2 MHz, while the beam wavelength is

1064 nm. Per each beam it can get at most 15 echoes, storing return time and intensity. In Fig. 1.2 there are CityMapper-2 specifications.

The plane was flying at speed and altitude so that the ground sample distance (i.e., how many meters the side of a pixel corresponds to in reality) was 5 cm on average and the average density of LiDAR points was greater than 20 points/m^2 with a resolution of 10 cm. In addition to raw data, CGR provided pre-processed data. They assign to each point red, green, blue and NIR indices, taken by photos. Furthermore, points were categorized into ground and off-ground points through an algorithm. It is based on the proposed one by Axelsson [53] in 2000, which uses progressive densification. The so found ground points were then used to create the Digital Terrain Model (DTM) [54]. DTM represents the share of bare ground, excluding buildings and high vegetation. Contra, the Digital Surface Model (DSM) takes into account all the elements, excluding multiple returns. It is like a bed sheet over the area. In Fig. 1.3 there is an example of DSM and DTM in a realistic context.

The classified point clouds (plus RGB-NIR indices) are stored in LAS files [55], while DTM and DSM are in ASCI binary ones. Each file covers a 500 m $\times$ 500 m area (aka "tile") of Bologna; there are 654 tiles. In Fig. 1.4 the subdivision of municipality into tiles is shown.

Per each LiDAR point there is other information in addition to coordinates, RGB-NIR and ground classification. Among them, CGR provides the return intensity, the number of returns the beam has made and which return is that point among all the returns of the beam. Unfortunately, there are some mistakes in the return data: in some cases, the number of returns is greater than the number of returned points per beam. Maybe it is due to the cleaning procedure carried out by CGR. In these cases I reordered the return number of the points so that they are consecutive without gaps and I set the number of returns as the number of returned points. The procedure may introduce errors, especially if two consecutive beams with multiple echoes contain mistakes, but statistically it is unlikely, since only a few beams have multiple echoes, and just three or four of them contain mistakes in a tile.



Leica CityMapper-2 product specifications

LEICA CITYMAPPER-2 POD

Consists of	
Nadir RGB camera	1 x Leica MFC150
Nadir NIR camera	1 x Leica MFC150-NIR, monochrome
Oblique RGB camera	4 x Leica MFC150, viewing angle 45°
LiDAR Unit	1 x Leica Hyperion2+
GNSS/IMU	Integrated NovAtel SPAN
System Controller Module	Integrated
Height / diameter	745 mm / 408 mm (lower diameter) / 435 (upper diameter)
Weight	57.5 kg
Max. system frame rate	0.9 sec
Designed for installation in Leica PAV200 with Leica Pod Lifter Heavy Load	

LEICA CITYMAPPER-2 VERSIONS

Leica CityMapper-2L	
Nadir lenses	
RGB	Leica D69.70/4.0 with 71 mm focal length 41.2° FOV across track 31.5° FOV along track
NIR	Leica D69.70/4.0-NIR with 71 mm focal length 41.2° FOV across track 31.5° FOV along track
Oblique RGB lenses	
Left/Right	Leica D69.112/4.0 with 112 mm focal length 45° ±10.1° FOV across track 26.8° FOV along track
Forward/Backward	26.8° FOV across track 45° ±10.1° FOV along track
RGB : NIR resolution	1 : 1.0
Nadir : Oblique focal length ratio	1 : 1.6
Flying height examples	380 m AGL @ 2cm GSD 945 m AGL @ 5cm GSD 1890 m AGL @ 10cm GSD 3780 m AGL @ 20cm GSD
Leica CityMapper-2S	
Nadir lenses	
RGB	Leica D69.112/4.0 with 112 mm focal length 26.8° FOV across track 20.3° FOV along track
NIR	Leica D69.70/4.0-NIR with 71 mm focal length 41.2° FOV across track 31.5° FOV along track
Oblique RGB lenses	
Left/Right	Leica D69.146/4.8 with 146 mm focal length 45° ±7.8° FOV across track 20.7° FOV along track
Forward/Backward	20.7° FOV across track 45° ±7.8° FOV along track
RGB : NIR resolution	1 : 1.6
Nadir : Oblique focal length ratio	1 : 1.3
Flying height examples	600 m AGL @ 2cm GSD 1490 m AGL @ 5cm GSD 2980 m AGL @ 10cm GSD 5960 m AGL @ 20cm GSD
Leica CityMapper-2H	
Nadir Lenses	
RGB	Leica D69.146/4.8 with 146 mm focal length 20.7° FOV across track 15.6° FOV along track
NIR	Leica D69.70/4.0-NIR with 71 mm focal length 41.2° FOV across track 31.5° FOV along track
Oblique RGB lenses	
Left/Right	Leica D69.189/5.6 with 189 mm focal length 45° ±6.0° FOV across track 16.1° FOV along track
Forward/Backward	16.1° FOV across track 45° ±6.1° FOV along track
RGB : NIR resolution	1 : 2.1
Nadir : Oblique focal length ratio	1:1.3
Flying height examples	780 m AGL @ 2cm GSD 1940 m AGL @ 5cm GSD 3880 m AGL @ 10cm GSD 7760 m AGL @ 20cm GSD

LEICA MFC150 / LEICA MFC150-NIR CAMERA HEAD

Sensor size (150MP)	14,192 x 10,640 pixels
Pixel size & type	3.76 um, BSI CMOS
Dynamic range	83 dB
Resolution A/D converter	14-bit
Data channel	14-bit proprietary compression
Motion compensation	Mechanical FMC
Spectral bands	
Leica MFC150 (Bayer pattern)	R (580 - 660 nm) G (480 - 590 nm) B (420 - 510 nm)
Leica MFC150-NIR	NIR (720 - 850 nm, monochrome)
Shutter	Max. speed 1/1000 sec Mechanical central shutter with up to 500,000 cycles Field exchangeable
Aperture	Automatically controlled aperture 7 half f-stop steps
Lens mount	Exchangeable lenses, positive mechanical connection

LEICA HYPERION2+ LIDAR UNIT [6]

Laser wavelength	1,064 nm
Laser divergence	0.23 mrad ($1/e^2$) nominal
Pulse repetition frequency	Up to 2 MHz (height dependent)
Return pulses	• Programmable up to 15 returns, including intensity • Full waveform recording option at down-sampled rates • Real-time waveform analysis and pulse extraction • Multiple-Pulses-in-the-Air (MPiA): Up to 35 MPiA zones simultaneously • Ambiguity resolution for targets in multiple simultaneous MPiA zones • Gateless MPiA
Intensity digitisation	14 bits
Operation altitude[1]	300 - 5,500 m AGL
Scanner pattern	Oblique scanning with options for constant point density or constant pulse rate
Scan speed	Programmable, 60-150 Hz (120-300 scans per second)
Field of view	20 - 40°
Min. vertical separation	0.5 m
Vertical accuracy [2, 3, 4]	< 5 cm 1 σ
Horizontal accuracy [2, 3, 4]	< 13 cm 1 σ

Figure 1.2: It shows the CityMapper-2 specifications. LiDAR information is in the second column. Credit by *Leica Geosystem* [52].

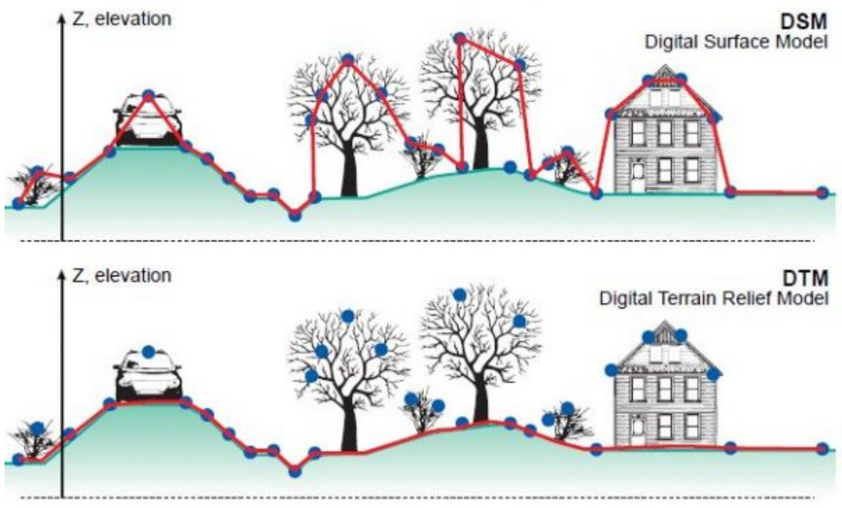


Figure 1.3: On the top we see Digital Surface Model as a red line. The line connects only higher points and not second returns. On the bottom there is the Digital Terrain Model as a red line. It connects the ground points and ignores data about cars, trees and buildings. Credit by 3dmetrica.

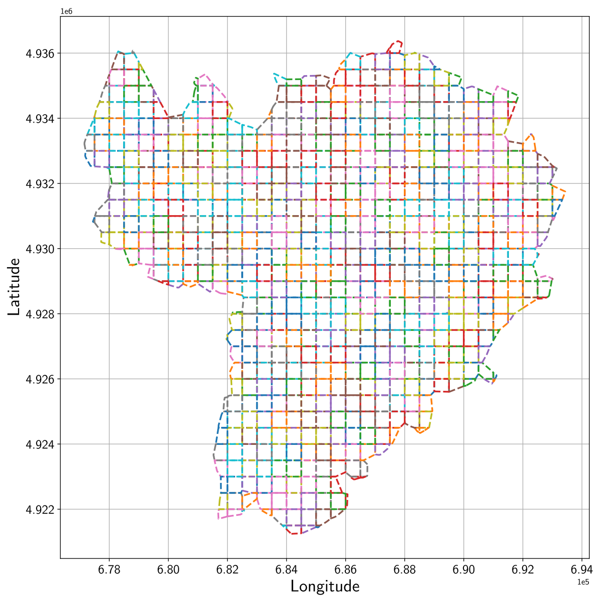


Figure 1.4: The picture shows how the municipality of Bologna area is subdivided in tiles of 500 m × 500 m. The tiles at the edges are incomplete and there are no data outside the municipality boundary.

Chapter 2

Point distributions

In this initial section, I investigate the distribution of points to gain a better understanding of our data and to perform geometric and physical analyses of the point cloud. I studied densities and how they change in tiles with different percentage of trees and buildings. These analyses will not be used directly to labelling point clouds, but they are useful to set features parameters, such as neighbourhood radius (see Sec. 3.3 and App. B) or *segment-lidar* resolution (see Sec. 4.1.1 and App. C).

These analyses are conducted to study point densities and how they change at different scales. In order to evaluate different real situations, I executed density analyses over four tiles: one has a high percentage of trees (Fig. 2.1a), one has a high percentage of buildings (Fig. 2.1d) and the remaining two are intermediate (Fig. 2.1b and Fig. 2.1c).

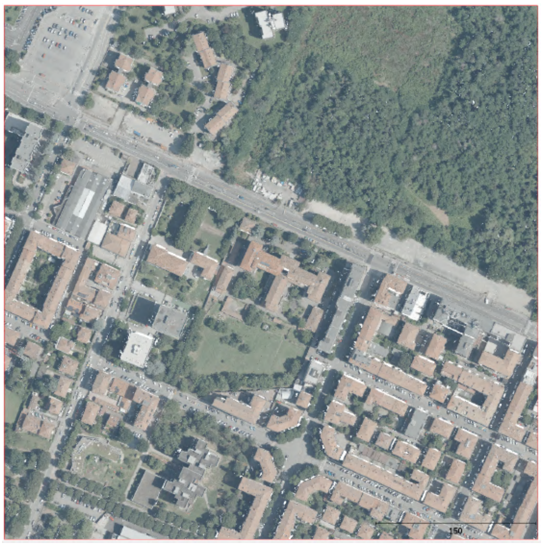
2.1 Density

One of the most useful information is how many points per unit of area do we have. The correct question would have be how many points per unit of volume because LiDAR are three dimensional data, but they are almost always spread on a two dimensional surface that lies in a three dimensional space. Furthermore, the aircraft flies at the same speed all of the time, so the LiDAR device emits the same amount of pulses all over the type of context whether it is flying over a forest or a built-up area, but the distribution in height changes consistently whether it flies over trees or asphalt.

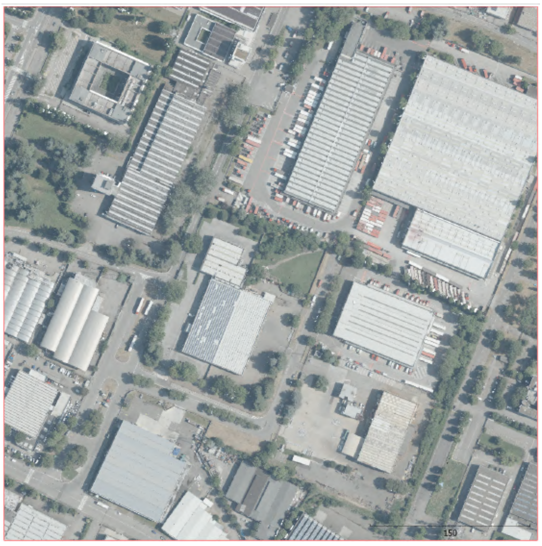
In light of all of this, I calculated the density this way: per each point I counted how many points there were inside a vertical cylinder with a radius of 0.564 m (a base area of about 1 m^2) centred in the selected point. The histograms for the four tiles (Fig. 2.1) are reported in Fig. 2.2, while their mean values are in the fifth column of Tab. 2.1.

Albeit the peak value shifts to higher density increasing the percentage of trees, the histograms are very similar to each other. There is just one peak per graph with only a right shoulder, which is bigger if there are more trees, highlighting higher density of points. Probably it is due to a greater spread of points in all three dimensions and not

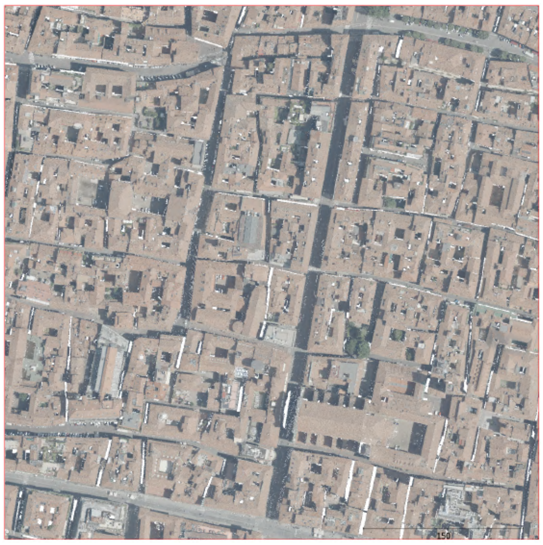
(a)



(b)



(c)



(d)

Figure 2.1: These are the point clouds whose density I analysed. a) has got a high percentage of trees and it has been taken over "via di Casaglia" and "via di Monte Albano". b) and c) have got both buildings and trees and they have been taken over "via Emilia" and "via Piave" (b) and over "via del Carrozzaio" and "via Cerodolo" (c). d) has got a high percentage of buildings and it has been taken over "via dell'Indipendenza" and "via Ugo Bassi".

only on a two dimensional surface

2.2 Trend at different scales

Since we are interested in the geometrical-physical model of LiDAR data, I conducted an analysis about point density. It is preparatory to determine characteristics of neighbourhood (shape and radius), which is used to calculate many features, as we shall see in the next chapter and in App. B.

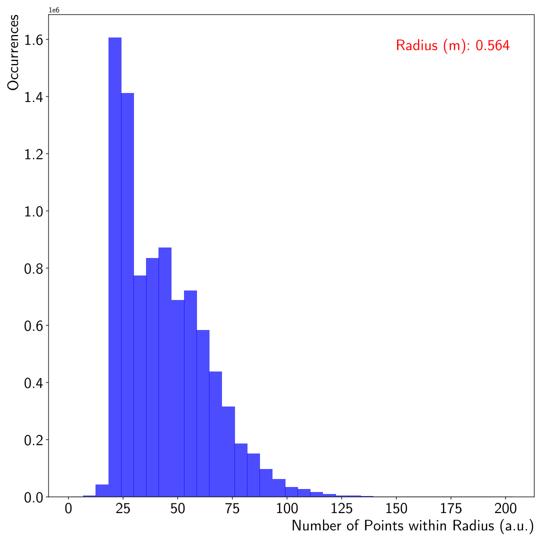
Two types of density have been studied. The first is the "common" one: per each point I set a radius and count how many points there were in the sphere. The second is the reverse: per each point I set a number of near points k and I found the minimum radius of a sphere so that it contained at least k points. I want to specify that I am talking about "density" but it is not exactly correct: I did not divide the number of points by the area or the volume, I just analysed how the number of points changed by fixing the radius and vice versa. These two analyses should be equivalent, but I executed both for a practical reason: we might be more interested in finding the first one, but it is more computationally demanding to execute with KD-Tree [56] (a very common and useful tools with LiDAR data [57]). Once I succeded in proving that they are the same analysis, we will always use the k-nearest density and then switch it.

If we consider just the first return per emitted pulses on a smooth surface, we expect the number of points inside the sphere to grow as the square of the radius, even if we are considering a sphere and not a circle. In the same conditions, the distance of the k-nearest point should grow as the square root of k. However in reality we can find irregular surfaces and, more importantly, multiple returns from the same emitted pulse. This can change the relationship between sphere radius and k-nearest distance.

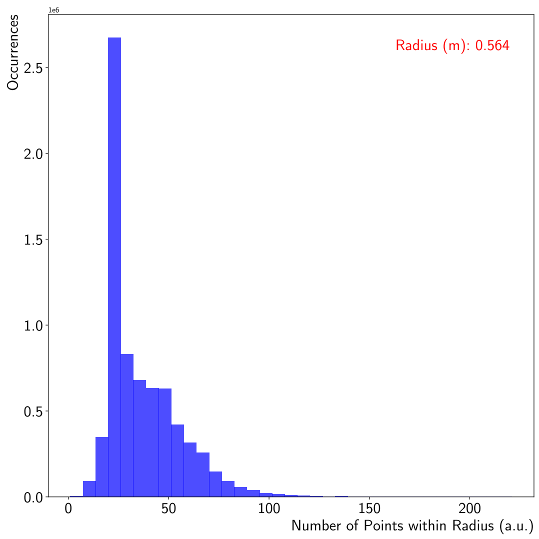
I calculated the k-nearest point distance with k between 2 and 6000 with a step of 100 and I calculated the number of points inside a sphere with a radius between 0.2 m and 12.0 m with a step of 0.2 m. Some of the histograms for the four tiles are shown in App. A.

Per each histogram I calculated its mean value and mean error (blue square), then I plotted them and evaluated a power law fit (red line). Since per each value in the plot we have more than six million points, the error bars are not visible; their values are about 1×10^{-3}.

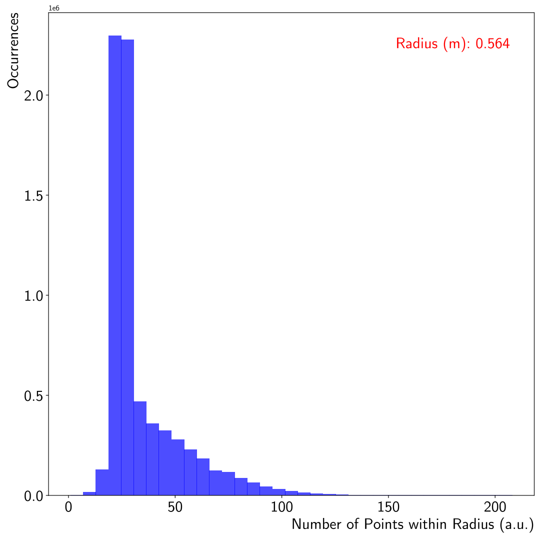
In Fig. 2.3 I report the k-nearest distance plots. The main trend is that the distance grows as the square root of k, but there are fluctuations. In Fig. 2.3a it grows faster than the other ones and it is related to the tile with many trees (Fig. 2.1a); the slowest (Fig. 2.3d) is the density of Fig. 2.1d, so the tile with almost just buildings. The same considerations are true for the number of points inside the sphere too (Fig. 2.4). All the plots grow as the square of the radius, but the more trees there are in the tile the faster they grow. In Tab. 2.1 I report all the coefficients of the fit functions for a fast check.



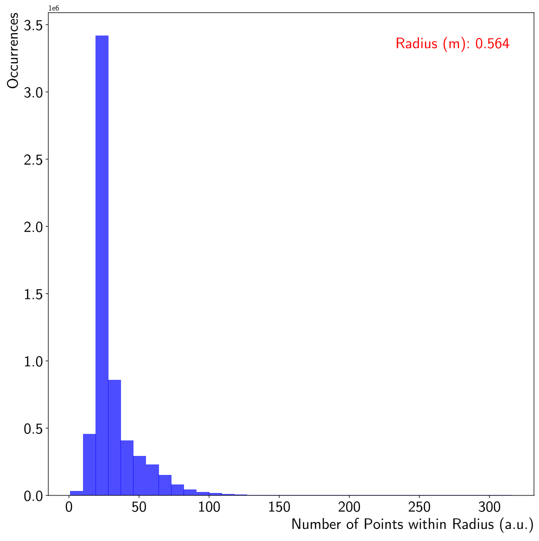
(a)



(b)

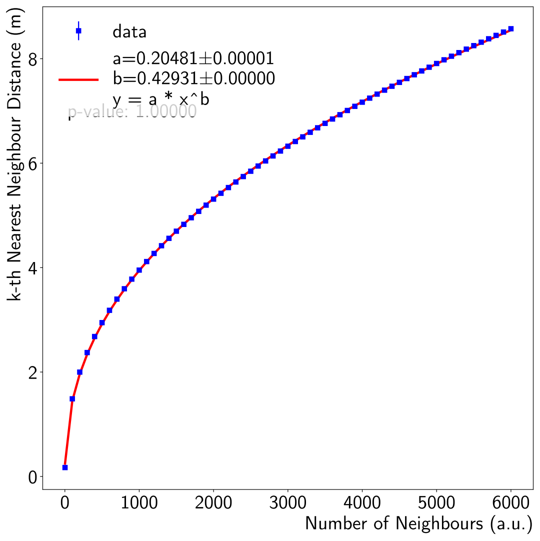


(c)

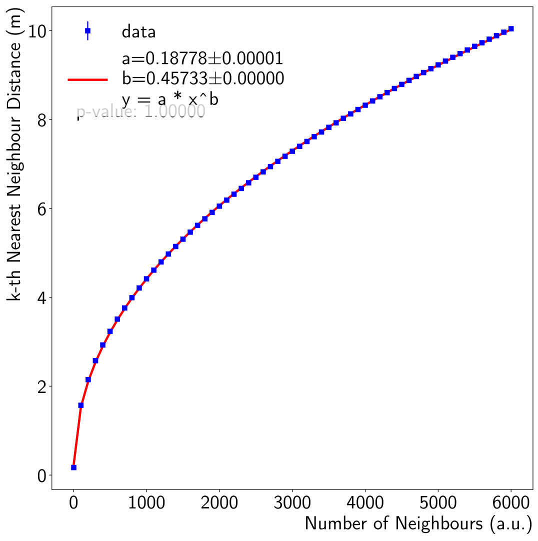


(d)

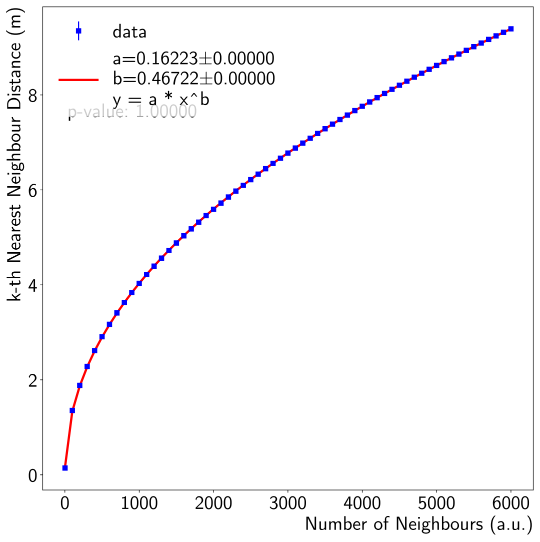
Figure 2.2: Histograms of the number of points inside a vertical cylinder with radius 0.564 m, so with base area around 1 m^2. a) b) c) and d) are the histograms of tiles a) b) c) d) in Fig. 2.1, respectively.



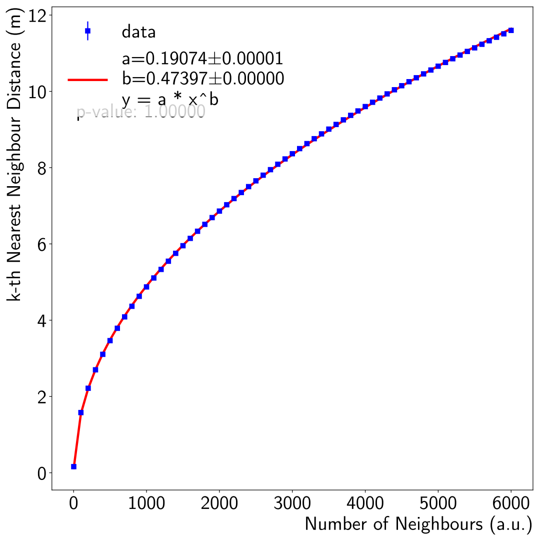
(a)



(b)

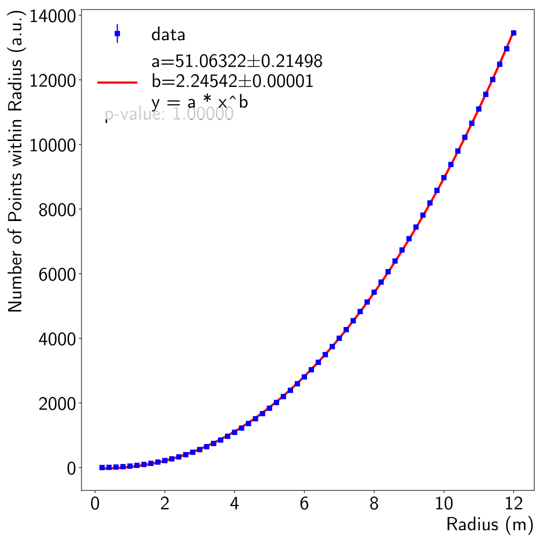


(c)

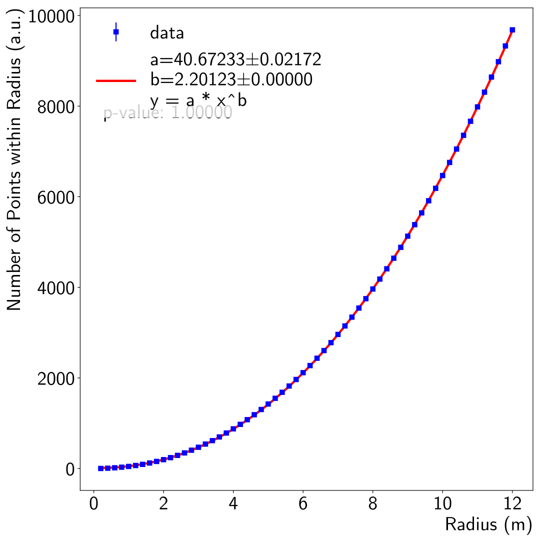


(d)

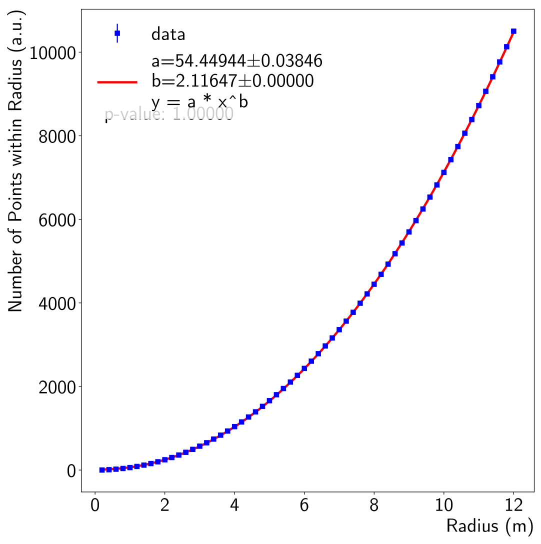
Figure 2.3: They show the correlation between the k (x axis) and the distance from the k-th point to the selected point (y axis). The data are blue squares and the error bars are not visible since they are too small. The red line is a power fit with parameters (a and b) found through Trust Region Reflective algorithm. We see that decreasing the number of trees (from a) to d)), the b parameter grows up, so the curve bends less and grows faster.



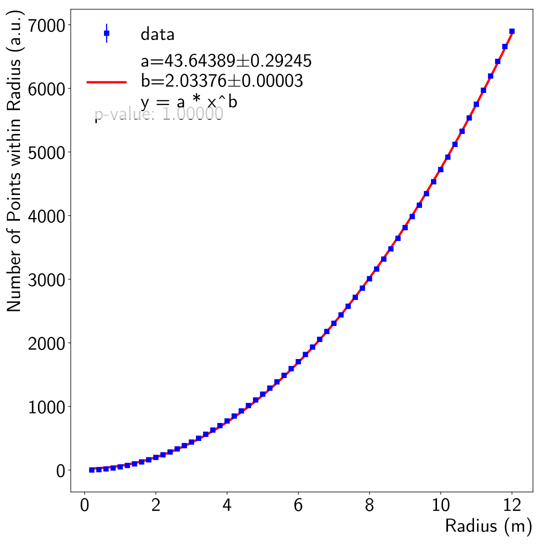
(a)



(b)



(c)



(d)

Figure 2.4: They show the correlation between the sphere radius (x axis) and the number of points inside the sphere (y axis). The data are blue squares and the error bars are not visible since they are too small. The red line is a power fit with parameters (a and b) found through Trust Region Reflective algorithm. We see that decreasing the number of trees (from a) to d)), the b parameter decreases, so the curve bends less and grows slower.

Table 2.1: Power exponents of densities fit. In the first column there are the indices of the analysed tiles: they are the same as Fig. 2.1, 2.3 and 2.4. In the second column there are the number of points in the point cloud. In the third column there are b parameter values of k-nearest distance density fits. In the fourth column there are b parameter values of number of points inside a sphere of radius r. In the fifth column there are the mean values of points inside a vertical cylinder with base area of $1\,\mathrm{m}^2$. I do not report errors because they are very small.

Tile index	Total points	b parameter		Density
		k-nearest	Radius	Pts in $1\,\mathrm{m}^2$
a	8889587	0.4293	2.245	43.508
b	7299399	0.4533	2.201	36.761
c	7087847	0.4672	2.116	34.054
d	6024754	0.4740	2.034	30.412

Chapter 3

Features

This chapter presents the features selected from LiDAR data. With the exception of RGB-NIR features, I will treat the LiDAR data as a point cloud, allowing for an examination of its geometric and physical characteristics.

In order to categorize points into buildings, cars, grass, rail, roads and trees, we need to extract features from the point cloud. In literature there are many observables related to LiDAR data and orthophotos, but we have chosen to ignore orthophotos for the time being, except for the colours extracted from them (see Sec. 1.3). In Sec. 3.1 I briefly summarize the LiDAR base variables that will be used, while in Sec. 3.2 the extracted features are reported. They both will be used to create machine learning models to automatically labelled the whole LiDAR Bologna dataset.

All the following figures show the feature values of point cloud related to the part of the city shown in Fig. 3.1. It shows trees, buildings, roads, cars and a courtyard with vegetation inside.

3.1 Provided features

As I wrote in Sec. 1.3 the LiDAR points were provided with many variables. The ones that I chose as features are:

- intensity: the fraction of emitted energy that returns to the detector;
- return number: which return signal is between all the returned ones generated by one emitted pulse;
- number of returns: how many return signals the emitted pulse generated;
- classification: whether the point is a ground or off-ground one.

Intensity provides information about the material of the hit object. A big fraction of energy is lost every time as can be seen in Fig. 3.2a-left, but its values are sufficiently

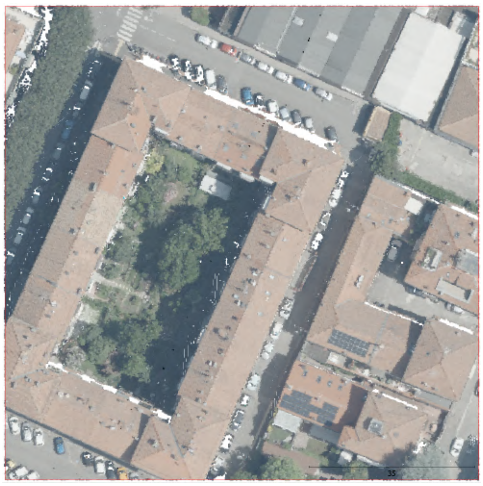


Figure 3.1: A picture of part of Bologna city used to show the feature characteristics. It describes an area of 100 m × 100 m in the suburbs between "via Bainsizza" and "via Ragazzi del '99". It contains trees, buildings, an artificial ground, cars and courtyard with vegetation inside.

spread to be useful. In Fig. 3.2a-middle we are able to distinguish elements and in Fig. 3.2a-right a threshold value of 1934 is set (intensity has not-normalized integer values).

Return number and number of returns are scalar values and their range is between one and eight, albeit CityMapper-2 is able to detect up to 15 returns per emitted beam. In Fig. 3.2b and 3.2c we see their plots (without thresholds). It is clear that they are useful for detecting trees, but they can get confuse with chimneys or other little objects.

Classification provides essential information about points (Fig. 3.2d). It is a boolean variable. Its only caveat will be the misunderstanding with bridges (see Sec. 4.2.3).

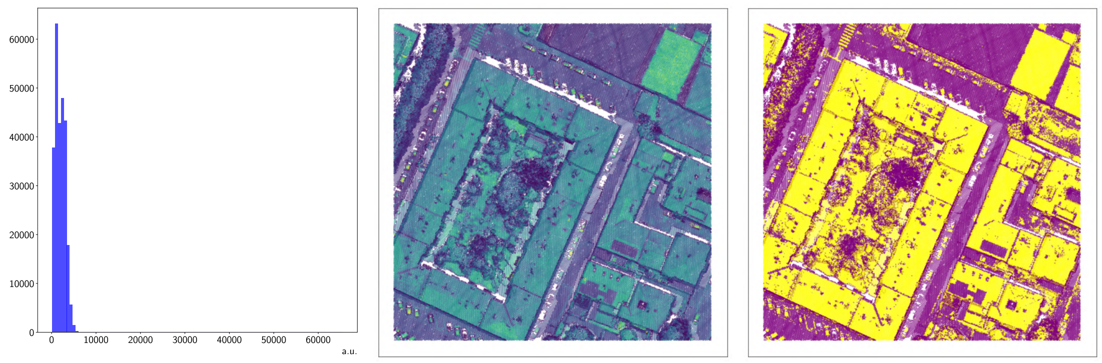
3.2 Extracted features

Different features are calculated from various point cloud information. Some are obtained from colours and near-infrared values, others from height; some are point qualities and others are local characteristics. I split them into groups according to their input data and I am going to describe each group in the following sections. I collected them in Tab. 3.1.

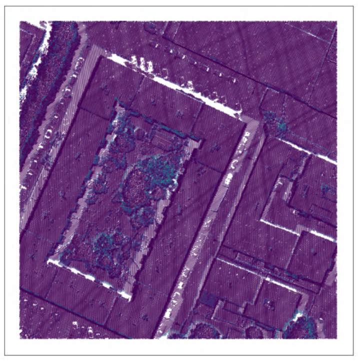
3.2.1 RGB-NIR features

All the observables of this group are punctual (depending only on information belonging to the analysed point). They use indices calculated from RGB and NIR.

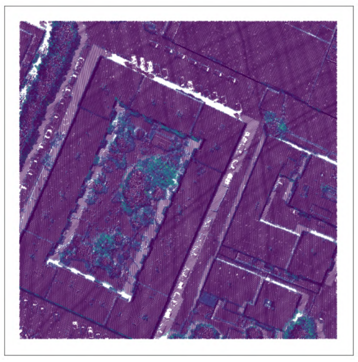
Normalized Difference Vegetation Index (NDVI) is a commonly used metric in LiDAR



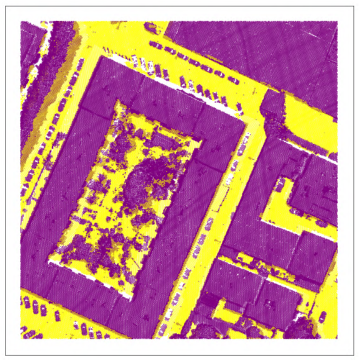
(a) Intensity



(b) Return number



(c) Number of returns



(d) Classification

Figure 3.2: Intensity (a), return number (b), number of returns (c) and classification (d) evaluated on a sub-tile of 100 m × 100 m (Fig. 3.1). For intensity I show histogram, plot and threshold plot (at 1934). It distinguishes between road, canopies and roofs quite well. Return number and number of returns highlights high complex surfaces (canopies edges and so on). Classification achieves very good resolution, identifying cars and bushes with high precision.

Table 3.1: Table of the calculated features. In the first column there is their group name, that is the same of their section. In the second column there is their name and in the third one a brief description of the group is provided.

Group	Feature	Description
RGB-NIR	NDVI NDWI SSI	Use RGB and NIR values to highlight vegetation and water
Height	N_h ΔZ MAD ΔZ_{fl}	Related to height of point over the ground and its fluctuations
Covariance	Planarity Sphericity Linearity Entropy	Calculate the covariance of points distribution inside neighbourhood and use eigenvalues to describe its shape
Plane	θ σ_θ^2	Find the local plane inside a neighbourhood and analyse the normal direction

studies [58]. Its formula is

$$NDVI = \frac{NIR - R}{NIR + R},$$

where R stands for red. It is typically used to identify vegetation since chlorophyll absorbs red light and reflects infrared waves [59] (see Fig. 3.3a). The right hump of the peak in Fig. 3.3a-left is related to vegetation. Indeed, applying a threshold at 0.2 we see how NDVI distinguishes between trees and other elements (Fig. 3.3a-right).

Normalized Difference Water Index (NDWI) is usually employed to detect water [60, 61], but our data does not detect water since it absorbs the LiDAR beam wavelength. However we decided to calculate it too, with the formula

$$NDWI = \frac{G - NIR}{G + NIR},$$

where G is for green, because it also detects vegetation and can help NDVI (see Fig. 3.3b). As with NDVI, we can use it to identify chlorophyll, as can be seen in Fig. 3.3b-left (left hump of the double peak) and in Fig. 3.3b-right (threshold at -0.1).

Spectral Shape Index (SSI) is related with NDWI: the latter is not able to distinguish between water and dark shadows, so SSI is generally evaluated to isolate them [61, 62]. In our case it is also used with vegetation, as it can be seen in Fig. 3.3c. It is not a normalized index, but a combination of red, green and blue values:

$$SSI = |R + B - 2G| \, .$$

It has no defined peak and therefore no good hand-threshold value. In Fig. 3.3c-right it is set at 2970.

3.2.2 Height features

This group features are related to z coordinate. One is related to the Digital Terrain Model, while two are defined thanks to analysed point neighbourhood. The last one depends on multiple echoes.

The z coordinate in LAS files is the altitude, namely elevation above the sea. I define N_h as the elevation above the ground [13]. To calculate it, I subtract from z the ground altitude, which is the DTM value at that coordinates. It is clearly useful to distinguish between ground and elevated points and between cars and trees or buildings, as shown in Fig. 3.4a. In Fig. 3.4a-right I set the threshold at 3.8 m and it clearly identifies trees and buildings.

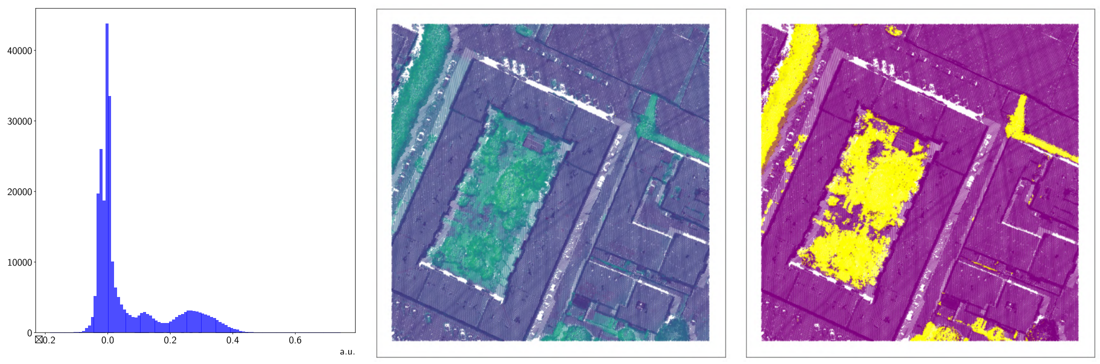
ΔZ is a local observable: it is the gap between the analysed point z and the z of the lowest point in the neighbourhood. It is useful to detect trees, where the canopy surface is not regular, but generally to distinguish between ground and off-ground points [63] (see Fig. 3.4b). The histogram (Fig. 3.4b-left) is complex and a threshold value is difficult to define: I chose 0.5 m but it detects both trees and little objects (cars, balconies and others).

Median Averaged Distance (MAD) is the median of the distance between N_h of the points in the neighbourhood of the analysed point and their median value:

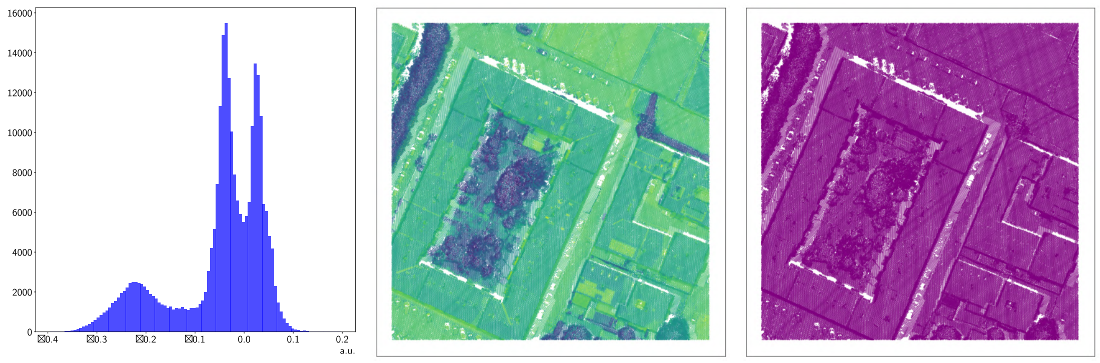
$$MAD = \underset{i}{median} \left[\left| N\_h_i - \underset{j}{median} \left(N\_h_j \right) \right| \right] \qquad i, j \in \text{neighbourhood} \, .$$

It highlights points whose neighbourhood has large variations of height (see Fig. 3.5a). In their study, Cai et al. [13] showed how this feature includes information on many other observables related to height value and variation. Setting threshold at 0.2 m, we remove the "flat" data (Fig. 3.5a-left) and we close off trees with a good (but not perfect) accuracy (Fig. 3.5a-right).

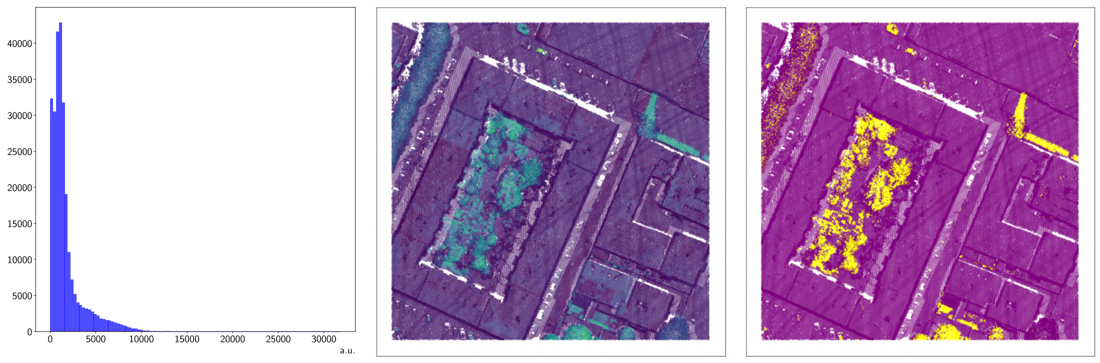
ΔZ_{fl} is related to multiple echoes. It is the z gap between the first and the last return of the same beam of the analysed point [63]. It assigns the same value per each point that is an echo of the same beam and it is equal to zero for the just-one return beam. It gets higher values for complex canopy, as shown in Fig. 3.5b. Of course the majority of points take on null value as shown in Fig. 3.5b-left and ΔZ_{fl} is not able to distinguish between tree canopies and chimneys or other little objects and edges (Fig. 3.5b-right, threshold 0.1 m).



(a) NDVI

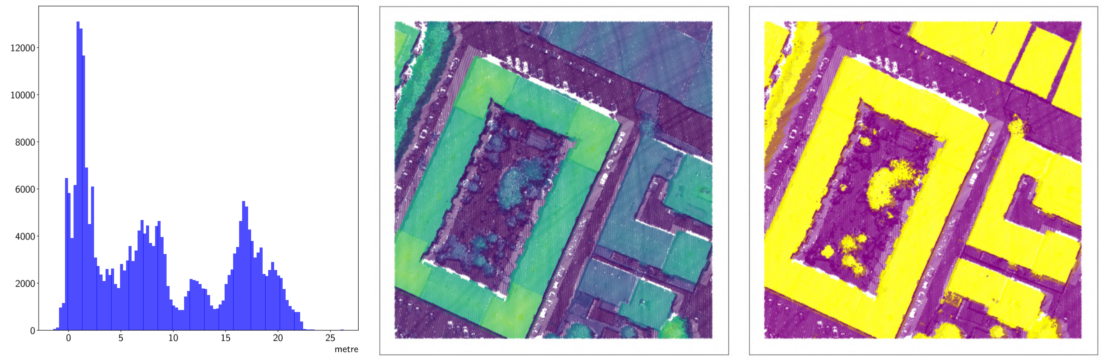


(b) NDWI

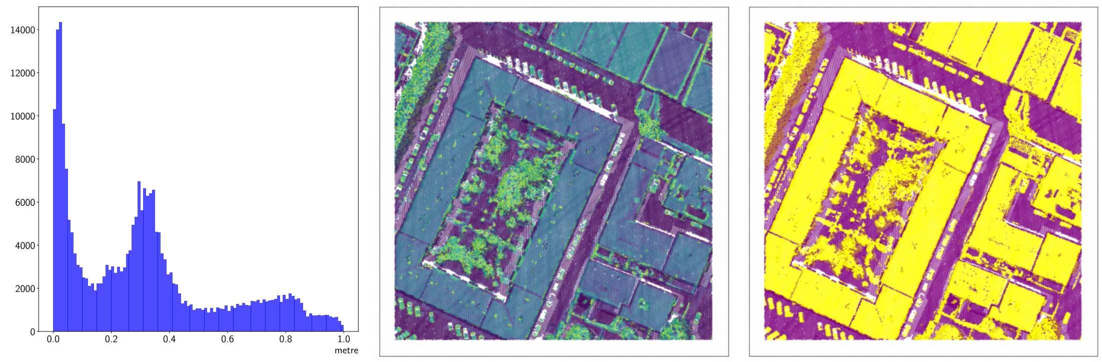


(c) SSI

Figure 3.3: NDVI (a), NDWI (b) and SSI (c) evaluated on a sub-tile of 100 m×100 m (Fig. 3.1). On the left there are histograms of the index occurrences in the sub-tiles: we notice that they are not trivial and there are some well defined peaks, especially in NDVI and NDWI histograms. In the middle there are scatter plots of their values: all three distinguished between high and low vegetation from the rest, but SSI is less spread. On the right there are scatter plots setting a threshold value: 0.2, -0.1 and 2970 respectively.

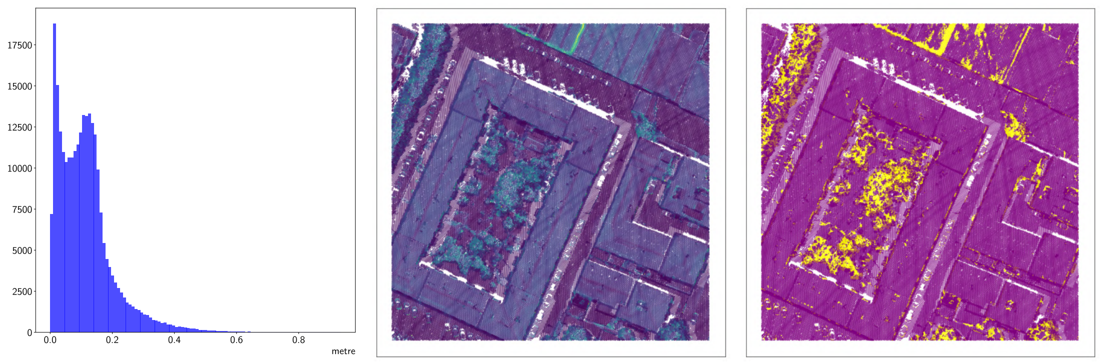


(a) N_h

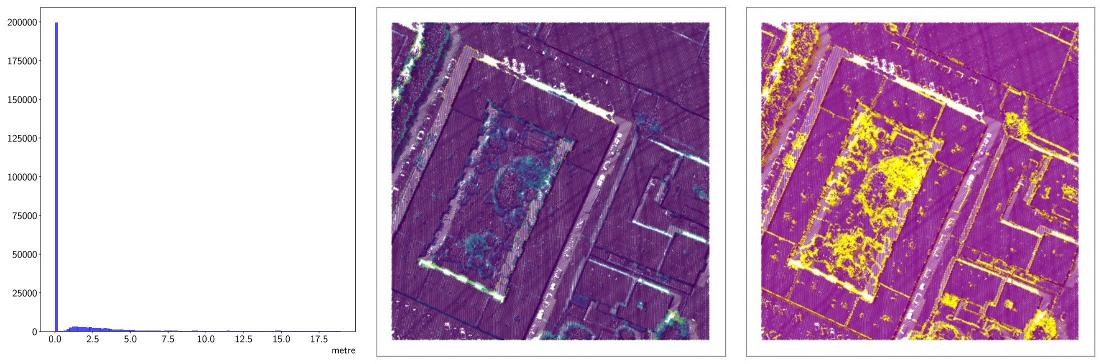


(b) ΔZ

Figure 3.4: N_h (a) and ΔZ (b) evaluated on a sub-tile of 100 m × 100 m (Fig. 3.1). On the left there are histograms of the index occurrences in the sub-tiles: we notice that they are not trivial and there are some well defined peaks. In the middle there are scatter plots of their values: N_h distinguishes between ground and off-ground; ΔZ between ground, buildings and trees. On the right there are scatter plots setting a threshold value: 3.8 m and 0.5 m respectively.



(a) MAD



(b) ΔZ_{fl}

Figure 3.5: MAD (a) and ΔZ_{fl} (b) evaluated on a sub-tile of $100\,\text{m} \times 100\,\text{m}$ (Fig. 3.1). On the left there are histograms of the index occurrences in the sub-tiles: we notice that MAD is not trivial, while ΔZ_{fl} has almost all value equal to zero. In the middle there are scatter plots of their values: MAD identifies trees canopy; ΔZ_{fl} highlights trees edges. On the right there are scatter plots setting a threshold value: $0.2\,\text{m}$ and $0.1\,\text{m}$ respectively.

3.2.3 Covariance matrix features

The following local features are related to the points distribution in a neighbourhood of the analysed point. Indeed, we can interpret the local point cloud as a distribution in three dimensions and therefore we can calculate its covariance matrix in relation to x, y and z. Per each point p_0 we find all near points and define a 3×3 matrix where cell C^{ij} is equal to

$$C^{ij} = \sum_k \left(p_k^i - \bar{p}^i\right)\left(p_k^j - \bar{p}^j\right) \qquad i, j = \{x, y, z\}\,,$$

where k is the index of the points inside the neighbourhood and $\bar{p}$ is the mean value of the points coordinates. Since the covariance matrix is hermitian (more precisely, it is symmetrical), it can be always diagonalized with real eigenvalues. The covariance matrix is positive semi-defined because

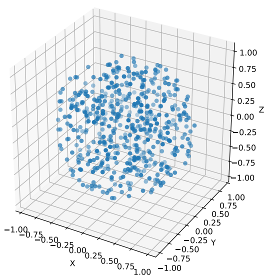
$$\mathbf{u}^T C \mathbf{u} = \mathbf{V}\,,$$

where C is the covariance matrix, $\mathbf{V}$ is the variance of a linear combination of the variables (x, y and z in our case) and $\mathbf{u}$ is a non-zero vector. It proves the positive semi-definition because variances are always greater or equal to zero. From a physical point of view it makes sense: its eigenvalues describe points scattering (variance) in the corresponding eigenvectors directions, so they must be at least no-negative. We can describe the local point cloud as being inside an ellipsoid with center in the analysed point, whose axes directions are covariance matrix eigenvectors and axes lengths are the corresponding eigenvalues.

From now on I will refer to the largest, the middle and the smallest eigenvalues as ℓ_1, ℓ_2 and ℓ_3, respectively. So ℓ_1 belongs to the direction wherein points are most scattered, i.e., the longest ellipsoid axis, while ℓ_3 belongs to the directions wherein points are less scattered, i.e., the shortest ellipsoid axis. They are very useful to identify near points distribution, since if they have a spherical distribution, eigenvalues are almost equals (Fig. 3.6a); if they have ellipsoidal distribution, eigenvalues differ (Fig. 3.6b); if points lies on a plane, ℓ_1 and ℓ_2 are much larger than ℓ_3 (Fig. 3.6c); else if points form a straight line, ℓ_1 is much larger than ℓ_2 and ℓ_3 (Fig. 3.6d). In Fig. 3.6 all the point clouds are synthetically generated.

In order to quantify these behaviours, I defined some features:

- Planarity identifies near points which lie on a plane;
- Sphericity takes bigger values as points follow a more isotropic distribution;
- Linearity detects points on a straight line.

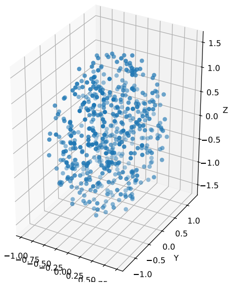


$\ell_1 = 0.21022$

$\ell_2 = 0.20156$

$\ell_3 = 0.18960$

(a) Spheric cloud

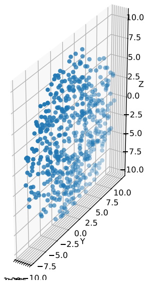


$\ell_1 = 0.50855$

$\ell_2 = 0.34094$

$\ell_3 = 0.18930$

(b) Elliptic cloud

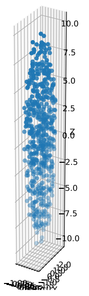


$\ell_1 = 22.436$

$\ell_2 = 18.829$

$\ell_3 = 0.21161$

(c) Plane cloud



$\ell_1 = 20.008$

$\ell_2 = 0.79605$

$\ell_3 = 0.20231$

(d) Straight line cloud

Figure 3.6: Synthetic point clouds and eigenvalues of their covariance matrices. a) in spheric clouds the eigenvalues are almost equal: points are isotropically distributed. b) when points loose isotropy, eigenvalues differ. c) if points lie on a plane, the third eigenvalues is almost nil compared to the other two. d) in a straight line just the first eigenvalues is not nil.

Their formulae are [64]:

$$\begin{aligned}
\text{planarity} &= \frac{\ell_2 - \ell_3}{\ell_1}, \\
\text{sphericity} &= \frac{\ell_3}{\ell_1}, \\
\text{linearity} &= \frac{\ell_1 - \ell_2}{\ell_1}.
\end{aligned} \tag{3.1}$$

Observing Eq. 3.1 we notice that their sum is always 1, so they are normalized. For this reason we can consider them as probabilities and calculate their Shannon entropy [65–67]:

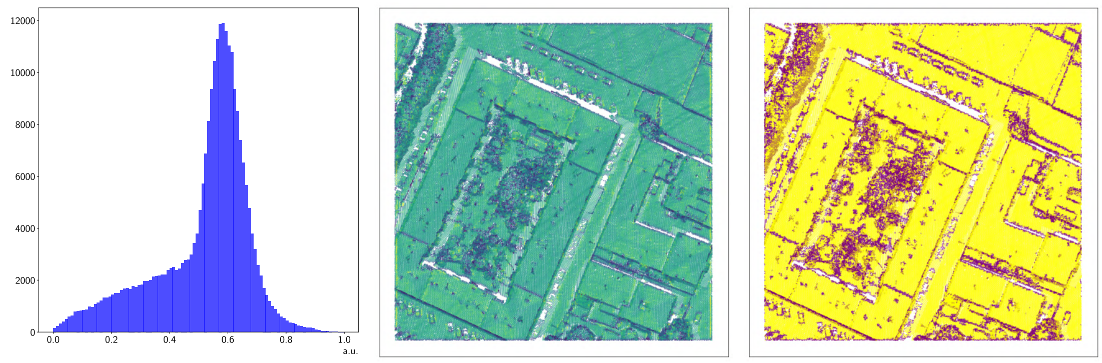
$$\text{entropy} = \mathit{plan} \log\left(\frac{1}{\mathit{plan}}\right) + \mathit{spher} \log\left(\frac{1}{\mathit{spher}}\right) + \mathit{lin} \log\left(\frac{1}{\mathit{lin}}\right).$$

Their behaviour is shown in Fig. 3.7 and 3.8. Planarity takes on high and homogeneous values on flat surfaces, as building roofs and streets, while on trees it takes on lower and irregular values (Fig. 3.7a). Its histogram has not separate peaks, so there is continuity, but shows a great spread of values. The peak is related to artificial flat surfaces, while the left-shoulder represents irregular elements (tree canopies). Setting a threshold is not trivial: with 0.4 it distinguishes between canopies and other elements, but it is not exact (Fig. 3.7a-right).

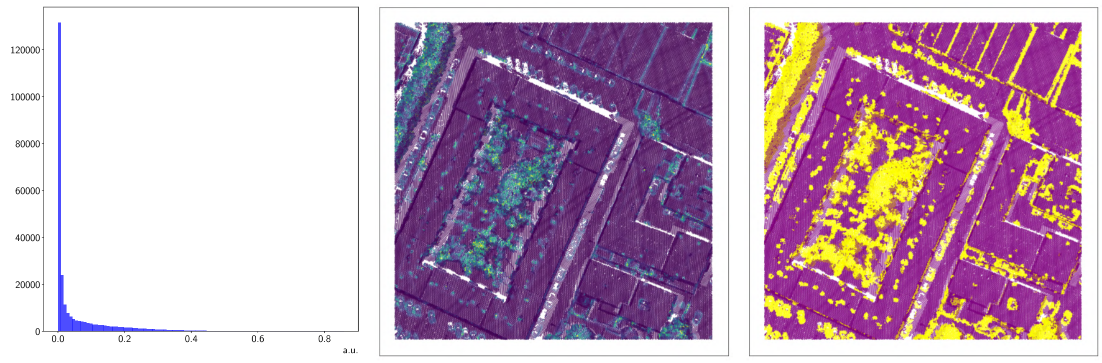
Sphericity usually takes on nil values, except for trees (Fig. 3.7b-left and -center) as its histogram reveals. It is because point clouds are (almost) never exactly a sphere, so sphericity does not assume all values between 0 and 1; however, if it is slightly bigger than zero, it is significant. Indeed, with threshold sets at 0.06 it highlights canopies, cars, chimneys and other small objects, as can be seen in Fig. 3.7b-right.

Linearity assumes almost all the possible values as it is shown in the histogram (Fig. 3.8a-left), but it takes on higher ones on buildings and trees edges, as shown in Fig. 3.8a-middle. Surfaces have values near the peak of ~ 0.4, while edges have higher linearity, but using this value as threshold we see that it does not achieve good results (Fig. 3.8a-right).

Entropy reveals point cloud irregularity, i.e., where points distribution and trend change suddenly, such as tree canopy, cars and chimneys (Fig. 3.8b). Its maximum value is $\log(3) \simeq 1.099$ and it happens when planarity, sphericity and linearity are equal (i.e. equiprobable): it means that the cloud is highly irregular and it could be described as a plane, a sphere and a straight line with the same degree of approximation. So irregular data can be related to trees or little structures such as chimneys and cars. When two indices are equiprobable and the third is zero, entropy is $\log(2) \simeq 0.693$. The peak is at ~ 0.7 and it could be due to zero sphericity and same probability for planarity and linearity, therefore on smooth objects. Using 0.8 as threshold in Fig. 3.8b-right, we split data between regular/big surfaces and irregular/small objects.

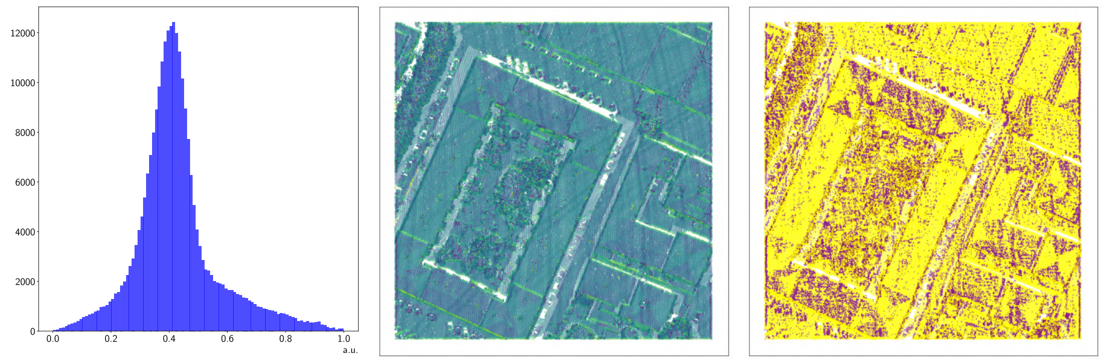


(a) Planarity

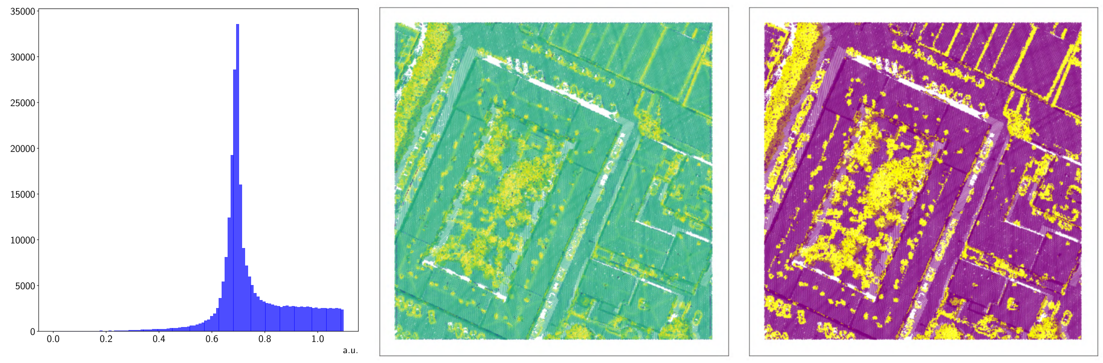


(b) Sphericity

Figure 3.7: Planarity (a) and sphericity (b) have been evaluated on a sub-tile of $100\,\mathrm{m} \times 100\,\mathrm{m}$ (Fig. 3.1). On the left there are histograms of the index occurrences in the sub-tiles: we notice that they only have one peak, but they have shoulders that reveal different behaviours in different real objects. The peak of planarity depicts flat surfaces and the left-shoulder represents the irregular points, while sphericity peak describes all points with the sphericity equal to zero. In the middle there are scatter plots of their values: they both are useful to distinguish between artificial/regular surface and natural elements. On the right there are scatter plots setting a threshold value: 0.4 and 0.06 respectively.



(a) Linearity



(b) Entropy

Figure 3.8: Linearity (a) and entropy (b) have been evaluated on a sub-tile of 100 m × 100 m (Fig. 3.1). On the left there are histograms of the indices occurrences in the sub-tiles: we notice that they only have one peak, but they have shoulders that reveal different behaviours in different real objects. Linearity highlights edges of buildings and canopy, while entropy highlights small or complex elements. In the middle there are scatter plots of their values: entropy is useful to distinguish between artificial/regular surface and natural elements, whereas linearity reveals edges. On the right there are scatter plots setting a threshold value: 0.4 and 0.8 respectively.

3.2.4 Plane inclination features

A very common feature about the neighbourhood is related to the inclination of the local plane compared to the vertical (height) [63, 68]. Once selected the neighbour points, a plane is defined that minimizes the distances between each point and itself. The distance is usually the norm $\mathbb{L}_{1.2}$ [69]. It is a good compromise between $\mathbb{L}_1$ and $\mathbb{L}_2$, since $\mathbb{L}_{1.2}$ ascribes a intermediate weight to outliers while $\mathbb{L}_1$ ascribes no weight and $\mathbb{L}_2$ blames too much. 1.2 is an heuristic and historic value and there is no reason why it should not be another. Then, the vector perpendicular to the plane is evaluated and we calculate the angle with the vertical, θ.

However, the neighbour points are too scatter to perform a good regression algorithm and the program returns the starting plane parameter (horizontal plane with vertical normal vector), so $\theta = 0°$ almost always.

So we decided to evaluate it in another way. As said in Sec. 3.2.3, the two eigenvectors related to the two higher eigenvalues of the covariance matrix define the two most scattered directions of the neighbourhood. Logically that a regression algorithm would reach the same result, since the plane which minimizes the distances with the points is orthogonal to the least scattered direction. Indeed, the two most scattered directions (so the eigenvectors related to the two higher eigenvalues) lie in the space defined by the plane. The advantage of using a covariance matrix rather than a regression method is that the former is deterministic (at least theoretically, it depends on the software) and it is already calculated for the other features. Furthermore, the regression method requires more time to be executed –and as said before, it does not work well with our point cloud.

To calculate θ we just need to calculate the arcocosine of the scalar product between the vertical versor $\mathbf{k} = (0,\ 0,\ 1)$ and the orthonormal eigenvector $\mathbf{v}_3$ related to the smallest eigenvalue ℓ_3:

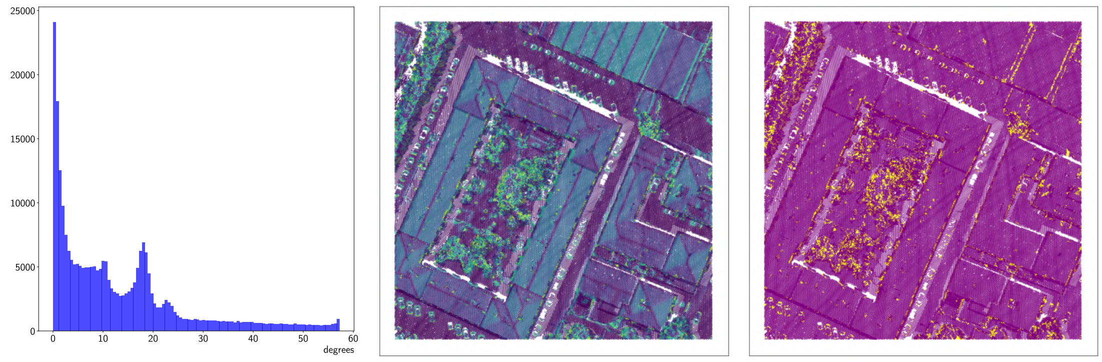
$$\theta = \arccos\left(\mathbf{v}_3 \cdot \mathbf{k}\right) .$$

The sign of the normal vector to a plane is not unique, so when we calculate the angle θ it can assume both positive and negative sign. Since the sign is not important, I take the absolute value of it.

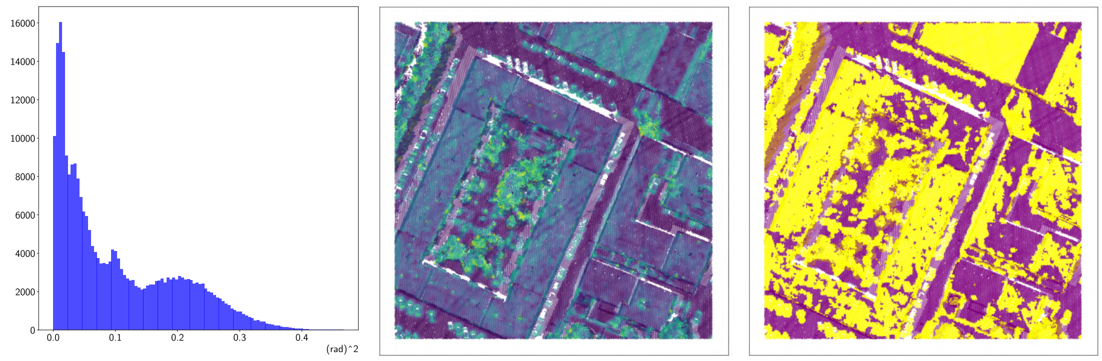
Once we have evaluated θ per each point, we can calculate another feature. We call σ_θ^2 (theta variance) the variance of the θ angles at points inside the neighbourhood. So per each point, we collect all the θ values of the points inside its neighbourhood (θ of the point i is calculated using the neighbourhood of point i) and calculate the variance (σ_θ^2 of point j is calculated using the θ values of points inside the neighbourhood of point j):

$$\sigma_{\theta j}^2 = \frac{\sum\limits_i \left(\theta_i - \hat{\theta}\right)^2}{N-1}, \qquad i = \{\textit{indices of points in } j \textit{ neighbourhood}\} .$$

In Fig. 3.9 I report the histograms, the scatter and the threshold plots about θ and σ_θ^2. θ values range from $0°$ to $57°$ as shown in Fig. 3.9a-left. Theoretically, θ must range



(a) Theta



(b) Theta_variance

Figure 3.9: Theta (a) and Theta_variance (b) have been evaluated on a sub-tile of 100 m $\times$ 100 m (Fig. 3.1). On the left there are histograms of the index occurrences in the sub-tiles: we notice that both have two peaks, θ at zero and one at $\sim 18°$, while σ_θ^2 at zero and at $\sim 0.2\text{rad}^2$. In the middle there are scatter plots of their values: they both detect not horizontal surfaces, but σ_θ^2 does it better. On the right there are the scatter plots are setting a threshold value: 26° and 0.13 rad^2 respectively.

to 90° but probably points are mainly distributed horizontally in each neighbourhood, so the third eigenvector is always nearer to the vertical direction than horizontal ones. There is a peak near 18° due to the slope of roofs of the building with inner courtyard. In Fig. 3.9a-middle θ takes zero value on the street, low values on roofs following their slope and many values on canopies. With a threshold of 26° only a few points related to canopies, cars and chimneys are highlighted.

The histogram of σ_θ^2 (Fig. 3.9b-left) has a spread peak at 0.22 rad^2. Qualitatively, Fig. 3.9b-middle allows distinguishing different elements quite well, but introducing a threshold reveals that this parameter is not sufficient to label points well.

3.3 Neighbourhood and Outliers

Many features are related to the characteristics of neighbouring points. Among the selected features, those related to the neighbourhood are ΔZ, MAD, ℓ_1, ℓ_2, ℓ_3, planarity, sphericity, linearity, entropy, θ and σ_θ^2. In literature the surround can differ in shape and radius. The two most common shapes are spheric and cylinder along the z-axis, while the radius varies between 1 m and 15 m.

The default spherical surround [13] is sometimes replaced by the cylindrical [63] one because of the characteristics of point cloud. Since the points are usually distributed on a plane except where multiple returns happen, the cylinder highlights the z spread and should include all the returns of beams. However it is more sensible to outliers.

Outlier points measured elevation is unreasonably more or less from their neighbouring points. Outliers are mainly measurements that do not conform the local surface geometry and do not belong to the topography of the interested area [70]. The outliers with too high elevation values are usually named “positive outliers”, while the others with too low elevation values are called “negative outliers”. The outliers can be caused by different sources. The positive ones have resulted from suspended objects such as birds or planes that reflect laser beams at high altitudes. It is believed that negative outliers depend on the reflection of beams among the glasses of buildings before they are detected, just like the multi-path effect of GPS. These specular reflections result in a longer travel time of the laser beams, so a lower elevation is calculated [71].

As said above, the radius varies widely. Behley et al. [72] emphasized the importance of selecting an appropriate radius to enhance object detection accuracy in an urban environment, suggesting that a radius of 1-2m is often affected for distinguishing between closely spaced objects. Of course radius depends on point density too: Chehata et al. [63] used a 15 m radius for their point clouds with a density of 2.5 pt/m^2.

I tried spherical and cylindrical shapes and I noticed that some features are more defined with spherical surround (usually covariance matrix features and ΔZ), others with cylindrical ones (MAD especially). Since computing both is very time demanding and spherical is more canonical and more features are better with this shape, I selected spherical neighbourhood.

To find a good radius, I calculated features setting radii equal to 0.5 m, 1 m, 1.5 m, 2 m and 3 m and all of them are better defined with radii smaller or equal to 1 m.

In App. B there is an in-depth analysis and figures related to different shapes and radii.

Chapter 4

Classifier

In this chapter I describe how I have created the random forest model to automatically classify LiDAR points into buildings, cars, grass, rail, roads and trees. Firstly I clarify how I labelled data, then why I chose random forest as classifier and how I trained and tested it.

4.1 Labelling

The first issue in creating an automatic classifier was labelling LiDAR points to train and test the machine learning model. Hand-labelling each point is impossible both for the amount of time required and for a matter of visual resolution. Indeed, there are at least $20\,\mathrm{pt/m^2}$, so if we wanted to distinguish each point from the others, we would not see the context and we would not know what we are watching: either we distinguish each point, or we reduce magnification and we could see point cloud as a picture.

We were searching for a semi-automatic technique to label data without introducing bias. In fact, if we had used an already trained model or an unsupervised model (as the K-mean algorithm), our model training and testing datasets would have been affected by the errors made by the labelling algorithm.

In order to try to overcome this issue, I followed the pipeline shown in Fig. 4.1. In this pipeline I used *segment-lidar* [73], a Python module that segments off-ground LiDAR data through their RGB values. It works using *Segment-Geospatial* (SAMGeo) [74] to benefit from *Segment Anything Mode* (SAM) [75] by Meta AI. In Sec. 4.1.1 I report a brief description of these three packages.

As previously mentioned, this segmentation procedure is applicable only to off-ground points. While *segment-lidar* offers a *Cloud Simulation Filter* to identify off-ground points (Fig. 4.2c), we opted to use the pre-existing classification provided by CGR (Fig.. 4.2b). This decision was made because we preferred to rely on classification values from a company specializing in this field, likely using purpose-built software. Additionally, this approach avoids introducing additional noise to the data.

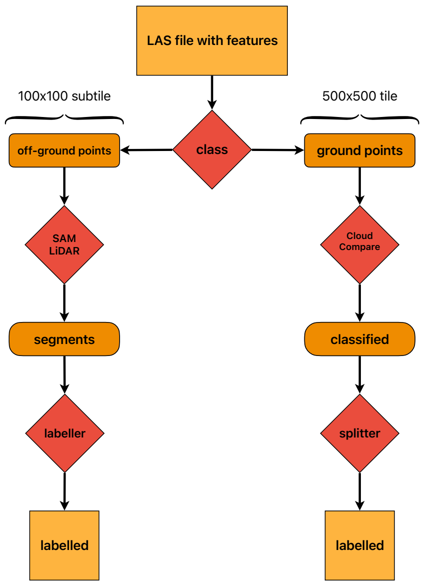


Figure 4.1: It is the diagram of the followed pipeline to create labelled LAS files starting from LAS files just with features.

Since SAM works on the picture generated by *segment-lidar*, the wide of the LiDAR point cloud is important. I decided to split each tile into 25 subtiles of 100 m × 100 m in order to better segment objects without rendering the picture too grainy (an example is shown in Fig. 4.3a). In App. C I illustrate how I selected this subtile size in order to allow SAM to better distinguish between different environment elements and to detect small objects too (e.g., cars).

I calculated the features before splitting the file, as many of these features are dependent on the neighborhood of each point. This approach ensures that only points near the edges of the entire tile are calculated without an isotropic surrounding, rather than all points on the edges of the subtiles.

As shown in Fig. 4.1, I used the classification feature to create two LAS files: one with only off-ground points (Fig. 4.3b), and the other with only ground points. Off-ground file is segmented by *segment-lidar* (Fig. 4.3c), then I hand-labelled the segments where all the points belong to the same class (tree, building and so on) (Fig. 4.3d). The unsegmented or not well segmented points were not labelled.

On ground points, a more hand-made work is necessary. To select and label points I used *CloudCompare* [76], a free tool that allows to view and manage LiDAR data and other point cloud data. Unfortunately, *CloudCompare* allows only to manage classification values of selected points; so, after the file has been labelled with labels saved in the classification variable, I executed a Python script to store the labels as a new variable and restore the original classification values (Fig. 4.3f). For this work I used LAS files with

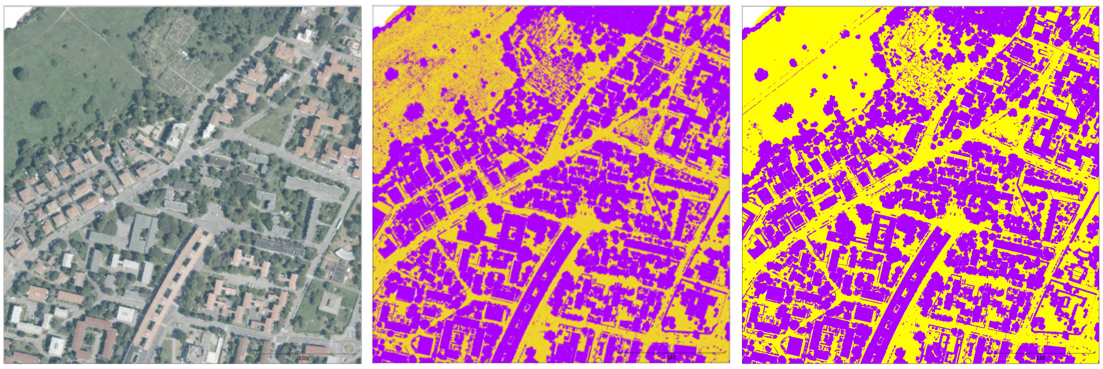

(a) RGB values **(b)** Classification values **(c)** CSF values

Figure 4.2: a) shows the RGB values of the LiDAR point cloud acquired near "via Corolano Vighi" and "via Leone Tolstoi". b) shows the classification values calculated by CGR, while c) shows the results of *Cloud Simulation Filter*, both on the same tile. In yellow there are the ground and in purple there are the off-ground points.

only ground points of the whole tile (Fig. 4.3e) –and not subtiles, since *CloudCompare* allows to work on the scale in real time.

4.1.1 Segmentation algorithms

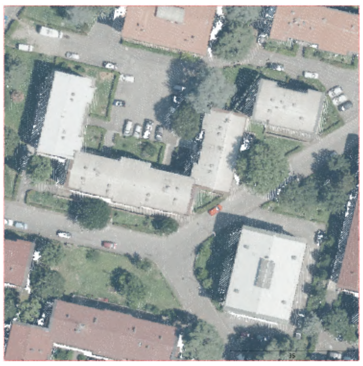
Here I report an overview of the three packages mentioned above.

Segment Anything Model SAM is a foundational computer vision model designed for versatile image segmentation tasks. A foundation model, also known as large AI model, is a machine learning or deep learning model that is trained on broad data such that it can be applied across a wide range of use cases [77]. The Stanford Institute for Human-Centered Artificial Intelligence's (HAI) Center for Research on Foundation Models (CRFM) created and popularized the term [78] describing *any model that is trained on broad data (generally using self-supervision at scale) that can be adapted (e.g., fine-tuning) to a wide range of downstream tasks* [79].

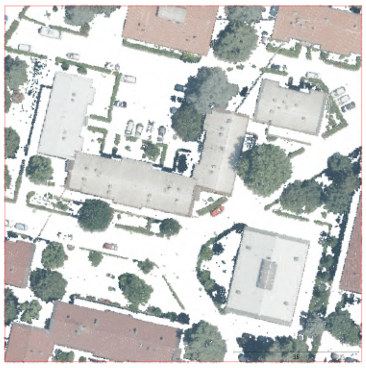
SAM architecture includes three main components:

1. a powerful image encoder computes an image embedding;
2. a prompt encoder embeds prompts;
3. the two information sources are combined in a lightweight mask decoder that predicts segmentation masks.

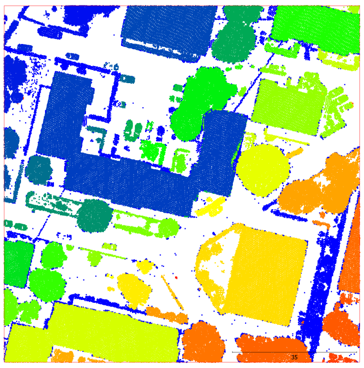
By separating SAM into an image encoder and a fast prompt encoder / mask decoder, the same image embedding can be reused with different prompts, amortizing the most time and resource-expensive process (i.e., image embedding).



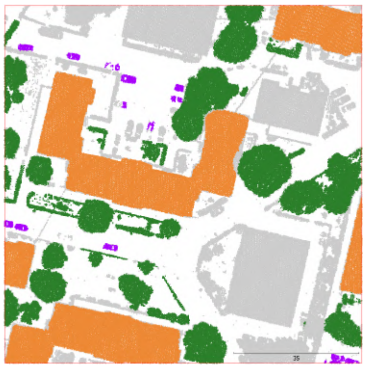
(a) Subtile RGB



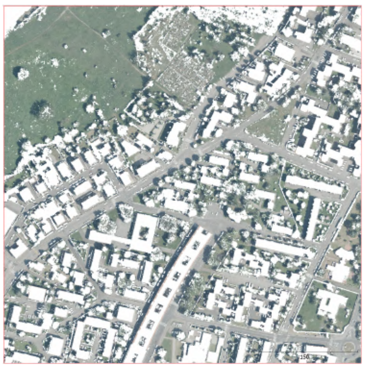
(b) Subtile off-ground only



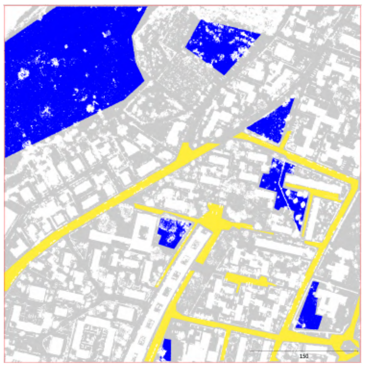
(c) Subtile segmented



(d) Subtile labelled



(e) Tile ground only



(f) Tile labelled

Figure 4.3: The pictures show: a) the RGB values of 100 m × 100 m subtile, b) only its off-ground points, c) the subtile segmented by SAM and d) the subtile hand-labelled. e) shows only the ground points of the whole tile shown in Fig. 4.2a, while f) shows the tile hand-labelled. The label colours are: blue for natural ground; yellow for artificial ground; purple for vehicles; orange for buildings; green for trees; light-grey for unlabelled.

SAM leverages a massive dataset of over one billion masks across eleven million images [75]. The huge amount of training data allows to overcome the principal task – segmenting images given a prompt. The final goal is to produce a broadly capable model that can adapt to many existing and new segmentation tasks via prompt engineering. This capability is a form of task generalization [80]. In this way SAM can perform a new, different task at inference time by acting as a component in a larger system. For example, to perform instance segmentation (as in our case), a promptable segmentation model is combined with an existing object detector.

In the paper about SAM, Kirillov et al. [75] tested four zero-shot transfer experiments [81]: they evaluated SAM on datasets and tasks that were not seen during training. The SAM image encoder was a MAE [82] pre-trained ViT-H [83]. *Vision Transporters* (ViTs) are a deep learning model that have shown promise in image segmentation tasks [84]. Unlike convolutional neural networks, ViTs employ self-attention mechanisms that allow them to model long-range dependencies and global context within images [85, 86]. This approach has demonstrated competitive performance in various computer vision tasks, including remote sensing image segmentation [87].

One of the zero-shot transport experiments was instance segmentation. SAM achieved reasonably close results, though certainly behind the benchmark model ViTDet [88], but SAM masks are often qualitatively better. The authors hypothesize that VitDet learns idiosyncrasies and biases from the datasets, while SAM can not do it because it is not trained on them. In any case, the pro of a foundation model is the unnecessary long training; no one expects it to outperform custom models.

Osco et al. [84] proved that SAM can be used with one-shot approach too. This technique concerns feeding SAM with a single example (or “shot”) of a new class in order to greatly improve the accuracy of inferences. Two of the best-known one-shot methods for SAM are PerSAM and PerSAM-F [89]. Given a single image with a reference mask, PerSAM localizes the target concept using a location prior to an initial estimate of where the object of interest is likely to be. The second method freezes SAM parameters and introduces two learnable weights for multi-scale masks. This one-shot fine-tuning variant requires training only two parameters and can be done in as little as ten seconds to enhance performance.

To overview the main features of SAM, we can summarize them as follows:

- interactive segmentation: users can provide prompts (e.g., points or bounding boxes) to guide the segmentation process, enhancing its usability in real-time applications [90];

- zero-shot learning: SAM can generalize to new objects without additional training, making it adaptable across various domains, including remote sensing [91, 92];

- one-shot learning: if just one example is provided to SAM, the trained model outperforms zero-shot learning at inference time [84];

- wide applicability: SAM has been successfully applied in diverse fields, such as automatic crater detection on planetary surfaces and precision agriculture, showcasing its versatility [93].

Segment Geospatial SAMGeo is an open-source Python package designed to simplify the process of segmenting geospatial data with SAM. The package leverages popular Python libraries to provide a straightforward interface for users to segment remote sensing images and to export the results in various formats, including vector and raster data. The segmentation can be run automatically, interactively in a graphical user interface (GUI), or by text prompts built upon Grounding DINO [94]. However, it is worth noting that the text prompt approach has its limitations, which may require parameter fine-tuning. SAMGeo aims to fill the gap in the Python ecosystem by providing a user-friendly, efficient and flexible geospatial segmentation tool without the need for training deep learning models [74].

Segment Lidar *Segment-lidar* projects off-ground LiDAR points into a plane and it creates a picture using the RGB points features. The software allows to choose the projecting plane and we decided to project points on the $x-y$ plane, in order to prevent overlapping of elevated-near structures.

The software allows to set the resolution of the image. Of course, it is a very important parameter because it sets what and how SAM will "see" in the picture and this affects the creation of segments. I chose a resolution of 0.25 which means that it creates an image of 250×250 pixels. I explain why I made this choice in App. C.

The generated image is comparable to a remote sensing image (airborne photo). It is the input that SAMGeo needs, so it uses SAM to perform instance segmentation over the generated image. We used the most detailed model type: ViT-H (rather than ViT-L or ViT-B). Since SAM is a prompt-segmentation model, we could provide a text prompt (e.g., "trees" or "cars"), but we noticed that it does not work correctly and the algorithm is not able to use the prompt to increase the quality of the masks. It is the same problem observed in SAMGeo.

In any case, when SAM returns masks over the image, *segment-lidar* recreates the $3D$ point cloud adding a variable to each point. In that variable it stores the segment ID, so we have the original point cloud segmented.

4.2 Machine Learning

4.2.1 Random Forest

A decision tree is a widely utilized data mining technique that serves as a model for classification and prediction. It structures data into a tree-like format, consisting of a root node, internal nodes, and leaf nodes, where each node represents a decision point based on specific features of the data. This non-parametric method is adept at handling large datasets without requiring complex parametric assumptions [95, 96].

A random forest (RF) model is an ensemble learning technique that constructs multiple decision trees during training and outputs the mode of their predictions for classification or the mean prediction for regression tasks. This method is particularly effective for handling complex datasets with numerous features and interactions. The key characteristics of the RF model are:

- ensemble learning: RFs aggregate predictions from multiple decision trees, enhancing accuracy and robustness against overfitting [97];
- feature importance: they can evaluate the importance of various input features, which aids in understanding the underlying data structure [98];
- versatility: RFs are applicable in diverse fields, such as predicting evapotranspiration in arid regions [99] and forecasting COVID-19 cases using demographic data [100];
- reduced overfitting: by averaging multiple trees, random forests mitigate the risk of overfitting common in single decision trees [101];
- handling missing data: they can maintain accuracy even with incomplete datasets, making them suitable for real-world applications[100].

While RFs are powerful, they may not always outperform simpler models in scenarios with limited data or when interpretability is crucial.

RFs are increasingly utilized for classifying LiDAR point clouds due to their robustness, efficiency, and ability to handle high-dimensional data. This machine learning technique excels in various applications, including urban object detection and vegetation classification. In their article, Ko et al. [102] show how geometrical features are taken advantage of RF models that achieve very high accuracy (greater than 91.2%). They are also able to manage the curse of dimensionality, namely we can provide them with many features without losing performance [103]. RFs identify which features are the most important and increase their weight: the automatic features selection is very useful in our work because we are interested in finding the most informative ones.

4.2.2 Results

The labelled data have been collected and merged from tiles and subtiles in *csv* files. There was a *csv* per each category: buildings, cars, grass, rails, roads and trees. Since cars and rails were smaller datasets I needed to subsample the other ones in order to

Table 4.1: The table reports the number of labelled points. In the second column there is the number of all labelled points per class. In the third column there is the number of labelled points used in the training dataset. In the fourth one there is the number of labelled points used in the testing dataset.

Class	Total	Training	Testing
Buildings	8919236	500000	125000
Cars	325040	260032	65008
Grass	6915368	500000	125000
Rails	525403	420322	105081
Roads	6918697	500000	125000
Trees	12942709	500000	125000
Sum	36546453	2680354	670085

avoid wrong behaviours by the model. In Tab. 4.1 I report the quantities of labelled points in training and testing datasets.

To create the RF model I used the *Yggdrasil Decision Forests* [104] which creates a RF composed by 300 decision trees.

The first model trained was used to select the neighbourhood radius of features (see Sec. 3.2 and 3.3 and App. B). I used as input the training dataset where all the features related to the neighbourhood (height, covariance and plane features of Tab. 3.1) had been calculated with radius equal to 0.5 m and 1.0 m. We can not calculate both every time because it is very time and memory-demanding, but I did it to select the most useful radius. As can be seen in the first three columns of Tab. 4.2, where the second and the third column show the weights – given by the RF model– of the features calculated with radius equal to 0.5 m and 1.0 m respectively, the features calculated with wider neighbourhood are usually more important than the other ones.

In the training of the final model I just provided the features evaluated in a sphere of 1.0 m of radius –plus the point features. Features weights are reported in the last column of Tab. 4.2. The confusion matrix [105] is shown in Tab. 4.3, while accuracies [106], precision, recalls [107] and F1-score (or Dice coefficient) [108] metrics are in Tab. 4.4.

In Fig. 4.4 and 4.5 there are six tiles –not used in the training and testing datasets and never hand-labelled or segmented– with their predicted labels.

4.2.3 Discussion

Generally the model achieves very good results, as can be seen in Tab. 4.3 and 4.4. The total accuracy is up to 95%, so it is an excellent result since it is our first model to

Table 4.2: The table reports weights of features in two RF models. The first column reports the features. In the second and third columns there are the weights of a RF trained using features calculated inside two neighbourhoods with radii equal to 0.5 m and 1.0 m, respectively. The fourth column reports the weights in the RF with only neighbourhood features at just 1.0 m radius, so the final model. Weights are provided by the RF model: higher weight means a more important feature to categorize points.

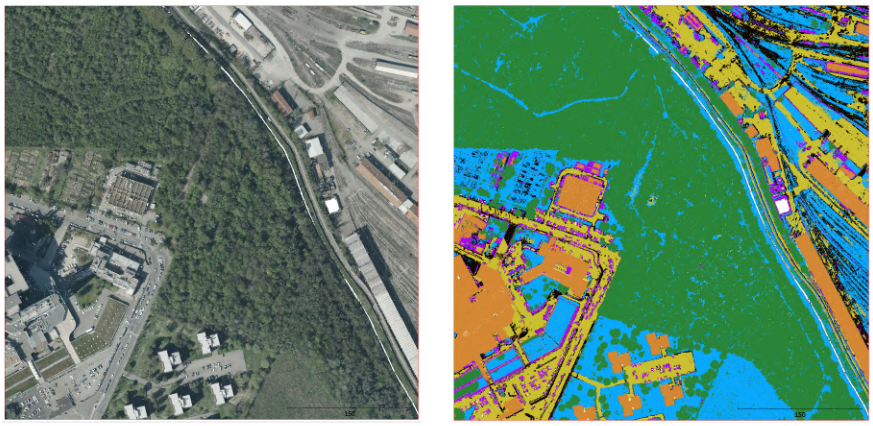
Feature	Both radius		Final model
	0.5 m	1.0 m	1.0 m
ΔZ	0.000404	0.005696	0.007417
MAD	0.000707	0.003768	0.007527
ℓ_1	0.000124	0.001151	0.001116
ℓ_2	0.001209	0.004273	0.005187
ℓ_3	0.000084	0.015486	0.019792
Planarity	0.000763	0.000304	0.000239
Sphericity	0.000078	0.003074	0.007412
Linearity	0.000515	0.000298	0.000230
Entropy	0.000710	0.000122	0.000084
θ	0.000310	0.000813	0.001081
σ_θ^2	0.000649	0.011548	0.015841
Intensity	0.039361		0.050976
Return number	0.000682		0.000785
Number of returns	0.000733		0.001203
Classification	0.398303		0.397365
NDVI	0.146232		0.151933
NDWI	0.181963		0.222114
SSI	0.005965		0.007657
N_h	0.119607		0.127376
ΔZ_{fl}	0.002112		0.002633

Table 4.3: Confusion matrix of the final model. In the rows there are the hand labels, in the columns there are the values predicted by the model. In each cell of the table there is the number of points hand-labelled the rows, which are classified as column by the model. A perfect model would create a diagonal matrix.

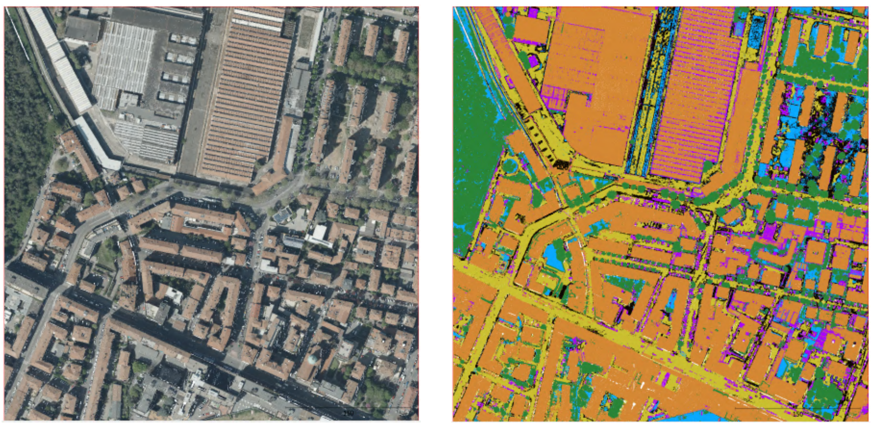
Prediction → Label ↓	Buildings	Cars	Grass	Rail	Roads	Trees
Buildings	117827	5953	223	0	256	740
Cars	1942	61921	175	0	286	684
Grass	28	116	123737	608	136	374
Rails	0	0	1744	100306	3031	0
Roads	763	912	1114	12862	109322	26
Trees	352	560	269	0	1	123817

Table 4.4: Table reports accuracy, precision, recall and Dice coefficient per each class. The model is the final RF. In the last row there is the total accuracy of the model and the average values of the other metrics.

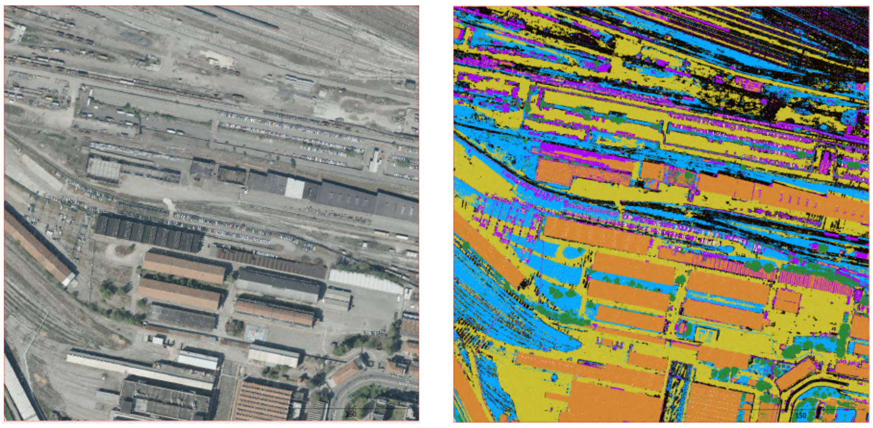
Class	Accuracy	Precision	Recall	F1
Buildings	0.943	0.974	0.943	0.958
Cars	0.953	0.891	0.953	0.921
Grass	0.990	0.972	0.990	0.981
Rails	0.955	0.882	0.955	0.917
Roads	0.875	0.967	0.875	0.919
Trees	0.991	0.985	0.991	0.988
Total	0.951	0.945	0.951	0.947



(a)

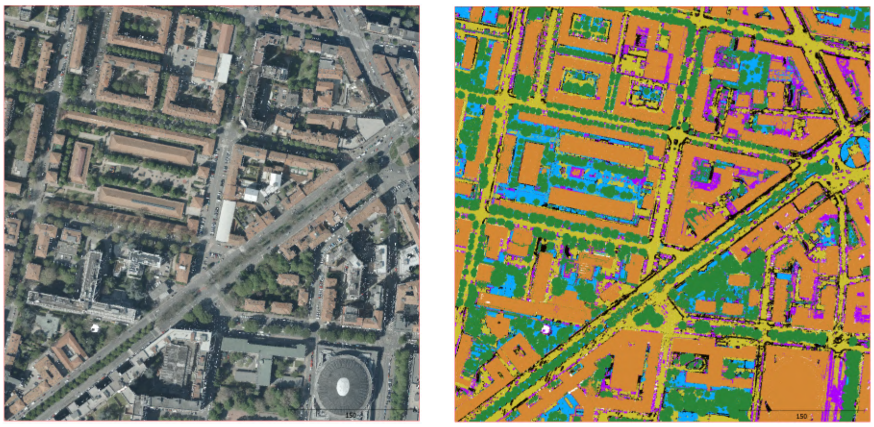


(b)

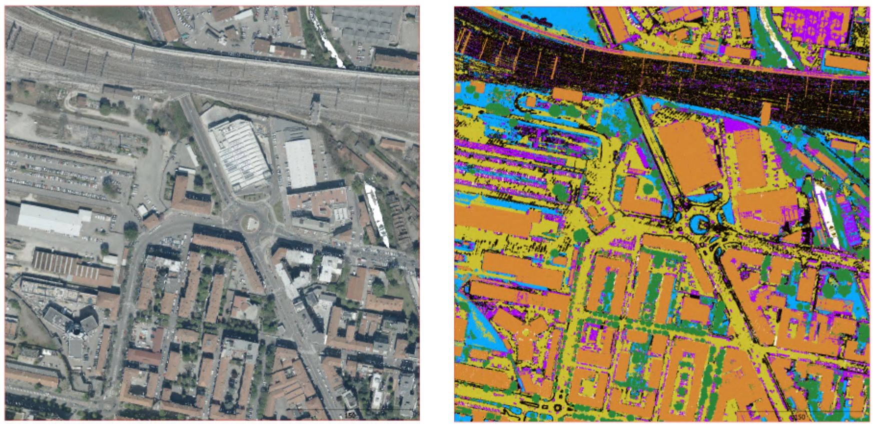


(c)

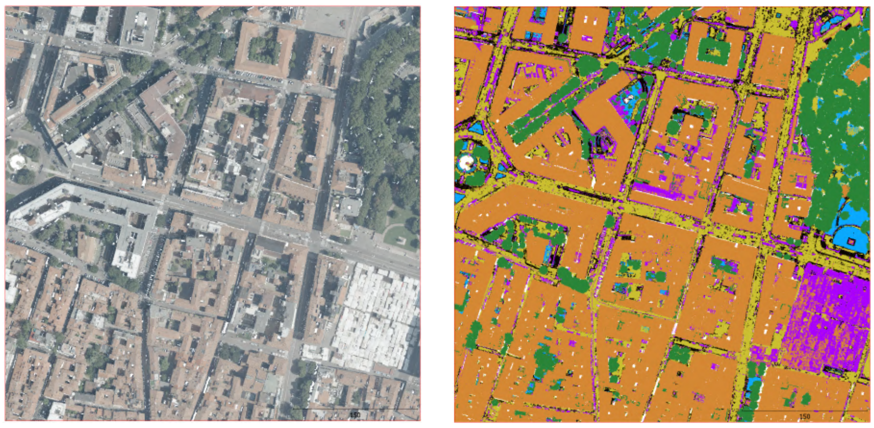
Figure 4.4: On the left the point clouds show RGB values; on the right they show predicted labels. Buildings are orange, cars are purple, grass is light blue, rail is black, roads are yellow, trees are green.



(a)



(b)



(c)

Figure 4.5: On the left the point clouds show RGB values; on the right they show predicted labels. Buildings are orange, cars are purple, grass is light blue, rail is black, roads are yellow, trees are green.

categorize Bologna LiDAR data. Furthermore, in the edges there is no visible quality reduction, even if the neighbourhoods are incomplete. However, there are of most certainly some classification mistakes.

Analysing the confusion matrix we notice that buildings are sometimes categorized as cars and vice versa. As shown in Fig. 4.4b, sometimes metallic or small elements on the top of roofs are predicted as cars, while buses are often categorized as buildings because they are higher than the other cars. Exemplary is Fig. 4.5c, where the market of “Piazza VIII Agosto” is in part a building and in part a car. I want to specify that in the training and testing datasets there is nothing similar to a market.

Rails are often confused with roads and vice versa and it is comprehensible, whereas most of the time rails are confused with grass (more than with roads) but grass is not confused with rails as frequently. The percentage of rails classified as grass is 1.40%, the vice versa is 0.49%. Probably it is a hand-labelling problem: it is common that on railways grass grows, so it creates confusion inside the model. In Fig. 4.4a all the grass is well classified and in Fig. 4.5b, where railways seem clean, there are a few points classified as grass. In Fig. 4.4c it is different: since they are dead-end tracks there is low vegetation in the ballast and they are categorized as grass.

There is confusion in roads with all the other classes except trees. I have already written about roads and rail (see Fig. 4.5a) and we can imagine why there is confusion with grass (for example on country roads or dirty roads). Probably roads are sometimes classified as cars or buildings because of bridges, where the classification features categorize car points as off-ground.

The tree category is very well classified and reaches the highest values in all the metrics. However some cars are categorized as trees. It could be due to vehicles parked behind branches which may create confusion.

Chapter 5

Conclusions

5.1 Summary

The purpose of this thesis work was to study the geometrical-physical model of point cloud and provide automatic tools to categorize LiDAR data. It was done in sight of the construction of the Digital Twin (DT) about municipality of Bologna.

DT are dynamic models with many advantages. By using some online sensors, they allow to accurately simulate the real environment and analyse it in real-time, testing theories, simulating scenarios and predicting results. All these things with the bonus of no influencing the physical environment [5].

Urban Digital Twins (UDT) would be capable of simulating the whole city on many levels: traffic flow, air quality, green area, power line, aqueduct, energy consumption [109] and so on. In order to create one UDT, the first step is recreating the real environment as a platform where simulations will be carried out.

LiDAR data have been widely employed to develop three dimensional maps of urban environments [34]. Airborne LiDAR technology has got a good enough resolution and sufficient easy acquisition method to make it the perfect instrument for this purpose.

I conducted analyses about point density and how it changes setting different neighbourhoods. The first result was that there are on average more points per m^2 than the lower limit stated in the specification (see Sec. 1.3). Then, analysing how point distribution changes at different scales and types, I noticed that it differs on the (real) environment on which the cloud is collected. This reveals that on trees more LiDAR points are elicited, as expected from complex structures such as canopies because of multiple returns per emitted pulse.

These initial assessments have suggested extracting geometrical features too, in addition to others related to height and colours. All the selected features –once the neighbourhood parameters had been defined– have been supplied to a random forest model to classify point by point the cloud in building, car, grass, rail, road or tree points.

The training and testing datasets were provided by some data about Bologna and hand-labelled for this purpose. *Segment Anything Model* [75] segmented off-ground points through *segment-lidar* Python package [73], while I manually segmented ground points. Then I hand-labelled the segments and I split the shuffled points into balanced training and testing datasets.

The final results are sufficiently good for our purposes. Indeed, we were interested in obtaining an initial coarse-grained model as a starting point for more complex models. As shown in Tab. 4.4 it reaches a total accuracy of 95%, with lower category accuracy of 88%, while precision, recall and F1-score metrics are all above 87%. The other goal was to select the most useful features. As reported in Tab. 4.2, now we have their ranking and we found the better hyperparameters of the neighbourhood (a sphere with radius 1.0 m).

5.2 Future improvements

The next step is trying to use other models to outperform current results. At the moment CINECA team is trying other machine learning models such as *Boost Decisions Forest* [110] and *Balanced Random Forest* [111].

In addition, we are initiating a new approach. We are implementing Convolutional Neural Network (CNN) on LiDAR. The core idea is to create three pictures on different scales per each point, but rather than using RGB values, image channels will be some of the features studied in this thesis work [68]. The images will be inputs of CNN, a truly common deep learning model in figures analysis that achieves excellent results [112].

On the other hand, other data may be provided to the classifier. We are thinking about integrating the real estate registry, both to outperform classification results and to update the municipality archives. We are also interested in combining LiDAR with orthophotos because many features can be extracted from them [13].

As far as the appearance of the DT is concerned, improvements and modifications will evolve along with it. The classifier –and more generally the creation of the simulation environment– will have to change to adapt and meet the demands of the DT. Therefore, as long as the DT is updated and expanded, the classifier will have to be constantly improved.

Appendix A

Density histograms

In this appendix I report histograms about some of the values of k and radius in the density analyses of Sec. 2.2.

In the first figure (A.1) I show the results of the k-nearest density. Each column reports the same tile results: almost tree on the left, almost buildings on the right, and intermediate ones in the middle. In Fig. A.2 there are results of the number of point density. The columns have the same organization of the previous figure.

It is very difficult to extrapolate information just by watching the graphs. In the k-nearest we see that tiles with trees have histograms spread on smaller ranges, highlighting a higher concentration of points, but it is difficult to guess the mean value. In the number of points inside the sphere we can make out the shift of the average toward greater values when the percentage of trees increases, as we described in Fig. 2.4.

In the first row of Fig. A.2 the selected radius defines a sphere with volume about $1\,\mathrm{m}^3$. We see that histograms are different from the ones with cylindrical shape and the base area of $1\,\mathrm{m}^2$ and that mean values do not follow the same raster order (higher density for more trees) as can be seen in Tab. A.1.

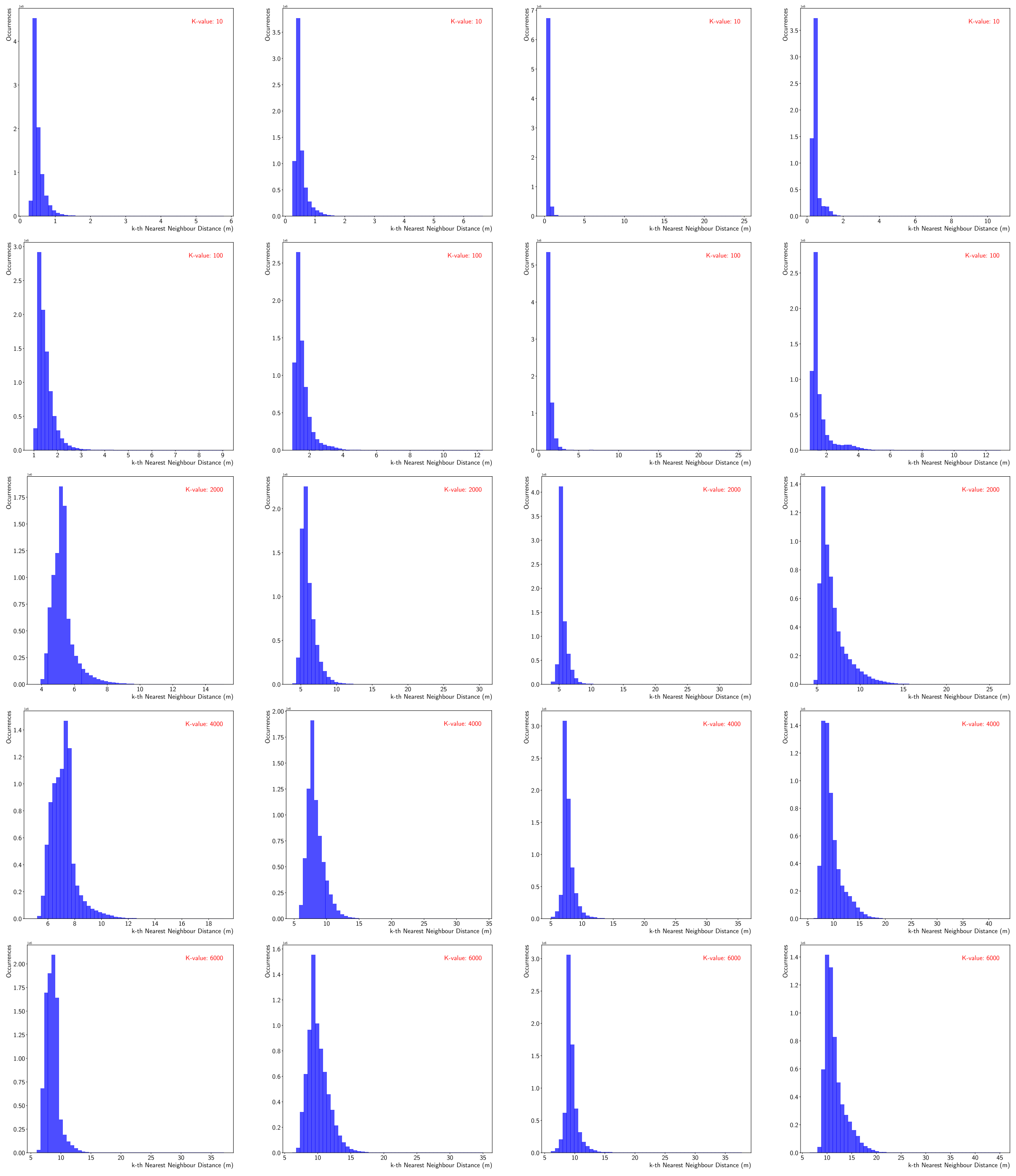


Figure A.1: The k-nearest neighbourhood distance histograms. The four columns report results about the tile a) b) c) and d) of Fig. 2.1, respectively. In the rows there are their histograms with k equal to 10, 100, 2000, 4000 and 6000, respectively.

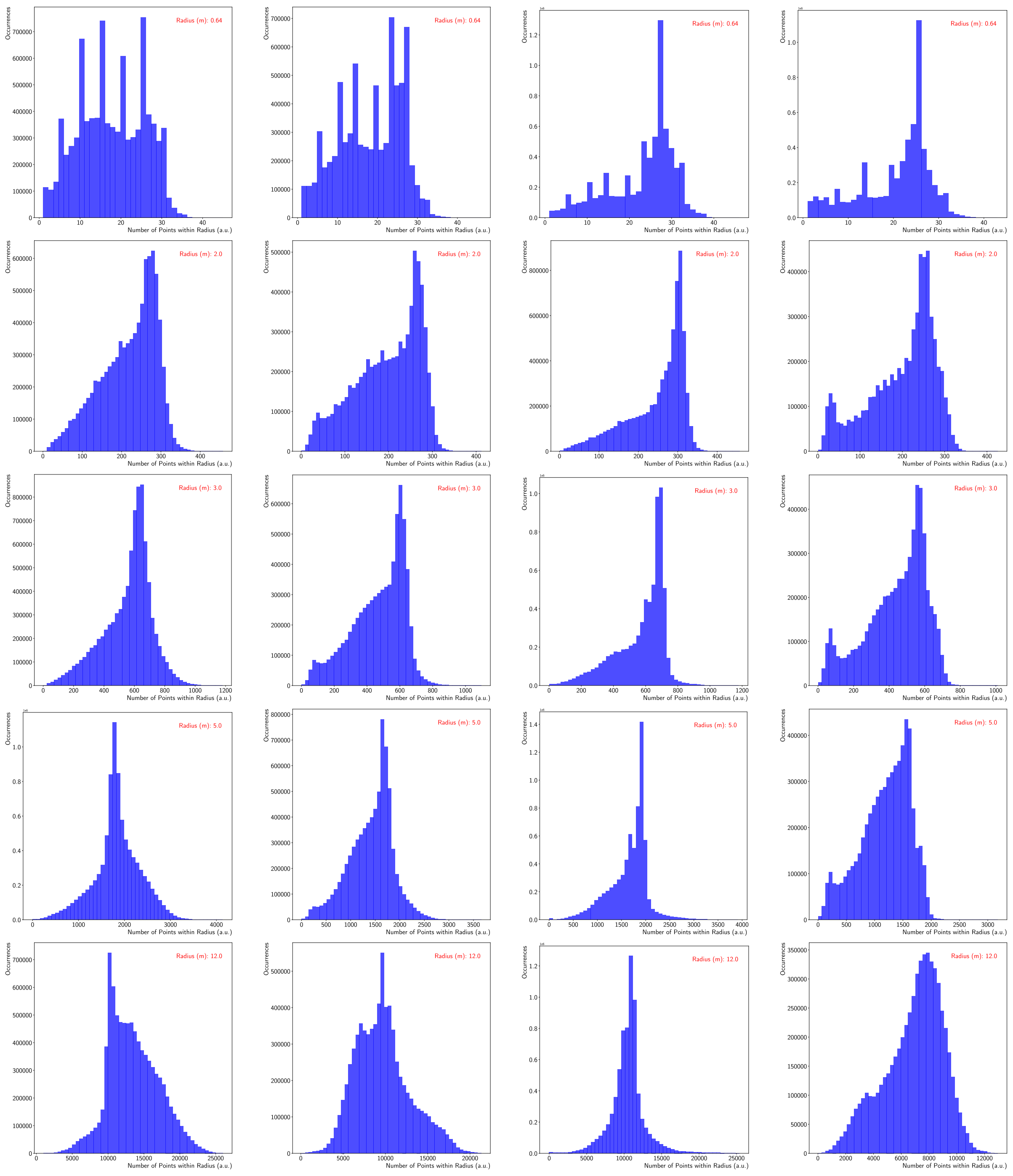


Figure A.2: The number of points inside sphere histograms. The four columns report results about the tile a) b) c) and d) of Fig. 2.1, respectively. In the rows there are their histograms with radius equal to 0.640 m, 2.0 m, 3.0 m, 5.0 m and 12.0 m, respectively.

Table A.1: The table reports mean distances of the k-nearest point (first five rows) and the mean numbers of points inside a sphere (last five rows). Their histograms are shown in Fig. A.1 and A.2, respectively. Second column report k values (first five rows) and radius values (last five rows). In the third, fourth, fifth and sixth column there are the mean values (in meter the first five, dimensionless the last five) of tile a) b) c) d) of Fig. 2.1, respectively.

Type	Fixed	a	b	c	d
	10	0.50855 ± 0.00006	0.50272 ± 0.00007	0.42894 ± 0.00006	0.48490 ± 0.00009
	100	1.48714 ± 0.00011	1.57026 ± 0.00012	1.355722 ± 0.00016	1.5782 ± 0.0003
k-nearest	2000	5.3111 ± 0.0002	6.0491 ± 0.0004	5.5922 ± 0.0004	6.8608 ± 0.0007
	4000	7.1674 ± 0.0003	8.3205 ± 0.0005	7.7615 ± 0.0004	9.6010 ± 0.0009
	6000	8.5744 ± 0.0004	10.0409 ± 0.0006	9.3920 ± 0.0005	11.5980 ± 0.0009
	0.640	17.771 ± 0.003	18.160 ± 0.003	23.193 ± 0.003	20.524 ± 0.003
	2.0	217.92 ± 0.02	196.63 ± 0.02	247.55 ± 0.02	200.00 ± 0.03
radius (m)	3.0	557.81 ± 0.05	470.02 ± 0.06	572.59 ± 0.06	441.67 ± 0.07
	5.0	1838.25 ± 0.17	1424.32 ± 0.17	1661.65 ± 0.15	1194.54 ± 0.18
	12.0	13451.7 ± 1.2	9684.3 ± 1.2	10502.9 ± 0.9	6899.1 ± 0.9

Appendix B

Neighbourhood selection

I have tested spherical and cylindrical shapes with different radii.

Delta Z In Fig. B.1 ΔZ with different surrounds are showed. We can see as spherical neighbourhoods detect more elements, since cylindrical ones almost highlight buildings edges. About the radius, 1 m (Fig. B.1b) is the compromise between a good definition (Fig. B.1c and B.1d are more blurred) and resolution (Fig. B.1a does not detect cars, for examples).

MAD MAD values are showed in Fig. B.2. Spherical surrounds are less useful than the cylindrical ones. About the radius, we can find the same considerations made for ΔZ: so 1 m is the best.

Planarity Planarity calculated with cylindrical neighbourhoods is very blurred, even with radius equal to 0.5 m (Fig. B.3e). Using a spherical surround with 0.5 m of radius (Fig. B.3a) we obtain a high resolution, but it is affected by noise; so we prefer 1 m of radius (Fig. B.3b). Planarity with radius 0.5 m and spherical shape (Fig. B.3a) is almost always 0. Probably it is due to too few points in the neighbourhood: it is difficult to find a plane, therefore a linear distribution is preferred. It can be a consequence of plane fly: the fly direction creates a preferential low scale distribution direction, so a linear distribution.

Sphericity There are not many differences between cylindrical and spherical neighbourhoods (Fig.B.4). Maybe the cylindrical one (Fig.B.4f) is a bit better than the spherical (Fig.B.4b) at 1 m.

Linearity Linearity with cylindrical neighbourhoods is very blurred (Fig. B.5e, B.5f, B.5g and B.5h). Radius 1 m is the best choice in spherical neighbourhood (Fig. B.5b). We notice that in Fig. B.5a (radius 0.5 m, shape spherical) linearity is almost always 1,

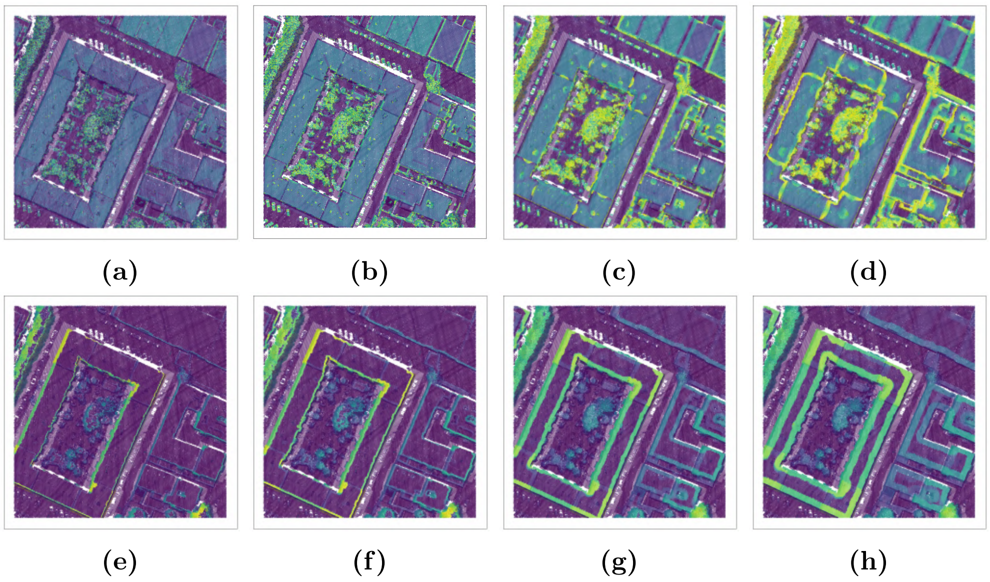


Figure B.1: Different surrounds of ΔZ related to $100\,\mathrm{m} \times 100\,\mathrm{m}$ subtile shown in Fig. 3.1. a), b), c) and d) show ΔZ values with spherical surrounds with radius 0.5 m, 1 m, 2 m and 3 m. e), f), g) and h) show ΔZ values with cylindrical surrounds with radius 0.5 m, 1 m, 2 m and 3 m.

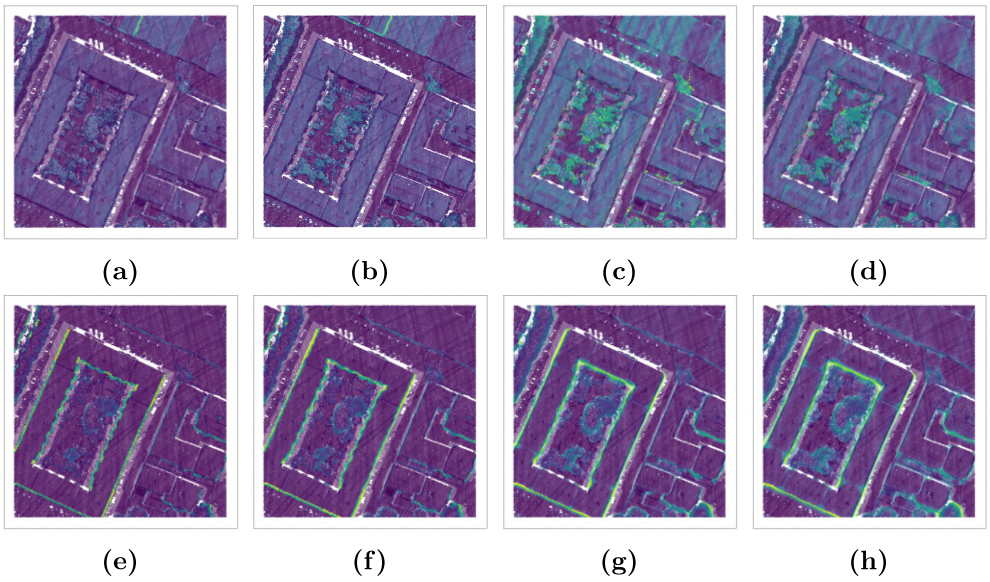


Figure B.2: Different surrounds of MAD related to $100\,\mathrm{m} \times 100\,\mathrm{m}$ subtile shown in Fig. 3.1. a), b), c) and d) show MAD values with spherical surrounds with radius 0.5 m, 1 m, 2 m and 3 m. e), f), g) and h) show MAD values with cylindrical surrounds with radius 0.5 m, 1 m, 2 m and 3 m.

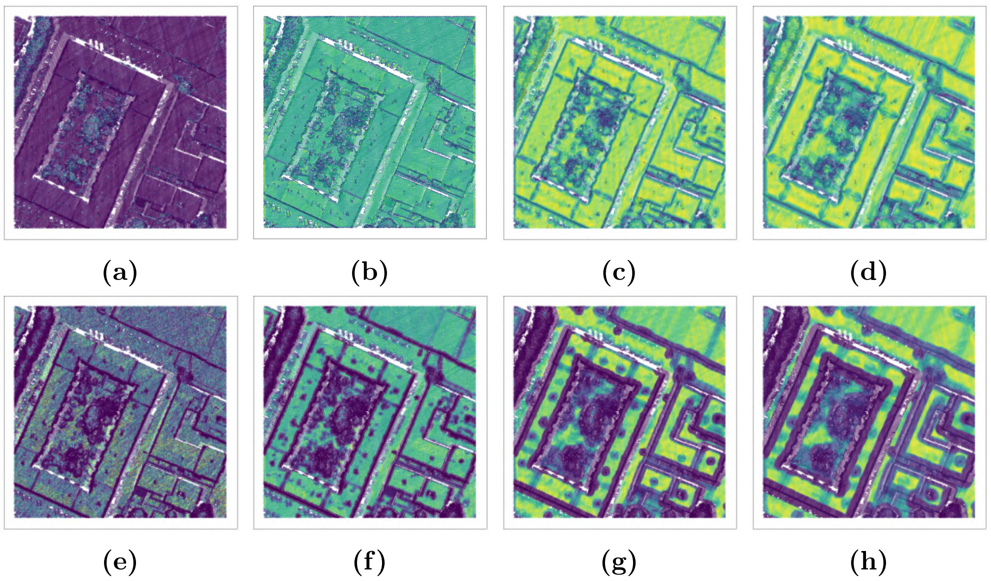


Figure B.3: Different surrounds of planarity related to 100 m × 100 m subtile shown in Fig. 3.1. a), b), c) and d) show planarity values with spherical surrounds with radius 0.5 m, 1 m, 2 m and 3 m. e), f), g) and h) show planarity values with cylindrical surrounds with radius 0.5 m, 1 m, 2 m and 3 m.

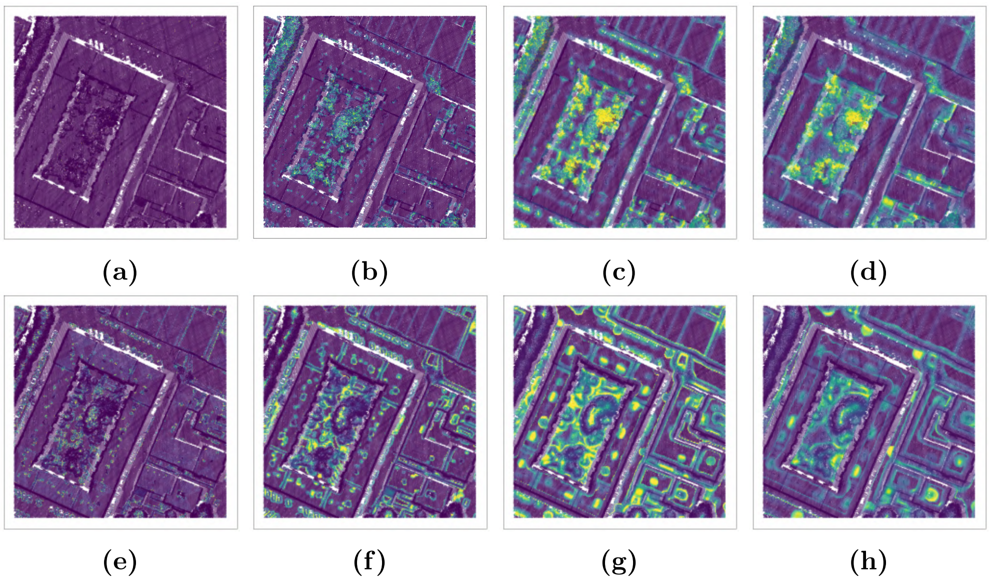


Figure B.4: Different surrounds of sphericity related to 100 m × 100 m subtile shown in Fig. 3.1. a), b), c) and d) show sphericity values with spherical surrounds with radius 0.5 m, 1 m, 2 m and 3 m. e), f), g) and h) show sphericity values with cylindrical surrounds with radius 0.5 m, 1 m, 2 m and 3 m.

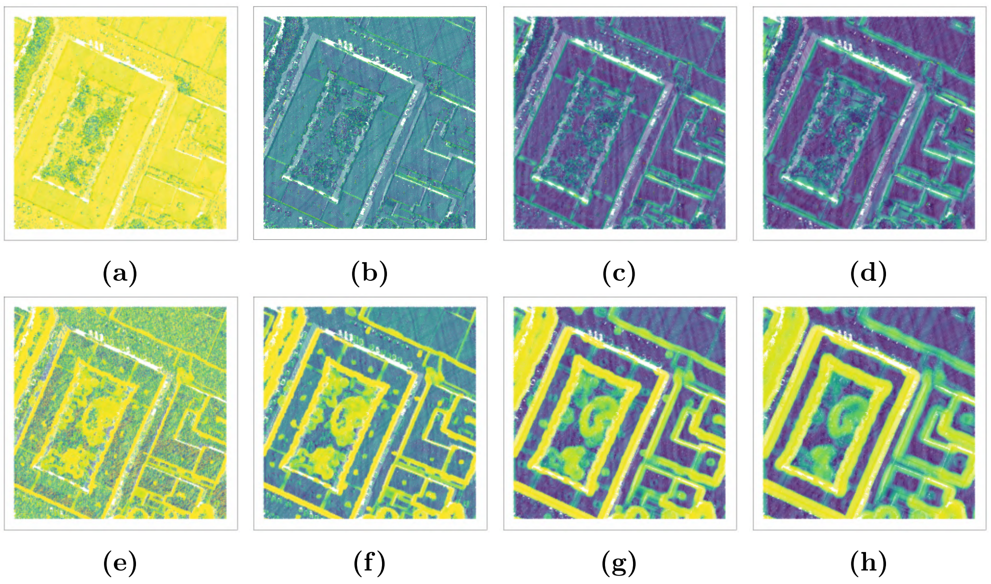


Figure B.5: Different surrounds of linearity related to 100 m × 100 m subtile shown in Fig. 3.1. a), b), c) and d) show linearity values with spherical surrounds with radius 0.5 m, 1 m, 2 m and 3 m. e), f), g) and h) show linearity values with cylindrical surrounds with radius 0.5 m, 1 m, 2 m and 3 m.

because planarity in the same condition is almost always 0 and in general sphericity is 0 except on canopies.

Entropy There are many differences in entropy changing shape and radius. In Fig. B.6a entropy has very low value, while just with radius equal to 1 m (Fig. B.6b) values are greater. With radius 0.5 m in a spherical neighbourhood, linearity is almost always 1, so entropy is almost always 0. Neighbourhoods with radius greater than 1 m generate blurred entropy values. Cylindrical shapes (Fig. B.6e and B.6f) highlight buildings and canopies edges, while spherical ones (Fig. B.6b) detect surfaces: each building roofs and tree canopies are uniform.

Theta Cylindrical neighbourhoods create blurred values, as it is shown in Fig. B.7e, B.7f, B.7g and B.7h. A small radius generates good feature, such as in Fig.B.7a and B.7b, respectively 0.5 m and 1 m.

Theta variance All the neighbourhoods with radius greater or equal to 2 m are blurred (Fig.B.8c and B.8g). Cylindrical shapes highlight the profiles but not the edges, while the spherical ones highlight local variations. The sharpest and useful neighbour is the spherical one with radius equal to 0.5 m (Fig. B.8a)

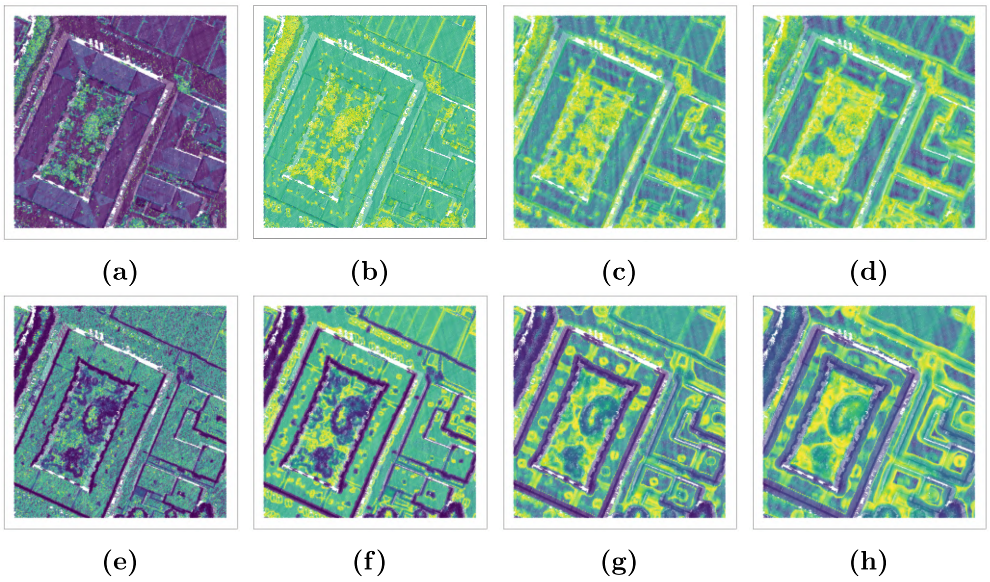


Figure B.6: Different surrounds of entropy related to 100 m × 100 m subtile shown in Fig. 3.1. a), b), c) and d) show entropy values with spherical surrounds with radius 0.5 m, 1 m, 2 m and 3 m. e), f), g) and h) show entropy values with cylindrical surrounds with radius 0.5 m, 1 m, 2 m and 3 m.

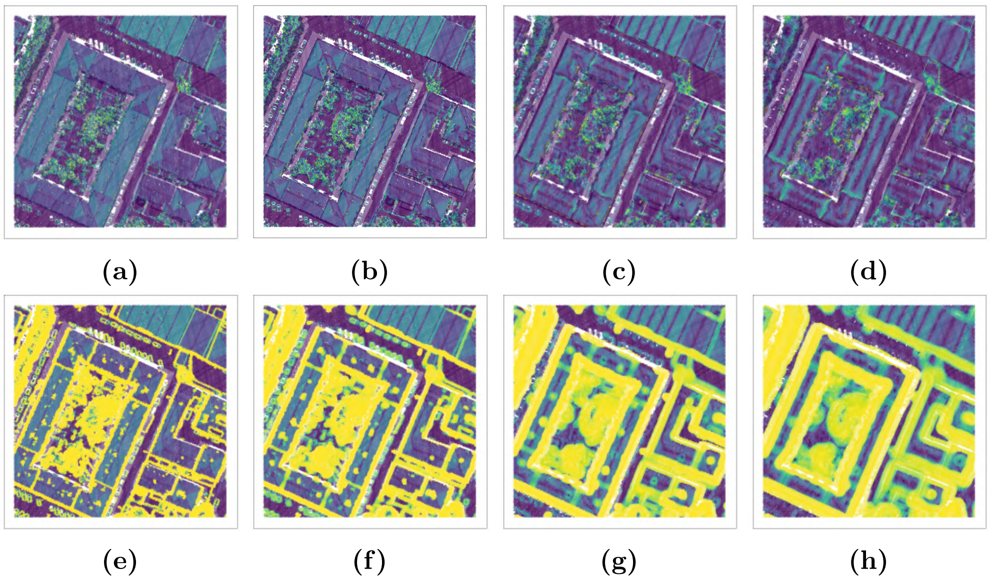


Figure B.7: Different surrounds of θ related to 100 m × 100 m subtile shown in Fig. 3.1. a), b), c) and d) show θ values with spherical surrounds with radius 0.5 m, 1 m, 2 m and 3 m. e), f), g) and h) show θ values with cylindrical surrounds with radius 0.5 m, 1 m, 2 m and 3 m.

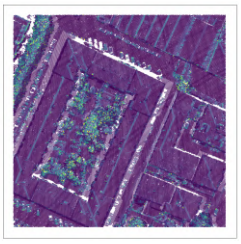
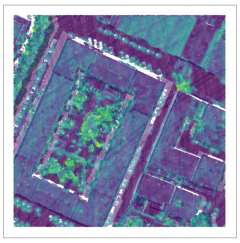
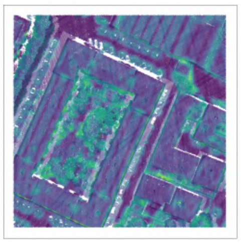
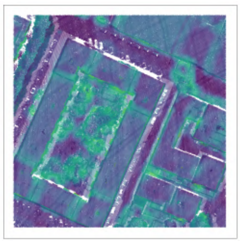
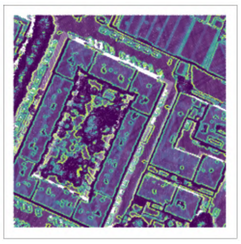
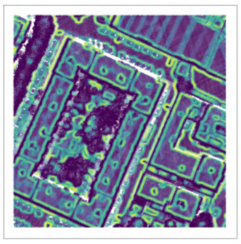
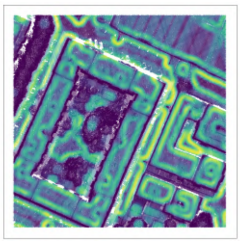
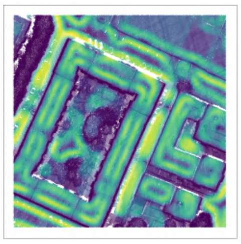

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

Figure B.8: Different surrounds of σ_θ^2 related to 100 m $\times$ 100 m subtile shown in Fig. 3.1. a), b), c) and d) show σ_θ^2 values with spherical surrounds with radius 0.5 m, 1 m, 2 m and 3 m. e), f), g) and h) show σ_θ^2 values with cylindrical surrounds with radius 0.5 m, 1 m, 2 m and 3 m.

Appendix C

Segmentation and Resolution

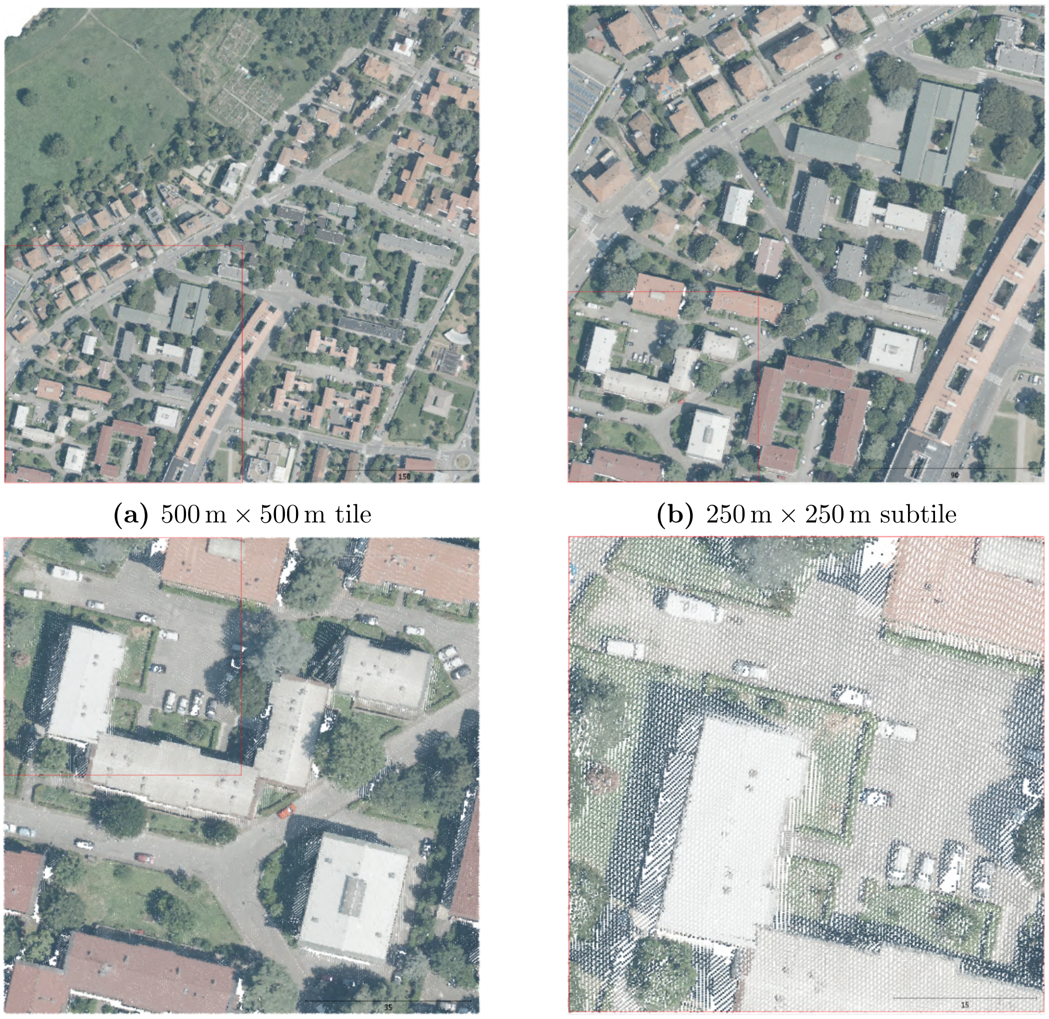
Segmentation

As mentioned in Sec. 4.1, the use of SAM requires some thoughts about the tile size. Since LiDAR data are converted into pictures, resolution and objects size in the picture are important.

In order to manage these aspects, I took an original tile of 500 m × 500 m (Fig. C.1a), then I spit it in four subtiles of 250 m × 250 m (Fig. C.1b), twenty-five subtiles of 100 m × 100 m (Fig. C.1c) and one hundred subtiles of 50 m × 50 m (Fig. C.1d). Then, I segmented the whole tile and all the subtiles, using classification variable to define which points do not belong to the ground. In the second column of Tab. C.1 I report the number of segments detected in the whole surface if it has been split or not. It is evident that the size of the segmented point cloud causes how many segments SAM can detect on the same 500 m × 500 m surface.

In Fig. C.2, the detected segments of one subtile per tested size and of the whole tile are shown (they are the same subtiles and tile of Fig. C.1). The number of segments per each of them is reported on the third column of Tab. C.1. A thorough analysis reveals that, using the whole tile (Fig. C.2a) or the 250 m × 250 m subtiles (Fig. C.2b), cars are rarely detected and many elements are grouped together (mixed segments). Obviously, using 50 m × 50 m subtiles (Fig. C.2d) more segments are detected and the probability of mixed segments is very low, but there is more noise too (many segments contain only very few points). Additionally, it is difficult to visually identify what those points are related to, so the hand-labelling procedure would be very hard (it is due to LiDAR points density).

After all these considerations, I chose to segment and hand-label subtiles of 100 m × 100 m (Fig. C.2c): they are small enough to segment cars and create a few mixed segments, but they are not too small to prevent a good visualization or too many small segments.



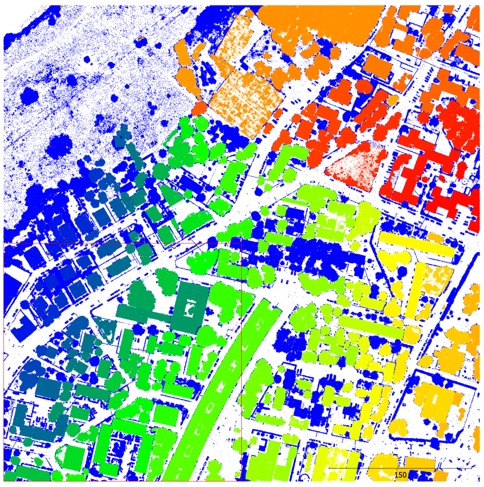
(a) 500 m × 500 m tile

(b) 250 m × 250 m subtile

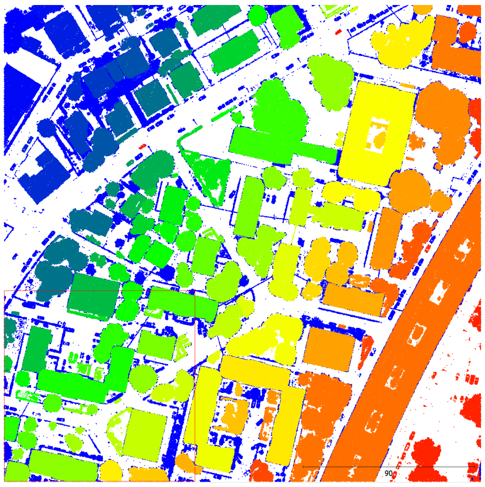
(c) 100 m × 100 m subtile

(d) 50 m × 50 m subtile

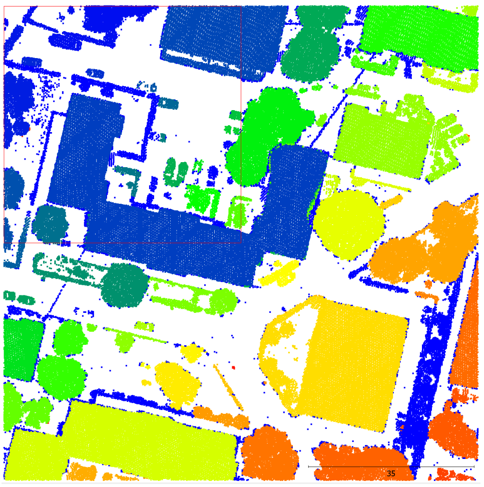
Figure C.1: a) shows the RGB values of the LiDAR point cloud acquired near "via Corolano Vighi" and "via Leone Tolstoi". b), c) and d) show RGB values of parts of the same point cloud with different enlargement: 250 m × 250 m, 100 m × 100 m and 50 m × 50 m respectively. The red bounding-box in the first three limits the point cloud of the next enlargement.



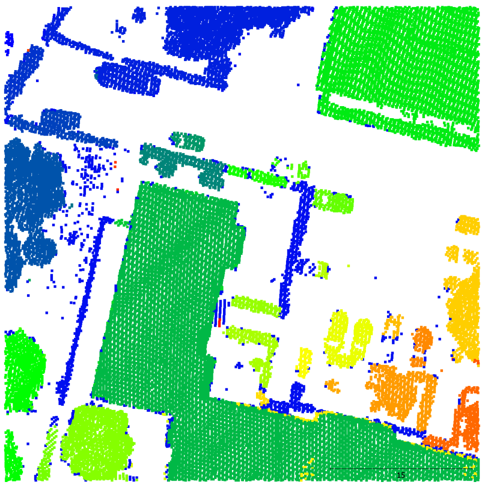
(a) 500 m × 500 m tile



(b) 250 m × 250 m subtile



(c) 100 m × 100 m subtile



(d) 50 m × 50 m subtile

Figure C.2: a) shows the segments of the LiDAR point cloud acquired near “via Corolano Vighi” and “via Leone Tolstoi”. b), c) and d) show segments of parts of the same point cloud with different enlargement: 250 m × 250 m, 100 m × 100 m and 50 m × 50 m respectively. The red bounding-box in the first three limits the point cloud of the next enlargement.

Resolution

As said in Sec. 4.1.1, *segment-lidar* allows to set the resolution of the generated image form LiDAR off-ground point cloud.

We can see the different generated images about a subtile (Fig. C.3a) setting resolution at 0.1, 0.25 and 1.0 respectively in Fig. C.3b, C.3c and C.3d. The number of pixels –and so their corresponding real size– changes consistently. If resolution is 1.0 over a $100\,\text{m} \times 100\,\text{m}$ subtile, *segment-lidar* generates an image of $100\,\text{px} \times 100\,\text{px}$. So if resolution is 0.25 and 0.1 it generates $250\,\text{px} \times 250\,\text{px}$ and $1000\,\text{px} \times 1000\,\text{px}$ respectively.

I applied the segmentation algorithm. In Tab. C.2 I reported the resolution, the number of segments in all the subtiles of the original tile and the number of segments in the subtile shown in Fig. C.3. We can notice that increasing the resolution the number of segments increases. It is a realistic trend as a higher resolution increases details. Graphically, it seems to improve segments quality too, at least between resolution 1.0 (Fig. C.3d) and 0.25/0.1 (Fig. C.3c and Fig. C.3b). Indeed, some real elements are merged or wrong split between them at 1.0. On the other hand, 0.1 resolution sometimes oversplits elements.

We must not watch just segmentation results, we have to take care of resolution consistency. LiDAR resolution is fixed and generating an image with more fine-grain resolution means using unrealistic data. All the tiles that I analysed have got at least 6×10^6 points, usually around 7×10^6. Since they cover an area of $500\,\text{m} \times 500\,\text{m}$, their density is at least $24\,\text{p/m}^2$. These results are confirmed by Sec. 2.1. If they had a regular grid structure, they would have been 20.4 cm distant one each other and we would have had 4.9 points per meter. Therefore we would have had 490 points per side in a $100\,\text{m} \times 100\,\text{m}$ subtile.

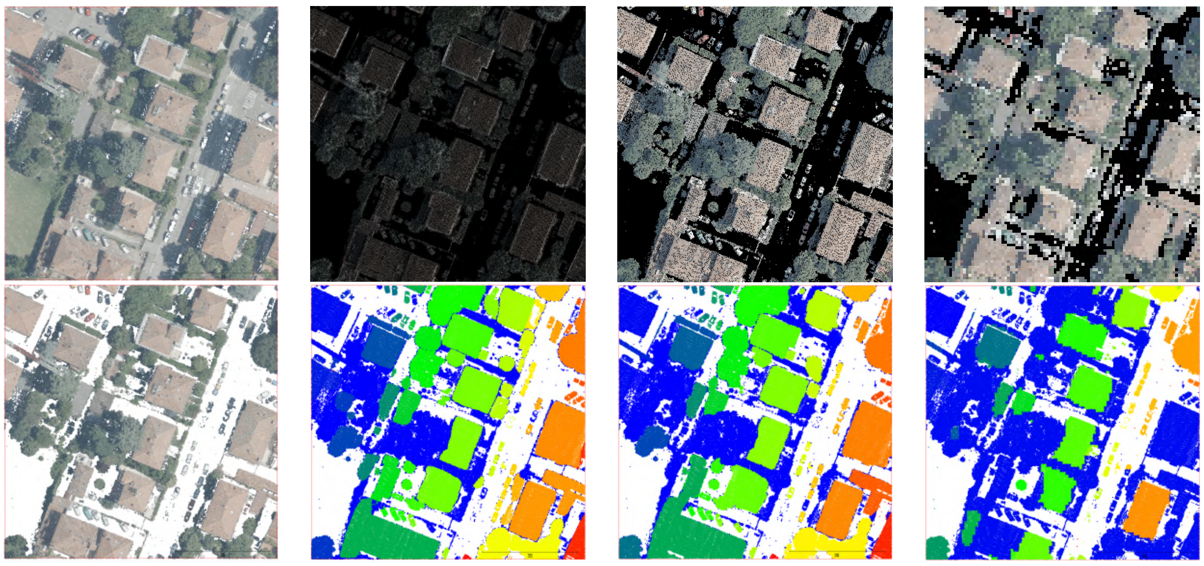
I set the resolution at 0.25 because the generated image would have $250\,\text{px} \times 250\,\text{px}$, almost half of LiDAR points. In this way we avoid scatter pixels over a black background as in Fig. C.3b but we try to take advantage of the whole information provided by the point clouds.

Table C.1: Table of the number of segments in tile and subtiles. The first column reports the size of the tile. The second one reports the number of segments in the whole 500 m × 500 m surface if it is split in subtiles or not. The third column reports the number of segments in the subtile shown in Fig. C.1 and C.2.

	Segments number	
Size	whole tile	single subtile
500	383	383
250	1002	260
100	1656	94
50	3285	45

Table C.2: The table shows the number of segments identified by SAM setting different resolution values in *segment-lidar*. In the first column there is the resolution parameter. In the second one there is the number of segments found in the whole tile. In the third column there is the number of segments found in the subtile shown in Fig. C.3a.

	Segments number	
Resolution	whole tile	single subtile
0.10	1656	112
0.25	1456	86
1.0	926	59



(a) Subtile **(b)** Res. 0.1 **(c)** Res. 0.25 **(d)** Res. 1.0

Figure C.3: a) point cloud (top) and only off-ground points (bottom) of the subtile showing RGB values. b) c) and d) generated images (top) and segments (bottom) of the same subtile calculated by *segment-lidar* with resolution equal to 0.1, 0.25 and 1.0 respectively. The generated images are 1000 px × 1000 px, 250 px × 250 px and 100 px × 100 px respectively.

Acknowledgements

I would like to thank my supervisor for the time dedicated to me, the food for thought, the opportunities granted, and the knowledge imparted. I hope that this work carried out in collaboration has produced the results he hoped for.

To the various members of the university, CINECA, FBK, and the Municipality with whom I collaborated, thank you for allowing me to work with you, as well as for the exchange of ideas and the technical and theoretical support.

Thanks to my family, friends, and colleagues for everything they have given me, hoping that I have offered just as much in return.

Bibliography

[1] Michael Grieves. “Digital Twin: Manufacturing Excellence through Virtual Factory Replication”. In: (Mar. 2015).

[2] Shalini Sharma Goel et al. “A review of Internet of Things: qualifying technologies and boundless horizon”. In: *Journal of Reliable Intelligent Environments* (2021). DOI: 10.1007/S40860-020-00127-W.

[3] Fei Tao et al. “Digital twin-driven product design, manufacturing and service with big data”. In: *The International Journal of Advanced Manufacturing Technology* 94.9 (Feb. 2018), pp. 3563–3576. ISSN: 1433-3015. DOI: 10.1007/s00170-017-0233-1.

[4] Benjamin Schleich et al. “Shaping the digital twin for design and production engineering”. In: *CIRP Annals* 66.1 (2017), pp. 141–144. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2017.04.040.

[5] Edward Glaessgen and David Stargel. “The digital twin paradigm for future NASA and U.S. air force vehicles”. In: Apr. 2012. ISBN: 978-1-60086-937-2. DOI: 10.2514/6.2012-1818.

[6] Cunbo Zhuang, Jianhua Liu, and Hui Xiong. “Digital twin-based smart production management and control framework for the complex product assembly shop-floor”. In: *The International Journal of Advanced Manufacturing Technology* 96.1 (Apr. 2018), pp. 1149–1163. ISSN: 1433-3015. DOI: 10.1007/s00170-018-1617-6.

[7] Margarita Angelidou. “The Role of Smart City Characteristics in the Plans of Fifteen Cities”. In: *Journal of Urban Technology* 24.4 (2017), pp. 3–28. DOI: 10.1080/10630732.2017.1348880.

[8] Petko Hristov et al. “Enabling city digital twins through urban living labs”. In: *The international archives of the photogrammetry, remote sensing and spatial information sciences* (2022). DOI: 10.5194/isprs-archives-xliii-b1-2022-151-2022.

[9] Siamak Ahmadzadeh Bazzaz. “The Application of Digital Twins in Sustainable Urban Planning: From Data Acquisition to 3D Virtualization”. In: (2024). DOI: 10.1109/vrw62533.2024.00349.

[10] Fathima Nishara Abdeen, Sara Shirowzhan, and Samad M.E. Sepasgozar. “Citizen-centric digital twin development with machine learning and interfaces for maintaining urban infrastructure”. In: *Telematics and Informatics* 84 (2023), p. 102032. ISSN: 0736-5853. DOI: https://doi.org/10.1016/j.tele.2023.102032.

[11] International Union of Pure and Applied Chemistry (IUPAC). *Lidar*. 2019. DOI: doi:10.1351/goldbook.L03514.

[12] Farnell Italia Srl, ed. *Introduzione alla tecnologia LiDAR*. 2024. URL: https://it.farnell.com/introduction-to-lidar-technology.

[13] Hengfan Cai et al. “Systematic Comparison of Objects Classification Methods Based on ALS and Optical Remote Sensing Images in Urban Areas”. In: *Electronics* 11.19 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11193041.

[14] Salem Morsy, Ahmed Shaker, and Ahmed El-Rabbany. “Classification of Multispectral Airborne LiDAR Data Using Geometric and Radiometric Information”. In: *Geomatics* 2.3 (2022), pp. 370–389. ISSN: 2673-7418. DOI: 10.3390/geomatics2030021.

[15] Xiaomei Lu et al. “Nearshore bathymetry and seafloor property studies from Space lidars: CALIPSO and ICESat-2.” In: *Optics Express* 30 20.20 (2022), pp. 36509–36525. DOI: 10.1364/oe.471444.

[16] Jaehoon Jung, Jaebin Lee, and Christopher Parrish. “Inverse Histogram-Based Clustering Approach to Seafloor Segmentation from Bathymetric Lidar Data”. In: *Remote Sensing* (2021). DOI: 10.3390/RS13183665.

[17] Teemu Kumpumaki et al. “Data-Driven Approach to Benthic Cover Type Classification Using Bathymetric LiDAR Waveform Analysis”. In: *Remote Sensing* (2015). DOI: 10.3390/RS71013390.

[18] Gunnam Kiran Kumar et al. “Systems and methods for efficient multi-return light detectors”. 2020.

[19] Benjamin D. Roth et al. “Simulations of Leaf BSDF Effects on Lidar Waveforms”. In: *Remote Sensing* 12.18 (2020), pp. 2909–. DOI: 10.3390/RS12182909.

[20] Krishna Mohan et al. “LIDAR Based Landing Site Identification and Safety Estimation For Inter Planetary Missions”. In: *2023 International Conference on Control, Communication and Computing (ICCC)*. 2023, pp. 1–5. DOI: 10.1109/ICCC57789.2023.10165199.

[21] Timothy P. Setterfield et al. “Real-World Testing of LiDAR-Inertial Based Navigation and Mapping for Precision Landing”. In: *2022 IEEE Aerospace Conference (AERO)*. 2022, pp. 1–10. DOI: 10.1109/AERO53065.2022.9843356.

[22] Eric Schindhelm et al. “A scanning LIDAR system for active hazard detection and avoidance during landing on Europa”. In: *2018 IEEE Aerospace Conference*. 2018, pp. 1–7. DOI: 10.1109/AERO.2018.8396503.

[23] Anshit Kumar Srivastava, Abhishek Singhal, and Aarti Sharma. “Analysis of Lidar-Based Autonomous Vehicle Detection Technologies for Recognizing Objects”. In: *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*. 2023, pp. 1–5. DOI: 10.1109/ISCON57294.2023.10112089.

[24] Pengcheng Shi, Yongjun Zhang, and Jiayuan Li. “LiDAR-Based Place Recognition For Autonomous Driving: A Survey”. In: *arXiv.org* (2023). DOI: 10.48550/arXiv.2306.10561.

[25] Sambit Mohapatra et al. “LiDAR-BEVMTN: Real-Time LiDAR Bird’s-Eye View Multi-Task Perception Network for Autonomous Driving”. In: *arXiv.org* (2023). DOI: 10.48550/arXiv.2307.08850.

[26] Niklas Kohlisch, Philipp Koch, and Stefan May. “LiDAR-Based Augmented Reality for the Development of Test Scenarios on Safety for Autonomous Operation of a Shunting Locomotive”. In: *2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2023, pp. 23–28. DOI: 10.1109/ICARSC58346.2023.10129540.

[27] Thomas Genevois et al. “Augmented Reality on LiDAR data: Going beyond Vehicle-in-the-Loop for Automotive Software Validation”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 971–976. DOI: 10.1109/IV51971.2022.9827351.

[28] David W. Illig et al. *Observations of ocean water properties with blue wavelength airborne lidar*. 2023. DOI: 10.1117/12.2665074.

[29] Daniel Tamburlin et al. “Testing the Height Variation Hypothesis with the R rasterdiv Package for Tree Species Diversity Estimation”. In: *Remote Sensing* 13.18 (2021), pp. 3569–. DOI: 10.3390/RS13183569.

[30] Michael T. Hirschmann. “Scale dependency of lidar-derived forest structural diversity”. In: *Methods in Ecology and Evolution* 14.2 (2023), pp. 708–723. DOI: 10.1111/2041-210x.14040.

[31] Bangyi Tao et al. “Precise detection of water surface through the analysis of a single green waveform from bathymetry LiDAR.” In: *Optics Express* 30 22.22 (2022), pp. 40820–40841. DOI: 10.1364/oe.468404.

[32] Sharad Mehta, Jack Peach, and Andrew Weinert. “To Expedite Roadway Identification and Damage Assessment in LiDAR 3D Imagery for Disaster Relief Public Assistance”. In: *Infrastructures* 7.3 (2022), pp. 39–39. DOI: 10.3390/infrastructures7030039.

[33] Efstratios Karantanellis, Vassilis Marinos, and George Papathanassiou. "Multitemporal Landslide Mapping and Quantification of Mass Movement in Red Beach, Santorini Island Using Lidar and UAV Platform". In: Springer, Cham, 2019, pp. 163–169. DOI: 10.1007/978-3-319-93124-1_20.

[34] Wai Yeung Yan, Ahmed Shaker, and Nagwa El-Ashmawy. "Urban land cover classification using airborne LiDAR data: A review". In: *Remote Sensing of Environment* 158 (2015), pp. 295–310. ISSN: 0034-4257. DOI: 10.1016/j.rse.2014.11.001.

[35] A. Bellakaout et al. "Automatic 3D Extraction of Buildings, Vegetation and Roads from LIDAR Data". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLI-B3 (2016), pp. 173–180. DOI: 10.5194/isprs-archives-XLI-B3-173-2016.

[36] Joachim Niemeyer, Franz Rottensteiner, and Uwe Soergel. "Contextual classification of lidar data and building object detection in urban areas". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 87 (2014), pp. 152–165. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2013.11.001.

[37] Rebecca L. Powell et al. "Sub-pixel mapping of urban land cover using multiple endmember spectral mixture analysis: Manaus, Brazil". In: *Remote Sensing of Environment* 106.2 (2007), pp. 253–267. ISSN: 0034-4257. DOI: 10.1016/j.rse.2006.09.005.

[38] Paolo Gamba, Fabio Dell'Acqua, and Belur V. Dasarathy. "Urban remote sensing using multiple data sets: Past, present, and future". In: *Information Fusion* 6.4 (2005). Fusion of Remotely Sensed Data over Urban Areas, pp. 319–326. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2005.02.007.

[39] Li Guo et al. "Relevance of airborne lidar and multispectral image data for urban scene classification using Random Forests". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 66.1 (2011), pp. 56–66. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2010.08.007.

[40] Mohammad Awrangjeb, Chunsun Zhang, and Clive S. Fraser. "Automatic extraction of building roofs using LIDAR data and multispectral imagery". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 83 (2013), pp. 1–18. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2013.05.006.

[41] Mafalda Oliveira and André R. S. Marçal. *Clustering LiDAR Data with K-means and DBSCAN*. 2023. DOI: 10.5220/0011667000003411.

[42] Juan C. Suárez et al. "Use of airborne LiDAR and aerial photography in the estimation of individual tree heights in forestry". In: *Computers & Geosciences* 31.2 (2005). Geospatial Research in Europe: AGILE 2003, pp. 253–262. ISSN: 0098-3004. DOI: 10.1016/j.cageo.2004.09.015.

[43] Shi Yuli Xu Fan Zhang Xuehong. “Research on Classification of Land Cover based on LiDAR Cloud and Aerial Images”. In: *Remote Sensing Technology and Application* 34.2, 253 (2019), pp. 253–262. DOI: 10.11873/j.issn.1004-0323.2019.2.0253.

[44] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV].

[45] Loic Landrieu and Martin Simonovsky. *Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs*. 2018. arXiv: 1711.09869 [cs.CV].

[46] Yangyan Li et al. “PointCNN: Convolution On X-Transformed Points”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/f5f8590cd58a54e94377e6ae2eded4d9-Paper.pdf.

[47] Yue Wang et al. “Dynamic Graph CNN for Learning on Point Clouds”. In: *ACM Trans. Graph.* 38.5 (Oct. 2019). ISSN: 0730-0301. DOI: 10.1145/3326362.

[48] Elyta Widyaningrum et al. “Airborne Laser Scanning Point Cloud Classification Using the DGCNN Deep Learning Method”. In: *Remote Sensing* 13.5 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13050859.

[49] Charles Ruizhongtai Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: (June 2017). DOI: 10.48550/arXiv.1706.02413.

[50] Xingzhong Nong, Wenfeng Bai, and Guanlan Liu. “Airborne LiDAR point cloud classification using PointNet++ network with full neighborhood features”. In: *PLOS ONE* 18.2 (Feb. 2023), pp. 1–14. DOI: 10.1371/journal.pone.0280346.

[51] CGR S.p.A, ed. *Compagnia Generale Ripreseaeree*. 2024. URL: https://www.cgrspa.com.

[52] Leica Geosystem, ed. *CityMapper 2*. 2024. URL: https://leica-geosystems.com/it-it/products/airborne-systems/hybrid-sensors/leica-citymapper-2.

[53] P. Axelsson. “DEM Generation from Laser Scanner Data Using Adaptive TIN Models”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 33 (2000), pp. 110–117. URL: https://api.semanticscholar.org/CorpusID:59795495.

[54] European Environment Agency, ed. *Digital Terrain Model*. URL: https://www.eea.europa.eu/help/glossary/eea-glossary/digital-terrain-model.

[55] American Society for Photogrammetry and Remote Sensing, eds. *LASER (LAS) file format exchange activities*. URL: https://www.asprs.org/divisions-committees/lidar-division/laser-las-file-format-exchange-activities.

[56] Songrit Maneewongvatana and David M. Mount. *Analysis of approximate nearest neighbor searching with clustered point sets*. 1999. DOI: 10.48550/arXiv.cs/9901013.

[57] GuoWei Lu et al. "Research on point cloud organization method based on KD tree". In: *International Conference on Geographic Information and Remote Sensing Technology (GIRST 2022)*. Ed. by Yang Wang. Vol. 12552. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. Feb. 2023, 1255226, p. 1255226. DOI: 10.1117/12.2667418.

[58] Qi Zhao and Yonghua Qu. "The Retrieval of Ground NDVI (Normalized Difference Vegetation Index) Data Consistent with Remote-Sensing Observations". In: *Remote sensing* (2024). DOI: 10.3390/rs16071212.

[59] Masahiko Taniguchi and Jonathan S. Lindsey. "Absorption and Fluorescence Spectral Database of Chlorophylls and Analogues". In: *Photochemistry and Photobiology* 97.1 (2021), pp. 136–165. DOI: 10.1111/PHP.13319.

[60] S. K. McFEETERS. "The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features". In: *International Journal of Remote Sensing* 17.7 (1996), pp. 1425–1432. DOI: 10.1080/01431169608948714.

[61] Yunhao Chen et al. "Hierarchical object oriented classification using very high resolution imagery and LIDAR data over urban areas". In: *Advances in Space Research* 43.7 (2009), pp. 1101–1110. ISSN: 0273-1177. DOI: 10.1016/j.asr.2008.11.008.

[62] W Dou and YH Chen. "Report of urban planning module in macres airborne remote sensing programme". In: *Malaysian Center for Remote Sensing (MACRES), Malaysia* (2005).

[63] Nesrine Chehata, Li Guo, and Clément Mallet. "Airborne LIDAR feature selection for urban classification using random forests". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38 (Jan. 2009). URL: https://www.researchgate.net/publication/242731335_Airborne_LIDAR_feature_selection_for_urban_classification_using_random_forests.

[64] H. Gross, Boris Jutzi, and U. Thoennessen. "Segmentation of tree regions using data of a full-waveform laser". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36 (Jan. 2007). DOI: 10.24406/publica-fhg-357022.

[65] C. E. Shannon. "A mathematical theory of communication". In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[66] Martin Weinmann et al. “Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 105 (2015), pp. 286–304. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2015.01.016.

[67] R. C. dos Santos et al. “Building Detection from LiDAR data using Entropy and the K-mean Concept”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W13 (2019), pp. 969–974. DOI: 10.5194/isprs-archives-XLII-2-W13-969-2019.

[68] Zhishuang Yang et al. “Segmentation and Multi-Scale Convolutional Neural Network-Based Classification of Airborne Laser Scanner Data”. In: *Sensors* 18.10 (2018). ISSN: 1424-8220. DOI: 10.3390/s18103347.

[69] Gang Xu and Zhengyou Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition*. Springer Netherlands, 1996. DOI: 10.1007/978-94-015-8668-9.

[70] Ali Akbar Matkan et al. “Spatial analysis for outlier removal from lidar data”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2014), pp. 187–190. DOI: 10.5194/ISPRSARCHIVES-XL-2-W3-187-2014.

[71] Yong Hu. “Automated extraction of digital terrain models, roads and buildings using airborne lidar data”. PhD thesis. Oct. 2003. ISBN: 9780494381311. URL: https://www.researchgate.net/publication/234478566_Automated_extraction_of_digital_terrain_models_roads_and_buildings_using_airborne_lidar_data.

[72] Jens Behley, Volker Steinhage, and Armin B. Cremers. *Efficient radius neighbor search in three-dimensional point clouds*. 2015. DOI: 10.1109/ICRA.2015.7139702.

[73] Anass Yarroudh. *LiDAR Automatic Unsupervised Segmentation using Segment-Anything Model (SAM) from Meta AI*. GitHub Repository. 2023. URL: https://github.com/Yarroudh/segment-lidar.

[74] Qiusheng Wu and Luca Prado Osco. “A Python package for segmenting geospatial data with the Segment Anything Model (SAM)”. In: *The Journal of Open Source Software* 8(89) (Sept. 2023). Ed. by Hugo Ledoux, p. 5663. DOI: 10.21105/joss.05663.

[75] Alexander Kirillov et al. “Segment Anything”. In: *arXiv:2304.02643* (2023). DOI: 10.48550/arXiv.2304.02643.

[76] *CloudCompare*. 2024. URL: https://www.cloudcompare.org/presentation.html.

[77] Competition & Markets Authority. *AI Foundation Model: Initial Report*. Sept. 2023. URL: https://assets.publishing.service.gov.uk/media/65081d3aa41cc300145612c0/Full_report_.pdf.

[78] Stanford Institute for Human-Centered Artificial Intelligence. *Introducing the Center for Research on Foundation Models (CRFM)*. 2021. URL: https://hai.stanford.edu/news/introducing-center-research-foundation-models-crfm.

[79] Rishi Bommasani et al. "On the Opportunities and Risks of Foundation Models". In: *arXiv.org* (2022). DOI: 10.48550/arXiv.2108.07258.

[80] Bruno Da Silva, George Konidaris, and Andrew Barto. "Learning Parameterized Skills". In: *arXiv.org* (2012). DOI: 10.48550/arXiv.1206.6398.

[81] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *International Conference on Machine Learning*. 2021. DOI: 10.48550/arXiv.2103.00020.

[82] Kaiming He et al. "Masked Autoencoders Are Scalable Vision Learners". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 15979–15988. DOI: 10.1109/CVPR52688.2022.01553.

[83] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv.org* (Oct. 2020). DOI: 10.48550/arXiv.2010.11929.

[84] Lucas Prado Osco et al. "The Segment Anything Model (SAM) for remote sensing applications: From zero to one shot". In: *International Journal of Applied Earth Observation and Geoinformation* 124 (2023), p. 103540. ISSN: 1569-8432. DOI: 10.1016/j.jag.2023.103540.

[85] Xiangtai Li et al. "Transformer-Based Visual Segmentation: A Survey". In: *T-PAMI* (2024). DOI: 10.1109/TPAMI.2024.3434373.

[86] Kunchang Li et al. "UniFormer: Unifying Convolution and Self-Attention for Visual Recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 45.10 (Oct. 2023), pp. 12581–12600. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2023.3282631.

[87] Abdulaziz Amer Aleissaee et al. "Transformers in Remote Sensing: A Survey". In: *Remote Sensing* 15.7 (2023). DOI: 10.3390/rs15071860.

[88] Yanghao Li et al. "Exploring Plain Vision Transformer Backbones for Object Detection". In: *ArXiv* abs/2203.16527 (2022). DOI: 10.48550/arXiv.2203.16527.

[89] Renrui Zhang et al. "Personalize Segment Anything Model with One Shot". In: *ArXiv* (May 2023). DOI: 10.48550/arXiv.2305.03048.

[90] Bertrand Chauveau and Pierre Merville. *Segment Anything by Meta as a foundation model for image segmentation: a new era for histopathological images.* 2023. DOI: 10.1016/j.pathol.2023.09.003.

[91] Mohanad Albughdadi et al. “Exploring Transfer Learning Using Segment Anything Model in Optical Remote Sensing”. In: *EGU General Assembly* (2024). DOI: 10.5194/egusphere-egu24-5769.

[92] Alberto Carraro, Marco Sozzi, and Francesco Marinello. “The Segment Anything Model (SAM) for accelerating the smart farming revolution”. In: *Smart agricultural technology* (2023). DOI: 10.1016/j.atech.2023.100367.

[93] Iraklis Giannakis et al. “Segment Anything Model (SAM) for Automatic Crater Detection”. In: *EGU General Assembly* (2024). DOI: 10.5194/egusphere-egu24-21146.

[94] Shilong Liu et al. “Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection”. In: (2024). DOI: 10.48550/arxiv.2303.05499.

[95] Yan-Yan Song and Ying Lu. “Decision tree methods: applications for classification and prediction”. In: *Shanghai archives of psychiatry* 27 (Apr. 2015), pp. 130–5. DOI: 10.11919/j.issn.1002-0829.215044.

[96] Xie Niuniu and Liu Yu-xun. “Notice of Retraction Review of decision trees”. In: 2010. DOI: 10.1109/ICCSIT.2010.5564437.

[97] Mourad Azhari et al. “A Comparison of Random Forest Methods for Solving the Problem of Pulsar Search”. In: 2019. DOI: 10.1007/978-3-030-37629-1_57.

[98] Chia-Hsiu Chen, Kenichi Tanaka, and Kimito Funatsu. “Random Forest Model with Combined Features: A Practical Approach to Predict Liquid-crystalline Property”. In: *Molecular Informatics* (2019). DOI: 10.1002/MINF.201800095.

[99] Min Wu et al. “Random forest predictive model development with uncertainty analysis capability for the estimation of evapotranspiration in an arid oasis region”. In: *Hydrology Research* (2020). DOI: 10.2166/NH.2020.012.

[100] Joseph Galasso, Duy M. Cao, and Robert Hochberg. “A random forest model for forecasting regional COVID-19 cases utilizing reproduction number estimates and demographic data”. In: *Chaos, Solitons & Fractals* 156 (2022), p. 111779. ISSN: 0960-0779. DOI: 10.1016/j.chaos.2021.111779.

[101] Arkajyoti Saha, Sumanta Basu, and Abhirup Datta. “Random Forests for dependent data”. In: *arXiv: Machine Learning* (2020). DOI: 10.48550/arXiv.2007.15421.

[102] Connie Ko et al. “Hybrid Ensemble Classification of Tree Genera Using Airborne LiDAR Data”. In: *Remote Sensing* (2014). DOI: 10.3390/RS61111225.

[103] Andreas Merentitis et al. "Automatic fusion and classification of hyperspectral and LiDAR data using random forests". In: 2014. DOI: 10.1109/IGARSS.2014.6946658.

[104] Mathieu Guillame-Bert et al. "Yggdrasil Decision Forests: A Fast and Extensible Decision Forests Library". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*. 2023, pp. 4068–4077. DOI: 10.1145/3580305.3599933.

[105] Stephen V. Stehman. "Selecting and interpreting measures of thematic classification accuracy". In: *Remote Sensing of Environment* 62.1 (1997), pp. 77–89. ISSN: 0034-4257. DOI: 10.1016/S0034-4257(97)00083-7.

[106] José Hernández-Orallo, Peter A. Flach, and Cèsar Ferri. "A unified view of performance metrics: translating threshold choice into expected classification loss". In: *Journal of Machine Learning Research* 13.1 (Oct. 2012), pp. 2813–2869. DOI: 10.5555/2503308.2503332.

[107] Mehdi S. M. Sajjadi et al. *Assessing Generative Models via Precision and Recall*. 2018. DOI: 10.48550/arXiv.1806.00035.

[108] Abdel Aziz Taha and Allan Hanbury. "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool". In: *BMC Medical Imaging* 15.1 (Aug. 2015), p. 29. ISSN: 1471-2342. DOI: 10.1186/s12880-015-0068-x.

[109] Usman Ali et al. "Review of urban building energy modeling (UBEM) approaches, methods and tools using qualitative and quantitative analysis". In: *Energy and Buildings* 246 (2021), p. 111073. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2021.111073.

[110] Manqing Dong et al. "Gradient Boosted Neural Decision Forest". In: *IEEE Transactions on Services Computing* 16.1 (2023), pp. 330–342. DOI: 10.1109/TSC.2021.3133673.

[111] Zahra Agusta and Adiwijaya Adiwijaya. "Modified balanced random forest for improving imbalanced data prediction". In: *International Journal of Advances in Intelligent Informatics* 5.1 (2019), pp. 58–65. ISSN: 2548-3161. DOI: 10.26555/ijain.v5i1.255.

[112] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *ArXiv* (2015). DOI: 10.48550/arXiv.1511.08458.