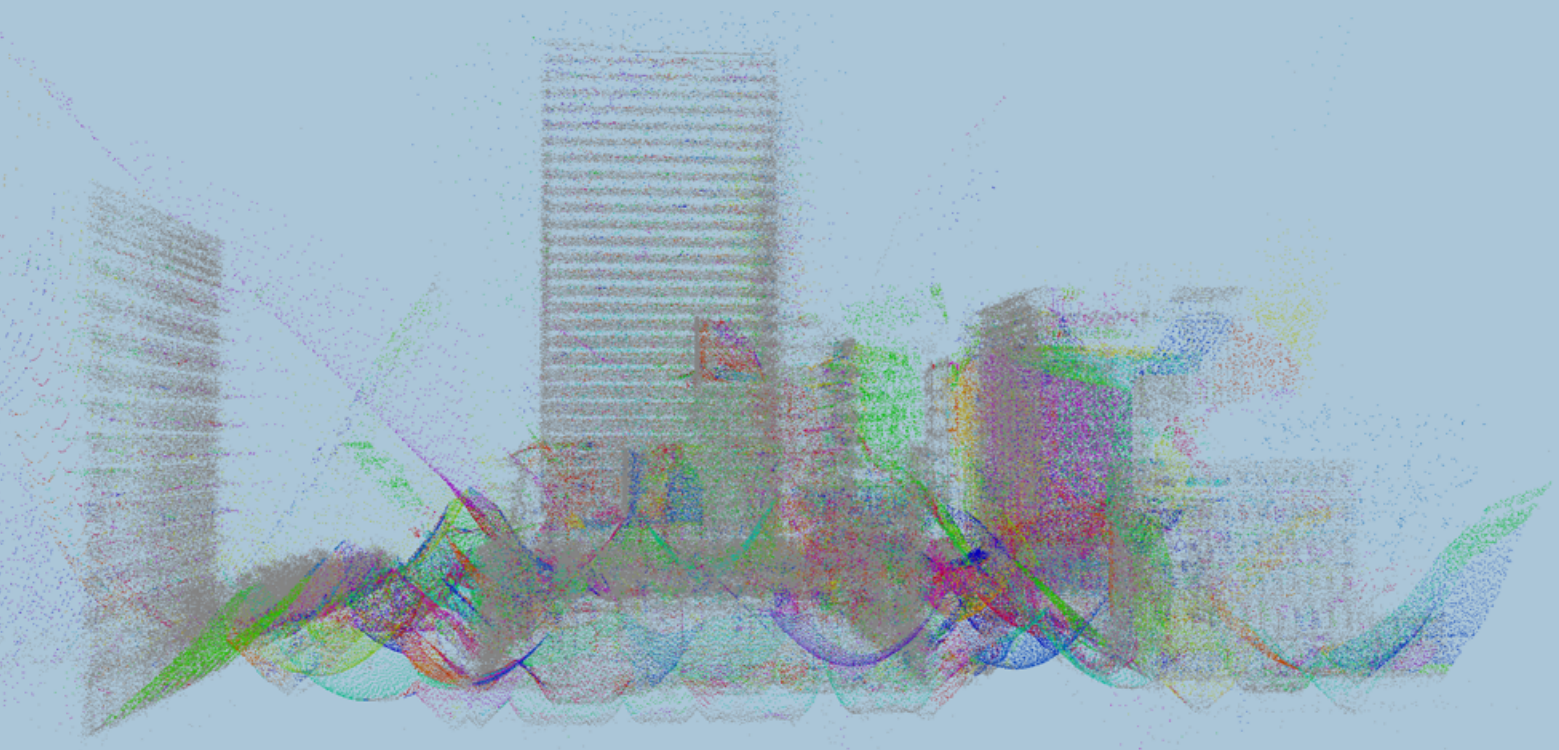
MSc thesis in Geomatics for the Built Environment

Semantic segmentation of point clouds with the 3D medial axis transform

Giulia Ceccarelli
July 2020



MSc thesis in Geomatics

Semantic segmentation of point clouds with the 3D medial axis transform

Giulia Ceccarelli

July 2020

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in collaboration with:

3D geoinformation group
Department of Urbanism
Faculty of the Built Environment & Architecture
Delft University of Technology

CycloMedia Technologies
R&D Cluster Data Analytics

Supervisors - TU Delft:	Dr. Ravi Peters
	MSc Weixiao Gao
Supervisors - CycloMedia:	Dr. Bas Boom
	MSc Arjen Swart
Co-reader:	MSc Zexin Yang

Abstract

A point cloud is a representation of shapes, organized in a 3D irregular structure. Point clouds are increasingly used in different applications, ranging from architectural preservation to computer vision. The 3D medial axis transform is a topology preserving, skeleton representation of shapes. It can be used to decompose an object in meaningful parts and to describe local and long range information of points in a point cloud.

In the past years, many deep learning methods for point clouds emerged. These are used for different applications, such as shape classification, object detection or semantic segmentation. In particular, the latter aim to classify each point in the input point cloud in subsets, based on their semantics.

This research investigates the integration of the 3D medial axis transform (MAT) in two deep learning methods for point clouds' semantic segmentation, PointNet++ and Superpoint Graph. In particular, the 3D MAT was used in PointNet++ as a point feature, to give context to local points. Then, it was used in Superpoint Graph as a geometric descriptor to partition a point cloud and as a edge feature in the superpoint graph (SPG).

The major findings of this research outline that the 3D MAT can be successfully used in PointNet++ as a point feature, improving the overall accuracy and loss values of the algorithm. Particularly two MAT derived properties used in this research output positive results, radii and separation angles. These can be combined with point coordinates and RGB information to bring additional knowledge on the geometry of the shape, representing its curvature and thickness. Furthermore, they can be integrated in a simple and effective way, without increasing computational or time effort in the algorithm.

The analysis carried out in Superpoint Graph depicts that the 3D MAT does not improve the initial geometric partition. In fact, adding geometric descriptors to the algorithm increases the difficulty in dividing the point cloud into simple shapes, creating artifacts. Furthermore, adding MAT information on superedges does not give added value to the SPG graph. The reason is that the SPG graph and the structured MAT are different than each other, in practice, as nodes represent diverse parts in the point cloud.

Contents

List of Figures

List of Tables

Acronyms

FPS farthest point sampling 29
IoU intersection over union 6
MAT medial axis transform v
MLP multi layer perceptron 8
MLS mobile laser scanning 1
MRG multi resolution grouping ix
MSG multi scale grouping ix
OA overall accuracy 6
SPG superpoint graph v

1 Introduction

A point cloud is a 3D representation of reality consisting of a set of points and additional information, such as color and intensity. In the past few years, the growing acquisition capabilities of instruments led to the increasing availability of point clouds. These are of high importance in various applications, ranging from architecture, surveying and heritage preservation to autonomous driving. A practical example comes from CycloMedia Technology, where point clouds are the backbone of an online tool for accurate urban analysis and measurements. To this end, a point cloud must be augmented with semantics.

Different point clouds acquisition techniques exist; mobile laser scanning (MLS) uses a Lidar scanner applied on a terrestrial moving platform. The acquisition method determines the quality of a point cloud, together with factors related to the scanned environment. For example in a urban location, buildings could not be fully represented due to the presence of artifacts or moving objects. Additionally, the distribution of objects in the urban environment influences the distribution and density of points. In fact, closer objects would be represented by many points; instead those far away would be scarcely represented.

Nowadays, various deep learning methods are emerging for semantic segmentation in point clouds, which aims to classify each point P_i of a point cloud P between K classes. These have to consider factors such as points density and artifacts. Furthermore, while deep learning methods have been widely used for structured inputs, such as images, a point cloud is an unordered set of points.

This property introduces a number of difficulties; a deep learning architecture that takes as input n 3D points, must be invariant to $n!$ permutations of the input set in data feeding order. Furthermore, a point cloud is invariant under certain geometrical transformations, as rotation and translation. Consequently the deep learning architecture must have the same properties. Last in a point cloud, points interact in space with a distance metric and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the interactions among local structures, [Qi et al., 2016].

To this end, the 3D MAT shows interesting properties. The 3D MAT is a skeleton representation of shapes, dual to the boundary of an object. This representation models the key properties of a shape and its topology in an explicit way, [Peters, 2018a]. Thus, the 3D MAT can be used as a shape descriptor, to organize and structure a point cloud in meaningful subsets.

The aim of this research is to develop a methodology to integrate the properties of the 3D medial axis transform in a point cloud semantic segmentation deep learning algorithm, understanding if MAT information can be exploited to improve results in existing deep learning methods.

1.1 Research questions

The main goal of this research is to integrate an existing deep learning algorithm with the 3D medial axis transform. This research is based on the previous study *Geographical point cloud modeling with the 3D medial axis transform* [Peters, 2018a], that analyzes the construction and application of the 3D medial axis transform for geographical point clouds. This thesis aims to investigate which information derived from the 3D MAT can be integrated in a deep learning algorithm for semantic segmentation, improving its results. In [Peters, 2018a] the 3D MAT is computed for airborne laser scanning point clouds, instead, the focus of this thesis is on mobile based point clouds. Thus, this thesis also aims to analyze how the 3D MAT is constructed for these point clouds.

The main research question for this project is:

How can the properties of the 3D medial axis transform be exploited in different deep learning methods for point cloud semantic segmentation?

The following sub-questions will also be relevant:

- How can the 3D medial axis transform be used to give context to local points in a point cloud, making the unary classification per point stronger? Which of the 3D medial axis transform properties are most helpful?
- Can the 3D medial axis transform be used to partition a point cloud into semantically homogeneous shapes? How can the 3D medial axis transform be used to enrich the node and edge information in a graph used as input for a graph convolution neural network?
- Can the 3D medial axis transform be used to improve the accuracy of an existent deep learning method?
- How important are the construction parameters of the 3D medial axis transform in the deep learning method?
- How does the performance on the real data-set compare with the one obtained on the synthetic one?

In order to answer these research questions, a thorough literature research was carried out, together with the in-depth analysis of the deep learning methods to be integrated with the 3D MAT. Furthermore, a number of different experiments involving the 3D MAT was performed. Last, the evaluation of the outcomes was carried out through accuracy and intersection over union metrics on all data-sets.

1.2 Research scope

This research focuses on the analysis of the properties of the 3D MAT under two perspectives. First, it includes the augmentation of a point based deep learning architecture, PointNet++ [Qi et al., 2017], with features derived from the 3D MAT. The point wise information obtained is combined with each point of the algorithm's input point cloud. This step aims to study point properties associated with the 3D MAT; thus, deep learning methods that need a regularization of the input, such as voxelization and 2D images, are not considered.

Second, it covers the study of point and graph properties of the 3D MAT in a graph based deep learning architecture, Superpoint Graph [Landrieu and Simonovsky, 2017]. In this algorithm, the 3D MAT information is first used to partition the input point cloud in homogeneous parts, defined as superpoints. These are then used to construct a graph where relations between nodes are defined as superedges. Here, the 3D MAT is used as an additional feature to superpoints and superedges. The aim is to investigate whether the 3D MAT can be used to partition a point cloud and if it can add useful knowledge in a graph convolution algorithm.

Last, this project focuses on mobile based point clouds, obtained with laser scanning or dense image matching. The reason for these choices is twofold. First the main point clouds used in this research are obtained from mobile based Lidar; second the medial axis transform will be more complete as materials such as glass will be represented. Specific studies on airborne point clouds are out of scope for this research.

1.3 Thesis outline

The next chapters of this thesis are structured as follows,

- Chapter 2 provides a theoretical background on deep learning, its main concepts and evaluation methods. Furthermore it illustrates deep learning on point clouds, focusing its applications and main methodologies. Last, it illustrates the definitions and computation procedures of the 3D medial axis transform.
- Chapter 3 outlines the methodology carried out in this research. In particular, it gives details on the preliminary steps, and the computation parameters for the 3D MAT. Then it describes the three integration methods studied, in Section 3.3 3D MAT as a feature, in Section 3.4 3D MAT to partition a point cloud and in Section 3.5 3D MAT as a graph attribute for graph convolutions. Last, the evaluation procedure is described.
- Chapter 4 gives information on the three data-sets used, 3DOM, SynthCity and CycloMedia's internal data-set. Then it lists the main tools used in this research.
- Chapter 5 illustrates the results of the main experiments conducted in the three phases of the research, together with a reflection on the meaning of each output.
- Last, Chapter 6 summarizes the main results obtained and provides critical answers to the proposed research questions. Then, it lists the main ideas on future work on related to this field of study.

2 Theoretical background

This chapter provides theoretical knowledge on the topics discussed in the later chapters on this thesis. Section 2.1 gives an overview on the main concepts related to deep learning; it outlines how a neural network is built and evaluated. Section 2.2 introduces deep learning in point clouds, its difficulties and main applications. Section 2.3 proposes an overview of deep learning methods for semantic segmentation, with a focus on point based and graph based neural networks. Section 2.4 illustrates the properties of the 3D medial axis transform, its construction methods and terminology. Topics treated in this Chapter build the foundation of this study, giving a background on choices made during the course of the research.

2.1 Concepts of neural networks

Deep learning methods are a subset of machine learning ones, characterized by the presence of layers in which data is processed. These reorganize data in increasingly complex representations before obtaining a meaningful output. In a deep learning algorithm, the number of neurons and layers, and their connections define the model or architecture of the neural network, which is represented by a directed acyclic graph. The architecture's constraints define a mapping from input to output through each neuron, identifying the purpose of the neural network. Chollet [2017]

A neural network takes an input, passes it through multiple layers and outputs a prediction based on the combined information of all the layers, as shown in Figure 2.1. A neural network is parametrized by its weights, which define the function of each layer. Thus, learning means finding the most appropriate weights for each layer. To do so, the difference between the predicted output and the true values has to be quantified. This is done through a loss function which outputs a loss score. Last, the values of the weights are updated to minimize the loss value, through the optimizer. This process occurs iteratively until the desired output is found.

2.1.1 Evaluation metrics

The final amount of correct predictions of a deep learning algorithm can be quantified through a confusion matrix. In the confusion matrix, diagonal elements represent the correctly predicted ones. With respect of each class, non-diagonal row elements are defined false positives, while column ones are false negatives. The former are incorrectly predicted due to an error of commission, these elements are assigned to a specific class but belong to

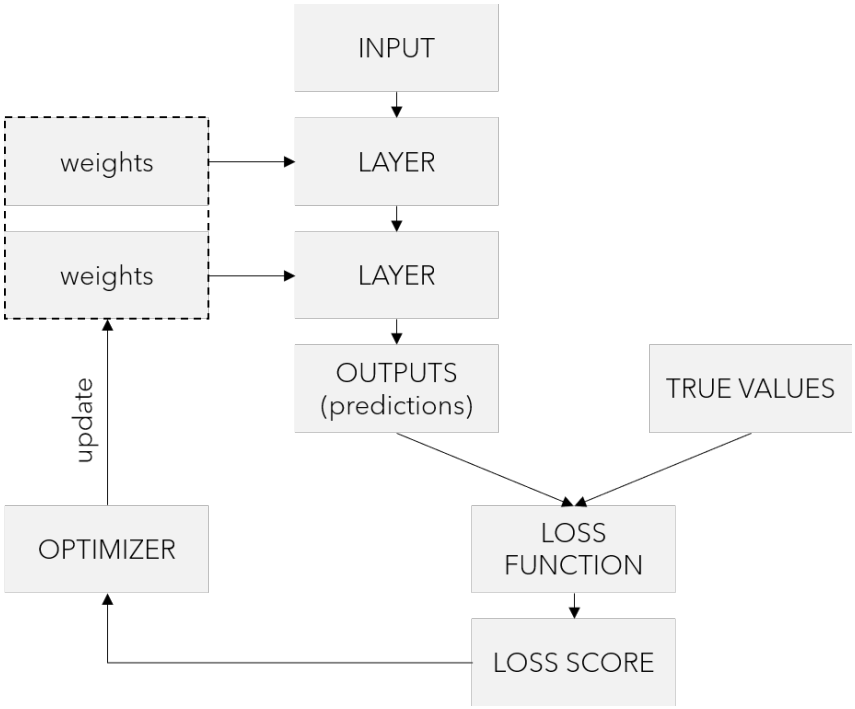


Figure 2.1: Neural network simplified flowchart - the input is processed in layers and produces an result. This is compared with the true values in the loss function, which outputs the loss score. This is then used to optimize the weights of each layer. [Chollet, 2017]

another. The latter are produced by an error of omission, they refer to elements that should have been assigned to a class but were erroneously labeled as another.

From the confusion matrix, different evaluation metrics can be computed. Two of these are the overall accuracy (OA) and the intersection over union (IoU). The overall accuracy is computed as the sum of the diagonal elements of the confusion matrix, divided by the total number of elements. This metric gives an overall clue of the performance of the algorithm for all classes.

$$OA = \frac{Truepositives}{Total}$$

The intersection over union (IoU) is a per class metric that is calculated as the ratio between the correct predictions, defined as true positives, and the sum between the false positives, false negatives and true positives. This metric enables the analysis of the results with a focus on each class, outlying the strengths and weaknesses of the algorithm.

$$IoU = \frac{TruePositives}{TruePositives + FalseNegatives + FalsePositives}$$

2.2 Deep learning in point clouds

The following paragraph is extracted from [Qi et al., 2016]. A point cloud is a subset of points from an Euclidean space, which has three main properties.

- *Unordered* Unlike pixel arrays in images or voxel arrays in volumetric grids, a point cloud is a set of points without specific order. In other words, a network that consumes n $3D$ point sets needs to be invariant to N! permutations of the input set in data feeding order.

- *Interaction among points* The points are from a space with a distance metric. It means that points are not isolated, and neighboring points form a meaningful subset. Therefore, the model needs to be able to capture local structures from nearby points, and the combinatorial interactions among local structures.

- *Invariance under transformations* As a geometric object, the point set is invariant to certain transformations. For example, rotating and translating points all together does not prevent a person to understand what the point cloud represents.

The above mentioned are intrinsic properties of point clouds. They have to be considered as requirements when implementing a point cloud algorithm. For example, this means that an algorithm should be able to assign the correct labels to the point cloud if all its points are rotated to a certain angle. Furthermore, additional factors have to be considered when implementing point clouds' algorithms, such as:

- *Complexity of observed scenes* In the scope of this research, point clouds represent complex urban scenes. The amount of objects that are captured and their variability in shape and dimension are factors that increase the difficulty when implementing a point cloud algorithm.

- *Occlusion* When capturing a point cloud in the urban environment, it is almost impossible to obtain a full representation of objects, such as buildings, due to the presence of other unwanted elements, for example artifacts, moving objects (pedestrians, cars) or street furniture. This means that one has to deal with the lack of information when implementing a point algorithm.

- *Irregularity in point distribution* When capturing a point cloud using a Laser Scanner, light beams are sent toward the objects and reflected back to the instrument. This information is used to compute the distance and angle of objects. The distribution of objects in the urban environment influences the distribution and density of captured points. In fact, closer objects would be first hit by the light beam, thus they would be represented by many points; instead those far away would be scarcely represented.

- *Amount of data captured* Point clouds representing urban environments are made of a huge quantity of points. To process them is highly computationally and time demanding. One has to develop strategies to reduce the weight of the point cloud without removing important information.

2.2.1 Applications

Deep learning algorithms are used on point clouds for different purposes, each one requires a different architecture.

3D oversegmentation methods are used to partition the point cloud into simple shapes, relying on the geometric attributes of neighboring points.

3D shape classification methods are used to classify each point in the point cloud among a set of classes. Then, these are aggregated to extract a class for the whole point cloud.

Classification: classify the point cloud among class set K.

$$f\colon P \mapsto k \mid k \in K$$

3D object detection methods are used to locate a given category of objects in a scene. Tracking methods aim to estimate the location of these objects in subsequent frames.

Object detection: cluster the point cloud in C parts/object.

$$f\colon p \mapsto c \mid p \in P, c \in C$$

3D point cloud segmentation methods can refer to semantic and instance segmentation. The first methods aim to classify the point cloud in subsets based on their semantics; the second perform semantic segmentation and additionally learn a label for each instance.

Semantic segmentation: classify each point of a point cloud between K classes.

$$f\colon p \mapsto k \mid p \in P, k \in K$$

Instance segmentation: cluster the point cloud into semantically characterized objects.

$$f\colon p \mapsto c \mid p \in P, c \in C$$
$$m\colon c \mapsto k \mid c \in C, k \in K$$

where $P = \{p_1, p_2, ..., p_n\}$ is the set of n points of the point cloud; $C = \{c_1, c_2, ..., c_m\}$ is the set of m objects or parts and $K = \{k_1, k_2, ..., k_l\}$ is the set of l classes. This research will focus on semantic segmentation.

2.3 Semantic segmentation in point clouds

Deep learning methods for semantic segmentation can be categorized in four main classes [Thomas et al., 2019], projection networks, point-wise multi layer perceptron (MLP) networks, graph convolution networks and point convolution networks.

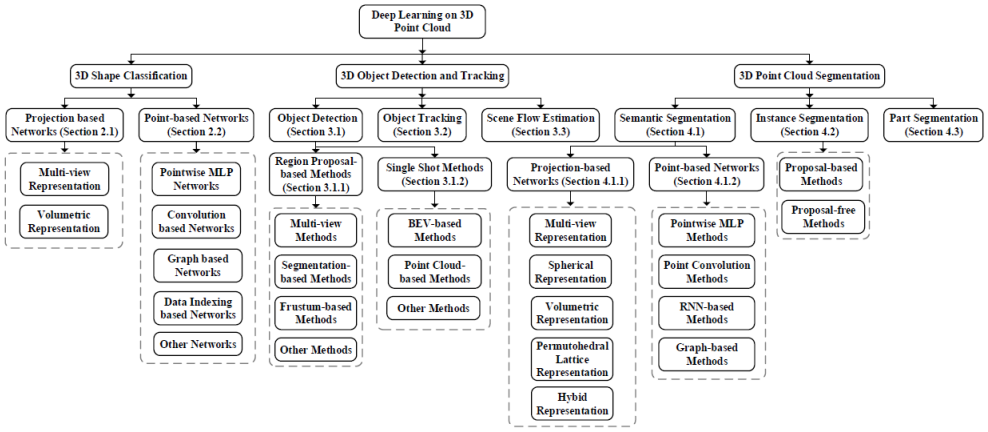


Figure 2.2: Deep learning in point clouds, [Guo et al., 2019]

2.3.1 Projection networks

In projection networks, the input point cloud is projected into a regular grid structure, such a 2D image or a 3D voxel. Different projection methods to 2D images exist, such as multi-view and spherical representations. In multi-view methods, the point cloud is rendered in a set of 2D images from different viewpoints; these encode information such as depth and normal vectors. Each pixel is then labeled using a 2D segmentation network. [Su et al., 2015] [Boulch et al., 2017]

Voxel based methods can be divided into dense or sparse representation methods. In the latter, the point cloud is discretized in a set of occupancy voxels, which are labeled using a 3D convolutional neural network. Finally, each point belonging to the same voxel is assigned the voxel's label. [Guo et al., 2019] Different methods exist to produce fine-grained results, an example is SEGCloud where the voxel predictions are propagated back to the point cloud using a fully connected network. [Tchapmi et al., 2017] Sparse representation networks aim to reduce memory and computation inefficiency of the dense methods. An example is SPLATNet, where the point cloud is projected into a permutohedral sparse lattice. A bilateral convolution method is applied to occupied parts only; finally, the output is interpolated back to the point cloud. [Su et al., 2018a]

2.3.2 Point-wise MLP networks

The first point-wise MLP network is PointNet, this algorithm uses a shared MLP on every point followed by max-pooling on all points. [Thomas et al., 2019] A MLP is a neural network with multiple fully-connected layers that use nonlinear activation functions to deal with data which is not linearly separable. [WILDML, 2019] These networks are able to approximate any continuous function. A max-pooling layer selects the maximum value from a patch of features. It helps to reduce the dimensionality of a representation by keeping only the most salient information. [WILDML, 2019]

After PointNet, different hierarchical architectures were developed to combine local neighborhood information at different scales. An example is PointNet++. This algorithm applies PointNet recursively on nested subsets of the input, learning from successively larger regions of the input. [Qi et al., 2017] Another example is PointSIFT [Jiang et al., 2018], where the SIFT module joins information from eight spatial orientations at multiple scales. [Guo et al., 2019] Last, RandLA-Net proposes a method for processing large point clouds, which is based on the random selection of points combined with a feature aggregation module to preserve local geometric information. [Hu et al., 2019]

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

PointNet is one of the first algorithms to take as input a point cloud and to output class labels or semantic segment labels per point. "Its basic idea is to learn a spatial encoding of each point and then aggregate all individual point features to a global point cloud signature." [Qi et al., 2017] A spatial encoding is a description of the point's characteristics using gradients. In this method, the authors use a symmetric function to identify informative points. A symmetric function is one that gives an output that is invariant to the order of the input, an example is the *max* function. Informative points are those that present the most outstanding characteristics. According to the authors, they correspond to the skeleton of an object based on to visualization. [Qi et al., 2016] Last, they use a fully connected layer to aggregate information learned in the network in a global descriptor. A global descriptor summarizes the information on the full geometry of the point cloud.

In PointNet points are simply represented by their coordinates and extra feature channels like color and normal vectors. [Qi et al., 2016] The network has three modules. “Input points are first aligned by a Spatial Transformer Network, independently processed by multi-layer perceptrons (MLPs), and max-pooled to summarize the shape." [Landrieu and Simonovsky, 2017] Point alignment is used to give the model rigid transformations invariance properties. To do so, the authors use a mini-network (Spatial Transformer Network) that applies an affine transformation matrix to the input. The idea is to “to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set". [Qi et al., 2016]

$$f(x_1, ..., x_n) \approx g(h(x_1), ..., h(x_n))$$

where *h* represent a multi layer perceptron and *g* is a combination of a single variable function and a max pooling function. [Qi et al., 2016] The *max* function gives the model permutation invariance properties; it takes as input *n* vectors and outputs a vector invariant to the input order. Qi et al. [2016] This is a global feature vector that describes the signature of the whole input. It is used in classification tasks to train a multi-layer perceptron. In semantic segmentation tasks, it is fed back to the single points to extract new point features, combining local and global information. Qi et al. [2016] Figure 2.3 shows PointNet architecture for semantic segmentation.

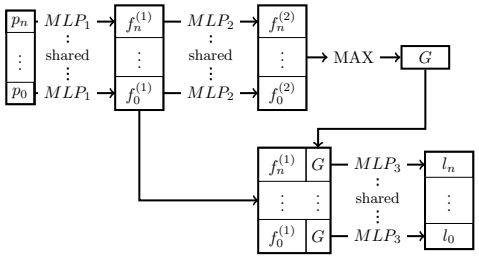


Figure 2.3: PointNet architecture for semantic segmentation [Qi et al., 2016] [Landrieu, 2019b]

PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space

PointNet++ is an extension of PointNet that applies the latter recursively on nested subsets of the input. This method is based on a hierarchical structure, which is composed by *set abstraction* levels made of a sampling layer, a grouping layer and a PointNet layer.

- In the *sampling layer*, input points are subsampled using farthest point sampling (FSP). This method chooses a new point which is the farthest from the rest of the points in the set. [Qi et al., 2017]
- In the *grouping layer*, the input is a set of points and a set of centroids. The output are groups of points corresponding to local regions around a given centroid. The neighbors of a centroid can be found using k-nearest neighbors or ball query search algorithms. [Qi et al., 2017]
- In the *PointNet layer*, local regions of points are converted into feature vectors of fixed length. First, point coordinates are described in relation to the centroid, then PointNet is used to extract the signature of the local region. [Qi et al., 2017]

Abstraction layers aim to be density adaptive, combining features at different scales according to their density. Two methods to group points used in PointNet++ are MSG and MRG, Figure 2.4:

- MSG consists of a grouping layers with different scales that build around the same centroid, and concatenating their features in a multi-scale feature.
- MRG is obtained as a combination of feature vectors at different levels. Given a local region, one feature vector is obtained from the information of all its points. The other is obtained as a combination of information of overlapping sub-regions.

Last, features have to be applied to all points in the point cloud. Thus the authors make use of a *feature propagation* layer based on inverse distance weighted average based on *k* nearest neighbors.

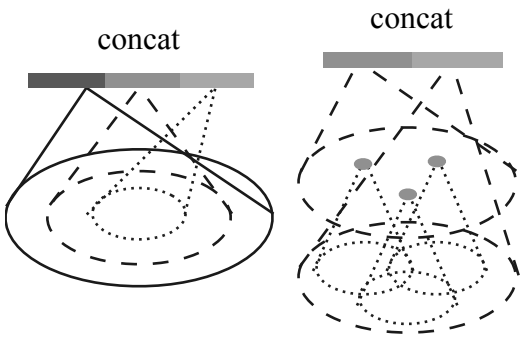


Figure 2.4: PointNet++ - left - multi-scale grouping, right - multi-resolution grouping [Qi et al., 2017]

2.3.3 Graph convolution networks

Graph convolution networks learn the weights on graph edges instead of points, an example is Superpoint Graph. Here, first the whole point cloud is partitioned geometrically in simple shapes (superpoints), which are structured in the Superpoint Graph. Then, a deep learning architecture is implemented. It consists of PointNets for superpoints embedding and graph convolutions for contextual segmentation. [Landrieu and Simonovsky, 2017] Other examples are PyramNet and PointGCR. In the latter, the input point cloud is structured in a directed acyclic graph, then a similarity matrix is constructed, describing the relations between points. Last, semantic features are assigned to points using convolution kernels in the PAN operator. [Kang and Ning, 2019] In PointGCR, an undirected graph is used to learn information on points along the channel dimension. [Ma et al., 2020]

Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs

This deep learning method represents the state of the art for outdoor and indoor LiDAR scans, its main aim is to tackle the limitations related to the size of point clouds.

The procedure is split in three steps:

- First, the whole point cloud is partitioned geometrically in simple shapes, defined superpoints, this is an unsupervised step. The resulting superpoints are structured in a SuperPoint Graph. This is an attributed directed graph where nodes represent the superpoints, while edges represent their adjacency relationship.
- Superpoints are assumed to be semantically homogeneous. Thus, they are downsampled to a maximum 100 points each.
- Third, a deep learning architecture is implemented. It consists of PointNets for superpoints embedding and graph convolutions for contextual segmentation.

The point cloud is described as a set of *n* 3D points, each characterized by its 3D position and additionally by color and intensity. For each point, the authors compute a set of features to describe their neighborhood: linearity, planarity, scattering, verticality. Furthermore, they compute the elevation of points normalized over the input point cloud. These features are used to partition the point cloud in simple shapes. Later, the authors identify an additional set of superedge features to describe their adjacency relationships. They are based on a symmetric Voronoi adjacency graph, on the size of the superpoint and from its eigenvalues. Figure 2.5 shows a list of these features and their description.

Feature name	Size	Description
mean offset	3	$\mathrm{mean}_{m \in \delta(S,T)}\ \delta_m$
offset deviation	3	$\mathrm{std}_{m \in \delta(S,T)}\ \delta_m$
centroid offset	3	$\mathrm{mean}_{i \in S}\ p_i - \mathrm{mean}_{j \in T}\ p_j$
length ratio	1	$\log \mathrm{length}\,(S)\,/\mathrm{length}\,(T)$
surface ratio	1	$\log \mathrm{surface}\,(S)\,/\mathrm{surface}\,(T)$
volume ratio	1	$\log \mathrm{volume}\,(S)\,/\mathrm{volume}\,(T)$
point count ratio	1	$\log \lvert S\rvert / \lvert T\rvert$

Figure 2.5: Superpoint graph - superedge features [Landrieu and Simonovsky, 2017]

PointNet is then used to learn superpoint embedding, each superpoint is thus scaled to unit sphere and represented by its normalized position, observation and geometric features of its elements. Then, its original metric diameter is added as a feature in the max-pooling step to ensure it is covariant with shape sizes. Last, superpoint embedding and information on its neighborhood are used in the graph convolution step for segmentation. In this step, superpoints refine their embedding based on the information of the superedges [Landrieu and Simonovsky, 2017].

2.3.4 Point convolution networks

Last, point convolution networks use a convolution kernel directly on the point cloud. Convolution kernels are linear filters that combine data of local neighborhoods. In point clouds, neighboring points are not uniquely defined; for each point one has to compute them, usually through a radius search, and define their influence based on their distance. [TERRA3D, 2019] An example is a point-wise convolutional operator that locates neighboring points in a kernel cells and convolves them with kernel weights. [Hua et al., 2017] Another example is KPConv, where convolution weights of points are computed as the Euclidean distance to kernel points. In this algorithm, the position of kernel points is not fixed, but it is the result of an optimization problem for the best coverage in a spherical space. [Thomas et al., 2019]

2.4 3D medial axis transform

This section is extracted from Geographical point cloud modeling with the 3D medial axis transform. It aims to give theoretical background on the 3D medial axis transform, pinpointing what properties can be exploited in a deep learning algorithm for semantic segmentation.

2.4.1 Definitions of the medial axis transform

The 3D medial axis transform is a representation of the shape of an object, dual to its boundary. It is a way to decompose an object into meaningful parts, preserving its mathematical meaning. The medial axis transform of an object consists of the set of its interior and exterior medial balls, Figure 2.6. The former are inside the object and define its skeleton, while the latter are outside of the boundary, defining the relation between different objects. Medial balls are defined by their maximality property; this states that a medial ball can not be contained by any other medial ball. [Peters, 2018a] Consequently, every medial ball is tangent to the boundary of a shape in two or more points and it never intersects it, also every medial ball is empty. A medial atom is a tuple of the center and radius of the same medial ball, consequently the MAT is the set of all the medial atoms of an object, Figure 2.7. The atom's local geometry can quantify the local characteristics of a shape like thickness and curvature, it makes the interaction between the atom and the corresponding surface points explicit and can be used to define a local coordinate system. [Peters, 2018a]

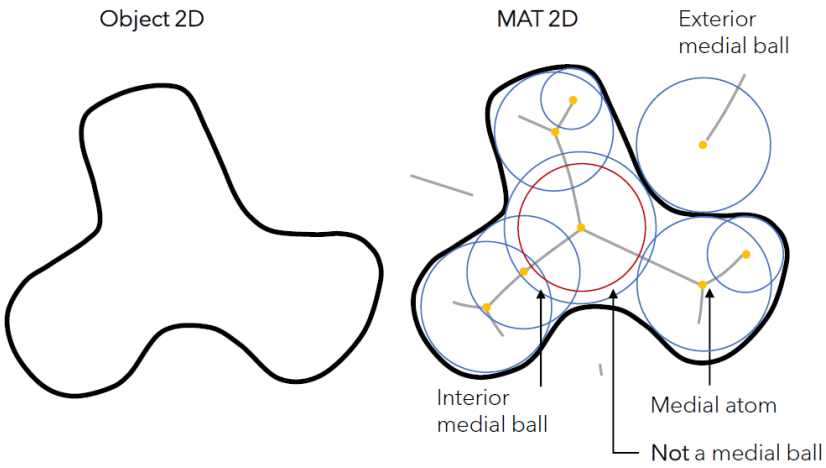


Figure 2.6: MAT - interior and exterior medial balls of a given shape

Below the definitions of the main components of the local geometry of the medial axis transform, represented in Figure 2.7:

– medial point: the center of the medial ball

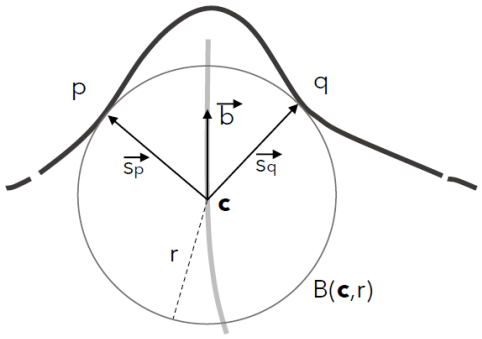


Figure 2.7: MAT - local geometry of the medial atom

- radius: the radius of the medial ball
- medial ball: the medial ball constructed with the medial point and radius
- feature point: the point on the boundary of the object corresponding to the medial point and radius
- spoke vector: the vector from the medial point to the feature point
- medial bisector: a unit vector that bisects the two spoke vectors
- separation angle: the angle between the spoke vectors

The medial axis transform has different properties, for example, it is topology preserving; this means that the 3D MAT has the same number of connected components of the boundary of the object. Furthermore it is compact, so its components are at most of dimensionality d-1 with respect to the dimensionality of the boundary. Last, it has a hierarchical composition that enables an easy traversal of its parts. [Peters, 2018a]

2.4.2 The unstructured MAT

The exact computation of the 3D MAT is difficult and computationally expensive [Peters, 2018a]; the fastest approximation method for the 3D MAT is the ball-shrinking algorithm. This takes an oriented point cloud as input, where points are associated with their normal vectors, and outputs a set of medial atoms as the MAT approximation. Thus, the output is a point cloud, called unstructured MAT, in fact its topology and structure are not computed. In particular, the unstructured MAT corresponds to the set of medial balls, represented by the medial atoms and attributed by the radius and the feature points. The algorithm takes into consideration that a medial ball is tangent to the boundary of the object but never intersects it and that it is always empty. Also, the assumption of the ball shrinking algorithm is that the center of a medial ball corresponding to a boundary point lies along its normal vector. The algorithm is initialized with a large medial ball that is iteratively shrunk until it doesn't contain other boundary points. [Peters, 2018a]

The main factor that influence the output of the algorithm is the normal vectors computation method. This influences the location of the medial atoms as well as the orientation of the interior and exterior medial axis. Other factors of high importance are the initial radius of the medial ball and two heuristic values, the denoise planar value and the denoise preserve one. The former is of special importance for geographical point clouds as it determines the degree of interaction between objects. The other elements help increasing the robustness of the ball-shrinking algorithm by reducing the effects of noise. In fact the medial axis transform is really sensitive to noise, whose presence can introduce significant distortions. In a noisy point cloud, the iterations of the ball shrinking algorithm increase, as instability of points leads to smaller medial balls in the neighborhood of the point. Medial balls constructed on noisy points present an abnormal separation angle value, compared to the medial balls of the previous iterations. Thus, the denoise planar and preserve values correspond to two threshold values that bound the separation angle.

2.4.3 The structured MAT

In the unstructured MAT, the topology and structure are not explicit; however, they can be computed in a structuration process that leads to the creation of the structured MAT. In fact, the unstructured MAT can be segmented on the atom geometry using a region growing algorithm, initialized on random atoms. This process can lead to two outputs, the separation of the MAT into disjoint parts, called medial clusters, or the separation of the MAT into medial sheets. To obtain the segmentation into medial clusters, the cutting condition is that medial balls of atoms in the same medial cluster intersect, while those belonging to different ones do not, Figure 2.9. To obtain the segmentation into medial sheets, property is that nearby medial atoms in a medial sheet have similar medial bisectors, Figures 2.8 and 2.10. The medial bisector is a vector tangent to the medial sheet, which changes greatly on junction curves or for atoms on different medial sheets.

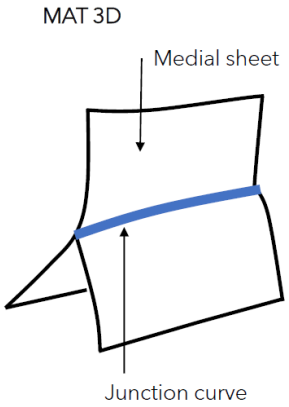


Figure 2.8: MAT - medial sheets

Last, the structured MAT is a point cloud representing each part or sheet, together with an adjacency or flip graph describing their connectivity. In the adjacency graph, each node

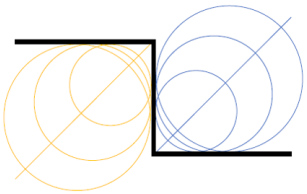


Figure 2.9: MAT cutting condition into medial cluster - medial balls of atoms in the same medial cluster intersect, while those belonging to different ones do not.

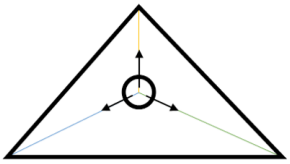


Figure 2.10: MAT cutting condition into medial sheets - nearby medial atoms in a medial sheet have similar medial bisectors.

correspond to a medial sheet, while edges describe the relations between them. Instead in the flip graph, nodes represent sheets, however edges relate interior and exterior medial sheets having common feature points.

3 Methodology

This chapter summarizes the main steps conducted in this research, with the aim to integrate the 3D medial axis transform in a deep learning algorithm. These steps are described on a theoretical and practical level, while the related experiments and results are listed in Chapter 5. First, an overview of the methodology is given in Section 3.1. Section 3.2 describes the preliminary studies necessary to select the deep learning algorithm and data-sets to work with. Furthermore it lists the main improvements performed on the data-sets and describes what are the main parameters for the 3D MAT computation. Section 3.3 outlines the experiments conducted with the 3D MAT as a feature in the PointNet++ algorithm [Qi et al., 2017]. Sections 3.4 and 3.5 detail the experiments conducted with the Superpoint graph algorithm [Landrieu and Simonovsky, 2017]. First, the 3D medial axis was used to partition a point cloud into simple geometric shapes; second it was used to enrich the nodes and edges of a graph for graph convolutions. Last, Section 3.6 describes the evaluation methods used to quantify the quality of the experiments' outputs.

3.1 Overview

The methodology proposed in this thesis focuses on the integration of the 3D medial axis transform in a deep learning algorithm. The methodology can be structured into five core phases, divided in preliminary steps, three MAT integration steps and evaluation steps, as shown in Figure 3.1.

Four core preliminary steps were needed.

- First, deep learning algorithms for semantic segmentation were analyzed and two were selected; these are PointNet++ [Qi et al., 2017] and Superpoint Graph [Landrieu and Simonovsky, 2017]. They were chosen as their structure should be compatible with the 3D MAT properties. The first is a point-based algorithm, while the second is a graph-based one.
- Second, the 3D MAT computation was tested on different data-sets, in order to select the open source ones to apply the methodology on. Open source data-sets are used to compare results with those obtained with other deep learning algorithms. These should be similar to CycloMedia's data-set and present a defined MAT structure to enable a clear analysis of the results.
- Third, all data-sets were pre-processed. Pre-processing steps include filtering, sub-sampling and normal vectors' computation.

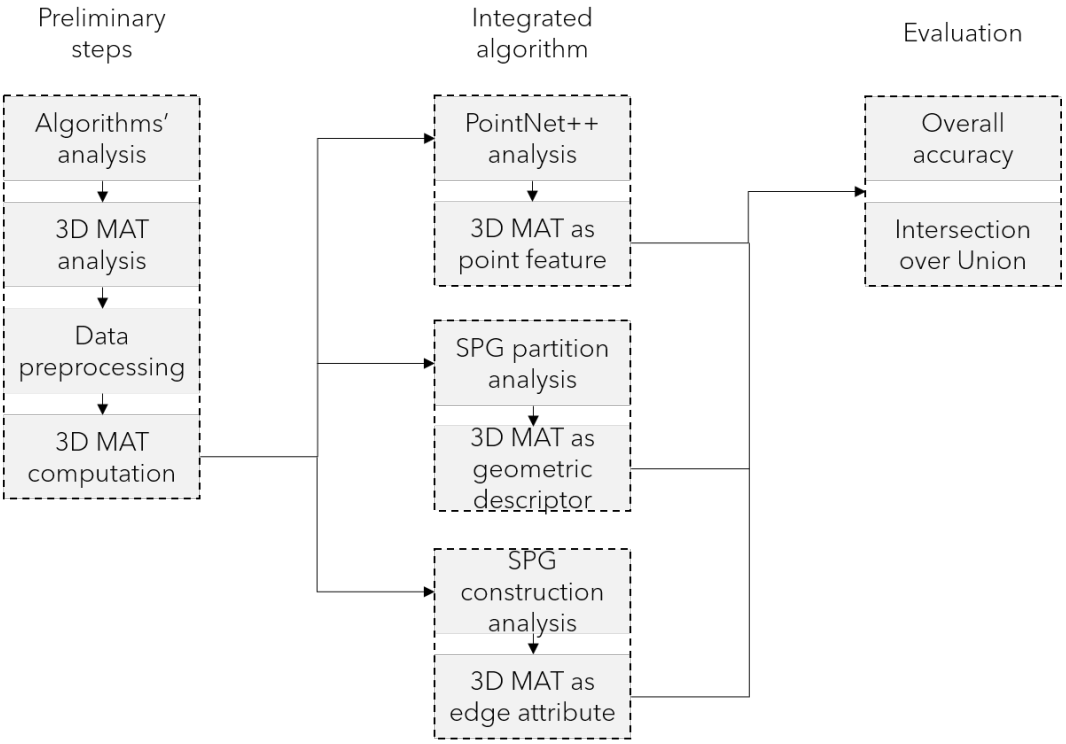


Figure 3.1: Overview of methodology

– Fourth, the unstructured and structured 3D MAT was computed for each point cloud, following the study of the parameters that influence its construction.

Then, three methodologies were followed to integrate the 3D MAT in the deep learning algorithms.

– First the 3D MAT was used as an extra point feature in PointNet++. Following the analysis of the algorithm's structure, its characteristics and hyperparameters, three core experiments were performed. In each experiment, MAT derived information was combined with the input point cloud to make the point classification stronger.

– Second, the 3D MAT was used as a geometric descriptor to partition a point cloud in Superpoint Graph. Here the point cloud is cut into homogeneous shapes, based on points' geometric information, prior to be input in the deep learning algorithm. MAT properties were used in the cutting algorithm with the aim to improve the partition and make it more similar to the structured MAT.

– Last, the MAT was used as an edge attribute in the SPG. Here two experiments were performed, using different MAT values which should enrich the graph's structure.

Finally, the quality of the experiments was quantified using two evaluation metrics, the overall accuracy OA and the intersection over union per class IoU. For each experiment, the

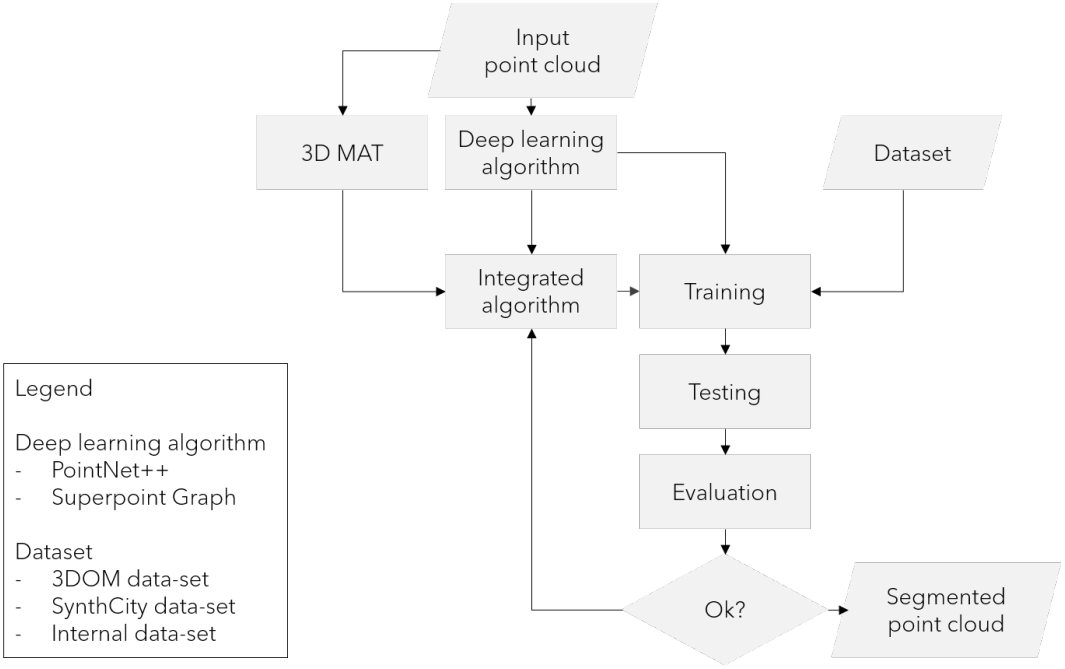


Figure 3.2: Flowchart

flowchart illustrated in Figure 3.2 was followed. This shows that the 3D MAT was integrated with a deep learning algorithm, PointNet++ or Superpoint Graph; then each data-set was trained, tested and evaluated to obtain a prediction. The prediction should quantify how valuable the MAT integration method is. In particular, for all experiments, the algorithm was first run with its default configuration. The result was then used as a base line and compared with the MAT experiments through the OA and IoU metrics.

3.2 Preliminary steps

3.2.1 Analysis of algorithms

Following the literature study, the assessment of existing methods and their characteristics was performed. Deep learning methods on point clouds follow different strategies to obtain per point class predictions. In this research, point wise MLP methods (Section 2.3.2) and graph based networks (Section 2.3.3) seem to be compatible. In fact, the first ones would incorporate information on the local geometry of the medial atom as a point feature; the second ones could exploit the organization of the 3D MAT in medial sheets. For these reasons, one algorithm for each category was selected. These should outperform other methods, obtaining state of the art results. Furthermore, they should be well structured and easy to integrate with additional code. Following these criteria, PointNet++ and Superpoint Graph

were selected. Both algorithms were first tested on the data-set they were implemented for. Then, supporting functions to read and convert the internal data-set were implemented.

3.2.2 3D MAT analysis for data-sets selection

Additionally in this phase, the visual analysis of the 3D medial axis transform was needed to identify the characteristics of CycloMedia's internal dataset and to select the public data-sets to use in this research.

First, using the software Geoflow[1], the 3D MAT was computed on different subsets of CycloMedia's internal data-set. A complete 360 degrees urban scene was first used; the same was cut in smaller point clouds and finally in single objects, see Figure 3.3, 3.4. The outputs show that the 3D medial axis transform is not clearly structured. This is due to the acquisition method of the point clouds that determines an incomplete representation of objects and a fast decreasing density of points. Given these results it doesn't seem feasible to exploit the graph structure of the 3D MAT, which wouldn't clearly define objects due to unwanted edge connections. However, each street object is characterized by a well defined cone, see Figure 3.4.

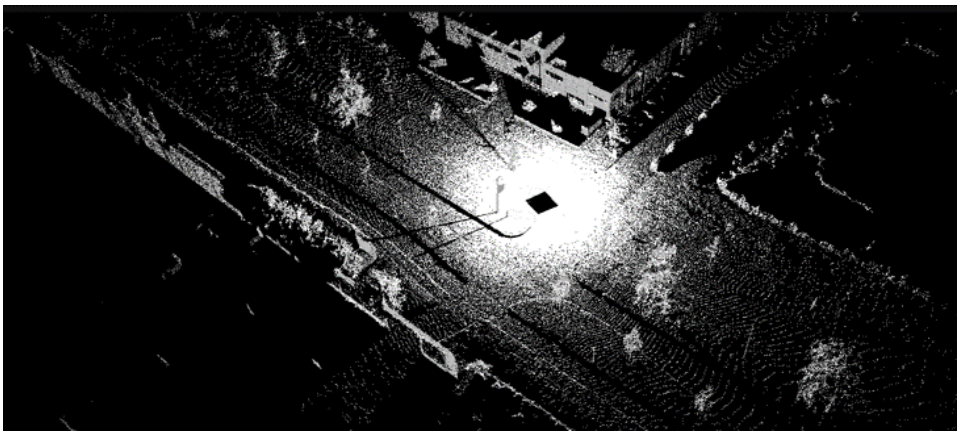


Figure 3.3: CycloMedia data-set, oriented point cloud

The same trials were conducted on different public benchmark data-sets. These should present some of the characteristics listed in Table 3.1. These are linked to two motivations, the similarity to CycloMedia's internal data-set and the possibility to clearly identify the MAT usability.

First, the experiments were conducted with the Paris Lille 3D data-set. [Roynard et al., 2017] In this data-set objects present holes in correspondence of glass, resulting in an incomplete 3D MAT. For this reason, the trials were performed again of a synthetic data-set [Griffiths and Boehm, 2019], Figures 3.5 to 3.7. In this case, the 3D medial axis transform is rather complete and could provide useful information for the scope of the research. In

[1] https://github.com/geoflow3d/geoflow

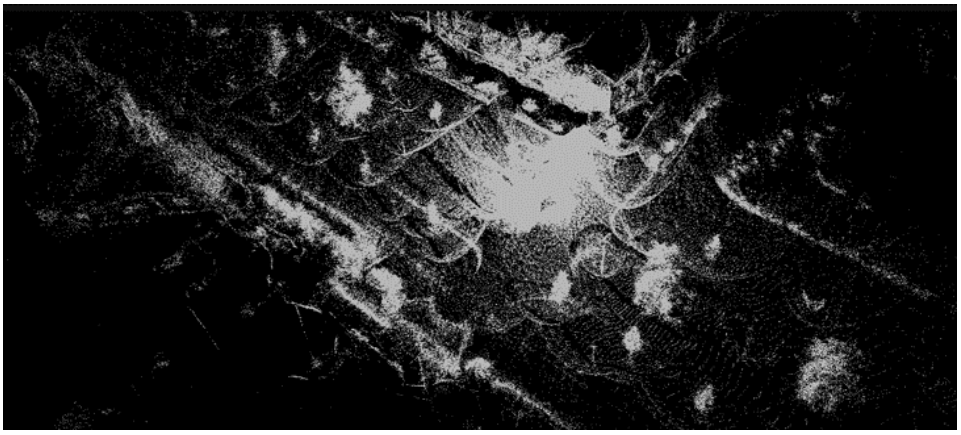


Figure 3.4: CycloMedia data-set, 3D MAT

Requirement	Motivation
MLS data-set	Similarity to CycloMedia's data-set
Low presence of noise	Analysis of MAT usability
Low presence of artifacts	Analysis of MAT usability
High points' density	Analysis of MAT usability
Homogeneous points' density	Analysis of MAT usability
Objects' geometry is fully represented	Analysis of MAT usability
Objects' materials are fully represented	Analysis of MAT usability

Table 3.1: Open source data-sets requirements and motivations

these images, the 3D medial axis transform is segmented and colored on an angle based threshold. Last, the experiments were conducted on the 3DOM data-set, which is a dense image matching point cloud of a 3D artifact that simulates a urban scenario [Özdemir et al., 2019]. The trials were meant to analyze the fitness for use of the public data-sets, that lead to the selection of the SynthCity and the 3DOM ones for this research. Table 3.2 shows the characteristics of each data-set.

	3DOM data-set	SynthCity data-set
MLS data-set		x
Low presence of noise	x	x
Low presence of artifacts		x
High points' density	x	x
Homogeneous points' density	x	x
Objects' geometry is fully represented	x	
Objects' materials are fully represented	x	x

Table 3.2: Open source data-sets characteristics

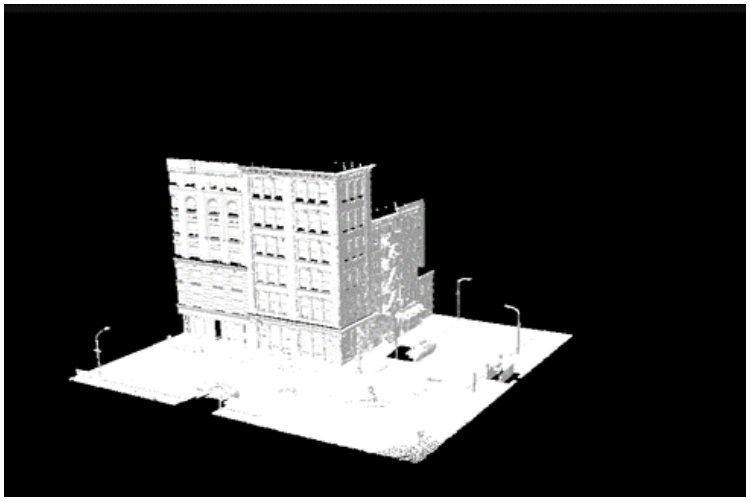


Figure 3.5: SynthCity data-set, oriented point cloud

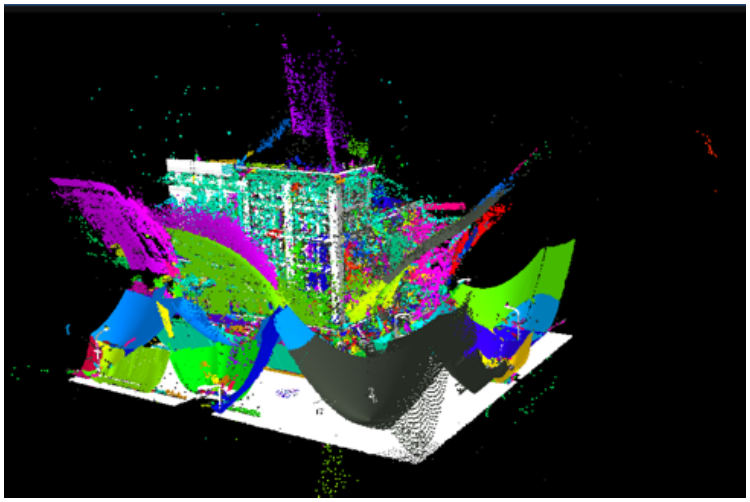


Figure 3.6: SynthCity data-set, 3D MAT

3.2.3 Data preprocessing

The second step of this phase of the research was data pre-processing and analysis. In this section the main procedures carried out are listed; for each data-set only a subset of these were performed, based on the data-set's peculiarities.

– Normal vector's computation and orientation is a crucial pre-processing step as it enables the accurate construction of the 3D MAT and the correct separation between interior and exterior sheets (Section 2.4.1). This step was carried out in CloudCompare, using the program's default parameters: plane local surface model, automatic

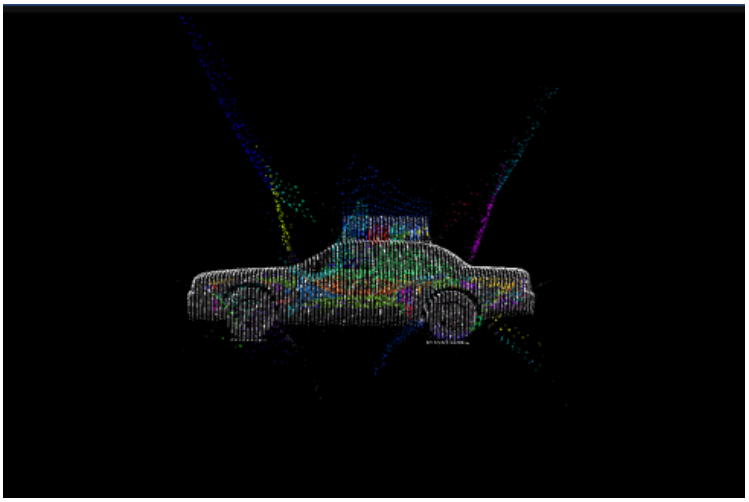


Figure 3.7: Car SynthCity data-set, oriented point cloud and 3D MAT

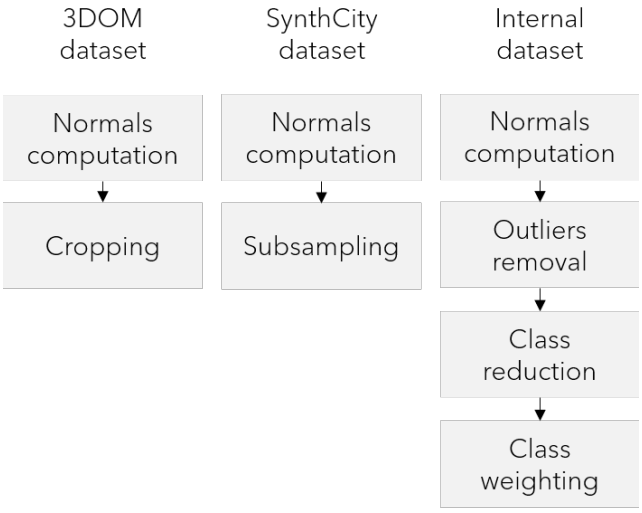


Figure 3.8: Data preprocessing pipeline

radius of the neighbors and orientation with minimum spanning tree (knn=6). Then, for each data-set, the computed normal vectors were oriented heuristically to obtain the best possible fit.

– Subsampling is the process of reducing the number of points in a point cloud. This step was carried out in CloudCompare with the purpose of reducing the weight of the data-sets while preserving fine structures. The spatial subsampling method was used; this involves selecting a the minimum distance between points in the output point cloud.

– Outliers removal is a way to clean the input data-set, by eliminating those elements of

the point cloud that do not respect a given condition. An example are points that lie at abnormal distances from the others.

– Cropping the input point cloud into smaller input files was performed to enable a better comparison between the training and test set and the validation set. This step is important as the files used in the deep learning algorithm and those used for evaluation should have similar characteristics, as number of points or represented objects.

– Classes' imbalance weighting consists of assigning a weight to each class of a data-set based on the number of times it is present in the data-set. This can be done through a bin count of the points in each class, followed by an inverse weighting method. This makes sure that classes that appear less are not penalized in the deep learning phase.

– Classes' reduction was needed for strongly imbalanced data-sets. This phase consisted in the identification of the classes most present in the data-set, together with the classes that could be merged and those that could be eliminated.

3.2.4 3D medial axis computation

In this research, the choice to compute the 3D MAT as a separate step and then using the output point cloud as input in the deep learning algorithm was made. This is due to the fact that many variables influence how the MAT can be computed and segmented. Thus, their study and their visual inspection through the software Geoflow[2] are important to obtain the most suited MAT to a specific data-set. Furthermore, this choice simplifies the later steps of the methodology, making the integration of the MAT more versatile in different workflows.

The 3D unstructured medial axis was computed for each data-set using first the software's default parameters, 3.3. Then different trials were performed in order to reach a satisfactory result. This should be a clean 3D medial axis transform, where adjacent structures are related through exterior medial sheets. The parameter mostly modified was the initial radius. As seen in Figure 3.10, a initial radius set to 200m would lead to an exterior sheet between the right building and the left tree; this is unwanted because these objects are far from each other in the point cloud, however their interaction in the MAT would influence their output features. Instead, the value 40m does not produce sheets between object distant than each other. The same custom parameters were later tested on the SynthCity and the internal data-set, resulting satisfactory for all. Thus, they were used in the course of this research.

	Default	Custom
Denoise planar	0.56	0.60
Denoise preserve	0.35	0.50
Initial radius	200	40

Table 3.3: 3DOM dataset unstructured MAT computation parameters

[2] https://github.com/geoflow3d/geoflow

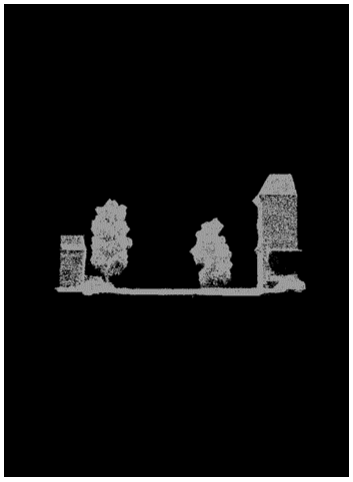


Figure 3.9: 3DOM point cloud

Figure 3.10: Unstructured MAT - default parameters

Figure 3.11: Unstructured MAT - custom parameters

Additionally, the structured MAT was computed in the software Geoflow[3]. The computation parameters of the region growth algorithms are shown in Table 3.4. In the trials, most parameters were kept equal or similar to the default ones. The parameter that most influences the output is the method one. In fact, it selects the cutting method for the structuration; in particular, "0" refers to the medial bisector, while "1" refers to the separation angle. Figures 3.12 and 3.13 show the former and the latter respectively. The use of the medial bisector as cutting condition for the structured MAT was chosen in this research (Section 2.4.3).

	Default	Custom
Ball overlap	1.5	5
Bisector angle	5	5
K	10	10
Method	0	0
Minimum count	10	25
Separation angle	5	10
Shape count	15	15

Table 3.4: 3DOM dataset structured MAT computation parameters

3.3 Medial axis transform as a feature in deep learning

This section describes the procedure followed to integrate the 3D medial axis transform in the PointNet++ algorithm, enriching the description of each point with information derived from the MAT.

[3]https://github.com/geoflow3d/geoflow

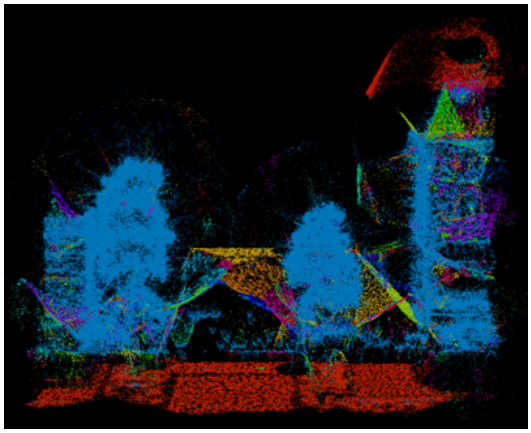


Figure 3.12: Structured MAT - medial bisector

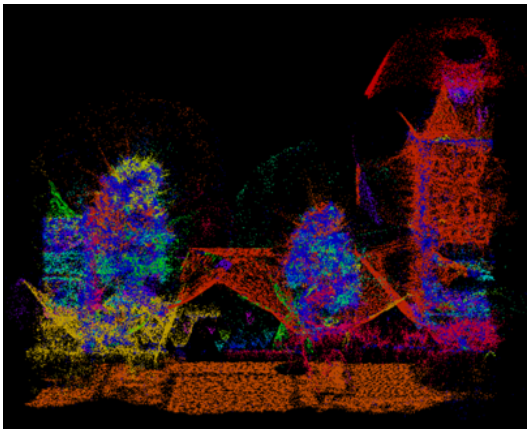


Figure 3.13: Structured MAT - separation angle

3.3.1 PointNet++ analysis

The first step needed in this phase was the in depth study of the PointNet++ algorithm structure. The purpose was understanding the main components of the algorithm and consequently identifying the optimal hyperparameters for training each data-set; moreover the goal was carrying out a training procedure for each data-set to be used as comparison for the later experiments.

PointNet++ structure

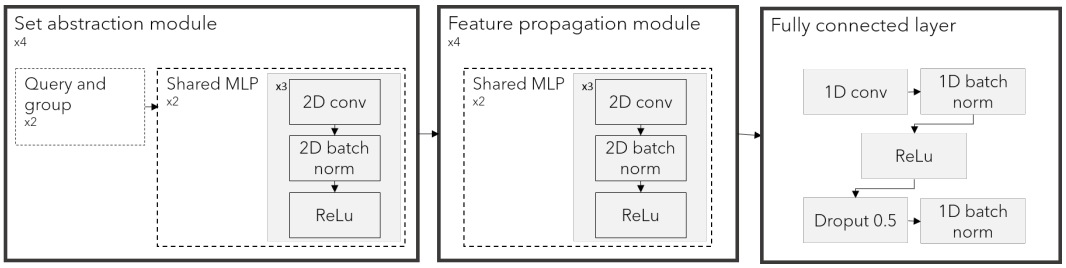


Figure 3.14: PointNet++ structure

As described in 2.3.2, PointNet++ is a point based deep learning architecture for point clouds semantic segmentation. Its structure is hierarchical and it is composed of set abstraction layers, with the aim to extract point information at different scales in the point cloud, Figure 3.14. Each set abstraction layer extracts multiple scales of local patterns and combines them according to local point densities [Qi et al., 2017]. Each set abstraction layer is made of a sampling layer, a grouping layer and a PointNet layer. In this research, the Pytorch

version of PointNet++, written by [Wijmans, 2018], was used. This is composed of four set abstraction modules that use a MRG method, each divided in:

Set abstraction layers

- *2 query and group modules*, the query modules select the centroids of the point cloud using the farthest point sampling (FPS); then, in the group modules the neighboring points of each centroid are selected.
- *2 shared MLP*, these modules correspond to the PointNet module described in the literature. They compute the embedding for each centroid at a different scale and concatenate them in a feature vector. Each MLP is composed by the below sequence of layers, repeated three times:
  - *2D convolution layer*
  - *2D batch norm layer*
  - *ReLu in place*

The set abstraction modules select a small subset of the full point cloud on which the PointNet module is used to learn local patterns. These are followed by four feature propagation modules and a fully connected layer; the former propagates the information learned in the set abstraction layers to all the points in the point cloud, using an inverse distance weighted method, the latter is used to derive the point wise predictions.

Feature propagation layers, the below sequence of layers is repeated two times:

- *2D convolution layer*
- *2D batch norm layer*
- *ReLu in place*

Fully connected layer:

- *1D convolution layer*
- *1D batch norm layer*
- *ReLu in place*
- *Droput layer*, during training, randomly zeroes some of the elements of the input tensor with probability, in this example 0.5, using samples from a Bernoulli distribution [Paszke et al., 2019].
- *1D convolution layer*

Last, the loss function used in this algorithm is the cross entropy loss, which is a measure of the difference between two probability distributions for a random variable or set of events. This function is particularly suited for imbalanced data-sets, as it is possible to input weights for each class.

PointNet++ hyperparameters

The Pytorch implementation of PointNet++, [Wijmans, 2018], uses nine hyperparameters that can be modified to tune the deep learning outcome, listed in Table 3.5. Among these parameters, two were modified in this research, the batch size and the number of points, while the others were kept constant. The first one regulates the number of batches of the point cloud that are processed at the same time, while the second determines the number of points in each batch. Thus, the first one determines on how many points in the point cloud the deep learning step is applied. In fact, the steps explained in Section 3.3.1 are repeated for each batch of the point cloud. The second defines the frequency with which the loss function is applied and consequently the network is optimized.

Hyperparameters	
Batch size	1 to 32
Number of points	9000 to 90000
Weight decay	0
Learning rate	0.001
Decay step	20
Batch norm momentum	0.9
Batch norm decay	0.5
Epochs	200

Table 3.5: PointNet++ hyperparameters

3.3.2 Using properties of the 3D MAT

As described in Section 3.2.4, the 3D MAT was computed as a pre-processing step and integrated with the PointNet++ algorithm as extra point features. Three properties of the MAT were used, the medial atom point coordinates, the radii and the separation angles related to each feature point (Section 2.4.1). These could contribute to the algorithm in different ways, in particular:

- When using the MAT coordinates as input features, the properties that link them to the surface points are not explicit. However, each surface point is related to two MAT coordinates, whose relative positions to the point strongly depict the geometry surrounding it. Thus, the assumption is that these properties can be derived in the deep learning algorithm through the subsequent abstraction of the input data-set.
- Radius and separation angle values introduce a numeric information which is derived by the MAT coordinates but agnostic to their location. This information defines clear patterns in the input and can be used to recognize different structures in the data-sets. The use of MAT properties in deep learning should add knowledge of the neighborhood of the point, depicting long range structures of the point cloud. In fact, the radii values puts in relation a surface point with another one not belonging to the immediate surroundings. Also, the separation angles represent the angle between these distant surface points, describing the curvature of the surface (Section 2.4.1).

The standard feature information used in PointNet++ to characterize points is xyz coordinates and RGB information of each point. To these, normal vectors information, intensity and many others can be added. In this research, it was chosen to carry out a training procedure with xyz and RGB information. This should be used as a comparison to the further experiments. The MAT information was added in three ways to each data-set; the interior and exterior MAT coordinates, the MAT interior coordinates and radius and separation angle were used as features, in addition to coordinates and RGB.

In the Pytorch implemetation of PointNet++ [Wijmans, 2018] , each data-set is modeled in a Python class. First the input files are read through supporting functions, then the desired extra feature information is combined with the point cloud, finally the data-set is cut into chunks of the dimension of the hyperparameter number of points. Each chunk represents a random subset of the whole point cloud and is processed separately in the deep learning steps described in Section 3.3.1. Thus, point features derived from the 3D MAT describe the relation among points that are not represented in the subset. This property is desirable as it adds an accurate information to the points, although the neighboring relations are lost.

Additionally, specific experiments were carried out on selected inputs. These experiments were used to get a deeper insight on the PointNet++ algorithm's functioning and on how the MAT properties contributed to the results.

- First, training procedures were carried out modifying the batch size and number of points parameters and adding weights to each class. This procedure was needed to test the potential of the algorithm for the selected data-sets, avoiding overfitting and defining the best possible initial configuration.
- Second, extra MAT features were added, for example, radii and separation angles were used separately to investigate the contribution of each to the final result; later only the radii and separation angles of the interior MAT were tested.
- Third, different normal vectors' computation methods were tested, in particular, Geoflow's default computation method was used.
- Last, a training procedure using radii and separation angles features but excluding the RGB information was carried out and Gaussian noise was added to the data-set to investigate its influence on the MAT.

3.4 Medial axis transform as a descriptor for a geometric partition

This phase of the research focused on the use of the 3D medial axis transform as a geometric descriptor for the partition of the point cloud. This analysis was carried out on the algorithm Superpoint Graph [Landrieu and Simonovsky, 2017], in which the partitioned point cloud is used as input in a deep learning algorithm for semantic segmentation (Section 2.3.3). In Section 3.4.1 the analysis of the current partition method used in Superpoint Graph is presented. In Section 3.4.2 the MAT properties that make it suitable for this purpose are outlined, together with the experiments conducted; in particular, the use of the 3D MAT

as a geometric feature in the cut-pursuit algorithm and its use as an edge attribute in the nearest neighbor graph.

3.4.1 Superpoint Graph geometric partition

Superpoint Graph is a graph based deep learning algorithm, which is composed of three main steps. First the point cloud is partitioned in homogeneous shapes and their relations are structured in an adjacency graph, the SPG. Second PointNet is used to learn their embeddings and last a deep learning algorithm based on graph convolutions is used to perform contextual segmentation (Section 2.3.3). In this paragraph, the description of the partition method is presented.

In the algorithm, the input is a point cloud for which the nearest neighbors graph is constructed. The nearest neighbor algorithm takes as input two knn values; the first is used to build the adjacency structure for the minimal partition, while the second refers to the number of neighbors used to compute the geometric features. The graph edges are weighted as the inverse of the distance between points, to penalize farther ones. Then the output neighbors are used as input in the compute_geof function, which computes four geometric features, linearity, planarity, scattering and verticality. To do so, the covariance matrix is built for each neighborhood together with its eigen values and eigen vectors. The four outputs should describe the geometric properties of the points and then be used to partition the point cloud. Thus, these are input in the cut-pursuit algorithm, together with the nearest neighbor graph information, point neighbors and edge weights, and a regularization parameter for the minimal partition, which decides the coarseness of the output. [Landrieu and Simonovsky, 2017] The cut-pursuit algorithm cuts the point cloud into constant connected components, the super points, minimizing the difference between the neighboring points and the geometric features summed to the regularization parameter times the edges weights.

$$\arg\min_{g\in\mathbb{R}^{dg}} \sum_{i\in C} \|g_i - f_i\|^2 + \mu \sum_{(i,j)\in E_{nn}} w_{i,j}[g_i - g_j \neq 0]$$

Thus, the main factors that can be modified to improve the partition are the number and type of geometric features, the weights on the edges of the nearest neighbors graph, together with two knn parameters, and the regularization parameter (μ). In this research, it was chosen to keep the knn parameters unchanged; also, after a number of trials on the influence of μ, this value was kept stable.

3.4.2 Using properties of the 3D MAT

The 3D MAT was computed as a pre-processing step, Section 3.2.4, and integrated in the algorithm in two main ways. First, the local geometry of the medial atoms, in particular radii, separation angles and medial bisectors, was added to the four default geometric features. Second, the 3D MAT was segmented into medial sheets; then, the segment information was used to strengthen the edge relations on the nearest neighbors graph used as input in the partition. The main intuition is that the local geometry of the medial axis describes the

structure of the point cloud in its proximity but also at a larger scale. Thus, the MAT properties could be useful in the cut-pursuit algorithm and consequently to the final predictions of SPG. In fact, improving the geometric partition into homogeneous shape should produce meaningful superpoints and superedges, thus better overall results.

The main experiments to improve the partition using features, were conducted using the radius, separation angle and medial bisector information. The latter is a unit vector parallel to the direction of the medial sheet, thus points on the same medial sheet should have a similar bisector. The radius should add knowledge on the distance between a point and a second surface point, while the separation angle on the curvature of the surface between them. Both the interior and exterior radii and separation angles were used in different combinations. In particular,

- interior and exterior radii and separation angles
- interior radii and separation angles
- exterior radii and separation angles
- interior and exterior radii
- interior and exterior separation angles

The MAT file was read through an helper function and the output was normalized, to be scaled as SPG's default geometric features. Then, it was combined with linearity, planarity, scattering and verticality and input in the cut-pursuit algorithm. In the original algorithm, verticality is given less importance by doubling the values of the other parameters. In this thesis, all values were kept equally important to clearly identify the influence of the MAT descriptors.

Additionally, the MAT was used to enrich the edges of the nearest neighbors graph, which is used to partition the point cloud. In the original algorithm, the edges weights' are computed as the inverse of the distance between a point and its neighbor. In this research, the edges between points belonging to the same medial sheet was strengthened. The structured MAT and the superpoints are similar on a theoretical basis; in fact, the medial sheets should represent a decomposition of the input into simple structures, similarly the superpoint is constructed as the partition of the input point cloud into simple and homogeneous parts. Thus, the assumption is that these structures are also similar in practice.

To investigate this hypothesis, the structured MAT was constructed as in Section 3.2.4, then the sheet identifiers were projected to the input point cloud. This procedure outputs two point clouds, one with the sheets corresponding to the interior MAT and one with those of the exterior. The interior MAT corresponds to the skeleton of the objects, while the exterior defines the relations between different ones. For this reason, the interior sheets were used. Then, the points belonging to the same superpoint were associated to their medial sheet value. For each superpoint the mode, or the most frequent occurrence, was calculated, in order to determine to which medial sheet were they related. Then for each superpoint and each medial sheet a point count was performed. This value and the mode value were used to find a similarity metrics between the superpoints and the structured MAT.

Finally, to strengthen the edge weights for neighbors belonging to the same sheet, three steps were needed. The construction of the structured MAT and association with the input

point cloud are described above. Then, in the nearest neighbor structure, the segment value corresponding to each neighbor was retrieved. If the point and the neighbor belonged to the same medial sheet, the relation would be labeled as one and stored in a new array. On the other hand, if they did not, the relation would be labeled as zero. Then, these values would be added to the distances in the edge weights count and input in the cut-pursuit algorithm.

To compare the different methods, first the algorithm was run with its default partition method and deep learning hyperparameters. Then, the different partitions were used as input in the algorithm, maintaining its hyperparameters. The goal was to clearly depict the influence of the partition in the final result.

3.5 Medial axis transform as a graph attribute for graph convolution

This phase investigates possible improvements related to Superpoint graph algorithm [Landrieu and Simonovsky, 2017]. Here, the properties derived from the 3D medial axis transform were used to compute new super point and super edge features and input to the deep learning algorithm. As in Section 3.4, the assumption is that the local geometry of the medial atom can be a powerful descriptor of shapes, thus it could improve the overall results of the deep learning algorithm. In Section 3.5.1 is explained how the super point graph (SPG) is constructed; in Section 3.5.2 is defined how the MAT was integrated to it.

3.5.1 Superpoint graph and deep learning

After the partition of the point cloud into homogeneous shapes, the Superpoint graph is built. In this adjacency graph, the nodes represent the homogeneous partitions, or superpoints, while the edges, or superedges, define the relationships among them. Each node and edge is enriched with handcrafted attributes. In particular, for the superpoints these are its centroid, length, surface, volume and point count; for the super edges they are the delta mean, standard deviation, norm and centroid and the ratio between the lengths, surfaces, volumes and point counts. Once the graph is constructed, the point cloud is reorganized in super points and input in the deep learning algorithm.

In the algorithm PointNet is used to learn embeddings on the superpoints. These are associated with the geometric features used to partition the point cloud, linearity, planarity, verticality and scattering, together with RGB information, elevation and distance to the center of the point cloud. Then, contextual segmentation is performed using graph convolutions. Here, superpoints refine their embeddings based on the features associated with the superedges. These features are derived from the superpoint and superedge attributes; by default they are the delta means and standard deviations, the logarithmic ratio between super points' surfaces, volumes, size and the ratio between centroids.

Furthermore, many deep learning hyperparameters can be modified to tune the learned output. The most relevant are divided in optimization arguments, model attributes, point cloud pre-processing, superpoint graph, PointNet parameters. In this research, the choice

to maintain the default hyperparameters was made, to simplify the analysis of the influence of the 3D MAT.

3.5.2 Using properties of the 3D MAT

The construction of the superpoint graph follows the partition of the point cloud into simple shapes. Thus, as in Section 3.4, the 3D MAT was computed as a pre-processing step; then the MAT was input into the compute_spg algorithm, to associate it to the superpoints. Here, the MAT was used in two main ways; first, the mean interior and exterior radii and separation angles were combined to the superpoints, while superedges were associated with the differences between them in two adjacent superpoints. Second, the same procedure was followed using the minimum and maximum radii and separation angles values. The main hypothesis is that a superpoint should be characterized by these features, which could thus enrich the superedges with additional knowledge.

To test these experiments, first a run of the algorithm using its default partition and hyperparameters was made. Then the different graph features were tried, keeping the same partition. The training procedure was carried out using the MAT properties in combination with the default ones and alone. The purpose was needed to understand how useful this information is and if it could improve the accuracy of the algorithm.

3.6 Evaluation

In this research, the results of the integration between the 3D medial axis transform and the deep learning algorithms, PointNet++ and Superpoint graph, can be quantified as the final amount of correct predictions that these algorithms produce. This amount can be computed for each input file through a confusion matrix, as described in Section 2.1.1. From the confusion matrix, two evaluation methods were mainly used to quantify the outputs of the three main experiments (Sections 3.3, 3.4, 3.5). These are the overall accuracy OA and the intersection over union IoU. The overall accuracy gives an overall clue of the performance of the algorithm for all classes. The intersection over union is a per class metric that enables the analysis of the results with a focus on each class, outlying the strengths and weaknesses of the algorithm.

4 Data-sets and tools

In this chapter, the data-sets and tools used for this research are presented. The first data-set is an internal set of point clouds from the company CycloMedia Technology, its characteristics and modifications applied during this project are listed in Section 4.1. The others are open source data-set; they are needed to compare the results outlined in Chapter 3 with the ones obtained with other deep learning algorithms. The use of these data-sets should facilitate the analysis of the outputs of the research, limiting the influence of factors that are present in real-data. The properties of the 3DOM data-set are outlined in Section 4.2, while those of the SynthCity data-set in Section 4.3. Last, Section 4.4 lists the main tools used and the main Python libraries.

	Acquisition method	Total # point clouds	# Points per point cloud	# Classes
CycloMedia's dataset	MLS	500	3M	82*
SHREC data-set	MLS	80	3M	6
3DOM data-set	DIM	1	28M	6
SynthCity data-set	MLS**	9	15M to 50M	9

*reduced to 17
**simulated

Table 4.1: Data-sets information

4.1 CycloMedia's internal data-set

CycloMedia's internal data-set consists of 500 mobile laser scanner point clouds with color information. Each point cloud represents an urban scene and is made of around 3 million points; Figure 4.1 shows one example. Each point cloud is obtained with mobile based Lidar from one car position. Then, a mesh of the point cloud is created and each point is associated with labels through the corresponding images.

The acquisition method determines the fast decreasing density of points, which is directly related to the distance of the objects from the car position. Figure 4.2 displays the number of neighbors of each point in a logarithmic scale, where the maximum is 7160 neighbors in the red area and the minimum in 1 in the blue areas. Figure 4.3 shows the resulting histogram. In order to reduce this problem, the point clouds were cleaned using the statistical outlier removal (SOR) filter [MediaWiki, 2015] in CloudCompare[1]. The filter computes first the average distance of each point to its neighbors, considering k nearest neighbors for each.

[1] https://www.danielgm.net/cc

Then it rejects the points that are farther than the average distance plus a number of times the standard deviation. In particular, k was set to two, while the standard deviation to three [MediaWiki, 2015]. Figures 4.4 and 4.5 show respectively an original point cloud and a filtered one. It can be seen that the filter keeps well defined structures, while eliminating those coarsely represented.

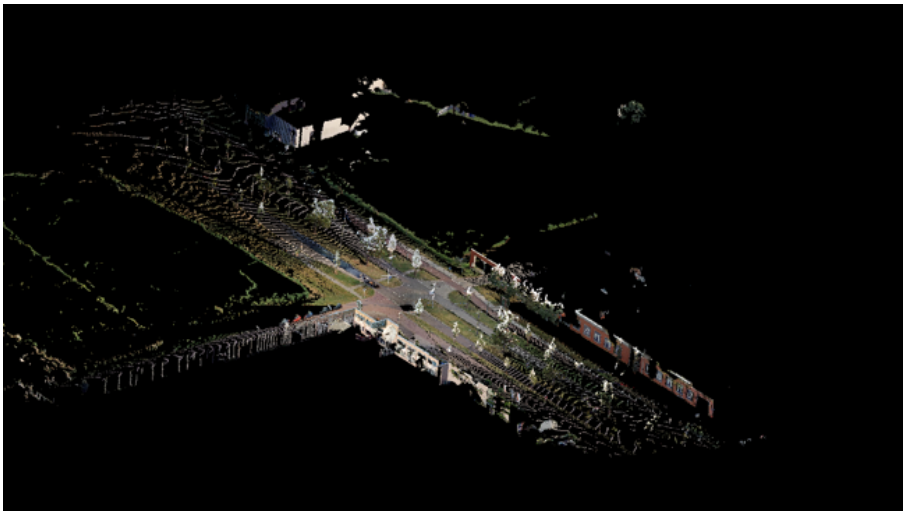


Figure 4.1: Internal data-set - rgb information

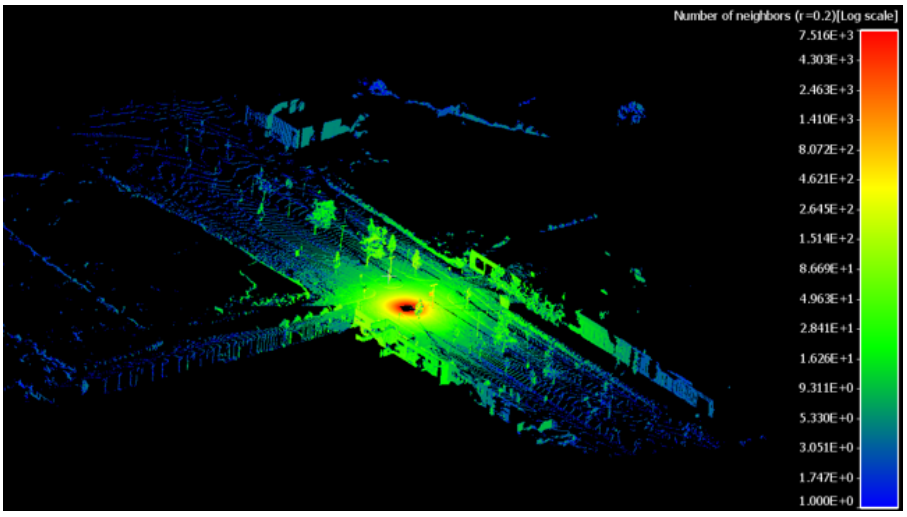


Figure 4.2: Internal data-set - density of points by number of neighbors, r = 0.2

This data-set is available in .laz format and it is segmented in 82 classes. The 82 classes represent objects from the urban environment, divided in barriers, flat constructions, structures, people, marking, nature, objects, traffic lights and vehicles. These classes are meant for image segmentation: furthermore, the data-set presents a strong classes' imbalance

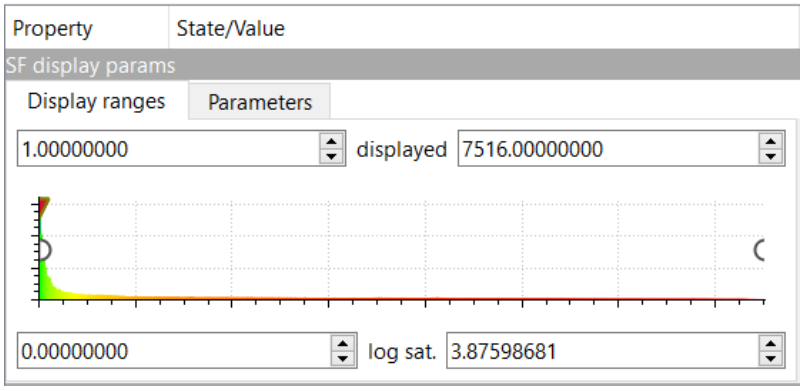


Figure 4.3: Internal data-set - density parameters

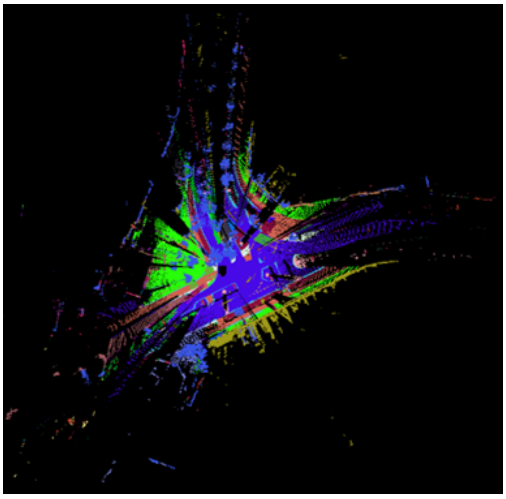


Figure 4.4: Internal data-set - point cloud

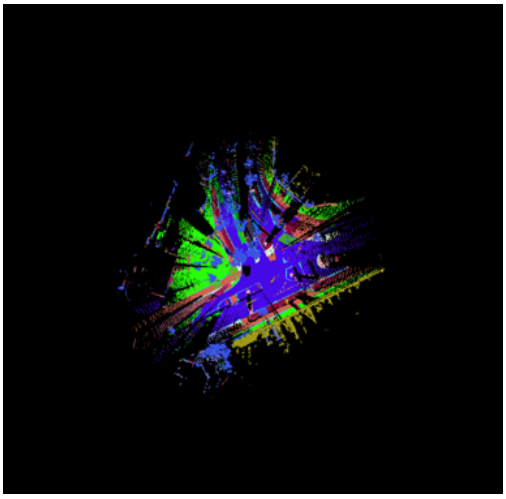


Figure 4.5: Internal data-set - SOR filter

as point clouds portray a high variety of scenes. For these reasons, the choice to reduce the number of classes was made; some were merged together and some removed from the data-set. Finally, 17 classes were obtained, see Table 4.2. As seen in Table 4.2, the class imbalance is reduced, however the most frequent one appears more than one hundred times more than the lowest one. Additionally, a weight was associated to each class in PointNet++. For each class, the weight is computed as the ratio between the sum of all elements and the number of occurrences of one class. The maximum bound for the weights is set to one hundred, in order to avoid giving an excessive importance to underrepresented classes.

Class #	Class name	Weight	Occurrences
1	Traffic light	100.0	212746
2	Billboard	29.8	1704139
3	Traffic sign	24.2	2103593
4	Support pole	16.0	3181278
5	Marking	15.5	3285654
6	Guard rail	25.8	1969373
7	Barrier	1.3	38545352
8	Building	0.3	200766977
9	Road	0.2	240810373
10	Bike lane	3.0	16671057
11	Curb	2.9	17414308
12	Low vegetation	2.0	25534303
13	Parking	1.9	27039516
14	Grass	1.4	37160530
15	Vehicle	1.3	40403381
16	Sidewalk	0.9	58704000
17	High vegetation	0.3	148432903

Table 4.2: Internal data-set - reduced classes, weights and occurrences

4.1.1 SHREC 2020

The SHREC 2020 data-set is a subset of CycloMedia's internal one, used in the 3D Object Retrieval 2020 competition[2]. This data-set is composed by 80 point clouds, split in training set, 60 point clouds, and test set, 20 point clouds. The data-set is manually labeled in 5 semantic classes, shown in Table 4.3.

Class #	Class name
0	Undefined
1	Building
2	Car
3	Ground
4	Pole
5	Vegetation

Table 4.3: SHREC data-set - classes

4.2 3DOM dataset

3DOM is a point cloud data-set obtained through dense image matching. These are acquired in a controlled environment over an ad-hoc 3D artifact that simulates a typical urban

[2]https://workshop.cgv.tugraz.at/3dor2020/

scenario [Özdemir et al., 2019]. The data-set is composed by a reference point cloud, which is unlabeled and presents around 28 million points, Figure 4.7. From this point cloud, the training and test set are cut, each is made of around 2 million points, Figures 4.8 and 4.9. This data-set is labeled into six semantic classes, shown in Table 4.4. Compared to the other data-sets, it presents a higher resolution of the objects. This data-set was chosen as objects are fully represented and it can be used to test the experiments with the MAT at its full potential. Figure 4.6 shows the density of the full data-set in the logarithmic scale; this is homogeneous through the whole point cloud. In this research, in addition to the training and test set, a validation set was cut from the full point cloud; this was then manually labeled in CloudCompare. The whole data-set, training, test and validation sets are shown in Figures 4.7 to 4.10.

Class #	Class name
0	Ground
1	Grass
2	Shrub
3	Tree
4	Facade
5	Roof

Table 4.4: 3DOM data-set - classes

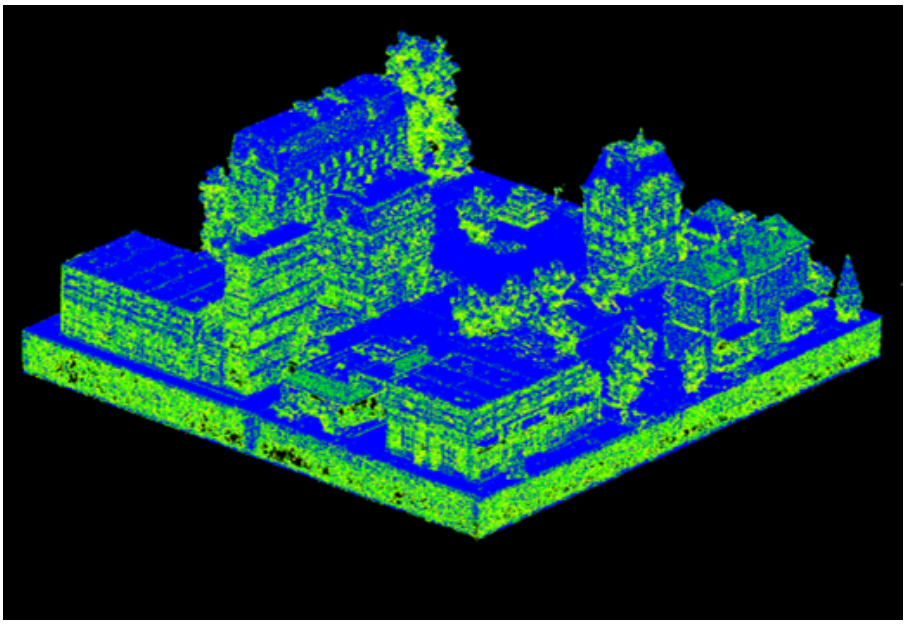


Figure 4.6: 3DOM data-set - density of points by number of neighbors, r = 0.2

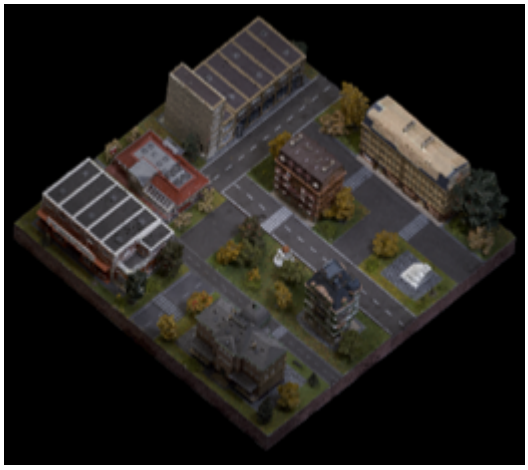


Figure 4.7: 3DOM data-set - full point cloud

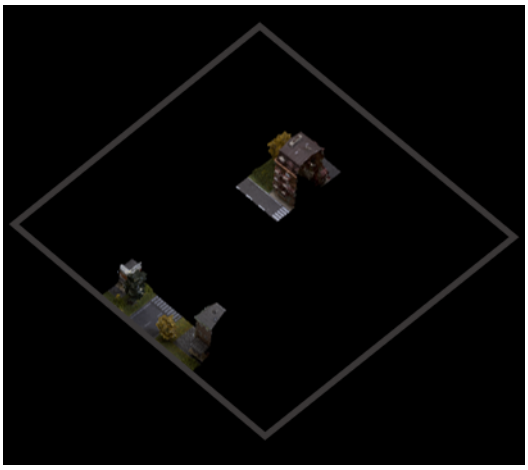


Figure 4.8: 3DOM data-set - training set

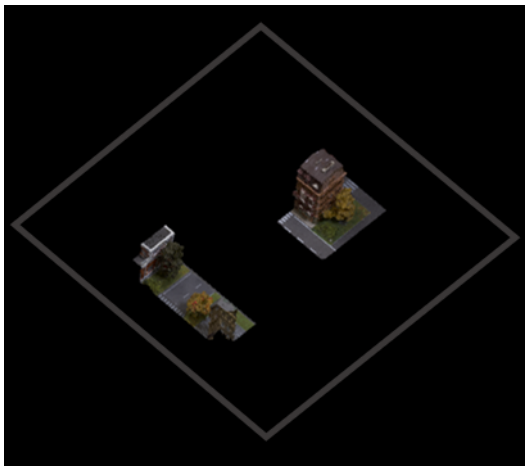


Figure 4.9: 3DOM data-set - test set

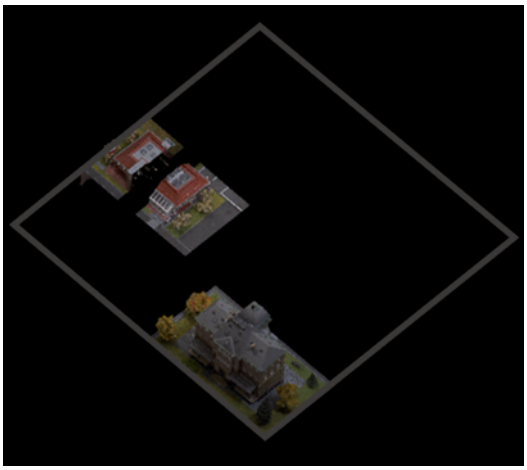


Figure 4.10: 3DOM data-set - validation set

4.3 SynthCity dataset

SynthCity is synthetic mobile laser scanner point cloud with color information, simulating a Velodyne scanner. It is composed of nine geographical areas, eight for training and one for testing; these are in .parquet format. The number of points ranges from 15 million to 52 million, for a total of 368 million points. Figure 4.11 shows a cropped area from the data-set. For the same area, the number of neighbors of each point was computed through CloudCompare, see Figure 4.12, the resulting histogram can be seen in Figure 4.13. Compared to CycloMedia's data-set, the maximum number of neighbors is smaller, given a search radius of 0.2. However, the density of points is more stable in the whole area, which is a desirable property. The point cloud is segmented in nine semantic classes: road, pavement, ground, natural ground, tree, building, pole-like, street furniture, car [Griffiths and Boehm, 2019]. This data-set is chosen because it is a mobile scanning data-set and it provides a complete representation of objects regardless of their material. This is a desirable property when working with the 3D medial axis transform, which consequently results more complete. In this research, the SynthCity data-set was subsampled in CloudCompare, setting the minimum distance between points to 0.01m. This process reduced the number of point by 80%, while preserving fine structures.

Class #	Class name
0	Building
1	Car
2	Natural ground
3	Ground
4	Pole like
5	Road
6	Street furniture
7	Tree
8	Pavement

Table 4.5: SynthCity data-set - classes

4.4 Tools

For this project, three categories of tools are needed: point clouds processing and visualization tools, medial axis transform tools and programming tools. In particular, CloudCompare[3], Lastools[4] and Mapple[5] are used to visualize, analyze, convert and export point clouds. Geoflow[6] is used to compute and analyze and extract the 3D medial axis transform for the chosen data-sets. Python is the programming language for the project used in the Pycharm development platform.

[3]https://www.danielgm.net/cc
[4]https://rapidlasso.com/lastools
[5]https://3d.bk.tudelft.nl/liangliang/software.html
[6]https://github.com/geoflow3d/geoflow

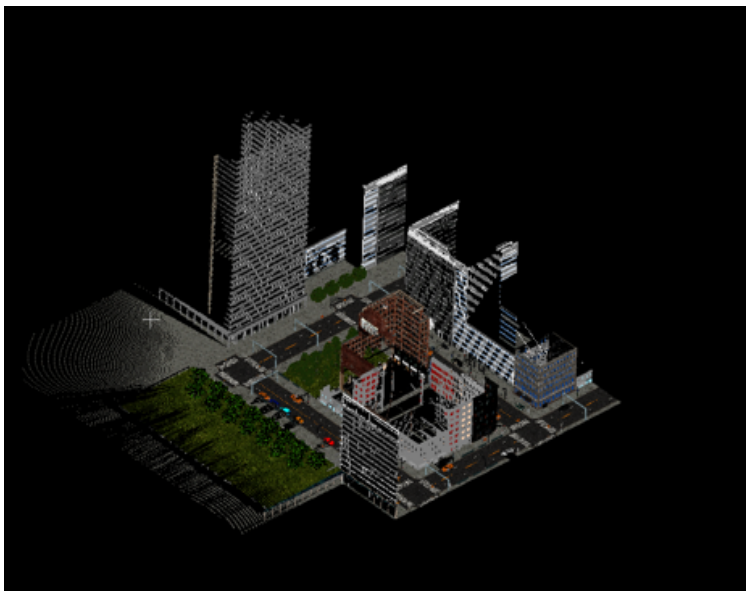


Figure 4.11: SynthCity data-set - rgb information

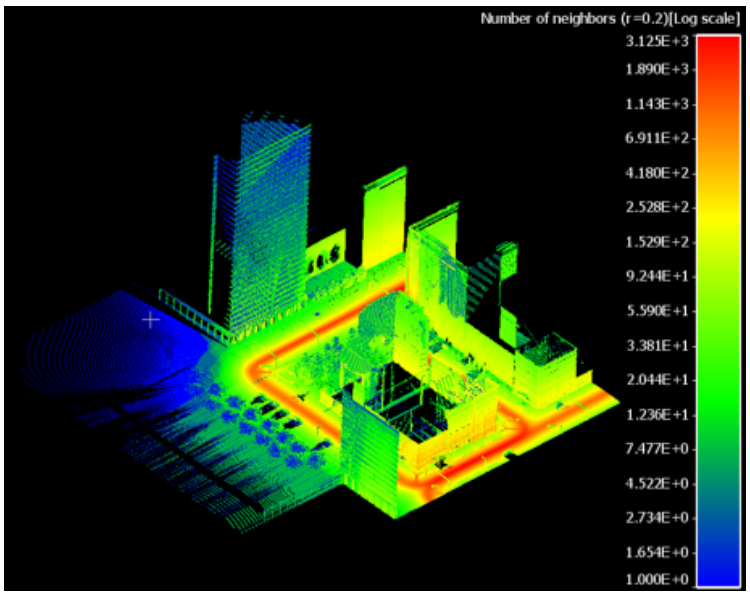


Figure 4.12: SynthCity data-set - density of points by number of neighbors, r = 0.2

The main libraries needed are: PyTorch[7], Laspy[8], Numpy[9], Scipy[10] and Parquet[11]. PyTorch

[7] https://pytorch.org
[8] https://pypi.org/project/laspy
[9] https://numpy.org
[10] https://www.scipy.org

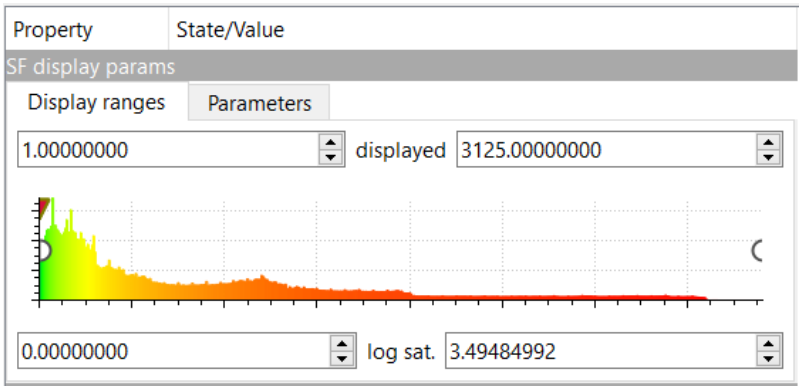


Figure 4.13: SynthCity data-set - density parameters

is used for deep learning tasks, Laspy to import and export las and laz files and Numpy is used to store and manipulate numbers' arrays. Scipy is needed for linear algebra tasks and statistics. Additionally format specific readers are needed, such as H5 reader to inspect the intermediate outputs of the deep learning process. Two open source algorithms are used, these are the PyTorch implementation of Pointnet++ [Wijmans, 2018] and Superpoint Graph [Landrieu and Simonovsky, 2017].

[11] https://pypi.org/project/parquet

5 Results and discussion

In this chapter, the main results obtained in the course of this research are presented. Section 5.1 outlines the outcomes of the experiments carried out with the PointNet++ algorithm; these are organized in three core groups which were performed for all the data-sets. The analysis of the other experiments performed is also presented. Sections 5.2 and 5.3 describe the experiments conducted with the Superpoint Graph algorithm. In the former, a visual analysis on the geometric descriptors is made together with the study of the derived partitions and their influence on the outcome. In the latter, the performances of the algorithm using different edge descriptors configurations are outlined.

5.1 Medial axis transform as a feature in deep learning

5.1.1 Core experiments

In this section, the main results for the three core experiments are presented. For each data-set, four graphs are shown; they depict the accuracy and loss values for the training set and the test set. The x axis represents the number of epochs. In the accuracy graphs, the accuracy value is between zero and one, where one represents the optimal outcome. This value is computed as the ratio between the number of correct predictions and the sum all of the predictions. Instead, in the loss graphs, the loss value of 0 represents the best result. The cross entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label [Fortuner, 2019].

Additionally for each data-set, the two evaluation metrics are presented for all of the experiments. The overall accuracy OA is computed as the above mentioned accuracy for one input file in the data-set. The intersection over union IoU is a per class metric that is calculated as the ratio between the correct predictions and the sum between the false positives and false negatives minus the correct predictions. Results show similar trends for all data-sets; in general, the use of MAT coordinates as features worsened the overall and per class performances, while the radii and separation angles improved the accuracy and decreased the loss.

When inputting interior and exterior, or only interior, coordinates the relation between them and the surface points is not explicit. The assumption is that it could be derived in the subsequent abstractions of the input data-set. However, not explicit relations and patterns between coordinates can not be detected by the deep learning algorithm and introduce artifacts and uncertainty. In fact, although the information derived by the coordinates is unique, the values used in this form are all different than each other and thus not usable.

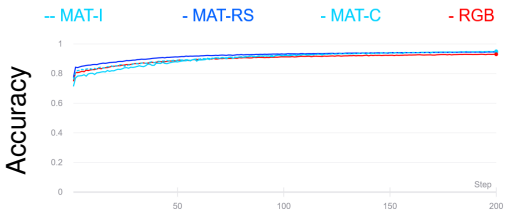
Instead, the radii and separation angles values give information on the point's surrounding geometry in an explicit way and are able to disclosure additional knowledge in the deep learning algorithm.

Name	Features
RGB	RGB
MAT-RS	RGB, interior and exterior radii and separation angles
MAT-C	RGB, interior and exterior MAT coordinates
MAT-I	RGB, interior MAT coordinates

Table 5.1: Core experiments legend

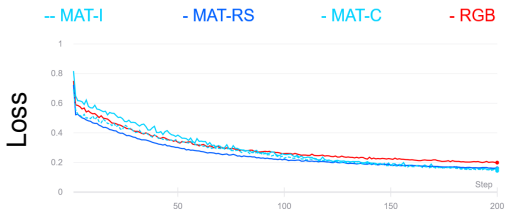
5.1.2 3DOM data-set core experiments

The core experiments conducted on the 3DOM data-set show how the radii and separation angle features can improve the performance of the training process, while the MAT coordinates introduce ambiguity. Figures 5.1 to 5.4 depict the interior and exterior radius and separation angle curve in blue, named MAT-RS, where these values were added to the RGB information, see Table 5.1. The default curve is represented in red, named RGB, the MAT coordinates and the MAT interior coordinates curves in light blue, the first solid while the second dashed. The MAT-RS curve presents the higher training accuracy and lower loss. The RGB curve is quite similar to the first one, bigger differences can be seen in the test accuracy and loss outputs. This proves that the radii and separation angles information lead to a better generalization of the learned patterns. Last, the MAT-C and MAT-I curves present similar trends to the above ones in the training data-set, while performing much worse in the test one, in particular on the loss values. This indicates how the coordinates information can easily overfit the training data.



Epochs

Figure 5.1: 3DOM - train accuracy



Epochs

Figure 5.2: 3DOM - train loss

Table 5.2 gives a deeper insight on the performances of each experiment per class. It can be seen that all classes intersection over union values increase using radii and separation angles features and that the grass and shrub classes present the biggest improvement. Figures 5.6 to 5.10 visualize the values of each feature on the point cloud, giving a clue on which information could be derived from them. In the RGB spectrum, grass and shrubs are quite similar; furthermore they do not present a great difference in height, differently

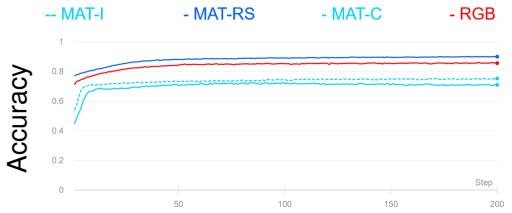


Figure 5.3: 3DOM - test accuracy

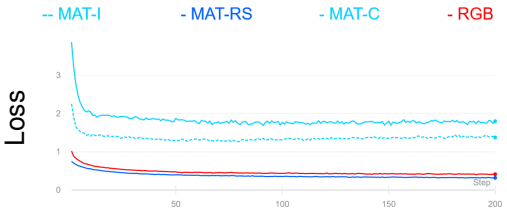


Figure 5.4: 3DOM - test loss

than trees. Instead, radii and separation angles clearly define these structures, helping their correct identification. This happens particularly for this data-set, which presents a high point density.

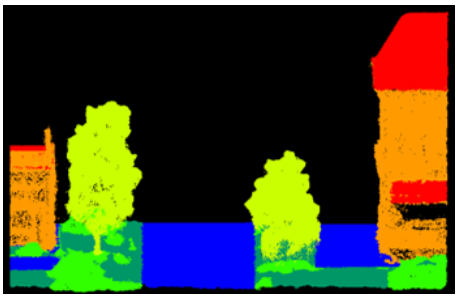


Figure 5.5: 3DOM - ground truth

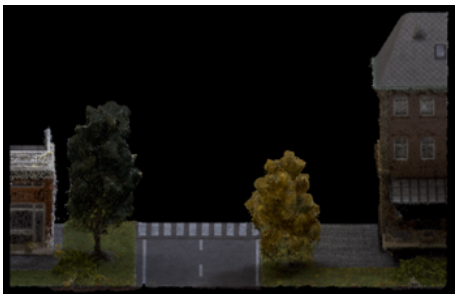


Figure 5.6: 3DOM - RGB

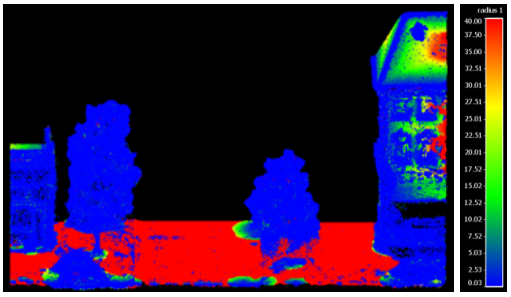


Figure 5.7: 3DOM - interior MAT radius

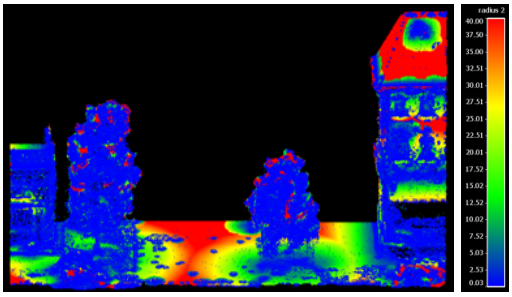


Figure 5.8: 3DOM - exterior MAT radius

In Table 5.2 can be also observed that all classes' accuracy strongly decreases when using the MAT coordinates, except for the roof class. This may happen because the former structures interact in the exterior MAT, sharing points and consequently properties. Instead, the roof structures present MAT exterior coordinates that are projected above the point cloud, thus not interacting with the other classes. An additional evidence of the fact that the interaction in the exterior MAT coordinates introduces ambiguity, is in the fact that ground intersection over union strongly improves when using only interior MAT coordinates as features.

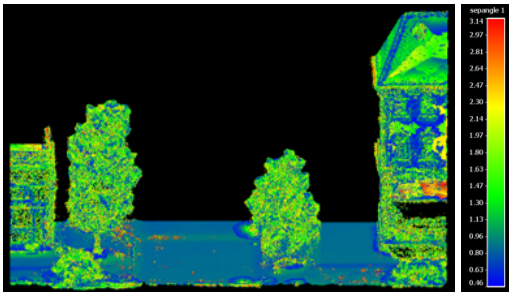


Figure 5.9: 3DOM - interior MAT separation angle

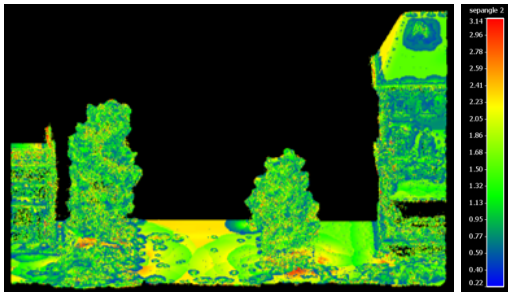


Figure 5.10: 3DOM - exterior MAT separation angle

The interior MAT for this class is a set of medial atoms shifted below the ground surface by the maximum radius of the medial ball. This is because the ground is a planar surface, thus the medial ball hardly shrinks. Keeping only this information and removing the exterior MAT is similar to not having this feature, in fact the result is comparable to the one with RGB only. A similar observation can be made for the grass class. The reason why its accuracy is decreased may be that this class is prone to be mistaken for the ground one, which presents a similar structure but is more represented in the data-set.

	RGB	MAT-C	MAT-I	MAT-RS	
OA	0.86	0.69	0.72	0.91	
IoU					
Ground	74.48%	59.12%	75.80%	83.98%	+9.50
Grass	34.49%	15.40%	14.39%	67.84%	+33.35
Shrub	42.78%	22.50%	22.47%	66.52%	+23.74
Tree	86.38%	50.27%	50.46%	91.34%	+4.96
Facade	88.48%	60.43%	61.91%	89.18%	+0.70
Roof	59.94%	57.32%	63.75%	68.59%	+8.65

Table 5.2: 3DOM dataset core experiments - overall accuracy and per class intersection over union

5.1.3 SynthCity data-set core experiments

Similar to the results for the 3DOM data-set, the SynthCity accuracy and loss curves for the training and test data-sets show that the radii and separation angles information improve the overall results, Figures 5.11 to 5.14. In this experiment, it can be seen more clearly that the MAT coordinates easily lead to overfitting in the training set, Figures 5.13, 5.14. Furthermore it can be observed that the use of MAT interior coordinates introduce ambiguity, but they lead to a better output than the exterior and interior ones. The reasons can be that adding more features, 6 rather than 3, could make it harder to extract information that generalize to the test set, especially if these do not express explicit relations between points. Furthermore

the exterior MAT describes the interaction between objects in a geographical point cloud, which could lead to more uncertainties as seen for the 3DOM data-set. Furthermore, it can be observed how the MAT coordinates fail to characterize fine structures such as poles and street furniture, Table 5.4.

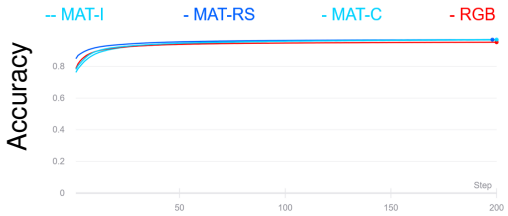


Figure 5.11: SynthCity - train accuracy

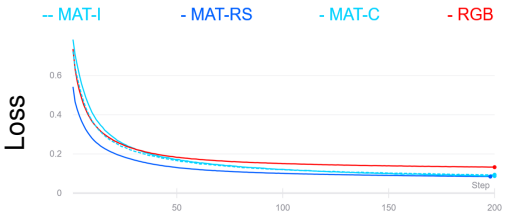


Figure 5.12: SynthCity - train loss

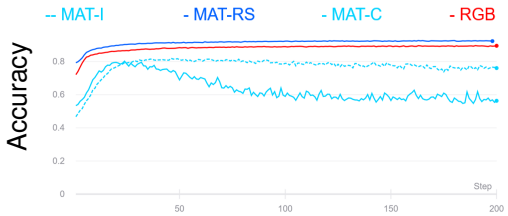


Figure 5.13: SynthCity - test accuracy

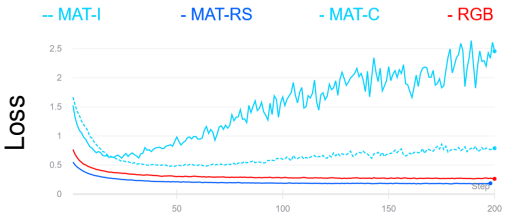


Figure 5.14: SynthCity - test loss

Figures 5.17 to 5.20 show a coarser resolution of the feature values if compared to Figures 5.61 and 5.10. This is due to the lower density of the SynthCity data-set with regards to the first one and to the fact that fully synthetic point clouds present almost no noise.

For this data-set, the classes intersection over union that present a higher improvement are the ground and natural ground ones, Table 5.4. This may happen because of the interaction between the natural ground and the trees, which also present a high improvement. In fact, it can be seen that the natural ground is always below the trees, thus radii and separation angles values might be influenced by them. In particular, Figures 5.19 and 5.20 show that the natural ground presents different separation angle values with respect to the ground. In fact, the separation angle values are constant (green in both figures) and perturbed only by the street furniture objects on them, such as pole structures. Instead, the natural ground exterior separation values vary greatly around trees which cover this class.

5.1.4 Internal data-set core experiments

The core results for the internal data-set follow the same trend as those for the 3DOM and SynthCity ones. Thus, even with real data, radius and separation angle introduce improve-

	RGB	MAT-C	MAT-I	MAT-RS	
OA	0.94	0.86	0.88	0.96	
IoU					
Building	97.90%	90.64%	92.04%	98.89%	+0.99
Car	71.58%	14.08%	24.27%	78.71%	+7.31
Natural ground	84.92%	50.53%	76.10%	93.16%	+8.24
Ground	45.49%	8.48%	15.13%	56.82%	+11.33
Pole-like	65.72%	0.00%	9.37%	66.84%	+1.12
Road	96.41%	83.46%	88.31%	97.99%	+1.58
Street furniture	34.50%	0.00%	0.31%	41.03%	+6.53
Tree	88.18%	69.98%	74.22%	95.58%	+7.40
Pavement	72.04%	65.03%	62.34%	78.83%	+6.79

Table 5.3: SynthCity dataset core experimets - overall accuracy and per class intersection over union

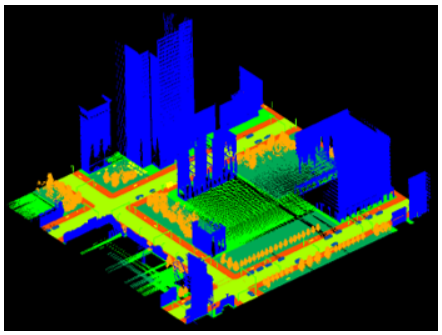


Figure 5.15: SynthCity - ground truth

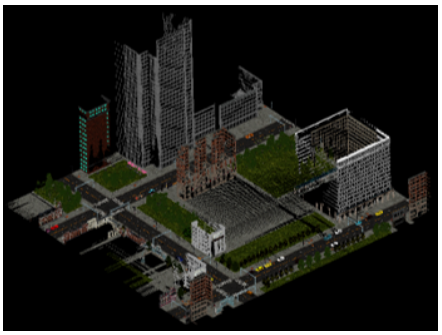


Figure 5.16: SynthCity - RGB

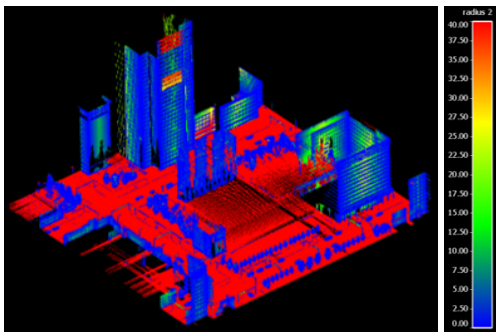


Figure 5.17: SynthCity - interior MAT radius

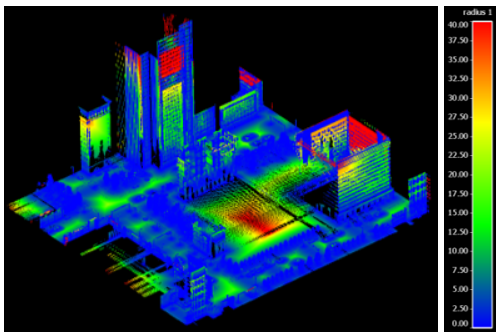


Figure 5.18: SynthCity - exterior MAT radius

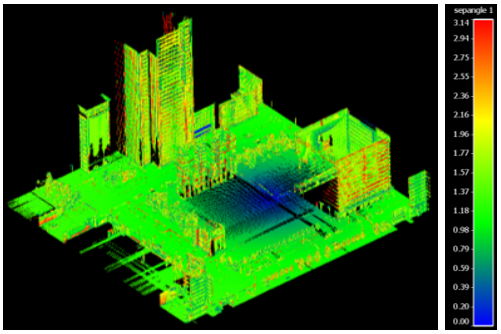


Figure 5.19: SynthCity - interior MAT separation angle

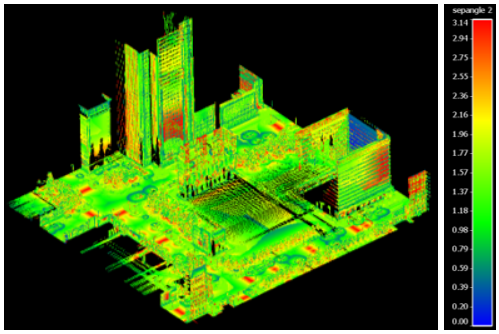
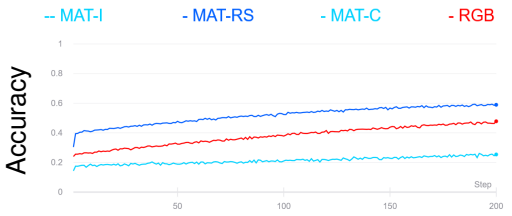


Figure 5.20: SynthCity - exterior MAT separation angle

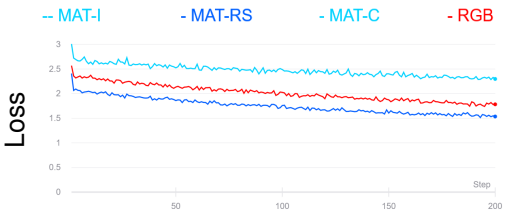
ments in the accuracy of the algorithm, while the use of coordinates does not. In fact, Figures 5.21 to 5.24 show that the MAT-C curve for the test set is not smooth, differently than the RGB and MAT-RS ones. This result is important, as the internal data-set of the company represents a multitude of locations and street objects. Consequently it presents a high number of classes and strong class imbalance. Furthermore, it is a real world data-set, thus re-projection errors, occlusions and varying density are present. For example, a car's shadow in an image can be classified as car instead of road.

Table 5.4 shows results for one validation point cloud. These are influenced by different factors, for example classes that present a 0% intersection over union value might not be present in the specific file. Furthermore, class imbalance might lead to these outputs; in fact, it can be noted that the road and high vegetation classes are the most represented ones and show a greater improvement in IoU.



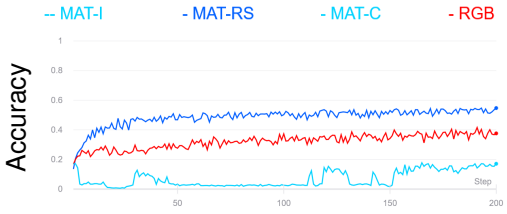
Epochs

Figure 5.21: Internal data-set - train acc.



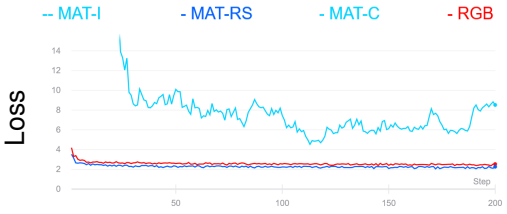
Epochs

Figure 5.22: Internal data-set - train loss



Epochs

Figure 5.23: Internal data-set - test acc.



Epochs

Figure 5.24: Internal data-set - test loss

	RGB	MAT-C	MAT-RS	
OA	0.24	0.01	0.49	
IoU				
Traffic light	0.00%	0.00%	0.00%	
Billboard	0.77%	0.00%	1.25%	+0.48
Traffic sign	4.17%	0.00%	4.45%	+0.28
Support pole	6.26%	1.42%	7.61%	+1.35
Marking	14.34%	0.00%	13.75%	-0.59
Guard rail	0.00%	0.00%	0.00%	
Barrier	2.72%	0.00%	3.50%	+0.78
Building	0.00%	0.00%	0.01%	+0.01
Road	22.21%	0.00%	56.29%	+34.08
Bike lane	0.00%	0.00%	0.00%	
Curb	3.99%	0.00%	8.91%	+4.92
Low vegetation	1.61%	0.00%	3.25%	+1.64
Parking	0.00%	0.00%	0.00%	
Grass	0.00%	0.00%	0.00%	
Vehicle	5.75%	0.00%	8.28%	+2.53
Sidewalk	0.00%	0.00%	0.00%	
High vegetation	49.03%	0.00%	61.67%	+12.64

Table 5.4: CycloMedia internal dataset core experimets - overall accuracy and per class intersection over union

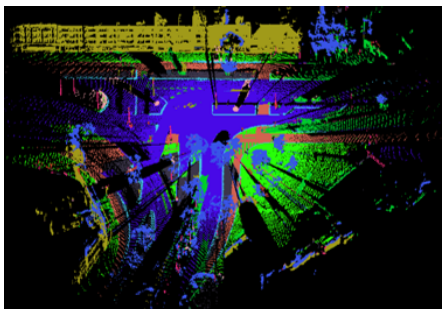


Figure 5.25: Internal data-set - ground truth

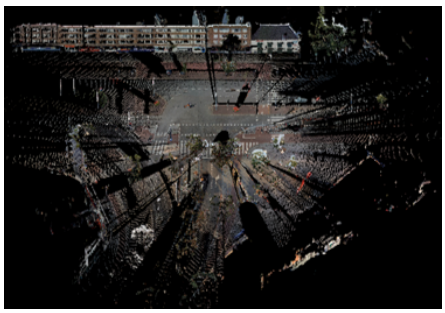


Figure 5.26: Internal data-set - RGB

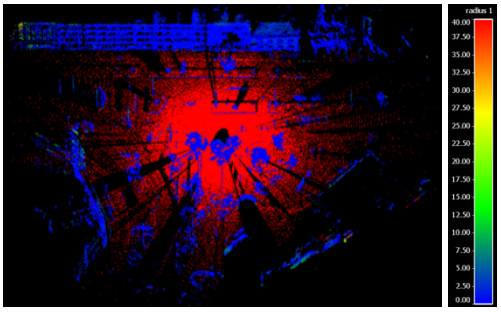


Figure 5.27: Internal data-set - interior MAT radius

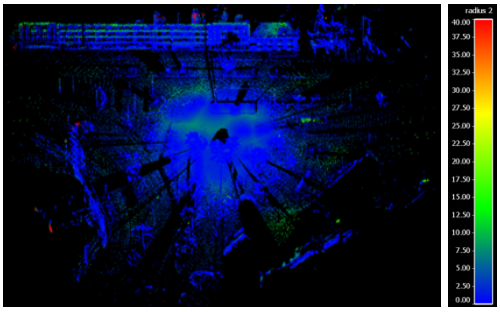


Figure 5.28: Internal data-set - exterior MAT radius

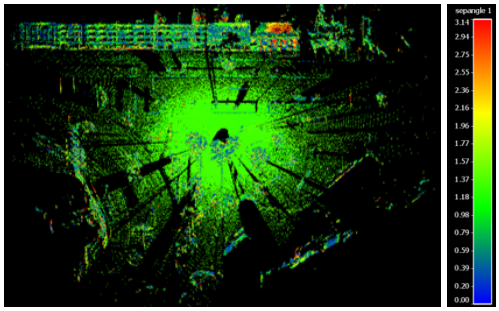


Figure 5.29: Internal data-set - interior MAT separation angle

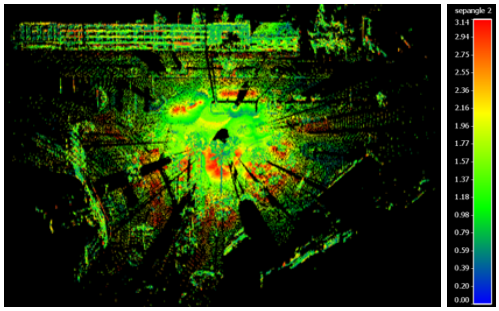


Figure 5.30: Internal data-set - exterior MAT separation angle

5.1.5 Other experiments

This section aims to summarize a selection of additional experiments that presented the most significant results. These were used to obtain a deeper knowledge on the influence of the MAT properties on the results of Section 5.1.1 and gain more insights on other factors that may be of interest.

3DOM - only radius and only separation angle

Name	Features
MAT-RS	RGB, interior and exterior radii and separation angles
MAT-R	RGB, interior and exterior radii
MAT-S	RGB, interior and exterior separation angles

Table 5.5: Only radius and only separation angle experiments - legend

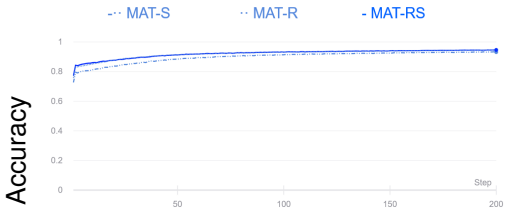


Figure 5.31: 3DOM radius and separation angle - train accuracy

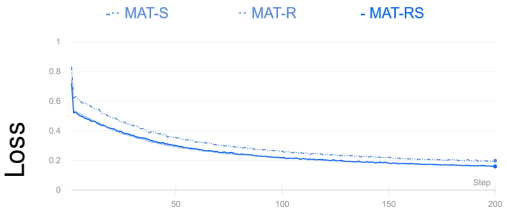


Figure 5.32: 3DOM radius and separation angle - train loss

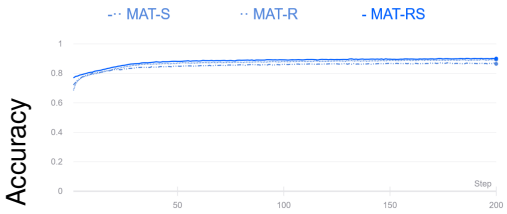


Figure 5.33: 3DOM radius and separation angle - test accuracy

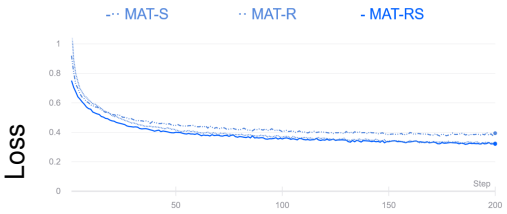


Figure 5.34: 3DOM radius and separation angle - test loss

As shown in Section 5.1.1 the radii and separation angles add useful per point information, increasing overall accuracy and per class results. Here, a further investigation was carried out to understand whether one of these factors contributed more to the result, and if so which one. In Figures 5.31 to 5.34 can be seen that the performance of the radius and separation

	RGB	MAT-R	MAT-S	MAT-RS
OA	0.86	0.90	0.88	0.91
IoU				
Ground	74.48%	82.76%	79.11%	83.98%
Grass	34.49%	64.09%	42.53%	67.84%
Shrub	42.78%	58.88%	53.20%	66.52%
Tree	86.38%	90.31%	86.92%	91.34%
Facade	88.48%	89.19%	87.72%	89.18%
Roof	59.94%	65.81%	66.79%	68.59%

Table 5.6: 3DOM dataset radius and separation angle experiments - overall accuracy and per class intersection over union

angle together (blue filled line - MAT-RS) is almost equal to that of the radius alone (dotted line - MAT-R). Instead, the separation angle alone performs slightly worse. Thus, it can be concluded that the radius information is more valuable than the separation angle one. However, the separation angle alone can still improve the outcome of the training procedure with respect to the use of xyz and RGB only, see Table 5.6.

3DOM - interior radius and separation angle

Name	Features
MAT-RS	RGB, interior and exterior radii and separation angles
MAT-RS-I	RGB, interior radii and separation angles

Table 5.7: Interior radius and separation angle experiments - legend

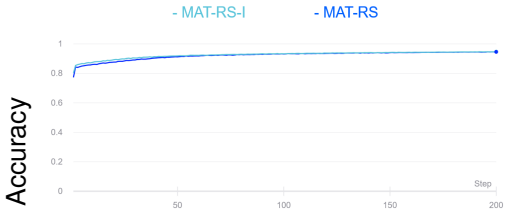


Figure 5.35: 3DOM interior radius and separation angle - train accuracy

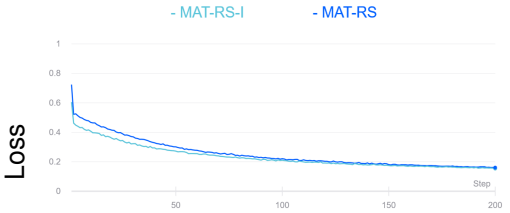


Figure 5.36: 3DOM interior radius and separation angle - train loss

Another experiment conducted on the use of radii and separation angles information is represented in Figures 5.35 to 5.38. Here the attention was drawn to the difference in results when using interior and exterior MAT information or only interior ones. It can be noticed here that the graphs' curves show almost identical results. This shows that the interior radii and

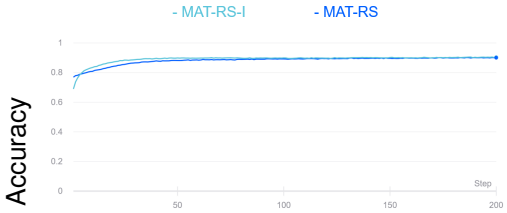


Figure 5.37: 3DOM interior radius and separation angle - test accuracy

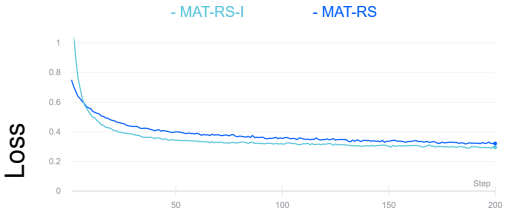


Figure 5.38: 3DOM interior radius and separation angle - test loss

separation angle would be enough to improve the outputs of the deep learning procedure.

3DOM - no RGB information

Name	Features
RGB	RGB
MAT-RS-RGB	RGB, interior and exterior radii and separation angles
MAT-RS	Interior and exterior radii and separation angles

Table 5.8: No RGB experiments - legend

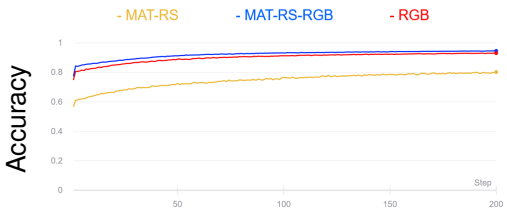


Figure 5.39: 3DOM no RGB - train acc.

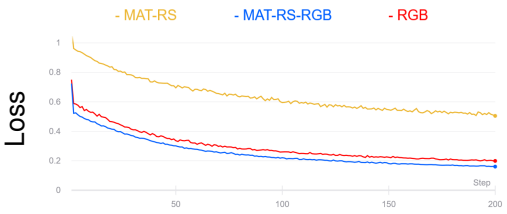


Figure 5.40: 3DOM no RGB - train loss

Figures 5.39 to 5.42 show a comparison between the RGB curves using xyz and RGB as features, the MAT-RS-RGB curves in blue (xyz, RGB and interior and exterior radii and separation angles as features) and the MAT-RS curves in yellow (xyz and interior and exterior radii and separation angle as features). This experiment was useful to understand to which extent the MAT information could be exploited. In fact, it can be seen that if they are used together with the RGB information, results improve; however, when RGB is not used, outcomes present a significant difference. Thus, it can be concluded that these features can not substitute the RGB values, but they can be successfully used in addition to it.

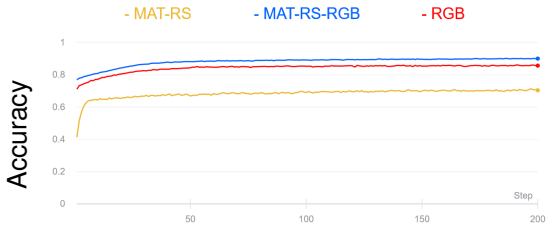


Figure 5.41: 3DOM no RGB - test accuracy

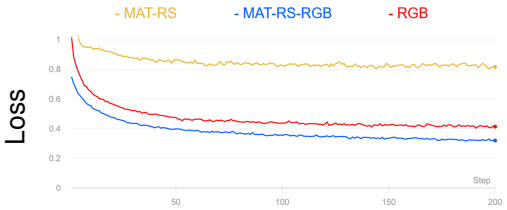


Figure 5.42: 3DOM no RGB - test loss

	RGB	MAT-RS	MAT-RS-RGB
OA	0.86	0.69	0.91
IoU			
Ground	74.48%	57.73%	83.98%
Grass	34.49%	17.70%	67.84%
Shrub	42.78%	10.84%	66.52%
Tree	86.38%	58.79%	91.34%
Facade	88.48%	59.73%	89.18%
Roof	59.94%	44.05%	68.59%

Table 5.9: 3DOM dataset no RGB experiments - overall accuracy and per class intersection over union

3DOM - Gaussian noise

Name	Features
RGB	RGB
RGB	RGB + Gaussian noise
MAT-RS	RGB, interior and exterior radii and separation angles
MAT-RS-NOISE	RGB, interior and exterior radii and separation angles + Gaussian noise

Table 5.10: Gaussian noise experiments legend

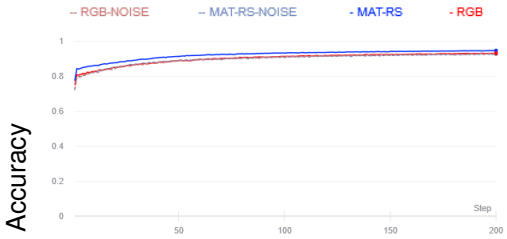


Figure 5.43: 3DOM Gaussian noise - train accuracy

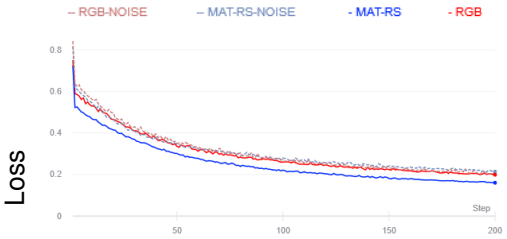


Figure 5.44: 3DOM Gaussian noise - train loss

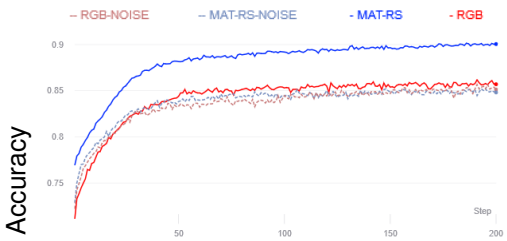


Figure 5.45: 3DOM Gaussian noise - test accuracy

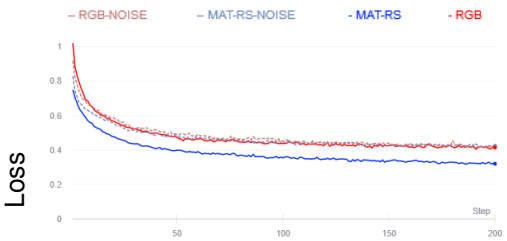


Figure 5.46: 3DOM Gaussian noise - test loss

Figures 5.43 to 5.46 show the performance of the training procedure using the default parameters in red and the radius and separation angle in blue; furthermore it displays the outcomes obtained with the same features, in light blue and pink, on the data-set modified with Gaussian noise. Gaussian noise is statistical noise having a probability density function (PDF) equal to that of the normal distribution [Wikipedia, 2020]. It was added to the data-set using the software Mapple[1], with a standard deviation of one. The graphs show how noise influence the usability of the MAT information; in fact the results for the noisy data-set overlap with those of the clean data-set using the default features. This happens because

[1] https://3d.bk.tudelft.nl/liangliang/software.html

the medial balls constructed for this data-set are over shrunk, leading to a set of values that are almost identical for the full data-set. As these features are so similar, they don't add any information, thus the MAT-RS-NOISE curves become equal to the RGB ones. Instead RGB values are not subject to noise since they are not a geometric information, thus the RGB-NOISE and the RGB performances are similar.

5.1.6 SHREC 2020 - bisector angles and spoke vectors

Name	Features
RGB	RGB
MAT-RS	RGB, interior and exterior radii and separation angles
MAT-SP	RGB, interior spoke vectors
MAT-BIS	RGB, bisector angles

Table 5.11: SHREC experiments legend

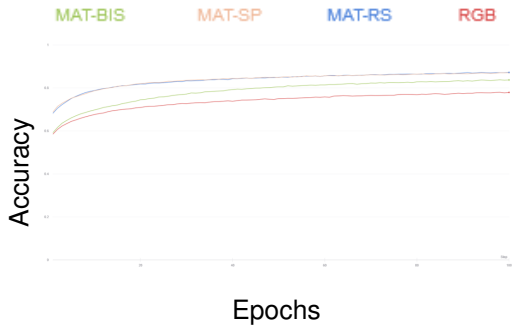


Figure 5.47: SHREC data-set - train acc.

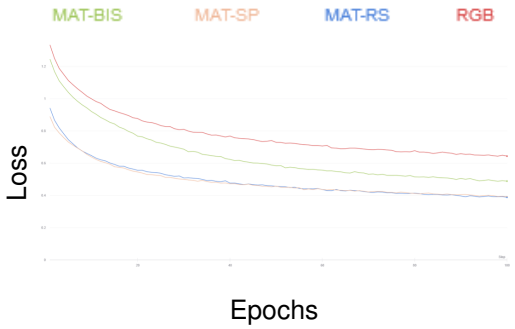


Figure 5.48: SHREC data-set - train loss

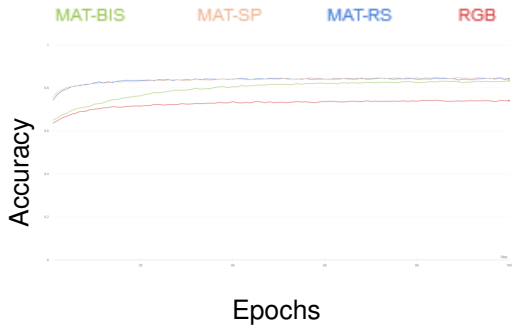


Figure 5.49: SHREC data-set - test acc.

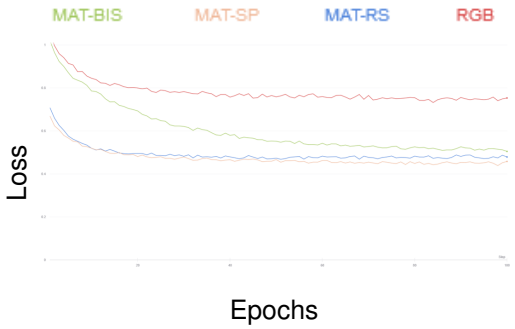


Figure 5.50: SHREC data-set - test loss

	RGB	MAT-RS	MAT-SP	MAT-BIS	
OA	0.84	0.89	0.87	0.84	
IoU					
Undefined	08.63%	09.71%	13.94%	09.22%	+5.31
Building	24.39%	54.49%	43.22%	38.64%	+30.10
Car	13.68%	22.22%	28.05%	22.25%	+14.37
Ground	88.10%	95.76%	94.65%	92.98%	+7.66
Pole	00.00%	00.00%	00.00%	00.00%	
Vegetation	73.85%	79.34%	76.10%	69.00%	+5.49

Table 5.12: SHREC data-set spoke vectors and bisector angles experiments experiments - overall accuracy and per class intersection over union

Figures 5.47 to 5.50 show training and test accuracy and loss for experiments conducted with the SHREC2020 data-set. These are meant to investigate how PointNet++ algorithm performs on a lager subset of CycloMedia's internal data-set and to test different MAT properties as features.

In general, the experiments conducted demonstrate that PointNet++'s performance increases greatly for the SHREC data-set, with respect to the behavior shown in Section 5.1.4. This happens because the data-set is composed by more point clouds, which are segmented in fewer classes. Furthermore, the data-set does not present significant labeling errors, as it was manually labeled. However, the data-set still has a high decreasing density of points and class imbalance. The latter does not seem to affect results greatly, as OA and IoU values for most classes are sufficient, see Table 5.12. Instead, the former leads to the high difficulty to recognize poles in the data-set, see Table 5.12.

Three trials were conducted, see Table 5.11. First, only RGB values were used as features, in order to set a comparison basis. Then, these were combined with interior and exterior radii and separation angles, interior spoke vectors or bisector angles. Results displayed in Figures 5.47 to 5.50 and in Table 5.12 demonstrate that all MAT values improve the performance of the PointNet++ algorithm. In particular, MAT-RS and MAT-SP experiments present overlapping curves and similar overall accuracy values. This result is expected, as spoke vectors represent the vector from the medial point to the feature point, disclosing similar information to radii and separation angles, see Section 2.4. The building class presents the highest grow in IoU value, followed by the car and ground ones. As can be seen in Figures 5.27 and 5.29 interior radii and separation angles clearly distinguish horizontal classes from vertical or scattered ones. Buildings, ground and vegetation IoU values grow in the MAT-RS experiment, while car and undefined IoU values improve in the spoke vector one. Instead, the MAT-BIS does not improve any class the most, however it presents a steeper curve compared to the other trials. Thus, this experiment could lead to better results compared to other experiments, if the algorithm runs for more epochs.

5.2 Medial axis transform as a descriptor for a geometric partition

In this section, the main outcomes for the experiments are presented, using the MAT to partition a point cloud. The partitioned point cloud is then used to create the superpoint graph and input to a graph convolution deep learning algorithm. The experiments refer to the 3DOM and SynthCity data-sets, the reason is that these data-sets have more suitable properties for a graph convolution method, compared to the internal data-set used in the Section 5.1. In fact, the computation of the MAT results more complete and accurate as these data-sets present less noise, have an homogeneous density of points and objects are fully represented.

The analysis of the four geometric descriptors used in the SPG algorithm is outlined; these are linearity, planarity, scattering and verticality. These are then compared to the radii, separation angles and the medial bisectors; the goal is to understand how characteristic a descriptor is and how similar they are among each others. Furthermore, the numerical and visual comparison between the partition methodologies is presented. Last, the outputs of the deep learning algorithm are commented. These refer to training procedures with 10 epochs and default hyperparameters. Thus, the results may be subject to change if modifications in the deep learning hyperparameters are made. In this analysis, first the default partition was computed; this should serve as comparison basis for the following experiments. The partitioned point cloud was then input to the deep learning algorithm. Then, the partitions using the MAT information were computed and fed to the deep learning algorithm using the same parameters to the default partition; this choice enabled the comparison between the trials and the quantification of the influence of the partition on the overall results.

The experiments on the synthetic data-sets show dissimilar results. In general,

- Introducing new features in the cut-pursuit algorithm increases the number of homogeneous parts identified.
- In the 3DOM data-set the growth is higher; instead the results are more stable in the Synthcity data-set.
- The medial bisector results diverge greatly if compared to the other trials with the 3DOM data-set, differently than the SynthCity data-set.
- A bigger number of segments does not automatically produce a worse result, as more parts could lead to a more accurate segmentation.
- A bigger number of segments leads to a higher computational demand, which is not a desirable result.
- The introduction of descriptors associated with the exterior MAT produces a confused and non-reliable segmentation.
- The medial bisector and the interior radii and separation angle values define the shape of objects.
- In general the training procedure using the default graph and produces more accurate outputs.

5.2.1 3DOM dataset experiments

For all the trials with the 3DOM data-set, the regularization strength was set to 0.1, an exception was made for trial 7, where the medial bisector was used. Here the regularization strength was fixed at 0.4. This is because keeping the regularization constant, the number of parts would increase by around 50 times if compared to the default configuration. Thus, the experiments with the medial bisector should not be compared with the others on a quantity basis. The experiments show that, keeping the regularization parameter constant, the number of parts identified by the cut-pursuit algorithm grows when MAT information is added (Table 5.13). Only the change in edge weight reduces the number of parts for all point clouds. It can be observed that the highest increase in number of parts occurs in trial 2 for most point clouds, where the exterior and interior radii and separation angles are added to the default values. Furthermore it can be seen how the same information, without the default values, produces a much smaller number of segments. Thus, an elevated number of descriptors creates difficulties in the minimization process of the cut-pursuit algorithm.

		Default and							
	1 default	2 MAT	3 rad sep in	4 rad sep out	5 rad in out	6 sep in out	7 bisector	8 MAT	9 edge weight
train1	642	1502	1212	1269	1661	869	646*	893	595
train2	709	1620	1651	1187	1782	943	844*	708	504
eval1	632	1831	1200	1220	1578	719	670*	946	528
eval2	765	1757	1109	1282	1626	1075	997*	701	556
val1	1685	3511	2300	2467	3084	1539	2218*	1689	1334

*experiments with the medial bisector were conducted setting the regularization parameter to 0.4

Table 5.13: 3DOM geometric partition experiments - number of parts

Feature descriptors

Figures 5.51 to 5.62 show the geometric descriptors used in this phase of the research. In particular, Figures 5.51 to 5.54 depict linearity, planarity, scattering and verticality; Figures 5.55 to 5.58 show the bisector angle, first the three values are visualized together as normal vectors, then each value is shown alone. Last Figures 5.59 to 5.62 show the radii and separation angles. In Figures 5.51 to 5.54 can be seen how the four default descriptors describe objects. In the first three images the difference between roofs and ground or grass classes is minimal; only verticality helps to recognize them, if the roof has a slope. This might not be important in the partition phase, where adjacent simple structures should be separated from each other. However, the four descriptors are later used to compute the superpoint attributes and are input in PointNet as features. Thus this might influence the results at a later stage.

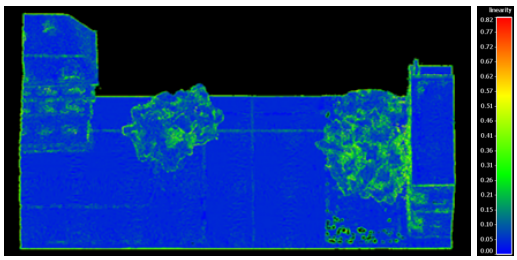


Figure 5.51: 3DOM - linearity

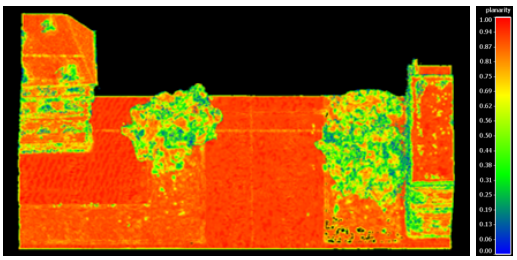


Figure 5.52: 3DOM - planarity

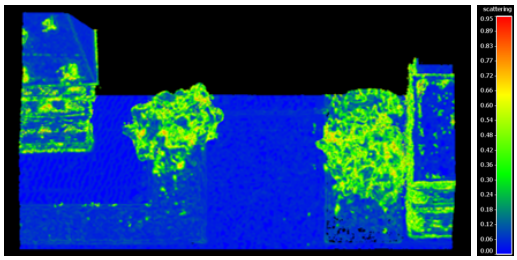


Figure 5.53: 3DOM - scattering

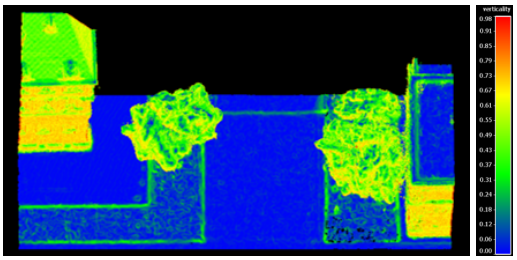


Figure 5.54: 3DOM - verticality

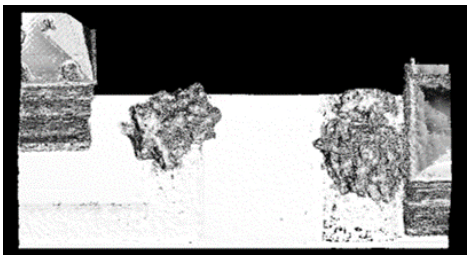


Figure 5.55: 3DOM - bisectors as normals

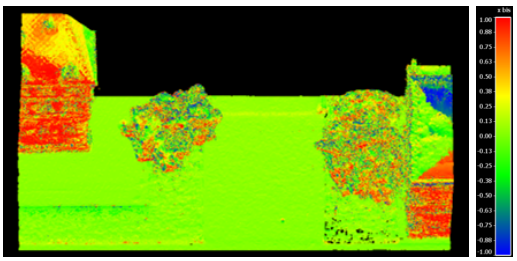


Figure 5.56: 3DOM - medial bisector 1

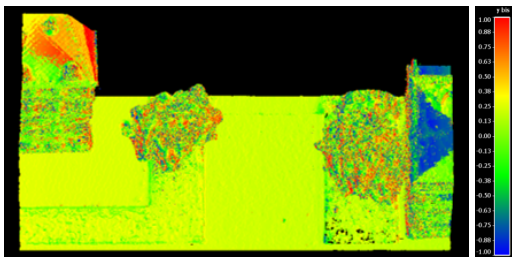


Figure 5.57: 3DOM - medial bisector 2

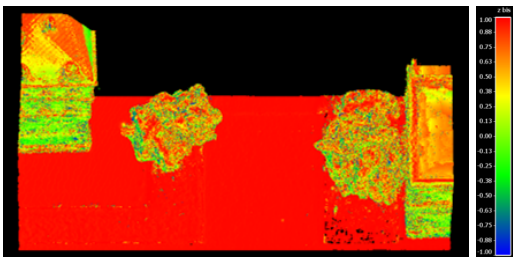


Figure 5.58: 3DOM - medial bisector 3

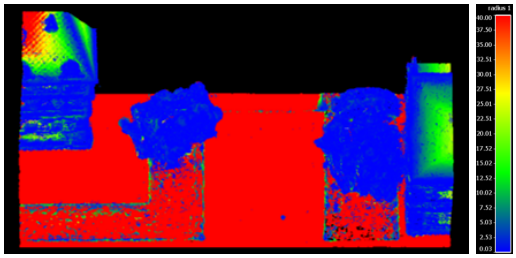


Figure 5.59: 3DOM - interior radius

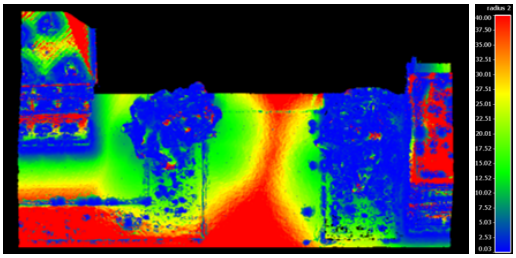


Figure 5.60: 3DOM - exterior radius

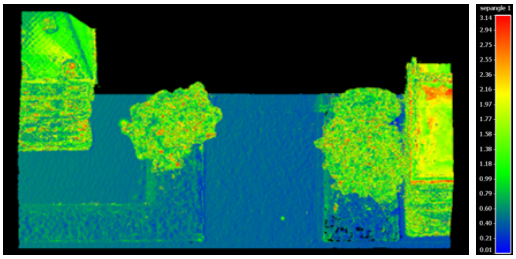


Figure 5.61: 3DOM - interior separation angle

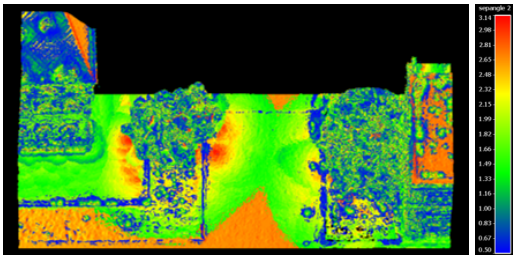


Figure 5.62: 3DOM - exterior separation angle

Partitioning

For all twelve descriptors can be observed how they give importance to small structures in the buildings' facades. This happens because of how the data-set is created, dense image matching, and consequently because of its high resolution. In particular, in Figures 5.63 to 5.70 can be observed how small structures can influence the partition. This problematic factor could be decreased by changing the MAT denoise values.

Also looking at Figures 5.8 and 5.10, it can be observed that the exterior radii and separation angles values creates incorrect divisions between the horizontal classes, grass and ground. In fact these values quantify the relation and influence between objects in the point cloud. For this reason, they might not be suited for the task of geometric partition. In Table 5.14 can be seen that the experiment where only exterior values were used produced the worst overall accuracy of all.

Figures 5.63 to 5.70 show the different partitions results. In general, all fail to recognize simple and homogeneous structures. For example, in Figure 5.63 the left side of the building is considered as one part; instead, its two sides should be separated. Figure 5.69 shows that the medial bisector well captures the different simple shapes. However this information introduces ambiguity in parts of the roof. As mentioned above, the exterior MAT information create incorrect partitions, Figures 5.66 to 5.68. Last Figure 5.71 shows a partition similar to the default one; here however the different edge strength penalized the creation of small parts, and thus simplified the final result.

Semantic segmentation

Table 5.14 gives an overview of the results of the deep learning procedure for the different trials. These aim to be a mean of comparison between different partition methods having homogeneous ground conditions, such as deep learning hyperparameters. However, the content of Table 5.14 could be different if, for each trial, the best hyperparameters were selected. Furthermore, the random factors that might occur in the training procedures and the small number of epochs may also influence the results. These factors should be taken into consideration when critically reading this table. It can be seen how the default configuration produces better overall accuracy (OA) results, while trial 5 produces the most homogeneos IoU per class results, while having the second best OA results. For most experiments, it can be observed that per class outputs are hardly balanced. In fact, sufficient results for one kind of objects produce almost zero IoU in the remaining classes. This can be seen in trial 1, in the grass class, in trial 2 in the shrub class and so on. The fact that this peculiarity is present in almost all trials should mean that it is not associated with the integration of the MAT or the partition method.

		Default and							
	1 default	2 MAT	3 rad sep in	4 rad sep out	5 rad in out	6 sep in out	7 bisector	8 MAT	9 edge weight
OA	74.36	64.78	62.57	23.53	71.53	64.65	67.25*	65.27	66.51
IoU									
Ground	47.48	30.89	29.97	30.89	30.95	36.96	59.66*	21.41	55.01
Grass	02.68	43.67	16.51	20.82	49.88	02.54	00.02*	36.67	19.27
Shrub	28.89	01.55	42.48	07.06	37.37	49.32	57.51*	00.12	36.70
Tree	66.78	66.46	67.37	29.85	71.00	17.93	64.13*	00.29	52.09
Facade	79.01	28.24	48.54	02.31	58.46	30.75	67.64*	39.48	63.25
Tree	51.74	21.54	00.63	13.40	42.23	44.23	03.08*	48.41	00.04

*experiments with the medial bisector were conducted setting the regularization parameter to 0.4

Results shown in this table are in %

Table 5.14: 3DOM geometric partition experiments - overall accuracy and intersection over union

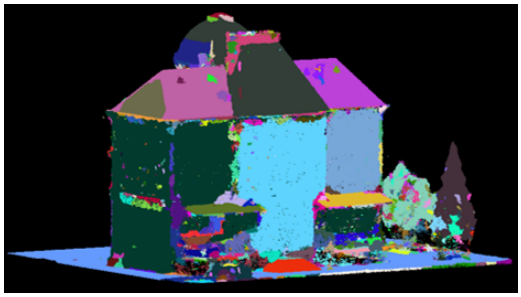


Figure 5.63: 3DOM - default partition

Figure 5.64: 3DOM - default + MAT partition

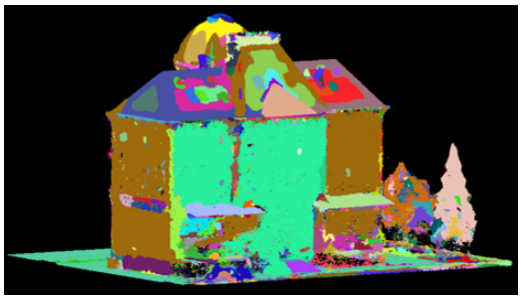


Figure 5.65: 3DOM - default + rad sep in partition

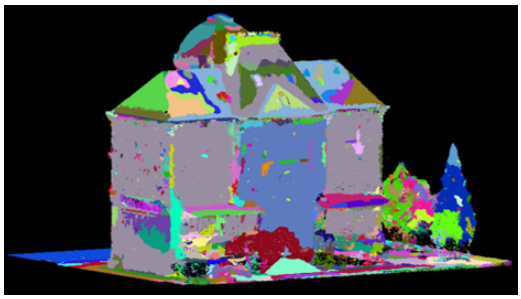


Figure 5.66: 3DOM - default + rad sep out partition

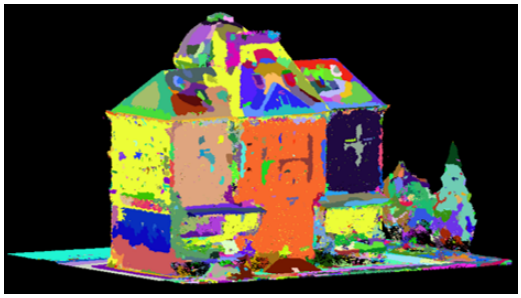


Figure 5.67: 3DOM - default + rad in out partition

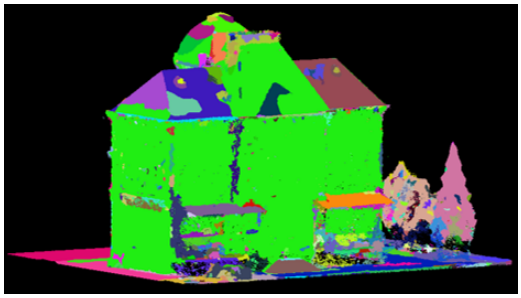


Figure 5.68: 3DOM - default + sep in out partition

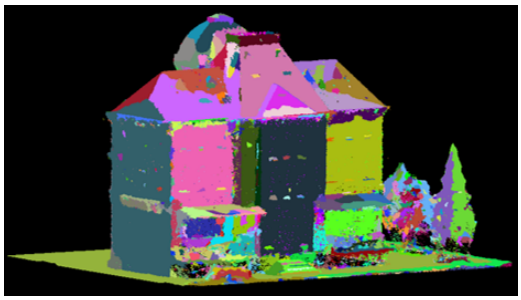


Figure 5.69: 3DOM - default + bisector partition

Figure 5.70: 3DOM - medial axis transform partition

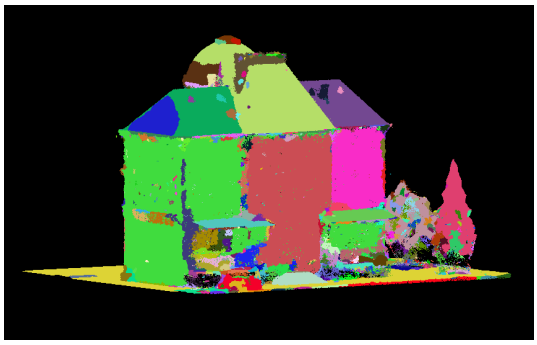


Figure 5.71: 3DOM - edge weight partition

5.2.2 SynthCity dataset experiments

Table 5.15 shows the number of parts, or superpoints, for each point cloud in the SynthCity data-set, setting the regularization strength parameter to 0.4. Experiments conducted on this data-set present different outputs with respect to those carried on with the 3DOM data-set, see Section 5.2.1. Here, the medial bisector led to an increase of the number of parts by a factor of 2 to 3, for this reason it was kept constant in the experiments. Instead, it led to and increase by a factor of 50 for the 3DOM data-set. Furthermore, in general the number of parts remained stable for all experiments, while it decreased for all point clouds using only MAT derived information. Last, modifying the edge weight did not lead to fewer superpoints, instead, the number slightly increased. The reason for these differences could be associated with the peculiarities of the two data-sets. In fact, the SynthCity data-set presents a more regular and more sparse distribution of points. This is because the data-set simulates a Velodyne scanner, thus angles of objects are sharper and surfaces are smoother than those of the 3DOM data-set. For these reasons, the information added to the cut-pursuit algorithm does not change the number of parts greatly.

Semantic segmentation

Table 5.16 gives an overview of the results of the deep learning procedure for a subset of the trials listed above. These were selected because they were judged more meaningful than the others, they aim to be a mean of comparison between different partition methods having homogeneous ground conditions, such as deep learning hyperparameters. Outputs are similar to those of the 3DOM data-set, see Section 5.2.1. In fact, the default configuration produces a better overall accuracy value and the best IoU results for the building, car, street furniture and tree classes. The MAT trial produces the best results for natural ground, ground, pole-like and road classes. However, most classes present very similar IoU values, thus these results can not be interpreted as an improvement related to the MAT information.

				Default and					
	1	2	3	4	5	6	7	8	9
	default	MAT	rad - sep in	rad - sep out	rad in out	sep in out	bisector	MAT	edge weight
area1	656	755	663	627	671	686	1575	426	701
area2	840	991	838	963	941	833	2176	565	981
area3	770	1017	866	783	996	853	1735	676	896
area4	832	875	859	930	1001	946	2001	548	912
area5	1064	1212	1220	1251	1237	497	2661	693	1172
area6	886	1202	963	1077	956	589	3053	791	969
area7	501	493	481	496	507	789	472	348	499
area8	472	382	467	512	550	1121	684	337	525
area9	557	780	674	742	740	506	1220	536	639

Table 5.15: SynthCity dataset radius and separation angle experimets - number of parts

	1 default	Default and 2 MAT	7 bisector	9 edge weight
OA	89.04%	85.28%	85.84%	80.71%
IoU				
Building	97.75%	96.36%	92.14%	94.81%
Car	66.37%	56.16%	42.47%	38.17%
Natural ground	00.20%	44.38%	01.83%	01.46%
Ground	06.76%	12.20%	11.39%	03.90%
Pole-like	42.52%	48.16%	01.04%	24.77%
Road	41.53%	46.56%	00.00%	41.52%
Street furniture	29.59%	15.87%	00.00%	18.20%
Tree	98.34%	96.69%	66.00%	94.80%
Pavement	00.04%	00.00%	00.00%	00.00%

Table 5.16: SynthCity dataset radius and separation angle experimets - overall accuracy and per class intersection over union

5.3 Medial axis transform as an attribute for graph convolution

In this section, the experiments to modify the superedge information to be used for graph convolution are presented. In particular, two main researches were performed, using the mean radius and separation angle or using their maximum and minimum values for each superpoint. In both procedures, the values were pre-computed and read by the partition algorithm. Then, the indices of the points that were part of each superpoint were retrieved, and the mean or the maximum and minimum values were computed. These were then added as nodes attributes in the SPG graph. Last, for each pair of superpoints, the difference between their node attributes was computed and associated with the corresponding edge features.

Tables 5.17 and 5.18 show the results for each trial. On the left column, those obtained using the default partition method and the default graph attributes are displayed. For each point cloud, the partition was computed twice and the resulting superpoints appear only slightly different than each other. However, results of the training process vary greatly for each class, as seen in Figures 5.73 and 5.77. This fact introduces uncertainty on the validity of the experiments. Figures 5.73 to 5.75 and Figures 5.77 to 5.79 outline also how results vary in each trial, while overall accuracy values are really similar than each other. Looking at Tables 5.17 and 5.18 and Figures 5.76 to 5.79, it can not be concluded that the different MAT edge attributes fully determine the outputs obtained.

	1 default	Default and 2 mean rad sep	3 mean rad sep
OA	74.36%	70.12%	74.04%
IoU			
Ground	47.48%	35.70%	29.40%
Grass	02.68%	20.97%	15.62%
Shrub	28.89%	60.32%	22.53%
Tree	66.78%	69.54%	70.86%
Facade	79.01%	43.78%	27.53%
Roof	51.74%	10.23%	62.69%

Table 5.17: 3DOM graph attributes mean radii and separation angles - overall accuracy and intersection over union

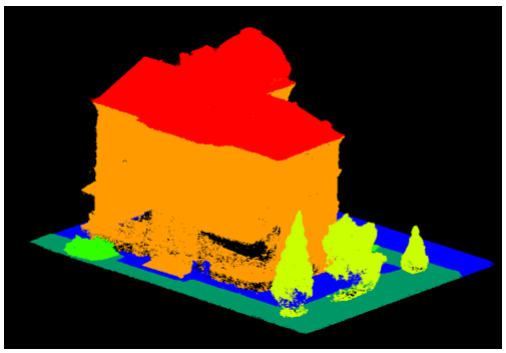


Figure 5.72: 3DOM - ground truth

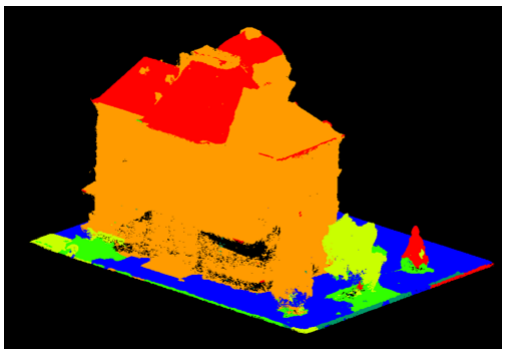


Figure 5.73: 3DOM - default graph attributes

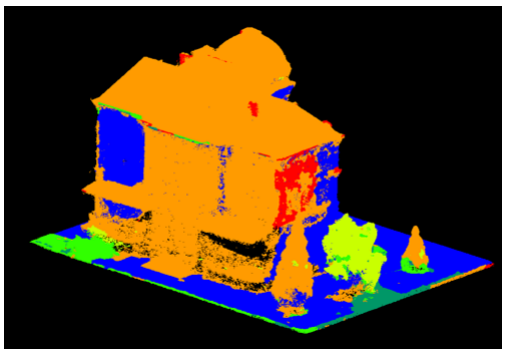


Figure 5.74: 3DOM - default graph attributes + mean radii and separation angles

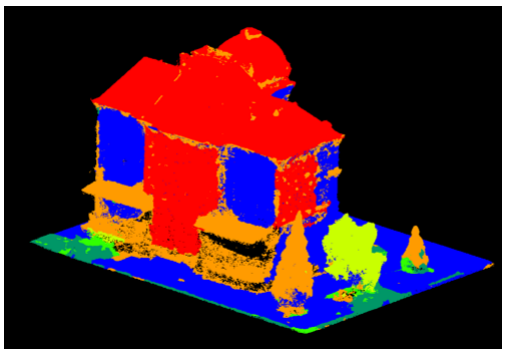


Figure 5.75: 3DOM - mean radii and separation angles graph attributes

	1	Default and 2	3
	default	max min rad sep	max min rad sep
OA	72.64%	73.64%	72.77%
IoU			
Ground	71.68%	53.63%	23.87%
Grass	00.11%	00.00%	00.08%
Shrub	05.62%	18.99%	37.53%
Tree	50.05%	47.52%	62.47%
Facade	69.48%	72.18%	33.28%
Roof	01.09%	28.69%	00.00%

Table 5.18: 3DOM graph attributes maximum and minimum radii and separation angles - overall accuracy and intersection over union

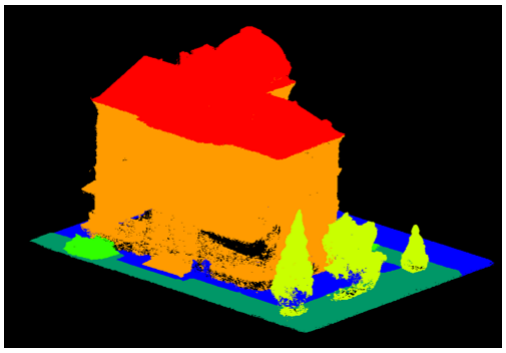


Figure 5.76: 3DOM - ground truth

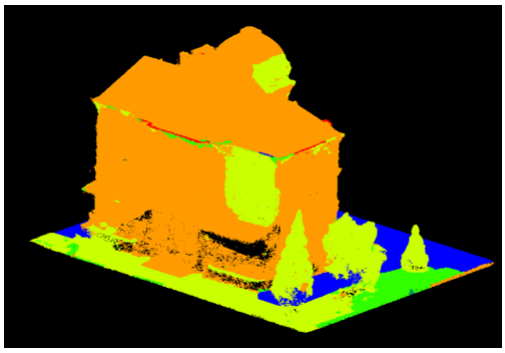


Figure 5.77: 3DOM - default graph attributes

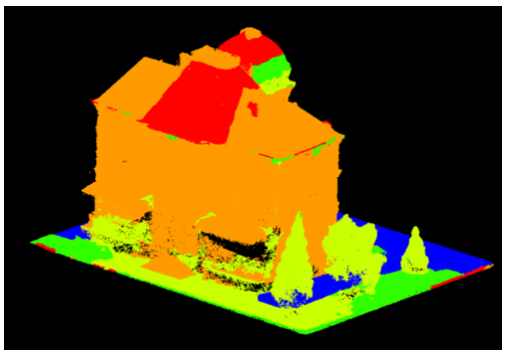


Figure 5.78: 3DOM - default graph attributes + maximum and minimum radii and separation angles

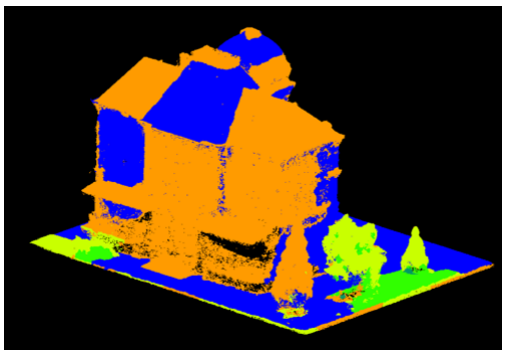


Figure 5.79: 3DOM - maximum and minimum radii and separation angles graph attributes

6 Conclusions and future work

This chapter summarizes the main outputs of this thesis. In Section 6.1 the answers to the research questions proposed in Chapter 1 are given, then the main scientific contributions of this research are listed in Section 6.1.1. Finally, a discussion on the improvements on the methodology which could be performed in the coming month is outlined in Section 6.2.

6.1 Research questions

To conclude this thesis, the research questions proposed in Chapter 1 are answered. First, the subquestions are considered, finally a general conclusion is given by answering the main research question.

How can the 3D medial axis transform be used to give context to local points in a point cloud, making the unary classification per point stronger?

The 3D medial axis transform can be used to give context to local points if integrated as an extra feature in a deep learning algorithm. In this research, radii, separation angles and MAT coordinates were employed. They were added to the algorithm PointNet++ in different combinations, first interior and exterior radii and separation angles were used together, then the 3D MAT coordinates and the interior MAT coordinates were used. Only the radii and separation angles experiments proved to be useful to improve the results for all of the tested data-sets, while the MAT coordinates introduced ambiguity in the algorithm. Furthermore, medial bisectors and spoke vectors were integrated in PointNet++ and tested on one data-set. The use of these information proved to improve OA and IoU in the experiment. Thus, radii, separation angles, medial bisectors and spoke vectors can make the classification per point stronger.

Which of the 3D medial axis transform properties are most helpful?

The radii, separation angles, medial bisectors and spoke vectors values derived from the 3D MAT can be integrated in a deep learning algorithm. Experiments conducted in this research show that radii and separation angles can be successfully used together as extra point features. However, radii information proved to be more suited than the separation angles' one. In fact, using the former alone would increase the accuracy while lessening the loss value more, if compared to the separation angles. Furthermore, it was shown that the interior MAT information can be used alone or combined with the exterior one, producing almost equal results. Furthermore, spoke vectors values and bisector angles were used in two separate training procedures. Spoke vectors output almost equal results to radii and separation angles. Instead medial bisectors produced slightly worse results, that however improved at

a faster pace. Thus, all MAT information tested demonstrated to improve OA and IoU in the experiments of this research. In particular, interior radii properties proved to be helpful.

How can the 3D medial axis transform be used to partition a point cloud into semantically homogeneous shapes?

The 3D medial axis was used in this research as a geometric descriptor in the algorithm Superpoint Graph. In particular radii, separation angles, medial bisectors and medial sheet values were integrated in the cut-pursuit algorithm. The first three were used as geometric descriptors; the latter was used to make the SPG more similar to the 3D MAT, strengthening relations between points belonging to the same medial sheet. Theoretically, these should improve the partition into homogeneous shapes, and consequently the overall results of the deep learning algorithm. However, results presented in this research do not show any improvement in the partition.

How can the 3D medial axis transform be used to enrich the node and edge information in a graph used as input for a graph convolution neural network?

In this research, the 3D MAT was used to enrich superpoints and superedges of the SPG, using the default SPG graph. The procedure included two experiments, associating the mean radii and separation angles to the superpoint or their maximum and minimum values. Then for both experiments, the superedge features would be computed as the difference of this information in adjacent superpoints. Results of this research do not enable to draw final conclusions on this topic. In fact, MAT information can result ambiguous if associated with the SPG. This is because the two graphs present theoretical similarities but not practical ones. Thus MAT information does not add valuable insights if associated with the default SPG.

Can the 3D medial axis transform be used to improve the accuracy of an existent deep learning method?

Yes, it can be used to improve the accuracy of an existing deep learning algorithm. The experiments conducted with PointNet++ show that results for the three data-sets used in this research improve when adding radii and separation angles as extra features in the deep learning algorithm. For all data-sets, OA and IoU values improve for all classes. In particular, for the 3DOM data-set, IoU values increase from 34.49% to 67.84% for grass and from 42.78% to 66.52% for shrubs. For the SynthCity data-set, IoU values increase from 45.49% to 56.82% for ground and from 84.92% to 93.16% for natural ground. Last for the internal data-set they increase from 22.21% to 56.29% for road.

How important are the construction parameters of the 3D medial axis transform in the deep learning method?

When computing the unstructured and structured MAT many parameters need to be taken in consideration. The initial radius, denoise planar and denoise preserve values can influence the cleanness and the degree of relation between objects in a geographical point cloud. Furthermore, the segmentation configuration can structure the MAT based on the medial bisector, separation angle or ball overlap. Thus, analyzing the MAT for different point clouds and understanding the best configuration for each, is relevant in the final usability of the 3D MAT.

How does the performance on the real data-set compare with the one obtained on the synthetic one?

Experiments conducted in PointNet++, where the MAT was used as an extra point feature, were tested on CycloMedia's internal data-set. Results show similar trends to those of the SynthCity and 3DOM data-set. In fact, adding the MAT radii and separation angles showed an improvement in the overall results. In general however, results present a much lower accuracy and higher loss. This is because of the characteristics of this data-set, such as fast decreasing point density and high variance in represented scenes and objects.

How can the properties of the 3D medial axis transform be exploited in different deep learning methods for point cloud semantic segmentation?

In this research, the main MAT properties used in a deep learning algorithm were the radii, separation angles, medial bisectors and medial sheets. These values were used in three main ways, in two algorithms PointNet++ [Qi et al., 2017] and Superpoint Graph [Landrieu and Simonovsky, 2017]. Results of the experiments show that the radii and separations angles' values can be successfully integrated as point features in a deep learning algorithm. Instead, it should not be added as geometric descriptor in the cut-pursuit algorithm or used to enrich the edge relations of the SPG, if the graph does not present a similar structure to that of the MAT.

6.1.1 Scientific contributions

The main scientific contributions of this research are:

- I proved that the 3D medial axis transform can be successfully integrated in a point-based deep learning algorithm as an extra feature. In particular, two MAT properties linked to the geometry of the medial atom were recognized as valuable, the radius and separation angle. These can make the classification of the point stronger, without introducing redundancy, without slowing the algorithm or adding computational effort.

- I demonstrated that radius and separation angle values can be successfully used for three different types of data-set, a dense image matching data-set of a urban model, a synthetic data-set simulating a mobile laser scanner and one of real data captured with mobile laser scanner.

- I proved that radii and separation angles values improve the IoU outputs for all types of semantic classes. In particular they can be used for horizontal classes, such as ground, shrubs and grass, where the RGB information is not as distinctive.

6.2 Discussion

The analysis and experiments conducted in this research propose a simple and effective way to integrate the MAT values as features in a point-based deep learning algorithm. Furthermore, they proved that adding the MAT values as geometric descriptors does not improve the geometric partition of the point cloud. Finally, they showed that enriching the superpoints and superedges in the SPG does not lead to better segmentation results.

To analyze the outputs of this research, one has to consider different factors that influence them. These concern 3D MAT computation and deep learning hyperparameters setting. The former include the characteristics of data-sets used, the normal vectors estimation and orientation methodologies and the number of parameters for the MAT computation. The latter determine the quality of the segmentation algorithm.

Noise is a factor that highly influences the usability of the 3D MAT. Data-sets used in this research don't present high Gaussian noise, thus the impact of the MAT integration can be quantified. However, results might be less meaningful if input data-sets present strong noise. The acquisition method is another data-sets' characteristics that influences the usability of the 3D MAT. Dense image matching point clouds present smoother angles between points with respect to Lidar point clouds; this factor could lead to faulty segments in the structured MAT. The computation and reorientation of normal vectors also has an impact on the final MAT quality. In fact, normal vectors' orientation determine the distinction between interior and exterior MAT and the quality of the MAT structuration.

Last, different parameters are used to construct and structure the 3D MAT; these should be chosen based on the characteristics of the data-sets. Similarly, many hyperparameters can be modified to tune the deep learning outcome. In this research, MAT parameters and deep learning hyperparameters were set and used throughout the course of the study. Visual analysis was fundamental in choosing the MAT parameters, while many trial runs were necessary to tune the deep learning outcome. Additionally, normal vectors were computed and reoriented manually for each data-set, to ensure the quality of the results. Methods used in this thesis are time consuming and thus not optimal for a real life application.

6.3 Future work

The 3D medial axis transform proved to add valuable information in a point-wise deep learning method for different data-sets. Future research for this topic can be pursued in three directions.

Analysis of different types of data-sets This research focused on the use of two mobile based laser scanner data-sets and one complete urban model, obtained with dense image matching. As acquisition methods influence the output of the research, the use of the 3D MAT on aerial based data-sets could be further analyzed.

Automatic parameters section per data-set In this study, the MAT construction and structuration parameters were chosen and investigated visually. These determine the degree of interaction between objects, cleanness and sheets characteristics in the MAT. Data-sets

present different sensitivity to the above parameters, based on what they represent and their density. The automatic selection of MAT parameters based on data-sets characteristics could facilitate the integration of the MAT for large data-sets.

Analysis of deep learning outcomes In this research, overall accuracy and intersection over union metrics were used to quantify the outputs obtained. Deeper analysis on the influence of the MAT on deep learning algorithms could lead to better understanding of the results and enhancements in the integration of the MAT. The analysis could be obtained with feature baseline attribution methods. These quantify the importance of each feature within the layers of the network, choosing one hyperparameter as a baseline in a gradient based approach.

Furthermore, the analysis of the SPG algorithm outlined two additional research directions on this algorithm.

Use of MAT adjacency as SPG In this research, MAT sheets were used as superpoints in the SPG algorithm and their adjacency was computed using the 3D Delaunay triangulation. However, the MAT graph presents an adjacency structure that differs from the above one. Adjacency information of the MAT sheets could be used to define the Superpoint graph. In order to maintain the structure of the SPG algorithm, where each point is associated to only one superpoint, this could be done in three ways. First, one could use either the interior or the exterior adjacency; however this would lead to the loss of valuable information. Second, one could duplicate points in the original point cloud; however this would cause extra computational effort. Last, one could use the MAT point cloud directly as input of the algorithm. In this way, each point would be uniquely related to a segment, without losing interior or exterior information.

Direct use of MAT point cloud In this study, the boundary point cloud was used as input in the SPG algorithm. MAT values such as radii, separation angles and segment ids were associated with the input points, leading to the use of only interior segment ids of the structured MAT. In a future research, the MAT point cloud could be input in the algorithm. This choice would enable the use of interior and exterior information simultaneously; furthermore, it would lead to a straightforward use of the MAT adjacency information to construct the Superpoint graph. In this implementation, labels of the boundary point cloud could be first associated with the MAT ones and later aggregated back on the original point cloud.

A Segmented data-sets

In this appendix, results of the experiments detailed in Sections 3.3 and 5.1 are shown. In particular, for each data-set the outputs are visualized on one point cloud and commented.

Name	Features
RGB	RGB
MAT-RS	RGB, interior and exterior radii and separation angles
MAT-C	RGB, interior and exterior MAT coordinates
MAT-I	RGB, interior MAT coordinates
MAT-SP	RGB, interior spoke vectors
MAT-BIS	RGB, bisector angles

Table A.1: Experiments legend

A.1 3DOM data-set

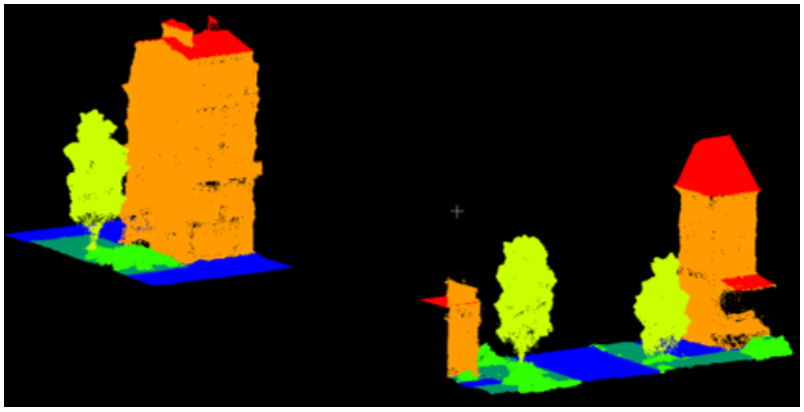


Figure A.1: 3DOM data-set - ground truth

Figures A.2 to A.5 display the outputs of the PointNet++ algorithm, where the 3D medial axis transform was integrated as a feature. Figure A.1 represents the ground truth labeled point cloud. Figure A.2 shows results using RGB values as features. Here, it can be seen that roofs are wrongly labeled as ground, in blue, while the shrub class and the grass one are often mistaken. Figures A.3 and A.4 outline that the use of coordinates as features introduces ambiguity in the algorithm. In the left side of the point cloud, it can be observed

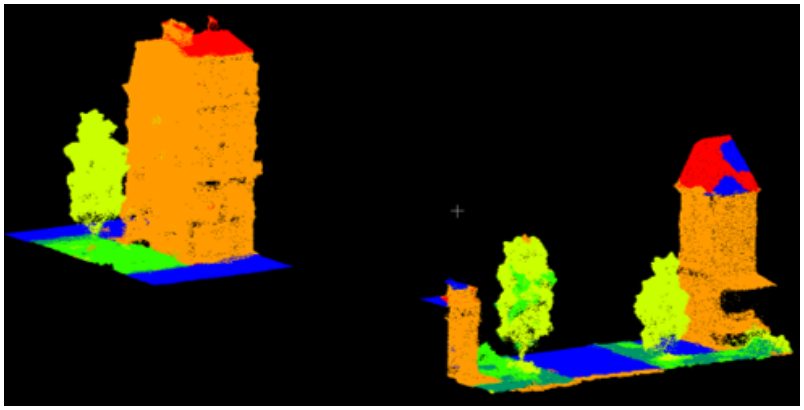


Figure A.2: 3DOM data-set - RGB

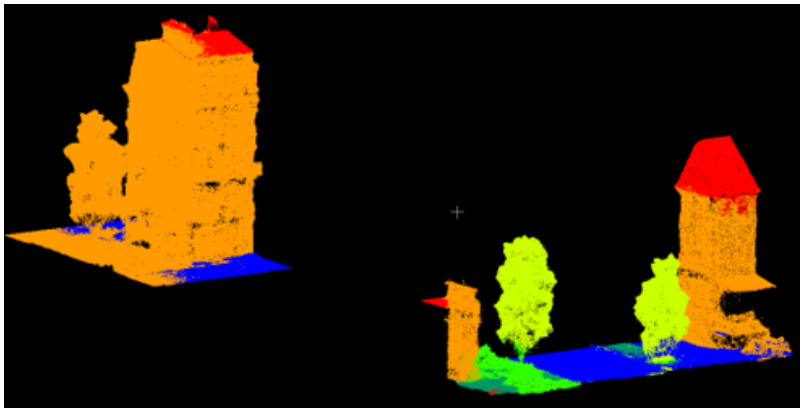


Figure A.3: 3DOM data-set - MAT-C

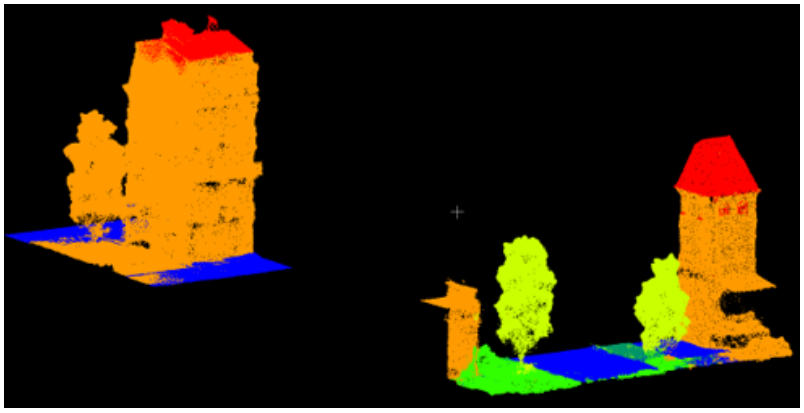


Figure A.4: 3DOM data-set - MAT-I

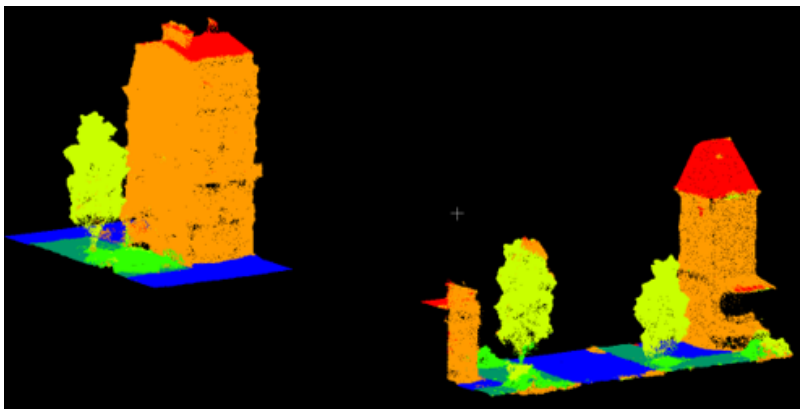


Figure A.5: 3DOM data-set - MAT-RS

that grass, shrub and tree classes are identified as buildings. Last, Figure A.5 shows the output of radii and separation angles as features. In the pictures, it can be observed how the identification of grass and shrub classes is cleaner; furthermore roof points are well classified.

A.2 SynthCity data-set

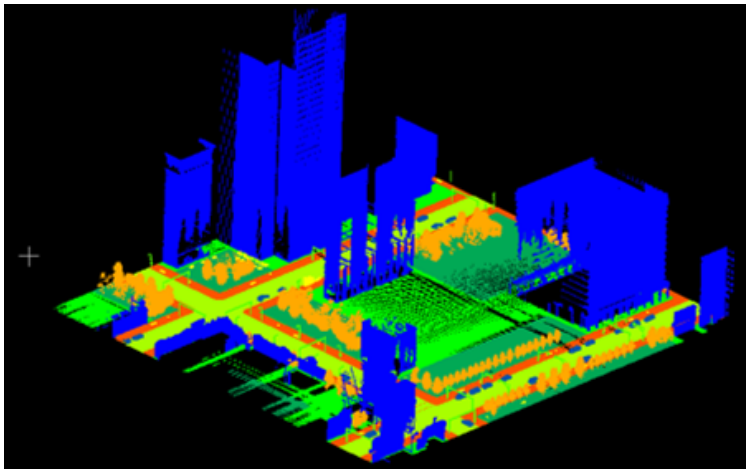


Figure A.6: SynthCity data-set - ground truth

Figures A.7 to A.10 display the results for the SynthCity data-set. Figure A.6 represents the point cloud labeled with ground truth values. Figures A.7 and A.10 display similar results; in the first, only RGB values were used, instead in the second radii and separation angles information were integrated in the algorithm. Also Figures A.8 and A.9 show similar outputs.

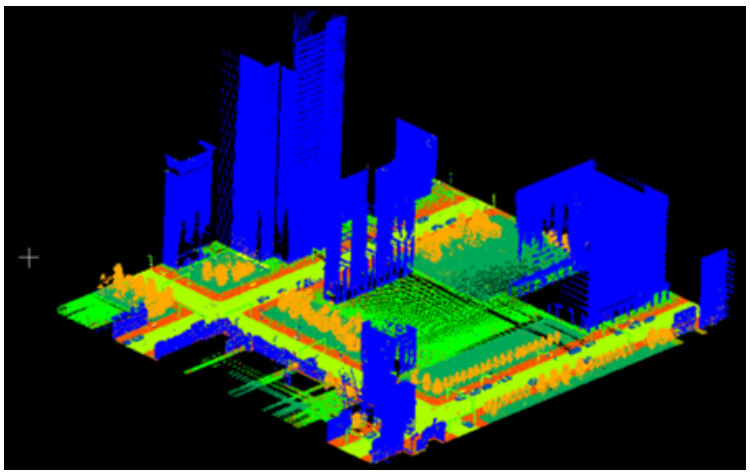


Figure A.7: SynthCity data-set - RGB

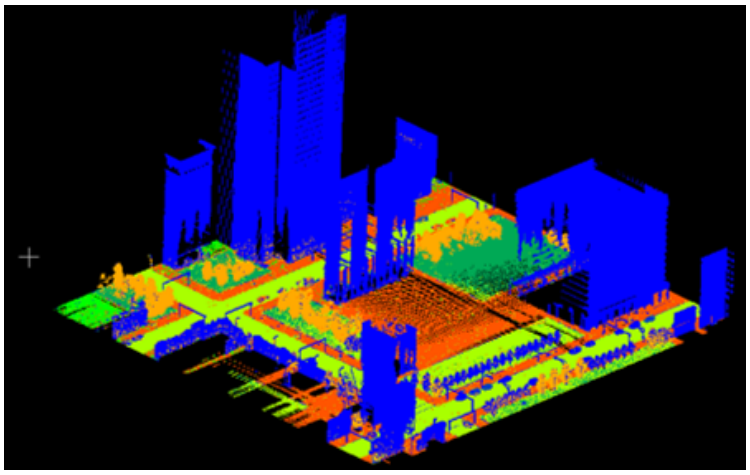


Figure A.8: SynthCity data-set - MAT-c

Here, it can be seen that using the MAT exterior and interior, or only interior, coordinates led to the wrong classification of the natural ground class, which was labeled as pavement, in red. Furthermore, street furniture objects and trees were labeled as buildings, in blue.

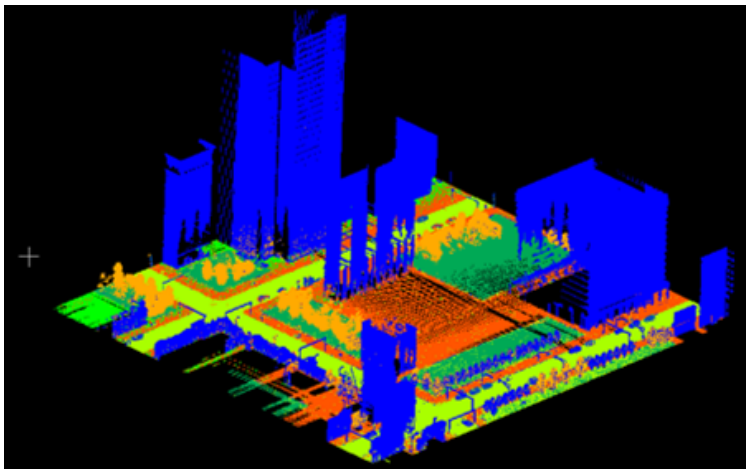


Figure A.9: SynthCity data-set - MAT-i

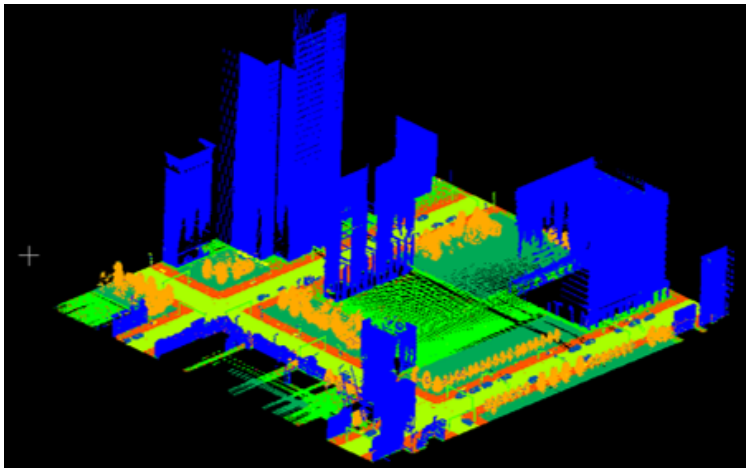


Figure A.10: SynthCity data-set - MAT-rs

A.3 CycloMedia data-set

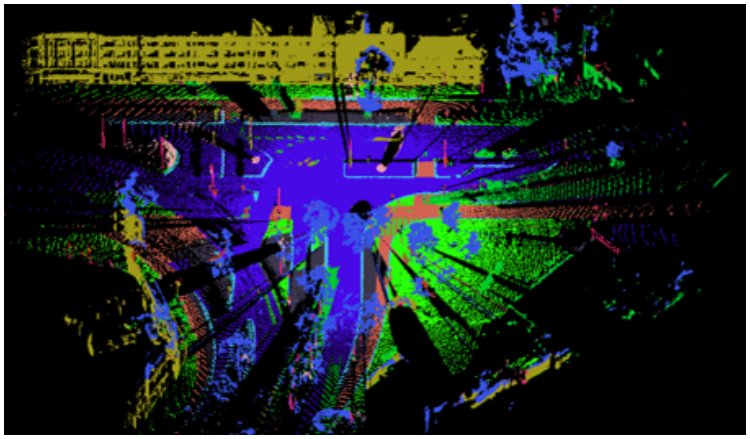


Figure A.11: CycloMedia data-set - ground truth

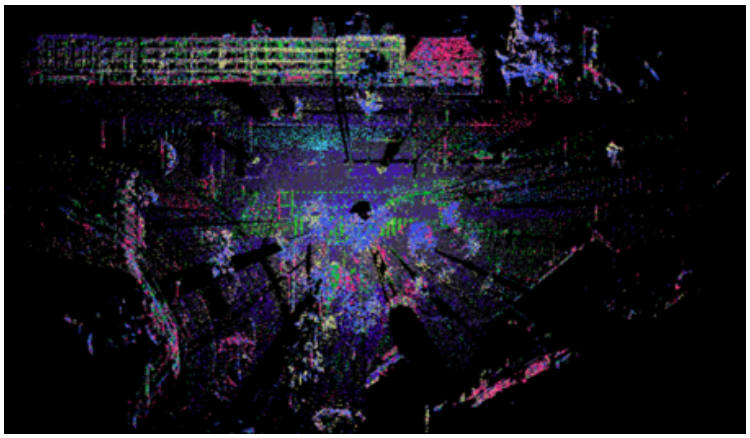


Figure A.12: CycloMedia data-set - RGB

Figures A.12 to A.14 show the outputs of experiments conducted with a subset of CycloMedia's internal data-set. Figure A.1 represents the ground truth labeled point cloud. In general, results are affected by the high variability of objects represented and of points' density. It can be observed how classes are not identified smoothly, but often adjacent points present different labels. In Figure A.12, it can be seen that the algorithm could hardly detect buildings and the main horizontal class, the road class. Instead, Figure A.14 depicts how radii and separation angles could improve the classification of the road class, while detecting zebra-crossings and curbs. Last, in all the images it can be seen that many vertical elements are labeled as a support pole, in pink. In particular, the use of MAT coordinates led to the classification of the full point cloud in one class.

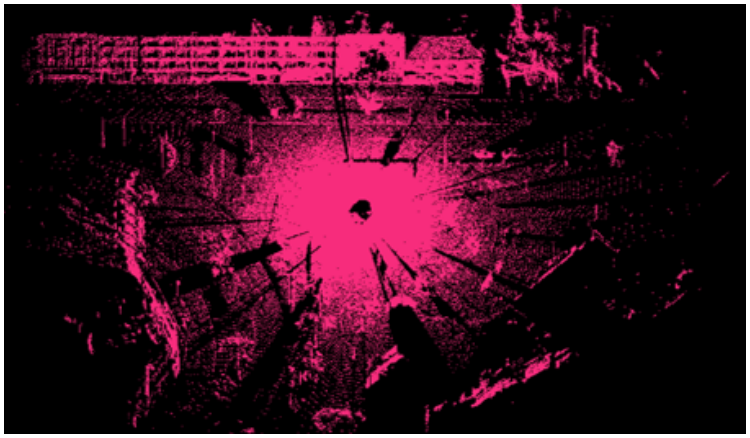


Figure A.13: CycloMedia data-set - MAT-c

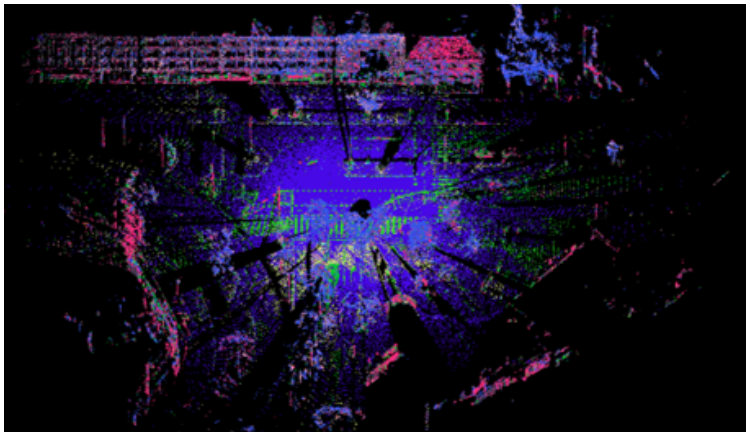


Figure A.14: CycloMedia data-set - MAT-rs

A.3.1 SHREC data-set

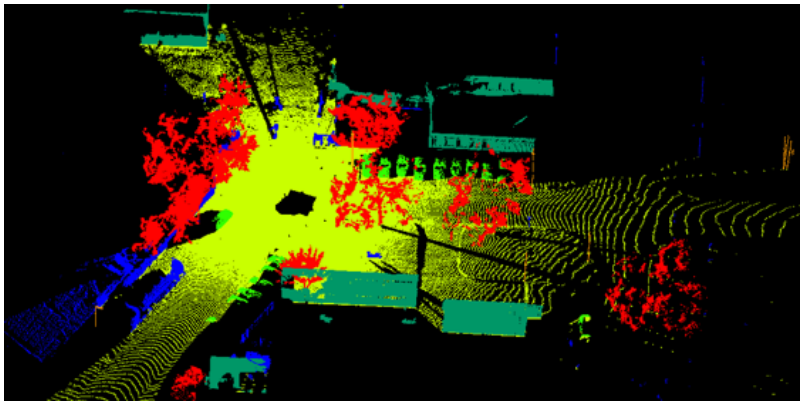


Figure A.15: SHREC data-set - ground truth

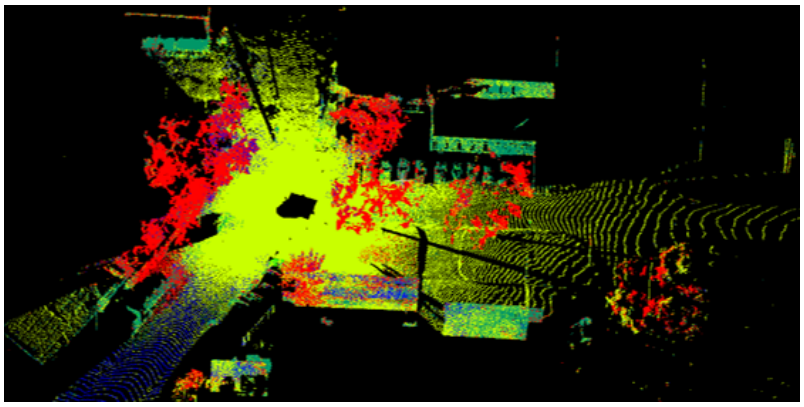


Figure A.16: SHREC data-set - RGB

Figures A.16 to A.19 display results of the experiments conducted with the SHREC data-set. Here, radii and separation angles were tested, furthermore interior spoke vectors and bisector angles were integrated as features. As the data-set is labeled in fewer classes, results are better than those presented for the CycloMedia's data-set. Figure A.16 shows results using RGB values as features. As in the experiments shown above, points are not smoothly classified. Instead, in Figures A.17 to A.19 improvements can be observed. In fact, MAT derived information helped classify nearby points with the same label. As can be seen in the images, the three experiments present similar outputs, proving that all three information are valuable.

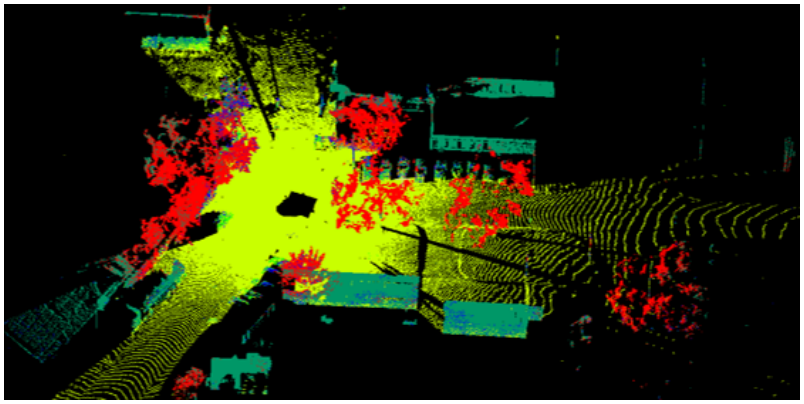


Figure A.17: SHREC data-set - MAT-rs

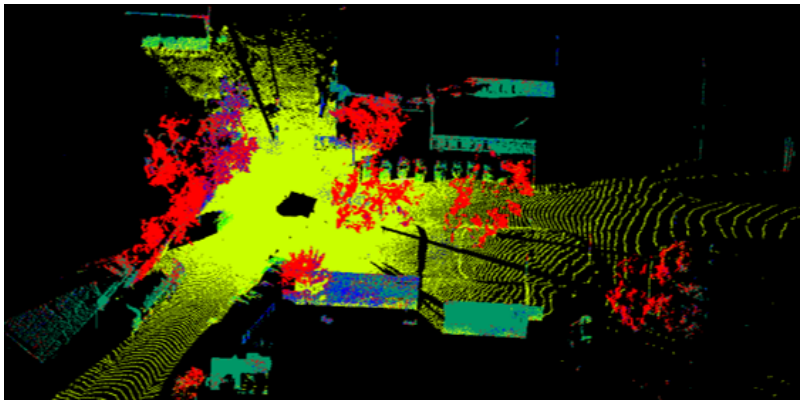


Figure A.18: SHREC data-set - MAT-sp

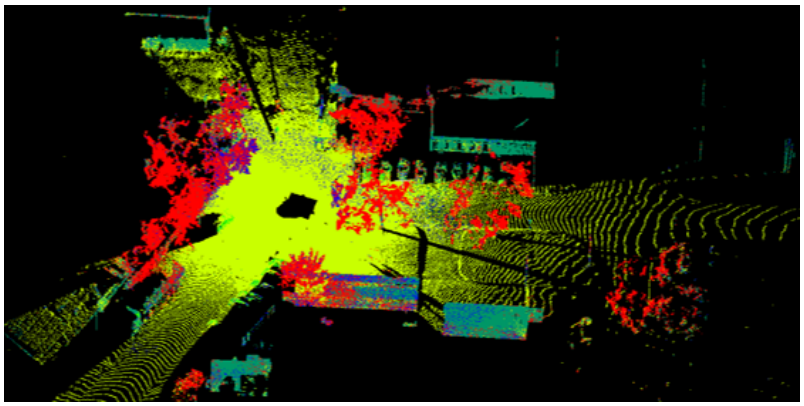


Figure A.19: SHREC data-set - MAT-bis

B Deep learning glossary

B.1 Concepts of neural networks

Layers and architectures Each layer is composed of multiple neurons; these are the basic unit of a neural network. The neuron is modeled after the brain's biological neuron; it can take multiple inputs, it applies a function to them and produces an output. The output can be then processed by another neuron or it can represent the final prediction of the architecture.

A neural network is composed of different layers (Figure **??**). These are:

- Input layer: it contains the input. Each neuron in this layer corresponds to an attribute of the data-set, an attribute is a quality of the data-set.
- Hidden layer: it processes the data through different activation functions. An activation function is used to model complex relationships between features. A Neural Network can contain one or more hidden layers, these can be fully connected or rely only on a subset of the previous neurons' outputs. A fully connected layer is one that takes as input the output of all neurons in the previous layer.
- Output layer: it is the final layer in the network, which contains the prediction of the model. A model is a data structure that stores a representation of a data-set.

Weights Within a neural network, the connections between all the elements are called synapses (lines in Figure **??**). They carry the weights which are applied to each element that "passes" on them. The weights are scalar values that can amplify or reduce the importance of a given neuron. Updating the weights is the main way in which a neural network learns.

Biases Biases are scalar values associated with the input of the neural network. These values guarantee the activation of a neuron for each layer even if their signal is weak [Patterson and Gibson, 2017]. Like weights, they are updated during the training process.

Activation functions Activation functions model how data, weights and biases are combined to derive a more complex representation of the input. In hidden layers, activation functions propagate the output of one layer to the next one, thus each neuron in the neural network applies an activation function to its input. Activation functions are used to introduce non-linearity to the model.

Loss functions Loss functions determine the quality of the model's output. The loss function computes a distance metric between the predicted values and the true ones, called loss value. The loss value is used in the optimization process, which consists in minimizing the distance, or error, between the above.

Optimizers Optimizers determine how the neural network is updated based on the output of the loss function.They use the loss value to update of weights and biases in the network applying the backpropagation algorithm.

Hyperparameters In neural networks, parameters used to improve the quality and speed in training are called hyperparameters. They control the optimization function, making sure that the model does not overfit or underfit the data and that it learns its structure quickly [Patterson and Gibson, 2017].

B.2 Components of a neural network

Architectures

Convolutional Neural Network uses convolutions to extract features from local regions of an input. Most CNNs contain a combination of convolutional, pooling and affine layers. [WILDML, 2019]

Multi Layer Perceptron is a Feedforward Neural Network with multiple fully-connected layers that use nonlinear activation functions to deal with data which is not linearly separable. [WILDML, 2019]

Recurrent Neural Network: it models sequential interactions through a hidden state, or memory. At each time step, an RNN calculates a new hidden state based on the current input and the previous hidden state. The term “recurrent” stems from the facts that at each step the same parameters are used and the network performs the same calculations based on different inputs. [WILDML, 2019]

Spatial Transformer Network: it applies spatial transformations, such as affine and projective transformations to crop out and scale-normalize the appropriate region of the input. [**?**]

Layers

Affine layer: it is a fully-connected layer in a neural network. It means that each neuron in the previous layer is connected to each neuron in the current layer. WILDML [2019]

Batch norm layer normalizes the incoming activations and outputs a new batch where the mean equals 0 and standard deviation equals 1. It subtracts the mean and divides by the standard deviation of the batch. A batch is a subset of the input. [Fortuner, 2019]

Convolution layer applies convolution to its input. A convolution is a linear operation that involves multiplication of weight (kernel/filter) with the input. [Fortuner, 2019]

Dropout layer takes the output of the previous layer’s activations and randomly sets a certain fraction (dropout rate) of the activations to 0, cancelling or ‘dropping’ them out. [Fortuner, 2019]

Max-pooling layer selects the maximum value from a patch of features. Pooling layers help to reduce the dimensionality of a representation by keeping only the most salient information.

Activation functions

Linear a straight line function where activation is proportional to input.

ReLu (Rectified Linear Units), its formula is $max(0, Z)$. Where Z is the weighted input of a neuron.

Sigmoid takes a real value as input and outputs another value between 0 and 1, its formula is $S(Z) = \frac{1}{1+e^{-Z}}$. Where Z is the weighted input of a neuron.

Softmax calculates the probabilities distribution of the event over 'n' different events. In other words, this function will calculate the probabilities of each target class over all possible target classes.

Loss functions

Cross-entropy loss, or log loss measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

Optimizers

Adaptive Moment Estimation (ADAM) computes adaptive learning rates for each parameter.

- First, it computes the exponentially weighted average of past gradients.
- Second, it computes the exponentially weighted average of the squares of past gradients.
- Third, these averages have a bias towards zero and to counteract this a bias correction is applied.
- Lastly, the parameters are updated using the information from the calculated averages.

Hyperparameters

Learning rate is a coefficient that is multiplied with the error gradient during the optimization process. It determines how much the parameters are adjusted, in fact a small learning rate lead to a slow optimization while large ones lead to larger updates.

Regularization helps prevent overfitting by controlling the weights in the neural network. It is represented by the greek letter lambda.

Momentum prevents the model to get stuck in local minima.

B.3 Terminology

Accuracy percentage of correct predictions made by the model.

Attribute a quality describing an observation (e.g. color, size, weight). In Excel terms, these are column headers.

Bias metric the average difference between your predictions and the correct value for that observation.

Bias term allow models to represent patterns that do not pass through the origin.

Categorical variables variables with a discrete set of possible values. Can be ordinal or nominal.

Classification threshold the lowest probability value at which we're comfortable asserting a positive classification.

Confusion matrix table that describes the performance of a classification model by grouping predictions into 4 categories.

- True positives: correctly predicted positive outputs.
- True negatives: correctly predicted negative outputs.
- False positives: incorrectly predicted positive outputs.
- False negatives: incorrectly predicted negative outputs.

Continuous variables variables with a range of possible values defined by a number scale.

Convergence a state reached during the training of a model when the loss changes very little between each iteration.

Embedding it maps an input representation, such as a word or sentence, into a vector.

Encoder a stack of several recurrent units where each accepts a single element of the input sequence, collects information for that element and propagates it forward.

Epochan epoch describes the number of times the algorithm sees the entire data set.

Feature with respect to a dataset, a feature represents an attribute and value combination. Color is an attribute. Color is blue is a feature. In Excel terms, features are similar to cells.

Feature vector a list of features describing an observation with multiple attributes.

Gradient the slope of our cost function at our current parameter values.

Instance a data point, row, or sample in a dataset. Another term for observation.

Label the "answer" portion of an observation in supervised learning.

Model a data structure that stores a representation of a dataset (weights and biases). Models are created/learned when you train an algorithm on a dataset.

Normalization restriction of the values of weights in regression to avoid overfitting and improving computation speed.

Noise any irrelevant information or randomness in a dataset which obscures the underlying pattern.

Observation a data point, row, or sample in a dataset. Another term for instance.

Outlier an observation that deviates significantly from other observations in the dataset.

Overfitting it occurs when your model learns the training data too well and incorporates details and noise specific to your dataset.A model is overfitting when it performs great on your training/validation set, but poorly on your test set.

Parameters they are properties of training data learned by training a machine learning model or classifier. They are adjusted using optimization algorithms and unique to each experiment. Examples of parameters include: weights in an artificial neural network.

Precision is the percentage of true guesses that were actually correct.

Recall it measures how "sensitive" the classifier is at detecting positive instances. In other words, for all the true observations in our sample, how many were found.

Regularization it is a technique utilized to combat the overfitting problem. This is achieved by adding a complexity term to the loss function that gives a bigger loss for more complex models.

Test Set a set of observations used at the end of model training and validation to assess the predictive power of your model. It is used to assess how generalizable is the model to unseen data.

Training Set a set of observations used to generate machine learning models.

Underfitting it occurs when your model over-generalizes and fails to incorporate relevant variations in your data that would give your model more predictive power. A model is underfitting when it performs poorly on both training and test sets.

Validation Set a set of observations used during model training to provide feedback on how well the current parameters generalize beyond the training set. If training error decreases but validation error increases, the model is likely overfitting and you should pause training.

C Reproducibility self-assessment

C.1 Marks for each of the criteria

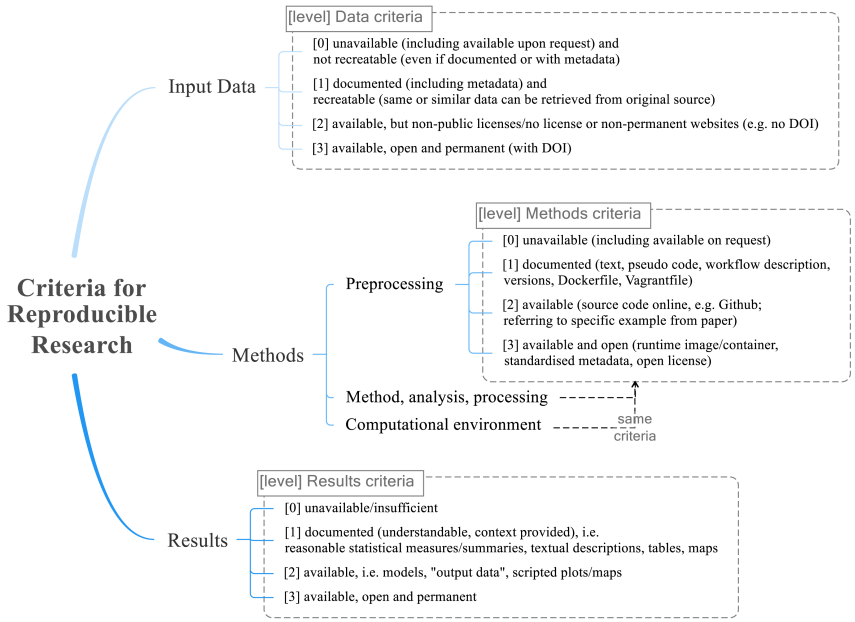


Figure C.1: Reproducibility criteria

Criteria for reproducible research:

- Input data: 2
- Pre-processing: 1
- Methods: 1
- Computational environment: 0
- Results: 1

C.2 Self-reflection

The experiments and analyses of this study were performed in collaboration with the company CycloMedia Technology. Information related to part of the input data or the computational environment can not be publicly available. In particular,

- Input data: in this research, three data-sets were used. The 3DOM data-set is available online on request, while the SynthCity data-set is downloadable through a non permanent website. The SHREC data-set, a subset of CycloMedia's internal data-set, is available online and downloadable through a non permanent website. The remaining part of CycloMedia's internal data-set is not public.
- Pre-processing: data-sets pre-processing methods are documented with text and workflow descriptions. Most of the methods were performed through CloudCompare. Data readers and conversions written in Python are not publicly available.
- Methods: he source code of the deep learning algorithms used in this research is available online on GitHub; the modifications to the algorithms are documented through text and workflow descriptions.
- Computational environment: this thesis was carried out using CycloMedia's tools and servers, thus the computational environment is not reproducible. Python libraries and software used are documented.
- Results: due to the constraints of the above criteria, results are only understandable and documented through text, graphs and summary information.

Bibliography

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer.

Boost (2015). Boost C++ Libraries. `http://www.boost.org/`. Last accessed 2015-06-30.

Boulch, A., Saux, B. L., and Audebert, N. (2017). Unstructured point cloud semantic labeling using deep segmentation networks. In *3DOR*.

Che, E., Jung, J., and Olsen, M. (2019). Object Recognition, Segmentation, and Classification of Mobile Laser Scanning Point Clouds: A State of the Art Review. *Sensors*, 19.

Chollet, F. (2017). *Deep Learning with Python*. Manning.

Coursera (2019). *Machine Learning*.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T. A., and Nießner, M. (2017). ScanNet: Richly-annotated 3d Reconstructions of Indoor Scenes. *CoRR*, abs/1702.04405.

Docs, R. t. (2019). *Machine Learning Glossary*.

Fortuner, B. (2019). Machine learning glossary.

Golovinskiy, A., Kim, V. G., and Funkhouser, T. (2009). Shape-based Recognition of 3d Point Clouds in Urban Environments. *International Conference on Computer Vision (ICCV)*.

Graham, B., Engelcke, M., and van der Maaten, L. (2017). 3d semantic segmentation with submanifold sparse convolutional networks. *CoRR*, abs/1711.10275.

Griffiths, D. and Boehm, J. (2019). Synthcity: A large scale synthetic point cloud. *CoRR*, abs/1907.04758.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. (2019). Deep learning for 3d point clouds: A survey.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., and Pollefeys, M. (2017). Semantic3d.net: A new Large-scale Point Cloud Classification Benchmark. *CoRR*, abs/1704.03847.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., and Markham, A. (2019). Randla-net: Efficient semantic segmentation of large-scale point clouds.

Hua, B., Tran, M., and Yeung, S. (2017). Point-wise convolutional neural network. *CoRR*, abs/1712.05245.

Huang, J. and You, S. (2016). Point Cloud Labeling using 3d Convolutional Neural Network.

Jatavallabhula, K. M., Smith, E., Lafleche, J.-F., Tsang, C. F., Rozantsev, A., Chen, W., Xiang, T., Lebaredian, R., and Fidler, S. (2019). Kaolin: A pytorch library for accelerating 3d deep learning research.

Jiang, M., Wu, Y., and Lu, C. (2018). Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *CoRR*, abs/1807.00652.

Kang, Z. and Ning, L. (2019). Pyramnet: Point cloud pyramid attention network and graph embedding module for classification and segmentation. *CoRR*, abs/1906.03299.

Landrieu, L. (2019a). *Deep Learning for 3D Point Cloud Semantic Segmentation*.

Landrieu, L. (2019b). Implementing pointnet on aerial lidar data.

Landrieu, L. (2019c). Implementing PointNet on Aerial LiDAR Data.

Landrieu, L. and Boussaha, M. (2019). Point cloud oversegmentation with graph-structured deep metric learning. *CoRR*, abs/1904.02113.

Landrieu, L. and Simonovsky, M. (2017). Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. *CoRR*, abs/1711.09869.

Li, J., Chen, B. M., and Lee, G. H. (2018a). So-net: Self-organizing network for point cloud analysis. *CoRR*, abs/1803.04249.

Li, Y., Bu, R., Sun, M., and Chen, B. (2018b). Pointcnn. *CoRR*, abs/1801.07791.

Liu, Y., Fan, B., Xiang, S., and Pan, C. (2019). Relation-shape convolutional neural network for point cloud analysis. *CoRR*, abs/1904.07601.

Ma, Y., Guo, Y., Liu, H., Lei, Y., and Wen, G. (2020). Global context reasoning for semantic segmentation of 3d point clouds. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2920–2929.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*.

MediaWiki (2015). Sor filter.

NVIDIA (2019). *Deep learning glossary*.

Özdemir, E., Toschi, I., and Remondino, F. (2019). A multi-purpose benchmark for photogrammetric urban 3d reconstruction in a controlled environment. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-1/W2:53–60.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Patterson, J. and Gibson, A. (2017). *Deep Learning: A Practitioner's Approach*. O'Reilly Media, Inc., 1st edition.

Peters, R. (2018a). *Geographical point cloud modelling with the 3D medial axis transform*. PhD thesis, Delft University of Technology. ISBN: 978-94-6186-899-2.

Peters, R. (2018b). masbcpp.

Peters, R. (2018c). skel3d.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). PointNet: Deep Learning on Point Sets for 3d Classification and Segmentation. *CoRR*, abs/1612.00593.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *CoRR*, abs/1706.02413.

Roynard, X., Deschaud, J., and Goulette, F. (2017). Paris-lille-3d: a large and high-quality ground truth urban point cloud dataset for automatic segmentation and classification. *CoRR*, abs/1712.00032.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition.

Schmohl, S. and Sörgel, U. (2019). Submanifold sparse convolutional networks for semantic segmentation of large-scale als point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:77–84.

Sharma, S. (2017). *Epoch vs Batch Size vs Iterations*.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M., and Kautz, J. (2018a). Splatnet: Sparse lattice networks for point cloud processing. *CoRR*, abs/1802.08275.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., and Kautz, J. (2018b). Splatnet: Sparse lattice networks for point cloud processing. cite arxiv:1802.08275Comment: Camera-ready, accepted to CVPR 2018 (oral); video summary: https://www.youtube.com/watch?v=5Lbg4l-t-DU.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 945–953.

Tchapmi, L. P., Choy, C. B., Armeni, I., Gwak, J., and Savarese, S. (2017). Segcloud: Semantic segmentation of 3d point clouds. *CoRR*, abs/1710.07563.

TERRA3D (2019). A convolution operator for point clouds.

Thomas, H., Qi, C. R., Deschaud, J., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. *CoRR*, abs/1904.08889.

Tsang, S.-H. (2019). *Review: STN — Spatial Transformer Network (Image Classification)*.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2018a). Dynamic Graph CNN for Learning on Point Clouds. *CoRR*, abs/1801.07829.

Wang, Z., Zhang, L., Zhang, L., Li, R., Zheng, Y., and Zhu, Z. (2018b). A Deep Neural Network With Spatial Pooling (DNNSP) for 3-D Point Cloud Classification. *IEEE Transactions on Geoscience and Remote Sensing*, PP:1–11.

Wijmans, E. (2018). Pointnet++ pytorch.

Wikipedia (2019). *Deep Learning*.

Wikipedia (2020). Gaussian noise — wikipedia, the free encyclopedia.

WILDML (2019). *Deep Learning Glossary*.

WILDML (2019). Deep learning glossary.

Xia, S. and Wang, R. (2018). Extraction of residential building instances in suburban areas from mobile LiDAR data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 144:453–468.

Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., and Huang, W. (2017). A Convolutional Neural Network-Based 3d Semantic Labeling Method for ALS Point Clouds. *Remote Sensing*, 9:936.

To my supervisors, Ravi, Weixiao, Bas and Arjen, for the discussions, insights and time.
To my "geomatics" friends, for sharing desperation and laughter over this two years.
To the CycloMedia team, for welcoming me in a great work environment.
To Francesco, for the long hours on the phone and the short trips.
To my parents and family, for the unconditional support.
To Sarah, for being there for me, in Delft and Haarlem.
To Tania, for sharing our meals and our cultures.
To everyone who has been there with me.

Thank you, dankjewel, ευχαριστω', grazie!
Giulia

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Helvetica.

TU Delft