DOI:10.15774/PPKE.ITK.2020.010

Three-dimensional Scene Understanding in Mobile Laser Scanning data

A thesis submitted for the degree of
Doctor of Philosophy

Balázs Nagy

Scientific adviser:
Csaba Benedek, Ph.D

Faculty of Information Technology and Bionics
Pázmány Péter Catholic University

Budapest, 2020

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

Acknowledgements

First of all I would like to express my sincere gratitude to my supervisor Csaba Benedek for his continuous support during my Ph.D study and work, for his motivation and patience. He is leading the way for me since my BSc studies and not just as a supervisor but a friend.
I have been a member of the Machine Perception Research Laboratory (MPLAB) at SZTAKI since my BSc studies, so I would like to special thank to Prof. Tamás Szirányi head of MPLAB for providing remarks and advice regarding to my research and always supporting my work.
Pázmány Péter Catholic University (PPCU) is also gratefully acknowledged, thanks to Prof. Péter Szolgay who provided me the opportunity to study here.
I thank the reviewers of my thesis, for their work and valuable comments.
I thank my closest colleagues from the SZTAKI Machine Perception Research Laboratory for their advice: Csaba Benedek, Andrea Manno-Kovács, Levente Kovács, László Tizedes, András Majdik and Attila Börcs. Special thanks to Levente Kovács and László Tizedes, who supported me their valuable advice and joint work.
Special thanks to Kálmán Tornai who coordinated and supported me during practice leading at PPCU.
Thanks to all the colleagues at PPCU and SZTAKI.
For further financial supports, thanks to the National Research, Development and Innovation Fund, under grant number K-120233, KH-125681 and KH-126688, the European Union and the Hungarian Government from the project 'Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications'

DOI:10.15774/PPKE.ITK.2020.010

under grant number EFOP-3.6.2-16-2017-00013 and 3.6.3 VEKOP-16-2017-00002 and the ÚNKP-19-3 New National Excellence Program of the Ministry of Human Capacities and the ÚNKP-20-4 new national excellence program of the ministry for Innovation and Technology from the source of the national research, development and innovation fund.

Last but not the least, I would like to thank my family: my parents and to my brother for supporting me spiritually throughout my Ph.D years and my life in general.

DOI:10.15774/PPKE.ITK.2020.010

Abstract

Three novel approaches are proposed in this thesis connected to 3D point cloud processing, namely semantic scene segmentation, point cloud registration, and camera-Lidar extrinsic parameter calibration. The proposed results are based on the combination of 3D geometry processing and deep learning based algorithms. The proposed methods have been evaluated in large point cloud databases containing various complex urban traffic scenarios and we have compared the proposed approaches to state-of-the-art methods and the results show significant improvements against the existing novel approaches.

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

Contents

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

List of Figures

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

List of Tables

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

Chapter 1

Introduction

In recent decades, huge progress has been made in the field of sensors for environment perception and mapping, which greatly influences the remarkable scientific progress in the field of object detection and classification [4, 15, 23, 24, 25, 26], scene segmentation and understanding [2, 26, 27, 28, 29, 30]. Several results have been adapted in real-world applications such as *driving assistance systems* and *mobile robot perception and navigation*, however, each application has unique requirements regarding to safety level, quality, and processing time, furthermore they rely on different sensors which make challenging the general usability of the scientific results. Nowadays, state-of-the-art autonomous systems such as self-driving vehicles, unmanned aerial and ground vehicles (UAV, UGV) and humanoid robots rely on a wide range of sensors. *Optical cameras* are able to capture high resolution, feature-rich data, *time-of-flight sensors* scan mostly interior areas in 3D, *radars* measure positions and velocities and *Lidars* are used for accurate 3D environment mapping. To utilize the advantages of each sensor the focus of scientific and industrial progress has been shifted towards robust sensor fusion.

Though the breakthrough of deep learning methods such as [25, 26, 27, 31] significantly improved the accuracy and generality of computer vision and scene understanding applications, we are still far from human-level performance. To train these models, a huge amount of precisely labeled data is required, furthermore in challenging situations such as *heavy traffic scenarios in dense urban areas*, *complex unknown regions like off-road scenes*, and *bad weather and illumination conditions* they still often underperform. Considering various sensor data can

DOI:10.15774/PPKE.ITK.2020.010

improve the accuracy of the perception and relying on multiple sensors can lead to more robust algorithms which are less influenced by environmental effects such as *heavy rain* and *illumination changes*.

Using detailed background information such as *High Definition (HD) maps* and *Geographic Information Systems (GIS)* [32, 33] to achieve accurate scene understanding and reliable localization is also an important research direction. Data fusion between on-board sensor data and detailed maps stored in offline databases is used in several industrial projects such as *Waymo and Uber self-driving vehicles* [34]. By the wide-spread emergence of smart cities with detailed map information, autonomous vehicles can utilize the static background information for navigation, localization, and contextual based scene analysis by registering their real-time captured on-board sensors data to the corresponding part of the detailed map.

Lidar technology has improved significantly in terms of resolution, scanning speed, and accuracy, furthermore the size and the cost of Lidar sensors have been decreasing steadily. Contrary to optical cameras, Lidars are not affected by the lighting conditions and illumination changes, they can operate more robustly under various weather conditions and they are able to obtain accurate, real 3D geometric information from the scene, so they are becoming one of the main cutting-edge technology in environment perception. Lidar sensors emit laser beams to capture the 3D geometric information of the scene and measure the echo time of the reflected beams from arbitrary object surfaces. Knowing the speed of light propagation (c) and the measured echo time (t) of the emitted beam the sensor is able to calculate the distance (s) of the given target: $s = (c \times t)/2$. In general, we can group laser scanning into three main categories: *hydro-graphic*, *aerial* and *terrestrial* laser scanning. Hydro-graphic Lidars are able to measure the depth of the seabed and they provide an accurate surface model while airborne Lidars are able to scan large areas for applications such as *urban planning* and *vegetation survey*. Terrestrial laser scanning (TLS) can be divided into *offline* and *real-time* data acquisition. Terrestrial mapping systems such as *Riegl VMX-450* and *FARO Focus* are used in various applications such as *road management and traffic control*, *city mapping*, and *cultural heritage conservation*. They are able to obtain very dense, feature-rich 3D data with precisely registered color

DOI:10.15774/PPKE.ITK.2020.010

information due to integrated cameras. While real-time sensors such as *Velodyne Lidars* are designed to navigate autonomous vehicles, so they provide relatively sparse, but also very accurate 3D measurements with a high-frequency update.

This thesis covers three different challenging research problems containing new scientific results related to point cloud segmentation, Lidar-based localization of self-driving vehicles, and camera-Lidar calibration. For data acquisition, two different types of Lidar scanner and a camera platform were used. In the following, I give a brief description of the investigated research problems which will be explained in more detail in the following chapters of the thesis. Table 1.1 gives a brief outline about the used datasets, sensors and references connected to the proposed topics.

- **Topic 1 - Deep learning based semantic labeling of mobile laser scanning data focusing on motion artifacts:**

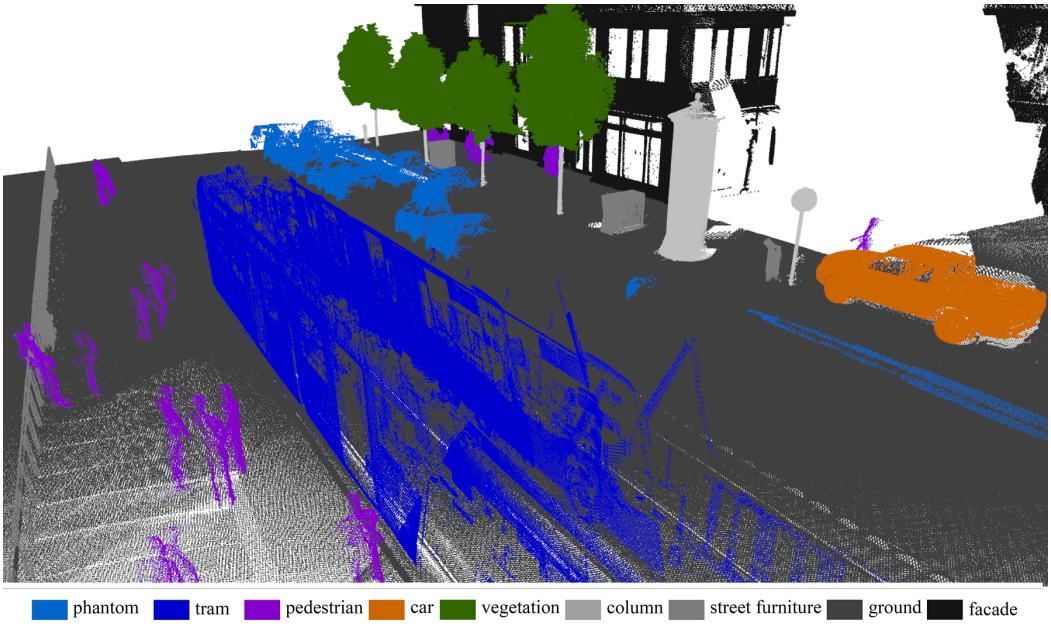


Figure 1.1: Semantic labeling results of the proposed method [2] (Topic 1).

Mobile mapping systems (MMS) are able to obtain very accurate and dense 3D point clouds from large areas for applications such as *urban planning* and *road management*. Though several point cloud segmentation models [26, 27,

DOI:10.15774/PPKE.ITK.2020.010

28, 29, 30] can be found in the literature, the high variety of existing laser scanners (obtaining point cloud data with very different characteristics) requires to adapt the models to the given dataset for accurate semantic labeling. In this thesis, the input data was provided by a *Riegl VMX-450* mobile mapping system, which contains two *Riegl VQ-450* laser scanners, a well design calibrated camera system, and an IMU/GNSS unit. The scanner registers the obtained frames into a global coordinate system and for each 3D points, it assigns RGB color values based on the calibrated cameras.

In the proposed approach for semantic labeling of dense point clouds, we have considered the characteristic of the data and we have proposed a two-channel 3D convolutional neural network (CNN) which is able to efficiently label the scene into different categories focusing on challenging scenarios such as elongated noise regions, artifacts (defined as *Phantom objects*) caused by the concurrent movement of the dynamic objects with the scanning platform such as *vehicles* and *pedestrians*. We have also tested our proposed method on publicly available datasets [35, 36, 37, 38] to show the robustness of the method and we have compared the model against different state-of-the-art methods.

- **Topic 2 - Robust, real-time registration between different type of point clouds and automatic localization approach for autonomous vehicles:**

  3D point cloud registration is a well-studied computer vision problem, so several solutions exist in the literature. Though some state-of-the-art methods are able to achieve very high accuracy, they usually assume that the characteristics of the source and target point cloud are the same or very similar, and in most cases, they rely on some iterative methods such as Iterative Closest Point ICP [39] which requires a suitable initial starting point for the registration process. We have proposed an object-based coarse alignment method for point cloud registration which can be refined with a point level registration step. Our method is able to robustly register point clouds with very different characteristics and it is able to handle large initial miss-alignment.

DOI:10.15774/PPKE.ITK.2020.010

Figure 1.2: Projection results of the proposed target-less, automatic camera-Lidar calibration method [1].

GPS based localization accuracy mostly in dense urban environment fluctuates between a wide range which makes it unreliable and according to our experiments in the streets of Budapest, Hungary, GPS position error can be larger than 10 meters. We have extended our point cloud registration method with a robust key-point extraction algorithm and we have proposed a localization approach that is able to precisely register a real-time captured Lidar point cloud obtained by an autonomous vehicle to a static dense background map.

- **Topic 3 - On-the-fly, automatic camera and Lidar extrinsic parameter calibration:**

  The main goal of the proposed method is to automatically calibrate a camera and a Lidar sensor relative to each other mounted onto the top of a moving vehicle. The proposed method works in an end-to-end manner without assuming any user interaction and it does not require any specific calibration objects such as 3D boxes and chessboard patterns. The main advantage of the method is to be able to periodically recalibrate the sensors on-the-fly i.e. without stopping the vehicle. We have re-defined the camera-Lidar calibration problem as a 3D point cloud registration task by generating 3D point clouds from the incoming consecutive camera frames

DOI:10.15774/PPKE.ITK.2020.010

using a Structure from Motion (SfM) method. We have proposed a two-stage registration method which is able to accurately register the point clouds. First, we transform the Lidar point cloud to the coordinate system of the generated SfM point cloud using an object-based alignment, and finally, we have proposed a control curve-based refinement method to handle the deformation problem of point clouds occurring during the SfM processing method. As a result of the registration, we are able to determine an accurate extrinsic parameter calibration (see Fig. 1) between the camera and the Lidar sensors.

1.1 Brief summary of point cloud segmentation datasets

Static point cloud datasets for 3D semantic segmentation is introduced in Chapter 2 related to Topic 1. Table 1.2 gives a short description of the main parameters of the databases and in Chapter 2 we analyze the datasets in more detail. This thesis focus on static point cloud segmentation, however we note that several dataset for semantic segmentation of sequential Lidar point cloud data can be found in the literature. In case of segmentation of dense static point cloud data, approaches have to process huge databases with millions of points contrary segmentation of sparse sequential point cloud data where the focus is on real-time data processing.

1.2 Outline of the thesis

In the following, we present the outline of the thesis. In Chapter 2 we propose a novel deep learning based approach for semantic labeling of dense point clouds collected in urban environment into different classes such as *ground*, *facade*, *vegetation*, *pedestrians*, *parking vehicles*, *moving dynamic objects called Phantoms here*, etc. We propose a 3D convolutional neural network approach to predict the class of the given samples trained in an end-to-end manner.

Chapter 3 introduces a novel object-level registration method based on a fingerprint minutiae matching [40] algorithm which is able to accurately align point clouds obtained by different laser scanners to a common coordinate system. A

DOI:10.15774/PPKE.ITK.2020.010

Table 1.1: Summary of datasets, sensors and references connected to the thesis.

	Topic 1	Topic 2	Topic 3
Datasets	SZTAKI CityMLS [2] Paris-rue-Madame [35] Terramobilita [36] Semantic3D.net [37] Oakland [38] Toronto-3D [102] Paris-Lille-3D [103]	SZTAKI CityMLS [2] own Velodyne HDL-64E Lidar data	own Velodyne HDL-64E Lidar data own camera stream
Sensors	Riegl VMX-450	Riegl VMX-450 Velodyne HDL-64E	Velodyne HDL-64E Point Grey optical cameras
References	OG-CNN [29] Multi-view [51] PointNet++ [26] SPLATNet^{xyz} [30] SPLATNet$^{xyz}_{rgb}$ [30] Markov [38]	Crossmodal [12]	Target-based [70] Bearing angle [72] Line based [73]

Table 1.2: Dataset comparison of dense semantic point cloud segmentation.

Dataset	Annotation	Classes	#points	Fields	Sensor	Year	Organization
Oakland3D	point-wise	5	1.6M	x, y, z, label	SICK LMS (MLS)	2009	Carnegie Mellon University
Paris-rue-Madame		17	20M	x, y, z, intensity, label	LARA2-3D (MLS)	2014	MINES ParisTech
TerraMobilita		15	12M	x, y, z, intensity, label	Riegl LMS-Q120i (MLS)	2015	University of Paris-Est
Paris-lille-3D		50	143M	x, y, z, intensity, label	Velodyne HDL-32E (MLS)	2018	MINES ParisTech
Toronto3D		8	78.3M	x, y, z, r, g, b, intensity, label	Teledyne Optech Maverick (MLS)	2020	University of Waterloo
Semantic3D		8	4009M	x, y, z, r, g, b, intensity, label	unknown (TLS)	2017	ETH Zurich
SztakiCityMLS		9	327M	x, y, z, r, g, b, label	Riegl VMX-450 (MLS)	2019	SZTAKI

novel localization approach is also proposed in Chapter 3 which is able to robustly determine the position and the orientation of an autonomous vehicle by registering its on-board Lidar data to a dense point cloud.

We present a novel, on-the-fly camera-Lidar extrinsic parameter calibration in Chapter 4. The proposed method works automatically without any user interactions and it does not require any target objects it relies on only the captured camera images and the obtained Lidar point cloud.

At the end of the thesis, a summary and conclusion can be found. Appendix A gives a brief insight into the mathematical and technical details of the used algorithms and concepts, while Appendix B summarizes the used abbreviations.

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

Chapter 2

Deep learning based semantic labeling of mobile laser scanning data

This chapter introduces a novel 3D convolutional neural network (CNN) based method to segment point clouds obtained by mobile laser scanning (MLS) sensors into nine different semantic classes, which can be used for high definition city map generation. The main purpose of semantic point labeling is to provide a detailed and reliable background map for self-driving vehicles (SDV), which indicates the roads and various landmark objects for navigation and decision support of SDVs. The proposed approach considers several practical aspects of raw MLS sensor data processing, including the presence of diverse urban objects, varying point density, and strong measurement noise of phantom effects cased by objects moving concurrently with the scanning platform. A new manually annotated MLS benchmark set called `SZTAKI CityMLS` is also introduced in this chapter, which is used to evaluate the proposed approach, and to compare our solution to various reference techniques proposed for semantic point cloud segmentation.

DOI:10.15774/PPKE.ITK.2020.010

2.1 Introduction

Self-localization and scene understanding are key issues for self-driving vehicles (SDVs), especially in dense urban environments. Although the GPS-based position information is usually suitable for helping human drivers, its accuracy is not sufficient for navigating a SDV. Instead, the accurate position and orientation of the SDV should be calculated by registering the measurements of its on-board visual or range sensors to available 3D city maps [10].

Mobile laser scanning (MLS) platforms equipped with time synchronized Lidar sensors and navigation units can rapidly provide very dense and feature rich point clouds from large environments (see Fig. 2.1), where the 3D spatial measurements are accurately registered to a geo-referenced global coordinate system [41, 42, 43]. In the near future, these point clouds may act as a basis for detailed and up-to-date 3D High Definition (HD) maps of the cities, which can be utilized by self driving vehicles for navigation, or by city authorities for road network management and surveillance, architecture or urban planning. However, all of these applications require semantic labeling of the data (Fig. 1). While the high speed of point cloud acquisition is a clear advantage of MLS, due to the huge data size yielded by each daily mission, applying efficient automated data filtering and interpretation algorithms in the processing side is crucially needed, which steps still introduce a number of key challenges.

2.1.1 Problem statement

Taking the raw MLS measurements, one of the critical issues is the *phantom* effect caused by independent object motions (Fig. 2.1.1). Due to the sequential nature of the environment scanning process, scene objects moving concurrently with the MLS platform (such as passing vehicles and walking pedestrians) appear as phantom-like long-drawn, distorted structures in the resulting point clouds [11]. It is also necessary to recognize and mark all transient scene elements such as pedestrians, parking vehicles [42] or trams from the MLS scene. On one hand, they are not part of the reference background model, thus these regions must be eliminated from the HD maps. On the other hand, the presence of these objects may indicate locations of sidewalks, parking places etc. Column-shaped

DOI:10.15774/PPKE.ITK.2020.010

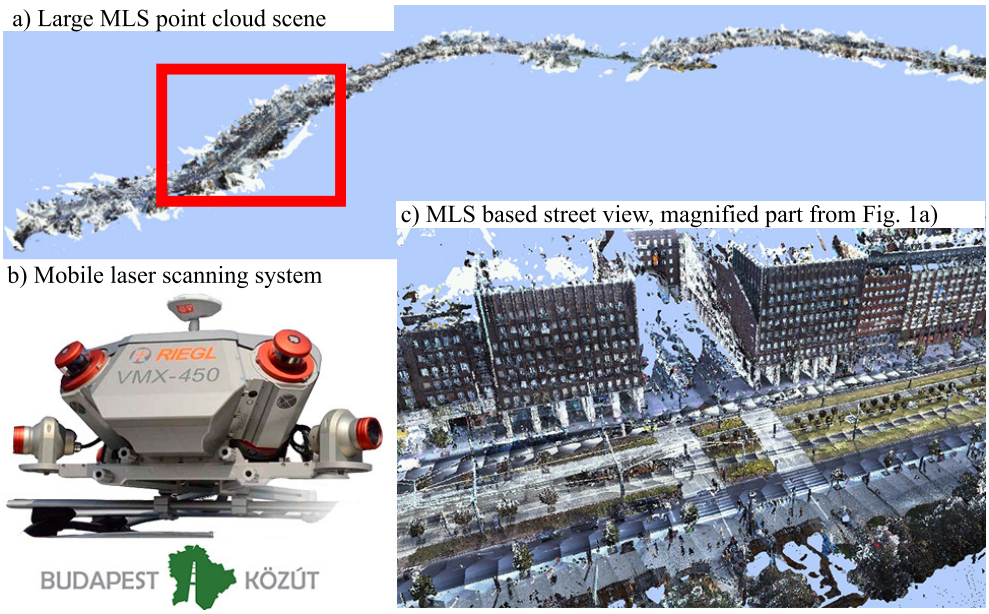


Figure 2.1: MLS sensor and a scanned road segment.

objects, such as poles, traffic sign bars [41], tree trunks are usually good landmark points for navigation. Finally, vegetation areas (bushes, tree foliage) should also be specifically labeled [43]: since they are dynamically changing over the whole year, object level change detection algorithms should not take them into account.

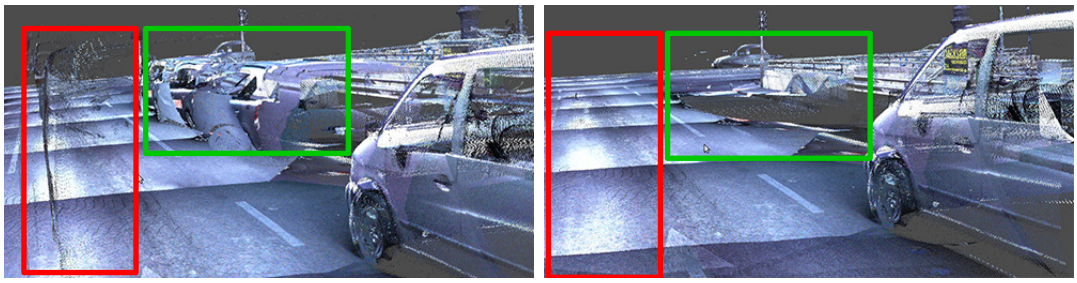
2.1.2 Sensors discussed in this chapter

In this chapter, we utilize the measurements of the Riegl VMX450 MLS system. The Riegl VMX450 MLS system is highly appropriate for city mapping, urban planning and road surveillance applications. It integrates two Riegl laser scanners, a well designed calibrated camera platform, and a high performance Global Navigation Satellite System (GNSS). It provides extremely dense, accurate (up to global accuracy of a few centimeters) and feature rich data with relatively uniform point distribution.

2.1.3 Aim of the chapter

To address the above complex multi-class semantic labeling problem we introduce a new 3D convolutional neural network (CNN) based approach to segment the

DOI:10.15774/PPKE.ITK.2020.010



(a) Raw MLS data with phantoms. (b) Result of phantom removal workflow.

Figure 2.2: Demonstration of the phantom effect in MLS data and the result of phantom removal workflow with the proposed approach.

scene in voxel level, and for testing the approach, we present the `SZTAKI CityMLS` benchmark set containing different labeled scenes from dense urban environment. Differently from previously proposed general point cloud labeling frameworks [23, 26], the present approach is focusing on challenging issues of MLS data processing in self-driving applications.

2.2 Related work

3D semantic segmentation is one of the most researched fields in point cloud based scene understanding. Many of the proposed segmentation methods deal with the segmentation of real-time obtained point cloud sequences for self-driving applications and real-time SLAM, others focus on the segmentation of dense point clouds acquired by mobile mapping systems for urban planning and infrastructure monitoring. During dense point cloud segmentation managing huge amount of data is challenging, furthermore several motion artifacts and noise occur in the obtained point clouds while in segmentation of real-time point cloud streams online data processing and the sparsity of the data are the main challenges [105]. In this thesis we focus on dense point cloud segmentation.

While a number of various approaches have already been proposed for general point cloud scene classification, they are not focusing on all practical challenges of the above introduced worklfow of 3D map generation from raw MLS data. In particularly, only a few related works have discussed the problem of *phantom*

DOI:10.15774/PPKE.ITK.2020.010

removing. Point-level and statistical feature based methods such as [44] and [45] examine the local density of a point neighborhood, but as noted in [46] they do not take into account higher level structural information, limiting the detection rate of *phantoms*. The task is significantly facilitated if the scanning position (e.g. by tripod based scanning [47]) or a relative time stamp (e.g. using a rotating multi-beam Lidar [48]) can be assigned to the individual points or point cloud frames, which enables the exploitation of multi-temporal feature comparison. However, in case of our examined MLS point clouds no such information is available, and all points are represented in the same global coordinate system.

Several techniques extract various object blob candidates by geometric scene segmentation [41, 4], then the blobs are classified using shape descriptors, or deep neural networks [4]. Although this process can be notably fast, the main bottleneck of the approach is that it largely depends on the quality of the object detection step.

Alternative methods implement a voxel level segmentation of the scene, where a regular 3D voxel grid is fit to the point cloud, and the voxels are classified into various semantic categories such as roads, vehicles, pole-like objects, etc. [43, 24, 29]. Here a critical issue is feature selection for classification, which has a wide bibliography. Handcrafted features are efficiently applied by a maximum-margin learning approach for indoor object recognition in [49]. Covariance, point density and structural appearance information is adopted in [50] by a random forest classifier to segment MLS data with varying density. However, as the number and complexity of the recognizable classes increase, finding the best feature set by hand induces challenges.

3D CNN based techniques have been widely used for point cloud scene classification in the recent years, following either *global* or *local* (window based) approaches. *Global* approaches consider information from the complete 3D scene for classification of the individual voxels, thus the main challenge is to keep the time and memory requirements tractable in large scenes. The *OctNet* method implements a new complex data structure for efficient 3D scene representation which enables the utilization of deep and high resolution 3D convolutional networks [23]. From a practical point of view, by OctNet's training data annotation opera-

DOI:10.15774/PPKE.ITK.2020.010

tors should fully label complete point cloud scenes, which might be an expensive process.

Sliding window based techniques are usually computationally cheaper, as they move a 3D box over the scene, using locally available information for the classification of each point cloud segment. The *Vote3Deep* [24] assumes a fixed-size object bounding box for each class to be recognized, which might be less efficient if the possible size range of certain objects is wide. A CNN based voxel classification method has recently been proposed in [29], which uses purely local features, coded in a 3D occupancy grid as the input of the network. Nevertheless, they did not demonstrate the performance in the presence of strong *phantom* effects, which require accurate local density modeling [45, 46].

[99] represents 3D objects as spherical projections around their barycenter and a neural network was trained to classify the spherical projections. Two complementary projections namely a depth variations projection of the 3D objects and a contour-information projection from different angles were introduced. A multi-view fusion network was introduced in [100] to learn to get a global feature descriptor by fusing the features from all. The model consists of three key parts: the feature extraction structure to extract the features of point clouds, the view fusion network to merge features of 2.5D point clouds from all views into a global feature descriptor, and a classifier composed of fully connected layers to perform classifications. Another multi-view technique [51] projects the point cloud from several (twelve) different viewpoints to 2D planes, and trains 2D CNN models for the classification. Finally, the obtained labels are backprojected to the 3D point cloud. These approaches presents high quality results on synthetic datasets and in point clouds from factory environments, where due to careful scanning complete 3D point cloud models of the scene objects are available. Application to MLS data containing partially scanned objects is also possible, but the advantages over competing approaches are reduced here [51].

Methods such as VolMap [106], LiSeg [108] and PointSeg [109] focus on real-time point cloud segmentation reducing the 3D parameter space to a 2D surface. VolMap [106] is a modified version of U-Net [110] taking a Bird-eye view image created from the 3D point cloud. Both LiSeg [108] and PointSeg [109] create a range view image from the point cloud and they use dilated convolution to

DOI:10.15774/PPKE.ITK.2020.010

extract relevant features for training. While LiSeg proposes a fully convolution network approach to segment the range image PointSeg is based on SqueezeNet [107], which introduces squeezing re-weighting layers.

PointNet++ [26] introduces a hierarchical neural network for point set classification. The method takes random samples within a given radius of the examined point, so it does not exploits density features. Results are demonstrated on synthetic and indoor data samples, with dense and accurate spatial data and RGB color information. PointConv [101] is a convolution based extension of the *PointNet++*, which allow to dramatically scale up the network and significantly improve its performance.

The *Similarity Group Proposal Network* (SGPN) [28] uses *PointNet++* as a backbone feature extractor, and presents performance improvement by adding several extra layers to the top of the network structure. However as noted by the authors, SGPN cannot process large scenes on the order 10^5 or more points [28], due to using a similarity matrix whose size scales quadratically as the number of points increases. This property is disadvantageous for MLS data processing, where a typical scene may contain over 10^7 points.

The *Sparse Lattice Network* (SPLATNet_{3D}) [30] is a recent technique which able to deal with large point cloud scenes efficiently by using a Bilateral Convolution Layer (BCL). SPLATNet_{3D} [30] projects the extracted features to a lattice structure, and it applies sparse convolution operations. Similarly to voxel based approaches, the lattice structure implements a discrete scene representation, where one should address under- and over-segmentation problems depending on the lattice scales.

2.3 Benchmark issues

A number of benchmark sets have already been published for 3D point cloud segmentation in urban environment, including MLS datasets Oakland [38] (1.6M points), Paris-rue-Madame (20M points) [35] and data from the IQmulus & TerraMobilita Contest (12M labeled points) [36]. However, their available annotated segments are relatively small, which make the development of supervised classification algorithms less relevant due to over fitting problems.

DOI:10.15774/PPKE.ITK.2020.010

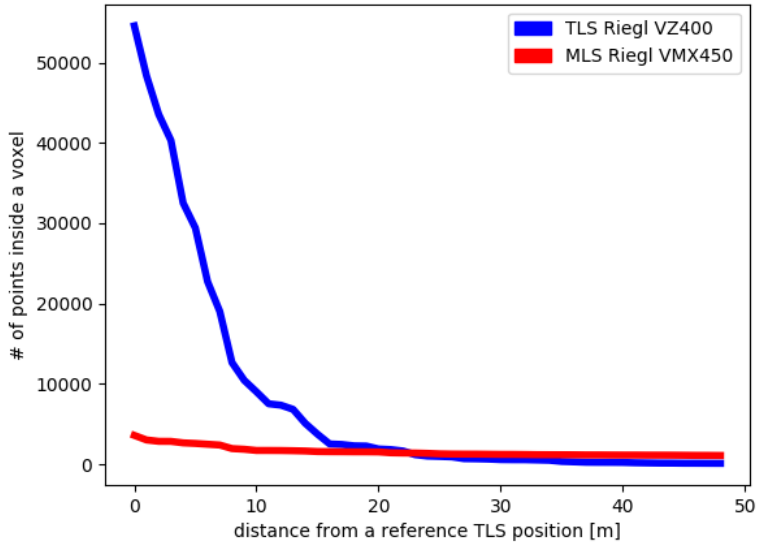


Figure 2.3: Point cloud characteristics comparison of measurements from the same region obtained by a static Riegl VZ-400 TLS sensor and a moving Riegl VMX-450 MLS system, respectively. Point density is displayed as a function of the distance from the TLS sensor's center position.

The Semantic3D.net benchmark [37] contains a considerable larger set of labeled data, however it has been created with static terrestrial laser scanners (TLS) which produce more accurate and in certain regions significantly denser point clouds than MLS. As shown in Fig. 2.3 the point density of a single TLS sensor is steeply decreasing as a function of the distance from the sensor, while applying mobile scanning, we can obtain a more uniform, but generally lower point density in the same region. In addition, the density characteristic of a large point cloud segment obtained by TLS from multiple scanning positions is strongly varying, since TLS operators may follow arbitrary trajectories and timing constraints during the scanning mission. Therefore, comparing two different TLS datasets may show significant differences, even if they have been recorded by the same scanner, but for different purpose or by different operators. As a consequence, developing widely usable object detection methods for large-scale TLS datasets needs careful practical considerations.

DOI:10.15774/PPKE.ITK.2020.010

On the other hand, MLS scene segmentation is today a highly relevant field of research, with strong industrial interest. In MLS data recording, the car passes with a normal 30-50 km traveling speed, following a more predictable trajectory (usually scans are preformed in both directions for a two-way road), therefore the effects of the driving dynamics on the obtained point cloud can be indirectly incorporated into the learning process. However, compared to TLS data, ghost filtering is more difficult, and the measurement noise is higher.

In this chapter, we utilize MLS data captured by a Riegl VMX-450 for real industrial usage by the Road Management Department of the Budapest City Council. Our new `SZTAKI CityMLS` dataset contains in total around 327 Million annotated points from various urban scenes, including main roads with both heavy and solid traffic, public squares, parks, and sidewalk regions, various types of cars, trams and buses, several pedestrians and diverse vegetation.

2.3.1 Data characteristic of MLS benchmarks

As shown in Fig. 2.4 the data characteristic of `SZTAKI CityMLS` is significantly different from TerraMobilita and Paris-rue-Madam data, making the proposal of the new benchmark indeed relevant. While Paris-rue-Madame database contains the most dense point clouds, due to registration issues of the recorded Rotating Multi-beam Lidar (Velodyne) frames, the obtained point cloud is quite noisy. On the other hand, the TerraMobilita database was captured with multiple 2D laser scanners yielding accurate spatial point cloud coordinates, but the measurements are sparse: depending on the speed of the scanning platform smaller objects may be composed of a few line segments only. As for `SZTAKI CityMLS`, the Riegl VMX-450 scans are well suited to industrial applications requiring dense, accurate and notable homogeneous point clouds.

DOI:10.15774/PPKE.ITK.2020.010

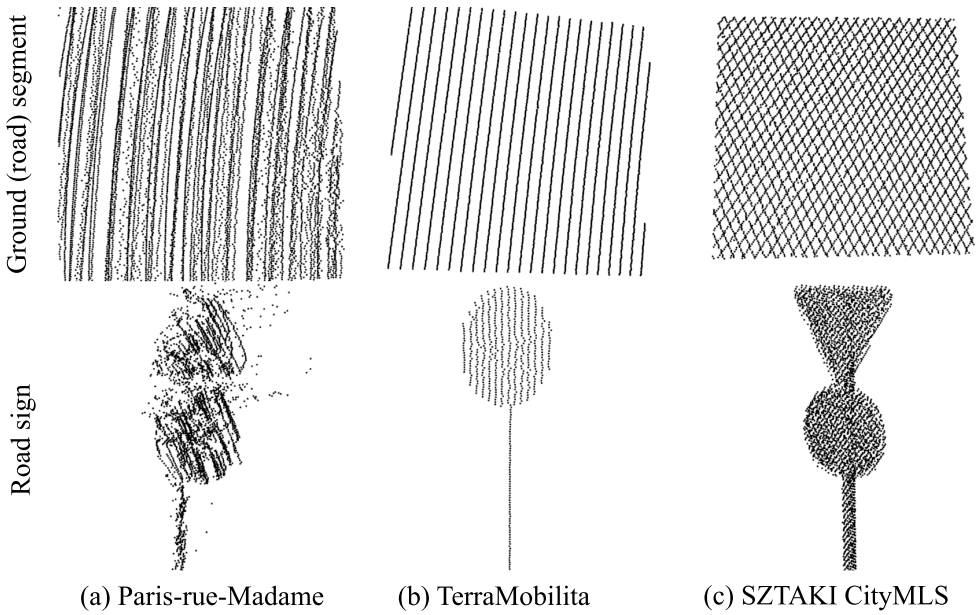
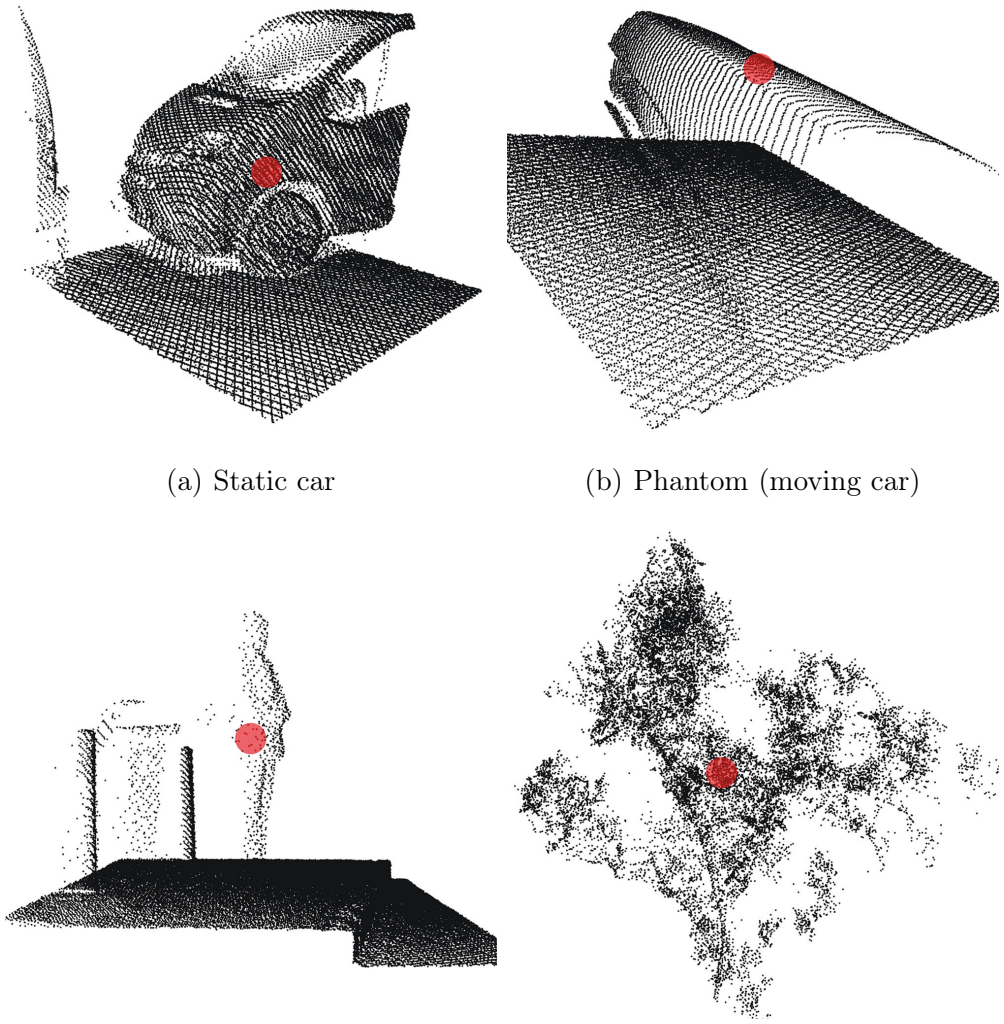


Figure 2.4: Data quality comparison between two reference datasets and the proposed `SZTAKI CityMLS` dataset.

2.4 Proposed approach

In this section, we propose a new 3D CNN based semantic point cloud segmentation approach, which is adopted to dense MLS point clouds of large scale urban environments, assuming the presence of high variety of objects, with strong and diverse *phantom* effects. The present technique is based on our earlier model [11] specifically developed for phantom detection and removal, which we extend for recognizing nine different semantic classes required for 3D map generation: *phantom*, *tram/bus*, *pedestrian*, *car*, *vegetation*, *column*, *street furniture*, *ground* and *facade*. As main methodological differences from [11], our present network uses a two channel data input derived from the raw MLS point cloud featuring local point density and elevation, and a voxel based space representation, which can handle the separation of tree crowns or other hanging structures from ground objects more efficiently than the pillar based model of [11]. To keep the computational requirements low, we implemented a sparse voxel structure avoiding unnecessary operations on empty space segments.

DOI:10.15774/PPKE.ITK.2020.010



(a) Static car (b) Phantom (moving car)

(c) Pedestrian (d) Vegetation

Figure 2.5: Different training volumes extracted from point cloud data. Each training sample consists of $K \times K \times K$ voxels (used $K = 23$), and they are labeled according to their central voxel (highlighted with red).

2.4.1 Data model for training and recognition

Data processing starts with building our sparse voxel structure for the input point cloud, with a fine resolution (used $\lambda = 0.1m$ voxel side length). During classification we will assign to each voxel a unique class label from our nine-

DOI:10.15774/PPKE.ITK.2020.010

element label set, based on majority votes of the points within the voxel.

Next we assign two feature channels to the voxels based on the input point cloud: point *density*, taken as the number of included points, and *mean elevation*, calculated as the average of the point height values in the voxel.

The unit of training and recognition in our network is a $K \times K \times K$ voxel neighborhood (used $K = 23$), called hereafter training volume. To classify each voxel v, we consider the point density and elevation features in all voxels in the v-centered training volume, thus a given voxel is labeled based on a 2-channel 3D array derived from K^3 local voxels. The proposed 3D CNN model classifies the different training volumes independently. This fact specifies the roles of the two feature channels: while the *density* feature contributes to model the local point distribution within each semantic class, the *elevation channel* informs us about the expected (vertical) locations of the samples regarding the different categories, providing impression from the global position of the data segment within the large 3D scene. The elevation of a given sample is determined by subtracting the actual ground height elevation from the geo-referenced height of the sample.

Fig. 2.5 demonstrates various training volumes, used for labeling the central voxel highlighted with red color. As we consider relatively large voxel neighborhoods with $K \cdot \lambda$ (here: $2.3m$) side length, the training volumes often contain different segments of various types of objects: for example, Fig. 2.5(b) contains both phantom and ground regions, while Fig. 2.5(c) contains column, ground and pedestrian regions. These variations add supplementary contextual information to the training phase beyond the available density and elevation channels, making the trained models stronger.

Fig. 2.6 represents the voxelized training samples. Based on our experiments *phantom* objects show more sparse data characteristic than other static background objects. The ground region under the *phantom* objects (Fig. 2.6 (b)) is more dense than the *phantom*, furthermore it can be seen a parking vehicle (Fig. 2.6 (a)) which also shows more dense data characteristic.

DOI:10.15774/PPKE.ITK.2020.010

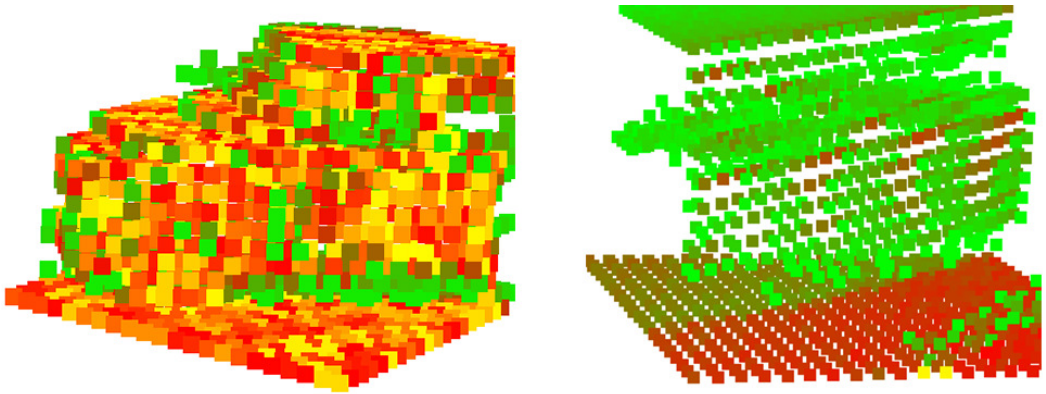


Figure 2.6: Voxelized training samples. The red voxels cover dense point cloud segments, while green voxels contain fewer points.

2.4.2 3D CNN architecture and its utilization

The proposed 3D CNN network implements an end-to-end pipeline: the feature extractor part (combination of several 3D convolution, max-pooling and dropout layers) optimizes the feature selection, while the second part (fully connected dense layers) learns the different class models. Since the size of the training data ($23 \times 23 \times 23$) and the number of classes (9) are quite small, we construct a network with a similar structure to the well known LeNet-5 [52], with adding an extra convolution layer and two new dropout layers to the LeNet-5 structure, and exchanging the 2D processing units to the corresponding 3D layers. Fig. 2.7 demonstrates the architecture and the parameters of the trained network. Each convolution layer uses $3 \times 3 \times 3$ convolution kernels and a Rectified Linear Unit (ReLu) activation function, while the numbers of filters are 8, 16 and 32 in the 1st, 2nd and 3rd convolution layer, respectively. The output layer is activated with a Softmax function. To avoid over-fitting, we use dropout regularization technique, randomly removing 30% of the connections in the network. Moreover to make our trained object concepts more general, we clone and randomly rotate the training samples around their vertical axis several times. The network is trained with Stochastic Gradient Descent (SGD) algorithm, and we change the learning rate in the training epochs as a function of the validation accuracy change.

DOI:10.15774/PPKE.ITK.2020.010

For more detailed information, we present some mathematical background and definition in Appendix A about the fundamentals of convolutional neural networks.

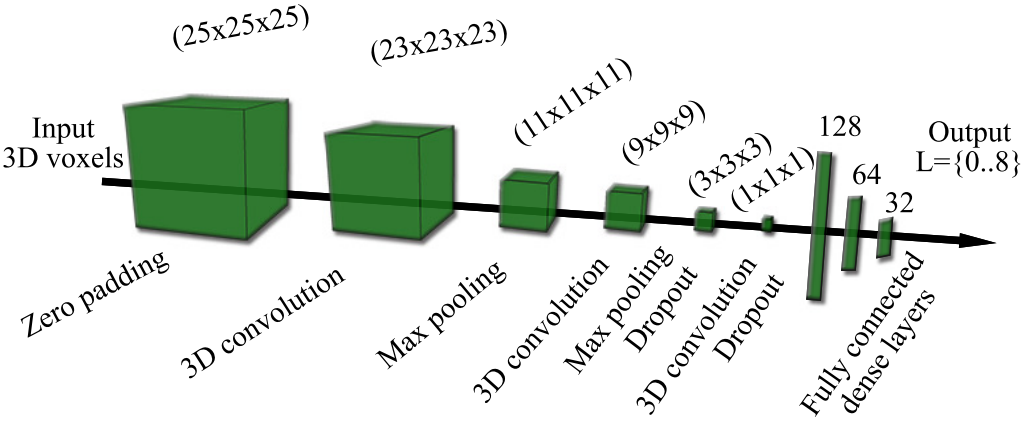


Figure 2.7: Structure of the proposed 3D convolutional neural network, containing three 3D convolution layers, two max-pooling and two dropout layers. The input of the network is a $K \times K \times K$ voxel (used $K = 23$) data cube with two channels, featuring density and point altitude information. The output of the network is an integer value from the set $L = 0..8$.

To segment a scene, we move a sliding volume across the voxelized input point cloud, and capture the $K \times K \times K$ neighborhood around each voxel. Each neighborhood volume is separately taken as input by the CNN classifier, which predicts a label for the central voxel only. As the voxel volumes around the neighboring voxels strongly overlap, the resulting 3D label map is usually smooth, making possible object or object group extraction with conventional region growing algorithms (see Fig. 1, 2.8).

2.5 Experimental Results and Evaluation

2.5.1 Point cloud annotation and training

Large-scale MLS scene annotation is a crucial step in deep learning based approaches. For this reason, we developed a user friendly 3D point cloud annotator tool, that allows operators to label arbitrary shaped 3D volumes quickly. We

DOI:10.15774/PPKE.ITK.2020.010

assigned unique labels to *occupied* voxels of the scene, using 10cm voxels which determines the spatial accuracy of the annotation.

In one step, the operator can mark a rectangle area on the screen, which defines with the actual viewpoint a 3D pyramid volume in scene's 3D coordinate system. Then, the annotated volume can be created through a combination of union and intersection operations on several pyramids.

With this tool we manually labeled around 327M points over a 30.000 m^2 area of the city, with more than 50m elevation differences, using the earlier defined nine classes. As a result of annotation, we created a new benchmark set called `SZTAKI CityMLS`[1].

Next, we divided our data into three non-overlapping segments used for training, validation and test, respectively. For training data generation, we randomly selected 100.000 voxels from each class's representative region in the *training segment* of the labeled data, and extracted the 2-channel $K \times K \times K$ voxel volumes around each training sample, which were used as the local fingerprints of the corresponding point cloud parts. This selection yielded in total 900.000 volumes, used for training the network. During the training process, we tuned the parameters of the classifier on a validation set, which contains 20.000 samples from each class, selected from the *validation segment* of the point cloud.

The quantitative performance evaluation of the network is performed on an independent test set (without any overlap with the training and the validation sets), including two million voxel volumes extracted from the *test segment* of the point cloud, representing the classes evenly.

2.5.2 Hyperparameter tuning

Voxel size λ and dimension of the data sample cube (K) are two important hyperparameters of the proposed model, which have to be carefully tuned with respect to the data density and the recognizable classes. We have optimized these parameters with a *grid search* algorithm, which yielded an optimum of $\lambda = 0.1$m and $K = 23$, regarding our Riegl VMX-450 data, as mentioned in Sec. 2.4.1. For

[1]The `SZTAKI CityMLS` dataset is available at the following url: `http://mplab.sztaki.hu/geocomp/SZTAKI-CityMLS-DB.html`

DOI:10.15774/PPKE.ITK.2020.010

Table 2.1: Performance analysis of the proposed C^{2}CNN method as a function of the voxel size parameter.

Parameters	Voxel size λ [m], using a fixed $K = 23$ kernel size						
	0.02	0.05	0.1	0.2	0.3	0.4	0.5
# of voxels	812500000	52000000	6500000	812500	240596	101563	52000
Precision	34.7	77.8	90.4	83.6	76.3	64.2	44.7
Recall	29.8	69.7	90.2	85.9	77.8	61.7	48.5
F measure	32.1	73.5	90.3	84.7	77.0	62.9	46.5

Table 2.2: Performance analysis of the proposed C^{2}CNN method as a function of the data cube size $K \times K \times K$

Parameters	Data cube's side length (K), using a fixed 0.1 m voxel size										
	7	11	17	21	23	25	27	29	31	37	41
Precision	58.4	72.5	81.1	87.6	90.4	88.5	86.4	83.2	78.9	72.8	69.4
Recall	55.7	69.7	82.6	87.1	90.2	89.1	87.2	85.6	82.2	71.2	69.6
F measure	57.0	71.1	81.8	87.4	90.3	88.8	86.8	84.4	80.5	72.0	69.5

further analysis, Table 2.1 shows the model performance as a function of different λ voxel size settings, with a fixed $K = 23$ value. We can observe a maximal performance at $\lambda = 0.1$m. Using smaller voxels, the model tends to oversegment the scene, while adopting a too large voxel size, the CNN-based label prediction yields coarse region boundaries.

On the other hand, Table 2.2 demonstrates the dependence of the results on the data cube's side length (K), with choosing a constant voxel grid resolution of $\lambda = 0.1$m. Using significantly smaller kernels than the optimal $K = 23$, the model can only consider small local voxel neighborhoods, which do not enable efficient contextual modeling. However, in cases of too large kernels, the training/test samples may contain significant noise and irrelevant background segments, which fact often leads to overfitting problems.

DOI:10.15774/PPKE.ITK.2020.010

2.5.3 Evaluation and comparison to reference techniques

We evaluated our proposed method against four reference techniques in qualitative and quantitative ways on the `SZTAKI CityMLS` dataset.

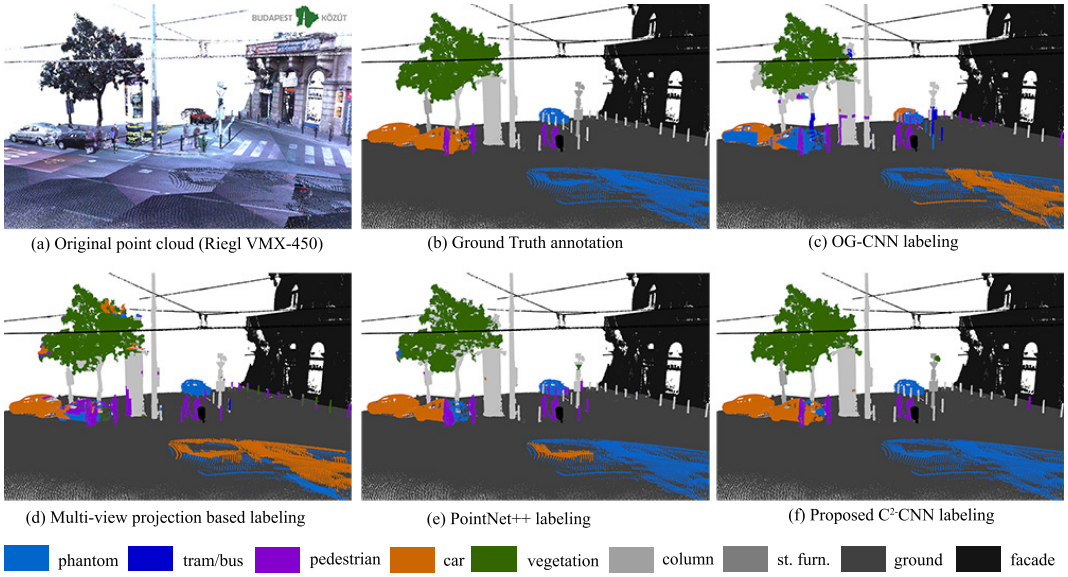


Figure 2.8: Qualitative comparison of the results provided by the three reference methods, (c) OG-CNN, (d) Multi-view approach and (e) PointNet++, and the proposed (f) C^2CNN approach in a sample scenario. For validation, Ground Truth labeling is also displayed in (b).

First, we tested a single channel 3D CNN [29], which uses a 3D voxel occupancy grid (OG) as input (OG-CNN). Second, we implemented a multi-view method based on [51], that projects the point cloud onto different planes, and achieves CNN classification in 2D. Third, we tested the PointNet++ [26] deep learning framework, using their publicly available source code. Finally we adopted the implementation of SPLATNet3D [30], by applying two different feature selection strategies.

Fig. 2.8 shows a sample scene for qualitative comparison of the manually edited Ground Truth, the outputs of the OG-CNN, multi-view and PointNet++ methods, and the result of the proposed two channel C^2CNN technique. We also evaluated the proposed and the reference methods in a quantitative way. Table

DOI:10.15774/PPKE.ITK.2020.010

Table 2.3: Quantitative evaluation of the proposed C^{2}CNN approach and the reference techniques on the new `SZTAKI CityMLS` dataset.

Class	OG-CNN [29]			Multi-view [51]			PointNet++ [26]			SPLATNet^{xyz} [30]			SPLATNet$^{xyz}_{rgb}$ [30]			Proposed C^{2}CNN		
	Pr	Rc	F-r	Pr	Rc	F-r	Pr	Rc	F-r	Pr	Rc	F-r	Pr	Rc	F-r	Pr	Rc	F-r
Phantom	85.3	34.7	49.3	76.5	45.3	56.9	82.3	76.5	79.3	82.5	80.9	81.7	83.4	78.2	80.7	84.3	85.9	**85.1**
Pedestrian	61.2	82.4	70.2	57.2	66.8	61.6	86.1	81.2	83.6	82.6	82.1	82.3	80.4	78.6	79.5	85.2	85.3	**85.2**
Car	56.4	89.5	69.2	60.2	73.3	66.1	80.6	92.7	86.2	81.5	90.0	85.5	81.1	89.4	85.0	86.4	88.7	**87.5**
Vegetation	72.4	83.4	77.5	71.7	78.4	74.9	91.4	89.7	90.5	87.1	88.2	87.6	86.4	87.3	86.8	98.2	95.5	**96.8**
Column	88.6	74.3	80.8	83.4	76.8	80.0	83.4	93.6	**88.2**	84.3	90.2	87.2	84.1	89.2	86.6	86.5	89.2	87.8
Tram/Bus	91.4	81.6	86.2	85.7	83.2	84.4	83.1	89.7	86.3	82.1	83.5	82.8	79.3	82.1	80.7	89.5	96.9	**93.0**
Furniture	72.1	82.4	76.9	57.2	89.3	69.7	84.8	82.9	83.8	84.7	86.2	**85.4**	82.6	81.3	81.9	88.8	78.8	83.5
Overall	76.9	74.2	75.5	72.5	73.4	72.9	85.6	87.5	86.5	83.5	85.9	84.7	82.5	83.7	83.0	90.4	90.2	**90.3**

Note: Voxel level Precision (Pr), Recall (Rc) and F-rates (F-r) are given in percent (overall values weighted with class significance).

2.3 shows the voxel level precision (Pr), recall (Rc) and F-rates (F-r) for each class separately as well as the overall performance weighted with the occurrence of the different classes. Note that Table 2.3 does not contain the values obtained regarding *facades* and *ground*, which classes proved to be quite easy to recognize for the CNN network (over 98% rates), thus their consideration could yield overrating the performance of the object discrimination abilities of the method.

By analyzing the results, we can conclude that the proposed C^{2}CNN can classify all classes of interest with an F-rate larger than 83%. The precision and recall rates for all classes are quite similar, thus the false negative and false positive hits are nearly evenly balanced. The two most efficiently detected classes are the tram/bus, whose large planar sides are notably characteristic, and vegetation, which usually correspond to unorganized point cloud segments on predictable positions (bushes on street level and tree crowns at higher altitude). Nevertheless, classes with high varieties such as phantoms, pedestrians and cars are detected with 85-87% F-rates, indicating balanced performance over the whole scene.

Since SPLATNet is able to consider both geometry and color information associated to the points, we tested this approach with two different configurations. SPLATNet^{xyz} deals purely with the Euclidean point coordinates (similarly to C^{2}CNN and all other listed reference techniques), while SPLATNet$^{xyz}_{rgb}$ also exploits *rgb* color values associated to the points. As the results confirm in the considered MLS data SPLATNet^{xyz} proved to be slightly more efficient, which is

DOI:10.15774/PPKE.ITK.2020.010

a consequence of the fact, that automated point cloud texturing is still a critical issue in industrial mobile mapping systems, which is affected by a number of artifacts. The overall results of the four reference techniques fall behind our proposed method with a margin of 14.8% (OG-CNN), 17.4% (multi-view), 3.8% (PointNet++), and 5% (SPLATNet^{xyz}) respectively. While the overall Pr and Rc values of all references are almost equal again, there are significant differences between the recognition rates of the individual classes. The weakest point of all competing methods is the recall rate of phantoms, which class has diverse appearance in the real measurements due to the varying speed of both the street objects and the scanning platform. For (static) cars, the recall rates are quite high everywhere, but due to their confusion with phantoms, there are also many false positive hits yielding lower precision. By OG-CNN, many pedestrians are erroneously detected in higher scene regions due to ignoring the elevation channel, which provides some global position information for the C^{2}CNN model, meanwhile preserving the quickness of detection through performing local calculations only.

Apart from the above detailed evaluation on the `SZTAKI CityMLS` dataset, we also tested our method on various existing point cloud benchmarks mentioned in Sec. 2.3. On one hand, we trained the C^{2}CNN method on the annotated part of the *TerraMobilita* dataset [36], and predicted the class labels for different test regions. Some qualitative results of classification are shown in Fig. 2.9, which confirm that our approach could be suited to this sort of sparser measurement set as well, however the number of annotated street objects for training should be increased to enhance the results. We can expect similar issues regarding the *Paris-rue-Madame* dataset [35], while our model does not suite well the *Semantic3D.net* data [37], where the point cloud density is drastically varying due to usage of static scanners.

DOI:10.15774/PPKE.ITK.2020.010

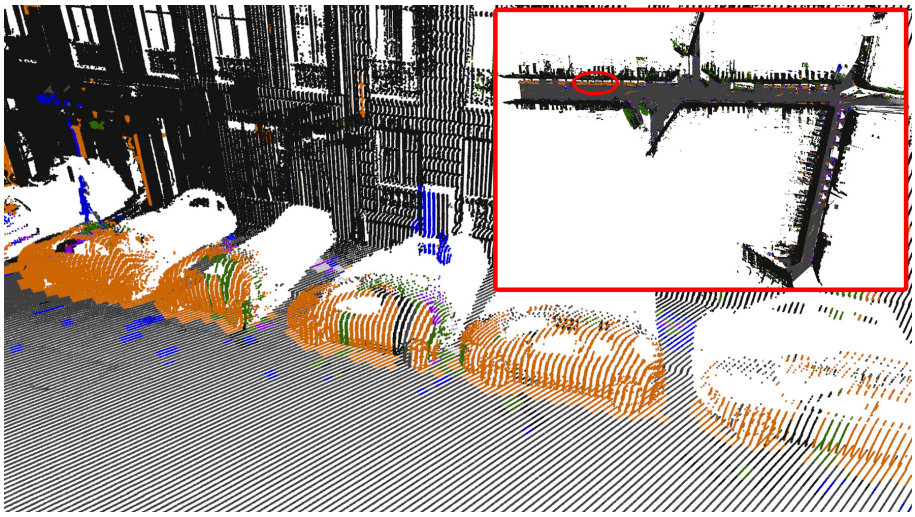


Figure 2.9: Test result on the TerraMobilita data.

Next, we demonstrate that our method can also be adopted to the *Oakland* point clouds [38]. Since that dataset is very small (1.6M points overall), we took a C^{2}CNN network pre-trained on our `SZTAKI CityMLS` dataset, and fine tuned the weights of the model using the training part of the *Oakland* data. Generally, the *Oakland* point clouds are sparser, but have a more homogeneous density than `SZTAKI CityMLS`. As sample results in Fig. 2.10 confirm, our proposed approach can efficiently separate the different object regions here, although some low-density boundary components of the vehicles may erroneously identified as phantoms. Using the *Oakland* dataset, we can also provide quantitative comparison between the C^{2}CNN method, the reference techniques from Table 2.3, and also the Max-Margin Markov Network (Markov) based approach presented in [38]. Table 2.4 shows again the superiority of C^{2}CNN over all references. Both Markov [38] and the C^{2}CNN methods are able to identify the vegetation, ground and facade regions with around 95-98% accuracy, but for pole-like objects, street furniture and vehicles the proposed method outperforms the reference technique with 8-10%. In addition, we have tested the proposed method on the *Toronto 3D* [102] and *Paris-Lille-3D* [103] databases. Without retraining the original network the qualitative results (Fig. 2.11) are promising, however fine-tuning the weights can increase the accuracy.

DOI:10.15774/PPKE.ITK.2020.010

Table 2.4: Quantitative comparison of the proposed method and the reference ones on the *Oakland* dataset. F-rate values are provided in percent.

Class	Markov [21]	PointNet++ [7]	OG-CNN [15]	Multi-view [18]	SPLATNet [20]	Proposed C^2CNN
Vegetation	97.2	91.1	87.3	70.4	84.2	96.5
Ground	96.1	91.8	88.8.1	73.4	92.9	98.6
Facade	95.7	96.3	80.7	68.7	90.1	97.7
Pole-like	64.3	79.2	52.1	45.9	70.6	73.3
Vehicle	67.8	68.0	59.4	60.5	66.2	74.7
Street fur.	59.3	73.4	64.7	59.2	66.8	71.4

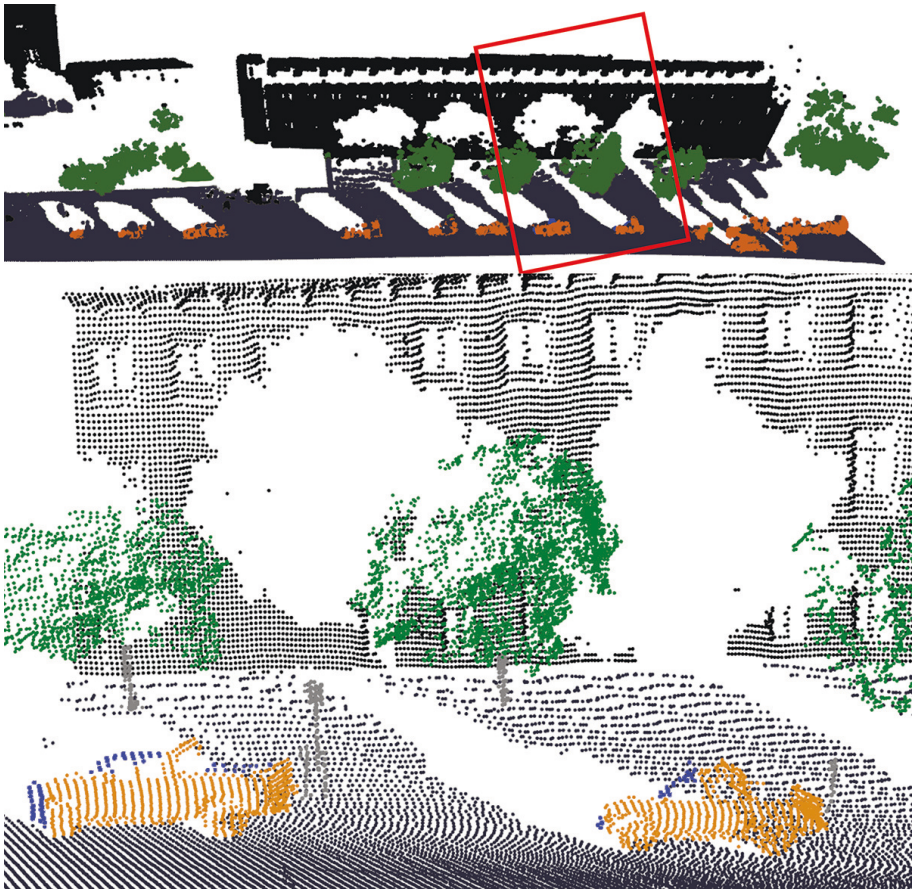
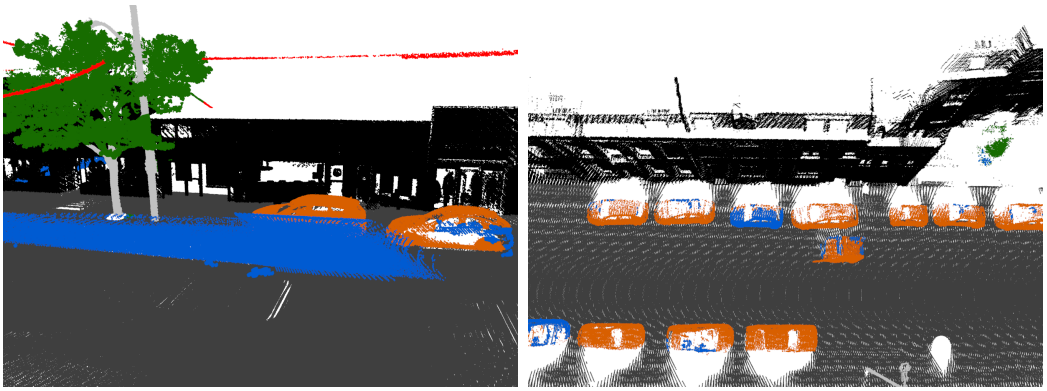


Figure 2.10: Test result on the Oakland data.

DOI:10.15774/PPKE.ITK.2020.010



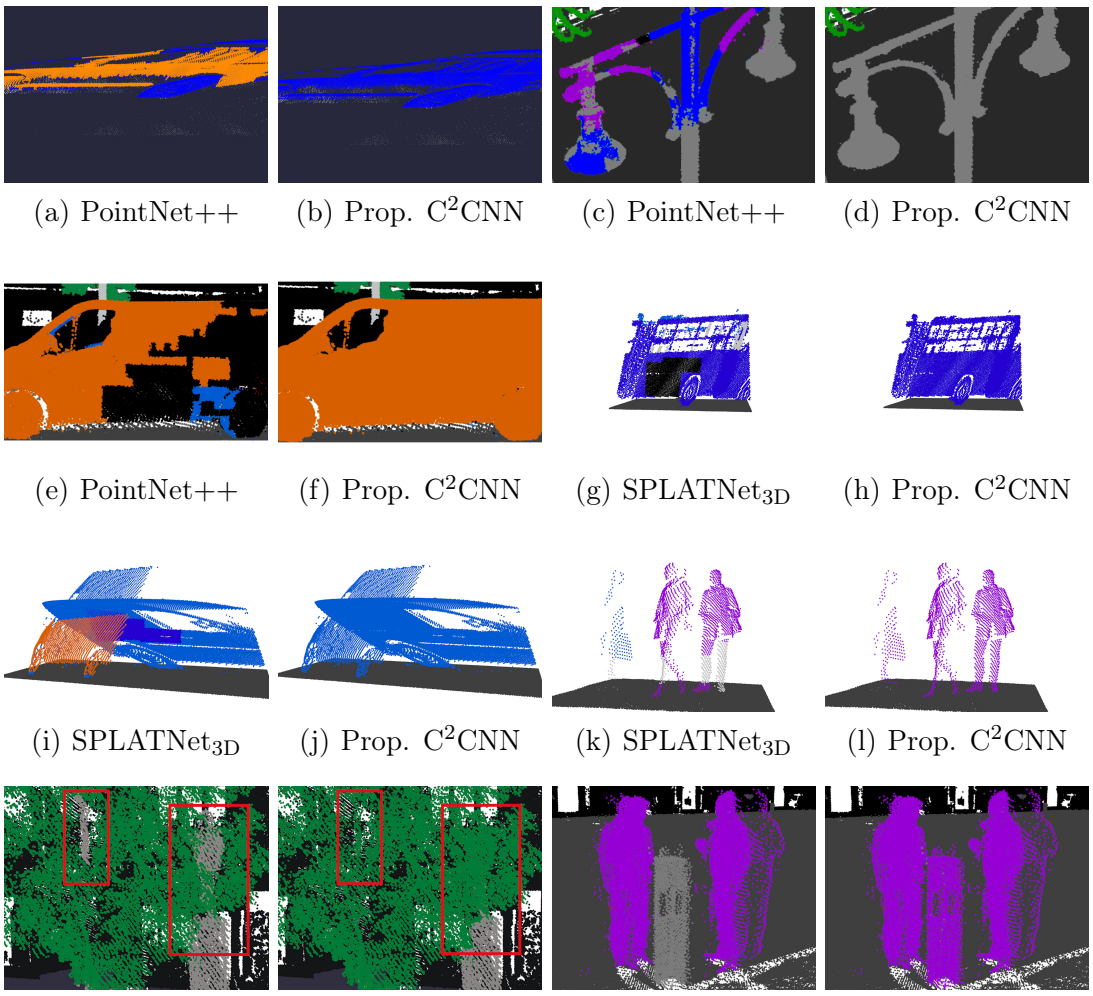
(a) Toronto 3D (b) Paris-Lille-3D

Figure 2.11: Qualitative segmentation results of the proposed C^{2}CNN method.

2.5.4 Failure case analysis of the proposed C^{2}CNN method and the PointNet++and SPLATNet$_{3D}$ references

In Fig. 2.12, we demonstrate typical failure scenarios of PointNet++, SPLATNet$_{3D}$ and the proposed C^{2}CNN, which are the three most successful methods according to Table 2.3. Experimental performance evaluation of PointNet++ and SPLATNet$_{3D}$ in their presenting articles [26, 30] has been restricted to indoor scenes, synthetic databases, or TLS based facade point clouds which are not affected by motion artifacts or heavy occlusion effects. As emphasized in Sec. 2.1 and 2.3 MLS data of the new `SZTAKI CityMLS` benchmark has significantly different characteristic from the existing datasets, and it presents particularly challenging issues such as phantoms, incomplete object segments and multiple occlusions between street objects and the 3D background scene. Some limitations of the PointNet++ approach are shown in three point could segments in Fig. 2.12(a)-(f) with comparative results obtained by proposed C^{2}CNN technique. Since PointNet++ is trained on local point neighborhoods, large phantom regions with inhomogeneous point density often mislead the process (Fig. 2.12(a)). On the other hand, without explicitly considering the global position information, pedestrians, phantoms or street furniture may be also detected on the height level of the tree crowns or street lamps (Fig. 2.12(c)). In other cases large planar vehicle parts may be confused with building facades (Fig. 2.12(e)).

DOI:10.15774/PPKE.ITK.2020.010



(a) PointNet++ (b) Prop. C^{2}CNN (c) PointNet++ (d) Prop. C^{2}CNN

(e) PointNet++ (f) Prop. C^{2}CNN (g) SPLATNet$_{3D}$ (h) Prop. C^{2}CNN

(i) SPLATNet$_{3D}$ (j) Prop. C^{2}CNN (k) SPLATNet$_{3D}$ (l) Prop. C^{2}CNN

(m) Ground truth (n) Prop. C^{2}CNN (o) Ground truth (p) Prop. C^{2}CNN

Figure 2.12: Typical failure cases of the proposed and the reference methods.

DOI:10.15774/PPKE.ITK.2020.010

2.5.5 Implementation details and running time

We implemented our training pipeline in Python using Keras and Tensorflow backend, while the further algorithmic modules were developed in C++ using OpenGL. Training the C^2CNN on the `SZTAKI CityMLS` dataset took around 36 hours, using a Nvidia Geforce GTX 1080 GPU with 8GB device and 64GB main memory. To train the proposed $23 \times 23 \times 23$ kernel size with $0.1m$ voxel resolution requires 969MB device memory, which can be scaled up to $70 \times 70 \times 70$ kernel size requiring around 10GM memory. The label prediction step takes less than 10^{-4} seconds for a 2-channel $23 \times 23 \times 23$ training volume. As an example, by processing a complete scene with ground area $56m \times 111m$, 19M included points and $0.1m$ voxel resolution, our sparse voxel based space representation yielded 1.75M voxels, thus the overall label prediction took around 3 minutes.

We have measured the prediction time of the PointNet++, SPLATNet_{3D} and the proposed C^2CNN techniques on a selected test scene containing 25 million points. We have experienced that while the PointNet++ showed the highest time complexity (563 sec), the running time of SPLATNet_{3D} (198 sec) and the proposed approach (153 sec) proved to be notably shorter. As for the time complexity of the training step, our network is significantly quicker than the two reference methods due to its smaller structure. Note that the proposed model operates on batches of voxel cubes containing local point cloud parts, so it is relatively straightforward to parallelize and scale the method to work on large scale datasets using multi GPU environments and cloud platforms.

2.6 Conclusion of the chapter

In this chapter, we have proposed a new 2-channel 3D CNN based technique to segment point cloud scenes obtained by Mobile Laser Scanning into nine different classes relevant for 3D High Definition city map generation. We have validated the efficiency of the approach in diverse and real test data from various urban environments, and demonstrated its advantages versus three baseline approaches. We have published a new fully annotated MLS benchmark for point cloud segmentation.

DOI:10.15774/PPKE.ITK.2020.010

Chapter 3

Robust registration between different type of point clouds and automatic localization approach for autonomous vehicles

In this chapter we introduce a Lidar based real-time and accurate self-localization approach for self-driving vehicles (SDV) in high resolution 3D point cloud maps of the environment obtained through Mobile Laser Scanning (MLS). Our solution is able to robustly register the sparse point clouds of the SDVs to the dense MLS point cloud data, starting from a GPS based initial position estimation of the vehicle. The main steps of the method are robust object extraction and transformation estimation based on multiple keypoints extracted from the objects, and additional semantic information derived from the MLS based map which we introduced in Chapter 2. We tested our approach on roads with heavy traffic in the downtown of a large city with large GPS positioning errors, and showed that the proposed method enhances the matching accuracy with an order of magnitude. Comparative tests are provided with various keypoint selection strategies, and against a state-of-the-art technique.

DOI:10.15774/PPKE.ITK.2020.010

3.1 Introduction

Self driving vehicles (SDV) offer several benefits for the society ensuring for example a decreased number of road accidents, and more effective traffic distribution on heavy roads. Since these vehicles are equipped with various sensors, apart from their original transportation functionality, they can also contribute to solving environment monitoring, mapping, surveillance, and change detection tasks [53], [54]. Taking the advantage of these massive, moving sensor parks on the roads, algorithms can forecast traffic jams, and they can automatically notify the community about particular or unusual events such as accidents, or police actions.

3.1.1 Problem statement

Accurate and robust localization and environment mapping are key challenges in autonomous driving. Although the GPS-based position information is usually suitable for helping human drivers, its accuracy is not sufficient for navigating a self driving vehicle. Instead, the accurate position and orientation of the SDV should be calculated by registering the measurements of its onboard visual or range sensors to available 3D high definition (HD) city maps [55, 56].

Point cloud registration task can be formulated as an optimization problem to find the best transformation matrix between the two point clouds which minimizes the squared error of point-wise distances. Assuming a rigid body transformation, since Lidar sensors measure real distances, the point cloud registration task has six degree of freedom: three translation and three rotation components.

So automatic point cloud registration is a key step in many applications such as simultaneous localization and mapping (SLAM) and mobile surveillance, especially if precise position information of the acquisition platform is not available due to lack of accurate navigation signals. However answering different functional requirements and due to the manufacturer's unique innovative approaches the available sensors may provide point clouds with very different density characteristics [58], limiting the general usability of standard point cloud registration techniques [59, 60], or methods developed for specific sensors [61].

DOI:10.15774/PPKE.ITK.2020.010

Rotating Multi-beam (RMB) Lidar laser scanners [15, 57] mounted on vehicle tops are efficient candidates to ensure robust positioning of SDVs. RMB Lidars measure directly the range information, and as active light based sensors, they efficiently perform under different illumination and weather conditions, offering accurate point cloud data with a large field of view in real-time. Although RMB Lidars provide strong geometric features about the environment, the captured point cloud data is quite sparse and inhomogeneous. In addition due to nature of the scanning process, objects are effected by occlusions and motion artifacts, thus robust object detection and classification on such measurements are not straightforward [15]. The point clouds of RMB Lidars are originally obtained in the sensor's local coordinate system, which can be shifted with the actually measured GPS position of the SDV. However in urban environments, due to lack of accurate navigation signals the error of GPS-based position estimation may be often around 2-10 meters.

(a) RMB Lidar scan (b) MLS point cloud (c) Segmented MLS cloud

Figure 3.1: Point cloud scenes captured at a downtown area using a Velodyne HDL64E Rotating Multi-Beam (RMB) Lidar sensor and a Riegl VMX450 Mobile Laser Scanning (MLS) system. Class color codes in the segmented cloud; black: *facade*, dark gray: *ground*, mid gray: *tall column*, bright gray: *street furniture*, green: *tree crowns*

3.1.2 Sensors discussed in this chapter

In this chapter, we utilize the measurements of the Velodyne HDL64E RMB Lidar sensor and the Riegl VMX450 MLS system. The Velodyne sensor was originally designed to help real-time perception of autonomous vehicles or robots. It provides a stream of relatively sparse (60-100 $\times$ 10^3 points/frame) point clouds

DOI:10.15774/PPKE.ITK.2020.010

with a temporal frequency of 10-15 fps. The spatial accuracy is around 1-2cm in the sensor's own coordinate system, but the point density quickly decreases as a function of the distance from the sensor. As a result of the rotating multi-beam scanning approach, the point clouds show typical ring patterns (Fig. 3.1(a)).

As we mentioned in Chapter 2, the Riegl VMX450 MLS system was designed to city mapping, urban planning and road surveillance applications. Contrary the sparse data of the Velodyne HDL64E sensor, it offers extremely dense and feature rich data with color information as shown in Fig. 3.1(b).

3.1.3 Aim of the chapter

For accurate self localization of the SDV, we have to register the sparse onboard RMB Lidar (Velodyne HDL64E) data with GPS based coarse initial position estimation to the dense and accurate MLS point cloud, used as HD map. For semantic segmentation of the MLS point cloud we used our C^2CNN method [2] proposed in Chapter 2. While point cloud registration is a deeply explored topic, matching 3D measurements with such different point density and characteristics is a highly challenging task. In this chapter, we propose an accurate and fast object based alignment algorithm between the RMB Lidar point clouds and the MLS HD map for self localization and we also introduce a showcase of an IMU-free SLAM based on sparse RMB Lidar data to demonstrate the general usability of the proposed point cloud alignment method.

3.2 Related work

Normal Distribution Transform (NDT) [60] and Iterative Closest Point (ICP) [59] algorithms are among the most cited methods in the field of point cloud registration. ICP has several different versions with various improvements: [62] extend the ICP with geometric constraints derived from local point neighborhoods, [63] use color information and [64] make improvements via tracking. However all of these methods are quite sensitive to the different density characteristics of the point clouds, particularly the typical ring pattern of the Velodyne sensor may mislead the registration process. Moreover all of the mentioned methods have a critical precondition: ICP based methods locally minimize the error, so they

DOI:10.15774/PPKE.ITK.2020.010

need a sufficiently accurate pre-alignment between the point clouds. In practice, the GPS based initial alignment with an error of several meters does not prove sufficiently good enough for this purpose. Other techniques focus on applications with larger displacement between between the point clouds: [65] and [66] extract local feature descriptors to find global correspondences, which approach can also be used to find an initial pre-alignment before the ICP process. However, as bottleneck, these algorithms have large computational cost even working with smaller point cloud parts, thus in real-time mapping, SLAM and localization applications they are not efficient. A technique has been introduced for scan alignment based on ICP [57], which solves data mapping at the object level by explicitly matching segments across scans rather than using standard point-to-point type of search. This method proved to be efficient for matching Velodyne frames, however the computational time remained 3-15 seconds per scan pair. Registering point clouds with different modalities and density characteristics has a limited bibliography. Non-ICP-based approaches have also been proposed e.g. [67] which exploits the nature of a rotating multi-beam Lidar (such as the Velodyne sensors) for plane detection, and applies real-time registration of the extracted planes. Although this method could lead to real-time SLAM, we must note that in many real world scenarios the plane detection step may mean a significant bottleneck of the process. A sequential technique for cross modal point cloud alignment has be proposed in [12], which extracts first abstract object patches in both point clouds, then it calculates a coarse alignment between the frames purely based on the estimated object centers, finally an NDT based point level refinement process is applied. As drawbacks, the object level matching may fail in case of several diverse object types, and the additional NDT steps induces significant computational overload.

3.3 Proposed approach

We propose a real-time (15 fps, the scanning speed of the sensor), robust object based alignment technique between sparse RMB Lidar measurements and dense MLS point clouds. The workflow of the new approach is shown in Fig. 3.2. As a preliminary step, we perform a semantic segmentation of the MLS data, and jointly exploit the raw point cloud and the extracted labels as a High

DOI:10.15774/PPKE.ITK.2020.010

Definition map, to support the registration process. For estimating the optimal scan matching transform, we adopt the fingerprint minutia matching algorithm [12, 40], which is able to find a robust transformation between two point sets even if the size of the point sets are different. We recommend a new key point selection technique that yields an accurate alignment without the computationally expensive point-level refinement step, and also provides more stable results in scenes with several - often only partially extracted - objects. In addition, we also apply semantic constraints for matching the object candidates obtained by a quick segmentation and patch analysis of the actual RMB Lidar frame, and the preliminary extracted segmentation labels of the HD map.

Our algorithm consists of four major steps: point cloud segmentation, abstract field object extraction, key point selection and object based coarse transformation estimation.

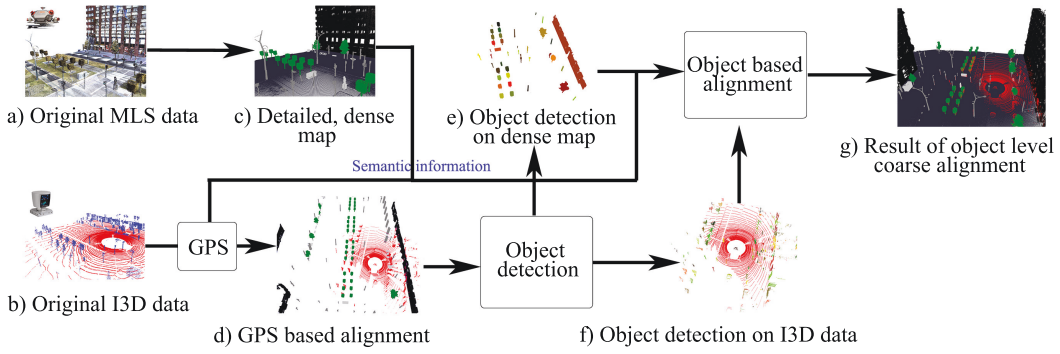


Figure 3.2: The workflow of the proposed point cloud alignment algorithm.

3.3.1 Creating the reference High Definition map

The Riegl VMX450 system rapidly provides detailed and very dense MLS point cloud data from large-scale outdoor environments. The captured data is georeferenced, therefore following semantic segmentation it can be directly used as a 3D High Definition (HD) map. Although manual labeling of billions of points is a highly resource intensive task, deep learning based point cloud classification approaches such as the PointNet++ [26] offer promising way to automate the process. However, dealing with urban MLS data a number of particular challenges

DOI:10.15774/PPKE.ITK.2020.010

appear - such as the phantom effect caused independent object motions [11] - , which are not handled by general point clouds segmentation algorithms. For this reason, we applied here a 3D convolutional neural network based technique [2] proposed in Chapter 2 developed particularly for MLS data filtering, by accommodating it to separate various urban classes such as *ground*, *facade*, *phantom*, *vehicle*, *pedestrian*, *vegetation* (bushes and tree crowns), *tall column* (including traffic sign holders and tree trunks) and *street furniture* (various further street objects such as benches, dustbins, short columns). Fig. 3.1(c) demonstrates the result of the labeling process.

Object separation in the segmented MLS HD map is quite straightforward: object samples, such as a particular traffic sign, are obtained from the corresponding segmented class regions by Euclidean clustering [68] in an offline way. For the subsequent scene registration process, we will use the objects of the MLS based map as landmarks in background model, therefore will ignore all dynamic (*phantom*, *vehicle*, *pedestrian*) or time-varying objects (*vegetation*), as well as classes of large regions (i.e. *ground* and *facade*). As a consequence, for scene matching we will rely on the extracted object samples of the *tall column*, and *street furniture* classes, which all have a static appearance, compact shape thus can be used as landmarks.

3.3.2 Real time object detection in the RMB Lidar point clouds

Since we are dealing with object based point cloud alignment, accurate and robust object detection is also essential in the RMB Lidar frames. On the other hand, the task is highly challenging here, due to the low and inhomogeneous point density, several partially scanned object shapes, occlusions and the real time requirement of the process. First, we only keep the points within a $r = 30$m radius region around the sensor's rotation axis (parameter r was optimized for Velodyne HDL64), as the distant regions are too sparse for reliable scene analysis.

DOI:10.15774/PPKE.ITK.2020.010

3.3.2.1 Ground removal

On one hand, the typical ring patterns of RMB Lidars particularly affect the ground regions, which phenomena can mislead the registration process. Furthermore, separation of field objects is facilitated by eliminating the ground which connects the object candidates in the raw frames.

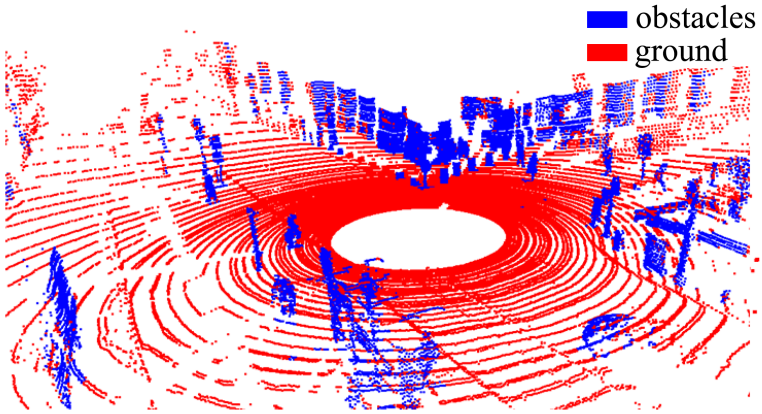
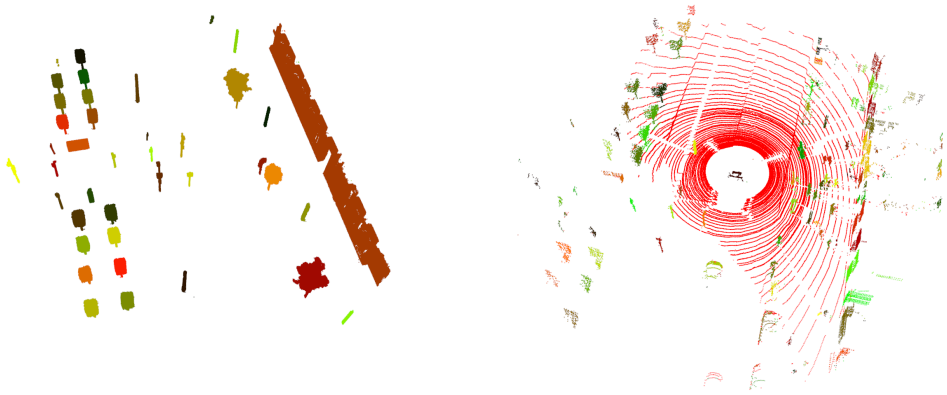


Figure 3.3: Result of ground removal in RMB Lidar point cloud.

Planar ground models are often used in the literature based on robust plane estimation methods such as RANSAC, however, they are less efficient in cases of significant elevation differences within the observed terrain parts (e.g. uphill and downhill roads). Instead, we apply a cell based locally adaptive terrain modeling approach based on [15]. First, we fit a regular 2-D grid with $0.2m$ rectangle width (i.e. grid distance) - optimized to urban environment according to [15] - onto the P_z horizontal plane of the RMB Lidar point cloud's local Euclidean coordinate system. We assign each point to the corresponding cell, which contains its projection to P_z. We mark the cells as ground (road) candidate cells where the differences of the observed maximal and minimal point elevation values are lower than $0.1m$, which condition admits up to 26° ground slope within a cell. Next, for obtaining a local elevation map, we calculate for the previously marked ground candidate cells the average of the included point elevation coordinates. To eliminate outlier values in the elevation map, resulted by e.g. flat car roofs,

DOI:10.15774/PPKE.ITK.2020.010

we apply a median filter considering neighboring ground cells. For the remaining non-ground cells - which presumptively contain the field objects - the local ground elevation value z_0 is interpolated using the neighboring ground cells, and all included points with elevation z_p are denoted as non-ground points, where $z_p - z_0 > \tau$ (used $\tau = 0.1$m). The result of ground removal is shown in Fig. 3.3.



(a) Landmark objects of the HD map (b) Detected objects in RMB Lidar frame

Figure 3.4: Extracted objects used for the alignment calculation. Color codes are the following (a) different landmark objects of the HD map are displayed with different color (b) red: ground/road, other colors: different detected objects in the RMB Lidar frame.

3.3.2.2 Abstract field object extraction

After ground removal, we cluster corresponding *non-ground* points to separate individual object candidates. This process is implemented in the 2D cell map with a region growing algorithm, where empty cells act as stopping criterion. Although in this way, some adjacent objects may be merged together due to the limited resolution of the grid, this 2D object detection approach proved to be by two orders of magnitude faster than traditional kd-tree based 3D Euclidean clustering algorithms. Let us note again, that unlike by offline HD map generation, here the processing speed should fulfill the real time requirements.

Fig. 3.4(b) allows us a qualitative analysis of the object detection step. Field object such as vehicles, columns or tree samples usually appear as separated blobs,

DOI:10.15774/PPKE.ITK.2020.010

while large facade regions are separated into smaller wall segments. Nevertheless, considering the above detailed limitations of the RMB Lidar point clouds, we do not perform a strict classification of the extracted object blobs, and will use all of them in the subsequent scene matching process.

3.3.3 Object based alignment

In this section, we aim to estimate the optimal geometric transform for registering the sparse *observation frame* recorded by the RMB Lidar to the MLS based HD map data. First, we use the GPS-based coarse position estimation of the vehicle (p_0) for an initial positioning of the *observation frame*'s center in the HD map's global coordinate system. To make the bounding area of the two adjustable point clouds equal, we also cut a 30m radius region from the MLS cloud around the current p_0 position.

Exploiting that the Lidar sensors provide direct measurements in the 3D Euclidean space up to cm accuracy, the estimated spatial transform between the two frames can be represented as a rigid similarity transform with a translation and a rotation component. On one hand, we search for a 3D translation vector (dx, dy and dz), which is equal to the originally unknown position error of the GPS sensor. On the other hand, we have experienced that assuming a locally planar road segment within the search region, the road's local normal vector can be fairly estimated in an analytically way from the MLS point cloud, thus only the rotation component α around the vehicle's up vector should be estimated via the registration step. In summary, we model the optimal transform between the two frames by the following homogeneous matrix:

$$T_{dx,dy,dz,\alpha} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & dx \\ -\sin\alpha & \cos\alpha & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

For limiting the parameter space, we allow a maximum 45° degree rotation (α) in both directions, since from the GPS data, we already know an approximate driving direction. For parameters dx and dy we allow $\pm$12m offsets, while for the vertical translation $\pm$2m. In a typical urban environment this limitation of the parameter space usually yields four times less calculation.

DOI:10.15774/PPKE.ITK.2020.010

We continue with the description of the transformation estimation algorithm. Instead of aligning the raw point clouds, our proposed registration technique matches various keypoints extracted from the *landmark objects* of the HD map (Sec. 3.3.1), and the *observed object candidates* in the RMB Lidar frame (Sec. 3.3.2). In addition, exploiting the semantic information stored in the HD, we only attempt to match keypoints which correspond to *compatible* objects. Therefore the remaining part of the algorithm consists of three steps, presented in the following subsections: i) *keypoint selection*, ii) defining *compatibility constrains between observed and landmark objects*, iii) *optimal transform estimation* based on compatible pairs of keypoints.

3.3.3.1 Keypoint selection

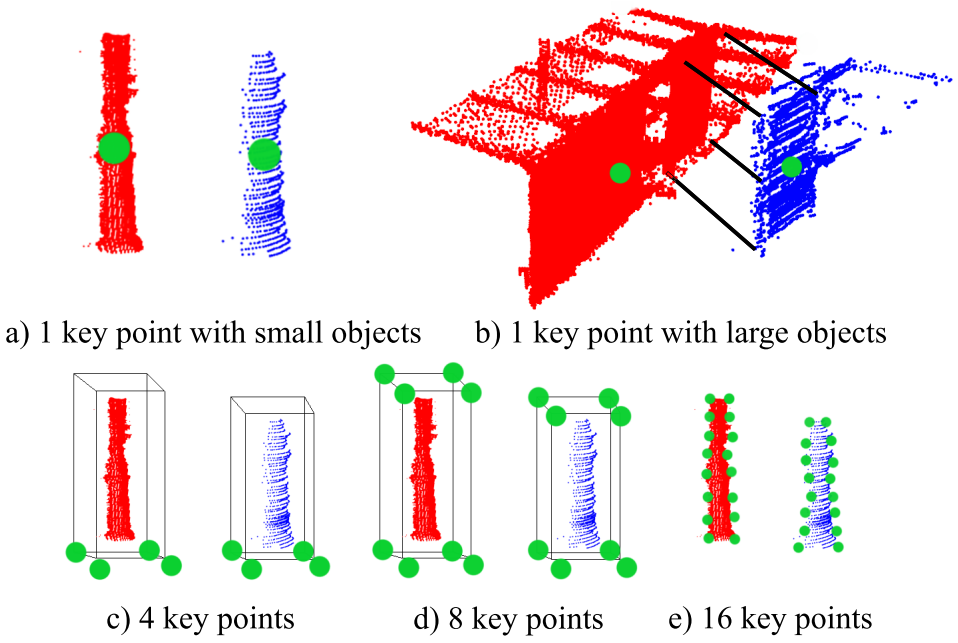


Figure 3.5: Choosing key points for registration.

A critical step of the proposed approach is keypoint extraction from the observed and landmark objects. A straightforward choice [12] is extracting a single keypoint from each object, taken as the center of mass of the object's blob (see Fig.

DOI:10.15774/PPKE.ITK.2020.010

3.5(a) and (b)). However, as discussed earlier, the observed RMB Lidar point clouds contain several partially scanned objects, thus the shape of their point cloud blobs may be significantly different from the more complete MLS point cloud segments of the same object, yielding that the extracted center points are often very different.

For the above reasons, we have implemented various multiple keypoint selection strategies. Beyond the single keypoint based registration approach, we tested the algorithm's performance using 4, 8 and 16 keypoints, whose alignment is demonstrated in Fig. 3.5(c),(d) and (e). As shown there, using the 4- and 8-keypoint strategies, the feature points are derived as corner points of the 3D bounding boxes of the *observed* and *landmark* objects. For the 16-keypoint case, we divide the 3D bounding box of the object into $2 \times 2 \times 4$ equal cuboid regions, and in each region we select the mass center of the object boundary points as keypoint.

Our expectation is here, that using several keypoints we can obtain correct matches even from partially extracted objects, if certain corners of the (incomplete) bounding box are appropriately detected. On the other hand using a larger number of keypoints induces some computational overload, while due to the increased number of possible point-to-point matching options, it may also cause a false optimum of the estimated transform.

3.3.3.2 Compatibility constrains between observed and landmark objects

As discussed earlier, we estimate the optimal transform between two frames via sets of keypoints. Since we implement an object based alignment process, our approach allows us to filter out several false keypoint matches based on object level knowledge. More specifically, we will only match point pairs extracted from *compatible* objects of the scene. According to the HD map generation process (Sec. 3.3.1), we can distinguish *tall column* and *street furniture* samples among the landmark objects, thus all landmark keypoints are derived from samples of the above two object types. Although such detailed object classification was not feasible in the RMB Lidar frame, we prescribe the following compatibility constraints:

DOI:10.15774/PPKE.ITK.2020.010

- a *tall column* MLS landmark object is compatible with RMB Lidar blobs, which have a column shaped bounding box, i.e. its height is at least twice longer than its width and depth.

- the ratio of the bounding volumes of compatible RMB Lidar objects and *street furniture* MLS landmark objects must be between 0.75 and 1.25.

Applying the above pre-defined constraint we increase of the evidence of a given transformation only if the objects pairs show similar structures, moreover in this way by skipping many transformation calculations we increase the speed of the algorithm.

Note that the RMB Lidar point clouds may contain various dynamic objects such as pedestrians and vehicles, which fulfill the above matching criteria with certain landmark objects of the MLS based map. These objects will generate outlier matches during the transformation estimation step, thus their effect should be eliminated at higher level. In a typical urban environment the proportion of good landmark objects is between 20 and 40 percents.

3.3.3.3 Optimal transform estimation

Let us denote the sets of all *observed* and *landmark* objects by $\mathcal{O}_o$ and $\mathcal{O}_l$ respectively.

Using the 3D extension of the Hough transform based schema [40], we search for the best transformation between the two object keypoint sets by a voting process (Fig. 3.6). We discretize the transformation space between the minimal and maximal allowed values of each parameter, using $0.2m$ disrectization steps for the translation components and $0.25°$ steps for rotation.

Next we allocate a four dimensional array to summarize the votes for each possible (dx, dy, dz, α) discrete quadruple, describing a given transformation. We set zero initial values of all elements of this array.

During the voting process, we visit all the possible O_o, O_l pairs of *compatible* objects from $\mathcal{O}_o \times \mathcal{O}_l$. Then, we attempt to match each keypoint of O_o to the corresponding keypoint in O_l, so that for such a keypoint pair o_o, o_l, we add a vote for every possible $T_{dx,dy,dz,\alpha}$ transform, which maps o_o to o_l. Here we iterate over all the discrete $\alpha^* \in [-45°, +45°]$ values one by one, and for each α^* we

DOI:10.15774/PPKE.ITK.2020.010

rotate o_o with the actual α^*, and calculate a corresponding translation vector $[dx^*, dy^*, dz^*]^T$ as follows:

$$\begin{bmatrix} dx^* \\ dy^* \\ dz^* \end{bmatrix} = o_l - \begin{bmatrix} \cos\alpha^* & \sin\alpha^* & 0 \\ -\sin\alpha^* & \cos\alpha^* & 0 \\ 0 & 0 & 1 \end{bmatrix} o_o$$

Thereafter we increase the number of votes given for the $T_{dx^*,dy^*,dz^*,\alpha^*}$ transform candidate. Finally at the end of the iterative voting process, we find the maximum value of the 4-D vote array, whose (α, dx, dy and dz) parameters represent the optimal transform between the two object sets.

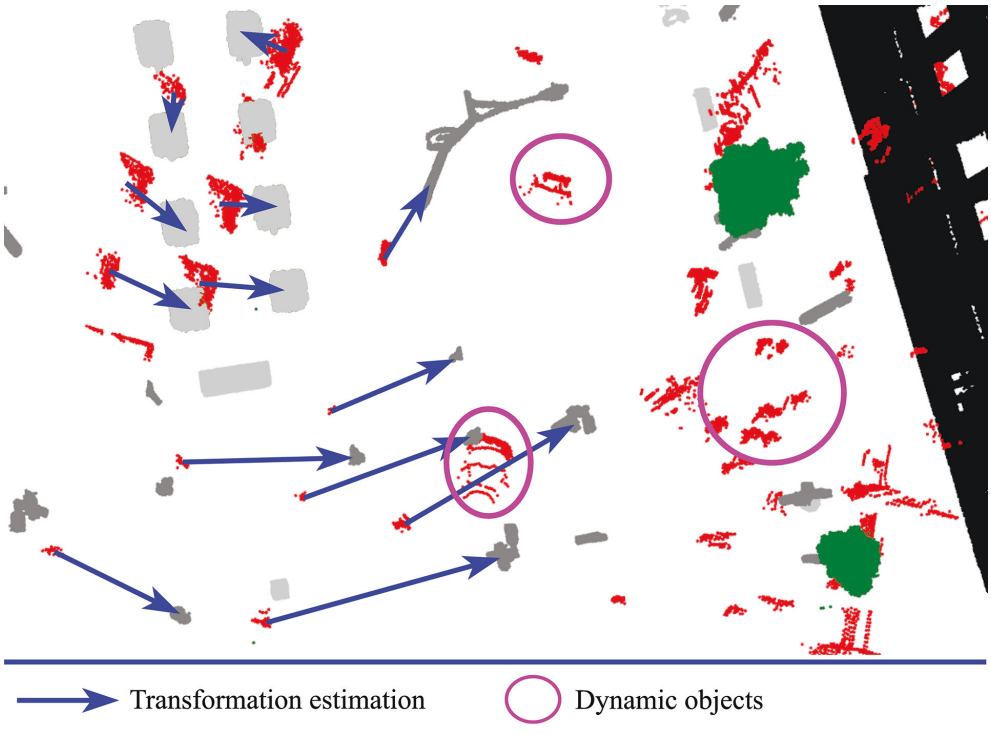


Figure 3.6: Illustration of the output of the proposed object matching algorithm based on the *fingerprint minutiae* approach [40]. Red points mark the objects observed in the RMB Lidar frame.

DOI:10.15774/PPKE.ITK.2020.010

3.4 Experiments and evaluation

We evaluated the proposed method on different scenarios from dense city areas, some qualitative results are shown in Fig. 3.7. During quantitative evaluation we compared the different keypoint selection strategies using our proposed model, and also compared our approach to the state-of-the-art cross-modal point cloud registration technique [12]. As evaluation metrics we used the average distance of the keypoints after the optimal point cloud alignment, since following our subjective visual verification this metrics proved to be relevant for numerical comparison of different matches.

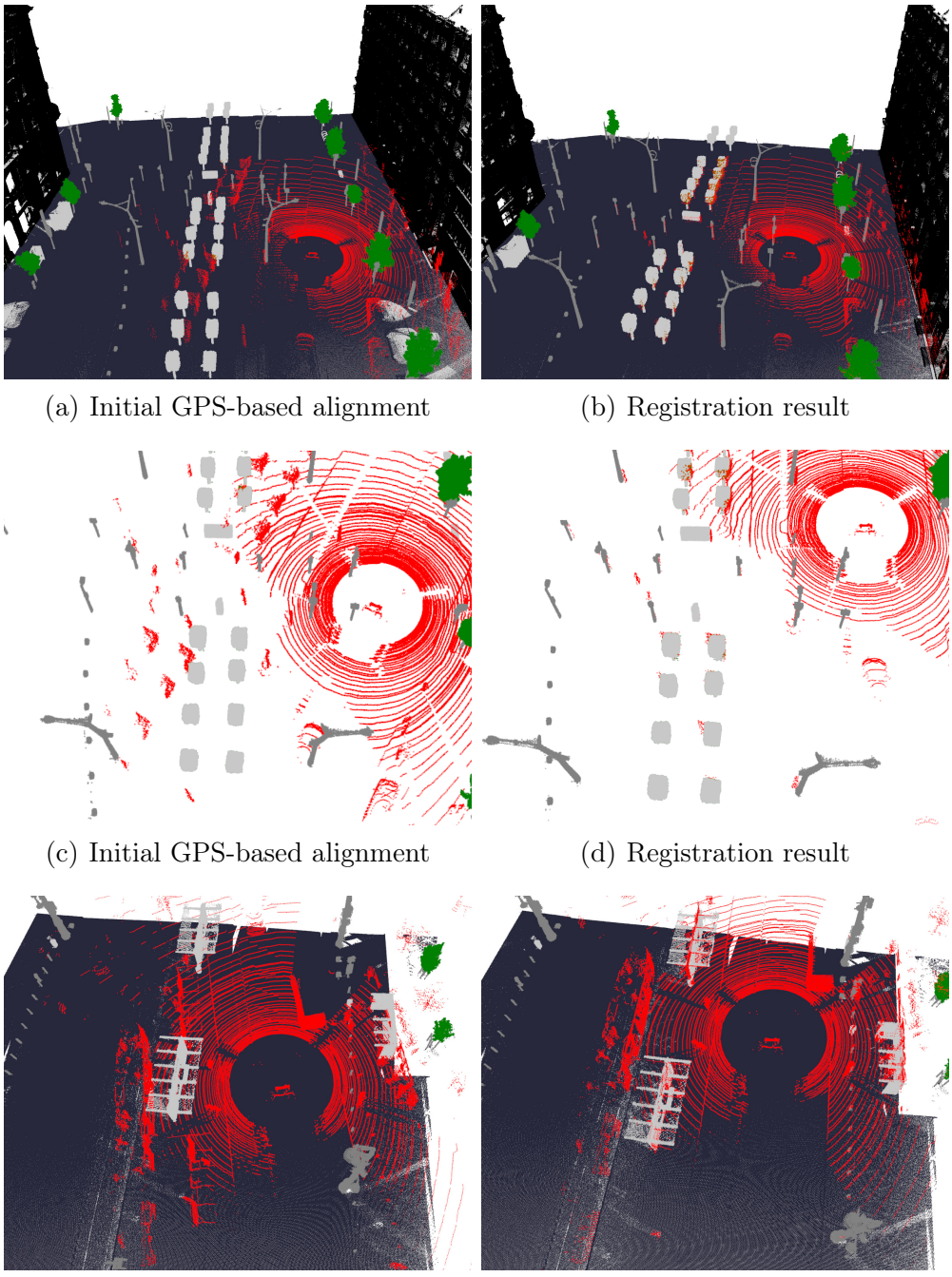
Fig. 3.8 demonstrates the result of the method comparison in 25 different RMB Lidar frames, where we displayed the calculated transformation scores in a logarithmic scale. Regarding the keypoint selection, Fig. 3.8 shows that the optimal strategy proved to be the 8-keypoint approach (shown in Fig. 3.5(d)), with an average error between 0.15 and 0.5 meters for the different frames. We can observe that using 1 or 4 keypoints, the resulting error is slightly higher than with applying the 8-point version. On the other hand, 16 keypoint selection suffers from ovefitting problems, since it yields large errors in some of the frames.

By comparing the proposed method to [12], we can confirm the clear superiority of our new technique with any keypoint selection variants. On one hand, the reference technique [12] only used the 2D object centers from a top-view projection to find the optimal transform, which solution was only appropriate to find a coarse match between the two point cloud frames. On the other hand, [12] did not use any object specific knowledge from the HD map, that highly contributed to eliminate false matches in our present model. In addition, since our method does not use the computationally expensive point level NDT refinement step, it is able to run with 15 frame/seconds (fps) on a desktop computer, in contrast to the 0.5 fps speed of [12].

3.4.1 Case Study on Vehicle Localization Based on the Semantically Labeled MLS Point Cloud

The proposed registration algorithm is based on the assumption that the reference landmark objects extracted from the MLS map correspond in majority to

DOI:10.15774/PPKE.ITK.2020.010



(a) Initial GPS-based alignment (b) Registration result

(c) Initial GPS-based alignment (d) Registration result

(e) Initial GPS-based alignment (f) Registration result

Figure 3.7: Results of the proposed registration approach with 8 keypoint selection strategy. RMB Lidar point clouds are displayed with red, while the MLS data is shown with multiple colors depending on the segmentation class. First two rows correspond to the same scene, just ground and facades are not displayed in the 2nd row for better visualization of the object based alignment.

DOI:10.15774/PPKE.ITK.2020.010

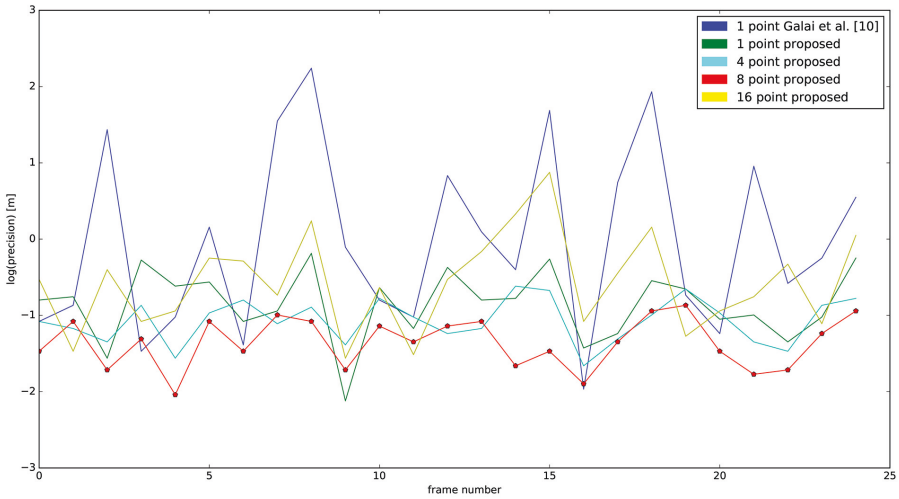


Figure 3.8: Evaluation of the proposed approach with various keypoint selection strategies and comparison to [12]

static and permanent scene elements (such as lamp posts, tree trunks, kiosks etc), while all the phantoms, and moving or movable objects of the MLS point cloud appear as noise factors during the estimation of the right transform. Obviously, all scanning artifacts have a great effect on the object assignment step, e.g. erroneously matching several *phantoms* in the MLS maps to static or dynamic objects of the RMB Lidar frames may increase the evidence of false global transforms in the Hough space. For this reason, semantic preliminary labeling of the MLS reference map is a critical step for this application.

Fig. 3.9 demonstrates the improvements on the registration results by exploiting the labels obtained by the proposed C^2CNN approach. In all subfigures, the RMB Lidar point cloud of the SDV is shown in *red*, while the MLS point cloud is displayed with the remaining different colors corresponding to the obtained C^2CNN-labels. The top row shows the purely GPS based alignment of the two point clouds, the only difference is that while Fig. 3.9(a) displays all points of the original MLS data, Fig. 3.9(b) contains the filtered static MLS regions only. We can observe here initial translation and rotation errors of around 7 meters and 8.5 degrees, respectively. The bottom row visualizes the registration results. In case of Fig. 3.9(c) the complete MLS point cloud was used as input of the regis-

DOI:10.15774/PPKE.ITK.2020.010

Table 3.1: Quantitative evaluation of the proposed point cloud registration technique based on the raw MLS point cloud, the semantically labeled point cloud using the proposed C^{2}CNN approach, and the manually labeled data, respectively.

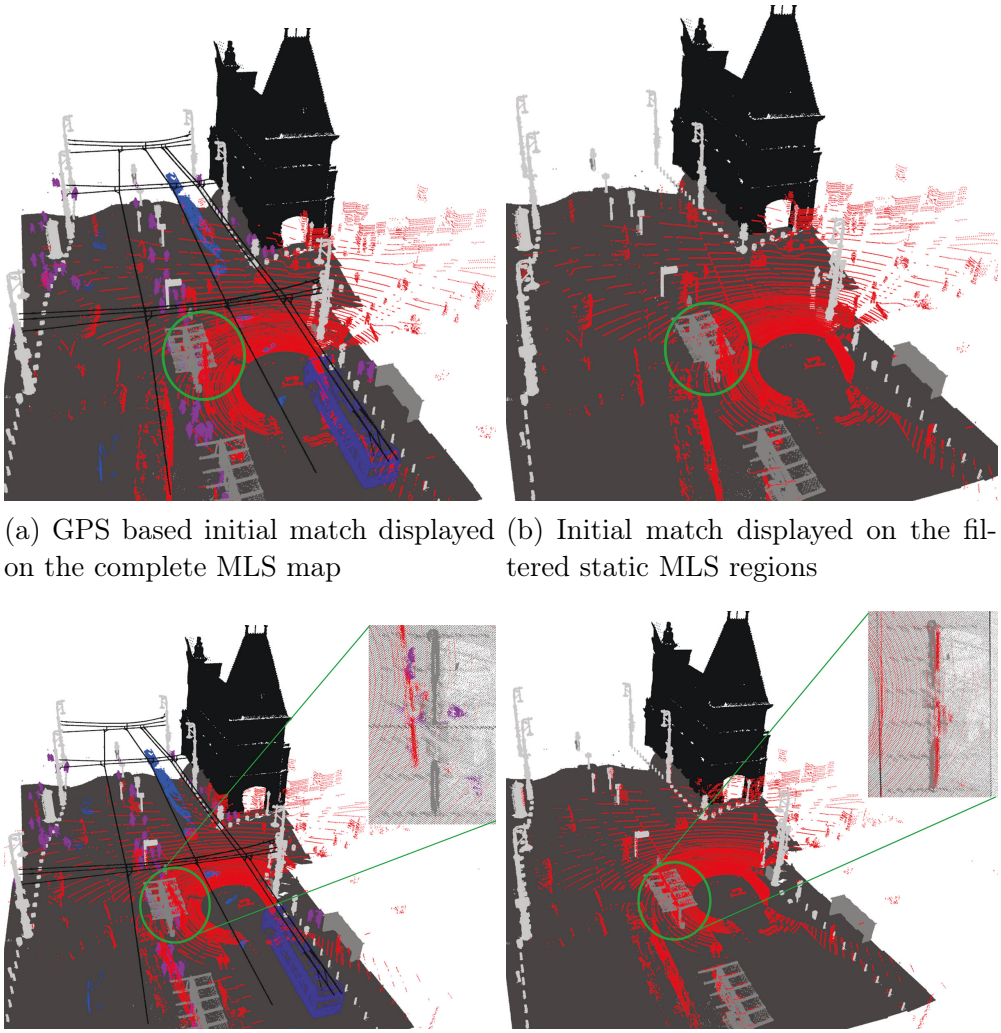
Dataset	Raw MLS point cloud		C^{2}CNN labeled data		Manually labeled data	
	s [m]	rot [deg]	s [m]	rot [deg]	s [m]	rot [deg]
Main roads	1.74	3.92	0.37	1.19	0.26	0.97
Narrow roads	1.37	2.38	0.29	0.83	0.18	0.78
Crossroads	2.42	4.02	0.45	1.33	0.29	0.89
Small #of phantoms	0.93	1.60	0.26	0.87	0.21	0.77
Large #of phantoms	2.14	3.53	0.48	1.37	0.28	0.95
Overall	1.72	3.09	0.37	1.18	0.24	0.87

Note: Translation distance error (s) is given in meter and rotation error is given in degree.

tration step, yielding notable inaccuracies. On the other hand, if we eliminate all phantoms and movable objects from the MLS map with C^{2}CNN, the registration process provides a successful output as shown in Fig. 3.9(d).

We also measured the advantages of the C^{2}CNN filter on registration accuracy in a quantitative way. We run the proposed registration algorithm between the actual RMB Lidar frame and the MLS scenes in three configurations, using as reference map (i) the raw MLS point cloud, (ii) the static point cloud filtered by C^{2}CNN, and the (iii) manually filtered static point cloud. We divided the point cloud scenes for registration evaluation into different tests set: based on location category we distinguished narrow streets, main roads and large crossroads, while we also separated MLS scenes with dense and sparse phantom effects, respectively. The resulting registration errors in offset (s) and rotation (rot) are shown in Table 3.1. We can see that the C^{2}CNN-based semantic filtering process significantly decreased the registration inaccuracies compared to the raw MLS data input, and the result's accuracy is very close to the output got by the utilization of the manually filtered map. The improvements are particularly significant in main roads and crossroads, where the presence of false landmarks is stronger without semantic labeling, and in the selected scenes with large phantom regions which could mislead the matching step working on raw data.

DOI:10.15774/PPKE.ITK.2020.010



(a) GPS based initial match displayed on the complete MLS map

(b) Initial match displayed on the filtered static MLS regions

(c) Registration result based on the unfiltered map

(d) Registration result on C^{2}CNN filtered map

Figure 3.9: Application of the proposed C^{2}CNN classification approach for point cloud registration enhancement. Automatic registration results of a sparse RMB Lidar point cloud (shown with red in all images), to the dense MLS measurements (remaining colors). Figure (c) shows the registration results on the raw point cloud with notable inaccuracies (different class colors in MLS only serve better visibility). Figure (d) demonstrates the output of successful registration based on removing the dynamic objects from the MLS point cloud using the proposed C^{2}CNN method.

DOI:10.15774/PPKE.ITK.2020.010

3.4.2 Case study of IMU-free SLAM based on RMB Lidar data

To demonstrate that the proposed registration algorithm can be easily adapted to other registration problems, we qualitatively demonstrated the result of the registration in a SLAM problem. Using the proposed registration algorithm on selected consecutive frames of a single Lidar sensor, an accurate 3D map of the urban environment can be constructed without the help of any external or internal navigation sensors such as GPS or IMU. The proposed object based alignment is able to register two consecutive frames up to 0.5m registration error accuracy, which can already be handled by the NDT step of the process. Fig. 3.10 shows efficient registration results using Velodyne HDL64 sensor. To keep low the computational cost of the NDT algorithm we only relied on the point cloud parts belong to the objects extracted during the proposed object based course alignment process. The proposed algorithm can be easily adapted for various point cloud alignment problems, such as registering point clouds from the same source or registering point clouds with very different density characteristics.

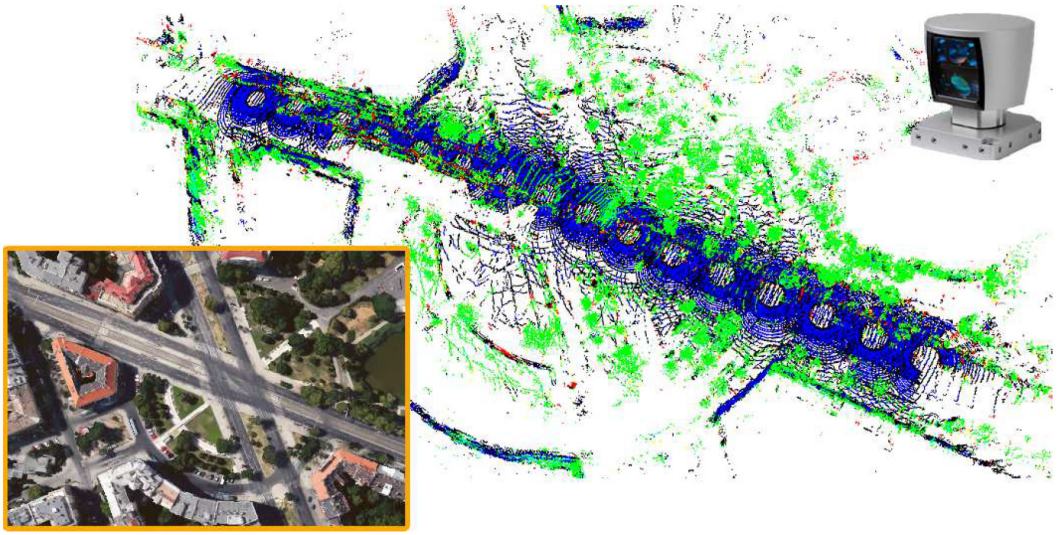


Figure 3.10: SLAM results with Velodyne HDL64 in Kosztolányi tér, Budapest (1.2M points from 80 frames captured at 3 fps from a moving vehicle).

DOI:10.15774/PPKE.ITK.2020.010

3.5 Conclusion of the chapter

In this chapter, we have proposed an object based point cloud alignment algorithm for accurate localization of self driving vehicles (SDV) equipped with a RMB Lidar sensor. Assuming that a High Definition (HD) point cloud map is available from the environment obtained by Mobile Laser Scanning technology, the problem is to solve the registration of point clouds with significantly different density characteristics. Apart from exploiting semantic information from the HD map, various keypoint selection strategies have been proposed and compared. We have experienced that the 8-keypoint approach yields a highly efficient solution for the problem, which is superior over other keypoint selection strategies and also over a state-of-the-art method.

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

Chapter 4

On-the-fly, automatic camera and Lidar extrinsic parameter calibration

Sensor fusion is one of the main challenges in self-driving vehicle and robotics applications. In this chapter we propose an automatic, online and target-less camera-Lidar extrinsic calibration approach. We adopt a structure from motion (SfM) method to generate 3D point clouds from the camera data which can be matched to the Lidar point clouds, thus we address the extrinsic calibration problem as a registration task in the 3D domain. The core step of the approach is a two-stage transformation estimation: first we introduce an object level coarse alignment algorithm operating in the Hough space to transform the SfM based and the Lidar point clouds into a common coordinate system. Thereafter we apply a control point based nonrigid transformation refinement step to register the point clouds more precisely. Finally, we calculate the correspondences between the 3D Lidar points and the pixels in the 2D camera domain. We evaluated the method in various real life traffic scenarios in Budapest, Hungary. The results show that our proposed extrinsic calibration approach is able to provide accurate and robust parameter settings on-the-fly.

DOI:10.15774/PPKE.ITK.2020.010

4.1 Introduction

Nowadays, state-of-the-art autonomous systems rely on wide range of sensors for environment perception such as *optical cameras*, *radars* and *Lidars*, therefore efficient sensor fusion is a highly focused research topic in the fields of self driving vehicles and robotics. Though the resolution and the operation speed of these sensors have significantly improved in the recent years, and their prices have become affordable in mass production, their measurements have highly diverse characteristics, which makes the efficient exploitation of the multimodal data challenging.

4.1.1 Problem statement

While real-time Lidars, such as Velodye's rotating multi-beam (RMB) sensors provide accurate 3D geometric information with relatively low vertical resolution, optical cameras capture high resolution and high quality image sequences enabling to perceive low level details from the scene. A common problem with optical cameras is that lighting conditions (dark, strong sunlight) largely influence the captured image data, while Lidars are able to provide reliable information less depending on external illumination and weather conditions. On the other hand, by simultaneous utilization of Lidar and camera sensors, accurate depth with detailed texture and color information can be obtained in parallel from the scenes.

Accurate Lidar and camera calibration is an essential step to implement robust data fusion, thus, related issues are extensively studied in the literature [69, 70, 71]. Existing calibration techniques can be grouped based on a variety of aspects [69]: based on the level of user interaction they can be semi- or fully automatic, methodologically we can distinguish target-based and target-less approaches, and in the term of operational requirements offline and online approaches can be defined (see Sec. 4.2).

4.1.2 Sensors discussed in this chapter

In this chapter, we focus on the measurements of the Velodyne HDL64E RMB Lidar sensor and a FLIR Blackfly USB3 camera with Fujinon 15mm-50mm lens. We

DOI:10.15774/PPKE.ITK.2020.010

have introduced the Velodyne Lidar sensor for localization of self-driving vehicle in Chapter 3 and as we mentioned it provides relatively sparse, inhomogeneous, but very accurate 3D measurements from its environment. The proposed camera is able to provide an image stream about $25 - 30$ frames/sec recording speed assuming 1288×964 resolution.

4.1.3 Aim of the chapter

In this chapter we propose a new fully automatic and target-less extrinsic calibration approach between a camera and a rotating multi-beam (RMB) Lidar mounted on a moving car. Our new method consists of two main steps: an object level matching algorithm performing a coarse alignment of the camera and Lidar data, and a fine alignment step which implements a control point based point level registration refinement. Our method relies on only the raw camera and Lidar sensor streams without using any external Global Navigation Satellite System (GNSS) or Inertial Measurement Unit (IMU) sensors. Moreover, it is able to automatically calculate the extrinsic calibration parameters between the Lidar and camera sensors on-the-fly which means we only have to mount the sensors on the top of the vehicle and start driving in a typical urban environment. In the object level coarse alignment stage we first obtain a synthesized 3D environment model from the consecutive camera images using a Structure from Motion (SfM) pipeline then we extract object candidates from both the generated model and the Lidar point cloud. Since the density of the Lidar point cloud quickly decreases as a function of the distance from the sensor we only consider keypoints extracted from robustly observable landmark objects for registration. In the first stage, the object based coarse alignment step searches for a rigid-body transformation, assuming that both the Lidar and the SfM point clouds accurately reflect the observed 3D scene. However in practice various mismatches and inaccurate scaling effects may occur during the SfM process, furthermore due to the movement of the scanning platform ellipsoid-shape distortions can also appear in the Lidar point cloud. In the second stage, in order to compensate for these distortions of the point clouds, we fit a Non-uniform rational B-spline (NURBS) curve to the

DOI:10.15774/PPKE.ITK.2020.010

extracted registration landmark keypoints, which step enables to flexibly form the global shape of the point clouds.

The outline of the chapter is the following: In Sec. 2 we give a detailed insight into the literature of camera-Lidar calibration, Sec. 3 introduces the proposed method and finally in Sec. 4 we quantitatively and qualitatively evaluate our fully automatic and target-less approach in real urban environment and we compare the performance of the proposed method against state-of-the-art target-based [70] and target-less calibration techniques [72, 73].

4.2 Related work

As mentioned above, extrinsic calibration approaches can be methodologically divided into two main categories: target-based and target-less methods.

As their main characteristics, target-based methods use special calibration targets such as 3D boxes [70], checkerboard patterns [74], a simple printed circle [75], or a unique polygonal planar board [76] during the calibration process. In the level of user interactions we can subdivide target-based methods into semi-automatic and fully-automatic techniques. *Semi-automatic* methods may consist of many manual steps, such as moving the calibration patterns in different positions, manually localizing the target objects both in the Lidar and in the camera frames, and adjusting the parameters of the calibration algorithms. Though semi-automatic methods may yield very accurate calibration, these approaches are very time consuming and the calibration results highly depend on the skills of the operators. Moreover, even a well calibrated system may periodically need re-calibration due to artifacts caused by vibration and sensor deformation effects.

Fully-automatic target-based methods attempt to automatically detect previously defined target objects, then they extract and match features without user intervention: Velas et al. [77] detect circular holes on planar targets, Park et al. [76] calibrate Lidar and camera by using white homogeneous target objects, Geiger et al. [74] use corner detectors on multiple checkerboards and Rodriguez et. al. [78] detect ellipse patterns automatically. Though the mentioned approaches do not need operator interactions, they still rely on the presence of calibration targets, which often should be arranged in complex setups (i.e. [74]

DOI:10.15774/PPKE.ITK.2020.010

uses 12 checkerboards). Furthermore during the calibration both the platform and the targets must be motionless.

On the contrary, *target-less* approaches rely on features extracted from the observed scene without using any calibration objects. Some of these methods use motion based [79, 80, 81] information to calibrate the Lidar and camera, while alternative techniques [69, 73] attempt to minimize the calibration errors using only static features.

Among motion based approaches, Huang and Stachniss [80] improve the accuracy of extrinsic calibration by the estimation of the motion errors, Shiu and Ahmad [79] approximate the relative motion parameters between the consecutive frames, and Shi et al. [82] calculate sensor motion by jointly minimizing the projection error between the Lidar and the camera residuals. These methods estimate first the trajectories of the camera and Lidar sensors either by visual odometry and scan matching techniques, or by exploiting IMU and GNSS measurements. Thereafter they match the recorded camera and Lidar measurement sequences assuming that the sensors are rigidly mounted to the platform. However, the accuracy of these techniques strongly depends on the performance of trajectory estimation, which may suffer from visually featureless (regions lacking structure and visual features), low resolution scans [61], lack of hardware trigger based synchronization between the camera and the Lidar [82], or urban scenes without sufficient GPS coverage.

We continue the discussion with single frame target-less and feature-based methods. Moghadam et al. [73] attempt to detect correspondences by extracting lines both from the 3D Lidar point cloud and the 2D image data. While this method proved to be efficient in indoor environments, it requires a large number of line correspondences, a condition which cannot be often satisfied in outdoor scenes. A mutual information based approach has been introduced in [83] to calibrate different range sensors with cameras. Pandey et al. [69] attempt to maximize the mutual information using the camera's grayscale pixel intensities and the Lidar reflectivity values. Based on Lidar reflectivity values and grayscale images Napier et al. [84] minimize the correlation error between the Lidar and the camera frames. Scaramuzza et al. [72] introduce a new data representation called

DOI:10.15774/PPKE.ITK.2020.010

the Bearing Angle image (BA) which is generated from the Lidar's range measurements. Using conventional image processing operations, the method searches for correspondences between the BA and the camera image. As a limitation, target-less feature based methods require a reasonable initial transformation estimation between the different sensors measurement [82], and mutual information based matching is sensitive to inhomogeneous point cloud inputs and illumination artifacts, which are frequently occurring problems when using RMB Lidars [69].

In this chapter, we propose two-stage fully automatic target-less camera-Lidar calibration method, which requires neither hardware trigger based sensor synchronization, nor accurate self-localization, trajectory estimation or simultaneous localization and mapping (SLAM) implementation. It can also be used by experimental platforms with ad-hoc sensor configurations, since after mounting the sensors to the car's roof top, all registration parameters are automatically obtained during driving. Failure of the SfM point cloud generation or SfM-Lidar point cloud matching steps in challenging scene segments does not ruin the process, as the estimation only concerns a few consecutive frames, and it can be repeated several times for parameter re-estimation.

Note that there exist a few end-to-end deep learning based camera and Lidar calibration methods [71, 85] in the literature, which can automatically estimate the calibration parameters within a bounded parameter range based on a sufficiently large training dataset. However, the trained models cannot be applied for arbitrary configurations, and re-training is often more resource intensive than applying a conventional calibration approach. In addition, the failure case analysis and analytic estimation of the limits of operations are highly challenging for black box deep learning approaches.

4.3 Proposed approach

Rotating multi-beam Lidar sensors such as Velodyne HDL64, VLP32 and VLP16 are able to capture point cloud streams in real time (up to 20 frames/sec.) providing accurate 3D geometric information (up to 100m) for autonomous vehicles, however, the spatial resolution of the measurement is quite limited and typical ring patterns appear in the obtained point clouds. While most of the

DOI:10.15774/PPKE.ITK.2020.010

online, target-less calibration approaches attempt to extract feature correspondences from the 2D image and the 3D point cloud data, such as various key points, lines and planes, we turn to a structural approach to eliminate the need for unreliable cross domain feature extraction and matching.

Our proposed approach is an automatic process consisting of a number of algorithmic steps for Lidar-camera sensor calibration, as presented in (Fig. 4.1). To avoid sensitive feature matching (2/3D interest points, line and planar segments), we propose a two-stage calibration method where first we use a Structure from Motion (SfM) [86] based approach to generate 3D point cloud from the consecutive image frames recorded by the moving vehicle (see Fig. 4.2). In such manner, the calibration task can be defined as a point cloud registration problem. Then, a robust object based coarse alignment method [10] is adopted to estimate an initial translation and rotation between the Lidar point cloud and the synthesized 3D SfM data. In this step connected point cloud components - called abstract objects - are extracted first, followed by the calculation of the best object level overlap between the Lidar and the synthesized SfM point clouds, based on the extracted object centers. A great advantage of the method is that although the number of the extracted object centers can be different in the two point clouds, the approach is still able to estimate a robust transformation. Following the coarse initial transformation estimation step we decrease the registration error using the Iterated Closest Point (ICP) point level method. Thereafter, to compensate the effects of the non-linear local distortions of the point clouds (SfM errors, and shape artifacts due to platform motion), we introduce a novel elastic registration refinement transform, which is based on non-uniform rational basis spline (NURBS) approximation (see details later). Finally, we approximate the $3D - 2D$ transformation between the Lidar points and the corresponding image pixels by an optimal linear homogeneous coordinate transform.

In our experiments, we observed a common problem that in SfM point clouds, dynamic objects such as vehicles and pedestrians often fall apart into several blobs due to several occlusions and artifacts during the SfM processing. This phenomenon significantly reduced the performance and robustness of the matching. To handle this issue we introduce a pre-processing step: before the coarse

DOI:10.15774/PPKE.ITK.2020.010

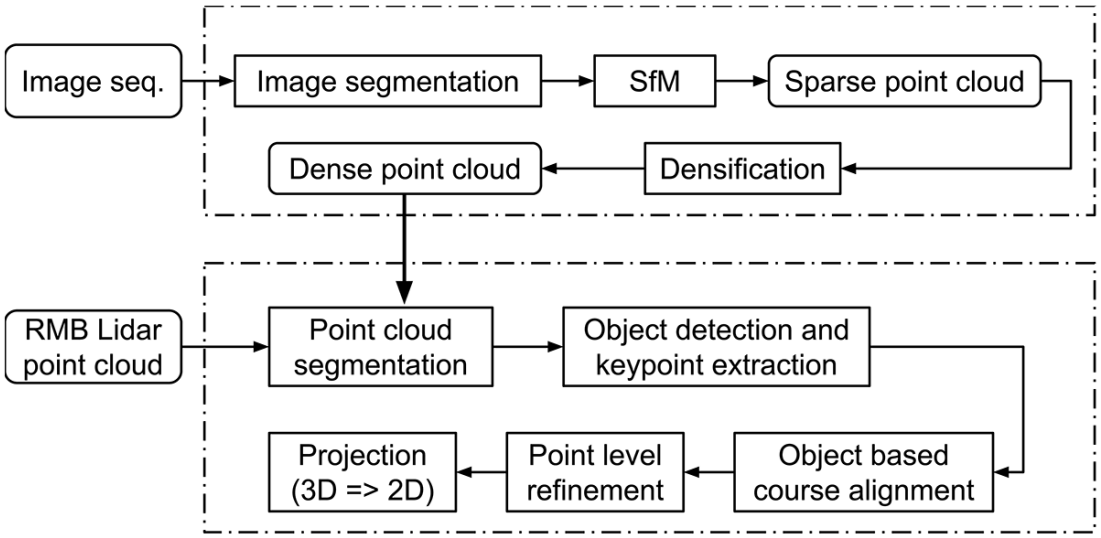


Figure 4.1: Workflow of the proposed approach.

alignment we eliminate the dynamic objects both from the Lidar and the synthesized SfM point cloud data by applying state-of-the-art object detectors: we use Mask R-CNN [27] to provide an instance level semantic segmentation on images, while the Point pillars method [25] detects dynamic objects in the Lidar point cloud.

As output, the proposed approach provides a 4×3 $\hat{T}$ matrix which represents an optimized linear homogeneous transform between the corresponding 3D Lidar points and the 2D image pixels. To generate $\hat{T}$, our algorithm calculates three matrices (T_1, T_2 and T_3) and a non-rigid transformation ($\mathfrak{T}_\star$): The first matrix, T_1 is calculated during the SfM point cloud synthesis, and it represents the (3D-2D) projection transformation between the synthesized SfM point cloud and the first image of the given image sequence which was used to create the SfM point cloud. Hence matrix T_1 can be used to project the synthesized 3D points to the corresponding pixel coordinates of the images. Matrix T_2 represents the coarse rigid transform (composed by a translation and rotation components) between the Lidar and SfM point clouds, estimated by the object level alignment step, while T_3 is the output of the ICP based registration refinement. The non-rigid body transformation $\mathfrak{T}_\star$ scales and deforms the local point cloud parts compensating

DOI:10.15774/PPKE.ITK.2020.010

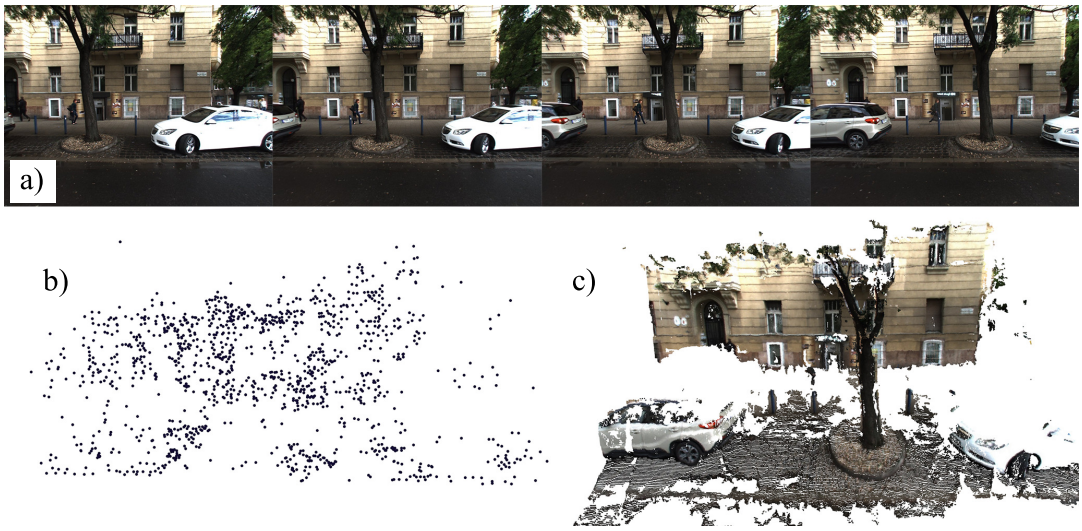


Figure 4.2: SfM point cloud generation (a) 4 from a set of 8 images to process. (b) Generated sparse point cloud (2041 points). (c) Densified point cloud (257796 points).

the distortion effects of the SfM and Lidar point cloud synthesis processes. Since $\mathcal{T}_\star$ cannot be defined in a closed form, we have to approximate the cascade of the obtained transforms T_1, T_2, T_3, $\mathcal{T}_\star$ with a single global homogeneous linear coordinate transform $\hat{T}$, using the *EPnP* [87] algorithm:

$$(T_1, T_2, T_3, \mathcal{T}_\star) \xrightarrow{\text{EPnP}} \hat{T}. \tag{4.1}$$

Note that we will also refer to the composition of the two 3D-3D rigid transform components as $T_R = T_2 \cdot T_3$. The steps of the proposed approach are detailed in the following subsections.

4.3.1 Structure from Motion (SfM)

As a first step, we generate a 3D synthesized point cloud from consecutive camera frames reinterpreting the calibration task as a point cloud registration problem. To generate the synthesized SfM point clouds we modified and extended the SfM pipeline [86] from the OpenMVG [1] library [88] which we describe in the following.

[1]Url: `https://github.com/openMVG/openMVG`

DOI:10.15774/PPKE.ITK.2020.010

To generate the synthesized point cloud we start capturing 1288×964 pixel sized image frames and we select a set of N ($N \geq 3$) images ($N = 8$ is used in this chapter). We include the current frame among the N frames to be processed if there exists a global movement of at least $th_{move} \geq 10$ pixels between consecutive frames. After gathering the N frames, we execute the following steps:

1. Image rectification: First we rectify the selected images (Fig. 4.2(a)) using the intrinsic camera parameters.

2. Semantic segmentation: Using a state-of-the-art instance level segmentation approach called Mask R-CNN [27] we label the dynamic objects (Fig. 4.3(a)) such as vehicles and pedestrians on the rectified frames.

3. Consecutive frame correspondences: L_2 fast cascade matching [104] is used to match the extracted SIFT feature points between the consecutive frames.

4. Structure from motion calculation: In this step we perform a SfM pipeline to generate a sparse point cloud (Fig. 4.2(b)) from the selected image sequence and we also store the 2D image pixel coordinates (Fig. 4.4) from all images that contributed to the calculation of the current 3D points. We assign unique IDs to all 2D interest points in all images and propagate the point IDs through the SfM pipeline along with the associated Mark R-CNN semantic segmentation labels into the generated point cloud. Thus, we will know which 2D points from which frames contributed to the calculation of each 3D point in the produced point cloud. (Fig. 4.3(b)).

5. Transformation calculation: Based on point density we select M points (in this chapter $M = 45$ and the point labels cannot be dynamic objects) from each processed frame and using the stored 3D-2D point correspondences (Fig. 4.4 (a-b)). We apply the *EPnP* [87] algorithm to estimate a transformation matrix T_1 of eq. (4.1) which is able to project the 3D points onto the corresponding 2D image pixels. Note that different T_1 matrices belong to every frame of the N images used to create the SfM point cloud. In practice, we calculate T_1 only for a single frame, selected via time stamp comparison with the actual Lidar point cloud. To validate if the estimated

DOI:10.15774/PPKE.ITK.2020.010

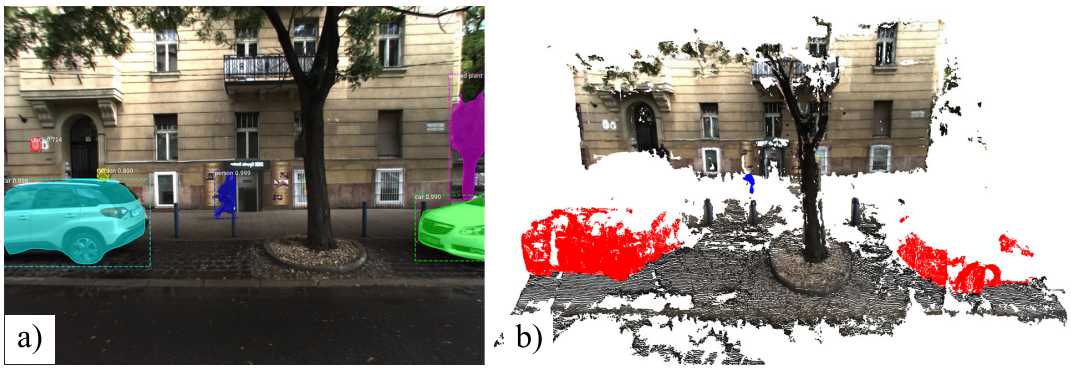


Figure 4.3: (a) Mask R-CNN [27] based instance level semantic segmentation. (b) Marking dynamic objects in the 3D SfM point clouds based on the 2D semantic segmentation (red: vehicle, blue: pedestrian)

projection matrix is correct, we re-project the 3D points to the corresponding images (Fig. 4.4 (c)) and we calculate the re-projection error based on the stored 3D-2D point associations. Using our method we measured 1.02 pixel error in average which we determined to be enough empirically to achieve a robust calibration result.

6. Sparse point cloud densification: For robust 3D object detection we have to densify the sparse point cloud because it contains only the tracked feature points without reflecting the structure of the objects. Fig. 4.2(c) demonstrates the result of the densification step by using the *PatchMatch* [89] algorithm from OpenMVS [1] library without any modification. We use the densified point cloud (Fig. 4.2(c)) and the estimated projection matrix in the next steps of the proposed approach.

Since our modified SfM pipeline can be performed on-the-fly, during the data capturing process we can periodically calculate the synthesized dense SfM point cloud from selected N consecutive frames. Thus the method can efficiently operate on experimental test platforms, where the movements of the vehicle may cause some displacements of the temporarily fixed sensors. Note that even industrial sensor configurations may need re-calibration at specific time intervals, which is also straightforward using our technique, since to perform our whole

[1]Url: `http://cdcseacave.github.io/openMVS`

DOI:10.15774/PPKE.ITK.2020.010

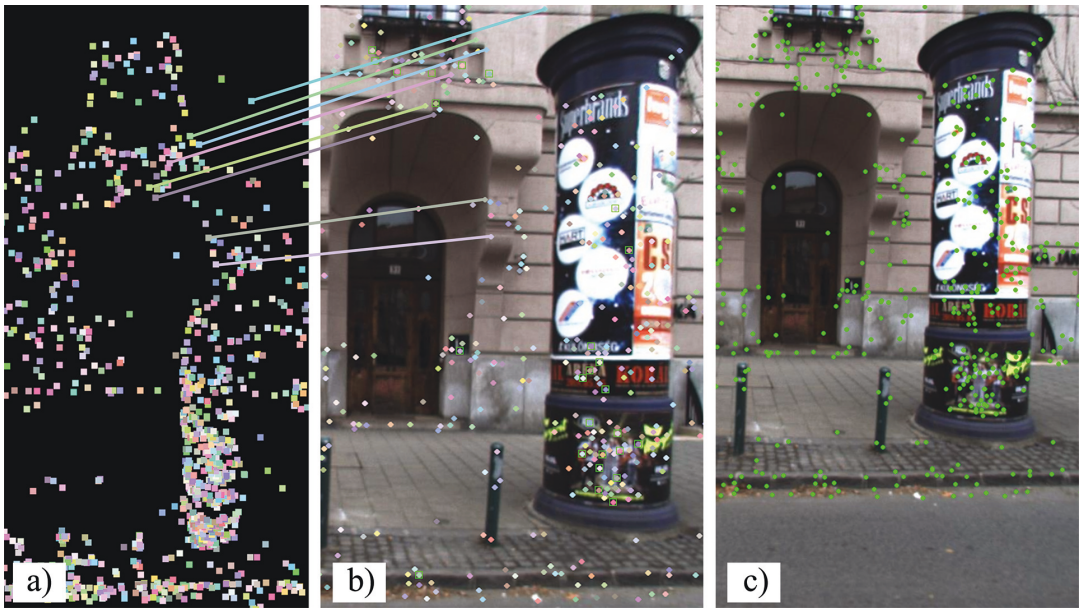


Figure 4.4: (a) Sparse cloud with each point assigned a unique color. (b) One frame showing color coded 2D points that contribute to the 3D point with the same color in (a) - also showing example correspondences. (c) Re-projection error

pipeline takes about 2-3 minutes (using an Intel Core i7 Coffee Lake processor), and it is not required to stop driving or using any calibration targets.

4.3.2 Point cloud registration

In the next phase of the proposed algorithm, we aim to transform the Lidar point cloud and the previously synthesized SfM point cloud (Sec. 4.3.1) to the same coordinate system.

Several point level registration methods can be found in the literature. Many of them are variants or extensions of the classical iterative ICP or the Normal Distributions Transform (NDT) [39] methods, which often fail if the initial translation is large between the point clouds. In addition, they are usually inefficient when the density characteristics of the two point clouds are quite different, furthermore the typical ring patterns of RMB Lidar data may mislead the registration process by finding irrelevant correspondences on the ground level instead of finding associations between the foreground regions [61]. In order to avoid such

DOI:10.15774/PPKE.ITK.2020.010

registration artifacts we proposed an object level point cloud alignment approach [10] to estimate an initial translation and rotation between the point clouds.

We experienced, that since the proposed SfM (Sec. 4.3.1) pipeline operates on image sequences of $N = 8$ frames, the global diameter of the generated SfM model is usually much smaller than the diameter of the recorded RMB Lidar point cloud. Therefore, to facilitate the registration, in the further processing steps we only consider the central segment of the Lidar measurement with a 15 meter radius around the sensor position.

4.3.2.1 Object detection

Instead of extracting local feature correspondences from the global scene, we detect and match various landmark objects in the SfM based and the Lidar point clouds. In the next step based on the landmark object locations we estimate the transform which provides the best overlap between the two object sets.

As a preprocessing step, we first detect the dynamic object regions which can mislead the alignment, thus it is preferred to remove their regions before the matching step. On the one hand, based on the predicted semantic labels in the camera images (Sec. 4.3.1), we determine the camera based 3D locations of the *vehicles* and *pedestrians* by projecting the 2D labels to the synthesized 3D SfM point cloud (see Fig 4.3). On the other hand, to eliminate the dynamic objects from the Lidar point cloud we adopt a state-of-the-art object detector called Point pillars [25] (see Fig. 4.6(b)), which takes the full point cloud as input and provides bounding box candidates for vehicles and pedestrians. As a result of the filtering, during the subsequent alignment estimation we can rely on more compact and robustly detectable static scene objects such as *columns*, *tree trunks* and different *street furniture* elements.

After the dynamic object filtering step we also remove the ground/road regions by applying a spatially adaptive terrain filter [61], then we extract connected component blobs (objects) from both the synthesized SfM and the Lidar point clouds.

Euclidean point distance based clustering methods are often used to extract object candidates from point clouds, for example [68] used kD-tree and Octree

DOI:10.15774/PPKE.ITK.2020.010

space partition structures to efficiently find neighboring points during the clustering process. However these methods often yield invalid objects, either by merging nearby entities to large blobs, or by over-segmenting the scene and cutting the objects into several parts. Furthermore, building the partition tree data structure frame by frame causes noticeable computational overload. Börcs et al. [15] proposed an efficient 2D pillar-structure to robustly extract object candidates using a standard connected component algorithm on a two level hierarchical 2D occupancy grid, however that method erroneously merges multiple objects above the same 2D cell even if they have different elevation values (such as hanging tree crowns).

To handle the merging problem of the 2D grid based approaches alongside the height dimension we propose an efficient sparse voxel structure (SVS) to extract connected components - called abstract objects - from the point cloud data (Fig. 4.5 (d-e)). While using only the pillar structure – which is similar to a traditional 2D grid approach [15] – the objects may be merged alongside the height dimension (see Fig. 4.5(d)), in a second step focusing on the vertical point distribution we can split the erroneously merged regions using the voxel level representation of the SVS (see Fig. 4.5(e)).

The most elementary building block of the SVS is the voxel (see Fig. 4.5(a)) which is a cube volume of the 3D space containing a local part of the given point cloud. Furthermore the vertically aligned voxels form a so called pillar structure (see Fig. 4.5(b)) which is an extension of the voxels to be able to operate as a simple 2D grid similarly to [15]. Creating a fully dense voxel model based on the bounding box dimensions of the point cloud is quite inefficient, since in a typical urban environment only a few segments of the bounding space contain 3D points. To ensure memory efficiency for our SVS, we only create a voxel if we find any data point in the corresponding volume segment. In our case, the average size of the point cloud bounding boxes is around $30 \times 30 \times 5$ meter, and we found that the optimal voxel resolution for object separation in a typical urban environment is 0.2 meter [15]. While in this case a full voxel model would require the management of around $562,500$ voxels, our proposed SVS usually contains less than $30,000$ voxels, which yields a great memory and computational gain, while efficient neighborhood search can be maintained with dynamically created

DOI:10.15774/PPKE.ITK.2020.010

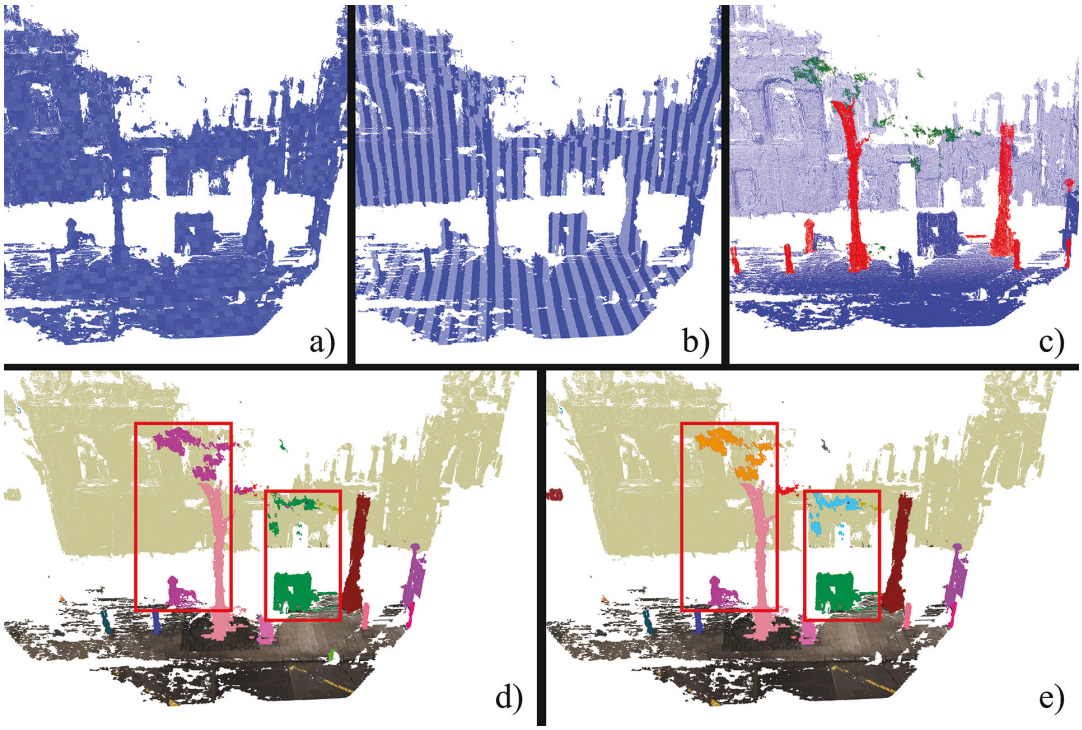


Figure 4.5: (a) Voxel representation of the proposed SVS. (b) Pillars representation of the proposed SVS. (c) Linearity and flatness features computed based on eigen value analysis. Blue: flat voxels, red: high linearity, green: scattered region. (d) Extracted objects from the point cloud using pillar representation. (e) Extracted objects from the point cloud using voxel representation.

DOI:10.15774/PPKE.ITK.2020.010

links within the new data structure. Note that we can find alternative sparse voxel structures in the literature such as [90, 91], however those methods are based on *Octree* structures, and they were designed for computer graphics related tasks such as ray tracing. On the other hand our SVS fully exploits the fact that in an urban environment most of the objects are horizontally distributed along the road, while vertical object occlusions are significantly less frequent.

Next we describe the object extraction process using our SVS. *First*, similarly to [15] we detect the ground regions using a spatially adaptive terrain filtering technique. *Second*, using the pillars structure of the SVS, we can quickly extract connected components (objects) by merging the neighboring pillars into object candidates. *Third*, considering the voxel level resolution of the SVS, we separate the erroneously merged blobs along the height dimension.

After the extraction of abstract objects, we assign different attributes to them, which will constrain the subsequent object matching steps (Sec. 4.3.2.2). We calculate the following three attributes for each object blob O:

1. *centroid* ($O.c$): The center of mass of the local point cloud belonging to the object

2. *bounding box* ($O.b$): *min*imal and *max*imal point coordinate values along each dimension.

3. *shape category* ($O.s$): we assign to each object the shape category *linear*, *flat* or *scattered* object (see Fig. 4.5(c))

The assigned *shape category* is based on a preliminary voxel level segmentation of the scene. For each voxel, we determine a *linearity* and a *flatness* descriptor, by calculating the principal components and the eigenvalues ($\lambda_1 \geq \lambda_2 \geq \lambda_3$) of the point set in the voxel, using the Principal Components Analysis (PCA) algorithm. By examining the variance in each direction, we can determine the linearity and flatness properties as follows [92]:

$$(1)\ Linearity = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

$$(2)\ Flatness = 1 - \frac{\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3}$$

DOI:10.15774/PPKE.ITK.2020.010

1. If λ_1 is large ($\lambda_1 > 0.7$) compared to the other two eigenvalues, it means that the included points lie on a line, so the voxel is part of a linear structure such as pole or wire.

2. If the total variance is defined by the two largest eigenvalues that means points lie on a plane.

3. If $Flatness > 0.6$ the voxel is classified as flat voxel. Typically wall segments and parts of larger objects are built from flat voxels.

4. Unstructured regions can also be detected using the third eigenvalues. If $\lambda_3 > 0.1$ that means point scattering is high in the region. Vegetation and noise can be detected efficiently using this property.

Note that the above threshold values should be empirically defined, since they depend on the characteristics of the given point cloud. Following the guidelines of [92], we fine-tuned the threshold parameters based on our experiments. Fig. 4.5(c) demonstrates the voxel level classification based on the linearity and flatness values. Finally, for each extracted abstract object we determine its shape category by a majority voting of its included voxels.

4.3.2.2 Object based alignment

At this point, our aim is to estimate a sufficient initial alignment between the synthesized SfM and Lidar point clouds based on the above detected abstract objects. In order to find the optimal matching, we use the extracted two sets of object *centroids* from the SfM based and the Lidar point clouds, respectively, and we search for an optimal rigid transform, T_2 of eq. (4.1), between the two point sets by an iterative voting process in the Hough space [10], which was described in details in Chapter 3. We observed that the search for the translation component of the transformation can be limited to a 15m radius search area (which is equal to the base size of the considered point cloud segments), while it is also crucial to roughly estimate the rotation component around the *upwards* axis. On the other hand, the other two rotation components are negligible, as the road elevation is usually quite short within the local road segments covered by the considered point

DOI:10.15774/PPKE.ITK.2020.010

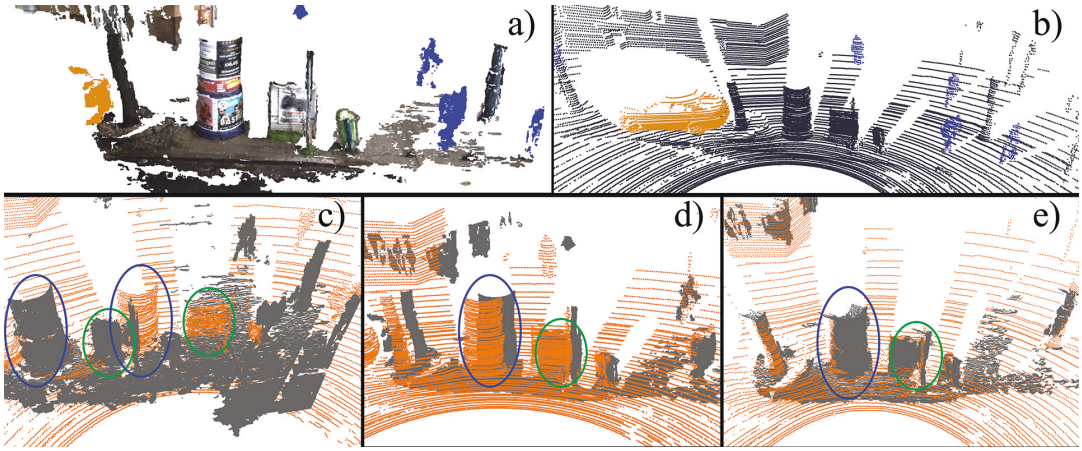


Figure 4.6: Demonstration of the object-based coarse alignment step (a) Mask R-CNN [27] based dynamic object filtering. (b) Point pillars [25] based dynamic object filtering. (c) Initial translation between the synthesized (dark grey) and the Lidar (orange) point cloud. (d) Object based coarse alignment without dynamic object removal (e) Proposed object based alignment after removing dynamic objects. Identically colored ellipses mark the corresponding objects.

clouds, thus the minor rotation errors caused by ignoring them can be corrected later through point level transformation refinement.

Estimation of the proper 3D translation vector (dx, dy, dz) among the three coordinate axes and the α rotation value around the *upwards* axis is equivalent to finding an optimal 3D rigid body transformation as we described in Chapter 3.3.3.

The transformation estimation can be defined as a finite search problem by discretization of the possible parameter space into equal bins. Considering that our transformation has four degrees of freedom (3D translation vector and a rotation component around the *upwards* axis) we allocate a 4D accumulator array $A[\alpha, d_x, d_y, d_z]$.

Let us denote the abstract object sets extracted from the SfM and the Lidar point clouds by $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively. During the assignment, we attempt to match all possible $O_1 \in \mathcal{O}_1$ and $O_2 \in \mathcal{O}_2$ object pairs, however to speed up the process and increase its robustness we also introduce a compatibility measurement between the objects. As discussed in Sec. 4.3.2.1, during the object detection step we assigned a *bounding box* $(O.b)$ and a *shape category* $(O.s)$ parameter to

DOI:10.15774/PPKE.ITK.2020.010

Algorithm 1 Object based point cloud alignment. Input: two point clouds $F1$ and $F2$, output: the estimated $TR = T_R$ rigid transformation between them. $Rot(\alpha)$ denotes a rotation matrix around the *upwards* axis.

procedure ALIGNMENT($F1$, $F2$, TR)
  $\mathcal{O}_1 \leftarrow ObjectDetect(F1)$
  $\mathcal{O}_2 \leftarrow ObjectDetect(F2)$
  Initialize 4D accumulator array A
  for all $O_1 \in \mathcal{O}_1$ **do**
    for all $O_2 \in \mathcal{O}_2$ **do**
      if $iscompatible(O_1, O_2)$ **then**
        for $\alpha \in [0, 359]$ **do**
          $O_2'.c \leftarrow Rot(\alpha) * O_2.c$
          $(dx, dy, dz) \leftarrow O_1.c - O_2'.c$
          $A[\alpha, dx, dy, dz] \leftarrow A[\alpha, dx, dy, dz] + 1$
        end for
      end if
    end for
  end for
  $\alpha, dx, dy \leftarrow FindMaximum(A)$
  $F1, T2 \leftarrow TransformCloud(F1, \alpha, dx, dy, dz)$
  $F1, T3 \leftarrow ICP(F1, F2)$
  $TR \leftarrow T3 * T2$
end procedure

DOI:10.15774/PPKE.ITK.2020.010

each extracted abstract object. We say that $O_1 \in \mathcal{O}_1$ and $O_2 \in \mathcal{O}_2$ are compatible, if the side length ratios of their bounding boxes are between predefined values ($0.7 \leq \frac{O_1.b_x}{O_2.b_x}, \frac{O_1.b_y}{O_2.b_y}, \frac{O_1.b_z}{O_2.b_z} \leq 1.4$) and their shape categories are the same ($O_1.s = O_2.s$).

Alg. 1 shows the pseudo code of the proposed object based alignment algorithm. The approach iterates through all the compatible $O_1 \in \mathcal{O}_1$ and $O_2 \in \mathcal{O}_2$ object pairs and it rotates O_2 with all the possible α values. We calculate the Euclidean distance between the rotated $O_2'.c$ and the $O_1.c$ centroid points in the same way as described in Chapter 3.3.3.3:

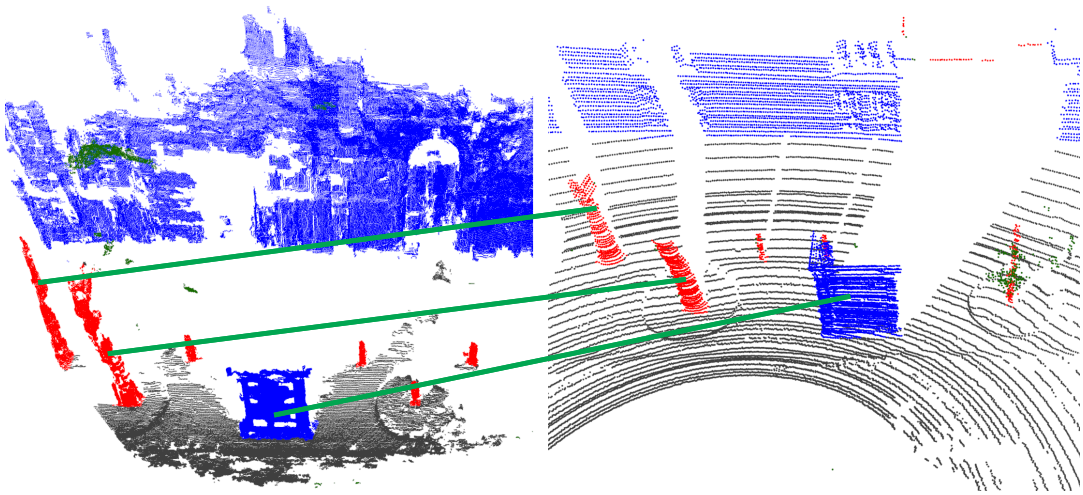
$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = O_1.c - \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} O_2.c \tag{4.2}$$

During a given iteration step the method addresses the accumulator array with the actual α, it calculates the corresponding dx, dy and dz translation components using eq. (4.2), then it increases the evidence of the transformation defined by the (α, dx, dy, dz) quadruple. At the end of the iterative process, the maximal vote value in the accumulator array determines the best transformation parameters i.e. the best α rotation and the dx, dy and dz translation components. Thereafter, using the estimated transformation matrix the Lidar point cloud is transformed to the coordinate system of the synthesized SfM model.

The above point cloud registration process is demonstrated in Fig. 4.6. Note that as we mentioned in the beginning of Sec. 4.3.2.1, we aimed to restrict the object matching step to static landmark objects, therefore we first removed the vehicle and pedestrian regions from the 3D scene (see Fig. 4.6(a) and (b)). As Fig. 4.6(c)-(e) confirm, this step may have a direct impact on the quality of the object based alignment algorithm: subfigure (c) shows the initial alignment of the Lidar and the SfM point clouds with a large translation error, subfigure (d) shows the result of the proposed coarse alignment algorithm without eliminating the dynamic objects from the point clouds in advance, while subfigure (e) demonstrates the output of the proposed complete algorithm which provides a significantly improved result.

To eliminate the discretization error of the object based alignment we can refine the estimation of the rigid transformation using a standard ICP [39] method

DOI:10.15774/PPKE.ITK.2020.010



a) SfM point cloud b) Lidar point cloud

Figure 4.7: Landmark correspondences between the detected objects in the SfM and Lidar point clouds. Assignment estimation is constrained by the computed Linearity and Flatness values. Red color represents linear voxels, blue color denotes flat structures and green marks noisy unstructured regions.

which yields a T_3 matrix of eq. (4.1), so that the unified rigid transform between the SfM and and Lidar point clouds is represented as $T_R = T_2 \cdot T_3$.

4.3.3 Control curve based refinement

In the previous section, we estimated an optimal rigid transformation $T_R = T_2 \cdot T_3$ between the synthesized SfM model and the Lidar point cloud, which is composed of a global translation and a rotational component. However, during the SfM pipeline various artifacts may occur such as *deformations* due to *invalid depth calculation*, effects which can significantly distort the geometry of the observed 3D scene. On the other hand, due to the vehicle motion, the Lidar point clouds may also suffer from shape distortion as detailed in the Introduction. To handle the local point cloud deformation problems during the point cloud registration process, we propose a control curve based refinement method, that calculates the non-rigid $\mathcal{T}_\star$ transform component of eq. (4.1) (see Fig. 4.8).

To estimate the local shape deformations of the point clouds, we use the

DOI:10.15774/PPKE.ITK.2020.010

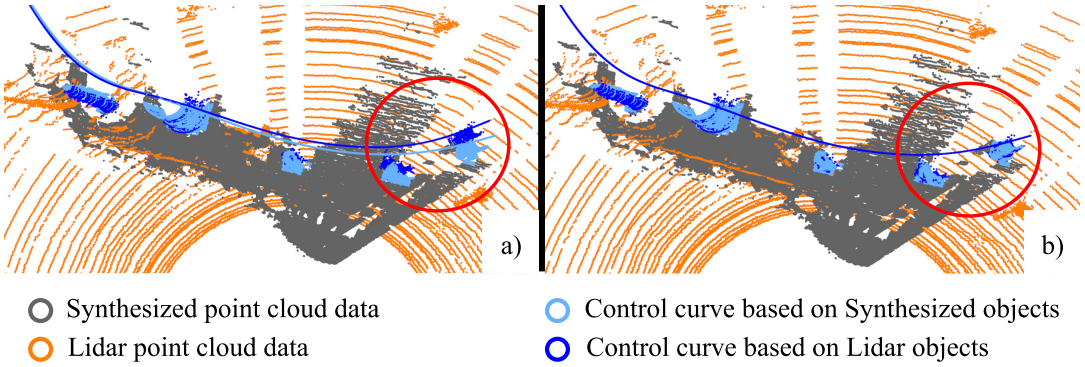


Figure 4.8: Demonstration of the control curve based registration refinement step. Image (a): output of rigid body transform based registration (T_R) with local registration artifacts; Image (b) corrections via registration refinement ($\mathcal{T}_\star$)

extracted object centroids as control points, and we fit separate NURBS curves to the control points of the SfM and the Lidar point clouds, respectively. As shown in Fig. 4.8(a), after the rigid body transformation based alignment there are point cloud regions which fit quite well, while in other (distorted) segments we observe significant local alignment errors.

For this reason we introduce a non-rigid curve based point cloud registration refinement step (see Fig. 4.9):

1. Using the extracted object centers as control points, we fit NURBS curves to the SfM and the Lidar point clouds. In Fig. 4.9 red dots mark the *control points* and bluish curves demonstrate the *NURBS curves*. Based on the assignment results of the *object based alignment step* (Sec.4.3.2.2), we can utilize the correspondences between the control points to obtain the coherent objects pairs.

2. Let us assume that based on the object alignment, the curve segments between O_1 and O_2 belong together. Between each neighboring control point pair we subdivide the curve segment into equal bins, so that we assign the first bin of the Lidar curve segment to the first bin of the SfM curve segment.

3. In the next step we assign each point of the Lidar point cloud to the closest bin of the Lidar based NURBS curve.

DOI:10.15774/PPKE.ITK.2020.010

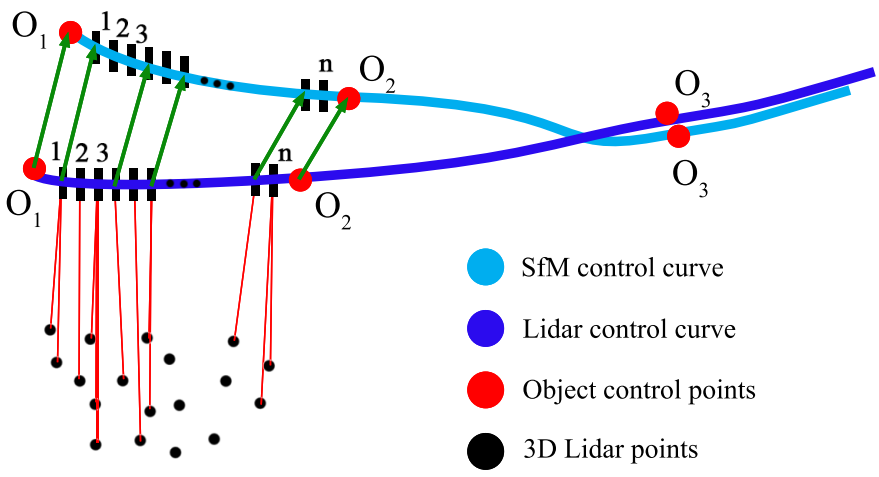


Figure 4.9: Demonstration of the curve based refinement step.

4. Taking the advantage that a NURBS curve is locally controllable, we can transform the local parts of the Lidar point cloud separately. We calculate a translation between each corresponding bins of the SfM and Lidar based curve segments by subtracting the coordinates of the Lidar bins from the corresponding SfM bins. Since we assigned each Lidar point to one of the curve bins, we can transform these points using the translation vector of the given curve segment, which points from the given Lidar bin to the corresponding SfM bin. Since each part of the Lidar point cloud is transformed according to the corresponding curve segment, this process implements a non-rigid body transformation.

Fig. 4.8 (b) show the results of the curve based registration refinement step, demonstrating a significant improvement in the initially misaligned point cloud segments.

4.3.4 3D point projection calculation

Our final goal is to estimate a transformation $\hat{T}$ which is able to project the 3D points from the original coordinate system of the Lidar sensor directly to the

DOI:10.15774/PPKE.ITK.2020.010

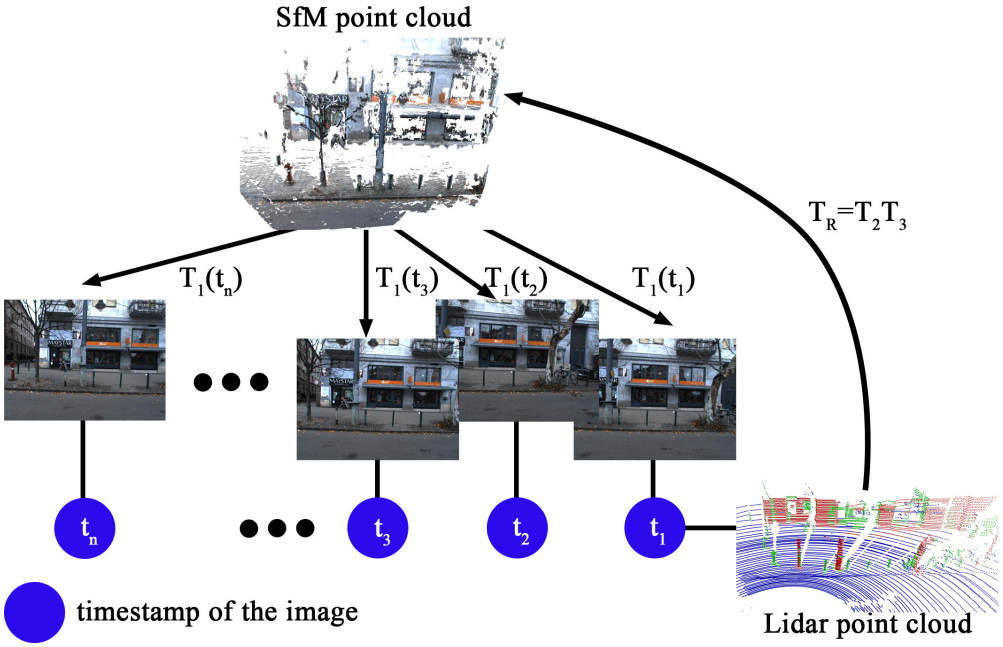


Figure 4.10: Workflow of the projection calculation process.

corresponding 2D image pixels. Fig. 4.10 demonstrates the proposed workflow of the point cloud projection. From $N = 8$ consecutive image frames using an SfM pipeline we estimate a dense SfM point cloud, and we also calculate and store a unique T_1 projection transformation between the generated dense SFM point cloud and the original image frames. Thereafter we take the Lidar point cloud with the closest time stamp to the first frame of the actual N-frame sequence and we estimate a transformation which is able to transform the Lidar point cloud to the coordinate system of the SfM point cloud. The estimated transformation consists of a rigid body component composed by an object based (T_2) and a point based (ICP) term (T_3), as well as a NURBS-based non-rigid transform component ($\mathcal{T}_\star$).

Since the non-rigid body transform $\mathcal{T}_\star$ cannot be defined in a closed form as a simple matrix multiplication, we calculate an approximate direct transformation between the original Lidar point cloud domain and the image pixels for preserving computational efficiency of the data fusion step. For this reason, with applying the above estimated transformation components (T_1, T_2, T_3 and $\mathcal{T}_\star$) we project

DOI:10.15774/PPKE.ITK.2020.010

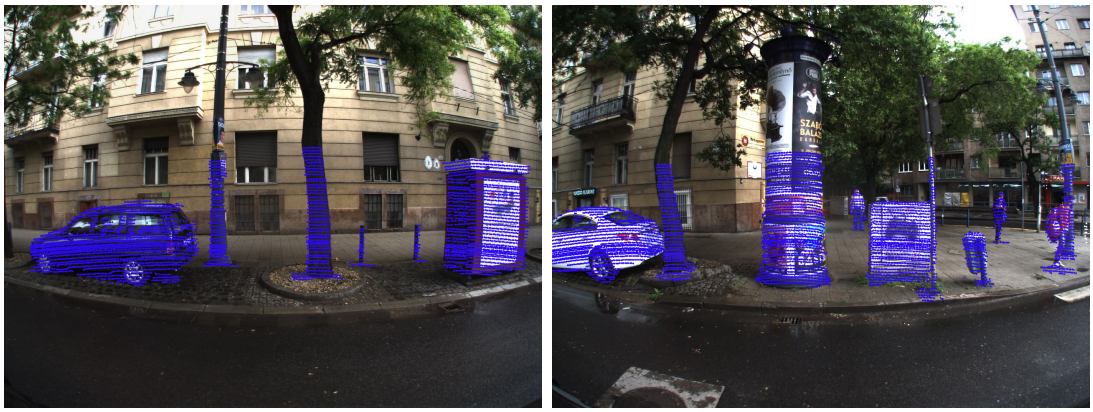


Figure 4.11: Qualitative point cloud projection results onto the image domain based on the proposed fully automatic, target-less camera-Lidar calibration approach. Blue color denotes the projected 3D points which belong to the object candidates detected during the course alignment process.

the centroid points of the extracted Lidar objects onto the camera images and we store the 3D-2D correspondences. In the next step, upon the available 3D-2D point correspondences we estimate a single homogeneous linear transformation $\hat{T}$ which can be used to project the raw Lidar points directly to the image domain (see Fig. 4.11). $\hat{T}$ is calculated by the OpenCV *solvePnP* function, which is based on the Levenberg-Marquardt optimization method [93]. As a final result, Fig. 4.11 shows the projection of the Lidar data to the image domain.

4.4 Experiments and Results

To evaluate our proposed *target-less* and *fully automatic* self-calibration method we compared it to two state-of-the-art target-less techniques [73, 72], and an *offline target-based* method [70]. We quantitatively evaluated the considered methods by measuring the magnitude and standard deviation of the pixel level projection error values both in the x and y directions along the image axes (Table 4.1).

We collected test data from overall *10* km long road segments in different urban scenes such as *boulevards*, *main roads*, *narrow streets* and *large crossroads*. To take into account the daily traffic changes, we recorded measurements both in

DOI:10.15774/PPKE.ITK.2020.010

rush hours in heavy traffic, and outside the peak hours as well. Since using the rotating multi-beam Lidar technology, the speed of the moving platform influences the shape of the recorded point cloud, we separately evaluated the results for measurements captured by *slow* and *fast* vehicle motion, respectively. Therefore we defined two test sets: set *Slow* contains sensor data captured at speed level between *5-30* km/h, while set *Fast* includes test data captured at speed above *30* km/h.

Figure 4.13 and 4.14 visually demonstrate the steps of the proposed workflow.

4.4.1 Quantitative evaluation and comparison

We measured the projection error of the proposed and the reference methods by projecting the 3D Lidar points onto the 2D image pixels, which was followed by visual verification. Before starting the test, we carefully calibrated the Lidar and camera sensors using the offline target-based reference method [70], which served as a baseline calibration for the whole experiment. During our *10* km long test drive, we chose *200* different key frames of the recorded measurement sequence, and by each key frame - based on the actual a Lidar point cloud and $N = 8$ consecutive camera images - we executed new calibration processes in parallel by our proposed approach and by the remaining automatic reference techniques [73, 72]. We paid attention to collect these key frames from diverse scenarios (*main roads*, *narrow streets*, etc.), and we also quantified the corresponding vehicle speed values to assign each key frame either to the *Slow* or to the *Fast* test set. We evaluated the performance of all considered methods by each key frame (i.e. after each re-calibration). To calculate the pixel projection error we manually selected *10* well defined feature points (e.g. corners) from different regions of the 3D point cloud, and using the calibration results of the different methods we projected these 3D feature points to the image domain. Finally we measured the pixel errors versus the true positions of the corresponding 2D feature points positions detected (and manually verified) on the images.

As shown in Table 4.1, both in the *Slow* and *Fast* test sets our proposed approach notably outperforms the two target-less reference methods [73, 72] in mean pixel error, and it also exhibits lower error deviation which indicates a more

DOI:10.15774/PPKE.ITK.2020.010

Set*	Method	x-error (pixel)		y-error (pixel)	
		Avg.	Dev.	Avg.	Dev.
Slow	Target-based ref.[70]	1.13	0.27	1.75	0.37
	Bearing angle [72]	4.56	2.15	4.58	1.74
	Line based [73]	3.36	1.45	3.47	0.98
	Proposed	2.58	0.73	2.82	0.88
Fast	Target-based ref. [70]	3.04	0.74	2.74	0.41
	Bearing angle [72]	4.87	2.46	4.42	1.91
	Line based [73]	5.01	1.93	4.01	1.49
	Proposed	2.84	0.95	3.14	0.82

Table 4.1: Performance comparison of the target-less proposed method with two state-of-the-art target-less reference methods and with a target-based (supervised) reference technique. Notes: *Test set names *Slow* and *Fast* refer to the speed of the data acquisition platform.

robust performance. Although in the *Slow* test set – in terms of accuracy – the proposed approach is slightly outperformed by the considered *offline target-based calibration* technique [70], it should be emphasized that the operational circumstances are quite different there. On one hand, according to our experiences the calibration of the camera-Lidar system using [70] takes more than two hours with several manual interaction steps, furthermore if the sensors are slightly displaced during the drive due to vibration, one must stop the measuring platform and re-calibrate the system offline. On the other hand, our proposed approach is able to calibrate the camera and Lidar on-the-fly and in a fully automatic way, even during driving without stopping the car.

We can also observe in Table 4.1, that in the *Fast* test set our proposed method can reach or even surpass the accuracy of the target-based method, since we can adaptively handle the vehicle speed dependent shape deformations of the recorded RMB Lidar point clouds. By increasing the speed of the scanning platform, we can observe that the shape of the obtained Lidar point cloud becomes distorted, and gets stretched in the direction of the movement. Since the offline, target-based method cannot re-calibrate the system during the drive, its accuracy may decline when the car accelerates. On the contrary, our proposed method is able to re-calibrate the system at different speed levels, and taking the advantage of the

DOI:10.15774/PPKE.ITK.2020.010

Step of our method	x-error (pixel)		y-error (pixel)	
	Avg.	Dev.	Avg.	Dev.
Object based alignment (OBA) on raw frames	4.78	2.26	5.21	2.47
OBA with preliminary dynamic object filtering	3.46	0.85	4.47	1.24
Final result also using ICP and NURBS-based refinement	2.58	0.73	2.83	0.88

Table 4.2: Performance evaluation of the proposed method at different calibration steps. First we measure the projection error using only the object based coarse alignment, then we extended the method with the dynamic object filtering based on Mask R-CNN and PointPillars techniques, and finally we measure the impact of the ICP and the proposed curve based refinement.

non-rigid transformation component, it can be better adapted to the non-linear shape distortions of the point cloud.

To quantify the improvements of the consecutive calibration steps of the proposed method, we also compared the pixel level projection errors at three different stages of our algorithm pipeline. In Table 4.2, the *first* row shows the results of using the *object based coarse alignment (OBA)* step on the raw Lidar frames; the *second* row refers to using OBA with preliminary *dynamic object filtering (DOF)*, finally the third row displays the error parameters obtained by the complete workflow including the *ICP and curve based refinement* steps. Since applying DOF can prevent us from considering several invalid object matches, we can see a large improvement in the robustness between the first and second stages, furthermore by exploiting the *point level ICP and curve based refinement* steps (third stage), the projection accuracy can be significantly upgraded.

4.4.2 Robustness analysis of the proposed method

Since our proposed method is able to re-calibrate the sensors periodically during the movement of the sensor platform, it is important to measure the robustness of the approach. We recall here that during the proposed algorithm pipeline, we already attempt to internally estimate the quality of the calibration: First following the SfM point cloud synthesis we re-project the generated point cloud to the original image pixels, and only continue the process if the re-projection error is smaller than 2 pixels. Second, following the estimation of both the rigid, and the NURBS-based non-rigid transformation components, we calculate the point

DOI:10.15774/PPKE.ITK.2020.010

level Hausdorff distance between the registered SfM and Lidar point clouds, and only accept the new calibration if the distance is lower than *5* centimeters. In summary, if these threshold-based constraints are not satisfied we reject to apply the current transformation estimation and we start a new calibration process.

In Sec. 4.4.1, we manually measured the errors based on 10 selected feature points in each Lidar frame. Due to the lack of availability of a Ground Truth (GT) transform [94], there is no straightforward way to extend this validation step for all points of the Lidar point cloud. However, we have performed an additional experiment to quantitatively measure the reliability and repeatability of the proposed approach. Although as discussed in Sec. 4.4.1, the target-based reference (TBR) method [70] itself may also suffer from 1-3 pixel registration errors, we observed that these errors can usually be considered constant when driving with an approximately standard average speed. Exploiting this fact, we used here TBR as a baseline technique, and we re-calibrated with 500 randomly selected seed frames the Lidar and the camera sensors using the proposed approach.

By each re-calibration step, we calculated the average point projection differences in pixels between our actual calibration and the TBR registration *considering all points of the Lidar point cloud*. Figure 4.12 shows the differences measured in pixels in the x and y directions over the subsequent calibration trials. Based on this experiment the proposed approach proved to be quite stable again, as the measured differences range between $1.2 - 1.8$ pixels in x direction, and $1.0 - 1.4$ pixels in y direction, without significant outlier values.

We also examined the impact of the roughness of discretization in the accumulator space in the voting process during the object based registration. If we increase the step size of the α rotation parameter around the *upward axis*, or if we divide the space of the possible translation components dx, dy and dz into larger bins, we loose accuracy, however the algorithm may show an increased robustness. On the contrary, if we choose a smaller rotation step size and we increase the resolution of the possible translation space we can reach more accurate results, however we must also expect an increasing number of unsuccessful registration attempts, as the maximum value in the voting array may become ambiguous. Based on our experiments we used an optimal rotation step size of 0.25 degrees for α, and an optimal resolution of 0.2 meter for the translation components dx,

DOI:10.15774/PPKE.ITK.2020.010

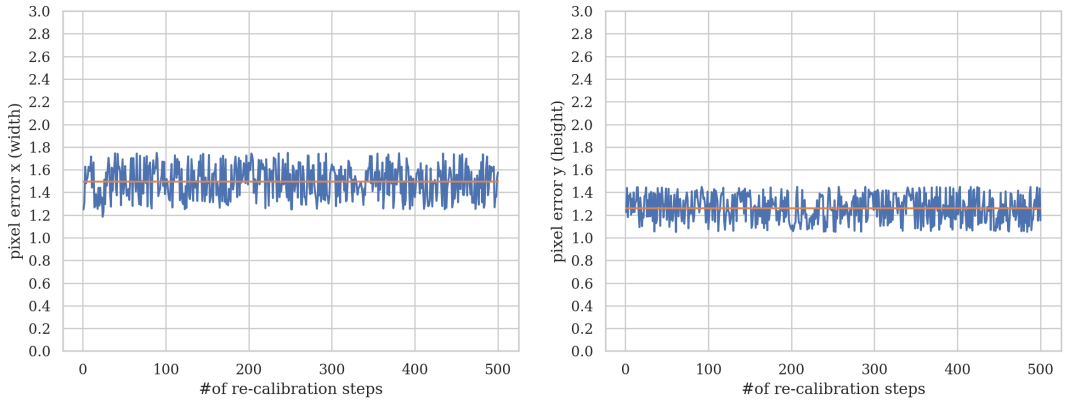


Figure 4.12: Robustness analysis of the proposed method. It shows the differences measured in pixels in the x and y directions over the subsequent calibration trials.

dy and dz. These experiments are consistent with our previous observations regarding an optimal cell resolution in the case of RMB Lidar sensors [14], keeping in mind that the accuracy can be enhanced in the subsequent point level ICP step.

There can be situations when the SfM pipeline cannot produce suitable synthesized point clouds. For example, the strong sunlight drives image intensities to saturation, and lead to a complete loss of image details limiting the traceable features during the SfM process. However the proposed approach is able to periodically repeat the calibration pipeline and we should only update the calibration when the current estimation improves the previously used one in terms of Hausdorff error between the SfM and Lidar point clouds. Since the object based alignment step of the proposed approach greatly depends on the landmark candidates (objects) used for registration, we preliminary analyze the scene using the computed linearity and flatness values and the extracted bounding boxes, and we only start the calibration pipeline if the number of the proper object candidates (column shaped objects such as traffic signs, tree trunks and poles) is more than three. Typically in main roads and larger crossroads the scenes contain several column shaped landmark objects, where the proposed self-calibration algorithm works more robustly and accurately. Based on our analysis if a scene is denoted as a good calibration location, the average rate of the proper landmark objects is between 20 and 40 percent. We note that static parking vehicles could be

DOI:10.15774/PPKE.ITK.2020.010

proper landmark objects, however during the SfM process, large objects may fall apart into several parts yielding erroneously detected objects which can lead to inaccurate registration.

4.4.3 Implementation details and running time

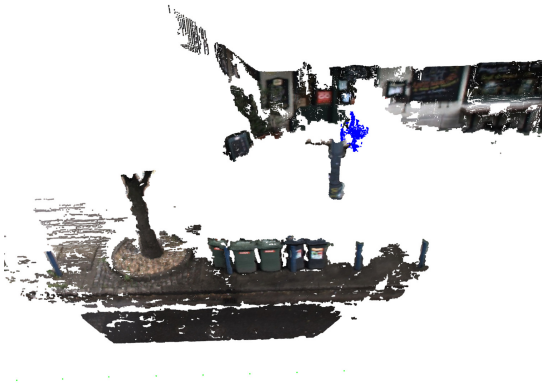
We implemented the proposed method using C++ and OpenGL. An NVIDIA GTX1080Ti GPU with 11 GB RAM was used to speed up the calculations. The running time of the entire calibration pipeline is less than 3 minutes. For data acquisition we used 1288×964 resolution RGB cameras and a Velodyne HDL-64E RMB Lidar sensor.

4.5 Conclusion

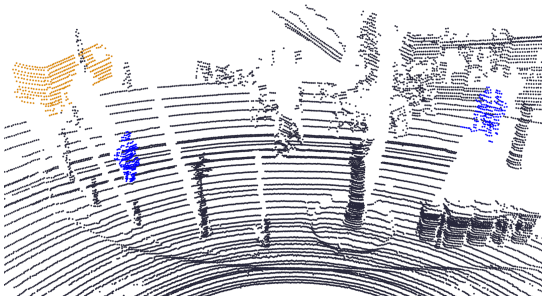
Chapter proposed a new target-less, automatic camera-Lidar calibration approach which can be performed on-the-fly, i.e., during the driving without stopping the scanning platform. Our method does not rely on any external sensors such as GNSS and IMU and we do not use hardware based time synchronization between the Lidar and the camera. The method can periodically re-calibrate the sensors to handle small sensor displacements and it also adapts to the distortion of the RMB Lidar point cloud.

We quantitatively evaluated and compared our method to state-of-the-art target-less techniques and we proved that our approach surpasses them. Furthermore we have shown that in some cases the proposed method even reaches the accuracy of the considered target-based reference method.

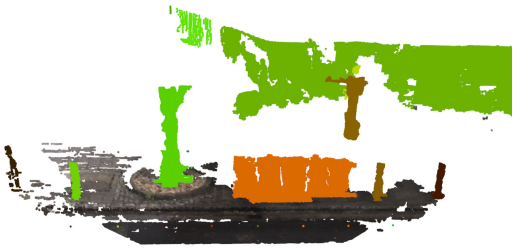
DOI:10.15774/PPKE.ITK.2020.010



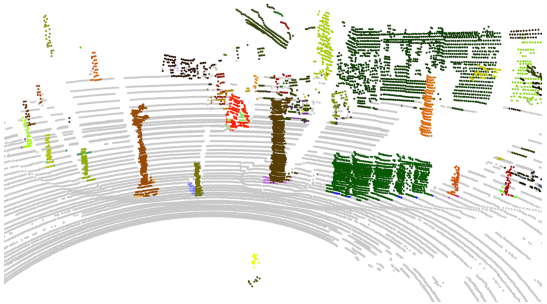
(a) Pedestrian detection (blue) in the SfM cloud [27]



(b) Pedestrian (blue) and vehicle (orange) detection in the Lidar cloud [25]



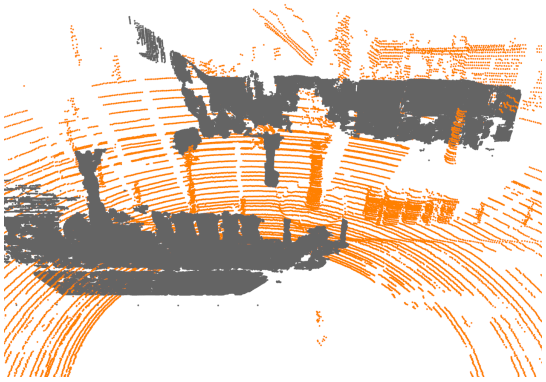
(c) Abstract object extraction in the SfM cloud



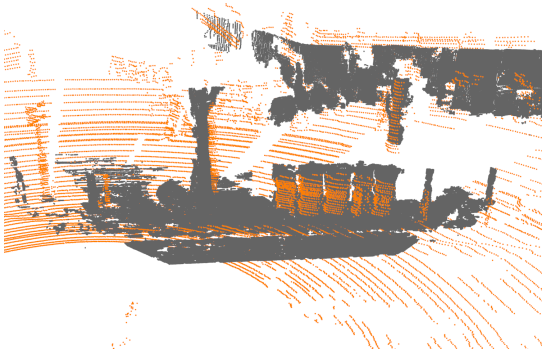
(d) Abstract object extraction in the Lidar cloud

Figure 4.13: Demonstration of intermediate steps of the proposed method in a selected scene. Results of the (a)(b) dynamic object filtering and (c)(d) 3D abstract object extraction steps, which are used as input of the proposed object based alignment technique (Sec. 4.3.2.2).

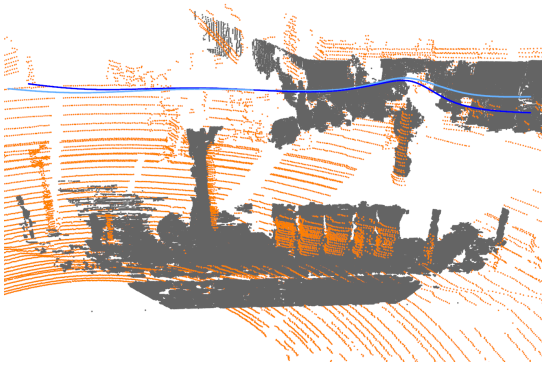
DOI:10.15774/PPKE.ITK.2020.010



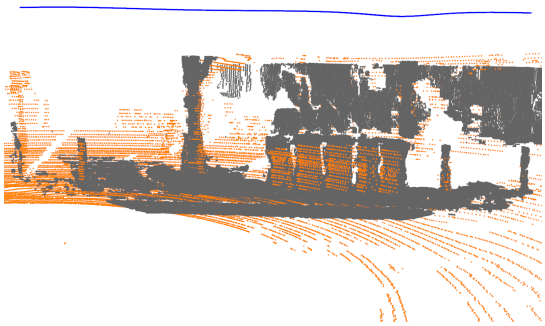
(a) Initial translation error between the point clouds



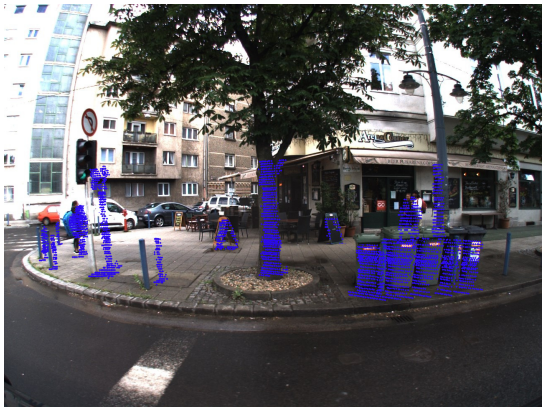
(b) Object based coarse alignment



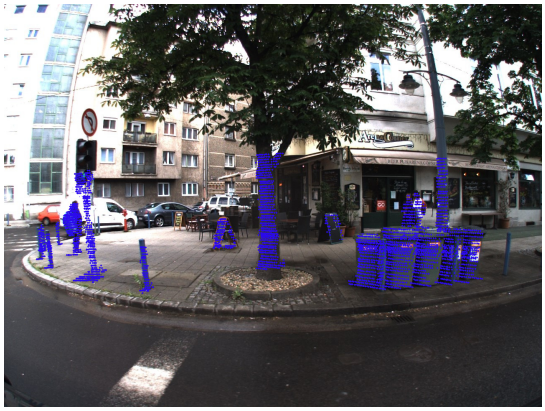
(c) Curve fitting based on object pairs



(d) ICP and curve based refinement



(e) Projection result after the coarse alignment



(f) Projection result after the refinement

Figure 4.14: Qualitative results of the proposed registration and projection steps.

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

Chapter 5

Conclusion

This thesis has been dealing with novel research connected to Lidar and camera-based environment perception. We investigated three challenging research problems, namely deep learning based point cloud segmentation, Lidar sensor-based localization of self-driving vehicles, and targetless camera-Lidar calibration. Deep convolutional neural network based classification and 3D geometric methods have been jointly proposed to solve the introduced problems. It has been shown both quantitatively and qualitatively that the proposed approaches are able to outperform the state-of-the-art methods and they were evaluated in real-life scenarios using real sensor measurements obtained by different mobile mapping systems and Lidar sensors such as Riegl VMX-450 and Velodyne HDL64E.

5.1 New Scientific Results

1. Thesis: I have proposed a novel scene segmentation model which is able to semantically label dense 3D point clouds collected in an urban environment. Using the segmented dense point clouds as detailed background maps I have proposed a Lidar-based self-localization approach for self-driving vehicles. I have quantitatively compared and evaluated the proposed approaches against state-of-the-art methods in publicly available databases.

DOI:10.15774/PPKE.ITK.2020.010

Published in [2][10][11][12]

The proposed scene segmentation model aims to semantically label each point of a dense point cloud obtained by a mobile mapping system such as Riegl VMX-450 into different categories. Since the collected point clouds are geo-referenced i.e. the sensor registers all incoming scans into the same coordinate system, dynamic objects moving concurrently with the scanning platform appear as a long-drawn, noisy region called *phantom objects* in this thesis. Furthermore, during the scanning process, the mapping system may scan the same regions several times yielding an inhomogeneous point density with the mentioned noisy phantom region. Though the mapping system assigns color information to the recorded 3D points, because of several occlusion and illumination artifacts this color information is often unreliable.

The proposed model is able to robustly segment the dense point cloud scenes collected in an urban environment and according to our experiments, the model can be adapted to segment point clouds with different data characteristics. To demonstrate the utility of the segmented dense map, I have proposed a real-time localization algorithm, which is able to register the Lidar data of a self-driving vehicle to the dense map.

1.1. I have proposed a voxel-grid representation of the point cloud data containing two feature channels derived from the density and height properties of the given point cloud segments. I have shown that the proposed voxelized data is a compact representation of the point cloud which can be used as a direct training input of CNN architectures. To experimentally validate the proposed approach and to show the benefits of the introduced voxel data representation I have created a new manually labeled mobile laser scanning (MLS) database and I have made it publicly available.

Several point cloud segmentation approaches exist in the literature [51, 28, 29, 26], however, most of them perform weakly in cases of inhomogeneous point density and most of them do not consider the global position of the given point cloud segments, but treat them just local independent training samples. The proposed

DOI:10.15774/PPKE.ITK.2020.010

data representation, before voxelizing the data, takes a local point neighborhood of the given sample, then it assigns a density and the global position value to each voxel of the sample referred to as a first and second channel. I have shown that the two-channel voxelized data is a compact representation of the raw point cloud and it can be efficiently learned using CNN architectures. To train and test deep neural networks I have implemented an efficient tool for point cloud annotation and I have created a large dataset for point cloud segmentation which contains around 500 million manually labeled points. I have made the dataset publicly available (`http://mplab.sztaki.hu/geocomp/SZTAKI-CityMLS-DB.html`).

1.2. For segmenting 3D point cloud data, I have proposed a new LeNet-style 3D CNN architecture which is able to efficiently learn the 2-channel voxelized data structure proposed in the previous thesis. I have evaluated the proposed method both on the proposed manually annotated dataset and on various well-known databases. I have shown the advantages of the proposed approach versus different state-of-the-art deep learning based models published in top journals and conferences in the last 5 years.

I have proposed a new 3D convolutional neural network which is able to efficiently extract different features from the voxelized data representation of dense MLS point clouds. I have quantitatively shown that considering the density and global position feature channels, the method is able to overcome other state-of-the-art methods in an inhomogeneous point cloud labeling task.

1.3. I have proposed a new real-time point cloud alignment method for point clouds taken with different sensors exhibiting significantly different density properties and I have proposed a vehicle localization technique using the segmented dense point cloud as a high-resolution map. I have experimentally evaluated the proposed method in various urban scenarios.

Due to the lack of signals, GPS based localization is often unreliable, thus the accuracy of it usually fluctuates between a wide range. According to our experiments, GPS position error can be larger to 10 meters in a dense urban environment such as the streets of Budapest, Hungary. I have proposed an object-based

DOI:10.15774/PPKE.ITK.2020.010

coarse alignment procedure for point cloud registration and I showed that the registration result of the method can be further refined with a point level registration step. I have proposed a new robust key-points extracting method, which is able to extract robust registration key-points from the 3D point cloud segments to make the coarse alignment process more reliable. I have experimentally shown that the proposed coarse alignment method is able to reduce the positioning error of the GPS measurement below 0.4 meter which is a significant improvement against the initial 10 meters translation error.

2. Thesis: I have proposed an automatic, target-less camera-Lidar extrinsic parameter calibration method and I have shown the advantages of the method versus different state-of-the-art algorithms.

Published in [1][7][9]

The main goal of the proposed method is to calibrate a camera and a Lidar sensor mounted onto the top of a moving vehicle in an automatic way. The proposed automatic method works in an end-to-end manner without any user interaction and it does not require any specific calibration objects such as 3D boxes and chessboard patterns. The main advantage of the method that it is able to periodically re-calibrate the sensors on-the-fly i.e. without stopping the vehicle.

2.1. I have proposed a redefinition of the camera-Lidar calibration task by generating a 3D point cloud from the consecutive camera frames using a Structure from Motion method and registering the Lidar and the SfM point cloud in the 3D domain.

Detecting meaningful feature correspondences between 2D and 3D domain is very challenging, since extracting the same features, points, or lines from a 2D image and a 3D point cloud domain is unreliable. To avoid this feature association problem, I have proposed a Structure from Motion pipeline to generate a 3D point cloud from the consecutive camera frames. I have formulated the camera-Lidar calibration problem as a point cloud registration task in the 3D domain so that the method aligns the Lidar point cloud to the coordinate system of the generated

DOI:10.15774/PPKE.ITK.2020.010

SfM point cloud. The point cloud registration consists of two main steps: an object-level alignment and a curve-based fine alignment process.

2.2. I have proposed a robust object-level registration technique between the 3D point cloud and the generated SfM point cloud data and I have proposed object filtering methods to make more robust the alignment.

I have modified the object-based coarse alignment process proposed in (Thesis 1) [10], and I have extended it with filtering methods using two state-of-the-art deep neural networks to eliminate noisy dynamic object regions which may erroneously miss-lead the registration process. For registration, I have relied on static street furniture objects such as *columns*, *traffic signs* and *tree trunks*, furthermore based on the deep learning based object filtering the proposed method is able to analyze the scene and it only starts a new re-calibration if an adequate number of static objects are detected in the given scene.

2.3. I have proposed a curve-based point cloud registration refinement algorithm which is able to correct the local deformation of the SfM point cloud.

During the SfM pipeline, local point cloud deformations and scaling errors can occur which may have a great effect on the registration process, therefore I have proposed a control curve based algorithm to eliminate these artifacts. Based on the static objects used in the coarse alignment process I have fitted a NURBS curve both to the Lidar and the SfM point cloud. The control curves describe the shape and the distortion of the point clouds and I have proposed an algorithm which is able to align the curve segments through a non-linear transformation i.e. it deforms the Lidar point cloud according to the shape of the SfM one. As a result of the refinement, the method is able to precisely register the point clouds and it is able the calculate the proper transformation matrix which projects the 3D Lidar points onto the corresponding 2D image pixels. I have quantitatively evaluated and compared the proposed method against state-of-the-art reference techniques on a 10 km long test set. I have chosen 200 different keyframes from various scenarios such as *main roads*, *narrow streets*, etc., and I have shown that the proposed approach is able to be a competitive alternative against state-of-the-art targetless methods and in some cases, it even overcomes the accuracy of target-based methods.

DOI:10.15774/PPKE.ITK.2020.010

5.2 Application of the Results

The developed algorithms can be used by various up-to-date or future computer vision systems, especially in the application fields of autonomous driving. Many of the proposed methods directly corresponded to research projects conducted with the participation of SZTAKI and PPCU in the previous years.

Various contributions in RMB Lidar based scene analysis have been adopted in automotive industrial projects. We also integrated the person surveillance module into a real-time demonstrator, which has been introduced at the Frankfurt Motorshow 2017 in the exhibition area of sensor producer Velodyne, at the Automotive Hungary 2017 exhibition, and in multiple Researchers' Night occasions (in Hungarian: Kutatók éjszakája), which are open yearly events for the public to visit research centers in Hungary.

5.3 Implementation details

The main framework for point cloud handling, processing and visualization was implemented in C/C++ and OpenGL while the neural network models was implemented and train in Python3 with TensorFlow and Keras backends. The hardware set up for training contains two Nvidia Geforce RTX 2080 Ti GPU with 2×11 GB device memory and 64 GB system memory.

DOI:10.15774/PPKE.ITK.2020.010

Appendix A

Supplementary Material

A.1 Lidar technology and used sensors in this thesis

The term Lidar was created as "light" and "radar". As for the technology background, Lidar sensors calculate the distance between the sensor and the target objects from the echo time of the emitted and the received laser beam where the beam spreads with the speed of light. The result of the measurement is a highly accurate 3D point cloud where the coordinates of the points are given in a local or global coordinate system depending on the type of the Lidar system and the application area.

A.1.1 Riegl VMX-450

VMX-450 (Fig. A.1(a)) is a high speed mobile laser scanning system for data acquisition. It is mounted onto the top of a moving car and it offers extremely dense, accurate and feature-rich data even at high driving speed. The system integrates two RIEGL VQ-450 laser scanners, IMU, GNSS and a well-design camera platform ensures mounting up to six digital cameras. Each of the two VQ-450 laser scanners provides low-noise, gap-less 360 degree frames at a measurement rate of 550 thousand points/sec. The camera system completes the scan data with precisely time-stamped images. Typical applications include mapping of transportation infrastructure, city modeling, network planning and surveying of mining.

DOI:10.15774/PPKE.ITK.2020.010

(a) Riegl VMX-450 mobile mapping system

(b) Velodyne HDL64E RMB Lidar sensor

Figure A.1: MLS technology used in this thesis.

A.1.2 Velodyne HDL64E

The Velodyne HDL64E (Fig. A.1(b)) sensor has been designed to navigate autonomous cars and ships. It has a 360 degree horizontal and a 26.8 degree vertical angle of view with $5 - 15$ Hz operating speed. It has 64 laser beams and it can produce 1.3 million points/second. It provides a 2.5D point cloud within a 120m range. Beside the 3D relative positions, the points also contain an intensity value what gives the strength of the returned laser beam. This intensity value highly depends on the the angle of incidence and the surface properties of the objects.

DOI:10.15774/PPKE.ITK.2020.010

A.2 Fundamentals of deep learning

In this section I give a brief insight into the core concept of deep learning focusing on convolutional neural networks. I introduce all elementary building blocks such as *different type of layers*, *activation functions* and *training strategies*, which were used in Chapter 2 to construct the proposed model. Note that these definitions are simplified containing the most important aspects to support the understanding of the thesis, however I cite the original papers for further details.

A.2.1 Convolution layer

Convolution layer [52] is designed to learn a set a filters (F) by sliding the filters across the input (I) volume and using the convolution operation to calculate the dot products between the input volume and the filters at each spatial position. The output of the convolution layer is a set of computed feature maps (FM) which are stacked along the depth dimension.

In our case, the input of the proposed 3D C^{2}CNN model (Chapter 2) is a 3D volume with two channels containing the density and global position information. We can define the proposed convolution operation as the following:

$$FM(x,y,z) = (F*I)(x,y,z) = \sum_{i_x}^{w_f}\sum_{i_y}^{h_f}\sum_{i_z}^{d_f}\sum_{i_c}^{d} I(x-i_x, y-i_y, z-i_z, i_c)\cdot F(i_x, i_y, i_z, i_c)$$

Filter F is an element $F \in \mathbb{R}^{w_f \times h_f \times d_f \times d}$, where w_f represents the width, h_f the height d_f the depth (3D volume) of the filter and d is the number of input channels to which the filter is applied. $I \in \mathbb{R}^{w_i \times h_i \times d_i \times d}$ is the 3D input volume (training samples), where d is the number of the channels (in our case $d = 2$, namely the density and the height properties).

Through the convolution layers various local features of the input volume can be detected by learning different filters, which are able to take into consideration the spatial structure of the input data (e.g. pixels in case of images and voxels in our proposed model). Since traditional neural networks implemented in a way that each input units interact with every output units, it makes hard to train these networks with huge number of parameters. The sparse connectivity

DOI:10.15774/PPKE.ITK.2020.010

of units and the parameter sharing (the same filter can be applied at different position of the input volume) technique of convolution layers make CNNs efficient feature extractor models.

The number of filters F in a given layer and the kernel size of the it are all hyperparameters that can be fine-tuned to improve the performance of the model.

A.2.2 Pooling layer

Pooling operation [96] is also applied using the sliding window technique and inside a local neighborhood of the given feature map it highlights the relevant features by spatially sub-sampling the input volume independently within each of its depth slices. Since pooling layers compute a fixed function of the input and sub-sampling it, they reduce the number of the parameters and they do not calculate any new parameters.

Max pooling and Average pooling are the most used pooling operations, where Max pooling simply select the maximum value from the given kernel (i.e. from the given receptive field), while Average pooling take the average of the values of the given receptive field.

A.2.3 Fully Connected layers

In fully connected (or dense) layers each unit (neuron) is connected to the neurons of the next layer. While convolution and pooling layers are responsible for extracting relevant features, dense layers are acting as regular artificial neural networks and their aim is to classify the extracted features.

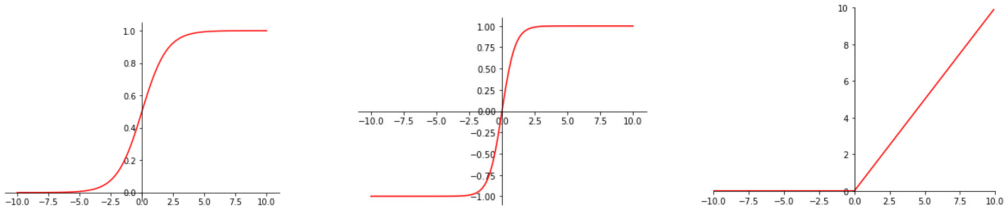
A.2.4 Dropout regularization

In deep learning regularization is a technique to prevent over-fitting. Dropout [95] simply ignores a certain set of neurons chosen by randomly at each training phase. Using Dropout regularization, neural networks are able to learn more robust and general features, since the model cannot rely on a specific set of neurons, because they are activated randomly.

DOI:10.15774/PPKE.ITK.2020.010

A.2.5 Activation functions

Since convolution layers learn linear filters, the calculated output usually activated by a non-linear activation function to allow the neural network to approximate complex, non-linear functions. The most common activation functions are the: *sigmoid*, *hyperbolic tangent*, and *ReLu* (see Fig. A.2).



(a) Sigmoid (b) Tanh (c) ReLu

Figure A.2: Most commonly used activation functions.

Rectified Linear Unit (ReLu) [97] computes the function $f(x) = max(0, x)$, which is simpler to implement and computationally more effective than the exponential ones such as sigmoid and tanh, furthermore ReLu activation greatly accelerates the convergence of stochastic gradient descent optimization process.

A.2.6 Loss function and training process

To start the training process, the parameters of the network must be initialized, which parameters are usually chosen by randomly. In the first stage (forward pass), the training data is propagated through the network and for each input x_i the network calculates a prediction $\hat{y}_i$. In the next step, the predictions are compared to the ground truth labels and a loss is calculated between them $loss(\hat{y}_i, y_i)$. The final step is the backward pass, where each parameters of the network are adjusted according to the calculated loss. To measure the performance of the network several metrics can be found in the literature. In this thesis (Chapter 2), we have proposed a multi label classification problem, so the categorical cross entropy loss was used.

DOI:10.15774/PPKE.ITK.2020.010

In this thesis, for each input data (3D volume with two channels) x_i, a ground truth label y_i is given, which corresponds one of the n_c classes, $n_c = 9$ in this work. Our aim is to minimize the cost function J:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} loss(\hat{y}_i, y_i)$$

over the complete training set, where θ represents the parameters (weights W and the biases b) of the network. To give a predicted class score to each prediction $\hat{y}_i$ we have used the *Softmax* activation function, which can be defined as the following:

Definition A.2.1. With $\mathbf{z} = (z_1, \ldots, z_n) \in \mathbb{R}^n$, the standard Softmax function $\sigma : \mathbb{R}^n \to \mathbb{R}^n$ is defined as $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$ for $i = 1, \ldots, n$.

The Softmax function ensures a discrete probability distribution over the classes. The aim is to maximize the probability of the correct class y_i, so that the negative log probability of that class needs to be minimized:

$$-\log(P(y_i \mid x_i); \theta) = -\log(\frac{\exp(s_i)}{\sum_{c=1}^{n_c} \exp(s_c)}),$$

where calculate s_i by taking the i'th row of W and multiple it with x:

$$s_i = \sum_{j=1}^{d} W_{ij} \cdot x_j,$$

and compute all s_c for $c = 1, ..., n_c$. Here W is the weight matrix and d is the dimension of the vector x.

For a multi-class classification problem, the standard loss function is the *cross-entropy*, which measures the difference between two probability distributions, so we used the cross-entropy loss to train our network over the full dataset $\{x_i, y_i\}_{i=1}^{N}$:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} -\log(\sigma(r_i)),$$

where r is the weighted sum of the output of the neurons:

$$r = \sum_i w_i x_i + b \tag{A.1}$$

DOI:10.15774/PPKE.ITK.2020.010

To minimize the cross-entropy loss and find the optimal network parameters over the training set, we used the *stochastic gradient descent* (SGD) [98] algorithm and the back propagation algorithm to compute the partial derivatives and update the parameters of the network according to the calculated loss.

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

Appendix B

Summary of Abbreviations

Chapter 1	
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
Lidar	Light Detection and Ranging Technology
HD	High Definition
GIS	Geographic Information System
TLS	Terrestrial laser scanning
MMS	Mobile mapping system
IMU	Inertial Measurement Unit
GNSS	Global Navigation Satellite System
CNN	Convolutional Neural Network
ICP	Iterative Closest Point
GPS	Global Positioning System
SfM	Structure from Motion
SGD	Stochastic Gradient Descent
Chapter 2	
SDV	Self-driving Vehicle
MLS	Mobile Laser Scanning
SGPN	Similarity Group Proposal Network
SPLATNET	Sparse Lattice Network
BCL	Bilateral Convolution Layer
RMB	Rotating Multi-beam
ReLu	Rectified Linear Unit
OG	Occupancy Grid
Pr	Precision
Rc	Recall
F-r	F-rates
Chapter 3	
SLAM	Simultaneous Localization and Mapping
NDT	Normal Distribution Transform
RANSAC	Random Sample Consensus

DOI:10.15774/PPKE.ITK.2020.010

Chapter 4

NURBS	Non-uniform rational B-spline
SVS	Sparse Voxel Structure
PCA	Principal Components Analysis
OBA	Object Based Coarse Alignment
DOF	Dynamic Object Filtering
GT	Ground Truth
TBR	Target-based Reference

DOI:10.15774/PPKE.ITK.2020.010

DOI:10.15774/PPKE.ITK.2020.010

References

The author's journal publications and book chapters

[1] **B. Nagy** and C. Benedek, "On-the-fly camera and lidar calibration," *MDPI Remote Sensing*, vol. 12, no. 7, 2020. (document), 1.2, 5.1

[2] **B. Nagy** and C. Benedek, "3D CNN-based semantic labeling approach for mobile laser scanning data," *IEEE Sensors Journal*, vol. 19, no. 21, pp. 10034–10045, 2019. (document), 1, 1.1, 1.1, 3.1.3, 3.3.1, 5.1

[3] C. Benedek, B. Gálai, **B. Nagy**, and Z. Jankó, "Lidar-based gait analysis and activity recognition in a 4d surveillance system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 1, pp. 101–113, 2018.

[4] A. Börcs, **B. Nagy**, and C. Benedek, "Instant object detection in Lidar point clouds," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 7, pp. 992–996, 2017. 1, 2.2

[5] A. Börcs, **B. Nagy**, and C. Benedek, "Dynamic environment perception and 4d reconstruction using a mobile rotating multi-beam Lidar sensor," in *Handling Uncertainty and Networked Structure in Robot Control*, vol. 42, pp. 153–180, Springer, 2015.

The author's international conference publications

DOI:10.15774/PPKE.ITK.2020.010

[6] O. Zováthi, **B. Nagy**, and C. Benedek, "Exploitation of dense MLS city maps for 3D object detection," in *International Conference on Image Analysis and Recognition (ICIAR), (virtual conference)*, Lecture Notes in Computer Science, (Póvoa de Varzim, Portugal), pp. 1317–1321, 2020.

[7] **B. Nagy**, L. Kovács, and C. Benedek, "SFM and semantic information based online targetless camera-lidar self-calibration," in *IEEE International Conference on Image Processing, (ICIP)*, (Taipei, Taiwan), pp. 1317–1321, September 2019. 5.1

[8] Y. Ibrahim, **B. Nagy**, and C. Benedek, "CNN-based watershed marker extraction for brick segmentation in masonry walls," in *16th International Conference on Image Analysis and Recognition, (ICIAR)*, vol. 11662 of *Lecture Notes in Computer Science*, (Waterloo, Canada), pp. 332–344, Springer, August 2019.

[9] **B. Nagy**, L. Kovács, and C. Benedek, "Online targetless end-to-end camera-Lidar self-calibration," in *16th International Conference on Machine Vision Applications, (MVA)*, (Tokyo, Japan), pp. 1–6, IEEE, May 2019. 5.1

[10] **B. Nagy** and C. Benedek, "Real-time point cloud alignment for vehicle localization in a high resolution 3D map," in *European Conference on Computer Vision (ECCV) Workshops*, vol. 11129 of *Lecture Notes in Computer Science*, (Munich, Germany), pp. 226–239, Springer, September 2018. 2.1, 4.3, 4.3.2, 4.3.2.2, 5.1, 5.1

[11] **B. Nagy** and C. Benedek, "3D CNN based phantom object removing from mobile laser scanning data," in *International Joint Conference on Neural Networks, (IJCNN)*, (Anchorage, USA), pp. 4429–4435, IEEE, May 2017. 2.1.1, 2.4, 3.3.1, 5.1

[12] B. Gálai, **B. Nagy**, and C. Benedek, "Crossmodal point cloud registration in the Hough space for mobile laser scanning data," in *23rd International Conference on Pattern Recognition, (ICPR)*, (Cancún, Mexico), pp. 3374–3379, IEEE, December 2016. (document), 1.1, 3.2, 3.3, 3.3.3.1, 3.4, 3.8, 5.1

DOI:10.15774/PPKE.ITK.2020.010

[13] C. Benedek, **B. Nagy**, B. Gálai, and Z. Jankó, "Lidar-based gait analysis in people tracking and 4D visualization," in *23rd European Signal Processing Conference, (EUSIPCO)*, (Nice, France), pp. 1138–1142, IEEE, August 2015.

[14] A. Börcs, **B. Nagy**, M. Baticz, and C. Benedek, "A model-based approach for fast vehicle detection in continuously streamed urban LIDAR point clouds," in *Asian Conference on Computer Vision, (ACCV), Workshops*, vol. 9008 of *Lecture Notes in Computer Science*, (Singapore), pp. 413–425, Springer, November 2014. 4.4.2

[15] A. Börcs, **B. Nagy**, and C. Benedek, "Fast 3D urban object detection on streaming point clouds," in *European Conference on Computer Vision (ECCV) Workshops*, vol. 8926 of *Lecture Notes in Computer Science*, (Zurich, Switzerland), pp. 628–639, Springer, September 2014. 1, 3.1.1, 3.3.2.1, 4.3.2.1, 4.3.2.1

The author's other publications

[16] O. Zováthi, L. Kovács, **B. Nagy**, and C. Benedek, "Multi-object detection in urban scenes utilizing 3D background maps and tracking," in *International Conference on Control, Artificial Intelligence, Robotics Optimization (ICCAIRO)*, pp. 231–236, 2019.

[17] **B. Nagy** and C. Benedek, "3D CNN alapú MLS pontfelhőszegmentáció," in *Conference of Hungarian Association for Image Analysis and Pattern Recognititon*, (Debrecen, Hungary), 2019.

[18] O. Zováthi, **B. Nagy**, and C. Benedek, "Valós idejű pontfelhőillesztés és járműlokalizáció nagy felbontású 3D térképen," in *Conference of Hungarian Association for Image Analysis and Pattern Recognititon*, (Debrecen, Hungary), 2019.

DOI:10.15774/PPKE.ITK.2020.010

[19] **B. Nagy**, B. Gálai, and C. Benedek, "Multimodális pontfelhőregisztráció hough tér alapú előillesztéssel," in *Conference of Hungarian Association for Image Analysis and Pattern Recognititon*, (Sovata, Romania), 2017.

[20] A. Börcs, **B. Nagy**, and C. Benedek, "Utcai objektumok gyors osztályozása lidar pontfelhősorozatokon," in *Conference of Hungarian Association for Image Analysis and Pattern Recognititon*, (Sovata, Romania), 2017.

[21] A. Börcs, **B. Nagy**, and C. Benedek, "Valós idejű járműdetekció lidar pontfelhő sorozatokon," in *Conference of Hungarian Association for Image Analysis and Pattern Recognititon*, (Kerekegyháza, Hungary), 2015.

[22] **B. Nagy**, C. Benedek, and Z. Jankó, "Mozgó személyek követése és 4d vizualizációja lidar alapú járáselemzéssel," in *Conference of Hungarian Association for Image Analysis and Pattern Recognititon*, (Kerekegyháza, Hungary), 2015.

Publications connected to the dissertation

[23] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Hawaii), pp. 6620–6629, 2017. 1, 2.1.3, 2.2

[24] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Singapore), pp. 1355–1361, 2017. 1, 2.2

[25] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (California, USA), June 2019. (document), 1, 4.3, 4.3.2.1, 4.6, 4.13(b)

DOI:10.15774/PPKE.ITK.2020.010

[26] C. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Conference on Neural Information Processing Systems (NIPS)*, (Long Beach, CA, USA), 2017. 1, 1, 1.1, 2.1.3, 2.2, 2.5.3, 2.3, 2.5.4, 3.3.1, 5.1

[27] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, (Venice, Italy), pp. 2980–2988, Oct 2017. (document), 1, 1, 4.3, 2, 4.3, 4.6, 4.13(a)

[28] W. Wang, R. Yu, Q. Huang, and U. Neumann, "SGPN: Similarity group proposal network for 3D point cloud instance segmentation," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, (Salt Lake City, UT), pp. 2569–2578, June 2018. 1, 1, 2.2, 5.1

[29] J. Huang and S. You, "Point cloud labeling using 3D convolutional neural network," in *International Conference on Pattern Recognition (ICPR)*, (Cancun, Mexico), pp. 2670–2675, 2016. 1, 1, 1.1, 2.2, 2.5.3, 2.3, 5.1

[30] H. Su, V. Jampani, D. Sun, S. Maji, V. Kalogerakis, M.-H. Yang, and J. Kautz, "SPLATNet: Sparse lattice networks for point cloud processing," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Salt Lake City, UT), pp. 2530–2539, June 2018. 1, 1, 1.1, 2.2, 2.5.3, 2.3, 2.5.4

[31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, USA), pp. 779–788, 2016. 1

[32] G. Máttyus, S. Wang, S. Fidler, and R. Urtasun, "Hd maps: Fine-grained road segmentation by parsing ground and aerial images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, USA), pp. 3611–3619, 2016. 1

[33] B. Yang, M. Liang, and R. Urtasun, "Hdnet: Exploiting hd maps for 3D object detection," in *Proceedings of The 2nd Conference on Robot Learning*

DOI:10.15774/PPKE.ITK.2020.010

(A. Billard, A. Dragan, J. Peters, and J. Morimoto, eds.), vol. 87 of *Proceedings of Machine Learning Research*, pp. 146–155, PMLR, 29–31 Oct 2018. 1

[34] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. F. R. Jesus, R. F. Berriel, T. M. Paixão, F. W. Mutz, T. Oliveira-Santos, and A. F. de Souza, "Self-driving cars: A survey," *ArXiv*, vol. abs/1901.04407, 2019. 1

[35] A. Serna, B. Marcotegui, F. Goulette, and J. Deschaud, "Paris-rue-madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods," in *International Conference on Pattern Recognition Application and Methods (ICPRAM)*, (Angers, France), 2014. 1, 1.1, 2.3, 2.5.3

[36] B. Vallet, M. Brédif, A. Serna, B. Marcotegui, and N. Paparoditis, "Terramobilita/iqmulus urban point cloud analysis benchmark," *Computers and Graphics*, vol. 49, pp. 126–133, 2015. 1, 1.1, 2.3, 2.5.3

[37] T. Hackel, N. Savinov, L. Ladicky, J. Wegner, K. Schindler, and M. Pollefeys, "Semantic3D.net: A new large-scale point cloud classification benchmark," in *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. IV-1-W1, pp. 91–98, 2017. 1, 1.1, 2.3, 2.5.3

[38] D. Munoz, J. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin markov networks," in *IIEEE on Computer Vision and Pattern Recognition (CVPR)*, (Miami, USA), pp. 975–982, 2009. 1, 1.1, 2.3, 2.5.3

[39] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal, and J. Hertzberg, "Evaluation of 3D registration reliability and speed - a comparison of ICP and NDT," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Kobe, Japan), pp. 3907–3912, May 2009. 1, 4.3.2, 4.3.2.2

[40] N. K. Ratha, K. Karu, S. Chen, and A. K. Jain, "A real-time matching system for large fingerprint databases," *IEEE Transactions on Pattern Analysis and*

DOI:10.15774/PPKE.ITK.2020.010

Machine Intelligence, vol. 18, pp. 799–813, Aug 1996. (document), 1.2, 3.3, 3.3.3.3, 3.6

[41] H. Zheng, R. Wang, and S. Xu, "Recognizing street lighting poles from mobile LiDAR data," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 55, no. 1, pp. 407–420, 2017. 2.1, 2.1.1, 2.2

[42] Y. Yu, J. Li, H. Guan, and C. Wang, "Automated detection of three-dimensional cars in mobile laser scanning point clouds using DBM-Hough-Forests," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 54, no. 7, pp. 4130–4142, 2016. 2.1, 2.1.1

[43] B. Wu, B. Yu, W. Yue, S. Shu, W. Tan, C. Hu, Y. Huang, J. Wu, and H. Liu, "A voxel-based method for automated identification and morphological parameters estimation of individual street trees from mobile laser scanning data," *Remote Sensing*, vol. 5, no. 2, p. 584, 2013. 2.1, 2.1.1, 2.2

[44] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "LOCI: fast outlier detection using the local correlation integral," in *International Conference on Data Engineering*, (Los Alamitos, CA, USA), pp. 315–326, 2003. 2.2

[45] S. Sotoodeh., "Outlier detection in laser scanner point clouds," in *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. XXXVI–5, pp. 297–302, 2006. 2.2

[46] J. Köhler, T. Nöll, G. Reis, and D. Stricker., "Robust outlier removal from point clouds acquired with structured light," in *Eurographics (Short Papers)*, (Cagliari, Italy), pp. 21–24, 2012. 2.2

[47] T. Kanzok, F. Süß, L. Linsen, and R. Rosenthal, "Efficient removal of inconsistencies in large multi-scan point clouds," in *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, (Pilsen, Czech Rep.), 2013. 2.2

DOI:10.15774/PPKE.ITK.2020.010

[48] J. Gehrung, M. Hebel, M. Arens, and U. Stilla, "An approach to extract moving objects from MLS data using a volumetric background representation," in *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. IV-1, 2017. 2.2

[49] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3D point clouds for indoor scenes.," in *International Conference on Neural Information Processing Systems (NIPS)*, (Granada, Spain), pp. 244–252, 2011. 2.2

[50] T. Hackel, J. D. Wegner, and K. Schindler, "Fast semantic segmentation of 3D point clouds with strongly varying density," *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. III-3, 2016. 2.2

[51] G. Pang and U. Neumann, "3D point cloud object detection with multi-view convolutional neural network," in *International Conference on Pattern Recognition (ICPR)*, (Cancun, Mexico), pp. 585–590, 2016. 1.1, 2.2, 2.5.3, 2.3, 5.1

[52] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998. 2.4.2, A.2.1

[53] B. Bayat, N. Crasta, A. Crespi, A. M. Pascoal, and A. Ijspeert, "Environmental monitoring using autonomous vehicles: a survey of recent searching techniques," *Current Opinion in Biotechnology*, vol. 45, pp. 76 – 84, 2017. 3.1

[54] M. Kang, S. Hur, W. Jeong, and Y. Park, "Map building based on sensor fusion for autonomous vehicle," in *International Conference on Information Technology: New Generations*, (Las Vegas, NV, USA), pp. 490–495, April 2014. 3.1

[55] H. G. Seif and X. Hu, "Autonomous driving in the icityhd maps as a key challenge of the automotive industry," *Engineering*, vol. 2, no. 2, pp. 159 – 162, 2016. 3.1.1

DOI:10.15774/PPKE.ITK.2020.010

[56] R. Matthaei, G. Bagschik, and M. Maurer, "Map-relative localization in lane-level maps for ADAS and autonomous driving," in *IEEE Intelligent Vehicles Symposium Proceedings*, (Dearborn, MI, USA.), pp. 49–55, June 2014. 3.1.1

[57] B. Douillard, A. Quadros, P. Morton, J. P. Underwood, M. D. Deuge, S. Hugosson, M. Hallström, and T. Bailey, "Scan segments matching for pairwise 3D alignment," in *IEEE International Conference on Robotics and Automation (ICRA)*, (St. Paul, MN, USA), pp. 3033–3040, May 2012. 3.1.1, 3.2

[58] J. Behley, V. Steinhage, and A. B. Cremers, "Performance of histogram descriptors for the classification of 3D laser range data in urban environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, (St. Paul, MN, USA), pp. 4391–4398, 2012. 3.1.1

[59] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision*, vol. 13, pp. 119–152, October 1994. 3.1.1, 3.2

[60] M. Magnusson, *The Three-Dimensional Normal-Distributions Transform – an Efficient Representation for Registration, Surface Analysis, and Loop Detection*. PhD thesis, Örebro University, December 2009. 3.1.1, 3.2

[61] O. Józsa, A. Börcs, and C. Benedek, "Towards 4d virtual city reconstruction from lidar point cloud sequences," *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, vol. II-3, pp. 15–20, 05 2013. 3.1.1, 4.2, 4.3.2, 4.3.2.1

[62] A. Gressin, C. Mallet, and N. David, "Improving 3D LIDAR point cloud registration using optimal neighborhood knowledge," *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, pp. 111–116, 2012. 3.2

[63] H. Men, B. Gebre, and K. Pochiraju, "Color point cloud registration with 4D ICP algorithm," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 1511–1516, May 2011. 3.2

DOI:10.15774/PPKE.ITK.2020.010

[64] A. Gressin, B. Cannelle, C. Mallet, and J.-P. Papelard, "Trajectory-based registration of 3D LIDAR point clouds acquired with a mobile mapping system," *Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS)*, pp. 117–122, 2012. 3.2

[65] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Kobe, Japan), pp. 3212–3217, May 2009. 3.2

[66] A. Mian, M. Bennamoun, and R. Owens, "On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes," *International Journal of Computer Vision*, vol. 89, pp. 348–361, Sep 2010. 3.2

[67] W. S. Grant, R. C. Voorhies, and L. Itti, "Finding planes in lidar point clouds for real-time registration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Tokyo, Japan), pp. 4347–4354, 2013. 3.2

[68] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), pp. 1–4, May 2011. 3.3.1, 4.3.2.1

[69] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics*, vol. 32, pp. 696–722, 2015. 4.1.1, 4.2

[70] Z. Pusztai, I. Eichhardt, and L. Hajder, "Accurate calibration of multi-lidar-multi-camera systems," in *Sensors*, vol. 18, pp. 119–152, 2018. 1.1, 4.1.1, 4.1.3, 4.2, 4.4, 4.4.1, 4.4.1, 4.4.2

[71] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "Calibnet: Geometrically supervised extrinsic calibration using 3D spatial transformer networks," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018. 4.1.1, 4.2

DOI:10.15774/PPKE.ITK.2020.010

[72] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4164–4169, 2007. 1.1, 4.1.3, 4.2, 4.4, 4.4.1, 4.4.1

[73] P. Moghadam, M. Bosse, and R. Zlot, "Line-based extrinsic calibration of range and image sensors," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3685–3691, 2013. 1.1, 4.1.3, 4.2, 4.4, 4.4.1, 4.4.1

[74] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," *IEEE International Conference on Robotics and Automation, (ICRA)*, pp. 3936–3943, 2012. 4.2

[75] H. Alismail, L. Baker, and B. Browning, "Automatic calibration of a range sensor and camera system," in *Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, (Zurich, Switzerland), pp. 286–292, October 2012. 4.2

[76] Y. Park, S. Yun, C. S. Won, K. Cho, K. Um, and S. Sim, "Calibration between color camera and 3D LIDAR instruments with a polygonal planar board," in *Sensors*, vol. 14, pp. 5333–5353, 2014. 4.2

[77] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of RGB camera with Velodyne LIDAR," in *WSCG 2014 Communication Papers Proceedings*, pp. 135–144, 2014. 4.2

[78] S. Rodriguez-Florez, V. F. V., and P. Bonnifait, "Extrinsic calibration between a multi-layer lidar and a camera," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, (Seoul, Korea), pp. 214 – 219, 09 2008. 4.2

[79] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form axdxb," in *IEEE Transactions on Robotics and Automation, vol. 5, no. 1*, pp. 16 – 29, 02 1989. 4.2

DOI:10.15774/PPKE.ITK.2020.010

[80] K. Huang and C. Stachniss, "Extrinsic multi-sensor calibration for mobile robots using the gauss helmert model," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Vancouver, Canada), pp. 1490–1496, 09 2017. 4.2

[81] K. H. Strobl and G. Hirzinger, "Optimal hand-eye calibration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Beijing, China), pp. 4647–4653, 10 2006. 4.2

[82] C. Shi, K. Huang, Q. Yu, J. Xiao, H. Lu, and C. Xie, "Extrinsic calibration and odometry for camera-lidar systems," *IEEE Access*, vol. 7, pp. 120106–120116, 2019. 4.2

[83] R. Wang, F. P. Ferrie, and J. Macfarlane, "Automatic registration of mobile lidar and spherical panoramas," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 33–40, June 2012. 4.2

[84] A. Napier, P. Corke, and P. Newman, "Cross-calibration of push-broom 2d lidars and cameras in natural scenes," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Karlsruhe, Germany), pp. 3679–3684, May 2013. 4.2

[85] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1803–1810, June 2017. 4.2

[86] P. Moulon, P. Monasse, and R. Marlet, "Global fusion of relative motions for robust, accurate and scalable structure from motion," in *IEEE International Conference on Computer Vision, (ICCV)*, (Sydney), pp. 3248–3255, Dec 2013. 4.3, 4.3.1

[87] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, pp. 155–166, 2008. 4.3, 5

DOI:10.15774/PPKE.ITK.2020.010

[88] P. Moulon, P. Monasse, R. Perrot, and R. Marlet, "OpenMVG: Open Multiple View Geometry," in *Workshop on Reproducible Research in Pattern Recognition*, pp. 60–74, 2016. 4.3.1

[89] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: a randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, p. 24, 2009. 6

[90] L. Samuli and K. Tero, "Efficient sparse voxel octrees," *IEEE transactions on visualization and computer graphics*, vol. 17, pp. 1048–59, 10 2010. 4.3.2.1

[91] K. Viktor, S. Erik, and A. Ulf, "High resolution sparse voxel dags," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 101:1–101:13, 2013. 4.3.2.1

[92] L. Matti, J. Anttoni, H. Juha, L. Jouko, K. Harri, K. Antero, P. Eetu, and H. Hannu, "Object classification and recognition from mobile laser scanning point clouds in a road environment," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–14, 10 2015. 4.3.2.1, 4.3.2.1

[93] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, jul 1944. 4.3.4

[94] E.-S. Kim and S.-Y. Park, "Extrinsic calibration between camera and lidar sensors by matching multiple 3D planes," *Sensors (Basel, Switzerland)*, vol. 20, 12 2019. 4.4.2

[95] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. A.2.4

[96] K. Yamaguchi, K. Sakamoto, T. Akabane, and Y. Fujimoto, "A neural network for speaker-independent isolated word recognition," in *The First International Conference on Spoken Language Processing, ICSLP 1990, Kobe, Japan, November 18-22, 1990*, ISCA, 1990. A.2.2

DOI:10.15774/PPKE.ITK.2020.010

[97] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *CACM*, 2017. A.2.5

[98] L. Bottou, "Online learning and stochastic approximations," 1998. A.2.6

[99] Z. Cao, Q. Huang, and R. Karthik, "3d object classification via spherical projections," in *2017 International Conference on 3D Vision (3DV)*, pp. 566–574, 2017. 2.2

[100] L. Zhang, J. Sun, and Q. Zheng, "3d point cloud recognition based on a multi-view convolutional neural network," *Sensors*, vol. 18, 2018. 2.2

[101] W. Wu, Z. Qi, and F. Li, "Pointconv: Deep convolutional networks on 3d point clouds," pp. 9613–9622, 06 2019. 2.2

[102] W. Tan, N. Qin, L. Ma, Y. Li, J. Du, G. Cai, K. Yang, and J. Li, "Toronto-3d: A large-scale mobile lidar dataset for semantic segmentation of urban roadways," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1.1, 2.5.3

[103] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-lille-3d: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification," *The International Journal of Robotics Research*, vol. 37, no. 6, pp. 545–557, 2018. 1.1, 2.5.3

[104] J. Cheng, C. Leng, J. Wu, H. Cui, and H. Lu, "Fast and accurate image matching with cascade hashing for 3d reconstruction," 06 2014. 3

[105] B. Gao, Y. Pan, C. Li, S. Geng, and H. Zhao, "Are we hungry for 3d lidar data for semantic segmentation?," *ArXiv*, vol. abs/2006.04307, 2020. 2.2

[106] H. Radi and W. Ali, "Volmap: A real-time model for semantic segmentation of a lidar surrounding view," *ArXiv*, vol. abs/1906.11873, 2019. 2.2

[107] F. N. Landola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016. 2.2

DOI:10.15774/PPKE.ITK.2020.010

[108] W. Zhang, C. Zhou, J. Yang, and K. Huang, "Liseg: Lightweight road-object semantic segmentation in 3d lidar scans for autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1021–1026, 2018. 2.2

[109] Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu, "Pointseg: Real-time semantic segmentation based on 3d lidar point cloud," *ArXiv*, vol. abs/1807.06288, 2018. 2.2

[110] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015. 2.2