Visual Cross-Modal Mapping, Labeling and Localization

by

Mahdi Elhousni

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

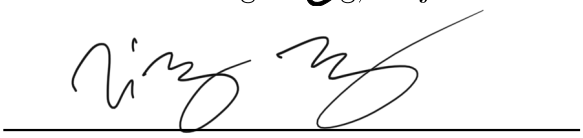
in

Electrical and Computer Engineering

September 2022

APPROVED:

Professor Xinming Huang, Major Advisor



Professor Ziming Zhang, Committee Member

Professor Emmanuel O. Agu, Committee Member

Abstract

Mapping, labeling and localization are central modules to most autonomous robot platforms. Not only do they make it possible for the robot to understand its surrounding environment and its location in it, it also can improve drastically the accuracy of other downstream tasks such as behavioral planning or dynamic object detection and tracking.

While it is possible to achieve reasonable results with traditional dynamic sensors such as IMUs and GPS, modern robotics systems have shown that visual based sensors, mainly laser or camera-based, are extremely well suited for these tasks, since the final localization results returned by such sensors do not only depend on the robot itself, but also on its surrounding environment. While both types of sensors possess numerous qualities that result into a good performance, they also suffer from some limitations. Such limitation can sometimes be overcome by using cross-modal approaches which have the unique advantage of benefiting from the best of both worlds, while still deploying a limited amount of sensors to save on cost and processing power. This thesis shows how we can take advantage of such methods.

First, we start by exploring the cross-modal mapping task in the case of 3D mapping for UAV's. Here, our goal is to be construct 3D maps, using data collected by UAV's equipped with a monocular camera only. Using height prediction and deep learning, we propose a method capable of accurately predicting the height value of each pixel in an input 2D camera image, which can be processed to form a 3D point cloud, thus replacing the traditional and costly "Structure From Motion" (SFM) based methods. This solution achieves state-of-the-art performance compared with all recently published height prediction methods.

Next, we explore the cross-modal labeling task. Labeling is typically necessary for any self-driving car that is traveling in an urban environment. In this work, we tackle the labeling of pre-built point cloud maps by taking advantage of the advancements made in camera based deep learning and show how we can predict relevant road data (such as road boundaries and traffic lane's locations) from 2D camera images, before processing them and projecting them to 3D, to automatically generate height quality labels.

Finally, we explore the case of cross-modal vehicle localization when 3D maps are unavailable. In this case, we use the popular and free OpenStreetMap (OSM) platform, to show how it is possible to accurately localize LiDAR sensors, without the need for point cloud maps or data training. Thanks to a constrained formulation of the popular particle filter method, we are able to track a moving vehicle on OSM, while keeping its position constrained to the road boundaries. This method achieves state-of-the-art performance compared with all satellite or OSM-based methods for LiDAR localization.

Acknowledgements

I would like to express my deepest gratitude to my advisor and mentor, Professor Huang, for his generous support and valuable guidance.

I would like to also thank Professor Zhang for having been always available to answer my questions and propose new ideas, Professor El-Korchi for giving me my first opportunity at WPI and Professor Agu for his thoughtful comments and advice.

Finally, I would like to thank my family for their support and encouragements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Visual Sensors: In this thesis, "Visual Sensors" mainly refers to laser-based or camera-based sensors. Both are uniquely equipped to deal with the mapping, labeling and localization tasks for robots: Laser-based sensors such as LiDARs are capable of providing accurate metric measurements to all the objects present on the line of sight of the sensor, making it uniquely adapted to the mapping aspect of autonomous driving, and as a consequence, capable of accurately achieving the localization task as well. On the other hand, camera-based sensors are capable of capturing rich texture-based keypoints, which can be matched across frames, and used as reference to calculate the displacement of the robot equipped with it or detect road structures and information that can help to safely guide the vehicle.

Unfortunately, both these sensor modalities suffer from some significant limitations: For instance, for the laser-based sensors, the inability to capture colors and textures may sometimes introduce ambiguity during the point matching process or the labeling of pre-built maps, and for the camera-based sensors, sudden changes

in the brightness levels or the absence of an adequate lighting source can cause complete failure of the localization or mapping systems.

Because of these issues, we propose to use cross-modal approaches to deal with them, in order to take advantages of the strengths of both types, at a minimum cost.

Cross-Modal Methods: In this thesis, we mean by "cross modal" methods, the approaches that use a single sensor output and attempt to generate a final result in a different modality. We mostly focus on two major modalities: 3D laser-based and 2D camera-based. Some examples of cross-modal approaches include: localizing 3D point clouds in a 2D camera-based map or using a 2D images to construct a 3D point cloud map. We focus only on single-sensor, single-map methods, which have the advantage of being more cost and resource effective than the traditional sensor fusion approaches, since each sensor is only used when it makes sense to do so.

Mapping: This represents the process of creating a map, which is an abstract and symbolic representation of a space that we are interested in. Mapping, first known as cartography, started in ancient times, with the first map dated as early as the 6th century BCE. Nowadays, and thanks to the advances in the area of visual sensors, maps have taken different shapes and can mean different things depending on the context. The most popular maps are undoubtedly satellite maps and/or maps that derive from satellite imagery: this includes also some popular simplified maps such as Google Maps or OpenStreetMaps. These maps can be very useful for localization, but their accuracy can sometimes be limited. Lately, a new type of maps has become very popular in the autonomous driving industry, which is HD maps. These are point cloud-based maps, which have been labeled with relevant road information and are used for localization, navigation and planning.

Labeling: In this thesis, we mostly talk about pre-built maps labeling, which

means adding relevant semantic information to maps, whether that is done manually or automatically. In the case of 2D camera-based maps, such as satellite maps or OSM, this usually refers to adding semantic labels to mark roads and walkways, in addition to markers of important buildings and/or businesses. On the other hand, in the case of 3D point cloud maps, labeling can be more complex, because in addition to the labels present in the 2D case, more details can be added such as lane positions on the ground or traffic lights location in 3D. These labels have multiple uses: they make it much easier for the robot to understand its surroundings, can help in the localization pipeline and are crucial to make correct and safe behavioral planning decisions.

Localization: As a natural result of mapping and labeling, localization plays a very important role in any autonomous robot system. Findings one's location has been a challenge for human for a very long time: from first using the sun and other famous stars which were known to be stable landmarks, human then evolved to use scientific instruments, such as the astrolabe which was used by navigator during long trips at sea. Eventually, the GPS was invented, representing a major advancement in global localization. However, researchers have lately showed that in order to obtain accurate localization for autonomous robots, visual sensors should be used because they make it possible for the robots to also consider its environment when trying to locate itself, and not only rely on its own movement approximation, which tend to be noisy.

1.2 Motivations

Autonomous robots are on track to become one of the main tools of the future. From self-driving vehicles and UAVs to home assistant robot, autonomy has clearly

become the next major milestone for robotics researchers. For robots to be fully autonomous, it is essential for them to be able to determine their position in their surrounding environment accurately and efficiently, so that they can safely interact with the world.

This as a result pushed the robotics community to invest more time and resources into solving the mapping, labeling and localization tasks, specifically using visual sensors. These tasks are typically co-dependent since building a map implies that we know the distance between its different component and their relative locations, and that obtaining an accurate localization solution can usually be facilitated by using of a map, whether it was built offline or optimized online over a restricted local area.

The main visual sensors that are typically deployed to solve these tasks are cameras and LiDARs. Both sensors have clear advantages when it comes to solving the tasks at hand, such as the ability to capture textures when using cameras, which facilitates the frame-to-frame matching and alignment, or the detection of relevant scene data, and the additional space dimension present in the LiDAR data, which results in accurate and dense representation of the environment. However, current commercial solution focus on using either one sensor or the other across these three tasks, resulting in expansive platforms and cost prohibitive robots for those opting for LiDAR only solutions, and in unstable and sometimes unsafe solutions when using cameras only. A more practical solution would be to utilize one sensor or the other only when the situation where it is used, and the cost implications are justified. Therefore, in this dissertation we explore a set of problems where cross-modal solutions involving LiDARs and camera are being used, in order to provide more cost-effective solution for the future. It is important to note that we make a clear distinction between cross-modal methods and sensor fusion methods, where

the first uses only one sensor at a time during deployment, while the second uses all available sensors at once.

In this thesis, our work will involve three of the main stages of the development of most autonomous robotic platforms: (a) building the map of environment where the robot will be evolving, which is demonstrated using the case of 3D mapping for UAV's using height prediction and cameras. (b) the labeling of pre-built maps, in our case the labeling of 3D point cloud HD maps for self-driving cars using camera data. (c) the localization of the robot in a pre-built and labeled maps, in this case localization of LiDAR equipped cars on camera-based maps such as OSM.

1.3 Contributions

We tackle three of the essential steps in any autonomous robot development: Mapping, labeling and localization.

Building Point cloud Maps Using Cameras and Height Prediction: Building accurate maps that can later be used is very helpful in improving both the accuracy and the efficiency of the localization stack. However, contrary to when using cars, UAV point cloud building can face certain challenges such as the heavy load of the LiDAR sensor when attached to a UAV, leading to shorter battery life and longer mapping process, or when using stereo camera rigs which can suffer from noisy outputs due to brightness changes or the need for very high-resolution cameras which can increase the cost significantly.

On the other hand, it is possible to combine a single monocular camera with deep learning in order to obtain height information that can be used to generate the needed point clouds. Also, thanks to the flexibility of proposed neural network, we are also able to generate semantic labels that cam used to construct semantic maps,

and surface normals which can help in generating 3D mesh maps.

In summary, our key contributions for this work are:

- We propose a triple-branch multi-task learning network, including semantic label, surface normal and height prediction.
- We introduce a denoising autoencoder as a refinement step for the final height prediction results.
- We achieve state-of-the-art performance on two publicly available datasets, and an extensive ablation study shows the importance of each step in the 3D reconstruction pipeline.
- We show through two applications how our height prediction pipeline can be used to reconstruct dense 3D point clouds with semantic labels.

Labeling Point cloud Maps Using Cameras and Deep Learning: Nowadays, the typical next step after building a map is to label it accordingly. This is especially important for self-driving cars that are traveling in an urban environment, since many road and driving laws are defined by road features and visual signs. Currently, the most popular method is to use manual labeling, since it guarantees the level of accuracy needed to safely drive. However, this can be at times both expensive and resource intensive.

Instead, we propose to use deep learning to label our point cloud maps, thus proving a more automated labeling process that is less depending on human operators. In addition to that, we decided to leverage camera data to do the labeling, by first detecting the relevant features of camera images using deep learning, then processing them and projecting them onto the pre-built LiDAR map using a pre-calculated camera-LiDAR extrinsic projection matrix.

In summary, our contributions can be summarized as a collection of algorithms and pipelines aiming to automatically label HD Maps for urban autonomous driving.

Localizing Point Cloud Scans in OpenStreetMaps: As stated before, localization is an essential component of any robots that is autonomously moving in its environment. For self-driving cars, 3D HD point cloud maps represent the gold-standard in the industry today. However, such maps are not available everywhere, and can be very challenging and expensive to build and deploy.

We propose to explore other map platform, although not designed originally for LiDAR localization, but can be used to simulate a LiDAR point cloud map. We focus on using OSM, a free and up to date 2D abstracted map, similar to what can be found when using Google Maps. By combining semantic segmentation with raycasting applied to OSM, we are able to generate simulated point clouds that can be used to compare with and localize the output LiDAR scans. We also make use of the road data present on OSM to design a constrained particle filter that guarantees a bounded localization error.

In summary, our key contributions for this work are:

- We propose a fast and consistent method to generate simulated top view LiDAR images from OSM, and accordingly show how we can use those images to accurately localize LiDAR point clouds in OSM.

- We propose a dual-input particle filter algorithm with an added constraint that the vehicle location must be on the road.

- We demonstrate the state-of-the-art accuracy of our method on the KITTI dataset, by comparing it to other LiDAR cross-modal localization methods by using OSM or satellite maps.

1.4 Publications

- LiDAR-OSM-Based Vehicle Localization in GPS-Denied Environments by Using Constrained Particle Filter. **M Elhousni**, Z Zhang, X Huang. Sensors 2022.

- A Survey on Visual Map Localization Using LiDARs and Cameras. **M Elhousni**, X Huang. 2022.

- Distance Transform Pooling Neural Network for LiDAR Depth Completion. Y Zhao, **M Elhousni**, Z Zhang, X Huang. IEEE Transactions on Neural Networks and Learning Systems. 2021.

- Height Prediction and Refinement from Aerial Images with Semantic and Geometric Guidance. **M Elhousni**, Z Zhang, X Huang. IEEE Access 9. 2021.

- DepthNet: Real-time LiDAR point cloud depth completion for autonomous vehicles. L Bai, Y Zhao, **M Elhousni**, X Huang. IEEE Access 8. 2020.

- A Survey on 3D LiDAR Localization for Autonomous Vehicles. **M Elhousni**, X Huang. IEEE Intelligent Vehicles Symposium (IV). 2020.

- Automatic Building and Labeling of HD Maps with Deep Learning. **M Elhousni**, Y Lyu, Z Zhang, X Huang. The Thirty-Second Annual Conference on Innovative Applications of Artificial (IAAI). 2020.

- Pedestrian Tracking with Gated Recurrent Units and Attention Mechanisms. **M Elhousni**, X Huang. IEEE International Symposium on Circuits and Systems (ISCAS). 2020.

- An interactive lidar to camera calibration. Y Lyu, L Bai, **M Elhousni**, X

Huang. IEEE High Performance Extreme Computing Conference (HPEC). 2019.

1.5 Outline

This dissertation is organized as follows:

Chapter 2: Cross-modal localization, labeling and mapping have been given increasingly more attention lately in the robotics community. This chapter gives an overview of the current state of the art for these three essential tasks.

Chapter 3: Mapping is a cornerstone of any localization system for autonomous robots and point clouds typically represent the best mapping format available today. This chapter discuss a cross-modal approach to generate UAV semantic point cloud maps using 2D camera images only.

Chapter 4: labeling point cloud maps is becoming standard across the self-driving cars industry. Here, we show a cross-modal method capable of using data predicted from camera images and deep learning to automatically label a 3D point cloud map.

Chapter 5: Finally, we tackle the localization task, by taking a closer look at cases where 3D point clouds are unavailable. Thus, we propose a cross-modal localization approach capable of accurately localizing LiDAR point clouds in OSM, using a road-constrained particle filter.

Chapter 6 draws the conclusions.

Chapter 2

An Overview of Visual Cross-Modal Mapping, Labeling and Localization

2.1 Cross-Modal Mapping

Mapping our surroundings is one of the oldest challenges faced by human beings. This reflects the importance of maps in general, and the role they play in localization and planning. For robots, maps represent a crucial tool, and multiple methods for buildings them have been proposed over the years. The most popular approach for robot map buildings is the SLAM approach, where the map is created online, while the robot is moving. This is typically done using a 2D or 3D LiDAR but can also be extended to stereo or satellite cameras. This method has the advantage of making it possible for robots to explore new environments and can be easier and faster to deploy. However, due to the inevitable issues of odometry drifting, and unless more advanced methods and constrains are used, the maps constructed using SLAM will

always lead to an accumulation of the odometry error, which is reflected in poor mapping accuracy.

A more traditional, but stable approach, is to do the mapping offline. This gives us the possibility to optimize the map by recording a sensor output during a pre-defined path and combining it with the pre-calculated ground truth odometry (obtained using an advanced Differential GPS system for example) and other optimization techniques, to make sure that the map is as accurate as possible before deploying it onto the robots. Therefore this method is today standard across multiple industries and disciplines, such as satellites maps using cameras for UAV's or HD point cloud maps using LiDARs for self-driving cars.

Visual offline mapping can take many forms, depending on which sensor is being used. When using LiDAR sensors, typical maps are either point cloud-based or mesh-based, resulting from the concatenations of the successive LiDAR scans using the ground truth. For cameras, they are usually used for top view mapping to construct geo-tagged satellite maps. This maps can then be processed further to produce abstracted 2D maps, such as google maps or OpenStreetMaps.

Visual cross-modal mapping for autonomous robots typically involves the ability to generate point clouds using camera data only, leading to building point cloud maps where either cameras or LiDARs can be used for localization. In this thesis, we explore a new cross-modal approach to visual mapping, by taking advantage of the ability to construct simulated LiDAR point cloud maps, using OSM as an input, combined with other techniques such as raycasting and semantic segmentation. An example of such maps can be seen in Fig. 2.1, in addition other classic visual maps.

However, stereo camera rigs have traditionally been the most popular camera sensor configuration to build point clouds, thanks to their ability to generate depth images that can be converted to point clouds, and concatenated to build a 3D

point cloud map [1] [2] [3] [4]. Although capable of producing impressive results, calculating 3D points position using stereo cameras can lead sometimes to very noisy point clouds, due many reasons: sharp changes in brightness, false keypoint matches between the two left and right views, low resolution cameras etc.

During the last few years with the success of deep learning in computer vision, and as an extension to the depth completion task, the task of depth prediction has gained a lot in popularity, thanks to the denser depth maps that it can produce and its increased resiliency to noisy inputs compared to stereo cameras. Using deep learning networks, researchers can show that it's possible to bypass the use of stereo cameras completely in favor of monocular cameras only, and to predict a depth value to each pixel in an input image, resulting in dense depth maps. Similarly, to the stereo camera case, these depth maps can then be used to generate 3D point cloud maps. First proposed by Eigen al. in [5], the authors present a two-stage approach with two CNNs, where the first one is used to generate a coarse prediction of the corresponding depth map to the the input RGB image, followed by the second network which is tasked with refining the first prediction locally. The use of residual networks was later introduced in [6] combined with the use of the reverse Huber loss, trained in an end-to-end fashion.

The success of monocular depth estimation led researchers to explore other possible cross-modal predictions, especially when it makes it possible to predict 3D data using 2D images as input, which can be very challenging when using computer vision techniques only. One such tasks is height prediction using UAV-captured images. First introduced in [7] using a classical encoder-decoder structure, it was later extended and combined with other tasks using multi-task networks in [7].

Figure 2.1: Examples of Visual Maps. From left to right: LiDAR map, satellite map, OSM and simulated LiDAR map.

2.2 Cross-Modal Map Labeling

Building the map is great for localization, however, it is usually not enough to achieve autonomy. For example, in the case of self-driving cars: After the mapping is done, and for the car to be able to autonomously travel in urban environment, it is necessary to add other relevant semantic information related to the road rules and road structures. Another example is the case of satellite maps: after stitching all the camera images together, it is necessary to label the roads and buildings, if we want to eventually transform them to 2D abstract maps such as OSM. This makes the maps much easier to process and use, since it only keeps the relevant information.

Labeling pre-built maps has always been knows as a tedious task. This is especially the case when dealing with 3D point clouds maps that contains millions of points. The current standard in the industry is to use manual labeling and label the maps using human operators. This is the case because self-driving cars and other autonomous systems today need very accurate labels, such as precise and abstracted waypoint-based labels representing specific road features such as the road boundaries or lanes and traffic sign positions etc. Multiple tools have been developed over the years, where some other point cloud or image processing techniques can be used by the labeler to facilitate the whole process, such as edge detection or clustering.

Some of the popular free manual labeling tools today for 2D images include:

- labelImg: the most popular graphical image annotation tool today, written in Python, specialized in object detection and designed to output labels as XML files in PASCAL VOC format.
- Computer Vision Annotation Tool (CVAT): a video and image annotation tool, capable of generating semantic and object detection labels in multiple popular formats.
- labelme: written in python, and capable of generating similar labels as CVAT.

Although less popular, free LiDAR annotation tools exist too:

- labelCloud: lightweight tool specialized in 3D bounding box labeling for point clouds.
- SUSTechPOINTS: similar to labelCloud but has asome additional features that make the labeling process more flexible.
- Vector Map Builder: an advanced labeling tools designed for HD map labeling for self-driving cars by Autoware. Contains most of the typical road features such as traffic signs, lanes, road markings etc.

Other popular research approaches follow the deep learning semantic segmentation scheme and try to predict a label to each point in the point cloud map [8] [9] or pixel in the satellite map. This can result in good semantic segmentation results in general but remains too dense to be processed effectively for the point cloud case, and can result in blurred and inaccurate borders for the camera case.

Cross-Modal map labeling is currently still an open question, which is why we decided to explore it in this thesis. In the case of HD point cloud maps, by taking

advantage of the unique textures captured with the onboard camera, we can label the point cloud map by detecting relevant areas on the images using deep learning, then process those results and project them to generate abstracted and simplified labels, which are adapted to the self-driving task.

2.3 Cross-Modal Map Localization

Localization is essential to any autonomous robots. Not only does it make it possible to track it, but it also makes it safe for the robots to move, by being able to avoid obstacles for example, when combined with other visual tasks. Localization can be done either in relative way, by using SLAM and always considering the starting point as the origin. This can be considered a "local" form of localization. On the other hand, "global" localization typically involves the use of a pre-built map.

Visual map localization can typically be divided into two major steps: place recognition and metric map localization. We show in Fig. 2.2 a Block Diagram of the typical Visual Map Localization pipeline: First, the Visual Place Recognition stage where the map is rasterized to produce a database of geo-tagged samples for more efficient processing. This is followed by the encoding into feature vectors of both the sensor output and all the samples from the map. A nearest neighbor search is then used to find the closest map sample to the sensor output, and thus produce a guess at the initial position of the vehicle in the map. Next is the Metric Map Localization stage where a registration algorithm is used to align the sensor output with the map, making it possible to track the vehicle.

For both of the major steps mentioned above, we will first start by exploring the traditional LiDAR and camera based methods, before the discussing the cross-modal methods.

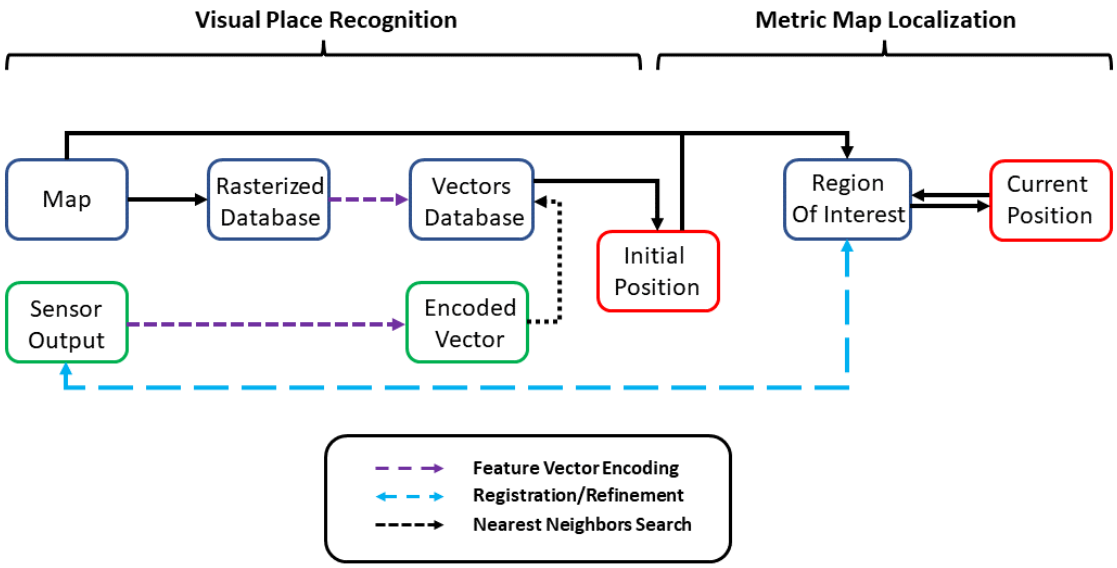


Figure 2.2: Visual Map Localization Block Diagram.

2.3.1 Place recognition

First, the vehicle (or robot) must find its initial location on the map, especially when no other sensor such as a GPS is available to provide an initial guess or a region of interest. The solution in this case is to use the Visual Place Recognition approach, where using only the input of our visual sensor and an intermediate representation, we can find the best match in the pre-built map. Visual Place Recognition is typically approached differently, depending on the sensors that is being used.

Camera-based place recognition (also called Camera Cross-View Localization) can be very challenging due to the large difference in viewpoint between the images collected by the ground vehicles, and the images extracted from the aerial maps. Fig. 2.3 shows different representation of same camera frame that can be used to solve this task. Because of the challenges mentioned above, most of the popular and successful methods rely on deep learning. This was first demonstrated in [10], by

Figure 2.3: Front view camera frame (top), followed by the same area in a satellite map (middle left) and in OSM (middle right). The final image is the polar projection of the satellite crop (bottom).

relying on a Faster R-CNN [11] to detect buildings then match them using a siamese network trained using the contrastive loss. Note that both the contrastive and triplet losses are very popular when trying to solve this challenge as we will see in the following cited publications. This was improved upon the following year in [12] by simplifying the first stage from object detection to CNN feature extraction followed by an encoding stage using the NetVLAD architecture [13]. The two previously cited works established a common basis which was typically used as a starting point to the methods that followed.

In [14], the authors proposed to attach a color encoded orientation map to the input queries during training and testing, which seems to improve the accuracy on the most challenging metrics. The importance of orientation alignment was furthermore represented in [15] where the authors showed that training using images that were pre-aligned first in terms of orientation will produce a siamese network that is capable of producing activation maps that perform better at pointing similar objects in different views. The activation maps, which were produced using GRAD-CAM

[16] can also be used during testing to approximate the orientation that best aligns the two views. Another approach to improve the accuracy of a siamese network trained for cross-view geo-localization is to take advantage of the results of traditional semantic segmentation networks and include them in the data augmentation procedure during training: this was done by removing different segmented objects in the ground images, to make the network more robust to temporal changes in the images. This, combined with a multi-scale attention module, produces better ranking and matching results.

The authors of [17] introduced the use of optimal features transport [18] to facilitate the extraction of similar features in both views. This was implemented in a way that allowed the end-to-end training on the network and showed great improvements across all metrics.

While most works use some sort of variation of the contrastive or triplet losses, the authors in [19] proposed their own metric, dubbed Soft Exemplar Highlighting Loss. In their formulation, this loss, combined with a polar transform applied to the aerial images to reduce the viewpoint gap, was used to assign different weights to the training examples depending on their difficulty, in an effort to emphasize meaningful hard samples and remove problematic ones. Another typical assumption in most cross-view geo-localization works in the literature is the one-to-one matching assumption between aerial and ground images. This does not always hold during testing and was the main motivation in [20]: in this work, the authors did not only attempt to predict the matching score between two samples, but also using a regression branch, predicted a latitude and longitude-based offset between the two inputs. Also, in addition to the triplet and regression losses, the authors introduced an IOU-based loss to better learn from semi-positive sample (meaning aerial samples with a non-zero offset).

Lately, because of the success of attention models in computer vision [21], more works have been trying to use the attention mechanisms [22] and the Transformer architecture [23] to solve this task, starting with [24] where the authors proposed to use what they call a Spatial-aware position embedding module to process both the ground and polar transformed aerial images, tasked with encoding the relative positions among object features extracted by the backbone network. This module consists of a max pooling block, followed by two fully connected layers in order to select the most important features. In [25] the authors proposed an architecture where first, for both views, 1D learnable encodings were combined with a set a CNN extracted features, before being fed into what the authors called a Layer-To-Layer Transformer: basically, a transformer with skip connections between timesteps. In [26], the authors attempted a pure transformer architecture which does not make use of CNN's as pre-processing step for feature extraction: this was done by following a two-stage procedure, where in the first step, two traditional Vision Transformer (ViT) architectures were trained using the triplet loss to generate embedding features for both street and aerial views. In the second stage, the aerial attention map generated from the first stage was used as guidance to crop and zoom-in on the most relevant portion of the image. This new generated aerial image was then used to finetune the aerial embedding using another ViT.

LiDAR place recognition has become very popular since HD point cloud maps have become the norm for many autonomous driving vehicles. Fig. 2.4 shows different depiction of the same LiDAR frame that can be used to solve this stage. The earlier attempts to solve this task tried to capitalize on the advances in keypoint detection and matching for point clouds. In [27], based on a random sampling procedure, keypoints were selected and encoded using a variation of the gestalt descriptor [28], before being matched using the nearest neighbor voting approach. In

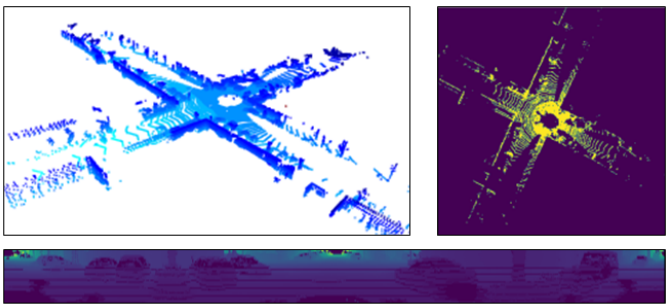


Figure 2.4: An example of LiDAR frame depicted as a 3D point cloud (top left), a BEV projection (top right) and a panoramic projection (bottom).

[29], the keypoints based place recognition task was solved by taking advantage of the geometrical relations between points: after extracting features using points of high curvature, the authors encoded the point cloud data into a 2D histogram based on the distances between them and their co-bearings, which resulted in a signature that was later used to match the point cloud with other scenes using the Approximate Nearest Neighbor Search. In [30] the authors proposed SegMatch, an algorithm based on segmentation results which were then used to construct feature descriptors. The matching of segments was achieved following a two-step approach: first using a random forest classifier, followed by a geometrical verification using RANSAC [31]. This was eventually extended in [32], by augmenting the SegMatch descriptor with a handcrafted spatiotemporal descriptor which was constructed following two stages of spatial and temporal feature pooling.

The authors of [33] proposed to take advantage of the intensity field returned by the LiDAR sensor to construct an intensity-augmented 3D keypoint descriptor named ISHOT, which was matched following a strategy combining probabilistic voting and nearest neighbor search. A similar method was used in [34] where the intensity field was central to the approach, but in this work, the intensity data was first projected to the 2D image space using a panoramic projection, before using a

traditional computer vision (CV) keypoint extractor and encoder, in this case ORB [35]. This was followed by a traditional CV matching procedure relying on PnP [31] and BoW [36]. Projecting 3D point clouds to 2D in order to take advantage of traditional CV techniques is a common method used when processing LiDAR data. Another method that utilizes this principle was proposed in [37], but this time, the Bird Eye View (BEV) projection was used. An appropriate descriptor named BVFT was proposed, and similarly to the previous discussed method, a BoW matching method was deployed, followed by ICP [38] refinement. Lately, approaches relying on encoding the full scan into some sort of compressed representation have become more popular, which resulted in the development of the popular ScanContext [39] encoder. In this work, the authors proposed a two-step process which results in a compressed and viewpoint invariant 3D tensor, where the position, orientation and height of each point were encoded. The resulting global descriptors were matched using a simple similarity score.

Lately, deep learning has been increasingly used to try and solve the LiDAR place recognition task, first by including it into semi-handcrafted methods such as [40, 41], where the point clouds were first pre-processed using a histogram based method to produce rotation invariant representations, which were then fed to a siamese neural networks, trained using the contrastive loss function in order to generate similar vector representations for similar point clouds. Likewise, the authors in [42] followed a similar strategy by first generating a rotation invariant representation, based on the semantic segmentation of the overhead projection of the point clouds, followed by a siamese neural network for feature extraction, and a MLP for similarity prediction. Another semi-handcrafted method was proposed in [43]. Here the authors started by generating an overhead projection of the point clouds, then processed them in order to generate two types of descriptors: a global one, gener-

ated using the NETVlad architecture [13], and a feature based one, generated using the SuperPoint [44] architecture. Both descriptors were combined, and matching was achieved using the SuperGlue algorithm [45]. End-to-end methods attempting to solve this problem have been proposed too, notably in [46], based on the combination of a graph neural layer with an optimal transport layer. The network was then trained using a distance-based matching loss that rewards closer points and penalize farther ones, instead of the typical binary ground truth used for matching. Graph neural networks were also used in [47]. Here the graph was generated based on semantic segmentation results of the point clouds, then fed into a graph neural network with the following steps: node embedding, graph embedding and graph-graph interaction.

Because of the scarcity and lack of availability of accurate HD point clouds maps, cross-modal approaches can be very useful, and researchers have been trying to solve the place recognition challenge when having a LiDAR point cloud as input by using freely available and sometimes opensource maps such as satellite maps or OpenStreetMaps (OSM). Solving this typically involves the use of deep learning since we not only have to deal with the gap in modality, but this is exacerbated by the gap in viewpoint too. Lately, the authors in [48] proposed a method where based on a predicted occupancy map from a satellite image, raycasting was used to generate simulated overhead LiDAR images, which were then combined with the overhead projections of the sensor inputs and fed into a DGCNN architecture [49] to predict a transformation offset, but also in a NetVLAD architecture to generated embeddings that could be used for place recognition. In [50], it was OSM that was used as main map. By taking advantage of the buildings and roads information's, the authors used raycasting to generate simulated overhead LiDAR images, which were matched with the LiDAR sensor's input using the Scan-Context [39] descriptor

discussed previously.

2.3.2 Metric map localization

Once the initial location is found, the robot can now start to navigate the map, while we track its movements as accurately as possible. We call this step Map Metric Localization which is achieved by enforcing both a temporal consistency between the subsequent frames provided by the input sensor, in addition to a spatial consistency, which is guaranteed by matching with the map's region of interest and can be seen as a correction to the first transformation that was calculates using the sensors inputs only. This second step runs in a recurrent fashion, as long as the localization error stays at a reasonable level, guaranteeing enough overlap between the sensor's outputs and the map's region of interest.

When it comes to metric map localization using cameras, the task typically suffers from the same issues faced when attempting to first solve the place recognition step, meaning the drastic difference in viewpoint. In addition to that, we now also must deal with classical odometry and map localization challenges such as the accumulation of positional error or the lack of sufficient overlap between the map and the sensor output. One of the earliest solutions was proposed in [51] based on the graph representation of the road network in OSM and the input of two cameras. Using the same setup, in [52], the proposed approach relies on the buildings structure represented in OSM, rather than the road network. Here, buildings geometry was extracted from the input point clouds using filtering and clustering and scored against the OSM buildings data using a 2D scoring function based on orthogonality, in order to keep track of the vehicle position in OSM.

The authors in [53] chose to use satellite maps instead. By using the depth information than can be generated using a stereo camera rig, the authors trained a

Ground-Satellite Dictionary to be able match features from both views. Localization was achieved by first extracting features and their feature vectors from the ground views, then queering up the aerial images containing features with the closest feature vectors. In [54], only a single monocular camera was used to find the vehicle position in the satellite map. This was achieved by training a siamese neural network to predict a similarity score between ground images and aerial regions of interest (ROI). The predicted similarity score was then used to update the weights in a particle filter [55] in order to localize the monocular camera in the map.

While multiple methods rely on extracting and matching visual features, others proposed to rely on extracting and matching visual landmarks instead. The landmarks used in [56] were poles. The authors first started by constructing a pole map by detecting poles using the disparity image that can be generated using stereo cameras, combined with edge detection and logistic regression. Subsequently, localization was achieved by detecting poles in the same way, and then using that information to update a particle filter, which was coupled with a Kalman Filter [55] for additional sensor fusion. As an extension to [24], another sensor fusion method was proposed in [57] to take advantage of the noisy GPS measurements that are usually available: using a modified triplet loss function, the authors argue that the rough GPS measurements of the ground and polar transformed aerial images in a pre-defined region of interest could be used to calculate a weight capable of scaling the contribution of each pair of images accordingly. The effectiveness of the method was later demonstrated by combining it with a particle filter. The same authors proposed later a more advanced method in [58] where in addition to the popular polar transform, they introduce a geometry-constrained projective transform that results in much more realistic ground looking images. In addition to that, a new fine-grained cross-matching solution was proposed: Based on the prediction of their

baseline network, a corresponding aerial image was selected, tagged with a rough GPS location. The authors then proceed to transform the aerial image using their proposed projective transform and a set of pre-defined positions. Finally, the SSIM similarity loss function was used to select the best matching one.

One final camera map representation, which is still sometimes used (although not very popular due to its sparsity), is Google StreetView. The authors of [59] transformed the closest panoramic image available in Google StreetView according to GPS to a set of eight rectilinear images, followed by a traditional homography-based feature matching, using SIFT features, to keep track of the vehicles position.

LiDAR localization using a pre-built map has been the most successful approach for autonomous driving vehicle in terms of accuracy. This is due to the rich amount of detail typically available in such maps, since every area is the result of multiple scans that were aligned and concatenated. 2D LiDAR localization has a long and rich amount of published research in the robotics community, especially for indoor scenarios. In contrast, we will mainly focus on 3D LiDAR which are more adapted to outdoor scenarios and are typically available in modern autonomous driving cars.

Earlier methods such as [60] relied on sensor fusion and particle filters to localize LiDAR equipped vehicles in point clouds maps. In [61] a solution to LiDAR map localization was proposed through the design of handcrafted features that could be matched across the map and the sensor input point clouds and which were based on the histogram of the frequency of points clusters sizes. Some works such as [62], only relied on the intensity field returned by the LiDAR sensor, and in [63] a method combining features and filters to deal with noisy LiDAR data due to rainy conditions was discussed: Feature extraction is based on the position and reflectivity of each point, followed by a combination of a particle filter (to process for vertical features) and a histogram filter (to process for ground features).

The authors in [53] drew inspiration from the NDT odometry algorithm [64] and proposed to use Gaussian Mixture Maps (GMM). By using the ground plane xy as a 2D grid, each cell in the grid can be filled using a one-dimensional Gaussian mixture that models the distribution over that cell's height. An efficient multi-resolution branch-and-bound search was used to match cells and align the sensor point cloud with the map. Compressing the 3D map into a 2D representation to achieve faster results has also been explored in [65] which proposed to use buildings footprints to generate a simplified segments-based map, which was then combined with NDT to solve the localization challenge.

If the authors are using the full 3D map, they sometimes have access to labels such as traffic lights or lanes, which can aid in the localization process. For example, the authors in [66] proposed to take advantage of the lane information to achieve lane-level accuracy using LiDARs. Roads were extracted mainly based on their height information, then lanes were detected using the intensity field returned by the LiDAR sensor. Finally, the map matching and pose tracking were achieved using a particle filter. [67] is an extension of the lane based localization but instead uses traffic signs (extracted using the points normals) as landmarks, and in [68], authors used poles and curbs to localize the vehicle in a HD map. A pole cost function and a curb cost function were proposed and fused to generate a rough guess at the vehicle's position.

Deep learning is very popular when talking about place recognition, so naturally researchers try to use it with this task as well. First, some methods only rely on the results of other neural networks to improve their localization pipeline: In [69] a system that combines LiDAR odometry with segmap's place recognition to reduce the LiDAR position drift was proposed. This was achieved by taking advantage of the matched segments and aligning them in order to finetune the transformation

obtained by the LiDAR odometry. In [70] the authors proposed a multi-vehicle collaborative approach aided by semantic segmentation. In the case of two vehicles for example, the proposed system enforces a geometrical and semantic consistency matching across the inputs of both vehicles. This produces a weighting matrix which subsequently used in an Expectation-Maximization algorithm to align the point clouds with the map.

End-to-end methods have also been proposed: the authors in [71] used a siamese network, which processes the panoramic projection of different cues generated from the point clouds (semantic labels, point locations etc.) and predicts two quantities: a similarity score representing the overlap between both inputs and a relative yaw angle. The predictions were combined with a particle filter to achieve LiDAR map localization. In [72], the authors proposed a network that attempts to learn the residual value between a traditional localization system and the ground truth. Relevant features were first extracted and fed into a miniPointNet [73] to generate their corresponding feature descriptors. A cost volume was then constructed in the solution space (x, y, z) and regularized with 3D convolutional neural networks. Additionally, an RNN branch was added to the network structure to guarantee the temporal smoothness of the displacement's predictions. Following the latest trends, [74] proposed to use attention mechanisms to solve the self-localization challenge in a point cloud HD map. The localization process was split in two phases: first, a landmarks association step where points association was achieved by combining kNN and local attention, followed by a global point cloud registration where the associations made in the first step were fed into a pose regression network which mainly contains a global attention/pooling layer followed by a MLP.

Cross-modal approaches have also been proposed to deal the metric map localization step. We will focus on methods that attempt to localize LiDAR point clouds on

camera-based maps, such as satellite maps or OSM(-like) maps: In [75], the authors proposed a handcrafted 4-bit semantic descriptor, based on buildings and intersections positions in OSM cropped images and LiDAR semantic range images, which was combined with a particle filter to achieve global map localization. This work showed that semantic segmentation can be a great tool to break the multi-modality issue. For satellite maps, the authors of [76] also leverage the correlation of the semantic segmentation results from both the LiDAR point cloud and the satellite images in order to optimize the soft cost function of a particle filter. More advanced deep learning-based methods have recently been proposed: In [77] and [78], a Generative Adversarial Network (GAN) [79] was trained to generate synthetic top view LiDAR images based on input satellite crops. The synthetic and real LiDAR images were then both fed to a neural network to predict the value of the displacement between frames in a cascaded fashion, by first predicting the rotation value, then using that to predict the translation offset.

It is also possible to localize camera data in LiDAR maps. Stereo cameras are the natural pick when trying to localize video data in LiDAR maps because we can process their output to transform the data from 2D to 3D, which makes its alignment with point clouds much easier: the method proposed in [80] attempted to localize a stereo camera in a 3D LiDAR map, in this case by first relying on visual odometry to provide an initial guess at the transformation, before fine tuning it, using the synthetic and stereo depth maps residual alignment.

More challenging is the task of localizing monocular camera images in 3D point cloud maps since they do not contain any depth or 3D information by definition. Some early attempts include a method that was proposed in [81] based on the idea of correlation between synthetic maps views and camera images. Here, the synthetic images were populated using the intensity returned by the LiDAR sensor,

instead of the depth data, which as a result produces synthetic images with a closer visual aspect to the camera images. Using a discreet number of possible synthetic images located around an initial pose guess, the authors used the Normalized Mutual Information (NMI) to evaluate them and determine the correct vehicle pose.

As with all other challenges, solution involving deep learning were soon showing great potential: In [82] and [83] the authors proposed CMRNET, a neural network capable of processing as input a RGB camera image and a synthetic depth map image and predicts as a result the relative pose between both inputs. A modified version of PWC-Net [84] (an optical flow prediction network), was used, and the original method was later improved with the incorporation of PnP and RANSAC as a post-processing step.

Chapter 3

Building Point Cloud Maps Using Cameras and Height Prediction

3.1 Introduction

Aerial imagery analysis was known as a very tedious task owing to the low quality of the acquired images and the lack of some appropriate automated process that could extract the relevant information from the data. Fortunately, recent advances in computer vision have made it possible to directly extract predefined patterns from the images, by applying some carefully designed algorithms. Moreover, deep learning brings in a new revolution to the field of aerial imagery analysis with more intelligence and better accuracy. As a result, multiple deep learning challenges related to aerial imagery processing, such as semantic segmentation [85, 86] and object detection [87, 88], have been routinely featured each year by the geoscience and remote sensing (GRSS) community [89],[90],[91].

This work focuses on the height prediction task that is to predict and reconstruct the corresponding height map, or in other words, predict the height value for every

pixel in the input aerial image. Predicting such height maps can be very useful in the subsequent task of 3D reconstruction. By obtaining the accurate height of each building or structure appearing in the input images, 3D models can be generated as an accurate representation of the surrounding world. These 3D models are crucial for GPS-denied navigation, or other fields such as urban planning or telecommunications. Theses reconstructions are traditionally done using Structure from Motion (SfM) [92, 93] technique with stereo camera rigs, which can be very sensible to noise and changes in lighting condition.

Motivation: 3D reconstruction using SfM has long been known as the go-to method to obtain 3D information from camera data. However, with the success of deep learning, the encoder-decoder models have shown impressive result in the monocular depth prediction task for autonomous driving cars. In the case of height prediction for UAV's, we propose to deploy a multi-task network to provide high quality height maps, using only a single RGB image as input. The additional branches that will be constructed will be tasked with predicting the semantic labels and surface normals corresponding to the input image and feeding all this additional information to the main branch in order to improve the height prediction. The reason we chose semantic labels and surface normals are as follow: for the first, knowing the semantic label of a certain pixel can help us get better height approximating, for example: a pixel that is labeled as ground will have a different height than a pixel that is labeled as building. For the second, they make it possible to improve the height predictions at the edges, where the surface normals change drastically, thus signaling the switch from one surface to another.

Approach: For the task of height prediction from aerial images, we propose a multi-task learning framework where additional branches are introduced to improve height prediction accuracy. Previous works have showed that multi-task learning

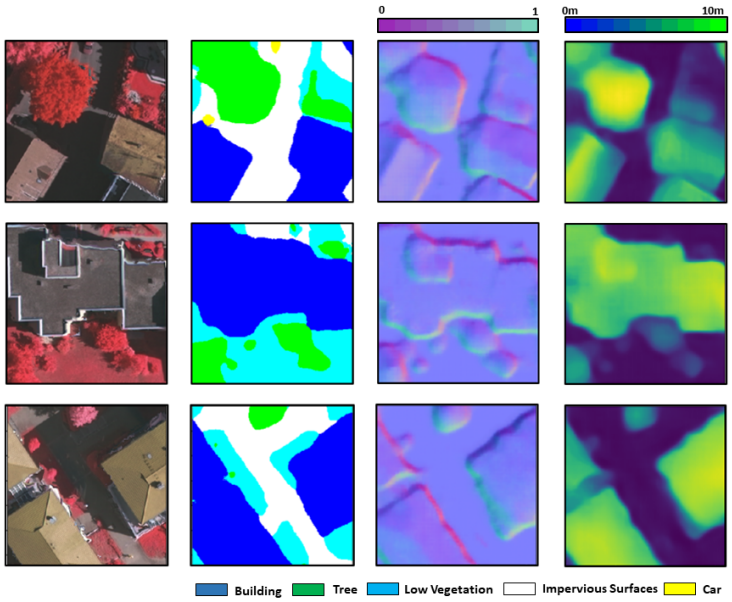


Figure 3.1: The outputs of our multi-task network. From left to right: The input RGB image, the output semantic labels, surface normals and height predictions.

helps improving the accuracy of height prediction networks by including semantic labels [94]. We propose to add a third branch to the multi-task network which will be devoted to predicting the surface normals, as shown on Fig. 3.1. In this configuration, the main height prediction branch will have access to both semantic and geometric guidance, improving the results of the height prediction network.

However, since the input is only an aerial image, our predictions sometimes can be noisy due to artefacts such as shadows or unexpected changes in color. Therefore, we introduce a refinement network which is a denoising autoencoder taking the outputs from the prediction network, removing the noise present in the prediction and producing a higher quality and more accurate height map. By combining these two steps, we are able to produce results that surpass the current state-of-the-art on multiple datasets. We are also able to produce reasonable semantic labels and surface normal predictions without additional optimizations.

In summary, our contributions in this work are the following: (a) We propose a

triple-branch multi-task learning network, including semantic label, surface normal and height prediction. (b) We introduce a denoising autoencoder as a refinement step for the final height prediction results. (c) We achieve state-of-the-art performance on two publicly available datasets, and an extensive ablation study shows the importance of each step in the 3D reconstruction pipeline. (d) We show through two applications how our height prediction pipeline can be used to reconstruct dense 3D point clouds with semantic labels.

3.2 Related Work

Multi-task learning: This learning framework aims at optimizing a single neural network that can predict multiple related outputs, each represented by a task-specific loss function [95]. Lately, this approach has become increasingly popular, especially in the area of autonomous driving cars, where multiple outputs (such as object detection, semantic segmentation, motion classification) are derived simultaneously from the input of camera images [96, 97].

Height prediction from aerial images: This task has received a considerable amount of attention by the deep learning and remote sensing communities, especially after the use of UAVs to collect aerial images has become widely accessible. The goal here is to generate a height value for each pixel in an input aerial image. In works such as [7],[98],[99], deep learning methods such as residual networks, skip connections and generative adversarial networks are leveraged in order to predict the expected height maps.

Other works such as [94, 100] proposed to reformulate the task as a multi-learning problem, by introducing neural networks capable of predicting both the height maps and the semantic labels simultaneously. These works showed that both outputs can

benefit from each other, during the simultaneous optimization process of the multi-task network. We choose to extend that formulation by including a third branch in our network tasked for predicting surface normals, which was inspired by previous works [101, 102] in the depth prediction task for autonomous driving cars. Surface normals are also known to be extremely useful during 3D reconstruction tasks and are required for surface and mesh reconstruction algorithms such as the Poisson surface reconstruction algorithm [103] or the Ball pivoting algorithm [104].

Denoising Autoencoders: Removing noise from images is a traditional task in computer vision. Over the years, many techniques were presented in the literature which can be broadly divided into two categories [105] : spatial filtering methods and variational denoising methods. The spatial filtering methods can either be linear, such as mean filtering [106] or Wiener filtering [107, 108], or nonlinear such as median filtering [109] or bilateral filtering [110]. These filtering methods work reasonably well but are limited. If the noise level becomes too high, these methods tend to lead to over-smoothing of the edges that are present in the image. On the other hand, in variational denoising methods, an energy function is defined and minimized to remove the noise, based on image priors or the noise-free images. Some popular variational denoising methods include total variation regularization [111], non-local regularization [112] and low-rank minimization [113].

Lately, a new trend based on deep learning autoencoders has shown great potential on image denoising. Autoencoder is a class of popular neural networks that has shown to be very powerful across multiple tasks such as segmentation of medical imagery [114], decoding the semantic meaning of words [115] or solving facial recognition challenges [116]. For our task, the most useful type of autoencoders available in the literature is the denoising autoencoder. As shown in [117], autoencoders can be trained to remove noise from an arbitrary input signal such as an image.

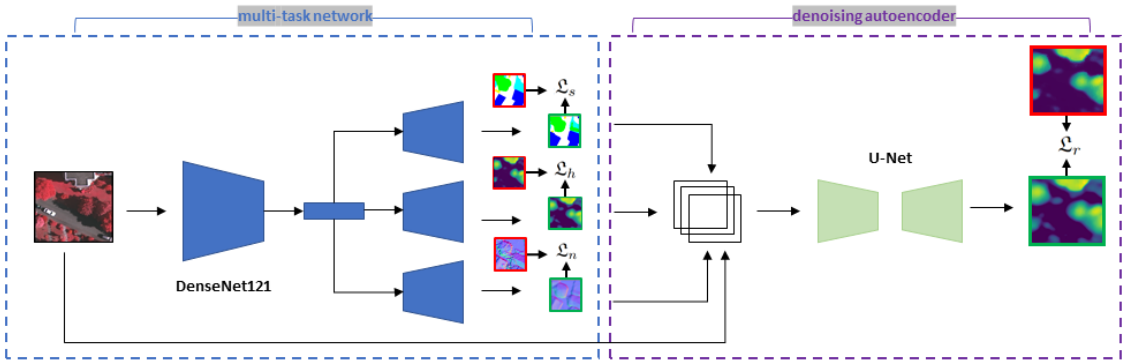


Figure 3.2: Our two stage height prediction and refinement pipeline. We use DenseNet121 to extract a global feature vector from the input aerial images, which is used to predict the normals map, semantic labels and a first guess at the height map (first stage, in blue). These results are concatenated with the input aerial image and fed into a denoising autoencoder to generate the refined final height map (second stage, in purple). Red boxes represent the ground truth, while green ones represent the networks predictions.

We propose to use denoising autoencoder to refine the height predictions from the multi-task learning network.

3.3 Method

3.3.1 Problem Formulation

Our main objective is to predict an accurate height map using only a monocular aerial image as input. We attempt to do so by constructing a two-stage pipeline, where two different networks are cascaded in serial. The first stage of our pipeline is a multi-task learning network, where the main branch is tasked with predicting preliminary height images, aided by semantic and surface normal information that was extracted by two additional branches of the neural network. The second stage can be seen as a denoising autoencoder: All the predictions from the multi-task network are concatenated and fed into the autoencoder, in order to deal with noisy

areas remaining in the height results from the first stage. This effectively produces sharper images that are closer to the ground truth. An overview of the full pipeline can be seen in Fig. 3.2.

Fundamentally, the height prediction task is a non-linear regression problem that can be formulated as:

$$\min_{\psi \in \Psi} \sum_{i} \ell(\mathbf{y}_i, \psi(\mathbf{x}_i)) \tag{3.1}$$

where $\psi : \mathcal{X} \rightarrow \mathcal{Y}$ denotes the height prediction mapping function from the feasible space Ψ, $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ denotes a loss function such as the least-square, $\mathbf{x}_i$ is the input aerial image and y_i is the output height map.

Predicting height only using a single branch neural network is possible. However, previous works such as [94, 100] showed that including additional branches to predict other related information such as segmentation labels can be beneficial for both tasks. In our case, in addition to predicting the height maps, we also predict semantic labels and surface normals, which provide semantic and geometric guidance by augmenting the main height prediction branch with information from the semantics and surface normal branches. More details can be found in the height prediction section below. Hence, our ψ function can now be defined as:

$$\psi(\mathbf{x}_i) = \{\mathbf{P}_h, \mathbf{P}_s, \mathbf{P}_n\} \tag{3.2}$$

where $\mathbf{P}_h$, $\mathbf{P}_s$ and $\mathbf{P}_n$ are the height, semantic and surface normal predictions respectively, that are trying to approximate $\mathbf{y}_i = \{\mathbf{P}^*_h, \mathbf{P}^*_s, \mathbf{P}^*_n\}$ where $\mathbf{P}^*_h, \mathbf{P}^*_s$ and $\mathbf{P}^*_n$ are the height, semantic and surface normal ground truth respectively. Finding a good approximation of the ψ function can be seen as the first stage in our proposed method.

Regression problems such as the one we are facing are difficult to solve due to the high number of values expected to be predicted. This makes our height prediction $\mathbf{P}_h$ noisy by definition, so the use of denoising autoencoders is appropriate in this situation.

First, we can write: $\mathbf{P}_h = \mathbf{P'}_h + e$ where $\mathbf{P'}_h$ is the clean height value, and e the noise inherent to our approximation of the function ψ. By introducing a denoising autoencoder, we can approximate the noise function γ such as $\mathbf{P}_h = \mathbf{P'}_h + \gamma(\mathbf{z_i})$, where $\mathbf{z_i}$ is the concatenation of the outputs of ψ with the input aerial image x_i. This makes it possible to re-write equations (3.2) as $\psi(\mathbf{x}_i) = \{\mathbf{P'}_h + \gamma(\mathbf{z_i}), \mathbf{P}_s, \mathbf{P}_n\}$. We can also now define the objective of the second stage of our method such as:

$$\min_{\gamma \in \Gamma} \sum_i \ell(\mathbf{P}^*_h, \mathbf{P}_h - \gamma(\mathbf{z_i})) \tag{3.3}$$

In this paper, our goal is to approximate both function ψ and γ by using two cascaded deep neural networks.

3.3.2 Height Prediction Network

We solve the height prediction problem via multi-task learning where, in addition to the main height prediction, semantic and surface normals predictions are conducted too. We found that by re-routing the information in the semantic and surface normal branches to the main height branch, our neural network can learn to predict more accurate height values, especially around the edges.

Fig. 3.3 shows our multi-task learning network architecture. We propose a convolutional neural network where we combine a pretrained encoder (tasked with extracting relevant features from the input aerial images), with three inter-connected decoder branches, one for each type of predictions respectively. We chose to use

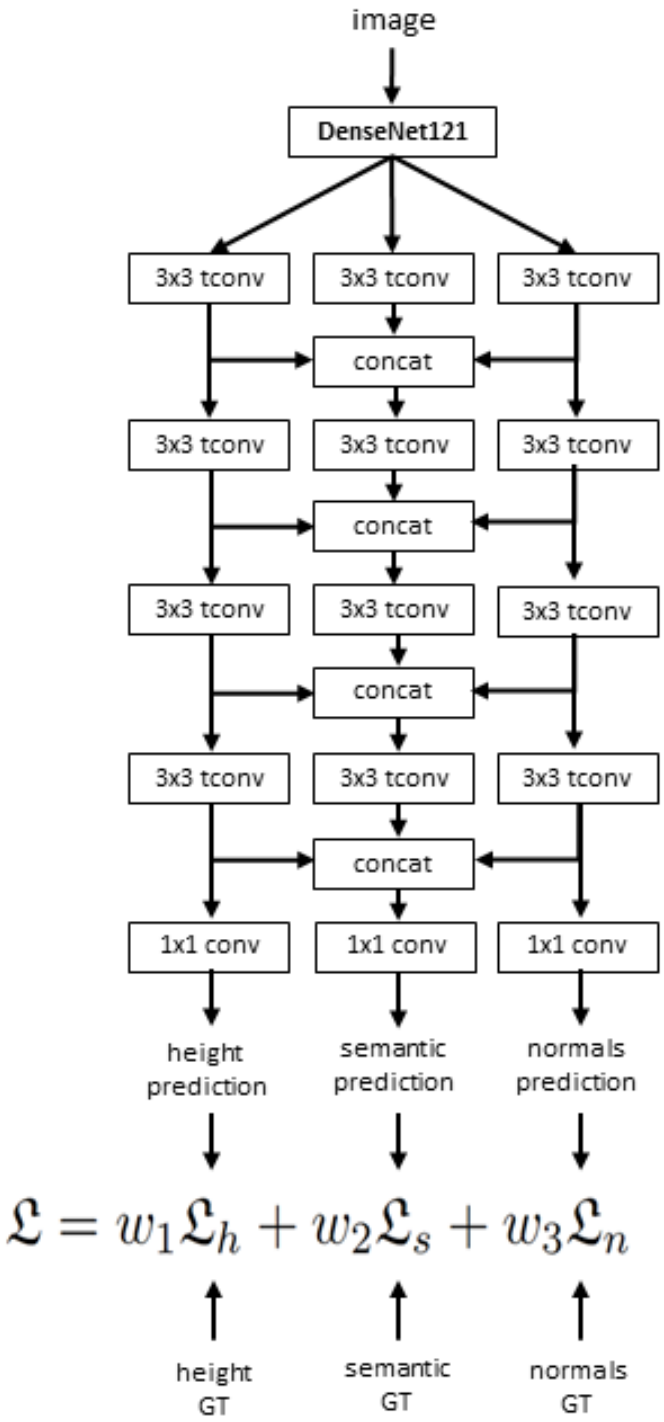


Figure 3.3: Architecture of our multi-task learning network for height, semantic and surface normals predictions. Note that each tconv block is followed by the ReLu function and drop out layers are inserted after each tconv layers in the main height prediction branch.

a DenseNet121 network, pretrained on ImageNet, as our main encoder. We show later in the experimentation section that DenseNet121 yields the best accuracy when compared to other popular architectures. Our decoder on the other hand is inspired by [6] and are characterized by being able to reconstruct the expected predictions efficiently. We list in Table 3.1 the different layers that we used.

	Layer	Output Size
Encoder	DenseNet121	(10,10,1024)
Decoder	$DeConv_1$	(20,20,1024)
	$Concat$	(20,20,3072)
	$Conv_{11}$	(20,20,1024)
	$Conv_{12}$	(20,20,1024)
	$DeConv_2$	(40,40,512)
	$Concat$	(40,40,1536)
	$Conv_{21}$	(40,40,512)
	$Conv_{22}$	(40,40,512)
	$DeConv_3$	(80,80,256)
	$Concat$	(80,80,768)
	$Conv_{31}$	(80,80,256)
	$Conv_{32}$	(80,80,256)
	$DeConv_4$	(160,160,64)
	$Concat$	(160,160,192)
	$Conv_{41}$	(160,160,64)
	$Conv_{42}$	(160,160,64)
	$DeConv_5$	(320,320,32)
	$Concat$	(320,320,96)
	$Conv_{51}$	(320,320,32)
	$Conv_{52}$	(320,320,32)
	$Conv_{out}$	(320,320,1)

Table 3.1: Height prediction network details.

This network is optimized by using a multi-objective loss function defined as:

$$\mathfrak{L} = w_1\mathfrak{L}_h + w_2\mathfrak{L}_s + w_3\mathfrak{L}_n \tag{3.4}$$

where $\mathfrak{L}_h = \frac{1}{n}\sum_{i=1}^{n}(P_h - P_h^*)^2$, $\mathfrak{L}_s = -\frac{1}{n}\sum_{i=1}^{n} P_s^* log(P_s)$, $\mathfrak{L}_n = \frac{1}{n}\sum_{i=1}^{n}(P_n - P_n^*)^2$

and w_1, w_2 and w_3 are weights set up according to the training dataset and the scale of each loss function: We found that by using weights that keep all the loss functions at the same scale, the CNN would converge faster and achieve higher final accuracy levels.

3.3.3 Height Refinement Network

As mentioned previously, the height prediction map $\mathbf{P}_h$ produced by the multi-task learning network still contains some noisy areas that must be refined in order to generate the final height prediction $\mathbf{P'}_h$. We introduce an autoencoder to estimate the noise and produce more accurate height map predictions.

We choose the popular U-Net architecture [114] as network structure. The input of the network is the concatenation of the multi-task network outputs $\mathbf{P}_h, \mathbf{P}_s$ and $\mathbf{P}_n$ with the aerial image $\mathbf{x}_i$, as shown in Fig. 3.2. Details of the different layers forming the denoising network are listed in Table 3.2. The loss function used to optimize this network is the mean square error between the refined height map and the ground truth : $\mathfrak{L}_r = \frac{1}{n}\sum_{i=1}^{n}(P'_h - P^*_h)^2 = \frac{1}{n}\sum_{i=1}^{n}(P_h - \gamma - P^*_h)^2$, with γ being the noise function defined in Eq. 3.

3.4 Experiments

3.4.1 Datasets

2018 DFC [118] dataset was released during the 2018 Data Fusion Contest organized by the Image Analysis and Data Fusion Technical Committee of the IEEE Geoscience and Remote Sensing Society. It was collected over the city of Houston, which contains multiple optical resources geared toward urban machine learn-

	Layer	Output Size
Encoder	$Conv_1$	(320,320,64)
	$MaxPooling$	(160,160,64)
	$Conv_2$	(160,160,128)
	$MaxPooling$	(80,80,128)
	$Conv_3$	(80,80,256)
	$MaxPooling$	(40,40,256)
	$Conv_4$	(40,40,512)
	$MaxPooling$	(20,20,512)
	$Conv_5$	(20,20,1024)
Decoder	Upsampling	(40,40,512)
	$Concat$	(40,40,1024)
	$Conv_6$	(40,40,512)
	$Upsampling$	(80,80,256)
	$Concat$	(80,80,512)
	$Conv_7$	(80,80,256)
	$Upsampling$	(160,160,128)
	$Concat$	(160,160,256)
	$Conv_8$	(160,160,128)
	$Upsampling$	(320,320,64)
	$Concat$	(320,320,128)
	$Conv_8$	(320,320,64)
	$Conv_{out}$	(320,320,1)

Table 3.2: Height refinement network details.

ing tasks such multispectral LiDAR, hyperspectral imaging, Very High-Resolution (VHR) imagery and semantic labels. Using the results of the multispectral LiDAR, it is possible to obtain Digital Structural Models (DSM) and Digital Elevation Models (DEM), which, if subtracted from one another, produces height maps that we can use as ground truth. Four tiles of data are used for training while ten tiles are used for testing.

ISPRS Vaihingen [119] dataset was released during the semantic labeling contest of ISPRS WG III/4. It was collected over the city of Vaihingen, Germany and consists of very high resolution true ortho photo (TOP) tiles, corresponding Digital Surface Models (DSM) and semantic labels. As it is usually done when

dealing with this dataset, we use the normalized DSM (nDSM) produced by [120] as ground truth for our height prediction. Sixteen tiles were used for training while seventeen tiles are used for testing.

Surface normal maps: The surface normal maps for both dataset are generated using the given height maps, following practices usually used for surface normal estimation from dense depth maps based on the Sobel operator [121]. The details are listed in Algorithm 1.

Algorithm 1 Surface normals generation

Input: Height map P_h
Ouput: Surface normals map P_n
$zx \leftarrow Sobel(P_h, 0)$
$zy \leftarrow Sobel(P_h, 1)$
$N \leftarrow stack(-zx, -zy, 1)$
$P_n \leftarrow \frac{N/\|N\|}{2} + 1$
return P_n

3.4.2 Implementation Details

Our training process is not end-to-end. This is due to the formulation and loss function that we chose in the second network, which assumes that the first network is trained when trying to approximate the noise that it could generate. Instead, we follow a two stages approach: we first remove the denoising autoencoder and only focus on training the multi-task network. To do so, random 320x320 crops are sampled from the aerial tiles and corresponding semantic, surface normals and height ground truth are used for training. Once the multi-task network converges, we freeze its weights and then plug into the denoising autoencoder to obtain the final height predictions. We train this second network following the same random sampling process used to train the first one. We use Tensorflow [122], a learning rate of 0.0002, a batch size of 64, the Adam optimizer[123] and a single RTX2080Ti

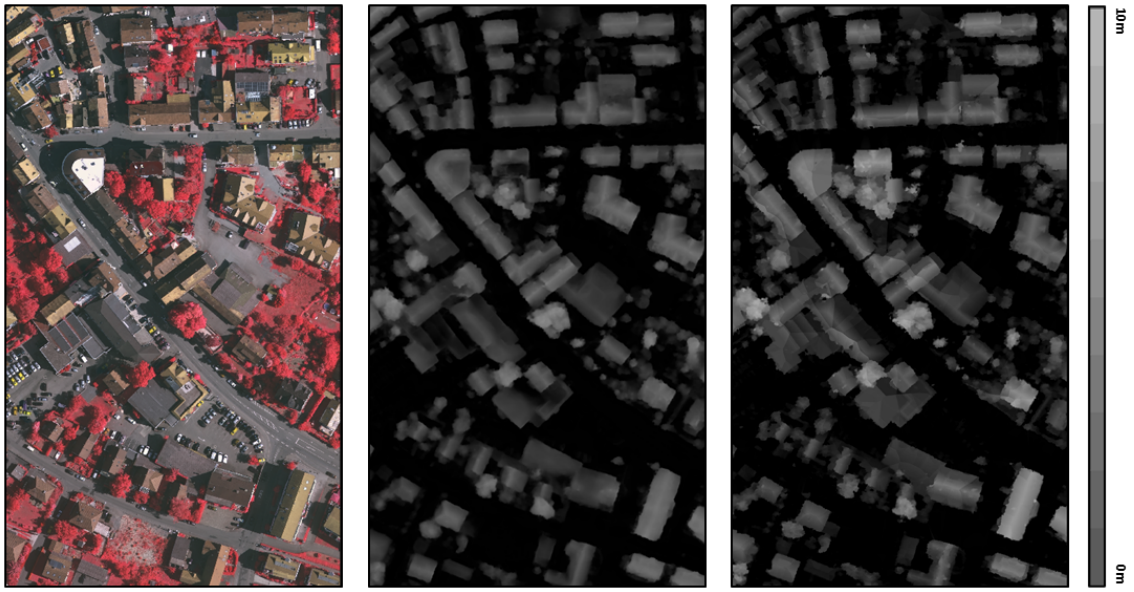


Figure 3.4: Qualitative comparison of a reconstructed tile from the testing dataset. From left to right: The input RGB tile, the height prediction and the height ground truth.

to train both stages. During training, we saw that altering the network's hyper parameters can sometimes have a slight effect of the convergence speed, but no significant effect on the final accuracy level.

Note that in the case of the DFC2018 dataset, the input VHR aerial tiles are ten times bigger than their corresponding DSM, DEM and semantic labels. To deal with that, we first down sample the aerial tiles ten times before starting to collect training crops.

3.4.3 Results

Height prediction results: The aerial tiles were reconstructed using a sliding window of the same size as of the training samples and with a constant step size. We use Gaussian smoothing to deal with overlapping areas. This makes it possible to deal with cases where different crops of the same area produce different height

values, while also protecting the final result from the "checkerboard effect". We report the results of our height prediction and refinement pipeline on both datasets in Table 3.3, where we use the mean square error (MSE), the mean absolute error (MAE) and root-mean-square error (RMSE) as metrics, all in meters. We also show a qualitative comparison in Fig. 3.4. When comparing with previous proposed methods in the literature, we can see that by using our multi-task network combined with the refinement step, we are able to surpass the state-of-the-art performance across all metrics on both datasets, with improvement up to 25%.

We credit this increase in accuracy to multiple factors. Firstly, the choice of our encoder (in this case DenseNet121), which is capable of extracting features that are relevant to this task. The second is the context information brought by our 2 additional branches in the multi-task prediction network. Knowing if a pixel falls on a building rather than the road, in addition to the orientation of its associated surface normal vector, helps the network predict height values better. Finally, the denoising autoencoder helps us deal with certain artefacts that tend to confuse the prediction network. We provide numerical analysis of these observations in the ablation study.

It is also interesting to note that we are able to achieve similar scores to methods which were trained on the high-definition aerial tiles directly without any down sampling as shown in Table 3.4. For reconstruction of the same sized area, such networks would take much longer processing time and significantly more computing resources than our proposed method.

Missing values in Table 3.3 were not reported by the cited publications. We also exclude the results reported by [99] because it did not follow the same training/testing split of the data.

Semantic label and surface normal predictions: Although this work does

	ISPRS Vaihingen			2018 DFC		
Method	MSE	MAE	RMSE	MSE	MAE	RMSE
Ours	**0.0042**	**0.036**	**0.062**	**6.92**	**1.37**	**2.57**
Carvalho [94]	0.0060	0.045	0.074	9.34	1.53	2.97
Srivastava [100]	-	0.063	0.098	-	-	-
IMG2DSM [98]	-	-	0.090	-	-	-

Table 3.3: Comparison with other height prediction methods on the ISPRS Vaihingen and the 2018 DFC datasets in meters.

Method	MSE	MAE	RMSE	Time (s)	Input Resolution
Ours	**6.92**	1.37	**2.57**	72	1192x1202
Carvalho VHR [94]	7.27	**1.26**	2.59	774	11920x12020

Table 3.4: Comparison with method trained on VHR aerial images.

not focus on the semantic label and surface normal predictions and only uses them to improve the height predictions, we share the results of those two branches and compare them with available methods in the literature in Table 3.5. Our results in Table 3.5 show that our multi-task network is able to produce semantic label results that are comparable with the state of the art on the Vaihingen dataset and acceptable ones on the DFC2018 (which has 20 classes compared to the 6 of the Vaihingen dataset). We use the following metrics for the semantic segmentation: The overall accuracy (OA), defined as the sum of accuracies for each class predicted, divided by the number of class, the average accuracy (AA), defined as the number of correctly predicted pixels, divided by the total of pixels to predict and Cohen's coefficient (Kappa), which is defined as $\mathfrak{Kappa} = \frac{p_0 - p_e}{1 - p_e}$, such as p_e is the probability of the network classifying a pixel correctly and p_0 is the probability of the pixel being correctly classified by chance. The network is also able to produce meaningful surface normal maps as seen on Fig. 3.1. Missing values in Table 3.5 were not reported by the cited publications.

	ISPRS Vaihingen			2018 DFC		
	Semantic Labels					
Method	OA	AA	Kappa	OA	AA	Kappa
Ours	85.6	74.8	**80.1**	51.89	47.01	49
Carvalho [94]	**87.7**	**85.4**	75.9	**64.70**	**58.85**	**63**
Srivastava [100]	78.8	73.4	71.9	-	-	-
Cerra [124]	-	-	-	58.60	55.60	56
Fusion-FCN [125]	-	-	-	63.28	-	61
	Surface Normals					
Method	MSE	MAE	RMSE	MSE	MAE	RMSE
Ours	0.0115	0.0642	0.1066	0.0620	0.2119	0.2572

Table 3.5: Semantic labels and surface normals results on the ISPRS Vaihingen and the 2018 DFC datasets.

3.4.4 Discussion

Height refinement: To demonstrate the usefulness of the aforementioned refinement network, we test our method with and without the denoising autoencoder, on both datasets. In Table 3.6, we compare the results obtained after both experiments and show that the refinement step always produces more accurate height maps, resulting in an increase of up to 16% in accuracy. By combining the information present in the semantic and surface normal inputs with the initial guess of the height produced by the previous network, the refinement network is able to concentrate on noisy areas where the height values are abnormal and fix them automatically. In addition, we compare our deep learning based denoiser with other popular non-learning denoising algorithms such as Bilateral Filtering (BF) [110] and Non-local Means (NIM) regularization [112].

We also show qualitatively on Fig. 3.5 that the refinement height maps are much closer to the ground truth and contains less noise than the direct output of the multi-task network.

Choosing the right encoder : Our network structure for height prediction

	ISPRS Vaihingen			2018 DFC		
Method	MSE	MAE	RMSE	MSE	MAE	RMSE
multi-task only	0.0045	0.043	0.065	7.36	1.50	2.64
multi-task + BF	0.0046	0.043	0.065	7.27	1.51	2.62
multi-task + NIM	0.0045	0.043	0.065	7.34	1.48	2.63
multi-task + Unet	**0.0042**	**0.036**	**0.062**	**6.92**	**1.37**	**2.57**

Table 3.6: Comparison of our height prediction methods with and without refinement, on the ISPRS Vaihingen and the 2018 DFC datasets in meters.

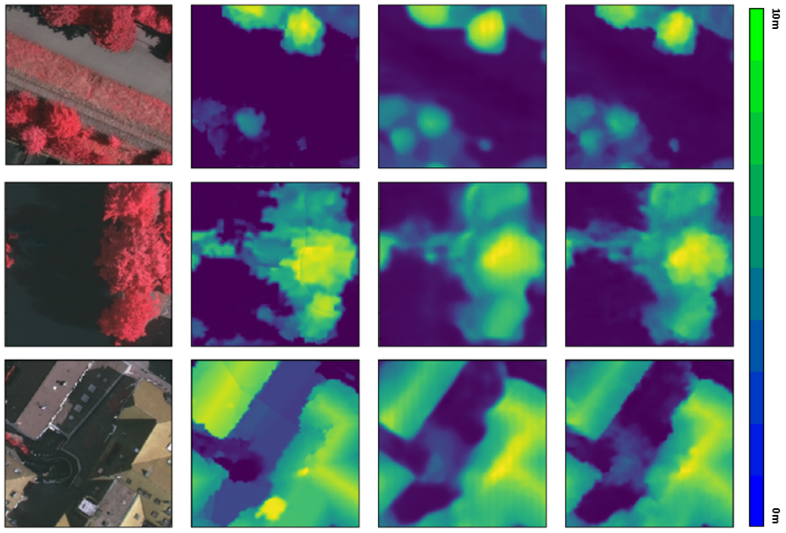


Figure 3.5: Qualitative comparison. From left to right: The input RGB image, the height prediction of our multi-task network, the refined height map of our denoising autoencoder and the ground truth.

is generic, since any off-the-shelf encoder can be used in the first stage to extract features from the input aerial image.

However, we show in Table 3.7 that DenseNet121 outperforms other popular encoder structures and produces the most accurate height maps. This is owing to the fact that DenseNet121 is much deeper than the other two networks and contains a higher number of skip connections between layers, making it possible to extract much finer features from the input image. All the networks are trained for the same number of epochs and using the same hyper parameters, such that it ensures the

Encoder	MSE	MAE	RMSE
ResNet101 [126]	18.95	3.33	4.19
VGG19 [127]	8.57	1.87	2.85
DenseNet121 [128]	**7.36**	**1.50**	**2.64**

Table 3.7: Encoder comparison on the DFC2018 dataset in meters.

fairness when comparing both the convergence speed and accuracy scores.

Geometric and semantic guidance : In this section, we show the effect of the geometric and semantic guidance in our method in both height prediction and height refinement stages. First, we show in Table 3.8 that using a multi-task network instead of a single task one improves the overall height prediction results. We also show in Table 3.9 that by concatenating all the results of the first stage as the input to the denoising autoencoder, we are able to generate more accurate and refined results compared to only using the height image as input. This shows that the semantic and geometric context information brought by two additional branches assist in producing more accurate height values.

	ISPRS Vaihingen			2018 DFC		
Method	MSE	MAE	RMSE	MSE	MAE	RMSE
single-task	0.0048	0.046	0.067	8.17	1.64	2.78
multi-task	**0.0045**	**0.043**	**0.065**	**7.36**	**1.50**	**2.64**

Table 3.8: Comparison of height prediction results of single and multi-task networks in meters.

	ISPRS Vaihingen			2018 DFC		
Method	MSE	MAE	RMSE	MSE	MAE	RMSE
single-input	0.0043	0.037	0.063	7.13	1.47	2.62
multi-input	**0.0042**	**0.036**	**0.062**	**6.92**	**1.37**	**2.57**

Table 3.9: Comparison of height refinement results of single and multi-input denoiser in meters.

Finding the right reconstruction step : The accuracy of our final tile reconstruction depends also on the step size of the sliding window that we choose when collecting the aerial crops. We show in Table 3.10 the different results corresponding to different step sizes. We found that a step size of 60 pixels results the best across both datasets.

	ISPRS Vaihingen			2018 DFC		
Step	MSE	MAE	RMSE	MSE	MAE	RMSE
80	0.00421	0.0363	0.0625	6.98	1.38	2.58
60	**0.00420**	**0.0362**	**0.0623**	**6.92**	**1.37**	**2.57**
40	0.00421	0.0362	0.0623	6.93	1.37	2.58

Table 3.10: Comparison of our reconstruction results (meters) based on the step size (pixels).

Visualizing the uncertainty : In order to investigate the performance of our pipeline more thoroughly, we generate uncertainty maps according to the method proposed in [129]. The results are displayed in Fig. 3.6 and show that most of the prediction errors can be attributed to the areas such as the edges of buildings due to the sudden changes in brightness and color, and trees where shadows introduce a significant amount of color noise.

3.5 Applications

In this section, we propose two applications to show how to take advantage of the results generated by our proposed pipeline. The first is 3D reconstruction of select buildings from a single aerial image. In the second application, we simulate a UAV flight over a certain area and show that we can reconstruct the entire 3D area by combining odometry and aerial images. In comparison to the classic SfM algorithm, our method provides a significant gain in speed, accuracy and density.

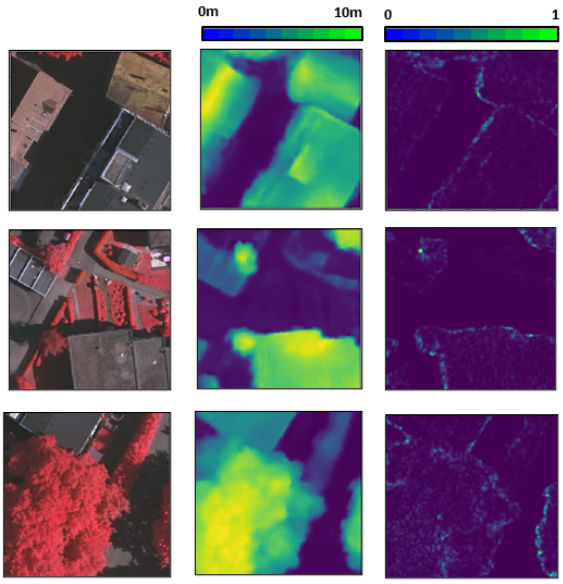


Figure 3.6: Uncertainty results. From left to right RGB Image, Height Prediction, Uncertainty Map. Prediction errors are mostly concentrated around the edges.

More importantly, our proposed method requires significantly smaller number of images since only minimal overlaps are necessary when taking the aerial shots.

3.5.1 Single Aerial Image 3D Reconstruction

Usually, in order to reconstruct the 3D shape of a building, multiple shots from multiple angles with significant overlap are necessary in order to apply the sequential surface from motion algorithm. We show in Fig. 3.7(b) that owing to our multi-task network, we are able to produce accurate 3D point clouds of the buildings using a single image only.

The proposed method is also capable of generating semantic point clouds in Fig. 3.7(c) and 3D meshes of buildings and their surrounding areas in Fig. 3.7(d) by leveraging the semantic labels and surface normals generated by the networks. Specifically, semantic point clouds are generated by projecting the semantic labels onto the point clouds, while the meshes are generated by combining the surface

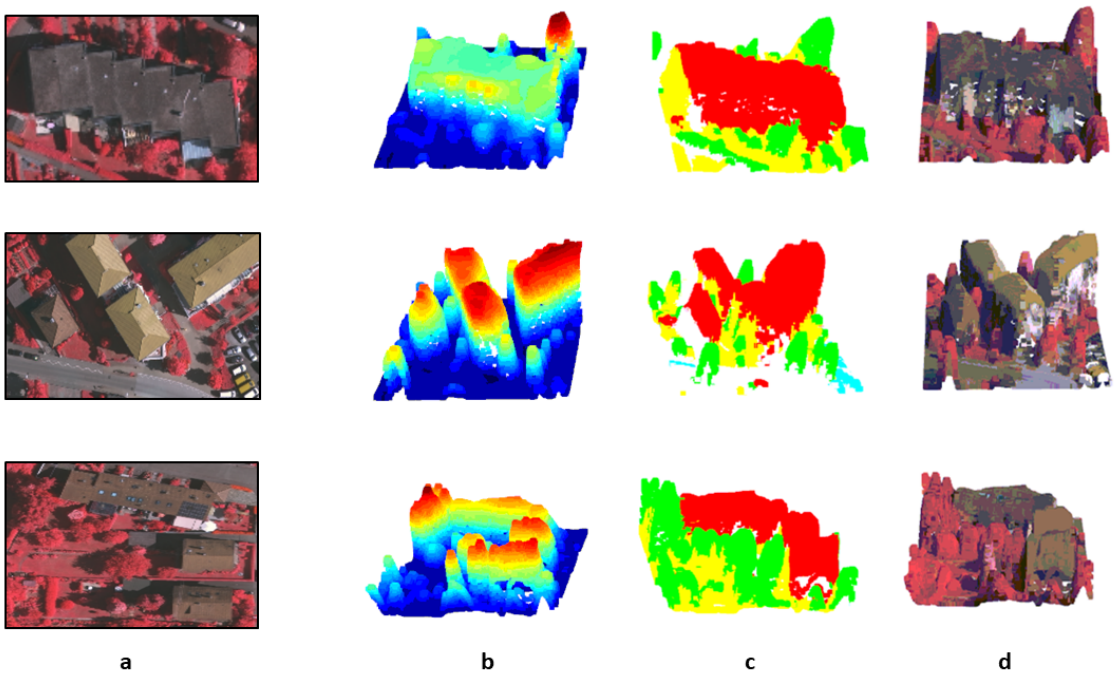


Figure 3.7: 3D reconstructions using a single image. (a) RGB Image, (b) Height Colorized Point Cloud, (c) Semantic Point Cloud, (d) RGB Colorized Mesh.

normals with the reconstructed point clouds using the ball pivoting algorithm [104].

3.5.2 Area Reconstruction with Simulated UAV Flight

3D reconstruction of urban areas is a very useful application. Similarly to what we mentioned in the first application, reconstructing an entire area would generally require a series of captured images with significant overlaps, by flying the drones in multiple passes over the same area, in order to generate a semi-dense point cloud.

In our case, we show in Fig. 3.8 that by using a single pass with a small number of captured images and minimal overlap (only to avoid gaps in the final reconstruction) we are able to produce accurate and dense 3D reconstructions. We also note that when we feed the same data to an SfM algorithm, it typically leads to failures since only a small number of features can be matched among the single-pass aerial shots.

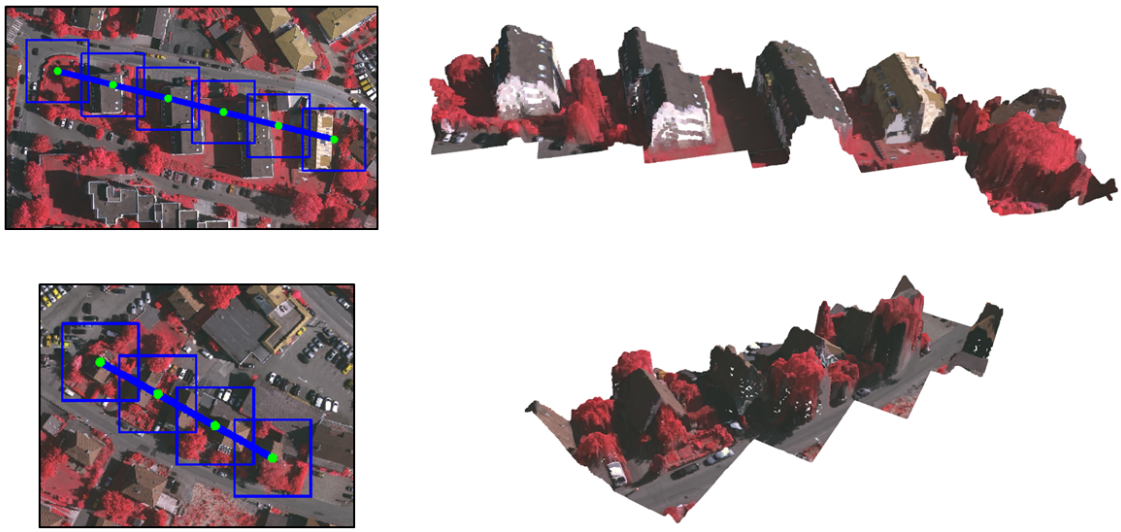


Figure 3.8: 3D reconstructions from simulated UAV flight. From left to right: Positions of the UAV images, Reconstructed 3D scene.

The data is collected by simulating a constant altitude UAV flight over a certain neighborhood in one of the tiles available in the testing datasets. The odometry is assumed to be known from on-board IMU or GPS sensors.

3.6 Summary

In this work, we propose a deep learning based two-stage pipeline that can predict and refine height maps from a single aerial image. We leverage the power of multi-task learning by designing a three-branch neural network for height, semantic label and surface normal predictions. We also introduce a denoising autoencoder to refine the predicted height maps and largely eliminate the noise remaining in the results of the first stage height prediction network. Experiments on two publicly available datasets show that our method is capable of outperforming state-of-the-art results in height prediction accuracy. In future work, we plan on exploring the computational efficiency of the proposed neural networks for their applications towards real-time

processing of aerial images. Future works include the use of additional connection between the three branches of the multi-task network in the first stage, or the addition of graph neural networks in the second denoising stage order to take advantage of the spatial connections between different segmented classes.

Chapter 4

Labeling Point Cloud Maps Using Cameras and Deep Learning

4.1 Introduction

During the last couple of decades, autonomous driving has become one of the most researched topics in the scientific community. This stems from the fact that scientists, governments and people in general are starting to realise the huge positive impacts that autonomous driving could have on our daily lives: According to [130], self-driving cars could reduce traffic fatalities by up to 94% by eliminating the accidents that are due to human error.

The race toward full autonomous driving cars, or level 6 autonomy such as it categorized by SAE International, has given rise to multiple new fields and was the catalyst to launch or greatly improve several new disciplines. This manifested itself in the field of deep learning, which has made it possible today to achieve a respectable level of autonomy when driving on roads that fall into the classic scenario box. Lanes and roads (or driveable regions) detection can be achieved with

neural networks trained to excel in task related to pixel-wise segmentation of images captured by cameras [131],[132],[133], making the car aware of where it is safe to drive. Other types of networks are trained to detect obstacles and classify them into several independent classes [134],[135],[11], sometimes with the help other sensors such as radars to mitigate the accuracy issue when it comes to depth and 2D images. This helps the car drive safely by avoiding obstacles on the road and take account of traffic rules represented by traffic signs or traffic lights.

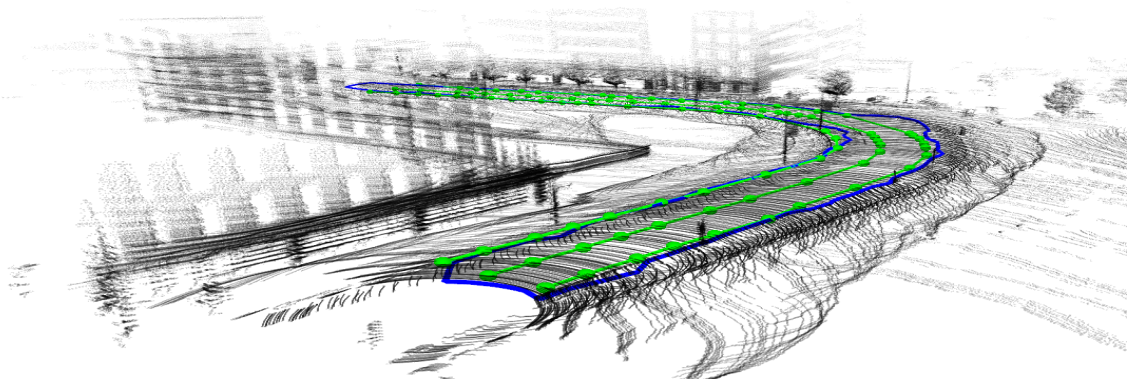


Figure 4.1: Point cloud, lanes coordinates and driveable region limits generated by our pipeline.

This approach has multiple advantages that revolve mostly around the real time aspect of the method and the fact that it is mostly built on top of cheap sensors. However, methods based solely on deep learning and cameras, while being very performant in highway scenarios for example, cameras are bound to fail when deployed in urban scenarios because cameras have major weaknesses, mostly related to brightness issues. These weaknesses could be covered by fusing the camera data with other more powerful and accurate sensors, such as LiDARs. This introduces the most accurate method to date to navigate and drive autonomously in urban areas: using HD Maps.

HD Maps are a combination between 3D point clouds and relevant semantic information. 3D point clouds can be used for localization by match the incoming

scans of the LiDAR when driving, with the pre-build and stored 3D point clouds. However, in order for these point clouds to be used for autonomous navigation, additional data has to be stacked on top of it: Information such as the position of the lanes, roads or traffic signs has to be labeled in order for the car the navigate safely while respecting basic traffic rules.

Motivation : The usual solution used today to label pre-built maps is to manually label them and store them. The shortcomings of this method are: the lack of accuracy, the high financial and labor cost in addition to how time consuming it can be. For the HD maps model of localization to be scalable, we need to introduce automation in the building and labeling process of these maps, where most of the work is done by the machine, and the human is only there to supervise, correct or validate.

Approach : A different approach would be to use the great results achieved by neural networks not to navigate directly on the road, but to build and label HD Maps offline before deploying them on cars. Therefore, we propose in this paper to use deep learning to automate the labeling process of HD Maps, by combining them with other methods to guarantee accuracy and robustness. This is done by first prediction relevant semantic labels on the input camera images. The predictions are then processed by removing outliers, introducing smoothing or clustering, in order to generate waypoint-based labels that can be used by the autonomous vehicle to safely navigate in the mapped environment. Our contributions can be summarized as a collection of algorithms and pipelines aiming to automatically label HD Maps for urban autonomous driving.

4.2 Related Work

Used by many major companies in the autonomous driving business and research community such as Waymo or Autoware [136], navigation using HD Maps has proven to be the most robust method up to date. HD Maps are maps based on laser data collected using a LiDAR sensor. The most recent HD maps are built using a 360 rotating LiDAR sensor, where after each full rotation, a scan is packaged and sent to the computer. Multiple scans are then accumulated in order to generate a point cloud where the 3D geometry of the surrounding environment is represented. Accumulating these scans can be done using multiple approaches: The output of an Extended Kalman Filter (EKF) [137] where IMU and odometry data have been fused can be used to align the point clouds. Other methods based on feature detection and feature matching [138],[139] allows us to find the right translation and rotation between two consecutive scans. Another approach based on an algorithm called the Iterative Closest Point (ICP) algorithm [140] can be used: the ICP is an algorithm designed to minimize the difference between two point clouds and thus finding the right transformation between them. ICP was the leading and most robust algorithm to align two consecutive point clouds until it was surpassed by the 3D Normal Distribution Transform (3D NDT) algorithm [141].

The 3D NDT algorithm was proposed by M. Magnusson et al. in order to help the deployment of autonomous mining vehicle. It builds on the 2D method of the same name [142] by transforming the point cloud into a probability density function which can be used with Newton's algorithm to match another point cloud.

In his paper, M. Magnusson shows that the 3D NDT algorithm outperforms the ICP algorithm, which was the standard 3D matching algorithm at that time. In order to improve both the speed and accuracy of the 3D NDT algorithm, an

EKF is used to fuse both IMU and GPS data to generate an initial guess of the transformation that is passed to the 3D NDT algorithm as a starting point for the minimization procedure.

After constructing the point cloud, it is it is necessary to add semantic information to it such as lanes and driveables regions in order for the car to be able to autonomously navigate itself. Traditionally, the way this information is labelled on the point cloud is by using a manual method. As an example, Autoware gives access to their user to an online tool named Vector Mapper, where they can load up their pre-constructed point clouds and label them manually. The labels are then exported and available for download as CSV files, called Vector Maps. However, manual labeling of point clouds tends to be extremely tedious, time consuming and not as accurate as we would like it to be. Also what could be a one step process is now divided into two independent ones where the point cloud is first built as first step then labeled as a second one. Since most of the information that is being labelled can be generated from deep learning networks, we propose in this paper to bypass the manual labeling process and use the predictions generated by the networks to label the point cloud automatically. We also propose other post-predictions methods to mitigate the false positives and uncertainty that tend to occur sometimes when using deep learning.

The deep learning networks that we deploy in this process (namely LaneNET and FCN8s) are monocular camera based and not LiDAR based. This choice stems from the fact that the level of accuracy reached by camera-based networks has yet to be matched by the LiDAR based ones and also by the fact that deploying multiple LiDAR networks at the same time will require a lot more processing power. The LiDAR might outperform the camera when it comes to range, but our algorithm here relays on accumulating information from successive samples which helps covering a

larger amount of space. In order to fuse both the camera predictions and the LiDAR point clouds, a transformation consisting of a translation and rotation between the camera and LiDAR is needed and can be found using a Lidar-camera calibration method. One of the proposed methods in the literature, which we used in this paper, is described in [143].

4.3 Methods

4.3.1 Mapping Pipeline

The mapping pipeline presented in this paper concentrates on constructing the 3D geometry of the surrounding environment. This is done thanks to the 3D NDT algorithm, used to align successive LiDAR frames, that were collected during a pre-defined path. For clarity purposes, we define two frames: The map frame F_m, which the origin of is the center of the first scan at the start of the map, the car frame F_c, which the origin of is the center of gravity of the autonomous car and the LiDAR frame F_l which the origin of is the center of the LiDAR sensor.

4.3.2 Road labeling

After building the map offline, we proceed to deploy our automated road labeling pipeline. We define a road R as polygone in the F_m frame limiting the areas where it is possible to drive, but not necessarily legal to do so. Road labeling is performed for two main reasons :

- It can guide the autonomous car when no lanes are present on the road, as it is sometimes the case for one way streets.

- It will help us later when doing lane labeling.

The road is detected using the camera, projected on the LiDAR data, refined to remove outliers and then accumulated onto the pre-built map using the output of the 3D NDT algorithm. We then compute the region occupied by the road and extract the road limits. We will explain each of these steps in the following. Fig. 4.2 shows an overview of the road labeling pipeline.

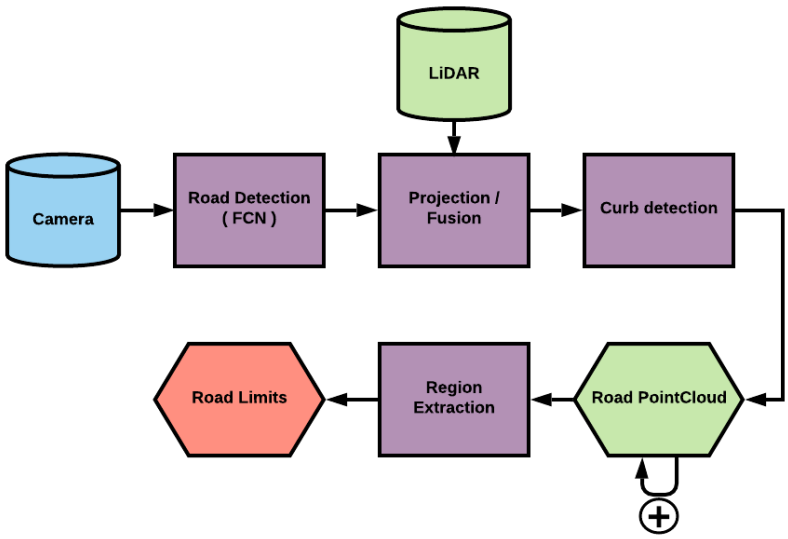


Figure 4.2: Road labeling pipeline

Detection : For detecting the road we use a Fully Convolutional Network. We apply the network to the front camera data, in order to segment the image into 2 regions: Road and Not Road. This results into a binary image that we will use in combination with the Lidar-Camera calibration to segment the point cloud of the road.

Projection : We start by using the camera parameters to crop the 360 point

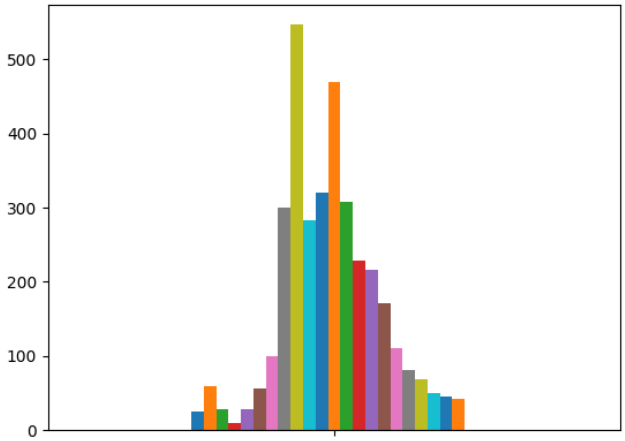


Figure 4.3: Histogram of the elevation z of the road point cloud.

cloud, so that we are only operating on the points that fall within the field of view of the camera. Then, using the Lidar-Camera calibration, we project the image on top of the point cloud while making sure that the color information from the image is preserved and transferred to the point cloud. This results into a binary point cloud where the road points are colorized differently than other points on the point cloud.

Curb detection : At this point, if the road detection results from the FCN were perfect, the labeling process would be done. However, this is not the case, since the road detected by the FCN tends to bleed on the edges of the curb, especially when the curb is hard to distinguish in the pictures because of shadows, brightness changes or the curb being too small. This means that we need to refine the results from the FCN by detecting if any curb portions were included into the predictions.

In order to do so, we use the colorized point cloud and the elevation according to the z axis. We start by extracting the road point cloud from the colorized point

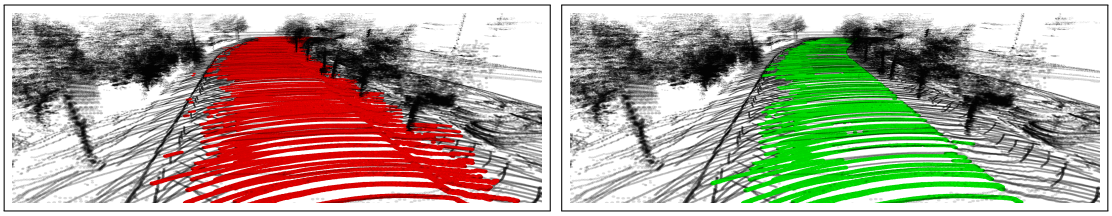


Figure 4.4: Road labeling before (Red) and after (Green) the curb detection.

cloud that we obtained previously using a color segmentation-based method, and then, as it is shown in Fig. 4.3 we display the elevation of the points in the road point cloud as histogram. This shows us that the points in the extracted road point cloud follow a bimodal distribution, meaning a distribution that contains two peaks, or in other word, two normal distributions with means μ_1 and μ_2 respectively and standard deviations σ_1 and σ_2 respectively. This makes sense because the first normal distribution represents the points on the curb, and the second one the points on the road. In this scenario, detecting the curb consists of separating the bimodal distribution into two normal distributions and excluding the distribution representing the curb points. In order to do that, we use a method commonly employed in computer vision for segmentation and clustering purposes called the Otsu method [144].

The Otsu method, applied to a bimodal distribution, calculates the optimum threshold separating the two classes (in our case "road" and "curb"). This makes it possible to exclude most of the points that lay on the curb as it shown on Fig. 4.4 and leaves us with a portion of points, which the elevation of, follows a normal distribution with a mean μ_1 and standard deviation σ_1. As a final check, and to remove the rest of the outliers still present, we apply the 68–95–99.7 rule to the resulting distribution and exclude all the points which the elevation lay outside of $\mu_1 - \sigma_1$ and $\mu_1 + \sigma_1$.

Region Extraction : In order to extract the limits of the driveable area, we need to compute the contour of the road point cloud projected onto the (x, y) plane. This is achievable by using a Concave Hull (CH) [145].
The CH is an algorithm based on the k-nearest neighbours approach and designed to generate a envelope describing the area occupied by a set of points in a plane. The envelope generated by the CH is used to generate a polygone describing the driveable area.

4.3.3 Lane labeling

We define a lane L as a set of points $L = \{p_1, p_2, ..., p_n\}$ where $p_i = \{x_i, y_i, z_i\}$ are the coordinates of the $i'th$ point in the F_m frame. Lane labeling is performed in order to help the autonomous car navigation process on the road by keeping it centred. The lanes are detected using the camera data, projected on the LiDAR data, clustered and smoothed to generate meaningful waypoints and then accumulated with the previous scans using the output of the 3D NDT algorithm. We also generate the missing lanes that were not detected in the camera data. We will explain each of these steps in the following. Fig. 4.5 shows an overview of the lane labeling pipeline.

Detection : For lane detection, we use LaneNET. This network was selected because it is able to detect all the lanes that are visible from the front view camera and not only the ego lanes. The network outputs a mask image of the same size as the input image, where the pixels belonging to the lanes are labeled and color coded. Similarly to what we did for the road labeling, the mask image will be combined with the Lidar-camera calibration to generate a lanes point cloud.

Projection : Since the Lidar-camera calibration will become less accurate the

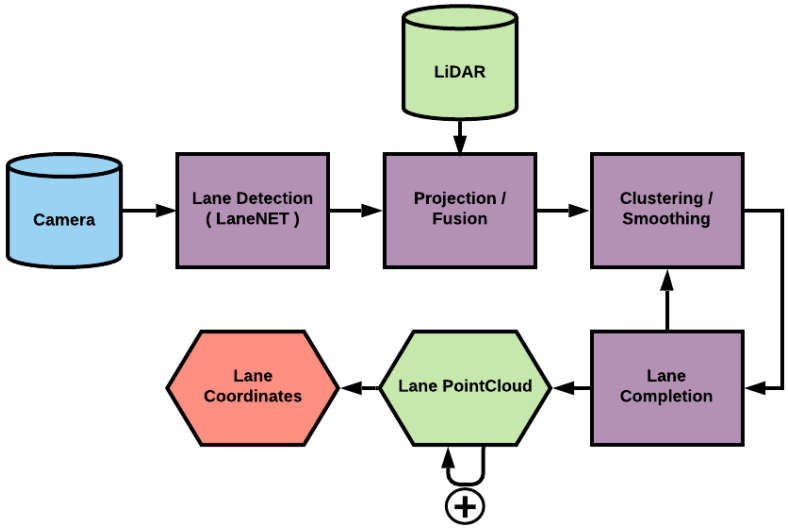


Figure 4.5: Lane labeling pipeline

further we are from car, we start by cropping the "camera field of view point cloud" to a certain distance L from the origin of the F_c frame before projecting the lanes on it. This helps preserving the shape of the lanes as we will be accumulating the projections and point clouds while advancing. Once again, we make sure that lanes in the resulting point cloud are colorized differently than the rest of the point cloud it was projected on. Finally, using a color segmentation method, we extract the points belonging to the lanes to form a "lanes point cloud".

Clustering & smoothing : The generated lanes point cloud is noisy and does not always follow a coherent geometry. Therefore, we set up a series of clustering and smoothing steps that will be applied to the lanes point cloud in order to generate a series of waypoints that could be used by the autonomous car to know the positions of the lanes in the space. The smoothing and clustering is applied on two different levels: first in the LiDAR frame F_l when dealing with a single frame, then second in the map frame F_m when the current frame has been accumulated with

the previous ones using the output of the 3D NDT algorithm.

We first start, in the F_l frame, by dividing the resulting lanes point cloud into equally separated Regions of Interests (RoI) in order to operate on the different lanes independently. These lanes are then clustered following the method proposed in [146] which is mainly based on the Euclidean distance between neighboring points, and which generate 3 points that describe the geometry and curvature of the lane. These points are then smoothed by being fitted to quadratic function before being transformed into the F_m frame and accumulated with the previous scans using the result of the 3D NDT algorithm. When in the F_l frame, the clustering follows the same method used in the first step, but with a higher tolerance and the smoothing is done by fitting the points to a logarithmic curve instead of a quadratic one.

We define the smoothing index N as the index in the lane L from where the smoothing process starts and the "look-back" parameter l that defines how much of the full lane do we want to include in the smoothing process. The smoothing and clustering of the lane points will be applied if certain criteria are met :

1. If the accumulated distance of the lane portion is superior to a threshold $L1$, that lane portion is smoothed, and the smoothing index, is moved to the $N'th$ position, where $N = n - m - l$ with n being the size of the full lane point L cloud and m the size of the last lane portion $L' = \{p_{n+m}, p_{n-m+1}, ..., p_n\}$ that was added during the last scan.

2. If we receive S scans without lane points, signaling the presence of an intersection for example, that lane portion is smoothed and the smoothing index is moved to the $n'th$ position, where n is the size of the full lane point cloud.

3. If the accumulated distance of the lane portion is superior to a threshold $L2$, that lane portion is clustered in order to produce reference points.

Setting up the thresholds $L1$, $L2$ and l is critical in order to obtain lane points that are not too curved/straight or too dense/light.

Lane completion : One of the other imperfections of deep learning that we need to deal with is missing lanes. LaneNET will sometimes not be able to detect a lane either because of a sudden change in brightness/contrast, the lane not being in the field of view of the camera, or the lane simply not being drawn on the road. We deal with this issue by combining the successful lanes that were detected, our curb detection algorithm and the fact that the lanes on the road are parallel.
We first start by using the curb detection results to check if all the lanes were detected : based on the position of the curbs and the lanes width (which is derived from the successful detections), we can conclude if the right number of lanes was detected. If a certain lane is missing, we use the closest left or right lane to it to generate it by fitting the lane points obtained from the last scan to a quadratic curve and then combining the normals to the curve at each of those points with the lanes width to generate a new lane.

4.4 Experiments

4.4.1 Experimental Setup

Our experimental setup consists of a Lincoln MKZ, equipped with one Velodyne Lidar VLP-16 and one FLIR PointGrey RGB camera. Both sensors are recording at 10 Hz and 30 Hz respectively. The car is also equipped with an IMU/GPS to assist the 3D NDT algorithm. The data is synchronized and recorded thought the Robot Operating System (ROS) as ROS bags and processed offline. Fig. 4.6 shows an image of our experimental setup vehicle.

Figure 4.6: Vehicle used for data collection.

We recorded and built multiple HD maps using the pipeline, from which we selected 5 scenarios.

Fig. 4.7 shows satellite imagery of the 5 selected areas. Each of these scenarios was selected for a specific reason:

- *straightRoad* was selected to demonstrate the effectiveness of the curb detection step in the road labeling pipeline.
- *curvedRoad* was selected to demonstrate the accuracy of the Smoothing and Clustering step in the lane labeling pipeline.
- *mergeLane* was selected to demonstrate that the lane labeling pipeline is able to handle scenarios where new lanes appear and that we are not restricted to the lanes that we started with.
- *intersection* was selected to demonstrate the results of merging multiple maps.
- *highway* was selected to demonstrate that the labeling pipeline is still effective in significantly bigger areas.

Figure 4.7: Aerial imagery of the selected areas.

Table 4.1: Scenarios details.

Scenario	Number of recordings	Number of LiDAR scans	Driveable region area (m^2)	Number of lanes
straighRoad	1	123	1374	3
curvedRoad	1	145	1346	3
mergeLane	1	112	1296	4
intersection	3	321	3282	6
highway	2	169	6173	7

The ground truth was manually labeled using the intensity of the LiDAR point cloud and satellite imagery as reference. When building and labeling our HD Maps using our automatic pipeline, we select key-frames from the LiDAR scans based on the synchronization with the camera and we set our thresholds such as $L1 = L2 = 30$ meters, $l = 5$ and $S = 5$. Table 4.1 presents some statistics regarding each scenario. Since we run our experiments on our own data, that no public labeling dataset is currently available and that we do not have access to any similar works code, we only compare our results to the ground truth.

4.4.2 Road Labeling

We present the results of our road labeling pipeline : Fig. 4.8 shows a qualitative comparison between the manually labeled ground truth and automatically labeled roads, with and without the curb detection algorithm. Table 4.2 lists the errors in the areas occupied by the road polygon before and after the curb detection algorithm. The results show that the curb detection algorithm is very effective in excluding the points that do not belong to the road. However, traffic on the road can sometimes lead to results where valid road points are excluded, as it is the case during the *mergeLane* recording. A potential solution would be to detect the objects on the road and exclude the points that belong to those objects before applying the curb detection algorithm.

When applying the curb detection algorithm, we found that obtaining great results depends highly on the number of bins we set when construct the elevation

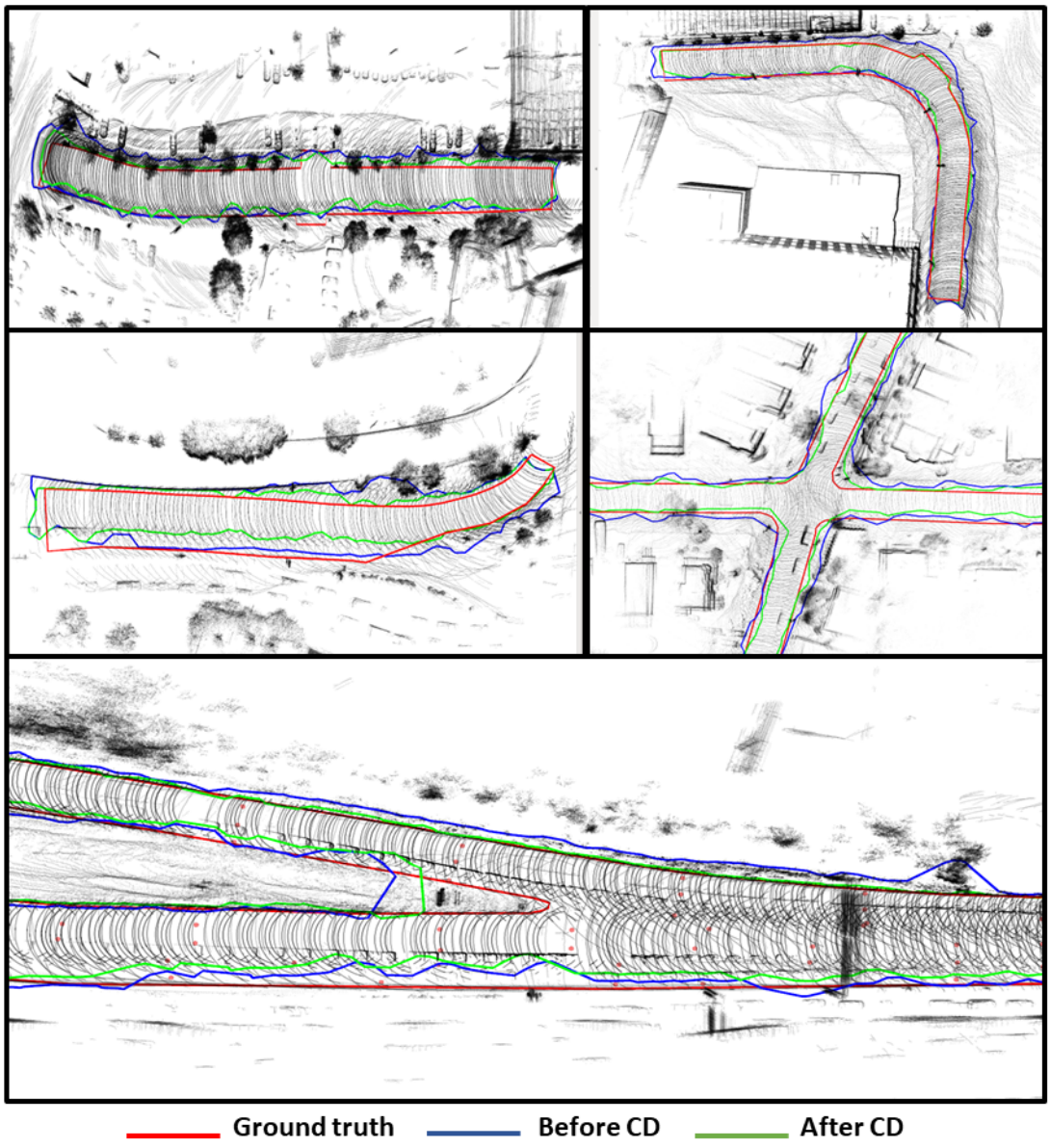


Figure 4.8: Qualitative roads comparison : Red is the ground truth, Blue is before the curb detection, and Green is after.

	ϵ_1	ϵ_2	δ
straighRoad	0.232	0.008	0.16
curvedRoad	0.261	0.034	0.22
mergeLane	0.252	0.127	0.30
intersection	0.357	0.017	0.18
highway	0.007	0.160	0.32

Table 4.2: Errors in the areas occupied by the labelled road. ϵ_1 and ϵ_2 are the errors (m^2) before and after the curb detection respectively. δ represents the percentage of points that were excluded.

Table 4.3: Errors in the areas occupied by the automatically labelled road depending on the number of bins in the elevation histogram.

Number of bins	5	10	25	50
Error value	0.263	0.051	0.167	0.008

histogram of the road point cloud that will be fed to the Otsu method. Table 4.3 lists the error values when varying the number of bins between the values $\{5, 10, 25, 50\}$ on the *straighRoad* scenario. We chose that example due to similarities in color and shade between the curb and the driveable region which tends to mislead the FCN network.

4.4.3 Lane Labeling

We also present the results of our lane labeling pipeline: Fig. 4.9 shows a qualitative comparison between the manually labeled ground truth and automatically labeled lanes. Table 4.4 lists the translation error mean values and the standard deviations in x and y of the automatically labelled lanes from the ground truth. When being evaluated, both the maps generated from the *intersections* and *highway* were split into two maps representing each of the roads present. These results show that the lane labeling pipeline is capable of accurately labeling the lanes and generating the

	$Lane_1$	$Lane_2$	$Lane_3$	$Lane_4$	σ_x	σ_y
$straighRoad$	0.918	0.605	0.300	N/A	0.102	0.190
$curvedRoad$	0.941	0.788	1.395	N/A	0.325	0.525
$mergeLane$	0.977	0.943	0.562	0.877	0.282	0.468
$intersection_1$	1.238	0.409	0.583	N/A	0.384	0.303
$intersection_2$	0.635	0.951	1.054	N/A	0.634	1.101
$highway_1$	0.507	0.625	0.764	N/A	0.613	0.619
$highway_2$	0.585	0.583	0.618	0.565	0.726	0.340

Table 4.4: Translation error (m) between the automatically labeled lanes and the ground truth.

missing ones. However, this last part is highly dependent on the success or not of the road labeling pipeline in finding the furthest curb.

4.4.4 Discussion

As the results show, automatic labeling from deep learning results is a viable option to replace human labeling when building HD Maps. However, some practices need to be taken in account in order to generate acceptable results:

- **Weather** : The recording should be gathered during good weather and preferably during a time of the day where sun reflections will not be an issue. This makes it easier for the deep learning network to detect lanes or road boundaries.
- **Traffic** : The recording should be gathered during a time where traffic is at its minimum. Cars on the road tend to throw off the curb detection algorithm by adding noise to the road point cloud and thus to the elevation histogram.
- **Merging maps** : Areas such as intersection or exit lanes need multiple recordings to be reconstructed. The recordings should cover as much overlapping

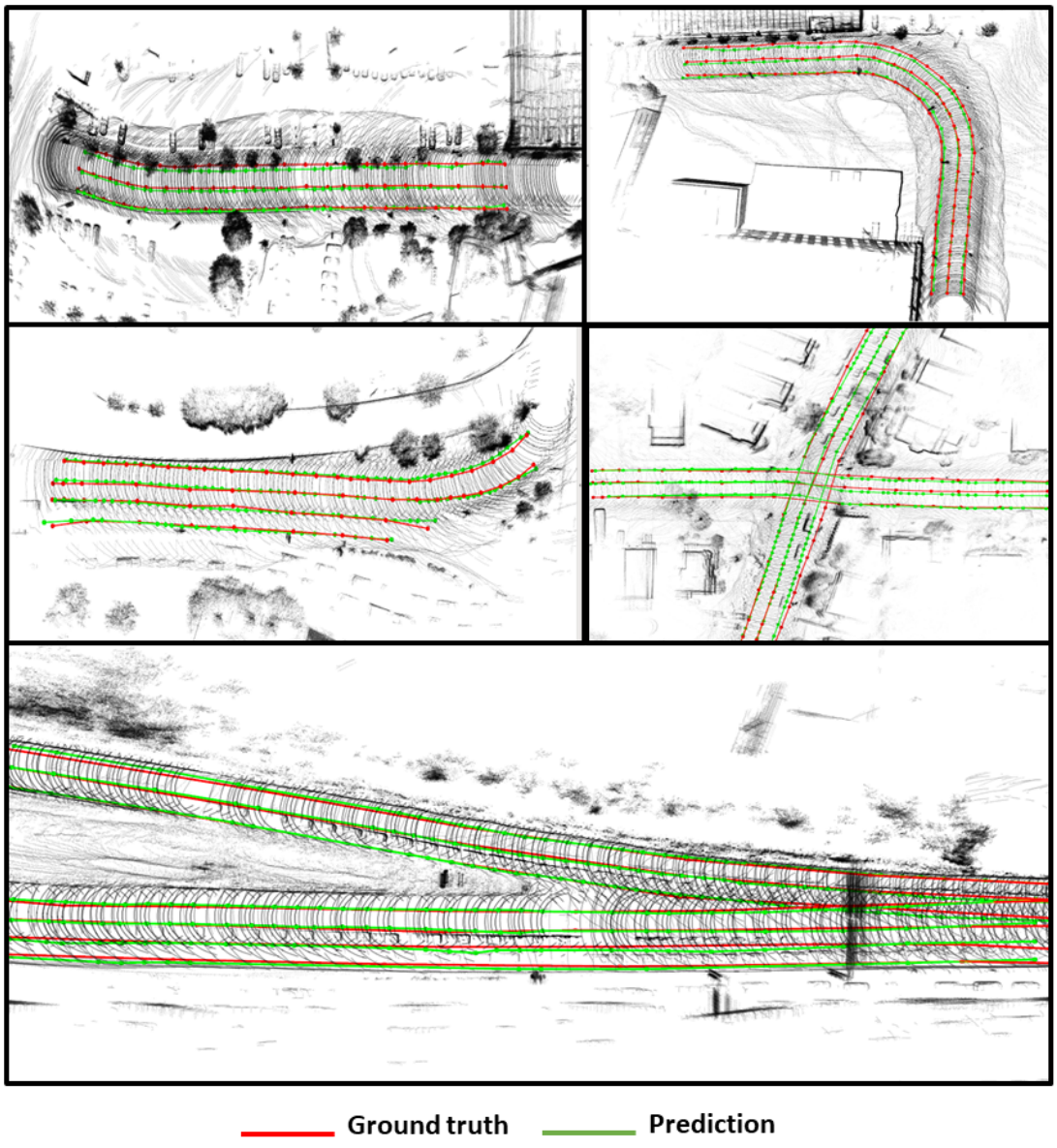


Figure 4.9: Qualitative lanes comparison : Red lanes are the ground truth and Green ones are automatically generated.

areas as possible to facilitate their merger. GPS data should be collected also to help construct an initial guess of the transformation between the maps generated by the different recordings.

4.5 Summary

In this work, we proposed a pipeline designed to automatically label HD Maps for autonomous driving cars. The pipeline relies on the results of deep learning networks trained to detect the driveable areas and the lanes. These results are then automatically post-processed in order to be corrected, improved or completed. A comparison between the results of our pipeline and a manually labelled ground truth proved the accuracy and effectiveness of the methods employed in this work. Future work includes automatic labeling of more details on the HD Maps such as traffic sign and traffic lights. Future works include the extension of this pipeline to the traffic light and traffic signs labeling, by using a similar method based on 2D camera object detection.

Chapter 5

Localizing Point Cloud Scans in OpenStreetMaps

5.1 Introduction

Localization is one of the key modules in an autonomous driving system. Knowing the precise location of the vehicle is critically important to perception and controls. LiDAR sensors have been proven to be very useful when attempting to solve localization challenges [147, 148, 149, 71], owing to their accurate and dense representations of the vehicle's surroundings. The localization results achieved by LiDARs can be further enhanced based on pre-built 3D maps [150, 72]. These 3D maps, sometimes called HD maps, are built by concatenating successive point clouds that were aligned by using additional data from other sensors (e.g., differential GPS), or by using a complex SLAM pipeline, which would typically include practices such as loop closure. Then the HD maps need to undergo a labelling process, either manually or by using an automated process [151].

Although these types of maps can help achieve a high-level accuracy when track-

ing the vehicle, they nonetheless come with drawbacks which make their deployment very challenging, due to the current technological issues. For instance,

- Expensive and time-consuming: Building and maintaining such maps requires us to drive repeatedly around all the areas where we would be interested in locating the vehicles, which not only can be tedious, but also very expensive.
- Large storage: HD maps are known to be very large due to the millions of points that they contain, which can also make their deployment difficult in real time.

Therefore, we argue that 3D point clouds may not be the best format to use in terms of maps. In contrast, we propose to combine LiDAR with a widely available, free, and routinely updated map namely OpenStreetMaps (OSM) [152].

Motivation: LiDAR sensors excel at representing the geometrical features of the surrounding environment. In the case of a vehicle driving in an urban setting, the surrounding environment is mostly composed of the road and buildings (nearby objects such as cars are included as well), due to the effects of occlusion. Naturally, these two classes are commonly labeled in OSM, and an intuitive way to use them would be to match them across modalities in order to localize a vehicle. Although some impressive advances were made in neural network alignment between camera and LiDAR such as [82, 83], matching RGB images extracted from an abstract map such as OSM with point clouds as accurately as possible for localization is very challenging. To address this issue, we propose to use ray casting, combined with a buildings mask that we extract from OSM, so that we can produce simulated LiDAR images. We also extract road masks from OSM and use them in two different ways: (1) as a secondary map where we will localize vehicles on the road extracted from the input point clouds; (2) as the constraint map to make sure that the final localization

result does not leave the road, in case of a noisy sensor input.

Objectives: Our goal in this method is to propose a non-learning, simple and stable method capable of localizing vehicles in OSM by only using LiDAR senors, with an accuracy level on par with or superior to GPS sensors (which is reported to be more than 5 m in an urban setting [153, 154]). This means that our method will be able to replace GPS, or act as backup when GPS is not available. It can also be used as an initial guess to other more advanced sensor suites.

Approach: The difficulty of aligning LiDAR point clouds with OSM is mainly owing to the modality gap between these two representations. Point clouds returned by a LiDAR are typically represented by an $(N, 4)$ unsorted array representing the (x, y, z) 3D locations and the reflectivity r of each point. In contrast, the OSM representation that we are using is an $(L, W, 3)$ RGB image, representing a top view of the environment, with different classes labeled accordingly.

A common first step when attempting to localize 3D point clouds in 2D aerial maps, is to proceed to a bird's eye view (BEV) projection, which means first cropping the point cloud to an acceptable region of interest, then projecting it on the ground plane, and re-scaling it according to a pre-defined scale (in this case, matching the scale from OSM), so that it can be represented as a 2D image, where each non-zero pixel stores the height value of the corresponding point. This results in a top view representation of the LiDAR point cloud, and a closer visual appearance to the top view 2D map. However, even after proceeding with this projection, the modality gap is still too big to attempt any direct matching between these two. In order to solve this issue, we propose to first extract a road and buildings mask from OSM, because those are the most common classes of objects in an urban environment. These masks are then used in combination with ray casting, to generate simulated top view LiDAR images. We use a dual input particle filter that attempts to align

the simulated LiDAR images with input point cloud BEV images. Because it is reasonable to assume that our vehicle should never leave the road, we also use the road mask to constrain the particle filter, making sure that the final solution always lies within the constrained region.

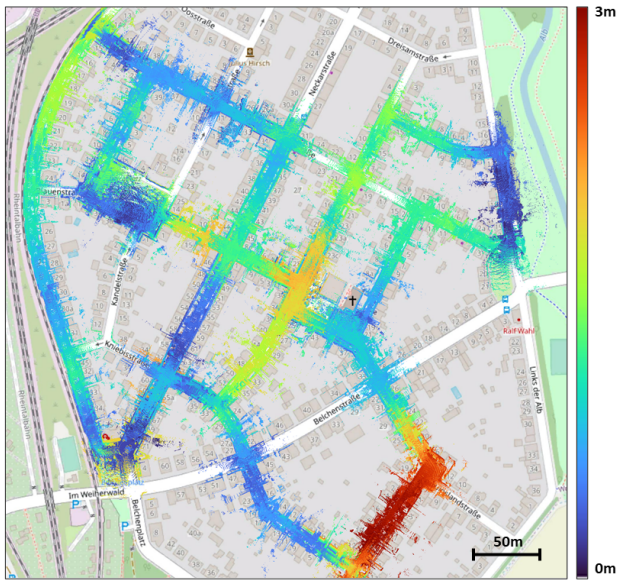


Figure 5.1: Result of our approach. LiDAR point clouds overlaid on top on OSM. Colors reflects the position error (m).

Our method does not rely on any type of machine learning and uses only open-source maps for LiDAR based localization input. It has the potential to be used anywhere as long as OSM is available and can act as a replacement of GPS when the GPS sensor or GPS signal is unavailable. Figure 5.1 shows an example of our method's results.

The contributions of this research can be summarized as follows: (a) We propose a fast and consistent method to generate simulated top view LiDAR images from OSM, and accordingly show how we can use those images to accurately localize LiDAR point clouds in OSM. (b) We propose a dual-input particle filter algorithm with an added constraint that the vehicle location must be on the road. (c) We

demonstrate the state-of-the-art accuracy of our method on the KITTI dataset, by comparing it to other LiDAR cross-modal localization methods by using OSM or satellite maps.

5.2 Previous Work

Cross-Modal LiDAR Localization: LiDAR localization [155], odometry, and SLAM are classical robotics challenges that have become popular with the advent of autonomous driving, owing to the richness of the data that the sensor produces. Lately, cross-modal LiDAR localization has been attracting more attention, due to the fact that it requires a significant amount of time, effort and resources when trying to build and maintain HD point cloud maps. This led researchers to look for alternative platforms that could be used as maps when trying to localize a vehicle driving in an urban environment. The most accessible and available maps to the public today are top view 2D maps such as OSM and satellite maps. OpenStreetMaps (OSM) are one type of such maps, which provide 2D shapes of most major structures, tagged with geo-localization data. Thanks to its open-source character, OSM has been very popular within the SLAM research community. In [156], a classifier is trained to distinguish between LiDAR points that fall on the road or not. Based on this information, a cost function is optimized, and a particle filter is used to localize the vehicle in OSM. Another approach was proposed in [157] and is based on matching building planes from LiDAR to cuboids in OSM. In [75], a handcrafted feature descriptor, based on buildings and intersections positions is proposed for global localization in OSM. Moreover, several other works attempted to solve the cross-modal localization task on OSM by using cameras rigs instead of LiDAR sensors, such as [158, 52, 159].

Another type of top view 2D maps are satellite maps. These maps provide real-world RGB images captured from an aerial position. In [76], the authors leverage semantic segmentation results from both the LiDAR point cloud and the satellite images in order to optimize the soft cost function of a particle filter. In [160], another particle filter is combined with a similarity network, which was trained to match point clouds that were projected to BEV with satellite map crops. In [78] and [77], a generative adversarial network [79] is trained to generate synthetic top-view LiDAR images based on input satellite crops. The synthetic and real LiDAR images are then both fed to a neural network to predict the value of the displacement between frames. In [161], the authors take advantage of the reflectivity field returned by the LiDAR sensor to match their top view projected scans to aerial imagery maps by using the normalized mutual information (NMI) technique [162]. Although these methods can be seen as "dense" methods because they use most, if not all, the points available to them, other methods, such as [163] try to extract and match features (represented by cropped patches) from both modalities by using a similarity-trained CNN. Although satellite maps might provide a more lifelike image of the surrounding environment, they tend to be cluttered with additional objects that are irrelevant, and sometimes detrimental to the localization results, such as cars and trees. That is the main reason why we chose to use OSM instead, which provides clear boundaries and edges of buildings and roads that our method can rely upon.

Constrained Particle Filters: Particle filter (PF) [164] is a sampling-based method which computes the weights of a set of hypotheses based on an observation and motion model, with a final result consisting of a weighted sum of the previously emitted hypotheses. Particle filters, also called Monte Carlo localization (MCL) [165], are very popular in the robotics field and have been used to solve localization challenges for both indoor [166] and large-scale outdoor scenarios [167].

A localization algorithm such as particle filters, owing to the context in which it is applied (where map information is available), does not suffer as much from drifting problems as some other odometry estimation methods. However, noisy observations and inaccurate motion estimation sometimes can lead to large errors.

A natural improvement to the PF would be to use external constraints to limit the final result within a "feasible" region and thus reduce the error produced by the final weighted result [168]. This is in part why constrained particle filters (CPF) were proposed, leading to multiple strategies. For instance, constraints were applied equally to all the particles in a straightforward manner, named the acceptance/rejection approach [169, 170], where particles that did not respect the constraints were simply discarded. Other methods such as [171] proposed a complex sampling scheme to only draw particles from the feasible region. Later methods focused on applying the constraints to the final weighted result rather than each hypothetical particle. In [172], the authors propose the mean density truncation approach, where an iterative sampling process gets rid of bad particles and samples good ones, thus gradually pushing the final position to the correct region. Improvements to the sampling process and computation cost were subsequently proposed in [173].

5.3 Method

Our goal is to propose a non-learning and reliable method that can be used to achieve accurate LiDAR 2D pose tracking in OSM. In this section, we first explain how to solve the cross-modality issue for comparing OSM and LiDAR point clouds, then we tackle the pose tracking task by using a road-constrained and dual input particle filter. Figure 5.2 shows an overview of our proposed method.

5.3.1 From OSM to LiDAR

Challenges: The biggest challenge when attempting to solve this task is to bridge the modality gap between unsorted LiDAR point clouds and RGB images from OSM. The most common feature between these two representations is the fact that they both contain geometric information on the shapes of the structures that form the surrounding environment of the vehicle, namely roads, buildings, and sometimes trees. On the other hand, the biggest difference is how both formats represent the data: point clouds are unsorted arrays containing the location information, in a predefined frame, of any point that was reached by the LiDAR sensor, whereas OSM visual maps are top-view RGB images containing rough shapes and class data.

The typical first step when attempting to localize LiDAR point clouds that were collected by a vehicle in a 2D aerial map is to start with a BEV projection, which displays the point cloud data from a top view, giving us a more similar visual appearance to the 2D top-view maps, and thus a first step toward breaching the cross-modality. Unfortunately, this step alone is not enough to accurately localize the vehicle in OSM, mainly due to the sparsity of the BEV image and the effects of occlusion of the point cloud data. This is why we need to use the data provided by OSM, namely the roads and buildings, to produce a simulated LiDAR image, which has similar characteristics as the input LiDAR BEV image.

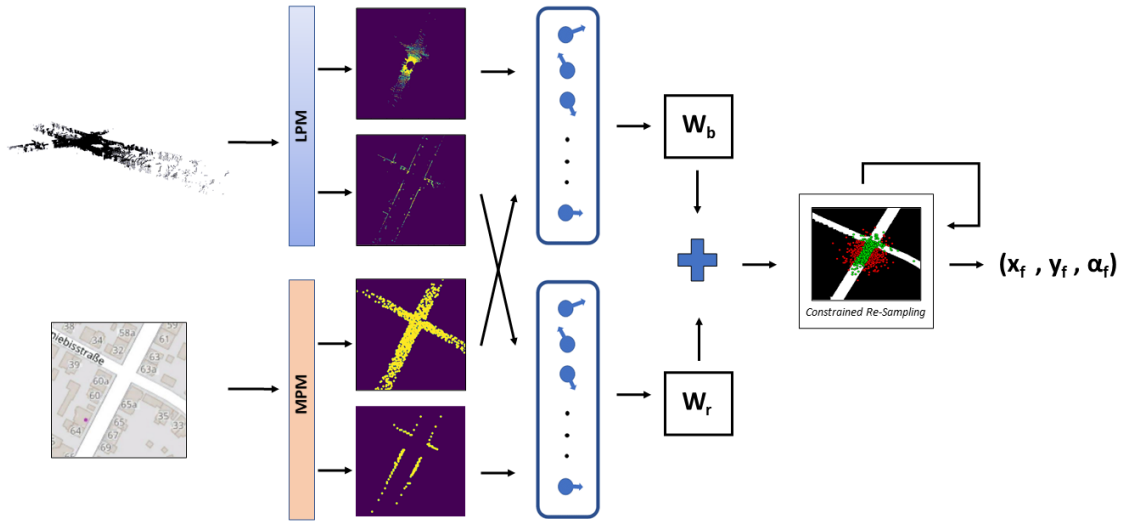


Figure 5.2: Our full method. The LiDAR point cloud and OSM region of interest are processed by the LiDar processing module (LPM) and map processing module (MPM), respectively, to produce four images, a pair of top-view road images and a pair of top-view building edges, with each pair containing a real and a simulated point cloud image. The two pair of images are processed by a dual input particle filter which produces a first estimate of the vehicle position, followed by a road check to verify if the estimated position is on the road or not. In the latter case, the constrained resampling is triggered, until the road check condition is satisfied.

Road and Building Masks: Two of the most clearly labeled classes of objects in OSM are buildings and roads. This information is valuable because those are the classes that are the most present in the point clouds produced by LiDAR sensors when traveling in an urban environment.

By using color segmentation (mostly solid white for roads and a specific shade of grey for buildings), combined with morphological transformations such as dilation and erosion, we are able to generate buildings and road masks that we can later use to generate simulated LiDAR top-view images. When extracting the roads, we use the standard OSM layer style. However, for the buildings, we use the public transport layer, to avoid noise resulting from building numbers and business names. The morphological transformations are mostly applied to the road mask, in order to

close holes caused by overlaid street names on the images, with a kernel size (5,5) for the dilation and (3,3) for the erosion. It is possible generate our own layers with styles that do not include any overlaid text, but that is out of the scope of this work, and we would rather use already available data. The resulting roads and buildings masks of an area in OSM are shown in Figure 5.3.

Figure 5.3: Road and building masks, extracted from OSM.

Simulated and Real LiDAR images: Before attempting to localize the vehicle, our method first starts by generating two pairs of images: a pair of road top-view point cloud images, and a pair of building edges, also from the top view. Both pairs share an important characteristic: they contain an image generated by using the true LiDAR input point cloud, and a simulated LiDAR image generated by using either the road or building masks, previously extracted from OSM. Next, we will explain how the true LiDAR images are generated, in what we call the LiDAR processing module (LPM), followed by how the simulated LiDAR images are generated, in the map processing module (MPM).

In the LPM, we define (x_L, y_L, z_L), the frame attached to the LiDAR sensor, with the z_L axis pointing upward. The LiDAR sensor itself is fixed to the roof of the car, which makes the (x_L, y_L) and ground planes parallel to each other. We first start by splitting our input LiDAR point cloud in two by using the height values z_{Li} of each point: a top section, meant to capture the edges of the buildings reached by the

LiDAR sensor, and a bottom section, which after being filtered by using RANSAC plane fitting [174], represents the road on top of which the vehicle is traveling. The two point clouds are then projected onto the (x_L, y_L) plane to generate two BEV LiDAR images. The LPM can be seen in Figure 5.4.

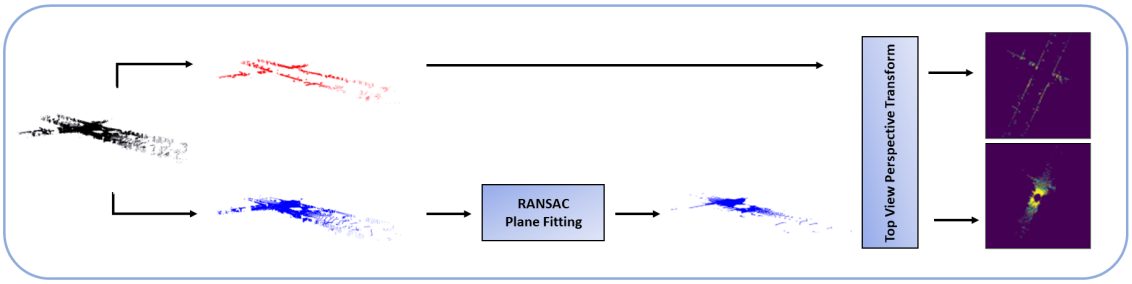


Figure 5.4: LPM. The LiDAR point cloud is divided into two sections using the height value of each point, a top section (capturing surrounding buildings walls) and bottom one (capturing the road). The bottom section undergoes RANSAC plane fitting to extract the road, then the two point clouds are projected to produce two top-view point cloud images.

In the MPM, for the simulated road point cloud, we apply rejection sampling on the road mask with a Gaussian proposal, centered on the supposed vehicle location in OSM (according to the initial guess of the starting position or the previous frame results), and limited to a predefined region of interest. The sampling here is done for two main reasons: first, to simulate the point distribution returned by the LiDAR sensor, which is typically denser around the center of the scan, and becomes sparser the more we move away from the sensor, and second to improve the speed processing, because using all non-zero pixel values in the later calculations would result in a significant slow-down of the whole method.

For the simulated top section of the point cloud, we proceed to use raycasting on the building mask at the same position and region of interest used to generate the simulated road point cloud. Raycasting is a common method used to simulate LiDAR point clouds in autonomous driving cars simulators such as Carla [175]. By

first detecting the obstacles in the surrounding environment (which are shown in the building mask in our case), we can use 2D raycasting and simulate a beam of light that stops when it hits an obstacle. By doing that in a 360-degree fashion, we are able to generate simulated LiDAR images. An example of this approach can be seen in Figure 5.5, and its results are compared with the images from the LiDAR sensor in Figure 5.6. The MPM is illustrated in Figure 5.7.

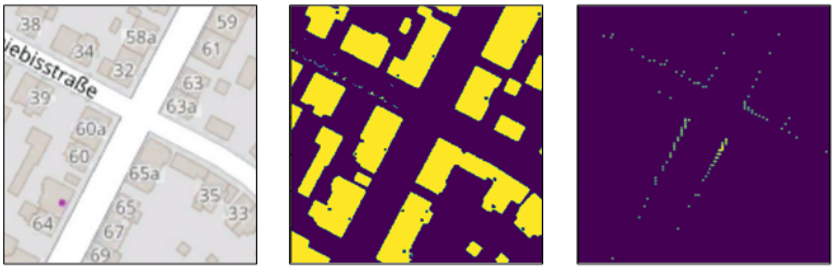


Figure 5.5: Steps of the raycasting process applied to OSM.

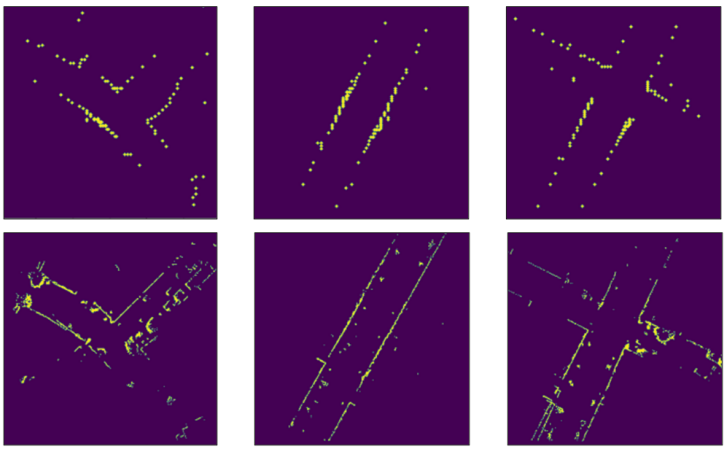


Figure 5.6: Comparison between LiDAR building images (**top**) and simulated LiDAR images by using raycasting (**bottom**).

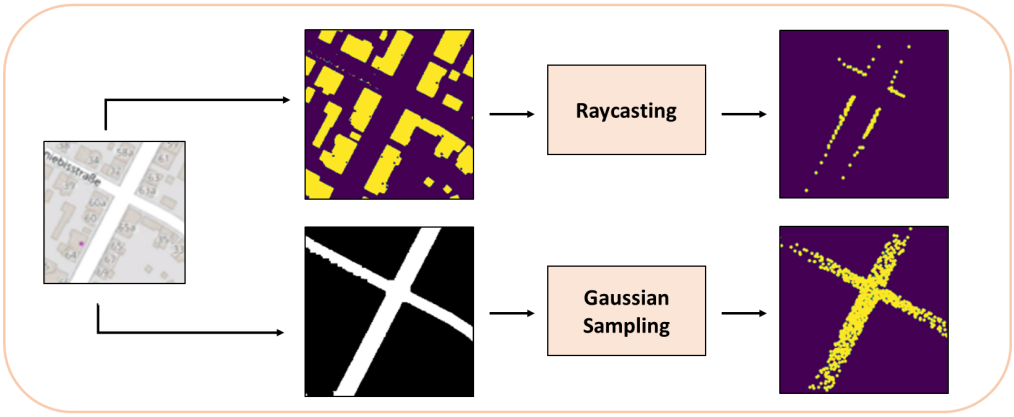


Figure 5.7: MPM. The OSM region of interest is segmented to produce a building and a road mask. Raycasting is applied to the building mask, whereas rejection sampling on the road mask with a Gaussian proposal is applied to the road mask, in order to produce two simulated top-view point cloud images.

Simulated LiDAR and OSM accuracy analysis: In our case, providing an accurate localization result depends heavily on the accuracy of the map used. We start by building a 3D point cloud road map and a building map by using the ground truth odometry and segmentation provided by KITTI (note: this is only used here for visualization purposes and not during the rest of the manuscript), which we then proceed to project onto an image by using the previously discussed BEV projection. Following that, we also build a simulated LiDAR road map and building map from OSM by using color segmentation and raycasting. By overlaying both maps from both modalities, we can see in Figure 5.8 that the maps generated by using OSM and raycasting are pretty accurate and match the maps created by using the LiDAR sensor. In come cases, occlusion of some buildings or the misrepresentation of road width on OSM can cause some mismatches; however, in the majority of cases, the shapes represented in both maps match.

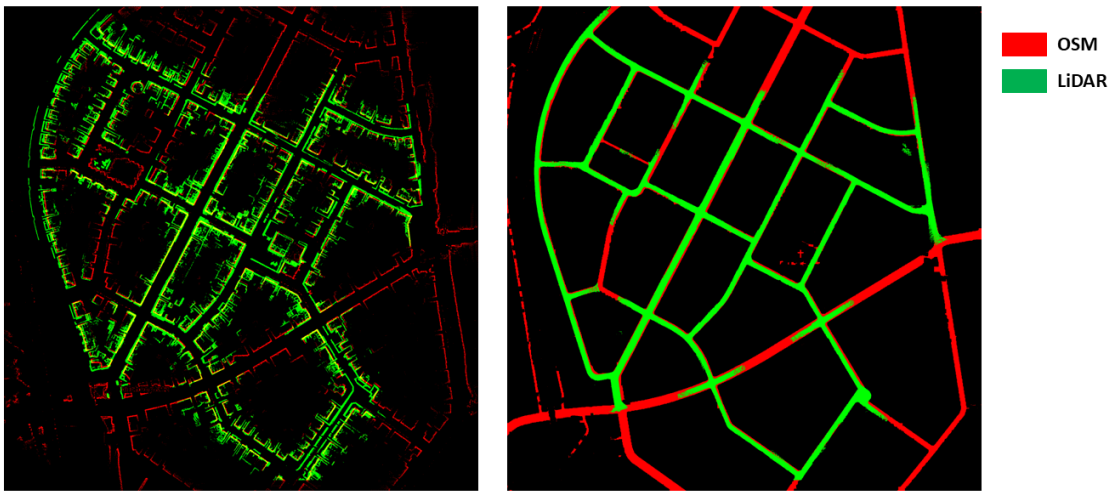


Figure 5.8: Comparison between OSM and LiDAR.

5.3.2 Constrained Particle Filter

Problem Formulation: In order to localize our vehicle's LiDAR in OSM, we use a constrained particle filter. In our configuration, we suppose that we have a guess of the initial pose of the vehicle in OSM (thanks to an external place recognition solution, or a GPS signal that was eventually lost) and each particle corresponds to a hypothetical position (x, y, α). The scale of the map is approximated by using the map size and its corner coordinates. After each LiDAR frame, we rely on our ICP based motion model to predict the particle positions by using two voxel-downsampled 3D LiDAR point clouds, and then update the particles weights by using results from the observation model. Low variance resampling [176] is triggered or not depending on the effective size of the particles [177], and finally the road check and subsequent constrained resampling is used if needed.

Motion Model: Our motion model relies on an ICP-based LiDAR odometry. We approximate the transformation between two subsequent LiDAR point clouds by using the Point-to-Point ICP algorithm [140]. We do not assume access to the

vehicle controls, and only use the ICP output to update the particles positions.

Observation Model: Our observation model is the pre-processed input LiDAR point cloud, so we can match it to our simulated LiDAR images. As it was explained in the previous section, for each frame, we have access to four top-view images, two resulting from the LiDAR point cloud and two resulting from the simulated LiDAR, containing the road and the building edges in both modalities. Inspired by [76], we propose to calculate the weights of our particles based on combining the reciprocal chamfer distance between the pixel locations of the point clouds in each pair of images. The two terms forming the cost function are weighted by the reciprocal of the standard deviation of all the distances calculated above, as a form of uncertainty constraint on the weight's distribution.

In summary, if d_c is the chamfer distance, N the number of particles, z_{r_i} and z_{b_i} the list of all pixel positions of the points in the road and buildings LiDAR point clouds top-view images respectively, transformed according to the ith particle, and m_r and m_b the list of all pixel positions of the points in the top-view images of the simulated point clouds. We define $d_{r_i} = \frac{1}{d_c(z_{r_i}, m_r)}$ and $d_{b_i} = \frac{1}{d_c(z_{b_i}, m_b)}$, and calculate σ_r and σ_b, which are the standard deviations of $d_b = \{d_{r_1}, d_{r_2}, \ldots, d_{r_N}\}$ and $d_r = \{d_{b_1}, d_{b_2}, \ldots, d_{b_N}\}$, respectively.

Finally, the weight of each ith particle can be defined as:

$$W_i = W_{ri} + W_{bi} = \frac{d_{r_i}}{\sigma_r} + \frac{d_{b_i}}{\sigma_b}. \tag{5.1}$$

We choose to use the reciprocal of the chamfer distance in order to emphasize strong particle candidates that align the most with the map. The inclusion of the standard deviation terms is there to favor the data source that produce the most "concentrated" set of particles, which can be interpreted as being less uncertain

about the final position. We attempted to calculate weights by using $exp(-d_c/\lambda)$ instead of $\frac{1}{d_c}$, but found the results to be very sensitive to the value of the parameter λ. We have also tested the use of a joint-probability formulation rather than the summed one, but saw no major difference in the final results.

Constrained resampling: When trying to approximate the correct position of the vehicle, we try to enforce a simple constraint: the vehicle position should always be on the road. This is done in the particle filter through a resampling procedure, inspired by the methods proposed in [173, 172], that aims to gently push the final weighted position toward the correct region (in our case, the road). This is done by enforcing the constraint on the weighted mean value of the particles, rather than the particles themselves.

After updating the weights of the particles, we check to see if the weighted mean vehicle position resulting from the particle filter is on the road. If it is the case, no intervention is needed. Otherwise, we discard one of the particles outside of the road and resample one from the most highly weighted area on the road. Typically, only a few resampling steps (less than 50) are needed to validate our road check. If we reach the maximum resampling iteration, we relax the sampling criteria for the next step of the particle filter. For speed purposes, when resampling good particles to replace bad ones, we use inductive sampling: meaning we resample from the particles that are already available to us. A global view of our constrained particle filter method for localization of LiDAR point clouds in OSM is presented in Algorithm 2 and Figure 5.2.

Algorithm 2 LiDAR-OSM Constrained Particle Filter Localization.

Input: Point cloud P, OSM Region of Interest J.
Output: Vehicle Position (x, y, α)

$Particles \leftarrow ParticleSampling(J)$
$(P_r, P_b) \leftarrow LidarProcessingModule(P)$
$(P'_r, P'_b) \leftarrow MapProcessingModule(J)$
$Weights \leftarrow UpdateWeights(P_r, P_b, P'_r, P'_b)$
If $n_{effectiveSize} < \frac{N}{2}$ **then** $(Particles, Weights) \leftarrow LowVarReSampling(Particles, Weights)$
$(x, y, \alpha) \leftarrow Estimate(Particles, Weights)$

// Split the particles on and off the road
$P_r, W_r, P_{nr}, W_{nr} \leftarrow OnOffRoadSplit(Particles, Weights, J)$

// Constrained Resampling
while $(x, y) \notin Road$ **and** $size(P_{nr}) > 0$
$P_{nr}, W_{nr} \leftarrow DeleteByIndex(P_{nr}, W_{nr}, argmin(W_{nr}))$
$P_r, W_r \leftarrow InductiveSampling(P_r, W_r)$
$Particles \leftarrow Concatenate(P_r, P_{nr})$
$Weights \leftarrow Concatenate(W_r, W_{nr})$
$(x, y, \alpha) \leftarrow Estimate(Particles, Weights)$

return (x, y, α)

5.4 Experiments

5.4.1 Dataset

We use the KITTI Odometry Benchmark [178] to test our method and compare to other cross-modal or road-constrained localization methods. The KITTI dataset was collected in Karlsruhe, Germany by using a VW stationwagon equipped with a variety of sensor modalities such as high-resolution color and grayscale stereo cameras, LiDAR sensor and a high-precision GPS/IMU inertial navigation system. The recorded scenarios are diverse, ranging from urban scenarios to highways or rural areas. Similar to the existing publications, we use the sequences (00, 02, 05, 07, 08, 09, 10) to test our approach. The naming of the sequences used here follows the KITTI Odometry Benchmark references instead of those from the raw KITTI dataset.

5.4.2 Implementation Details

During our experiments, we set the following parameters. Before using ICP, the LiDAR point clouds are first downsampled by using Voxel Downsampling with a resolution of 0.1 m, and we only keep points within a 30-m distance of the car, leaving around 50 k points for each point cloud. In the LPM, and because the LiDAR sensor is placed above the vehicle's roof, we define the LiDAR point cloud top section as any point with a positive elevation value, and the opposite for the bottom section. After the projection to the top view, and before calculating the cost of the particle filter, we drastically downsample the road and building point clouds by only keeping 1000 points by using random sampling. This number was determined through trial and error, by increasing the number of points, until no improvements in accuracy were visible. During testing, we extract tiles from OSM-

Table 5.1: Comparison of the lengths of each of the tested KITTI sequences.

Sequences	**00**	**02**	**05**	**07**	**08**	**09**	**10**
Lengths (kms)	3.72	5.06	2.20	0.8	3.21	1.70	1.5

sized $(320, 320)$ pixels and we use 100 particles to achieve pose tracking. Finally, if one of two simulated point clouds has less than 50 points, we drop that point cloud and only use the second as input to the cost function. This would typically happen in areas where no buildings are around (in the beginning of sequence 09 for example) and only the road can be used to localize our vehicle.

5.4.3 Results

For context, we list the lengths of the tested KITTI sequences in Table 5.1. The results of our cross-modal localization approach, in comparison to other previously proposed methods in the literature are shown in Table 5.2 and 5.3. For both translation and rotation results, we use the accumulated mean error defined as $\frac{\sum_{i=1}^{N} |p_i - \overline{p_i}|}{N}$ to report our results, where p_i and $\overline{p_i}$ are the predicted and ground truth poses, respectively, at the timestamp i. We compare our results to [75], which proposes a method to localize LiDAR point clouds in OSM by using handcrafted feature descriptors, [76] which proposes a method to localize LiDAR point clouds in satellite maps by using the correlation between the semantic segmentation of both modalities and [179], which presents a road-constrained monocular visual localization method. The results tagged [140] are those obtained when using the output of the ICP motion model only without the subsequent proposed observation model. Values marked N/A and rotation error values were not provided by the publications. Figure 5.9 shows a qualitative comparison between our final results and the ground truth.

In addition to the final accuracy that we achieve, if we take the length of the

Table 5.2: Comparison of the translation error on KITTI dataset in meters (m).

Sequences	00		02		05		07		08		09		10	
Lengths (km)	3.72		5.06		2.20		0.8		3.21		1.70		1.5	
	mean	max	mean	max	mean	max	mean	max	mean	max	mean	max	mean	max
[76]	2.0	12.0	9.1	35	N/A	N/A	N/A	N/A	N/A	N/A	7.2	20	N/A	N/A
[75]	20	150	N/A	N/A	25	150	25	120	N/A	N/A	25	50	180	50
[179]	3.7	14.01	11.32	25.6	4.02	8.7	N/A	N/A	4.67	11.96	5.72	11.57	N/A	N/A
[140]	51	162.2	114	307	20.75	70.34	7.07	15.47	62.99	195.22	35.5	96.2	14.73	29.6
Ours	**1.37**	**3.34**	**3.37**	**7.55**	**1.45**	**3.38**	**1.62**	**3.50**	**3.60**	**7.73**	**2.88**	**6.85**	**1.56**	**3.32**

Table 5.3: Comparison of the mean rotation error on KITTI dataset in degrees (°). Best results are in bold.

Sequences	**00**	**02**	**05**	**07**	**08**	**09**	**10**
[140]	58.5	40.2	9.3	13.2	44.3	20.7	6.9
Ours	**1.15**	**3.50**	**2.3**	**1.97**	**3.28**	**2.68**	**2.19**

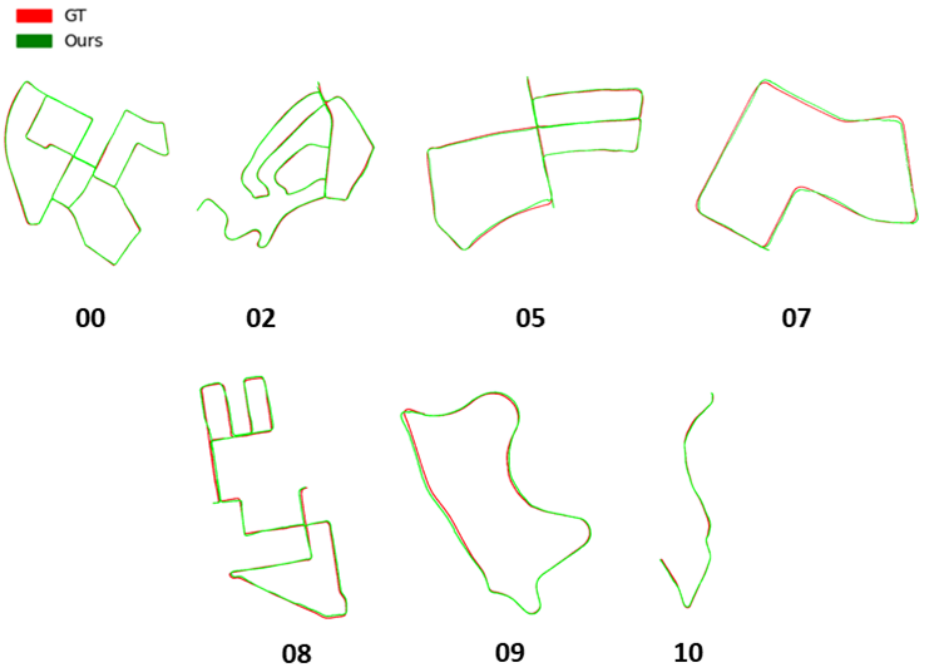


Figure 5.9: Qualitative results of our cross-modal pose tracking method on the KITTI dataset.

sequences into account, our method performs the best on sequence 00. This can be explained by the dense building layout and clear road boundaries of the area. On the other hand, the sequence where we struggle the most is sequence 08. The main reason behind that is the amount of trees and other foreign objects that are picked up in the LiDAR scan, which can produce noisy top-view images, thus leading the PF to wrong estimates. Fortunately, the constraint that we impose helps to keep the error bounded and makes it easier for the PF to correct itself later on.

5.4.4 Discussion

Constrained Particle Filter: As stated above, the main goal of the constrained particle filter is to make sure that our final vehicle location is always on the road. We typically need to enforce this condition when the sensor data collected by the LiDAR is too noisy for the PF to produce accurate results. For example, in Figure 5.10, we show in purple the areas where the road constraint was enforced, and the position corrected by the particle filter. This happens in areas where foreign objects such as tree trunks and thick vegetation are occluding the buildings, or when an intersection structure is not correctly represented on OSM, causing the particle filter to be misled.

Figure 5.10 also shows that our method does not rely on the constraint all the time and/or only tries to stay within the limits of the road. Instead, we are able to track our position correctly most of time by using the input data, and only trigger the constrained resampling in some corner cases, i.e., LiDAR scans are blocked by trees. As an additional example of a good case scenario, we show a region where the localization was successful, which can be explained by the fact that no major occlusion is occurring in that area, and that most structures present on and around the road are correctly represented in OSM.

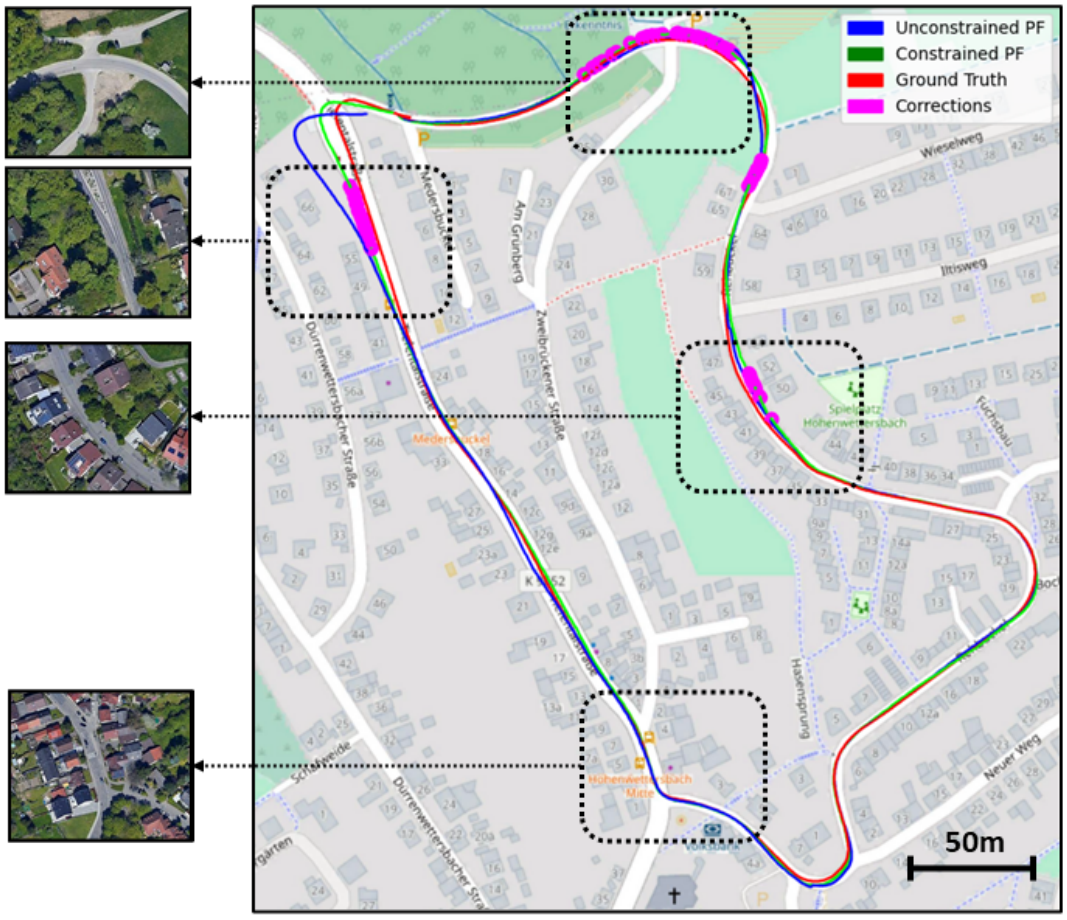


Figure 5.10: The effects of the constrained particle filter on sequence 09 of the KITTI dataset. Here, we show three cases where the constrained particle filter had to correct itself using the road structure, in addition to a case where it successfully estimated the right position using the output of the motion and observation models only.

Finally in Table 5.4, we show results that demonstrate the superiority of our constrained PF formulation to the acceptance/rejection approach used in [170] by comparing the mean translation error over two sequences. The goal here is to compare two ways of applying the constraints in a particle filter: first, applying the constraints to the final estimation of the particle filter, and second, applying the constraints while drawing the hypothesis. The latter can lead to wrong estimations (especially when intersections are involved), which can result in discontinued tra-

Table 5.4: Constraint particle filter mean translation error comparison.

Method	Sequence 05	Sequence 09
Ours (m)	1.4	2.8
Acceptance /Rejection (m)	17.8	3.2

jectories or even total failure of the pose tracking system. In contrast, our approach gently pushes the estimated pose towards the road whenever it is triggered, thus producing a smoother trajectory estimation.

Dual Input Particle Filter: Our method uses both the road and building edges to come up with a guess at vehicle's position by using the PF, preceding the road check validation. We use a standard deviation-based normalizer to give more value to the input, resulting in the best approximation. This is only safe to do if the chamfer distances calculated when using both inputs are correlated, which is verified in the KITTI dataset and demonstrated in Figure 5.11 for both a single random sample and across a full sequence.

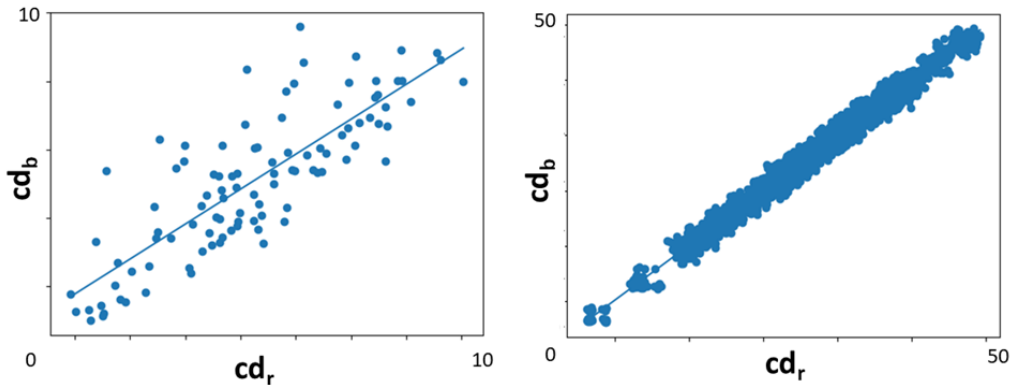


Figure 5.11: Chamfer distance correlation between road and building point clouds. On the right, mean distance values across sequence 05. On the left, distance values for a single random frame in sequence 00.

In Figure 5.12, we show that it is necessary to use both inputs in order to get

the best results possible. Figure 5.12 shows a list of heatmaps for different cases and the use of different weight formulations. We can see from the first case that of the weights resulting from the building edges, only d_b tend to result in a coarse distribution, whereas for the road only d_r tend to be smoother. This shows that both sets of weights, when combined, can be complimentary and result in a better final weighted sum position. While the heatmaps of the weights resulting from the buildings only in the first case could be interpreted as "more accurate" than the ones produced by the road, the second case that we show presents a situation where the buildings are occluded, resulting in noisy set of weights. If we only had access to these weights, then our position estimation would be way off compared to the results, we would get by combining both sets of weights, because the weights resulting from the road point cloud are reasonable in most cases. Finally, we show a case where both inputs result in distributions that are slightly off, whereas their combination is much more centered around the vehicle's true position.

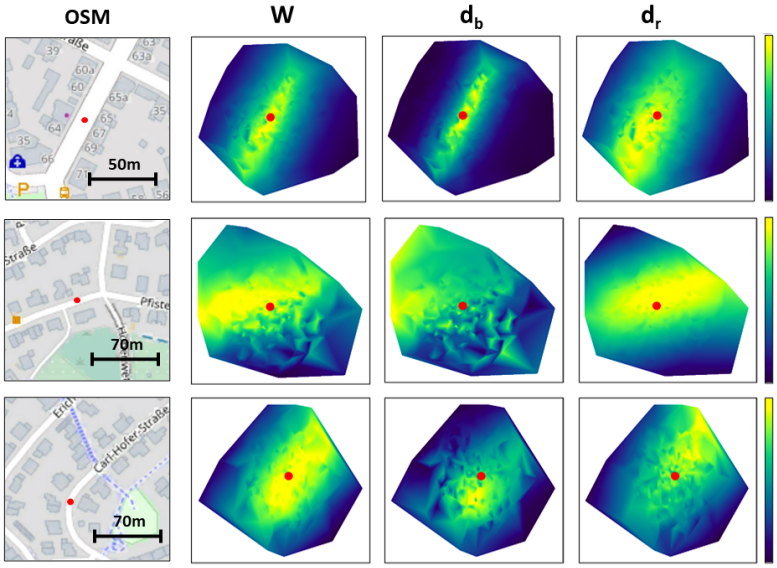


Figure 5.12: Interpolated heatmaps representing the weight distribution of the particles for different weights formulation, according to Equation (5.1). Red dots represent the true vehicle position.

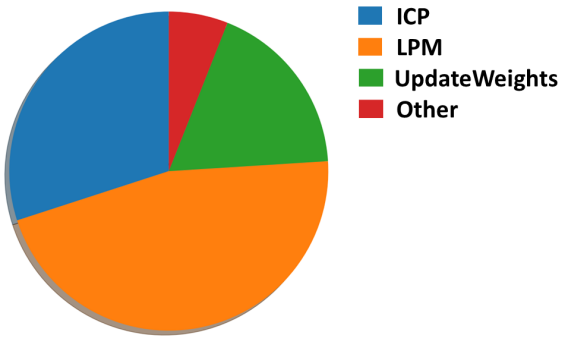


Figure 5.13: Runtime distribution.

Note that it is also very useful to have access to both, in areas where only one of them is available: For example, at the beginning of sequence 09, no buildings are available and only the road can be used, as can be seen in top area of Figure 5.10.

Runtime analysis: Our method runs at a maximum speed of 10Hz. The LPM takes the most amount of time with more than forty percent of the total runtime, followed by ICP with thirty percent and the $UpdateWeights$ step with a little bit less than twenty percent. The constrained resampling, when triggered, consumes less than one percent of the total runtime on average. Figure 5.13 shows the runtime distribution for each part of the algorithm.

Robustness: We apply voxel-downsampling to all the sensor inputs in order to speed up the whole pipeline. Table 5.5 shows a comparison of the results of our method on sequences 07 and 10 when using different voxel sizes. These results show that the voxel size has little effect on the accuracy of our method, giving it the potential to work with LiDARs that produce sparser point clouds than the ones used in this study.

We also show a comparison in Table 5.6 between the use of random sampling and voxel sampling. Our results show that voxel downsampling is superior, which can be explained by its ability to conserve the general structure of the point clouds.

Table 5.5: Voxel-downsampling resolution and mean translation (m) error comparison.

Voxel Resolution (m)	**Sequence 07**	**Sequence 10**
0.01	1.5	1.6
0.1	1.6	1.5
1	1.4	1.7

Table 5.6: Voxel-downsampling and random sampling mean translation error (m) comparison.

Sampling Method	**Sequence 00**	**Sequence 07**	**Sequence 09**
Random Sampling	2.53	3.59	9.69
Voxel Sampling	1.37	1.62	2.88

In contrast, random sampling can sometime return wildly different point clouds from one frame to the other, leading to a decrease in accuracy when using ICP as a motion model, but it also can end up selecting most its points from a noisy or occluded area, which can negatively affect the accuracy of the particle filter.

5.5 Summary

This work proposes a general non-learning method for vehicle localization by using LiDAR and OSM. We first validate the accuracy level of the OSM data needed for vehicle localization. Subsequently, buildings and road masks are extracted to generate simulated LiDAR images by using raycasting. These images are compared with the BEV projection of the input LiDAR point cloud by using particle filters. In addition, the extracted road mask is applied as a constraint of the vehicle location, by making sure that the output location of our system is on the road, thus keeping our location error bounded. The evaluation results on KITTI show that the proposed method outperforms other existing cross-modal and road-constrained localization

methods that were based on OSM and/or satellite maps. We also provide a detailed technical discussion and ablation studies to explain the advantages of the proposed method and demonstrate its ability to produce accurate pose tracking results. Future work will include a better way of dealing with occlusions in the sensor data, in addition to sensor fusion methods for more precise localization.

Chapter 6

Conclusion

In this thesis, we covered three of the main building blocks, used in most autonomous robots: mapping, labeling and localization. Contrary to what is typically done in most commercial applications, and instead of relying on a single visual sensor as main data source during the three phases mentioned above, we explored the possibility of using cross-modal approaches, that bridge between camera and laser-based sensors. This makes it possible for us to deploy a minimum amount of sensors and save on cost, time and processing power.

Our first challenge what to solve the cross-modal mapping task of generating 3D point clouds using camera images captured by UAVs. This was done by proposing a two-stage height prediction pipeline, which combines a multi-task network with a denosing autoencoder, in order to predict state of the art height values. The multi-task network used in the first stage, combines the semantic labels and surface normals branches with the height prediction branch to generate a coarse height map. In the second stage, the denoising autoencoder combines all the predictions of the first stage, and generates a final height map, which is much accurate and noise resilient than the first one. These final height maps can then be used either by

themselves to generate 3D point clouds of the areas captured by the UAV or combined with the available semantic and surface normals prediction to build semantic point clouds or 3D mesh maps.

The next cross-modal task that we tackled was the labeling of pre-built point cloud maps using camera images. Commonly done using manual labeling, this can be both expensive and time consuming. We proposed instead, to use images captures using an onboard camera, to detect relevant features using deep learning, before processing them to remove noise and projecting them onto the pre-built 3D point cloud maps. Thus, we are able to automatically generate accurate and abstracted road and lanes labels, that can be used by a self-driving car, to guide itself in an urban environment. This work can be extended on the future, by adding the detection and labeling of other road features, mainly traffic sign and road markings.

Finally, we also explored the cross-modal localization challenge of LiDAR localization in OSM. Due to the scarcity of HD maps around the world, we propose to use OSM as a map instead, since it is freely available almost everywhere and is routinely updated by its own users. Thanks to semantic segmentation, and by relying on the raycasting technique, we are able to use OSM to generate simulated 2D point clouds. This made it possible for us to compare the output of the LiDAR sensor with the OSM data, and thus localize the point clouds in OSM. We also use the road boundaries extracted from OSM, as constrain to keep our localization error bounded. This was made possible thanks to a constrained formulation of the particle filter algorithm. As a results, we were able to obtain the state-of-the-art accuracy, when compared to all the recent LiDAR localization methods in OSM or satellite maps.

Bibliography

[1] Ben Zhang and Denglin Zhu. A stereo slam system with dense mapping. *IEEE Access*, 9:151888–151896, 2021.

[2] J.M. Saez and F. Escolano. A global 3d map-building approach using stereo vision. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 2, pages 1197–1202 Vol.2, 2004.

[3] Francisco Rovira-Más, Qin Zhang, and John F Reid. Stereo vision three-dimensional terrain maps for precision agriculture. *Computers and electronics in agriculture*, 60(2):133–143, 2008.

[4] Shuhuan Wen, Xin Liu, Hong Zhang, Fuchun Sun, Miao Sheng, and Shaokang Fan. Dense point cloud map construction based on stereo vins for mobile vehicles. *ISPRS Journal of Photogrammetry and Remote Sensing*, 178:328–344, 2021.

[5] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[6] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual net-

works. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.

[7] Hamed Amini Amirkolaee and Hossein Arefi. Height estimation from single aerial images using a deep convolutional encoder-decoder network. volume 149, pages 50–66. Elsevier, 2019.

[8] Jing Huang and Suya You. Point cloud labeling using 3d convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2670–2675, 2016.

[9] Liqiang Zhang, Zhuqiang Li, Anjian Li, and Fangyu Liu. Large-scale urban point cloud labeling and reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 138:86–100, 2018.

[10] Yicong Tian, Chen Chen, and Mubarak Shah. Cross-view image matching for geo-localization in urban environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3616, 2017.

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[12] Sixing Hu, Mengdan Feng, Rang MH Nguyen, and Gim Hee Lee. Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7258–7267, 2018.

[13] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Pro-*

ceedings of the IEEE conference on computer vision and pattern recognition, pages 5297–5307, 2016.

[14] Liu Liu and Hongdong Li. Lending orientation to neural networks for cross-view geo-localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5624–5633, 2019.

[15] Sijie Zhu, Taojiannan Yang, and Chen Chen. Revisiting street-to-aerial view image geo-localization and orientation estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 756–765, 2021.

[16] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[17] Yujiao Shi, Xin Yu, Liu Liu, Tong Zhang, and Hongdong Li. Optimal feature transport for cross-view image geo-localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11990–11997, 2020.

[18] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[19] Yulan Guo, Michael Choi, Kunhong Li, Farid Boussaid, and Mohammed Bennamoun. Soft exemplar highlighting for cross-view image-based geo-localization. *IEEE Transactions on Image Processing*, 31:2094–2105, 2022.

[20] Sijie Zhu, Taojiannan Yang, and Chen Chen. Vigor: Cross-view image geo-localization beyond one-to-one retrieval. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 3640–3649, 2021.

[21] Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention mechanisms in computer vision: A survey. *Computational Visual Media*, pages 1–38, 2022.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[23] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[24] Yujiao Shi, Liu Liu, Xin Yu, and Hongdong Li. Spatial-aware feature aggregation for image based cross-view geo-localization. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] Hongji Yang, Xiufan Lu, and Yingying Zhu. Cross-view geo-localization with layer-to-layer transformer. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] Sijie Zhu, Mubarak Shah, and Chen Chen. Transgeo: Transformer is all you need for cross-view image geo-localization. *arXiv preprint arXiv:2204.00097*, 2022.

[27] Michael Bosse and Robert Zlot. Place recognition using keypoint voting in large 3d lidar datasets. In *2013 IEEE International Conference on Robotics and Automation*, pages 2677–2684. IEEE, 2013.

[28] Stanley Bileschi and Lior Wolf. Image representations beyond histograms of gradients: The role of gestalt descriptors. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[29] Marian Himstedt, Jan Frost, Sven Hellbach, Hans-Joachim Böhme, and Erik Maehle. Large scale place recognition in 2d lidar scans using geometrical landmark relations. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5030–5035. IEEE, 2014.

[30] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart, and Cesar Cadena. Segmatch: Segment based place recognition in 3d point clouds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5266–5272. IEEE, 2017.

[31] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[32] Kavisha Vidanapathirana, Peyman Moghadam, Ben Harwood, Muming Zhao, Sridha Sridharan, and Clinton Fookes. Locus: Lidar-based place recognition using spatiotemporal higher-order pooling. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5075–5081. IEEE, 2021.

[33] Jiadong Guo, Paulo VK Borges, Chanoh Park, and Abel Gawel. Local descriptor for robust place recognition using lidar intensity. *IEEE Robotics and Automation Letters*, 4(2):1470–1477, 2019.

[34] Tixiao Shan, Brendan Englot, Fábio Duarte, Carlo Ratti, and Daniela Rus. Robust place recognition using an imaging lidar. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5469–5475. IEEE, 2021.

[35] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[36] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):591–606, 2008.

[37] Lun Luo, Si-Yuan Cao, Bin Han, Hui-Liang Shen, and Junwei Li. Bvmatch: Lidar-based place recognition using bird's-eye view images. *IEEE Robotics and Automation Letters*, 6(3):6076–6083, 2021.

[38] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces, 1994.

[39] Giseop Kim and Ayoung Kim. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809. IEEE, 2018.

[40] Huan Yin, Li Tang, Xiaqing Ding, Yue Wang, and Rong Xiong. Locnet: Global localization in 3d point clouds for mobile vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 728–733. IEEE, 2018.

[41] Huan Yin, Yue Wang, Xiaqing Ding, Li Tang, Shoudong Huang, and Rong Xiong. 3d lidar-based global localization using siamese neural network. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1380–1392, 2019.

[42] Lin Li, Xin Kong, Xiangrui Zhao, Tianxin Huang, Wanlong Li, Feng Wen, Hongbo Zhang, and Yong Liu. Rinet: Efficient 3d lidar-based place recognition using rotation invariant neural network. *IEEE Robotics and Automation Letters*, 7(2):4321–4328, 2022.

[43] Yanhao Li and Hao Li. Lidar-based initial global localization using two-dimensional (2d) submap projection image (spi). In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5063–5068. IEEE, 2021.

[44] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[45] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[46] Kai Fischer, Martin Simon, Stefan Milz, and Patrick Mäder. Stickylocalization: Robust end-to-end relocalization on point clouds using graph neural networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2962–2971, 2022.

[47] Xin Kong, Xuemeng Yang, Guangyao Zhai, Xiangrui Zhao, Xianfang Zeng, Mengmeng Wang, Yong Liu, Wanlong Li, and Feng Wen. Semantic graph based place recognition for 3d point clouds. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8216–8223. IEEE, 2020.

[48] Tim Y Tang, Daniele De Martini, and Paul Newman. Get to the point: Learning lidar place recognition and metric localisation using overhead imagery. *Proceedings of Robotics: Science and Systems, 2021.*, 2021.

[49] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[50] Younghun Cho, Giseop Kim, Sangmin Lee, and Jee-Hwan Ryu. Openstreetmap-based lidar global localization in urban environment without a prior lidar map. *IEEE Robotics and Automation Letters*, 7(2):4999–5006, 2022.

[51] Marcus A Brubaker, Andreas Geiger, and Raquel Urtasun. Map-based probabilistic visual self-localization. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):652–665, 2015.

[52] Augusto Luis Ballardini, Daniele Cattaneo, Simone Fontana, and Domenico Giorgio Sorrenti. Leveraging the osm building data to enhance the localization of an urban vehicle. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 622–628. IEEE, 2016.

[53] Ryan W Wolcott and Ryan M Eustice. Fast lidar localization using multiresolution gaussian mixture maps. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2814–2821. IEEE, 2015.

[54] Dong-Ki Kim and Matthew R Walter. Satellite image-based localization via learned embeddings. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2073–2080. IEEE, 2017.

[55] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[56] Robert Spangenberg, Daniel Goehring, and Raúl Rojas. Pole-based localization for autonomous vehicles in urban scenarios. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2161–2166. IEEE, 2016.

[57] Zimin Xia, Olaf Booij, Marco Manfredi, and Julian FP Kooij. Cross-view matching for vehicle localization by learning geographically local representations. *IEEE Robotics and Automation Letters*, 6(3):5921–5928, 2021.

[58] Yujiao Shi, Xin Yu, Liu Liu, Dylan Campbell, Piotr Koniusz, and Hongdong Li. Accurate 3-dof camera geo-localization via ground-to-satellite image matching. *arXiv preprint arXiv:2203.14148*, 2022.

[59] Pratik Agarwal, Wolfram Burgard, and Luciano Spinello. Metric localization using google street view. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3111–3118. IEEE, 2015.

[60] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: science and systems*, volume 4, page 1. Citeseer, 2007.

[61] Keisuke Yoneda, Hossein Tehrani, Takashi Ogawa, Naohisa Hukuyama, and Seiichi Mita. Lidar scan feature for localization with highly precise 3-d map. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 1345–1350. IEEE, 2014.

[62] Juan Castorena and Siddharth Agarwal. Ground-edge-based lidar localization without a reflectivity calibration for autonomous driving. *IEEE Robotics and Automation Letters*, 3(1):344–351, 2017.

[63] Chen Zhang, Marcelo H Ang, and Daniela Rus. Robust lidar localization for autonomous driving in rain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3409–3415. IEEE, 2018.

[64] Martin Magnusson. *The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection*. PhD thesis, Örebro universitet, 2009.

[65] Ehsan Javanmardi, Mahdi Javanmardi, Yanlei Gu, and Shunsuke Kamijo. Autonomous vehicle self-localization based on multilayer 2d vector map and multi-channel lidar. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 437–442. IEEE, 2017.

[66] Farouk Ghallabi, Fawzi Nashashibi, Ghayath El-Haj-Shhade, and Marie-Anne Mittet. Lidar-based lane marking detection for vehicle positioning in an hd map. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2209–2214. IEEE, 2018.

[67] Farouk Ghallabi, Ghayath El-Haj-Shhade, Marie-Anne Mittet, and Fawzi Nashashibi. Lidar-based road signs detection for vehicle localization in an hd

map. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1484–1490. IEEE, 2019.

[68] Guang Chen, Fan Lu, Zhijun Li, Yinlong Liu, Jinhu Dong, Junqiao Zhao, Junwei Yu, and Alois Knoll. Pole-curb fusion based robust and efficient autonomous vehicle localization system with branch-and-bound global optimization and local grid map method. *IEEE Transactions on Vehicular Technology*, 70(11):11283–11294, 2021.

[69] Dávid Rozenberszki and András L Majdik. Lol: Lidar-only odometry and localization in 3d point cloud maps. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4379–4385. IEEE, 2020.

[70] Yufeng Yue, Chunyang Zhao, Mingxing Wen, Zhenyu Wu, and Danwei Wang. Collaborative semantic perception and relative localization based on map matching. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6188–6193. IEEE, 2020.

[71] Xieyuanli Chen, Thomas Läbe, Andres Milioto, Timo Röhling, Jens Behley, and Cyrill Stachniss. Overlapnet: a siamese network for computing lidar scan similarity with applications to loop closing and localization. *Autonomous Robots*, pages 1–21, 2021.

[72] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-net: Towards learning based lidar localization for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6389–6398, 2019.

[73] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition, pages 652–660, 2017.

[74] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Attention-based vehicle self-localization with hd feature maps. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 76–83. IEEE, 2021.

[75] Fan Yan, Olga Vysotska, and Cyrill Stachniss. Global localization on openstreetmap using 4-bit semantic descriptors. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–7. IEEE, 2019.

[76] Ian D Miller, Anthony Cowley, Ravi Konkimalla, Shreyas S Shivakumar, Ty Nguyen, Trey Smith, Camillo Jose Taylor, and Vijay Kumar. Any way you look at it: Semantic crossview localization and mapping with lidar. *IEEE Robotics and Automation Letters*, 6(2):2397–2404, 2021.

[77] Tim Yuqing Tang, Daniele De Martini, Dan Barnes, and Paul Newman. Rsl-net: Localising in satellite images from a radar on the ground. *IEEE Robotics and Automation Letters*, 5(2):1087–1094, 2020.

[78] Tim Y Tang, Daniele De Martini, Shangzhe Wu, and Paul Newman. Self-supervised localisation between range sensors and overhead imagery. *arXiv preprint arXiv:2006.02108*, 2020.

[79] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[80] Youngji Kim, Jinyong Jeong, and Ayoung Kim. Stereo camera localization in 3d lidar maps. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[81] Ryan W Wolcott and Ryan M Eustice. Visual localization within lidar maps for automated urban driving. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183. IEEE, 2014.

[82] Daniele Cattaneo, Matteo Vaghi, Augusto Luis Ballardini, Simone Fontana, Domenico G Sorrenti, and Wolfram Burgard. Cmrnet: Camera to lidar-map registration. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1283–1289. IEEE, 2019.

[83] Daniele Cattaneo, Domenico Giorgio Sorrenti, and Abhinav Valada. Cmrnet++: Map and camera agnostic monocular visual localization in lidar maps. *arXiv preprint arXiv:2004.13795*, 2020.

[84] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[85] Kaiqiang Chen, Kun Fu, Menglong Yan, Xin Gao, Xian Sun, and Xin Wei. Semantic segmentation of aerial images with shuffling convolutional neural networks. *IEEE Geoscience and Remote Sensing Letters*, 15(2):173–177, 2018.

[86] Dimitrios Marmanis, Jan D Wegner, Silvano Galliani, Konrad Schindler, Mihai Datcu, and Uwe Stilla. Semantic segmentation of aerial images with an ensemble of cnss. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 2016*, 3:473–480, 2016.

[87] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning roi transformer for oriented object detection in aerial images. In *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition, pages 2849–2858, 2019.

[88] Igor Ševo and Aleksej Avramović. Convolutional neural network based automatic object detection on aerial images. *IEEE geoscience and remote sensing letters*, 13(5):740–744, 2016.

[89] Bertrand Le Saux, Naoto Yokoya, Ronny Hansch, and Saurabh Prasad. 2018 ieee grss data fusion contest: Multimodal land use classification [technical committees]. *IEEE geoscience and remote sensing magazine*, 6(1):52–54, 2018.

[90] Bertrand Le Saux, Naoto Yokoya, Ronny Hansch, Myron Brown, and Greg Hager. 2019 data fusion contest [technical committees]. *IEEE Geoscience and Remote Sensing Magazine*, 7(1):103–105, 2019.

[91] Naoto Yokoya, Pedram Ghamisi, Ronny Hänsch, and Michael Schmitt. 2020 ieee grss data fusion contest: Global land cover mapping with weak supervision [technical committees]. *IEEE Geoscience and Remote Sensing Magazine (GRSM)*, 8(1):154–157, 2020.

[92] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with a contrario model estimation. In *Asian Conference on Computer Vision*, pages 257–270. Springer, 2012.

[93] Pierre Moulon and Pascal Monasse. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3248–3255, 2013.

[94] Marcela Carvalho, Bertrand Le Saux, Pauline Trouvé-Peloux, Frédéric Champagnat, and Andrés Almansa. Multitask learning of height and semantics from aerial images. IEEE, 2019.

[95] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[96] Marvin Teichmann, Michael Weber, Marius Zoellner, Roberto Cipolla, and Raquel Urtasun. Multinet: Real-time joint semantic reasoning for autonomous driving. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 1013–1020. IEEE, 2018.

[97] Dingfu Zhou, Jin Fang, Xibin Song, Liu Liu, Junbo Yin, Yuchao Dai, Hongdong Li, and Ruigang Yang. Joint 3d instance segmentation and object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[98] Pedram Ghamisi and Naoto Yokoya. Img2dsm: Height simulation from single imagery using conditional generative adversarial net. volume 15, pages 794–798. IEEE, 2018.

[99] Chao-Jung Liu, Vladimir A Krylov, Paul Kane, Geraldine Kavanagh, and Rozenn Dahyot. Im2elevation: Building height estimation from single-view aerial imagery. *Remote Sensing*, 12(17):2719, 2020.

[100] Shivangi Srivastava, Michele Volpi, and Devis Tuia. Joint height estimation and semantic labeling of monocular aerial images with cnns. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 5173–5176. IEEE, 2017.

[101] Thanuja Dharmasiri, Andrew Spek, and Tom Drummond. Joint prediction of depths, normals and surface curvature from rgb images using cnns. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1505–1512. IEEE, 2017.

[102] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[103] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction.

[104] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999.

[105] Linwei Fan, Fan Zhang, Hui Fan, and Caiming Zhang. Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art*, 2(1):7, 2019.

[106] Rafael C Gonzalez, Richard Eugene Woods, and Steven L Eddins. *Digital image processing using MATLAB*. Pearson Education India, 2004.

[107] Anil K Jain. *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

[108] Jacob Benesty, Jingdong Chen, and Yiteng Huang. Study of the widely linear wiener filter for noise reduction. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 205–208. IEEE, 2010.

[109] Ioannis Pitas and Anastasios N Venetsanopoulos. *Nonlinear digital filters: principles and applications*, volume 84. Springer Science & Business Media, 2013.

[110] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. *Bilateral filtering: Theory and applications*. Now Publishers Inc, 2009.

[111] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[112] Guy Gilboa and Stanley Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling & Simulation*, 7(3):1005–1028, 2009.

[113] Ivan Markovsky and KONSTANTIN Usevich. *Low rank approximation*, volume 139. Springer, 2012.

[114] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[115] Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. Autoencoder for words. volume 139, pages 84–96. Elsevier, 2014.

[116] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International conference on artificial neural networks*, pages 44–51. Springer, 2011.

[117] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. volume 11, 2010.

[118] Yonghao. Xu et al. Advanced multi-sensor optical remote sensing for urban land use and land cover classification: Outcome of the 2018 ieee grss data fusion contest. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(6):1709–1724, 2019.

[119] Michael Cramer. The dgpf-test on digital airborne camera evaluation–overview and test design. *Photogrammetrie-Fernerkundung-Geoinformation*, 2010(2):73–82, 2010.

[120] Markus Gerke. Use of the stair vision library within the isprs 2d semantic labeling benchmark. 2014.

[121] Irwin Sobel. An isotropic 3x3 image gradient operator. *Presentation at Stanford A.I. Project 1968*, 02 2014.

[122] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[123] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[124] Daniele Cerra, Miguel Pato, Emiliano Carmona, Seyed Majid Azimi, Jiaojiao Tian, Reza Bahmanyar, Franz Kurz, Eleonora Vig, Ksenia Bittner, Corentin Henry, et al. Combining deep and shallow neural networks with ad hoc detectors for the classification of complex multi-modal urban scenes. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 3856–3859. IEEE, 2018.

[125] Yonghao Xu, Bo Du, and Liangpei Zhang. Multi-source remote sensing data classification via fully convolutional networks and post-classification processing. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 3852–3855. IEEE, 2018.

[126] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[127] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[128] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[129] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[130] National Highway Traffic Safety Administration et al. Automated driving systems 2.0: A vision for safety. *Washington, DC: US Department of Transportation, DOT HS*, 812:442, 2017.

[131] Ze Wang, Weiqiang Ren, and Qiang Qiu. Lanenet: Real-time lane detection networks for autonomous driving. *arXiv preprint arXiv:1807.01726*, 2018.

[132] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[133] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[134] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[135] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[136] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[137] Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Intelligent autonomous systems 13*, pages 335–348. Springer, 2016.

[138] Ji Zhang and Sanjiv Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, 2017.

[139] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4758–4765. IEEE, 2018.

[140] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[141] Martin Magnusson, Achim Lilienthal, and Tom Duckett. Scan registration for autonomous mining vehicles using 3d-ndt. *Journal of Field Robotics*, 24(10):803–827, 2007.

[142] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.

[143] Yecheng Lyu, Lin Bai, Mahdi Elhousni, and Xinming Huang. An interactive lidar to camera calibration. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6. IEEE, 2019.

[144] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[145] Adriano Moreira and Maribel Yasmina Santos. Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. 2007.

[146] Radu Bogdan Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

[147] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.

[148] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019.

[149] Younggun Cho, Giseop Kim, and Ayoung Kim. Deeplo: Geometry-aware deep lidar odometry. *arXiv preprint arXiv:1902.10562*, 2019.

[150] Xieyuanli Chen, Ignacio Vizzo, Thomas Läbe, Jens Behley, and Cyrill Stachniss. Range image-based lidar localization for autonomous vehicles. *arXiv preprint arXiv:2105.12121*, 2021.

[151] Mahdi Elhousni, Yecheng Lyu, Ziming Zhang, and Xinming Huang. Automatic building and labeling of hd maps with deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13255–13260, 2020.

[152] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org , 2017.

[153] Krista Merry and Pete Bettinger. Smartphone gps accuracy study in an urban environment. *PloS one*, 14(7):e0219890, 2019.

[154] Michael G Wing, Aaron Eklund, and Loren D Kellogg. Consumer-grade global positioning system (gps) accuracy and reliability. *Journal of forestry*, 103(4):169–173, 2005.

[155] Mahdi Elhousni and Xinming Huang. A survey on 3d lidar localization for autonomous vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1879–1884, 2020.

[156] Philipp Ruchti, Bastian Steder, Michael Ruhnke, and Wolfram Burgard. Localization on openstreetmap data using a 3d laser scanner. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5260–5265. IEEE, 2015.

[157] Christian Landsiedel and Dirk Wollherr. Global localization of 3d point clouds in building outline maps of urban outdoor environments. *International journal of intelligent robotics and applications*, 1(4):429–441, 2017.

[158] Augusto Luis Ballardini, Simone Fontana, Axel Furlan, Dario Limongi, and Domenico Giorgio Sorrenti. A framework for outdoor urban environment estimation. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2721–2727. IEEE, 2015.

[159] Augusto Luis Ballardini, Simone Fontana, Daniele Cattaneo, Matteo Matteucci, and Domenico Giorgio Sorrenti. Vehicle localization using 3d building models and point cloud matching. *Sensors*, 21(16):5356, 2021.

[160] Mengyin Fu, Minzhao Zhu, Yi Yang, Wenjie Song, and Meiling Wang. Lidar-based vehicle localization on the satellite image via a neural network. *Robotics and Autonomous Systems*, 129:103519, 2020.

[161] Ankit Vora, Siddharth Agarwal, Gaurav Pandey, and James McBride. Aerial imagery based lidar localization for autonomous vehicles. *arXiv preprint arXiv:2003.11192*, 2020.

[162] Frederik Maes, Andre Collignon, Dirk Vandermeulen, Guy Marchal, and Paul Suetens. Multimodality image registration by maximization of mutual information. *IEEE transactions on Medical Imaging*, 16(2):187–198, 1997.

[163] Mykhail Uss, Benoit Vozel, Vladimir Lukin, and Kacem Chehdi. Efficient discrimination and localization of multimodal remote sensing images using cnn-based prediction of localization uncertainty. *Remote Sensing*, 12(4):703, 2020.

[164] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1-2):99–141, 2001.

[165] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 2, pages 1322–1328. IEEE, 1999.

[166] Henri Nurminen, Anssi Ristimäki, Simo Ali-Löytty, and Robert Piché. Particle filter and smoother for indoor localization. In *International Conference on Indoor Positioning and Indoor Navigation*, pages 1–10. IEEE, 2013.

[167] Martin Adams, Sen Zhang, and Lihua Xie. Particle filter based outdoor robot localization using natural features extracted from laser scanners. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 2, pages 1493–1498. IEEE, 2004.

[168] Bradley Ebinger, Nidhal Bouaynaya, Robi Polikar, and Roman Shterenberg. Constrained state estimation in particle filters. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4050–4054, 2015.

[169] Lixin Lang, Wen-shiang Chen, Bhavik R Bakshi, Prem K Goel, and Sridhar Ungarala. Bayesian estimation via sequential monte carlo sampling—constrained dynamic systems. *Automatica*, 43(9):1615–1622, 2007.

[170] Hyondong Oh and Seungkeun Kim. Persistent standoff tracking guidance using constrained particle filter for multiple uavs. *Aerospace Science and Technology*, 84:257–264, 2019.

[171] Zhonggai Zhao, Biao Huang, and Fei Liu. Constrained particle filtering methods for state estimation of nonlinear process. *AIChE Journal*, 60(6):2072–2082, 2014.

[172] Nesrine Amor, Nidhal Bouaynaya, Petia Georgieva, Roman Shterenberg, and Souad Chebbi. Eeg dynamic source localization using constrained particle filtering. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. IEEE, 2016.

[173] N Amor, G Rasool, N Bouaynaya, and R Shterenberg. Hand movement discrimination using particle filters. In *2018 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–5. IEEE, 2018.

[174] Orazio Gallo, Roberto Manduchi, and Abbas Rafii. Cc-ransac: Fitting planes in the presence of multiple surfaces in range data. *Pattern Recognition Letters*, 32(3):403–410, 2011.

[175] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[176] Simon Parsons. Probabilistic robotics by sebastian thrun, wolfram burgard and dieter fox, mit press, isbn 0-262-20162-3. *The Knowledge Engineering Review*, 21(3):287–289, 2006.

[177] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE transactions on Robotics*, 23(1):34–46, 2007.

[178] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[179] Shuai Yang, Rui Jiang, Han Wang, and Shuzhi Sam Ge. Road constrained monocular visual localization using gaussian-gaussian cloud model. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3449–3456, 2017.