UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Vladyslav Fediukov

Detection of changes in maps using LiDAR point clouds

Master's Thesis (30 ECTS)

Supervisor: Dmytro Fishman, MSc

Tartu 2020

Detection of changes in maps using LiDAR point clouds

Abstract:

Self-driving cars are one of the most vibrant fields of Robotics and applied Artificial Intelligence. At the current level of the industry development, maps are the main source of the vehicle's knowledge of the surrounding environment. Providing basic knowledge for the motion planning and global navigation, they are an essential part of the self-driving car's autonomy. Therefore, they should be kept up to date. Locations that have changed need to be visited again and remapped. Change detection can be done by comparing two point clouds obtained at the same spatial location, but at different points of time. For the pairwise comparison of the point clouds, first we should select the overlapping road parts, then the alignment is performed and finally the distance metrics are calculated.

The aim of my thesis is to develop a pipeline that detects changes along the route of the autonomous car. The pipeline includes dynamic object filtering, point clouds alignment and evaluation of their difference. The developed pipeline is evaluated on the Oxford RobotCar dataset since it contains different routes through the same places for many month, which can provide a set of significant changes on the road. To our knowledge, there were no previous attempts to create an automated pipeline for the map's change detection with the LiDAR point clouds.

The results show that the constructed pipeline can detect significant changes with the LiDAR input data. The developed pipeline is the first step towards eventually ensuring real-time updates of the map for self-driving vehicles, which will help cars to operate in the city more optimally.

Keywords:
Autonomous vehicles, localization, mapping, point clouds registration, change detection

CERCS: P170 (Computer science, numerical analysis, systems, control)

Kaartide muutuste tuvastamine LiDAR-i punktpilvede abil

Lühikokkuvõte:

Ise-juhtivad autod on üks kõige kiiremini arenevaid valdkondi robootika ja tehisintellekti rakendamisel. Tööstuse praegusel tasemel on kaardid ümbritseva keskkonna teadmiste peamiseks allikaks. Ise-juhtivatele autodele autonoomsuse võimaldamise üheks oluliseks osaks on kaardid, mistõttu on oluline nende hoidmine ajakohasena ja muutunud situatsiooniga kohad tuleb uuesti kaardistada. Muutuste tuvastamiseks saame võrrelda samas asukohas aga erineval ajahetkel kogutud punktipilvi. Punktipilvede võrdlemiseks valitakse alustuseks välja sama asukohta katvad punktipilved, seejärel täpsustatakse nende omavahelist joondamist ja viimasena arvutatakse välja nendevahelised erinevused.

Käesoleva magistritöö eesmärgiks on välja arendada automaatne töövoog, mis tuvastab muutused ise-juhtiva sõiduki teekonnal. See sisaldab liikuvate objektide väljafiltreerimist, punktipilvede joondamist ja muutuste tuvastamist. Väljatöötatud töövoogu on hinnatud Oxford RobotCar andmestikku kasutades, kuna see sisaldab erinevaid teekondi, mis läbivad samu asukohti mitme kuu pikkusel perioodil. See lubab eeldada, et on mitmeid oluliselt muutunud situatsiooniga asukohti. Autorile teadaolevalt varasemad katsed luua lidari punktipilvedest muutuste tuvastamiseks automaatne töövoog puuduvad.

Tulemused näitavad, et lidari andmete põhjal on võimalik loodud töövoogu kasutades olulised muutused tuvastada. See on esimeseks sammuks suuremas protsessis, mis võimaldab kaartide reaalaja uuendamise ise-juhtivatele autodele ning seeläbi nende tunduvalt optimaalsema kasutamise linnas.

Võtmesõnad:

Autonoomsed sõidukid, lokaliseerimine, kaardistamine, punktpilvede registreerimine, muudatuste tuvastamine

CERCS: P170 (Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria))

Contents

1 Introduction

In the recent decade, self-driving cars became one of the most thriving and vibrant topics in applied Robotics and Data Science. More and more autonomous vehicles are becoming involved in our lives and changing them dramatically. A lot of research articles were written about the cars' autonomy, from the motion planning [10] to the effect on our everyday routines and other social questions [8].

Clearly, we should ensure safe and efficient motion algorithms for self-driving cars. Thus, the autonomous vehicle community should pay a lot of attention to the cars' route planning and ability to orientation. The self-driving car uses a map for navigation and planning, so it is important to keep the map up to date. Big cities are very dynamic, road infrastructure changes there each day. This can change the throughput of the road and decrease the efficiency of route planning or even prevent cars from navigating properly and provoke crashes. Therefore, a map's change detection is a very important task for the modern autonomous vehicles industry.

Generally, maps are updated by the rarely reassembling of the information from the frequently traveled routes, which can not provide actual data about a road condition. The main aim of this thesis is to create an automated approach for the map's change detection.

Existing approaches, which are working with the LiDAR data, are not concentrating on the notable changes detection task. They are analyzing the general possibility to detect any changes (e.g. cars or trash cans), by registering two point clouds and taking pairwise distances. This work is dedicated to the specific problem of detecting serious changes in the urban environment, for example roadworks or changes in the road surface.

In the following chapter is an overview of related work. In the chapter 3 provided background information, needed to understand this thesis. Chapter 4 is dedicated to the exact methods description and experiments are described in the section 5. Finally, results and conclusion could be found in the chapters 6 and 7 respectively.

2 Related work

The research about the changes detection using LiDAR point clouds is relatively scarce. Changes detection using registration techniques is studied mostly for image-based analysis [22, 17, 6]. This work attempts to use LiDAR point clouds because they provides the most precise localization of the vehicle on the cities streets. And localization is the first step in the task of the map change detection.

The problem of change detection with the point clouds is also not actively investigated area, with few published papers in the last years [39]. In mentioned paper Dempster-Shafer theory was applied to fuse the data from several point clouds and estimate the occupancy of the grid on the map.

The Dempster-Shafer theory is a generalization of the Bayesian subjective probability. It is a framework for modeling observation error. The Dempster combination rule combines belief functions from different independent sources and sets the belief about the object's existence on the map. Belief function is a generalization of the probability measure and could be interpreted as a degree of belief [31]. The Dempster combination rule is using the sensor confidence and the field-of-view (FoV) of the perception sensors to update the belief of the object's existence on the HD map.

Authors of the previously mentioned paper [39] try to identify any changes in the urban environment, including small objects like trash cans and dynamic objects like cars and cyclists, while this work concentrates on the detection of the significant changes, which possibly affect the road traffic. In the mentioned work authors also compare different distance metrics, like average point to surface distance and Hausdorff distance, therefore it would be a good possibility to compare some of their outcomes with the results of this thesis.

The problem of the map change updates in self-driving vehicles was addressed by the authors of the SLAMCU algorithm [16]. SLAMCU's main function is updating the map while the car drives, meaning the on-line addition of new objects to the map and deleting the objects that no longer exist.

Not all detected changes should be incorporated into the map. Some of these changes could be just sensor errors. That is why we should apply probabilistic reasoning for the determination of the change significance. The SLAMCU algorithm uses the Dempster-Shafer (evidence) theory to set the probability for the object's existence in the High Definition map. This means that based on the sensor's confidence and information, we can speculate about the possible changes on the map if the car encountered some mismatch.

In the SLAMCU paper [16], three possible object types for the map update were proposed: "normal", "delete" and "new". Normal objects, which are detected by sensors and were present in the prebuilt map, are involved in the localization update. This means

that the car position on the map is estimated in relation to them. The objects which were estimated as "deleted" by the Dempster-Shafer theory are not used in the localization process to avoid collisions. To include a new object in the map, a vehicle should precisely localize itself first.

The posterior probability of the map after incorporating new observations is usually estimated with the particle filters. This is a type of algorithm which uses random sampling for the posterior estimation for the case when observations could be noisy or corrupted. A Rao–Blackwellized particle filter (RBPF) was suggested [16] for concurrent localization and mapping. RBPF, in essence, takes only a subset of variables during the sampling and marginalizes all others. This action increases the filter's performance in comparison with other particle filters. By using RBPF together with the Dempster combination rule, the authors achieved good performance in objects' existence estimation using several sensors. This algorithm is robust to the noisy data and can show a good estimation of the epistemic uncertainty.

In this work, the focus will be mostly on the simplified version of the change detection, but not on the map update due to the complexity of the latter. Also, in this work only one sensor input is used, so applying described techniques will be an unnecessary complication. However, any further work on this project will definitely require the incorporation of detected map changes to the pipeline.

In contrast to the standard HD maps [20], the map also could be represented as an occupancy grid with the number of Gaussian mixtures along the z-dimension (height), with Bayesian inference used for estimating the position [37, 19].

3 Background

In this section is presented and overview of the related topics, needed for the thesis understanding. Subsection 3.1 describes different approaches to localization and mapping which are the first step of the process. This is because autonomous vehicle should correctly estimate its own position, to perform correct comparison with the map. It is possible to detect relevant changes on the map by taking the same parts of the road and comparing them. Subsection 3.2 is dedicated to the detection of changes in existing maps and corresponding methods. To ensure a correct comparison of the same spatial location on the different days, dynamic objects (e.g. cars) that can introduce noise should be removed. An approach to filtering out dynamic objects is discussed in Subsection 3.3 alongside the concept of Neural Networks, which are used for the filtering task.

3.1 Localization and mapping

Autonomous vehicles, which localization modules fail may cause unwanted damage, for example by driving on the pedestrian zone. Therefore, determining coordinates of the ego-vehicle in the world coordinate frame (self-localization) is essential for every autonomous system. In order to localize itself on the streets, let alone, safely navigate to the target destination, a self-driving vehicle needs to have a reliable prior belief about its environment, that is why we need a map.

Mapping in autonomous driving is the process of creating the precise digital replica of the vehicle's immediate environment. The most accurate maps are called High Definition (HD) maps. HD maps enable centimeter precision, which allows self-driving cars to navigate safely even in cases when the signal from the Global Navigation Satellite System (GNSS) is either blocked or deemed unreliable. One of the main components of the HD map is the point clouds collected by LiDARs. LiDAR is a spinning device with several laser beams, which scans the surrounding environment. Laser beams reflect from the obstacles around the car. The beam traveling time from the sensor to the object and back defines the distance between this obstacle and a car. As a result, LiDAR produces a set of points in 3D space called point clouds. These points represent obstacles in the neighborhood of the car. Laser beams are physically located at a different height in a LiDAR device. This way, the laser can reach objects from different angles and detect objects from multiple points of view. Each point in such point clouds is described by the 3D coordinates - a location where a laser beam was reflected from the object in the surrounding. The most recent versions of LiDAR include up to 64 rotating beams to enable greater density of the resulting point clouds. Therefore, LiDARs are very effective for building reasonable 3D point clouds and thus a basis for HD maps, that accurately

represent the surrounding environment. In this thesis, we define an HD map as an assembled collection of LiDAR recorded point clouds. HD maps should be kept updated otherwise the autonomous vehicles (AV) will not be able to operate in the respective region. AV may have an inaccurate view of the environment and make wrong, potentially harmful decisions. Thus the main goal of this thesis is to detect changes with respect to already collected HD maps.

Change detection with respect to the prebuilt map will be done by the distance calculation between two point clouds from the same spatial location, where one point cloud represents the prebuilt map and another one reflects the current route. The first step for this process is a coarse localization and the first subsection dedicated to this task.

3.1.1 Global Navigation Satellite System and Inertial Navigation System

In this work, we need localization for determining the point clouds from the same places to compare them and determine the significance of the changes. The standard approach to localization is using Global Navigation Satellite Systems (GNSS), which allows coarse localization with meter precision. GNSS estimates a position using a set of orbital satellites that send and receive the signal from a car and estimate the time spent on the path of the signal from the satellite to the Earth and back. For an accurate localization, at least 4 satellites should have a car in their field of view. Next, the position of a vehicle could be calculated using existing well-established techniques.

Regrettably, satellite signals can be distorted by reflection from tall buildings, concrete walls, or clouds. Therefore the GNSS coordinates could be extremely noisy. To alleviate this problem, a coordinate refinement step is performed using readings from the Inertial Navigation System (INS). The central part of the INS is the Inertial Measurements Unit (IMU), which consists of three orthogonal accelerometers and gyroscopes. These sensors measure acceleration along each axis together with angular velocity, creating a more complete picture of the vehicle dynamics. A few key parameters are required to initialize INS, namely starting position, velocity, altitude and Earth rotation rate. INS helps to keep vehicle positions up to date by continuously incorporating changes in position and altitude that are determined from the orientation and acceleration, measured by the IMU.

In this work, two types of coordinates are used, raw GNSS coordinates and coordinates corrected with the INS data. Location estimates from the GNSS are combined with the IMU measurements through the Kalman filter. It is a computational algorithm that produces a robust track of the vehicle [36]

3.1.2 Point cloud registration

Despite that sophisticated localization methods can provide good precision for vehicles, it is not enough for the correct estimation of the changes with respect to the existing map. Therefore, our aim is to find the exact match between the reference point cloud (which represents the map) and a point cloud from the current route. This can be achieved by a point cloud registration, meaning aligning them in one coordinate system in a way that minimizes the distance between them.

There are two types of registration: rigid and non-rigid. The first one includes only rotations, translations and reflections, while the second one also allows anisotropic scaling and skews, which makes non-rigid registration very sensitive to noise. Only rigid registration should be used with the urban street scans because they represent physical objects (e.g. buildings and roads) that cannot be scaled or skewed for the better match.

However, the exact matching of point clouds is often not possible because of several problems [24]: unknown non-rigid spatial transformation; big dimensionality of the point cloud; presence of outliers, noise and missing points. All these can cause registration algorithms to get stuck in local minima and total disruption of the process. Researchers should dedicate a lot of time to validate that registration is working on this step, visually estimating the quality of the point cloud alignment. This introduces additional complexity in the creation of automated pipelines.

Standard approaches for registration tasks are Iterative Closest Point (ICP) [4] or Coherent Point Drift (CPD) [24]. They will be described in the Methods section. During the experiments, one of the main problems which have occurred is the falling of the registration methods into local minima, which produces wrapped point clouds. This is a known problem, however not solved yet.

3.2 Metrics

After localization and mapping are done, one can proceed to the comparison of the two point clouds. One of the point clouds is a map (prior knowledge) and another one is a representation of the current route of the vehicle (new knowledge). Generally, the significant distance between point clouds would signify the change in the environment. It should be mentioned that the accuracy of the change detection directly depends on the method of distance calculation. There are multiple distance metrics that vary in their complexity and the ability to perceive the nature of objects representation in 3D space. The simplest and well-known metric is Euclidean distance [32]. Euclidean distance between two vector X and Y with length N, defined as:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

where x_i and y_i are elements of the vectors X and Y respectively.

The most popular approach used in the majority of all point cloud processing tools consists of two steps. First, the reference point cloud is transformed into the mesh (surface) using the interpolation of the nearest point and its neighbors. Second, the distances between the points of the second point cloud and the reference mesh are calculated and then averaged. In other words, the normal vectors to the mesh from each point are averaged. Another conventional method is the Hausdorff distance [7]. Hausdorff distance [15] between two vectors X and Y is defined as:

$$d_H(X,Y) = max(sup_{x \in X} inf_{y \in Y} d(x,y), sup_{y \in Y} inf_{x \in X} d(x,y))$$

In this work, the Hausdorff distance and average of normal vectors from points to the mesh will be computed and compared.

ROC curve is used to estimate the performance of the model and chose the threshold. It builds the curve with the False positive rate (share of False positives out of all negatives) on the X-axis and True positive rate (share of True positives out of all positives) on the Y-axis. Based on the cost of each metric, the best threshold is chosen.

3.3 Dynamic objects filtering

For correct and adequate comparison of the maps, dynamic objects should be removed from the point clouds, because these objects are not taken into account during the map building. One of the most successful object detection methods is neural networks. They will be used for dynamic objects filtering in this work.

A neural network is a mathematical model for function approximation. It consists of neurons, an activation function and weighted connections between these neurons. A neuron receives input information, applies some function to it and gives this output to the neurons in the next layer. Neurons are also called perceptrons. Usually, they operate with nonlinear functions, because neural networks mainly aim to approximate nonlinear dependencies. Example of the perceptron is depicted on the Figure 1.

Convolutional Neural Network (CNN) is a neural network architecture for processing data with a known grid-like topology, like images. An example of this Neural network could be seen in Figure 2. They use convolution instead of general matrix multiplication in their layers. Convolution could be seen as a weighted average of the function $f(t)$ over time t. Let $w(t-a)$ be the weight function, where a is the age of measurement. Usually, we would like to give recent results the bigger weight and decrease the weight

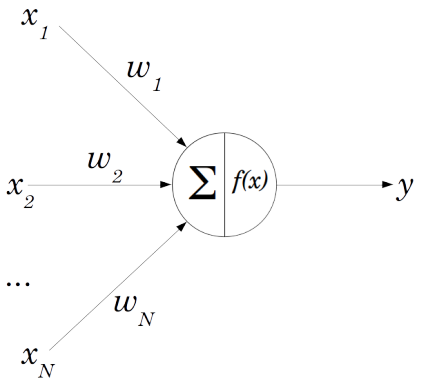

Figure 1. Visualization of the perceptron. A neuron takes a linear combination of inputs and applies a nonlinear function to it. [1]

as the measurement age increases. Rewriting this more formally, convolution operator is defined in the following way:

$$s(t) = \int x(a)w(t-a)da$$

The arguments x and w are often referred to as input and kernel, respectfully. Convolution is defined for each function, for which the above integral is defined.

Many Convolutional Neural networks have a specific layer, called Max pooling, which is depicted on Figure 3. This is a specific layer, used for feature extraction and down-sampling. This layer extracts maximal values from arbitrarily determined regions and forms a down-sampled version on the input.

An encoder is a type of neural network, which aims to learn an encoding or representation of the data. It takes a dataset as input and outputs learned features. This technique is used for dimensionality reduction and extraction of the most important information from the input data [12]. A good example of an encoder could be a neural network, which takes images as input and outputs the learned heatmap of the features. This heatmap highlights specific corners and lines which characterize learned objects. A decoder is a neural network that takes learned features as an input and applies these features onto initial images to classify or segment depicted objects. On the Figure 4 example of such kind of neural networks could be seen.

To build a consistent and informative map, one should ensure that all dynamic objects are filtered out from the input data. Generally, by dynamic objects, practitioners understand other vehicles, pedestrians and cyclists. Due to the specifics of the available pretrained neural network used in this work, we will consider only one type of dynamic object - other vehicles. A common practice for similar tasks is to use a neural network that will detect dynamic objects from their input [27, 26]. Unfortunately, object detection

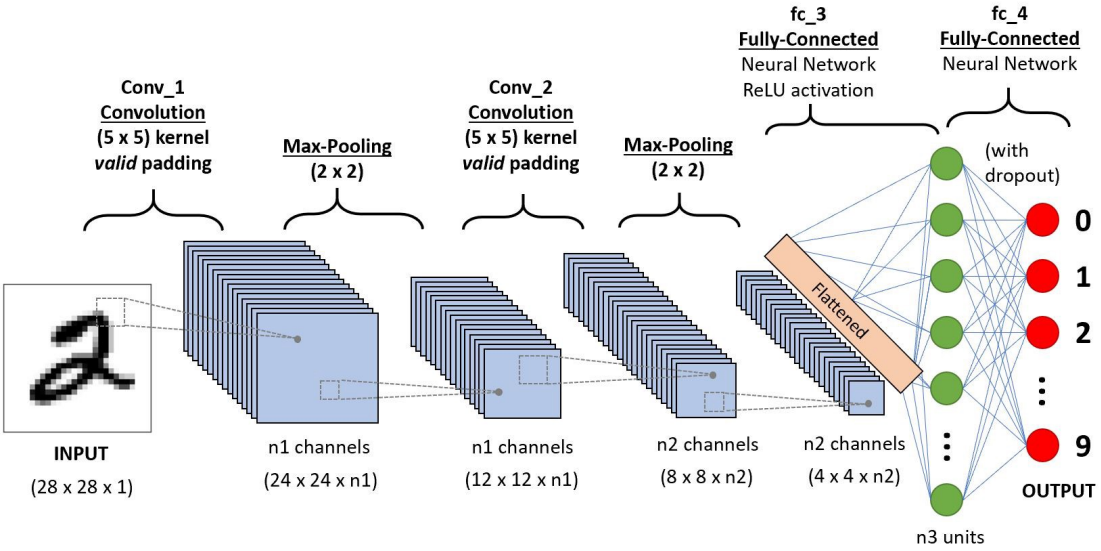

Figure 2. Visualization of the convolution neural network. [2]

in the point clouds could not be done in the same way as in the images due to two key differences: an image is 2D and dense, while point cloud is 3D and sparse. Point clouds are characterized by the absence of order in the points, connections between the points and invariance to certain transformations. From the data structures point of view, point cloud represents an unstructured set of points, unlike the conventional data structures for neural network input. All these reasons require a separate set of methods to handle point cloud data. Point cloud features (objects shapes, objects intensities) have certain statistical properties, for example, they are often invariant to transformations, such as rotation and translation, since they do not change segmentation of the point cloud. These features are classified into two types: intrinsic (local) [1] and extrinsic (global) [29]. Global features are contours, shapes and lines, which represent an image as a whole, while local features represent parts of the image that contain specific objects. Global features are used in tasks like image classification and local features are primarily used for more high-level tasks, like object recognition. Deep learning has been shown to work well on 3D shapes [38]. Another approach is using the Multiview CNNs: they rendered 3D point clouds into 2D images that were further analyzed using convolution neural networks. This architecture is depicted on Figure 5. However, this method requires a lot of data preprocessing and does not show good performance with the point clouds. This approach enabled robust shape detection, while scene understanding was out of reach. Other proposed methods - Spectral CNN and Feature-based DNN, were shown to be limited by the representation power of the feature extractors [38].

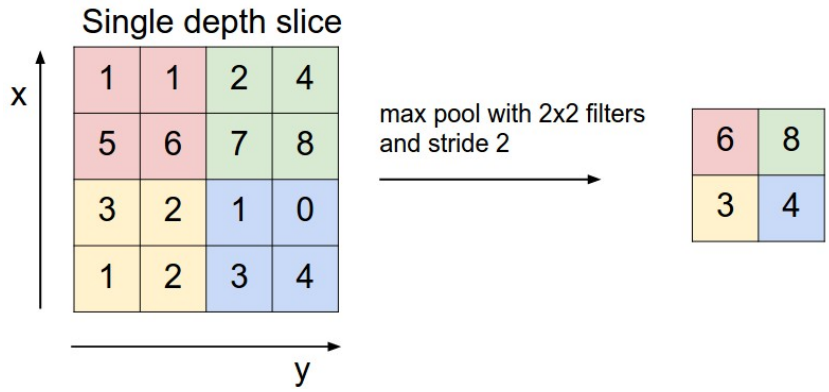

Figure 3. Max pooling. From each region, in the input, only a maximal value is chosen and transferred to the output. [3].

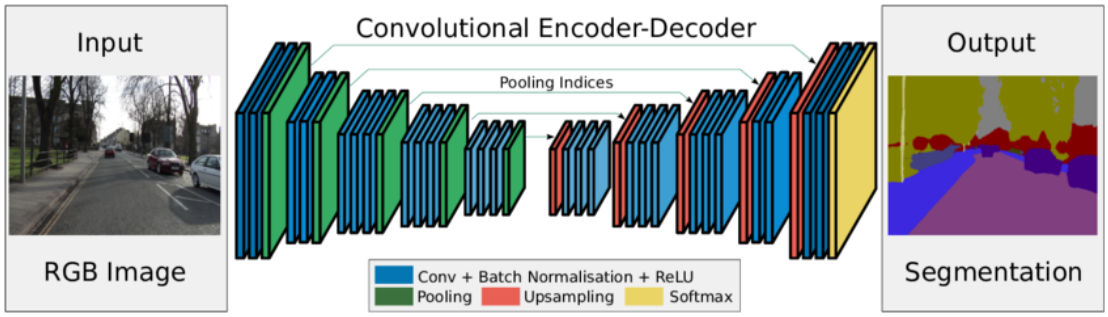

Figure 4. Visualization of Segnet Encoder-Decoder architecture. [2]

4 Methods

The main tool for precise localization is a point cloud registration. In subsections 4.1 and 4.2 an overview of the two most widespread methods for the point cloud registration will be given. Part 4.3 will be dedicated to the Point Pillars neural network, which is the method for removing dynamic objects (e.g. cars) from the point cloud. These objects should not be accounted for because they are not a permanent part of the environment and they introduce noise to the point clouds registration and comparison.

4.1 Iterative Closest Point

The most popular and one of the oldest algorithms for the point clouds registration is Iterative Closest Point (ICP) algorithm. The main idea of this algorithm is to find a

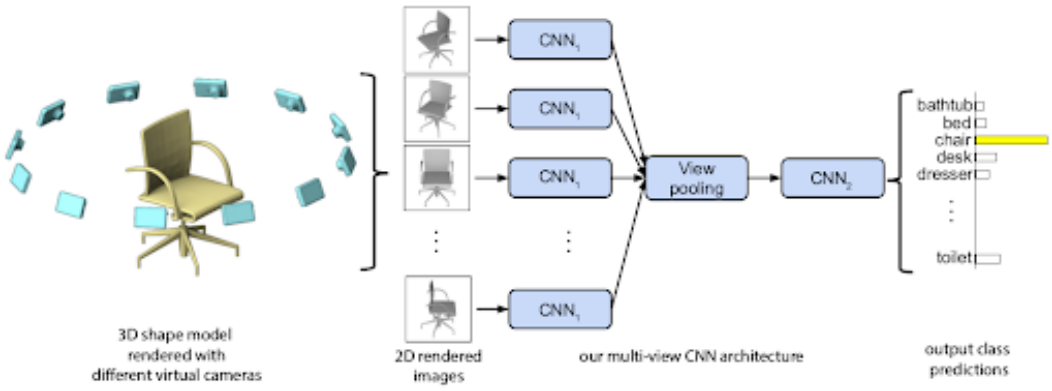

Figure 5. Visualization of the Multiview CNN [33]

point from another geometrical entity that is the closest to the given point. This method minimizes the MSE of the distance between a given point and all points of another geometrical entity over 6 degrees of freedom. ICP always monotonically converges to the nearest local minimum. Therefore, good initial rotation and translation matrices are critical for convergence to the global minima. Still, they are hard to obtain in the automated pipeline because the objective function is high-dimensional and not convex [4]. To overcome the aforementioned problem, one can do rough initial registration using, for example, the Random sample consensus (RANSAC) algorithm. It provides an approximate estimation that could be further refined with the ICP. The RANSAC algorithm randomly samples a small number of points from the observed data and tries to estimate the parameters of a model, which describe the desired points distribution. Next, the algorithm calculates a share of all points from the initial dataset which fits into this model with a predefined permissible error. RANSAC stops when the predefined share of points is fully described by the model. Until then, the algorithm proceeds with the next iteration, where a new sample set is taken and the new model is created. [9]

The registration task in ICP could be represented as a transformation of a point set P for the best alignment with a model X. Let $y \in X$ be the closest point for P and Y set of closest points. The distance between the individual point p and model X is defined as follows:

$$d(\overline{\mathrm{p}}, X) = min_{\overline{\mathrm{x}} \in X} ||\overline{\mathrm{p}} - \overline{\mathrm{x}}||$$

Once a set of closest points Y was obtained for the point set P, a registration transformation, denoted as $Q(P, Y)$, could be computed using least-squares. This transformation consists of the rotation and translation vectors (p and q). To obtain the desired transformation, for each point in P, the former vector is multiplied on the given point and the latter is then added.

Rotations could be described as a vector of quaternions. Quaternion is the quotient

(quantity produced by the division of the two numbers) of two directed lines in a three-dimensional space or equivalently is the quotient of two vectors. Quaternions are usually represented as $a+bi+cj+dk$, where a, b, c, d are real numbers and i, j, k are fundamental quaternion units. [13]

Taking the unit axes X, Y, Z and θ as rotation angle in radians, respective quaternion will be:

$$\cos\frac{\theta}{2} + iX\sin\frac{\theta}{2} + jY\sin\frac{\theta}{2} + kZ\sin\frac{\theta}{2}$$

Given the initial point set P with N points and reference point set X, the ICP algorithm is [4]:

1. set $P_0 = P$ and initialize the transformation quaternion $q = (1, 0, 0, 0, 0, 0, 0)$
2. Compute the closest points $Y_k = C(P_k, X)$ compute the best registration between the P_k and Y_k: $q = (q_k, d_k) = Q(P_0, Y_k)$ by minimization of the root means square point to point distance, where q_k is rotation part of quaternion and d_k - translation part of the quaternion.
3. Apply the registration: $P_{k+1} = q(P_0)$
4. compute the mean square error between the P_{k+1} and Y_k: $d_k = \frac{1}{N_k}\sum_{i=1}^{N_p} ||\overline{\mathrm{y_{i,k}}} - \overline{\mathrm{p_{i,k+1}}}||^2$
5. Stop the loop if the change $d_k - d_{k+1}$ is smaller than a threshold $\tau > 0$

4.2 Coherent Point Drift

The second most popular method is Coherent Point drift (CPD) [5]. It is a probabilistic algorithm for the pairwise point clouds registration. In this method the first point cloud is represented as a Gaussian Mixture Model (GMM) and the second point cloud as a reference. GMM is a probabilistic method that assumes the presence of several normally distributed sub-populations inside one hybrid population. For the CPD, the points of the first point cloud are considered as the Gaussian Mixture Model centroids. The alignment of two point clouds is viewed as a problem of probability density estimation since the method aims to fit the GMM centroids to the reference point cloud via the likelihood maximization.

The desired registration transformation should preserve the initial topological structure of the point cloud, meaning all its points are moved together coherently. The most optimal transformation that converts the Gaussian Mixture Model into a fit for the second point cloud is found by the maximum likelihood estimation (MLE) method. It is a technique for parameters estimation in a given distribution. In the CPD method, the estimated

parameters are the centroids of the GMM and the matrix of covariances between them. With Θ as estimated parameters of distribution and $P(\Theta|X)$ as the likelihood of these parameters given data X, the likelihood function is defined as follows:

$$F(\Theta) = P(\Theta|X)$$

In the case of rigid registration, coherency is imposed by reparametrization of GMM centroid locations with rigid parameters and the closed-form solution can be derived.

We can rewrite the registration problem for CPD more formally. Let D denote the dimension of the point sets, N and M indicate the number of points in two point sets, Y and X represent the one point cloud as GMM and second as reference. Then, the GMM probability density is:

$$p(x) = \sum_{m=1}^{M+1} p(m)p(x|m)$$

And probability of the generated points given centroids m:

$$p(x|m) = \frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}} \exp -\frac{||x - y_m||^2}{2\sigma^2}$$

The authors also added uniformly distributed noise to the mixture model: $p(x|M + 1) = \frac{1}{N}$ to account for the noise and outliers.

After setting equal membership probabilities $p(m) = \frac{1}{M}$ for all GMM components, the mixture model could be rewritten as:

$$p(x) = \omega\frac{1}{N} + (1 - \omega)\sum_{m=1}^{M+1} p(x|m),$$

where ω is the weight of the noise that is set manually.

GMM centroids locations are reparameterized using a set of parameters Θ. The Expectation-maximization (EM) algorithm is used to find parameters Θ and σ^2 from the log-likelihood model:

$$E(\Theta, \sigma^2) = -\sum_{n=1}^{N} \log \sum_{m=1}^{M+1} p(m)p(x|m)$$

The expectation-maximization algorithm is used for parameters estimation in statistical models that depend on the unobserved latent variables [35]. The algorithm consists of two steps. At the expectation step, the expectation of the logarithm of the likelihood is evaluated. This likelihood is calculated based on the estimation of the current parameters and maximized at the maximization step.

The correspondence probability between y_m and x_n is defined as:

$$p(m|x_n) = \frac{p(m)p(x_n|m)}{p(x_n)}$$

In the case of rigid registration, the transformation of GMM centroids locations is:

$$\tau(y_m; R, t, s) = sRy_m + t,$$

where R is $D \times D$ rotation matrix, t is $D \times 1$ translation vector and s is scaling. The objective function for Expectation-Maximization algorithm, in this case, is:

$$Q(R, t, s, \sigma^2) = \frac{1}{2\sigma^2} \sum_{m=1}^{M+1} P^{old}(m|x_n)||x_n - sRy_m - t||^2 + \frac{N_p D}{2} \log\sigma^2,$$

subject to $R^t R = I, det(R) = 1$.

In the case of affine transformation (non-rigid registration), the objective function will be unconstrained:

$$Q(B, s, \sigma^2) = \frac{1}{2\sigma^2} \sum_{m,n=1}^{M,N} P^{old}(m|x_n)||x_n - (By_m - t)||^2 + \frac{N_p D}{2} \log\sigma^2,$$

where B is a $D \times D$ affine transformation matrix and t is a $D \times 1$ translation vector. Generally, non-rigid registration is more challenging because it represents a broad class of possible transformations. This may lead to an ill-posed problem since the existence of a solution is not guaranteed in case of the infinite dimensionality of the transformation space [30]. To avoid these ill-posed situations, a Tikhonov regularization framework is added to the objective function [34].

Regularization $\phi(v)$ is included in the objective function. Then it becomes:

$$f(v, \sigma^2) = E(v, \sigma^2) + \frac{\lambda}{2}\phi(v)$$

CPD algorithm includes three hyperparameters: ω, λ, β. $\omega (0 < \omega < 1)$ is an estimation of the noise ratio in the point cloud, λ and β are the smoothness of regularization. β defines the width of smoothing Gaussian filter

$$G(s) = \frac{1}{\sigma\sqrt{2\pi}} \exp -\frac{1}{2}\frac{s^2}{\beta^2}$$

in regularization term:

$$\phi(v) = \int_{R^D} \frac{|v(s)|^2}{G(s)} ds,$$

and λ represents the trade-off between regularization and the goodness of fit for maximum likelihood [24].

There is no answer to the question of which algorithm is better, with different authors reporting different results [40, 5]. Though, CPD has high memory complexity, which can lead to problems while working with dense point clouds with more than 100 000 points. Because of the need to store the posterior probabilities matrix, the memory complexity becomes $O(M \times N)$, where M and N are the numbers of points in two clouds. With each point cloud having approximately 50 000 points, it took up to 300 GB of memory and 1 hour to compute the registration. A common method to overcome this limitation is to use the undersampling in each separate voxel.

4.3 Point Pillars Neural Network

To achieve successful registration, all noise, like dynamic objects, should be removed. One of the most popular and efficient methods for dynamic object detection is a Point Pillars neural network [18]. It is integrated as a package into Autoware software, which is used for this work, and pretrained weights for the network are in the open access. Therefore, it is very convenient to use this method. Point Pillars utilizes PointNet, a specific architecture of a neural network to learn a representation of point clouds organized in vertical columns (pillars) [28]. Point Pillars encodes 3D point clouds in the way which helps to detect and classify objects in the point cloud. The architecture of the Point pillars neural network is depicted on the Figure 6.

PointNet learns a representation of the input point cloud by estimating the set of keypoints, which roughly corresponds to the edges of the target object which we aim to segment or classify. To select informative points in point clouds, authors suggest using max pooling as a function for global feature extraction. Unlike the structured 2D array that defines a picture, a point cloud is an unstructured set of points. This fact poses a problem for convolution since it assumes that two neighboring points in the array represent two neighboring pixels in the picture. To make it work with the unstructured point set, researchers discretize it into several projections and apply convolutions on them. This process should be permutation invariant, that is why simple symmetric activation functions are used. Authors state that Max pooling works better than other symmetric activation functions, like Average pooling. The last step of the PointNet is a fully-connected layer that aggregates features generated by Max pooling to create one global shape descriptor [28].

The main idea behind this network is converting the initial point cloud into a stacked row- and column-pillars for further feature extraction. These features could be transformed into the 2D pseudo-image, where the backbone CNN could be applied. Features from the backbone are used to predict the 3D bounding boxes for objects in the point

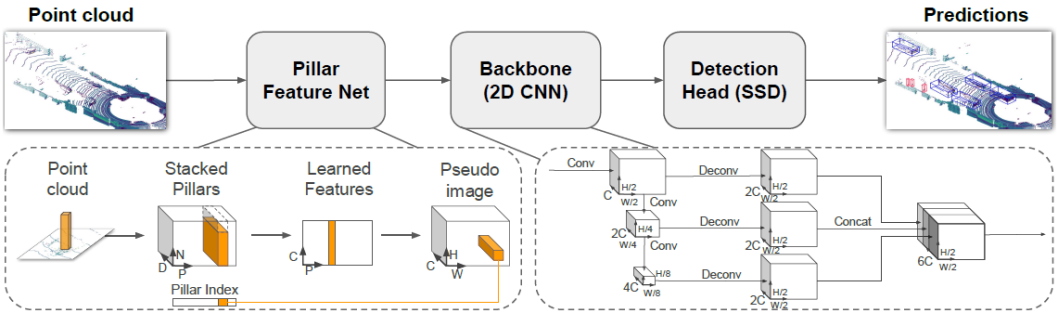

Figure 6. Structure of the Point Pillars Neural network [28].

cloud.

The first step of the Point Pillars algorithm is creating a pseudo-image from the point cloud. To do that, an initial point cloud is split in the pillars, where each pillar is very sparse or even empty. There is a trade-off between speed and accuracy: the thinner the pillars are, the finer the feature detection is, but it requires more time. The bigger pillars work faster with lower accuracy due to smaller pseudo images and the inability to capture some small features. The second step is the Backbone which has two subnetworks. The first one is top-down which extracts features at the very small spatial resolution, and the second one is down-top which concatenates and upsamples features from top-down subnetwork. For Head Detection authors have used the Single Shot Detector [21], which uses a variety of different possible boxes on the (pseudo)image and tries to find the one with the best intersection over the union. This architecture shows the best performance among LiDAR-based object detection methods on the KITTI dataset benchmarks [18]. PASCAL criteria was calculated as a match of the ground truth bounding box and predicted one with at least a 70% overlap. Pretrained weights for Point Pillars show 74.31% on the KITTI dataset [4].

4.4 Dataset

In this work, we used the Oxford RobotCar dataset [23]. This is the only dataset that consists of the big amount of routes through the same places collected during one year and in comparison with the other datasets, its raw input data is available alongside the pre-processed information.

Having these multiple routes was important to find the number of sufficient changes, which could be detected in the same locations.

The dataset was released in September 2016 and was collected from May 2014 until November 2015 in Oxford, United Kingdom. All completed routes are depicted in Figure

[4]http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d

7. This dataset is distributed under the CC BY-NC-SA 4.0 license. Up to 20 million video frames were collected by 4 cameras (one trinocular) and point cloud collected by 3 LiDARs LD-MRS. LiDAR produces almost the same amount of the scans, like video frames, which could be assembled in an arbitrary number of the point cloud. An example of such a point cloud is depicted in Figure 8. The corresponding coordinates were collected with the NovAtel SPAN-CPT, which is a GNSS-INS receiver. Each route in the dataset contains tags that provide meta information about the route, e.g. the quality of the GPS signal, weather conditions or the presence of roadworks.

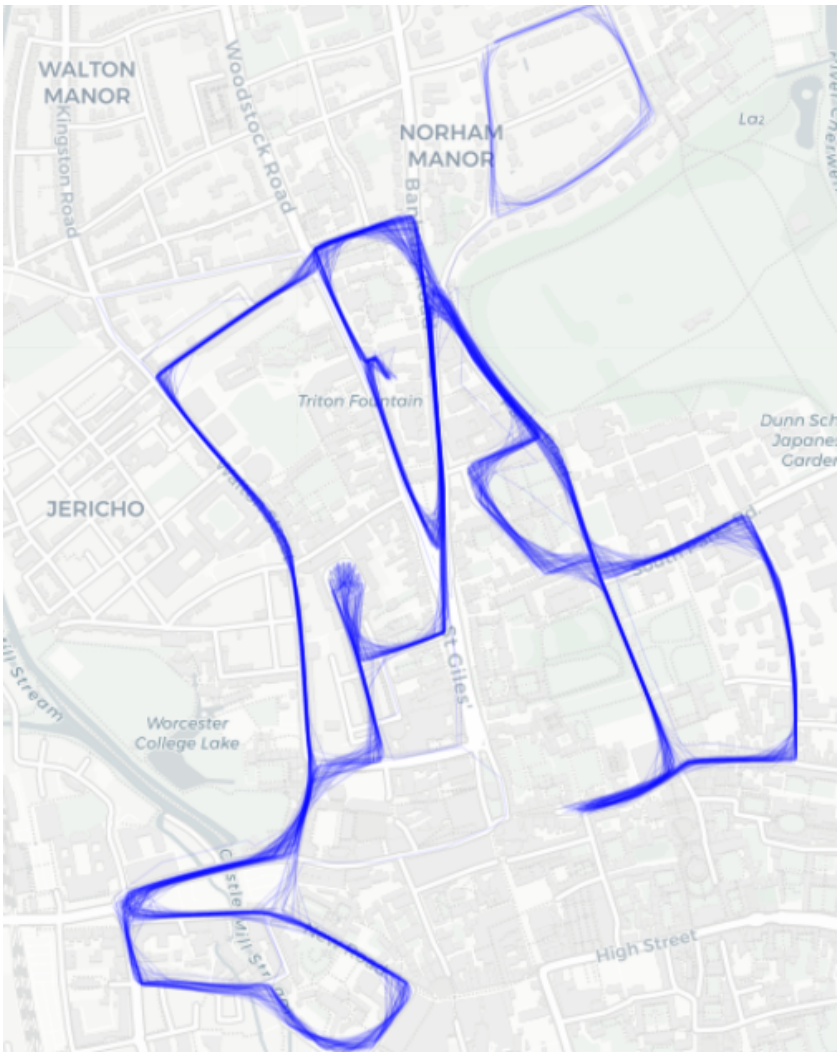

Figure 7. All rides from the Oxford RobotCar dataset after filtering routes with corrupted GNSS signal.

Although self-driving cars researchers are mainly using the KITTI dataset [11] for

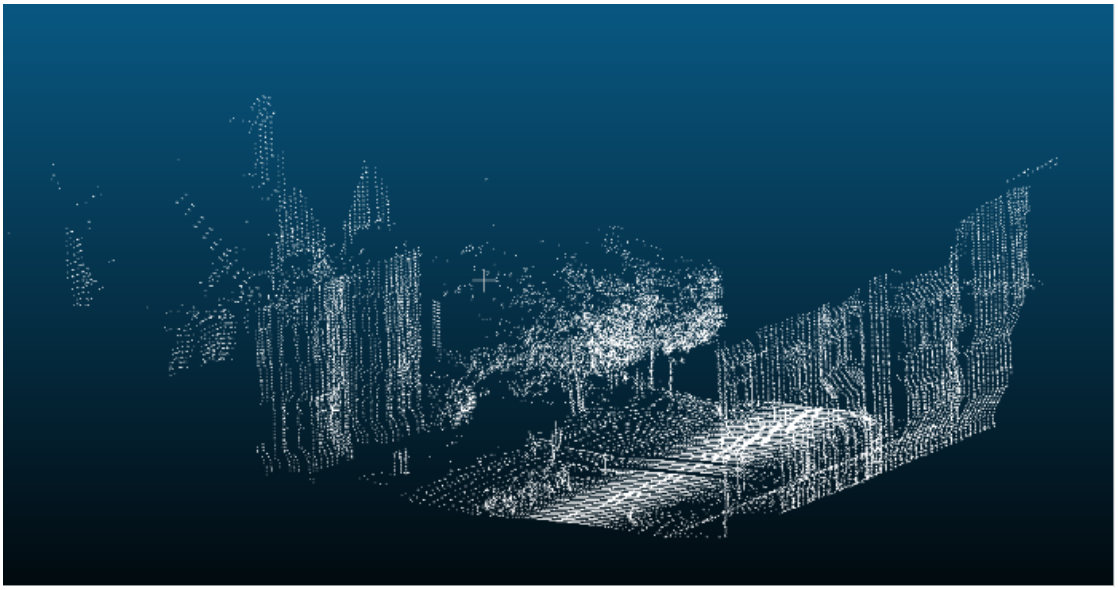

Figure 8. Example of the arbitrarily constructed point cloud from the dataset.

their experiments, due to the specificity of the task an Oxford dataset was preferred. This dataset provides data for the same places for one year, so it is possible to spot out some significant long-term changes which should be updated on the map. It was relatively easy to find one route with and one route without such significant change (like roadworks) owing to the provided tags for each route. Experiments with this unconventional dataset could also be beneficial since quite a lot of the newly developed methods are overfitted on the KITTI dataset and show moderate performance in real life. As an additional feature of this work, it can be estimated if the Point Pillars pre-trained weights can be used for the RobotCar dataset.

5 Experiments

The first step of detecting changes between two routes requires the selection of corresponding point clouds to compare. Using GNSS coordinates of the first route we localize the car in the environment and select the road segment with the same location from the second route.

We empirically found out that the optimal length of the road segment that enables the detection of significant changes is about 10 to 20 meters. This also creates point clouds of reasonable size, up to 150 000 points. Thus, the first route is split into 20 m chunks, each determined by the start and end coordinates. Next, we look up the timestamps of both routes when the car was driving at the given segment and select the corresponding point clouds using Oxford Tools [5]. It finds LiDAR scans for corresponding timestamps and combines them into point clouds, one for each road chunk.

This procedure resulted in 438 road chunks for each ride. Now labels for each road chunk should be determined since the locations with changes are not specified. Best we can do is to look up the labels connected to the entire route and take two trips that follow the same route, where one has a label "roadworks" and another does not. To determine the exact locations of the road changes, images from each route were viewed manually and corresponding road chunks were labeled. Only four roadworks were on the streets, which results into 7 road chunks (one roadwork can be detected on several chunks) labeled positively. Therefore, the obtained dataset is extremely imbalanced and this fact could complicate further analysis.

The second step is removing dynamic objects from the point clouds (e.g. cars). Dynamic objects are removed using the Point Pillars neural network. In this work, its implementation (lidar_point_pillars package [6]) in Autoware was used.

The Point Pillars package is implemented in the ROS system, which has different coordinate systems for frames of car and LiDAR. Origins of their coordinate systems differ because LiDAR is often mounted on the car's roof. This transformations between the coordinate systems are described in the tf transformation tree. Therefore, the package requires additional initialization of the tf parameters, which are responsible for the transition of the coordinates from the LiDAR coordinate system to the vehicle's one. Users should find these parameters on their own, which can take a long time. Incorrect parameters for the tf transformation can lead to meaningless results, e.g. the predicted bounding box will not correspond to the actual dynamic object in the point cloud. In this thesis, the selection of the tf transformation was done by the grid search in the set of possible rotation and translation parameters.

To work with the lidar_point_pillars package, one should write subscriber and pub-

[5] https://github.com/ori-mrg/robotcar-dataset-sdk

[6] https://gitlab.com/autowarefoundation/autoware.ai/core_perception/tree/master/lidar_point_pillars

lisher scripts, because ROS packages work as separate services, which can receive and transfer messages. Publisher script should transform the standard NumPy [25] array that represents the point cloud to the PointCloud2 message. This was done via ros_numpy package functions [7]. The transformed point clouds were sent to the Point Pillars neural network. During its launch, the user should set parameters like neural network pre-trained weights and a launch file. In the launch file, the tf transformation should be indicated. For this work, the weights pretrained on the KITTI dataset [8] were taken. The lidar_point_pillars package is built as a service, which receives messages with the point clouds and sends with the detected objects.

The last step that needs to be done before the registration of point clouds is their voxel-grid downsampling. On average, a point cloud that represents approx. 20 meters of the road has up to 150 000 points and this amount of points is seemingly redundant and computationally prohibitive. For the downsampling, the corresponding space of the point cloud is split into voxels of small size (usually with edges less than a meter) and all points inside the voxel are averaged. The downsampling is necessary mainly due to memory issues. During the registration process, temporary matrices of the point-to-point transformation sometimes could exceed several hundreds of gigabytes (with existing python libraries). In this way, the initial distribution of the points is approximated with some geometry information loss [14]. However, the empirical evaluation of the results suggests that the small size of voxels (like the one used in this thesis) can provide a reasonably good representation of the initial surface structure. In this work, the downsampling was done with the Open3D python library [9], where the user should determine the size of the voxels. Downsampling inside the voxels could also be used as a noise filter.

Next, the registration of point clouds was done with the Iterative Closest Points and Coherent Points Drift algorithms implemented in python packages: Open3D library for the ICP registration and pycpd [10] for the CPD registration. The registration was run on the 2 Intel(R) Xeon(R) CPU E5-2660 processors with 300 GB of memory on 1 node, however, even this amount of memory was not enough. Hence, the CPD implementation in python definitely requires voxel-grid downsampling. The ICP algorithm requires much less memory to compute, but the downsampling of point clouds was also beneficial in terms of the time spent on registration.

After the registration, the non-overlapping parts of the two point clouds have been cut off using the functionality provided in the Open3D package. This is important because big non-overlapping parts could cause distortions in the distance metrics between the point clouds.

Given the registered pairs of point clouds, a distance between them can be computed.

[7] https://github.com/eric-wieser/ros_numpy

[8] https://github.com/k0suke-murakami/kitti_pretrained_point_pillars

[9] http://www.open3d.org/

[10] https://github.com/siavashk/pycpd

For each pair of point clouds registered with two algorithms both Hausdorff and averaged point-to-surface distance metrics were calculated, which indicates the difference in these point clouds.

The final step is choosing the threshold which will separate the significant and insignificant changes. The threshold is chosen using the ROC curves. The schematic representation of the pipeline is depicted on the Figure 9.

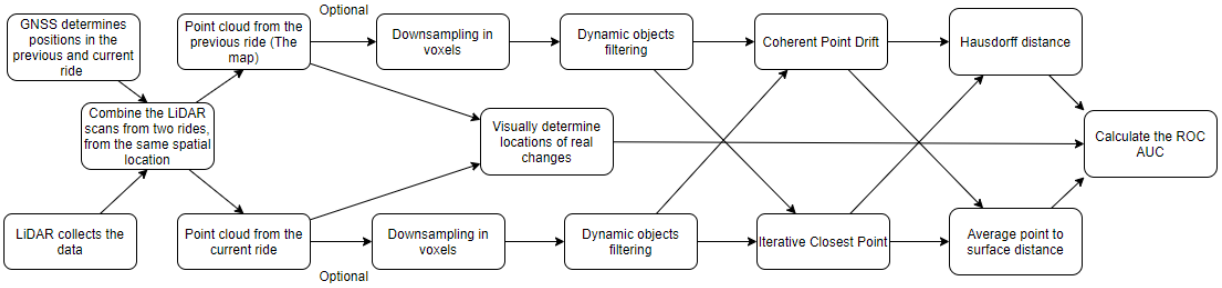

Figure 9. The scheme of the pipeline.

6 Results

6.1 Dynamic objects detection

According to Section 4.3, we used a Point Pillars network with weights pre-trained on the KITTI dataset to identify dynamic objects in the point clouds and remove them from the data. One can see examples of the Point Pillars network results on the Oxford RobotCar dataset on Figures 10, 11. The latter clearly show the successful detection of a car in point clouds. Due to no labels and bounding boxes in the Oxford dataset, the performance was tested visually on 20 objects and 16 of them were detected correctly. Taking the 95% confidence interval, the PASCAL criteria on the pretrained weights is from 62.7% to 97.5%, with an average of 80%. At the current stage of the project, this performance was considered acceptable.

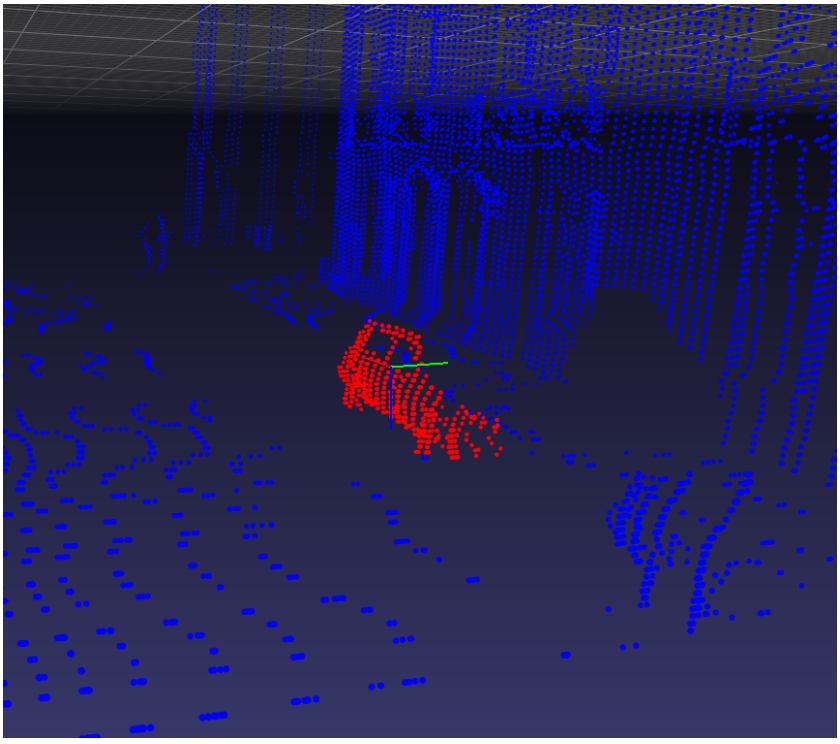

Figure 10. Result of the dynamic object detection with Point Pillars applied to the Oxford RobotCar Dataset. The red dots indicate the points predicted as belonging to the dynamic objects.

Dynamic objects were detected in 108 out of 876 point clouds (438 point clouds from two rides) and on average they contain only 0.7% of the points in the point cloud.

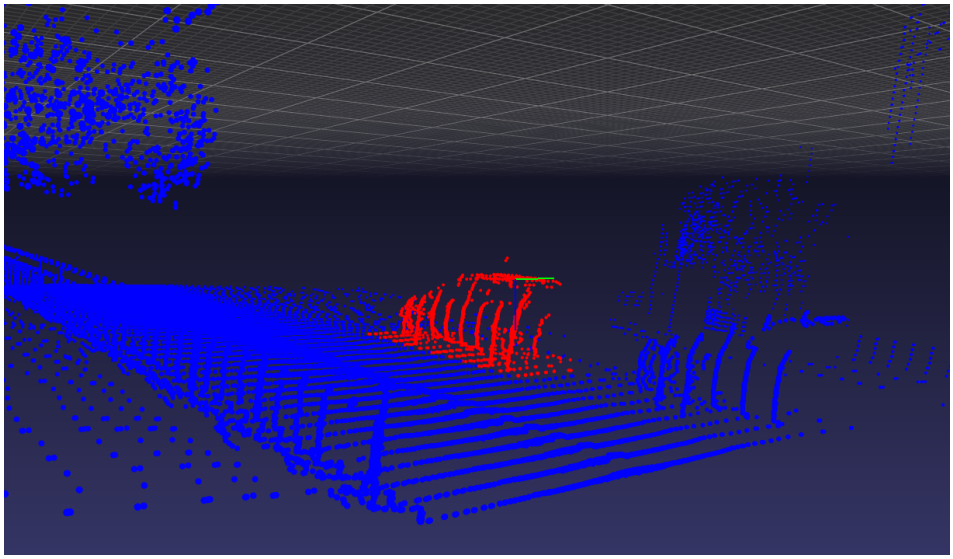

Figure 11. Sometimes the predicted bounding box captures nearby points from the roads or other objects.

6.2 Changes detection

In this work, the four real changes of the road related to the roadworks will be considered as the ground truth positive labels. They are depicted in Figures 22, 23, 24, 25. Change is determined as a visible roadwork construction (e.g. fences) on or near the road, which has appeared during the current path (first point cloud) and was not present on the map (second point cloud); or it was present on the map, but disappeared during the current ride.

In total, there are 8 approaches on how to calculate the resulting difference: ICP or CPD registration techniques, Average point-to-surface or Hausdorff distance metrics, with or without dynamic objects filtering.

Each approach takes point clouds from the same locations, filters or not the dynamic objects, registers them with two registration techniques and calculates two distance metrics. The significance of the changes is determined via the distance score. A researcher is responsible for choosing the threshold to determine which distance indicates a significant change. This was done by calculating ROC AUC scores and assessment of the results visually. The threshold should be relatively high to detect a significant difference between the point clouds, but on the other hand, a too high score would indicate point clouds warped by the registration process. That is why there should be not only a lower threshold but also an upper one.

Locations of the real changes depicted in Figure 12 as red circles and numbered

accordingly.

Figures 13, 14, 15 present results from the comparison of the two routes, when CPD registration with point-to-surface distance was used and dynamic objects were filtered. The color indicates the rate of the change between two routes. From green (no change or insignificant changes) to red (significant change). The black color indicates places where registration has failed and two point clouds are not aligned well enough to compare them. This happens when the optimization of the objective functions for the registration processes is falling into the local minimum.

On the screenshot from CloudCompare [11], Figure 16, we can clearly see the changes on the road from the roadwork constructions. LiDAR can easily detect solid fences, therefore the blockage of the half of the road by the fences will be easily detected. In contrast, small fences with holes could be less visible.

ROC AUC was used for the comparison. On the Figure 17 depicted all methods without dynamic objects filtration. Dynamic object filtration affect the methods' performance very insignificantly for the Coherent Point Drift method. This is clearly seen on the Figures 18 and 19, where ROC curves for the CPD registration are compared with respect to the dynamic objects filtration. But for the Iterative closest point algorithm, objects filtering gives even worse results. Comparison is depicted on the Figures 20, 21.

From the Figure 17 we can conclude that unlike other researches [39], the best performance was shown by the CPD registration with the averaged point to surface distance, not by the Hausdorff distance. The reasons behind the worse performance of other methods may be that Hausdorff distance is very sensitive to the outliers in the point clouds and ICP registration much more often falls in the local minima and is unable to find optimal alignment.

Median distances for each approach was built, to investigate the effect of the objects filtration on the methods' performance. Dynamic objects filtering does not affect the distance between the point clouds with significant difference or with registration failures, but these cases are outliers. Maybe object filtering can decrease the distance in the already close point clouds, where dynamic objects should be theoretically the main source of differences. The median distances for all methods could be seen in the Table 1. As could be seen from this table, dynamic objects filtering introduces no or little effect on the distances.

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

Table 1. Median distance for each possible approach.

Registration method	Hausdorff distance	Average point to surface distance
ICP without filtering	6.363	0.177
ICP with filtering	6.379	0.173
CPD without filtering	6.533	0.404
CPD with filtering	6.399	0.411

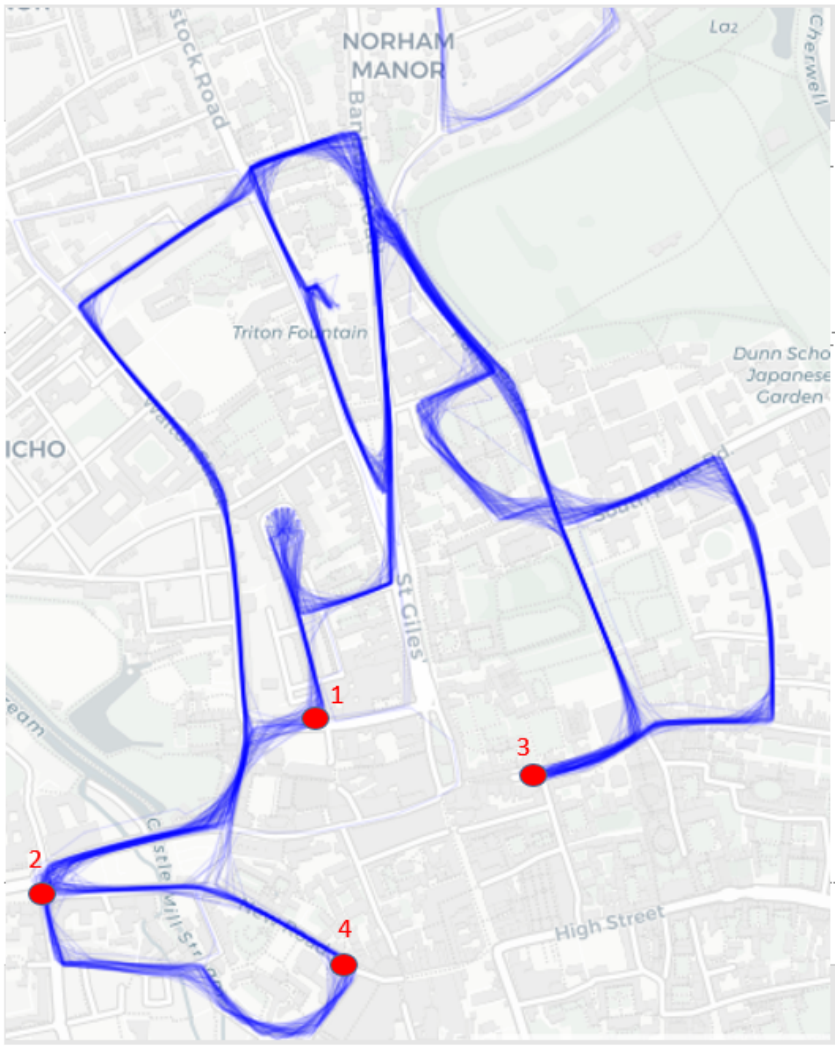

Figure 12. Locations of the 4 real changes on the map of all taken routes.

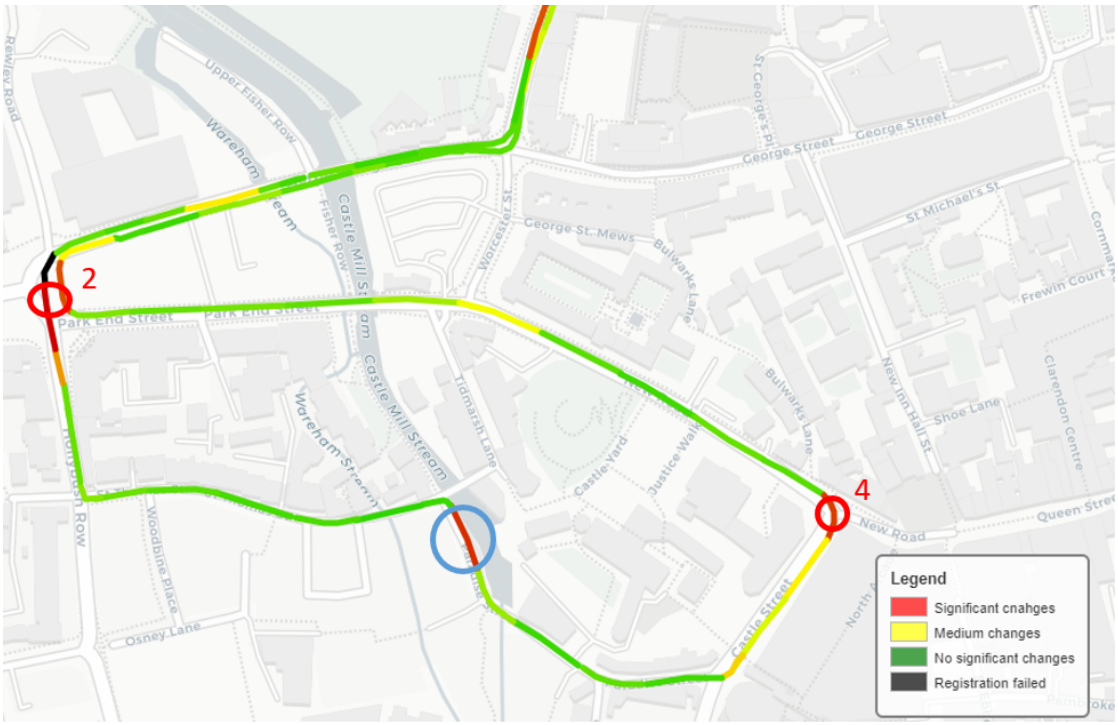

Figure 13. The figure shows distance scores for the road segment with places 2 and 4. Both places have a red circle and red color which means significant change. Chunk with red color, which is not the true change, indicate with the blue circle as False positive result.

Figure 14. The figure shows the result of the algorithm on a route near place 1. Apparently, changes are not significant enough to be considered by the pipeline as relevant. Therefore, it is considered as a False Negative.

Figure 15. Visualization shows place 3 which has a high score.

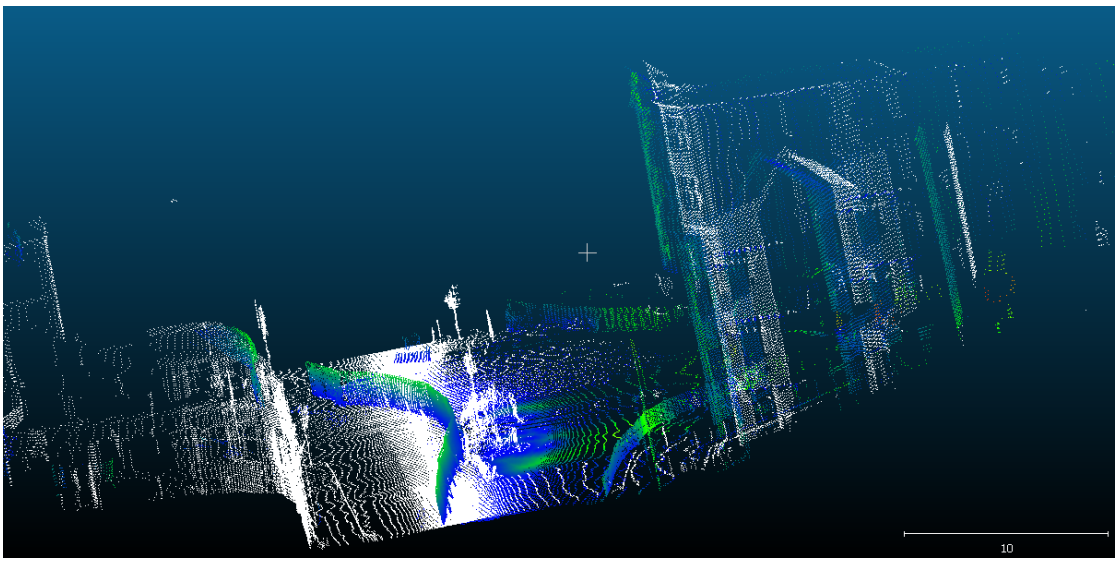

Figure 16. Detailed visualization of changes in place 2 in CloudCompare.

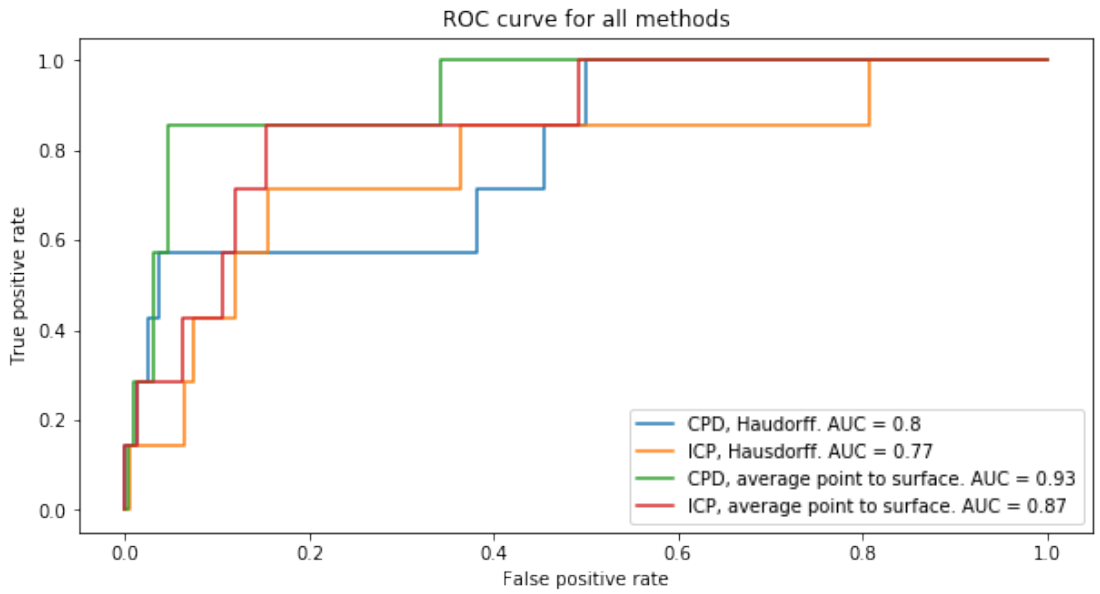

Figure 17. ROC curve for each approach with corresponding AUC scores.

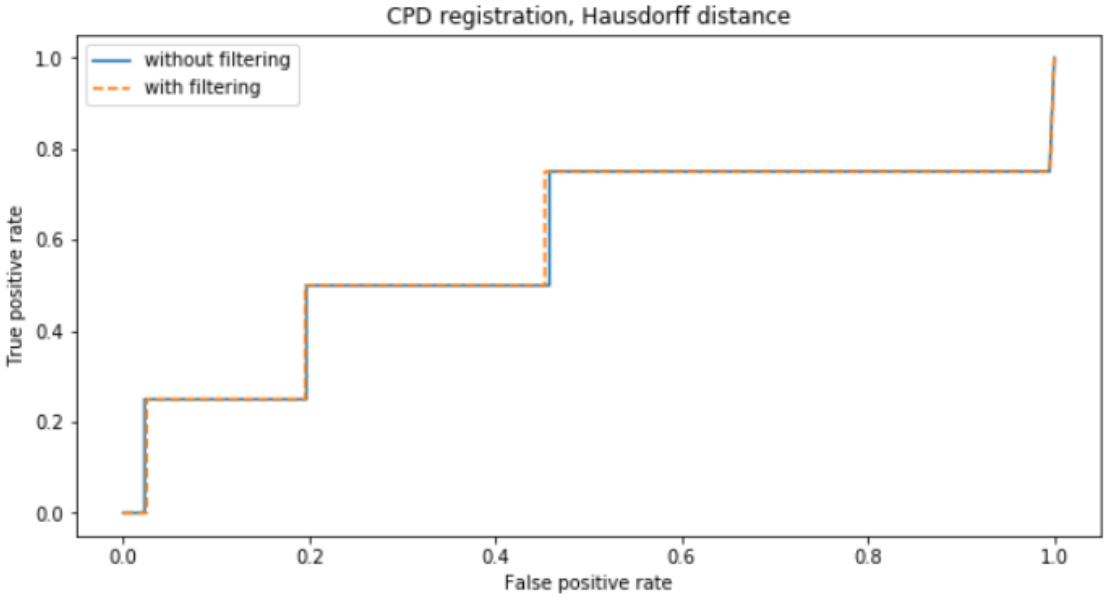

Figure 18. ROC curve for CPD registration, Hausdorff distance and different options for the filtering.

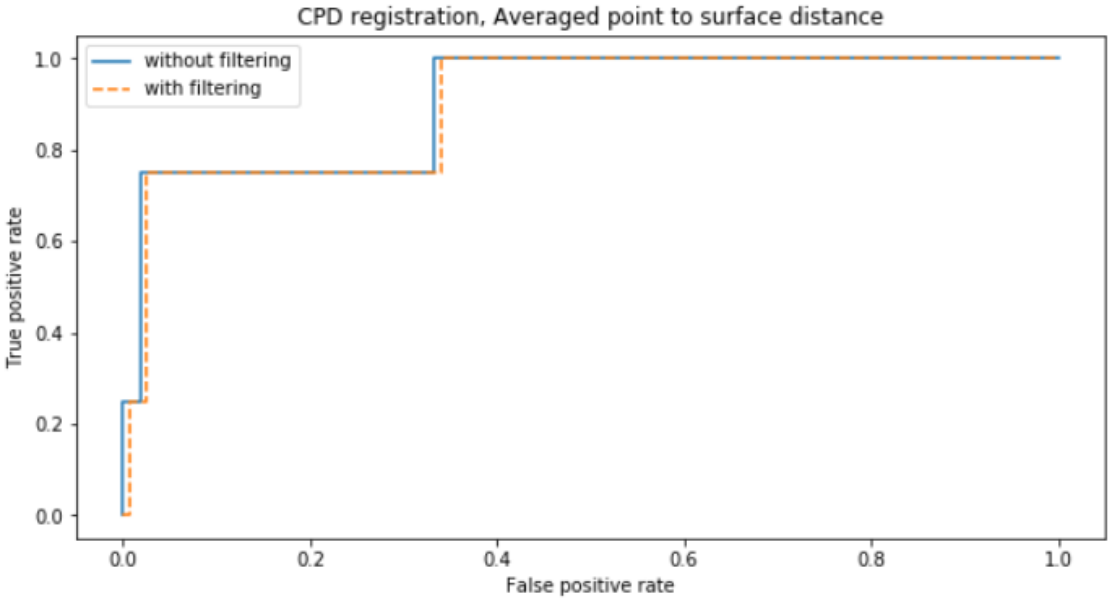

Figure 19. ROC curve for CPD registration, Averaged point to surface distance and different options for the filtering.

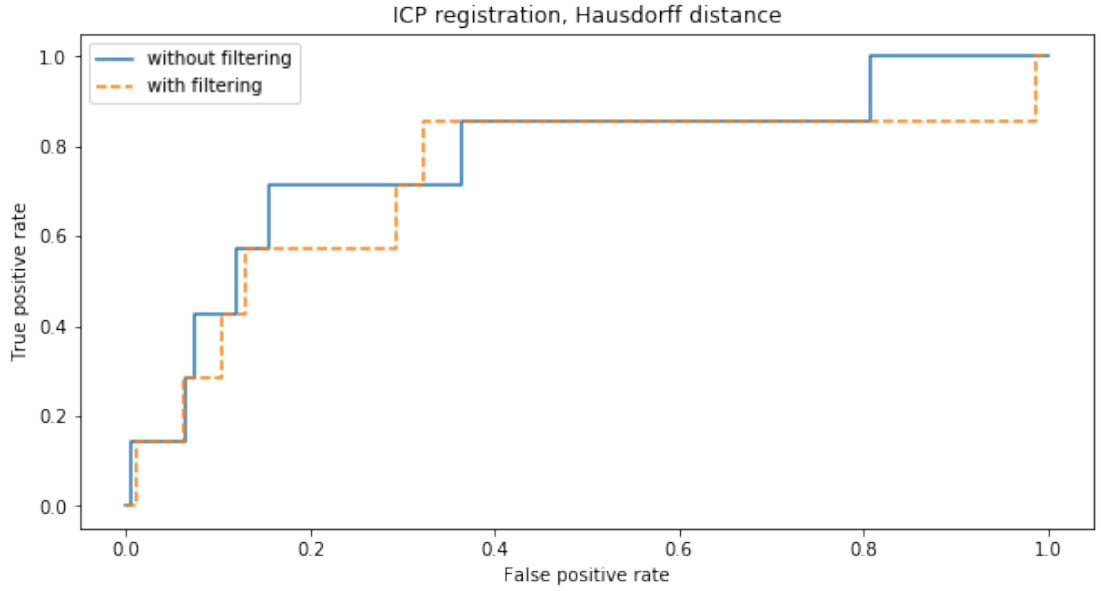

Figure 20. ROC curve for ICP registration, Hausdorff distance and different options for the filtering.

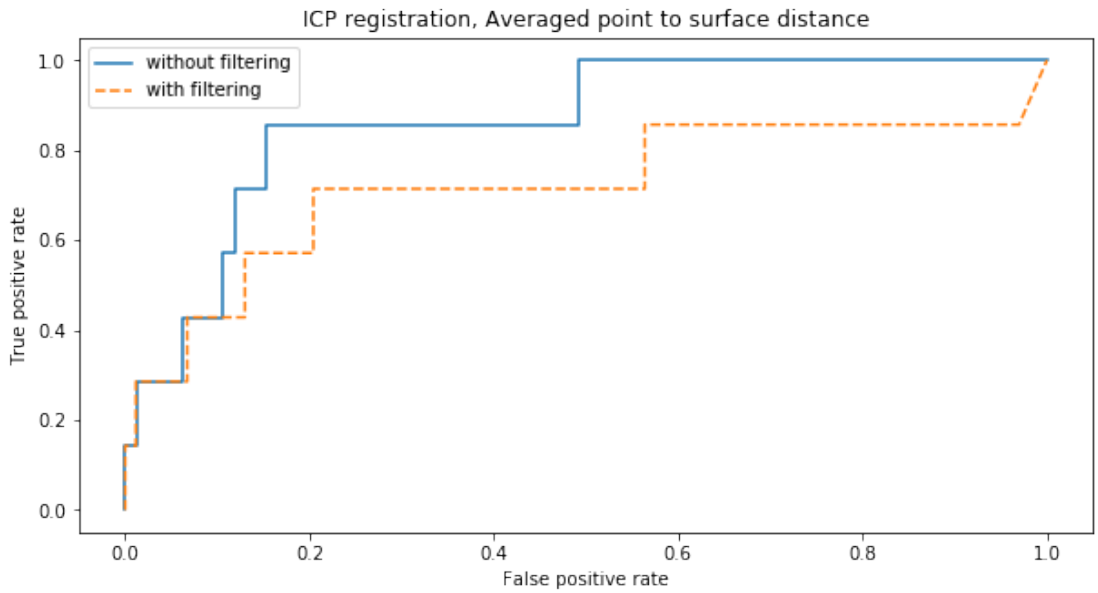

Figure 21. ROC curve for ICP registration, Averaged point to surface distance and different options for the filtering.

Figure 22. The condition on the place 1 on the prebuilt map and during the current ride.

Figure 23. The condition on the place 2 on the prebuilt map and during the current ride.

Figure 24. The condition on the place 3 on the prebuilt map and during the current ride.

Figure 25. The condition on the place 4 on the prebuilt map and during the current ride.

7 Conclusions

Considering all said above, automatic updates of the maps is a perspective and achievable goal, however, a lot of further research should be done. Approaches tested in this work have shown good performance with the significant changes, like big solid fences, while short and holey fences it is almost impossible to detect. For example change in place 1 was not detected with any approach, mainly due to the characteristics of the change that have occurred in this situation. LiDAR beams simply go through the holes and can not collect sufficient numbers of points associated with this fence.

One of the main limitations of this work is that existing algorithms and tools are unable to assess the registration results automatically which makes inevitable visual evaluation. The point registration techniques are very susceptible to local minimum and creating insufficient alignment. Registration failures result in the high number of false positives and were ignored when performance of the registration algorithms was assessed.

One of the objectives which were studied in this work is how the dynamic objects filtering influences the pipeline's performance.This work shows that the dynamic objects filtering either doesn't affect the results or makes them worse. Most probably this is due to a very coarse and rough deletion of the dynamic objects. Object detection is done using bounding boxes around the object. With the objects filtration, a part of the road could be deleted, or any other points which are not related to the dynamic object, but happens to be near it. A possible solution to this problem is the detection of the points related to the road. Possibly, road points should be deleted altogether. The road points are redundant because they form a static part of the point clouds. Therefore deleting them will increase the influence of the changes on the road, from the new objects like fences, new constructions, etc. Also, points hidden by the car could not be restored, therefore a significant part of the information is lost.

Possible improvement could be done by increasing the accuracy of dynamic object detection and not only deleting the cars' points, but also predict points hidden by cars. This RobotCar dataset does not have any labeled points, which is why it was impossible to train the Point Pillars neural network on this dataset. Further work may include retraining the Point Pillars neural network on the RobotCar dataset. With the aim not only delete the objects, but also add points which were excluded due to the cars.

Further work will be concentrated on the several possible ways of development: change detection using camera images, fusing of the LiDAR data and camera images for changes detection and concentration on the map updates, using some particular or any kind of input information. For the change detection from images, different visual-SLAM methods could be used. Some algorithms like DynaSLAM [3] could also filter dynamic objects from their input, therefore concentrating only on the static objects which will

be present in the map. In the case of the sensor fusion, the research should rely on the works which utilize Dempster-Shafer theory [39, 16] for the estimation of our beliefs in each sensor and combination of these beliefs. Potentially, sensor fusion should give a very comprehensive description of the surrounding environment and in this way give us the possibility to estimate any changes very precisely.

The final aim of change detection is a map update. To achieve this, more effort should be spent on the perception task. With solved segmentation tasks, for both camera images and point clouds, we can classify the type of detected changes and renew the map. This work initiates the investigation of a consistent pipeline for the automated maps update, with a combination of different sensors, which is necessary for today's autonomous vehicles researchers community.

8 Acknowledgement

I would like to thank my supervisor Dmytro Fishman for his continuous and inspiring support, guidance and ideas on this project. In addition, I would like to express my gratitude to Alina Vorontseva, Artjom Lind, Dmitry Fediukov, Edgar Sepp, Maxandre Ogeret and Tambet Matiisen for their thoughtful technical and methodological support.

This work was carried out in part in the High Performance Computing Center of University of Tartu.

This work was funded by European Social Fund via Smart Specialization project with Bolt.

References

[1] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. "The wave kernel signature: A quantum mechanical approach to shape analysis". In: *2011 IEEE international conference on computer vision workshops (ICCV workshops)*. IEEE. 2011, pp. 1626–1633.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[3] Berta Bescos et al. "DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 4076–4083.

[4] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes". In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. International Society for Optics and Photonics. 1992, pp. 586–606.

[5] Jennifer R Cutter et al. "Image-based Registration for a Neurosurgical Robot: Comparison Using Iterative Closest Point and Coherent Point Drift Algorithms." In: *MIUA*. 2016, pp. 28–34.

[6] Xiaolong Dai, Siamak Khorram, and Heather Cheshire. "Automated image registration for change detection from landsat thematic mapper imagery". In: *IGARSS'96. 1996 International Geoscience and Remote Sensing Symposium*. Vol. 3. IEEE. 1996, pp. 1609–1611.

[7] Thomas Eiter and Heikki Mannila. "Distance measures for point sets and their computation". In: *Acta Informatica* 34.2 (1997), pp. 109–133.

[8] Daniel J Fagnant and Kara Kockelman. "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations". In: *Transportation Research Part A: Policy and Practice* 77 (2015), pp. 167–181.

[9] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[10] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. "Real-time motion planning for agile autonomous vehicles". In: *Journal of guidance, control, and dynamics* 25.1 (2002), pp. 116–129.

[11] Andreas Geiger et al. "Vision meets robotics: The kitti dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[13] William Rowan Hamilton. “VI. On quaternions; or on a new system of imaginaries in algebra”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 29.191 (1846), pp. 26–31.

[14] Xian-Feng Han et al. “A review of algorithms for filtering the 3D point cloud”. In: *Signal Processing: Image Communication* 57 (2017), pp. 103–112.

[15] Felix Hausdorff. “Grundzüge der Mengenlehre, Leipzig”. In: *de Gruyter & Co* 1927 (1914), p. 1935.

[16] Kichun Jo, Chansoo Kim, and Myoungho Sunwoo. “Simultaneous localization and map change update for the high definition map-based autonomous driving car”. In: *Sensors* 18.9 (2018), p. 3145.

[17] Thomas Knudsen and Brian P Olsen. “Automated change detection for updates of digital map databases”. In: *Photogrammetric Engineering & Remote Sensing* 69.11 (2003), pp. 1289–1296.

[18] Alex H Lang et al. “Pointpillars: Fast encoders for object detection from point clouds”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12697–12705.

[19] Jesse Levinson and Sebastian Thrun. “Robust vehicle localization in urban environments using probabilistic maps”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 4372–4378.

[20] Rong Liu, Jinling Wang, and Bingqi Zhang. “High Definition Map for Automated Driving: Overview and Analysis”. In: *The Journal of Navigation* (), pp. 1–18.

[21] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[22] John G Lyon and Jack McCarthy. *Wetland and environmental applications of GIS*. CRC Press, 1995.

[23] Will Maddern et al. “1 year, 1000 km: The Oxford RobotCar dataset”. In: *The International Journal of Robotics Research* 36.1 (2017), pp. 3–15.

[24] Andriy Myronenko and Xubo Song. “Point set registration: Coherent point drift”. In: *IEEE transactions on pattern analysis and machine intelligence* 32.12 (2010), pp. 2262–2275.

[25] Travis Oliphant. *Guide to NumPy*. Jan. 2006.

[26] OA Pakhomova and O Ja Kravets. “Efficiency analysis of dynamic object detection in computer vision system”. In: *Journal of Physics: Conference Series*. Vol. 1203. 1. IOP Publishing. 2019, p. 012048.

[27] Florian Piewak et al. “Fully convolutional neural networks for dynamic object detection in grid maps”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 392–398.

[28] Charles R Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[29] Radu Bogdan Rusu et al. “Aligning point cloud views using persistent feature histograms”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 3384–3391.

[30] Lars Ruthotto and Jan Modersitzki. “Non-linear Image Registration.” In: *Handbook of Mathematical Methods in Imaging* 2 (2015), pp. 2005–2051.

[31] Glenn Shafer. “Dempster’s rule of combination”. In: *International Journal of Approximate Reasoning* 79 (2016), pp. 26–40.

[32] Robert Simson et al. *The elements of Euclid*. Desilver, Thomas & Company, 1838.

[33] Hang Su et al. “Multi-view convolutional neural networks for 3d shape recognition”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 945–953.

[34] Andrey N Tikhonov and Vasiliy Y Arsenin. “Solutions of ill-posed problems”. In: *New York* (1977), pp. 1–30.

[35] Peter GM Van der Heijden, Jos Dessens, and UIf Bockenholt. “Estimating the concomitant-variable latent-class model with the EM algorithm”. In: *Journal of Educational and Behavioral Statistics* 21.3 (1996), pp. 215–229.

[36] Guowei Wan et al. “Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 4670–4677.

[37] 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)*. IEEE. 2015, pp. 2814–2821.

[38] Zhirong Wu et al. “3d shapenets: A deep representation for volumetric shapes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.

[39] Wen Xiao, Bruno Vallet, and Nicolas Paparoditis. “Change detection in 3d point clouds acquired by a mobile mapping system”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* 1.2 (2013), pp. 331–336.

[40] Hao Zhu et al. “A review of point set registration: From pairwise registration to groupwise registration”. In: *Sensors* 19.5 (2019), p. 1191.

Appendix

I. Glossary

LiDAR - Light Radar
GNSS - Global Navigation Satellite System
INS - Inertial Navigation System
IMU - Inertial Measurement Unit
ICP - Iterative Closest Point
CPD - Coherent Point Drift
SLAM - Simultaneous Localization and Mapping
SLAMCU - Simultaneous localization and mapping changes update
HD map - High Definition map
RBPF - Rao–Blackwellized particle filter
CNN - Convolutional Neural Network
DNN - Deep Neural Network
GPS - Global Positioning System
ROS - Robot Operating System
ROC curve - Receiver operating characteristics curve

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Vladyslav Fediukov**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Detection of changes in maps using LiDAR point clouds,
   (title of thesis)

   supervised by Dmytro Fishman.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Vladyslav Fediukov
10/03/2020