Eötvös Loránd University

Faculty of Informatics

Dept. of Software Technology And Methodology

Powerline tracking and extraction from dense LiDAR point clouds

Supervisor:
Máté Cserép
Assistant Lecturer

Author:
Friderika Mayer
Computer Science for
Autonomous Systems MSc

Budapest, 2020

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

DIPLOMAMUNKA TÉMABEJELENTŐ

Hallgató adatai:

Név: Mayer Friderika

Neptun kód: RIIXJI

Képzési adatok:

Szak: autonómrendszer-informatikus, mesterképzés (MA/MSc)

Tagozat: Nappali

Belső témavezetővel rendelkezem

Témavezető neve: *Cserép Máté*

munkahelyének neve: ELTE IK, Programozáselmélet és Szoftvertechnológiai Tanszék

munkahelyének címe: 1117 Budapest, Pázmány Péter stny. 1/C.

beosztás és iskolai végzettsége: egyetemi tanársegéd, programtervező informatikus MSc

A diplomamunka címe: *Powerline tracking and extraction from dense LiDAR point clouds*

A diplomamunka témája:

(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben diplomamunka témájának leírását)

LiDAR or Light Detection and Ranging is a surveying method that measures distance to a target by illuminating the target with laser light and measuring the reflected light with a sensor. With the differences in laser return times and wavelengths we can create 3D point clouds. LiDAR is used to make high-resolution maps. With the increasing popularity of laser scanning technologies and photogrammetry, point cloud processing turned into an important field of research. LiDAR 3D maps are used by many fields, such as control of autonomous cars, geography, archaeology and forestry. The goal is to create and implement a fast, robust and efficient algorithm which detects and follows power lines. The input for the method is a 3D point cloud and a smaller point cloud, which is the seed. The task is to find the cable, then follow it through on its full length. For the detection the algorithm searches for a direction vector in the point cloud of the seed, for which the shape of the cross-section is circle. The output is a point cloud which consist every point of the cable and a polygon, which indicates the spatial centreline of the cable. The implementation will be realized in C++. The input point cloud contains railroad data and the results will be used for verifying the conditions of the railroads. The data is provided by MLS (Mobile Laser Scanning) method. MLS has become quite popular in recent years and has numerous applications. There are some difficulties, such as there are masts and cantilevers, which are connected to the power lines, the method should ignore these as well as the surrounding environment.

Budapest, 2019.12.01.

Contents

Chapter 1

Introduction

LiDAR or Light Detection and Ranging is a surveying method that measures distance to a target by illuminating the target with laser light and measuring the reflected light with a sensor. With the differences in laser return times and wavelengths we can create 3D point clouds. LiDAR is used to make high-resolution 3D models. With the increasing popularity of laser scanning technologies and photogrammetry, point cloud processing turned into an important field of research. LiDAR 3D maps are used by many fields, such as control of autonomous cars, geography, archaeology and forestry.

Power lines are vital components of the railway infrastructure. The cables can degrade from natural phenomenons such as growth of trees – which might undermine infrastructure and even result in widespread power failures and forest fires –, or storms [1]. It is critical to monitor the status of power lines on a regular basis to ensure safe and reliable transmission. The monitoring of these cables are usually done manually, which is time and resource consuming. An automated process can be more precise, fast and efficient, which means safety and reliability.

In this work the goal is to create and implement a fast, robust and efficient algorithm which detects and follows power lines. The input of the method is a 3D point cloud and a smaller point cloud, which is the seed. The seed contains only power line points in a small section from the original point cloud. This information is used for locating the transmission lines and filtering out irrelevant points. The

task is to find the cable, then follow it through on its full length. For the detection the algorithm searches for a direction vector in the point cloud of the seed, for which the shape of the cross-section is circle. The output is a point cloud which consist every point of the cable and a polyline, which indicates the spatial centreline of the cable. The input point cloud contains railroad data and the results will be used for verifying the conditions of the railroads. There are some difficulties, such as there are masts and cantilevers, which are connected to the power lines, the method should ignore these as well as the surrounding environment.

The data is provided by MLS (Mobile Laser Scanning) method. MLS has become quite popular in recent years and has numerous applications. The point cloud was collected by a Riegl VMX-450 high density mobile mapping system (MMS) mounted on a railroad vehicle, operating at $60km/h$. The sampled area was about $18.5km$ long and $130m$ wide rural railroad segment in Hungary. The dataset is massive, containing $1,544,178,582$ points.

Chapter 2

Background

2.1 LiDAR

LiDAR stands for Light Detection And Ranging. LiDAR uses laser to measure distances. First it emits a laser pulse on a surface, after that it catches the reflected laser back to the source with sensors, finally it measures the time that the laser has travelled and calculates the distance. LiDAR has 360 degree field of vision thanks to its moving parts. Figure 2.1 represents the principle of range measurement using laser. The laser is transmitted from the transmitter and the reflected energy is captured by the receiver. [2]

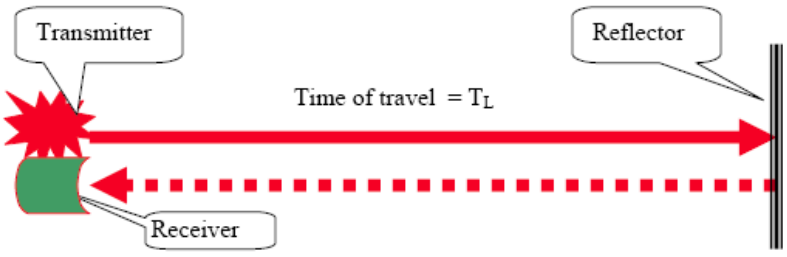


Figure 2.1: Principle of range measurement

2.2 Data acquisition methods

The most important data acquisition methods are MLS (mobile laser scanning), ALS (aerial laser scanning) and TLS (terrestrial laser scanning). We can also get 3D data from Kinect®-like devices.

2.2.1 MLS

MLS systems contain an imaging unit and a navigation unit. Imaging unit incorporates laser scanners and digital cameras. Navigation unit includes GNSS (Global Navigation Satellite Systems) and INS (inertial navigation system) systems. For instance GPS is a GNSS system. GNSS systems are SAT based navigation systems for positioning. INS is composed of a computer, motion sensors and rotation sensors. For example, in [3] a 2D mobile laser scanning system was used, consisting of a 2D laser scanner Z+F Profiler 9012A (ZOLLER+FRÖHLICH 2016) and an INS iMAR iNAV-FJI-LSURV (IMAR 2016). The set-up is shown in Figure 2.2.

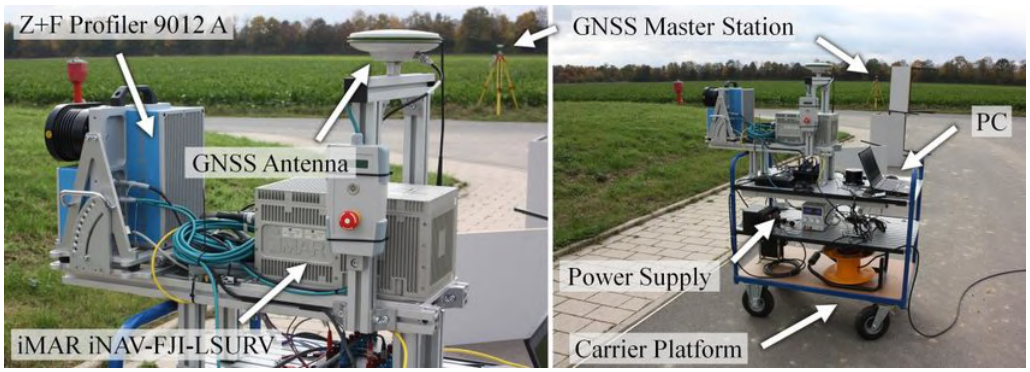


Figure 2.2: 2D Mobile Laser Scanning System

2.2.2 ALS

ALS or Airborne Laser Scanning systems are composed of GNSS systems and LiDAR scanners. These are usually mounted on aerial vehicles. This approach provides high-density clouds for 3D surface models with transmitted laser beams. The operation of an ALS system is shown in Figure 2.3. The image shows schematically a laser scanner and its main components. [4]

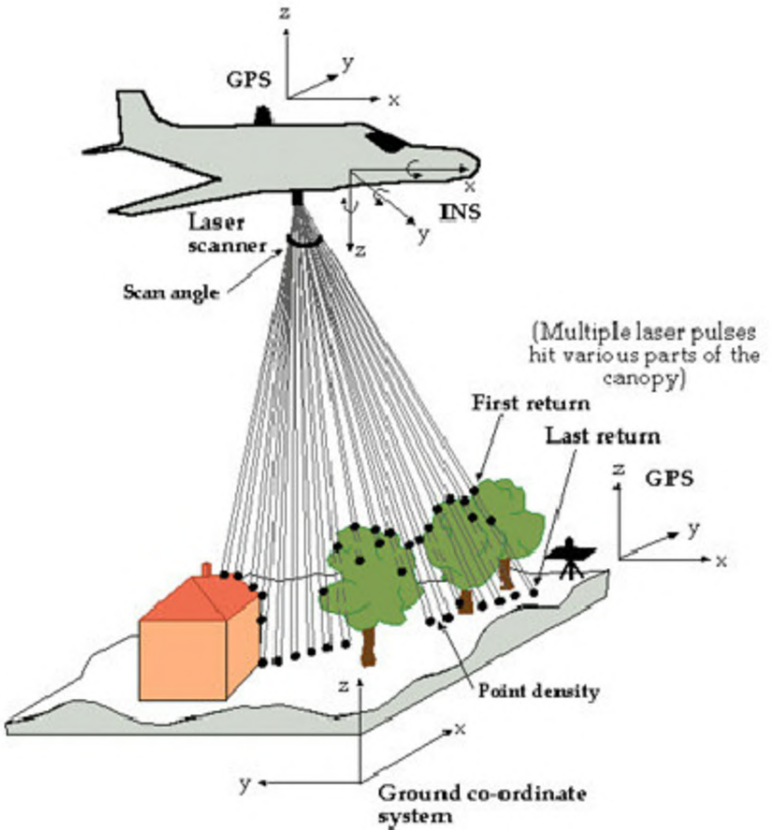


Figure 2.3: Operation of an Aerial Laser Scanning system

2.2.3 TLS

TLS systems are ground-based version of the airborne method. It is used for high-resolution mapping of terrain, vegetation, and other landscape features. This technology has limited distances in range (50-300 m). Similarly to ALS, TLS contains active sensors.[5]. These sensors emit laser signals to calculate distances based on the time delay of the returned laser pulses. They create a dense array from the distance return values, which can be transformed into digital 3D landscape models. [6] For example, in [7] for ground-based terrestrial laser scanning two hyperspectral cameras, a retro reflective cylinder and a terrestrial laser scanner were used. (2.4)

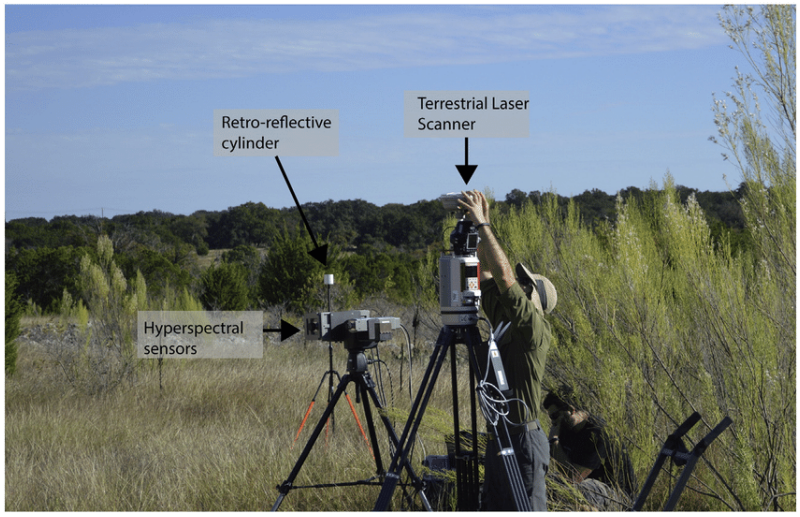


Figure 2.4: Ground-based TLS

2.3 Data formats

The most common data formats for 3D point clouds are PLY, LAS, XYZ and PCD.

2.3.1 LAS

LAS is the most widespread data format for LiDAR data. It stores binary data in little-endian format. Files consist a public header block, Variable Length Records, Point Data Records and Extended Variable Length Records. The header describes for example the format and the number of points. It can store GPS time, RGB and NIR color and wave packet information [8]. In this work LAS file format was used. A LAS or LAZ file contains no enforced ordering of the data points by any attribute, therefore spatial indexing is required to for example query neighbouring the points. Though a binary format, LAS files can grow extremely large, therefore compression is often used on it. LAZ [9] is a losslessly compressed variant of LAS. Table 2.1 and 2.2 shows the formats of the Public Header Block and the Point Data Record Format 0.

Table 2.1: Public Header Block

Item	Format	Size	Required
File Signature ("LASF")	char[4]	4 bytes	*
File Source ID	unsigned short	2 bytes	*
Global Encoding	unsigned short	2 bytes	*
Project ID - GUID data 1	unsigned long	4 bytes	
Project ID - GUID data 2	unsigned short	2 bytes	
Project ID - GUID data 3	unsigned short	2 bytes	
Project ID - GUID data 4	unsigned char[8]	8 bytes	
Version Major	unsigned char	1 byte	*
Version Minor	unsigned char	1 byte	*
System Identifier	char[32]	32 bytes	*
Generating Software	char[32]	32 bytes	*
File Creation Day of Year	unsigned short	2 bytes	*
File Creation Year	unsigned short	2 bytes	*
Header Size	unsigned short	2 bytes	*
Offset to point data	unsigned long	4 bytes	*
Number of Variable Length Records	unsigned long	4 bytes	*
Point Data Record Format	unsigned char	1 byte	*
Point Data Record Length	unsigned short	2 bytes	*
Legacy Number of point records	unsigned long	4 bytes	*
Legacy Number of points by return	unsigned long[5]	20 bytes	*
X scale factor	double	8 bytes	*
Y scale factor	double	8 bytes	*
Z scale factor	double	8 bytes	*
X offset	double	8 bytes	*
Y offset	double	8 bytes	*
Z offset	double	8 bytes	*
Max X	double	8 bytes	*
Min X	double	8 bytes	*
Max Y	double	8 bytes	*
Min Y	double	8 bytes	*
Max Z	double	8 bytes	*
Min Z	double	8 bytes	*
Start of Waveform Data Packet Record	unsigned long long	8 bytes	*
Start of first Extended Variable Length Record	unsigned long long	8 bytes	*
Number of Extended Variable Length Records	unsigned long	4 bytes	*
Number of point records	unsigned long long	8 bytes	*
Number of points by return	unsigned long long[15]	120 bytes	*

Table 2.2: Point Data Record Format 0

Item	Format	Size	Required
X	long	4 bytes	*
Y	long	4 bytes	*
Z	long	4 bytes	*
Intensity	unsigned short	2 bytes	
Return Number	3 bits (bits 0 – 2)	3 bits	*
Number of Returns (given pulse)	3 bits (bits 3 – 5)	3 bits	*
Scan Direction Flag	1 bit (bit 6)	1 bit	*
Edge of Flight Line	1 bit (bit 7)	1 bit	*
Classification	unsigned char	1 byte	*
Scan Angle Rank (-90 to +90) – Left side	char	1 byte	*
User Data	unsigned char	1 byte	
Point Source ID	unsigned short	2 bytes	*

2.3.2 PLY

PLY files store vertices. The format can be binary or ASCII. It can contain properties such as colour, transparency, surface normals, texture coordinates and data confidence values.

2.3.3 XYZ

XYZ format is a very simple way to store points in 3D space. The point coordinates are written on a single line, separated by spaces, tabs or commas. It's also used in chemistry for storing molecule geometry.

2.3.4 PCD

PCD is the native file format of PCL (Point Cloud Library). It contains two sections: a header file with informations like the number, size, dimensionality and data type of the point cloud, and an ASCII or binary section which defines the data.

2.4 Reviewed methods

2.4.1 L_1-Medial Skeleton of Point Cloud

The method L_1-Medial Skeleton of Point Cloud [10] by Huang et al. can be used on raw data with and also can deal with noise, outliers and large areas of missing data. The method is fast and robust.

L_1-median

L_1-median represents a unique global center of a given set of points. In this method the so-called L_1-medial skeleton was introduced. With this structure one can get an optimal set of projected points $X = \{x_i\}_{i \in I}$:

$$\underset{x}{\operatorname{argmin}} \sum_{i \in I} \sum_{j \in J} ||x_i - q_i|| \theta(||x_i - q_i||) + R(X)$$

where $Q = \{q_j\}_{j \in J} \subset \mathbb{R}^3$, the first term is a localized L_1-median of Q, $R(X)$ regularizes the local point distribution of X. $\theta(r) = e^{-r^2/h(h/2)^2}$ is the weight function, where h indicates the size of the supporting local neighbourhood for L_1-medial skeleton construction.

Algorithm

The input is $Q = \{q_j\}_{j \in J} \subset \mathbb{R}^3$, unoriented, raw point set. The output is a curve skeleton representing a one-dimensional local center of the shape underlying the input. The main steps are

- Random set of point selected from the data
- Each point is iteratively projected and redistributed to the center of the input points within its local neighbourhood
- The size of the neighbourhood is increased

The algorithm can be non-uniform and the generated branches may be off-centered, the solutions are for these problems: density-based weighting and re-centering.

L_1-medial construction This method searches for a set of local L_1-median centers.

With regularization term R we can prevent accumulation one points are already contracted onto their local center positions. $R(X)$ adds a repulsion force whenever a skeleton branch is formed locally.
After regularization the algorithm identify the branch points. First, all sample points are considered as non-branch points. For finding outliers the K-nearest neighbourhood method was used.

Results

The method was tested with various shapes. According to the tests the algorithm is robust against noise, outliers, non-uniformity and missing data. It can handle well non-cylindrical and complex shapes as well. The average running time is about 1 minute for 100K points.

2.4.2 3D Geometric Analysis of Tubular Objects Based on Surface Normal Accumulation

The method by Kerautret et al. [11] is a centerline extraction algorithm based on surface normal accumulation.

Algorithm

The first step is to compute a 3D accumulation image, which counts for each voxel how many faces of input data have their normal vector pointing through the voxel. The next step is a tracking algorithm, which extracts an approximate centerline by following local maxima in the accumulation image. The third step is optimization, with this step we can remove digitization effects.

First step The input of this algorithm is a set of faces, their associated normal vectors and a 3D digital space. The output is the number of normal vectors and a vector estimating the tube local main directions. From position and normals of faces of an input mesh, the first algorithm computes an accumulation image by a directional scan starting from a face center in the direction of the face inward normal.

Second step The second algorithm is a patch center tracking algorithm in one direction with a given starting point and an orientation. The input is the output from the first algorithm, the output is the centerline.

Third step The third step is the optimization. This algorithm outputs a perfectly centered spine line. In this step the method fit an error, which is as the sum of the squared difference between the known tube radius and the distance between the tube center and its associated input mesh points.

Results

The algorithm was tested on a dataset which contains several types of metallic tubes. The method was tested on objects which contained faces in a range of between 444 and 654450. The running time was always less than 30 seconds. Some errors are present near bent areas. This method is robust to missing data and to perturbations.

2.4.3 Modeling of Sparsely Sampled Tubular Surfaces Using Coupled Curves

This method by Schmidt et al. [12] uses B-splines for curve approximation.

Algorithm

The method defines a tube as a function mapping a curve parameter $u \in \mathbb{R}$ to the $(D+1)$-dimensional vector $(\mathbf{a}^T(u), t(u))^T$, where $\mathbf{a} : \mathbb{R} \to \mathbb{R}$ is the corresponding tube thickness function. The tube surface points are $X = \{x_i, \ldots, x_n\}, x_i \in \mathbb{R}$ The algorithm minimizes the energy

$$E_{data}(\mathbf{a}, t) := \sum_{i=1}^{n} \psi(||\mathbf{a}(u_i| - x_i|| - t(u_i))^2)$$

where $u_i := \text{argmin}_u \, ||x_i - \mathbf{a}(u)||$ is the curve parameter projection of x_i and $\psi(p^2)$ is a robust distance measure.
The method also uses smoothness terms, which penalizing axis curvature and tube thickness variations.

Parametrization using B-Splines The curves are approximated with open B-splines of degree p, therefore the nodes at the spline endpoints are repeated $p + 1$ times. The algorithm requieres point set X, initial cylinder and parameters λ, μ, τ. The algorithm returns the coupled B-spline model (a, t) The first step is to initialize each model (a, t) with two knots fitting the initial cylinder, the next step is to compute the initial energy $E(a, t)$. Until convergence, the method minimizes $E(a, t)$, insert the knot and re-parametrize the model.

Extension to multiple tubes For multiple tubes we can place a seeding cylinder for each. For this approach in every iteration step we can partition our point set into subsets. In this case the energy is the sum over all of the models' energies.

Results

The approach is able to follow tubes of very complex bending patterns and also allows for moderate thickness variations. The error stays below 10% for low noise.

2.4.4 Blood Vessel Segmentation and Centerline Tracking Using Local Structure Analysis

This method [13] was created for tracing blood vessels by Kumar et al.

Algorithm

The input of the algorithm is the initial position seed, the direction seed and the radius. First, the eigenvectors are calculated by Eigen analysis of the Hessian matrix at the seed point. For cross-section detection Canny edge detection was used. The next step is the calculation of the radius by propagating outwards from the seed towards the border along the vessel cross directions and an average of all the edge intensities is taken as the local adaptive threshold at that blood vessel cross-section. The method uses the fact that there are significant change in the radius at the blood vessel bifurcation. The last step is combining all the circleness output at trunk and modified vesselness output at bifurcations. The algorithm terminates if seeds go out of the connected blood vessels or the image region.

Trunk Analysis For this the algorithm uses the so-called circleness filter. This filter helps finding a definite center. It's based on 2D Eigen analysis of the 2D Hessian matrix obtained from the cross-section image. The seed points for the cross-sections are saved into a list.

Bifurcation Analysis The algorithm runs this step when there's a significant change in radius between neighbouring blood vessel cross-sections. The method calculates the modified multiscale vesselness. The bifurcating vessel direction is calculated using Eigen analysis at each center and then used to find the next possible center or seed locations. The seeds are saved into a list.

Results

The algorithm was runned on images. The fastest test run was 7 seconds with $150 * 150 * 150$ image size. The slowest test run was 24 seconds on a $272 * 499 * 88$ sized image. Bifurcating vessel detection accuracy increases with increasing radius.

2.4.5 Catenary System Detection, Localization and Classification Using Mobile Scanning Data

The following algorithm described by Pastucha in [14] uses RANSAC and DBSCAN. There are some assumptions which have to be present for the algorithm, these are: the road must be an almost flat surface with a locally small scope, curbs must be in its immediate vicinity and have a predetermines height difference.

Algorithm

The recognized objects are divided into two groups cables and support structures.

Algorithm structure There are four main steps of the algorithm.

- Data preparation
- Initial density analysis
- Support structure verification and classification
- Data classification clarification

The Support structure verification and classification part contains two criteria: the cantilever criteria and the vertical mast and horizontal beam criteria.

Data Preparation It consists three procedures: ground filtering, reducing point cloud size and setting the calculation order. For ground filtering the TerraScan software's classification algorithms were used. By reducing the point cloud size the method limits the calculation of the later algorithm to points positioned up to a set distance horizontally from the registered trajectory. The last prodecure is an indexing system which keep order in point cloud processing and able to directly relate to specific parts of the data.

Initial Density Analysis Point density is used to limit the search for support structures.

Support Structure Verification and Classification It contains two criteria the vertical mast criteria, with this, we can recognize support structures directly next to the track and horizontal beam criteria, which is used for classifying multi track solution support structures. For detecting the cables RANSAC algorithm was used.

Data Classification and Clarification The classification is divided into two separate operations, the first considers support structures, the second catenary wires. For perform the classification, a modification of DBSCAN was used.

Results

The method was able to detect over 97% of existing support structures, the rate of false positives was less than 0.2%. Out of 1645 poles and portal structures, 1581 were classified correctly.

2.4.6 Automatic railway power line extraction using mobile laser scanning data

This method [15] works with Mobile laser scanning system. The system was installed on the train and was travelling at 120 miles an hour. The point density is

approximately 3000 points per square meter.

Algorithm

The method combine three steps: significant point cloud data segment, power line extraction, joint region judgment.

Point cloud segment In this step they filtered out the ground point. Features used for obtain the power line area: there are protective fences between edge of railways and protective fences elevation higher than trajectory, trajectory coordinates are known, trajectory is about 2m above rails, horizontal suspension wire power line is located more than 4m above the trajectory. By knowing these features we can build a bound box around the power line area.

Power line extraction For this step a self-adaptive space region growing method was used. Algorithm: Generate seeds. Select a point as seed and a growth direction in the initial position of cloud. Generate a cross section according seed and its direction and make it to grow at a scalable length. Using PCA and information entropy to determine the label of the power line point cloud which is included in the growth space. Generate a new seed and a new direction overlap it again until this seed end. Select the other seed until all seeds have been selected.

Joint region judgment In this step they are labelling points. There are three labels: straight power line, 2D suspension joint, 3D suspension joint. For extracting space features PCA (Principal Component Analysis) was used. Information entropy evaluation was used for computing entropy value and classifying the joint kind with the entropy value. Information entropy is a measurement for the chaotic degree or dispersion degree of distribution.

3D power line fitting As a last step they reconstruct the 3D model of the power line with the catenary curve, which can be computed by

$$z = a(x^2 + y^2) + b\sqrt{x^2 + y2} + c$$

A, b and c should be computed by point cloud data.

Results

The power lines and horizontal suspension line can be separated from vertical suspension lines and electrical poles. The method works well in 3D space.

2.4.7 Automated Recognition of Railroad Infrastructure in Rural Areas from LIDAR Data

This method [16] considers every element of the railroad infrastructure. In this paper we only consider the algorithm for the overhead power cable recognition.

Algorithm

One can get a good approximation of the position of the overhead power cables compared to the track bed. The first step is to get a 3D vector which is connecting each non-track bed point ot its closest point on the track bed.

$$nV_z \leq 0.8$$

where

$$nV_i = \frac{V_i}{\sqrt{V_x^2 + V_y^2}} + V_z^2$$

$n_V z$ is the normalized Z component of vector V and its value is between zero and one. They selected a threshold which includes points only belonging to cables. The above equation selects not only cable points, but cantilevers and masts too. They filter out the false points by searching for points which belong to a linear neighbourhood. To this end, three-dimensional PCA is employed by eigenvalue decomposition of the covariance matrix of local neighbourhoods. Points identified as belonging to a linear neighbourhood include points on contact cables, catenary cables, and return current cables. For identifying points belonging to contact cables a region growing algorithm was used. The method distinguishes the contact cables from catenary cables and return current cables due the fact that contact cables don't have curved shape like the other two types of cables.
Catenary cables can be clustered by their feature that they lie immediately above the contact cables. Return current cables lie above catenary cables.

Results

The method was tested on LiDAR data, with around 12.5 million points. In case of contact cables, there were 1 object in the data set, the algorithm recognized 1 true positive, 30 true negatives, 0 false positive and 0 false negative. For catenary and return current cables the results are the same. The accuracy is above 94.72% in every case, the precision is above 95.87%.

2.4.8 Extraction of tubular shapes from dense point clouds and application to tree reconstruction from laser scanned data

This method [17] is based on an original Hough transform combined with generalized open active contours.

Algorithm

This method contains extraction of tubular shapes from dense point clouds and application to tree reconstruction from laser scanned data. The approach is called STEP (Snakes for Tuboid Extraction from Point clouds).
The basic steps of the STEP algorithm are the following. The first step is to design a point-normal circles Hough transform, which has three main elements: reducing the complexity of the HT by using the normal directions of the points, a filter which reduces the complexity of the subsequent space analysis and maximal Hough space elements are selected as seeds for contour growing.
The second step growing open active contours. In this step the method uses Hough space to identify sets of tuboids. First, they intialise a 4D curve and examine if the stopping criterion is reached, if yes, they examine if the curve is long enough, if yes, they carry out an inverse HT, then they get a single tuboid as a series of 3D circles, they add this tuboid to the result set and start over the algorithm from the second step. If the stopping criterion isn't reached, curve growing and energy minimization methods are used until the curve satisfies the criterion. After that if, the curve isn't long enough, they start over the method from second step.

Results

The method was tested on four datasets.
The first dataset was ranging from $14pts/cm^2$ to $14000pts/cm^2$. The estimated cylinder radii error was less then 0.5cm. With modification on this data the results were similar.
The second data set contained simulations of time-of-flight camera acquisitions. The tuboid reconstruction procedure took in average 110 seconds.
The third data set was a natural forest point cloud. The error was less then 0.4cm. The last data set was taken from the SimpleTree©. The STEP method extracted the tree's structure within the range of 1 min 20 sec to 2 min, including 2 to 5 seconds for the HS computation.

2.4.9 Automated Inspection of Railway Tunnels' Power Line Using LiDAR Point Clouds

This method [18] uses classification and RANSAC for power line detection.

Algorithm

Classification of Points The first step is to label the points, the labels are: lining, ground, railway tracks, overhead line, and cantilevers. First, the base is a railway tunnel point cloud, during pre-processing they divided the original point cloud into 20m long segments. The classification process is based on the geometrical characteristics of the points. Rail tracks are classified by SVM classification, the cantilever and the power line cable are classified using RANSAC.

Contact wire detection Firstly, they distinguished the contact wire from the suspension wire and other elements. The method uses RANSAC for this task.

Suspension wire detection It is known that the suspension wire is always located in the maximum height of the remaining non-classified points of the power line. They also considered, where are the cantilevers, which are the highest points of the catenary curve.

Contact wire height and deflection The contact wire height can be specified by using voxelization on the rail track points, during this process they select the voxels with the biggest z coordinate, after that they fit a plane to the surface of the rails. The next step is voxelization of the contact wire, the result is only one point inside each 50 cm voxel. With these data they measure the distance between the point and the plane, which gives us the contact wire height. The next step is to measure deflection by selecting from the suspension wire points the two most extreme points of the span. They draw a line between these two points, these are located in the highest z coordinate. From the maximum point-line distance between the previously fitted line and each point in the point cloud sections forming a span, the outcome is the deflection.

Results

Recognized elements: contact wire and suspension wire. There were 3 test cases. In the first case, during contact wire detection there were 2011 true positive, 818 false positive and 761 false negative points, the precision was 96.09%, recall was 96.36% and F-score was 96.22%. Data for suspension wire: 7269 true positives, 411 false positive, 379 false negative, precision was 94.65%, recall was 96.04%, and F-score was 94.85%. In the case of the two other test cases, the results were almost the same.

Chapter 3

Methodology

3.1 Dataset

3.1.1 MMS

Mobile Mapping Systems work by transmitted laser beams and making digital raster photos. The mounted imaging sensors usually LiDARs and radars. The output is a dense 3D digital map. The Riegl VMX-450 MMS 3.1 which was used for this project has two spinning mirror laser scanners, digital cameras, GPS, inertial navigation system (INS) and a computer. It was mounted on a railroad vehicle which was operating at 60 km/h. The laser scanners recorded 1.1 million point / sec and the mirrors had 12,000 RPM rotational speed. The system operates with 4-6 cameras and a LadyBUG panoramic camera. The cameras took a photo in every 5 meters.

Figure 3.1: The Riegl VMX-450 sensor

3.1.2 Point cloud

The scanned rural railroad segment is in Hungary, from Szabadszállás railway station to Csengőd railway station (Figure 3.2). It was recorded in 2016. The dataset has an average 3D range precision of 3 mm and a maximum threshold of 7 mm. The positional accuracy was around 3-5 cm. 104,352 digital photos were taken during the process. The examined area was 18.5 km long, 130 m wide and it was scanned in both directions. There are almost 1.55 billion points in the point cloud, the data is stored in LAS format. The files include intensity and RGB data, but this information wasn't used in any of the methods. Figure 3.3 represents the visualized 3D point cloud.

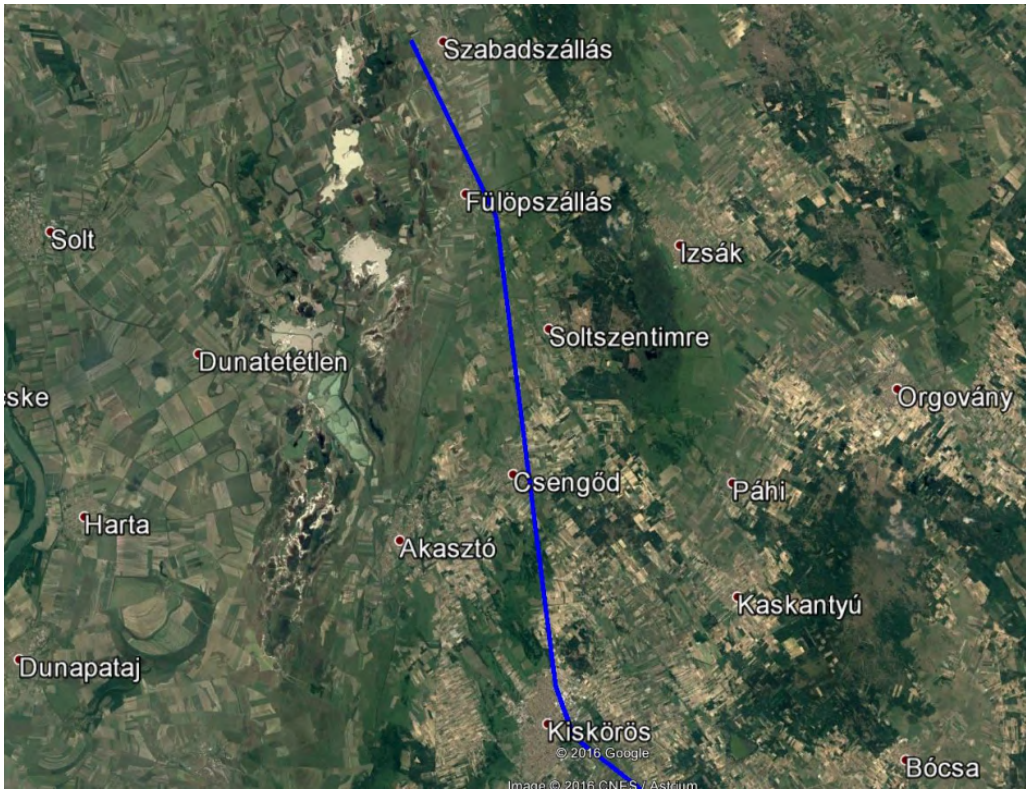


Figure 3.2: The covered area

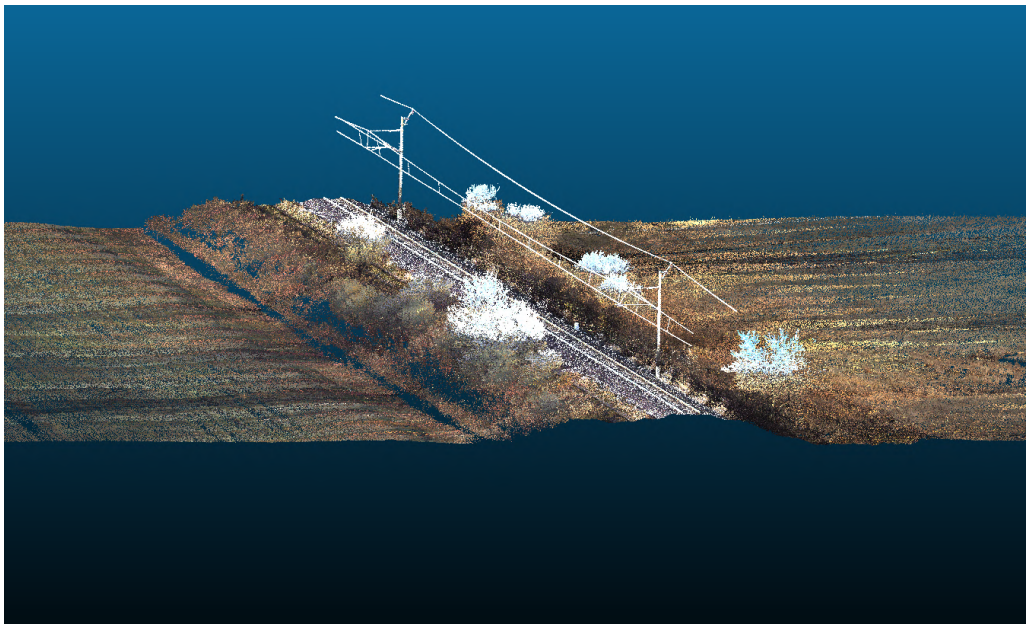


Figure 3.3: 3D visualization of the point cloud

3.2 Analysis

In this work two solutions are considered. The preprocessing method is the same in both cases.

3.2.1 Preprocessing

The first step is to reduce the point cloud size with the help of the seed point cloud. The seed point cloud is a segment of the original dataset and only contains the power lines. With this knowledge, we can filter out points, which are located below the wires. After filtering the calculations are faster and more effective. This process is carried out by finding the lowest point in the seed point cloud and filtering out the points that are placed 1 meter below this point. The preprocessed dataset is shown in Figure 3.4. Since the RANSAC algorithm could detect dense bushes or trees as lines, an another preprocessing step needs to be introduced. Since we assume that the data set doesn't contain curves, a width filter can be used. The points are filtered out by rotating the point cloud to be parallel with a given axis according to the seed cloud, then finding the minimum and maximum in the seed and erasing the unnecessary points. The distance can be adjusted by a parameter.

In case of smaller, for instance 100 meter sections, RANSAC also works well with only the first filter. For this dataset 1.2 meter proved to be the best.

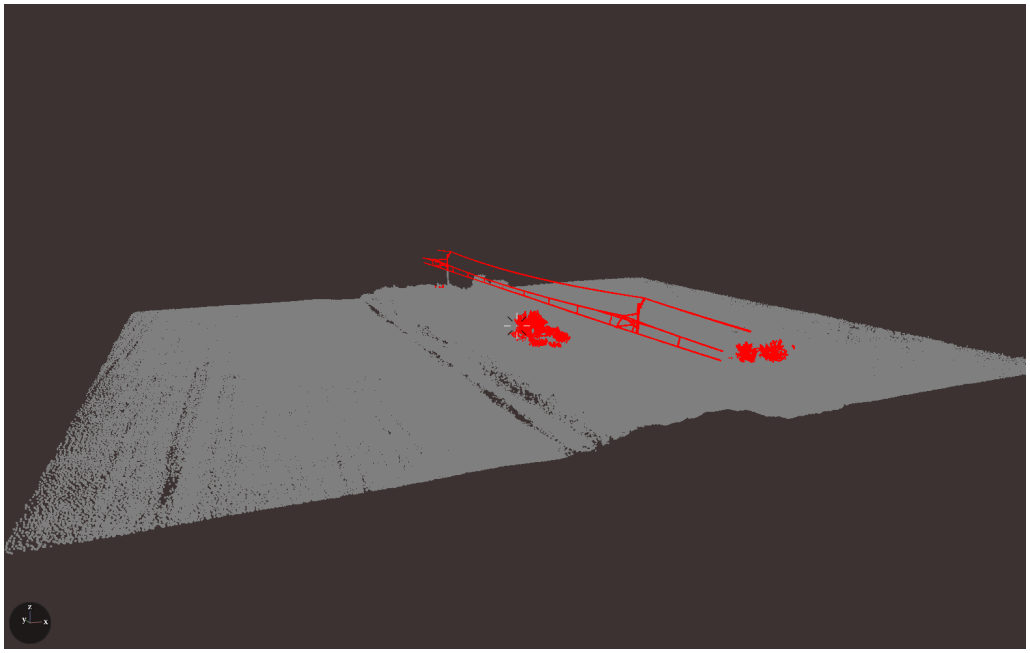


Figure 3.4: Preprocessed point cloud (the result is marked with red)

3.2.2 RANSAC

Several reviewed methods used RANSAC (Random sample consensus). This algorithm was created for fitting mathematical models in point clouds, in our case the model is a 3D line. It was first introduced in 1981 by Martin A. Fischler & Robert C. Bolles [19]. RANSAC is a robust estimation technique. The algorithm can cope with outliers and seek for the optimal result. Unlike other smoothing techniques, which are using as much of data as possible, RANSAC rather use the smallest amount of data as feasible and later enlarges the set. For example, if we're trying the find a single 2D line, the approach will use only two initial points, since this is enough for define a line. RANSAC assumes that all the measurements have the same influence, so it can maximize the number of inliers. [20] It is widely used and popular in computer vision. It has some modified versions like PROSAC, MSAC, MLESAC and the randomized R-RANSAC. The method can be slow if the inlier ratio is low and because of the noise, it takes several times longer than theoretically expected. RANSAC The visual representation of the functioning of RANSAC algorithm can be seen on Figure 3.5. In this work the PCL implementation of the algorithm was used,

with the parameter of an axis and a threshold of 0.9 meter. This parametrization proved to be the most reliable considering that the cables aren't straight lines.

Algorithm

The basic RANSAC algorithm according to Konstantinos G. Derpanis [21].

Algorithm 1 RANSAC

1: Select randomly the minimum number of points required to determine the model parameters.
2: Solve for the parameters of the model.
3: Determine how many points from the set of all points fit with a predefined tolerance ϵ.
4: If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold τ, re-estimate the model parameters using all the identified inliers and terminate.
5: Otherwise, repeat steps 1 through 4 (maximum of N times).

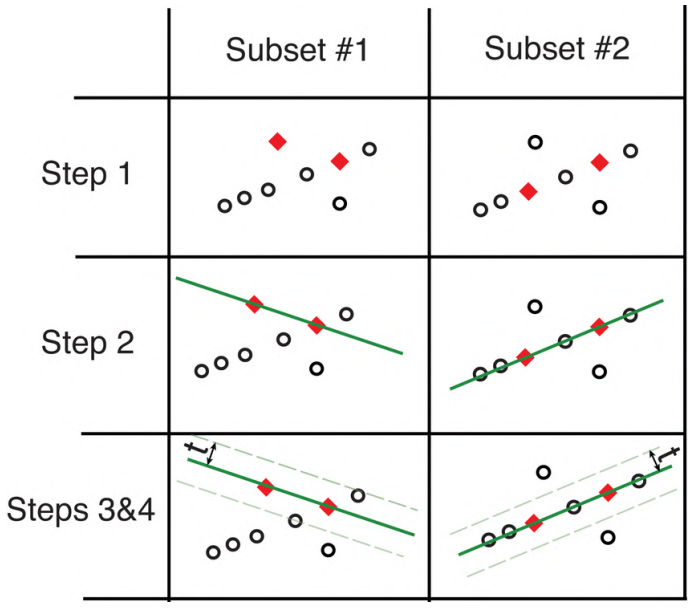


Figure 3.5: Visual representation of RANSAC iterations [20]

3.2.3 Hough transform

The Hough transform also popular solution for the problem of identifying power lines. It is a feature extraction technique which uses a voting procedure. It was first used for detecting lines in images, later it was extended for identifying arbitrary objects. The method is popular and mostly used in digital image processing and computer vision. Nowadays we use the algorithm which was introduced by Richard O. Duda and Peter E. Hart [22]. It uses normal parametrization, for example, in the case of a straight line the geometry is $x \cos\theta + y \sin\theta = \rho$. The transform is usually considered in 2D space, thus in this paper a 3D modification was used.

Generalized Hough transform

This is the general method which can be used in 2D. The first step is edge detection, for example with Canny edge detector. The next step is to map these point to the Hough space and to create an accumulator array. Before the voting process, the all entries in the accumulator array are initialized 0. For every edge point and – for example in the case of circles – for every possible radius compute the objects' parameters and increment the associated accumulator array entry. The last step is to find local maxima in the accumulator array. Neeti Taneja, Mohammad Shabaz and Vinayak Khajuria in [23] sums up how the circular Hough transform is carried out step by step in case of iris detection with preprocessing steps and the usage of Linear Hough Transform for eyelid detection (List 3.2.3). Figure 3.6 shows the main steps of Circular Hough Transform.

Step 1 Initializing the iris radius

Step 2 Scaling the image

Step 3 Gaussian filtering

Step 4 Creating edge map using canny edge detection

Step 5 Detecting inner boundary using Circular Hough transform

Step 6 Detecting outer boundary inside of located iris using Circular Hough Transform

Step 7 Detecting eyelid using Linear Hough Transform

Step 8 Displaying the segmented image

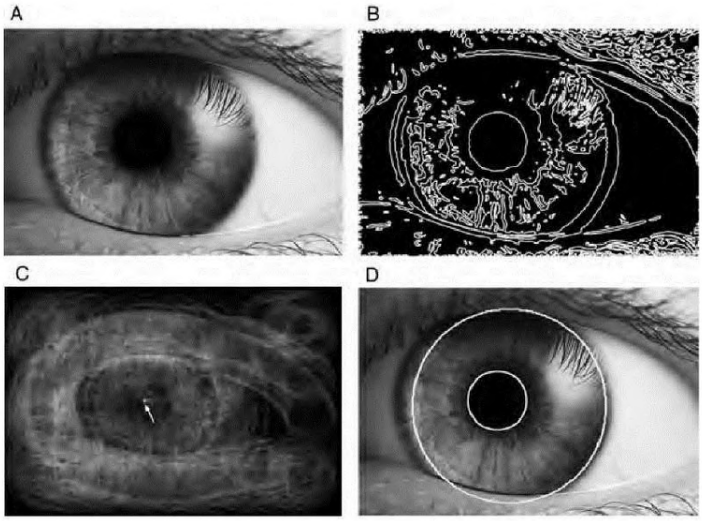


Figure 3.6: Circular Hough transform steps: (A) original image;
(B) edge detection result; (C) accumulator array; (D) detected circle [23]

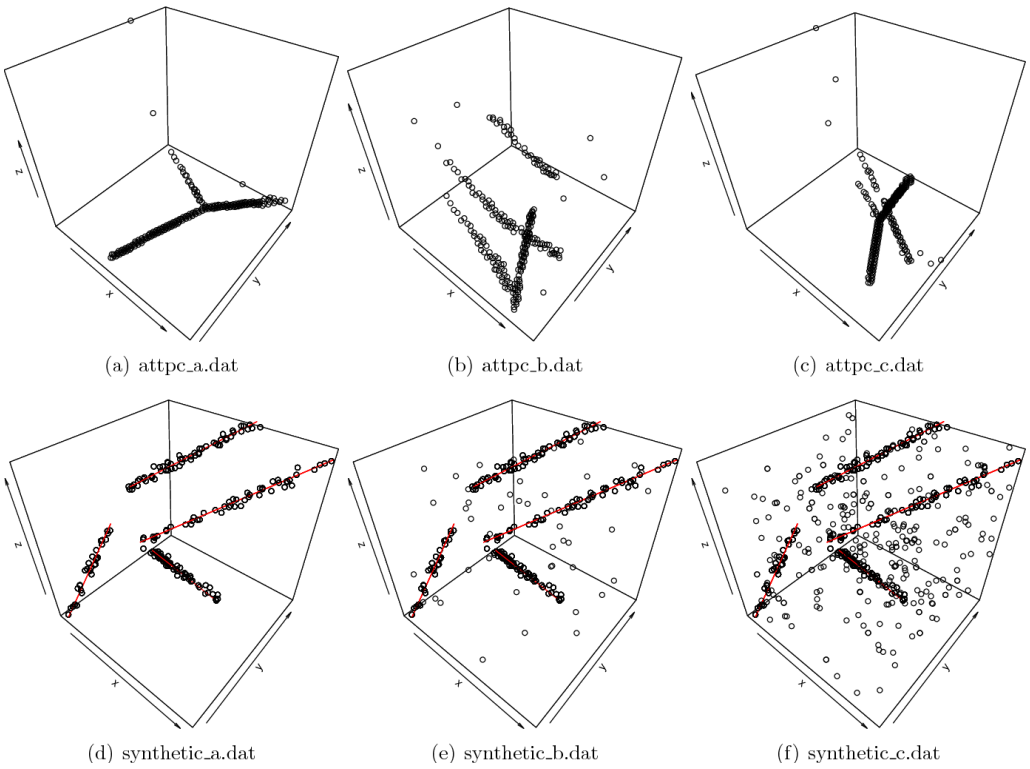
Hough transform for 3D line detection

Christoph Dalitz, Tilman Schramke and Manuel Jeltsch created a method for 3D line fitting using Hough transform. [24] They proposed a new scheme based on Roberts' minimal and optimal line representation to discretize the Hough parameter space in 3D. The discretization uses the tessellation of Platonic solids (in 3D space these are regular, convex polyhedrons). They used the following iterative modification of the transform.

Algorithm 2 Iterative Hough transform

1: Discretization of the parameter space for all lines crossing the point cloud volume.
2: Hough transform of the point cloud X based on the discretization from step 1.
3: Determination of the line parameters corresponding to the highest voted accumulator cell.
4: Finding all points $Y \subseteq X$ close (i.e., distance less than cell width) to the line.
5: Determination of the optimal line going through Y with an orthogonal least squares fit.
6: Finding all points from X close to the fitted line and their removal from X and from the accumulator array.
7: Repetition of steps 2 to 6 until X contains too few points or the specified number of lines has been found.

This algorithm has $\mathcal{O}(2nN_1 + 3n^2/n_{min})$ asymptotic upper bound, where X is the number of points in the dataset, N_1 is the number of directions and $n = |X|$. The algorithm can fail if there is not enough memory for the accumulator array, overflow is generated in and accumulator array counter and if the point cloud has too many identical points. The method works well in case of outliers. Figure 3.7 displays the results of the above introduced 3D Hough transform. The research group published their implementation of the method which was used as a base in this work.



(a) attpc_a.dat (b) attpc_b.dat (c) attpc_c.dat

(d) synthetic_a.dat (e) synthetic_b.dat (f) synthetic_c.dat

Figure 3.7: 3D Hough transform for line detection [24]

3.3 Region growing algorithm

3.3.1 Region growing methods

Region growing algorithms usually used for solving image segmentation problems, since this is the first step of a variety of image analysis and visualization tasks. The algorithms start with a point that meets a detection criterion to grow the point in all directions or a specified direction to extend the region. These procedures usually created for a specific task, thus don't have universal capability. [25]

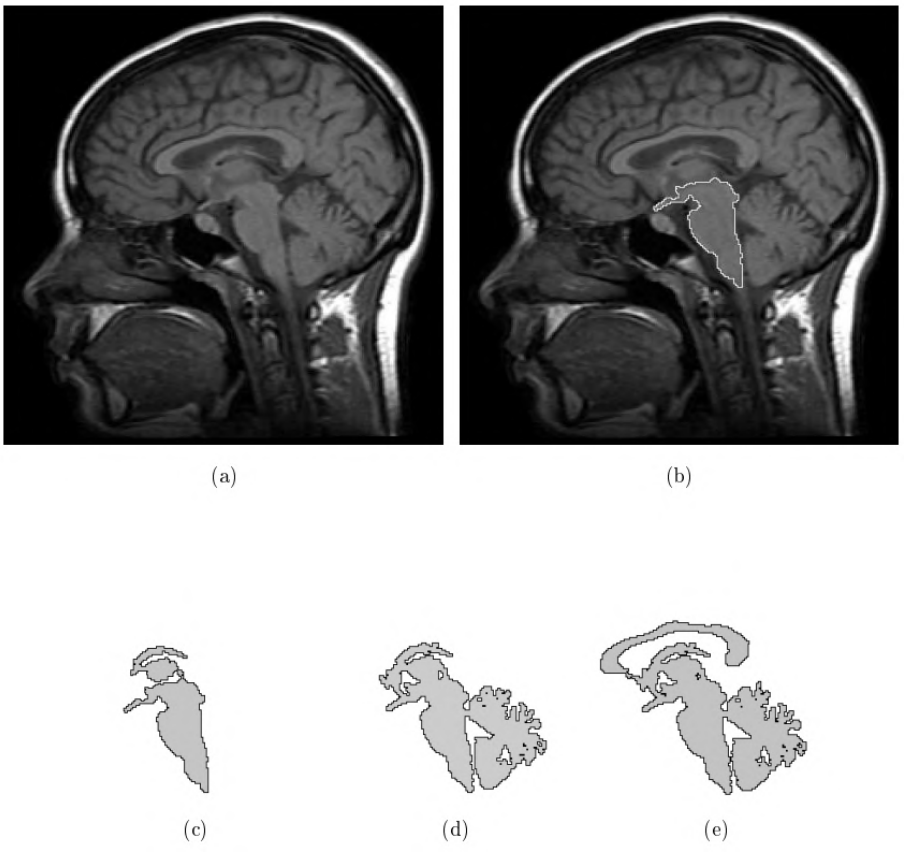


Figure 3.8: (a) Original MRI image. (b) Segmentation result of brain stem. (c), (d) and (e) Segmentation results [25]

3.3.2 The method

The algorithm is based on Automatic railway power line extraction using mobile laser scanning data [15] method which was introduced in Section 2.4.6. Since the paper wasn't detailed enough some steps were changed in this implementation. The original paper assumes that the trajectory is known. In this case we don't have this data, so we assumes that the trajectory is almost parallel with axis Y. Using RANSAC the method finds a line in the seed dataset. With this line and its parameters we can create a transformation which rotate the seed cloud to be parallel with the specified axis. After this a projection is performed, it projects the data onto the given axis. The method creates user defined number of grids, the default is 4 (it means we will have 4 grids both vertically and horizontally), according to the performed tests. Using the grids we can find every cable. Since the seed cloud doesn't contain unnecessary points at this point only three grids aren't empty, thus there's no need to define a parameter which should select some of the grids with the highest element number – as it was required in the original algorithm –. The implemented algorithm can search in both directions, so the seed can be anywhere. For simplicity only the forward case will be considered. The paper calls the used bounding boxes center points seeds, in the upcoming statements the word seed refers to the centers, not the seed cloud. While generating starting seeds we only need a small section of the point cloud. The following method is a self-adaptive region growing algorithm, staring from these center points. The cables are around 0.4 meter wide, this usually holds in Europe. The first step is to create a bounding box around the center point, the starter box length is 0.4 meter. The algorithm adds points to the cable point array according to which points are contained by this box. The method steps forward by using only the last 75% of the points on the specified line. Problems may occur because of supporting structures. If the actual box contains too much points (user parameter, the default is 110, considering the point density in the dataset) we should try and create a new box with only the quarter of the original length. After this step the new result data number can be less than 2 points, which means, that we are in a cross-section, so instead of evaluating the box length reduction, we're deleting surplus points along X axis. An another problem that can occur along support structures is that the method can stuck. For solve this problem, if the actual

seed point's y value won't grow, we skip forward, by increasing the y value of the center point. In this case we can still have problems with cross-sections, so it is necessary to check if the updated bounding box contains any points. If not, the method doesn't move forward, instead filters out every point which has x value 0.5 meter further from the closest 100 points' average x value, after this the process continues as stated before. All steps are carried out for all of the power lines. The extracted result point clouds are shown in Figures 3.9 and 3.10. The method is summarized in Algorithms 3 and 4.

Algorithm 3 Self-adaptive region growing method, first step

Funct Find seeds(*gridCount*)

1: Find a line in the seed point cloud using RANSAC
2: Rotate the seed point cloud to be parallel with y axis, using the parameters of the found line
3: Project the seed dataset onto y axis
4: Create grids with given number, *gridCount*
5: Select the grids which are not empty
6: Calculate the center of the points which are contained by the grids

Algorithm 4 Self-adaptive region growing method, second step

Funct Extract power lines($boxLength, maxPointNumberPerBox$)

1: Select an initial seed point
2: Create initial bounding box with given size, $boxLength$
3: **while** Max point y value of cable smaller than max y value of point cloud **do**
4: Select points with biggest y value from the given bounding box
5: Create new bounding box around the center of the selected points
6: **if** The y value of the new center is $\geq$ the center of the old bounding box **then**
7: Add $boxLength$ to the y value of actual seed point
8: **end if**
9: **if** The actual bounding box is empty **then**
10: Step back by decreasing the y value of the seed point by $boxLength$
11: Calculate the last 100 cable points' x average x value
12: If a point is further by 0.5 meter than the average (on x axis), remove this point
13: **end if**
14: **if** The number of points are too much, according to parameter $maxPointNumberPerBox$ **then**
15: Reduce $boxLength$ by its quarter
16: Find points which are inside the reduced bounding box
17: **if** The new number of points are smaller than 2 **then**
18: Use the new bounding box
19: **else**
20: Remove points which have the biggest x values from the original box
21: **end if**
22: **end if**
23: Add the content of the bounding box to the cable point array
24: **end while**
25: Create grids with given size
26: Select the grids which are not empty
27: Calculate the center of the points which are contained by the grids

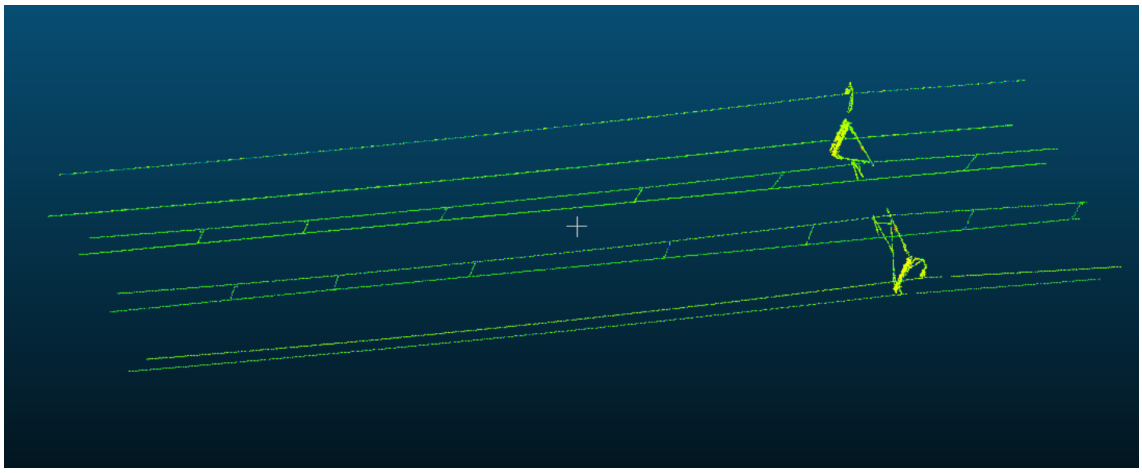


Figure 3.9: Result of significant point cloud segmentation in the original method [15]

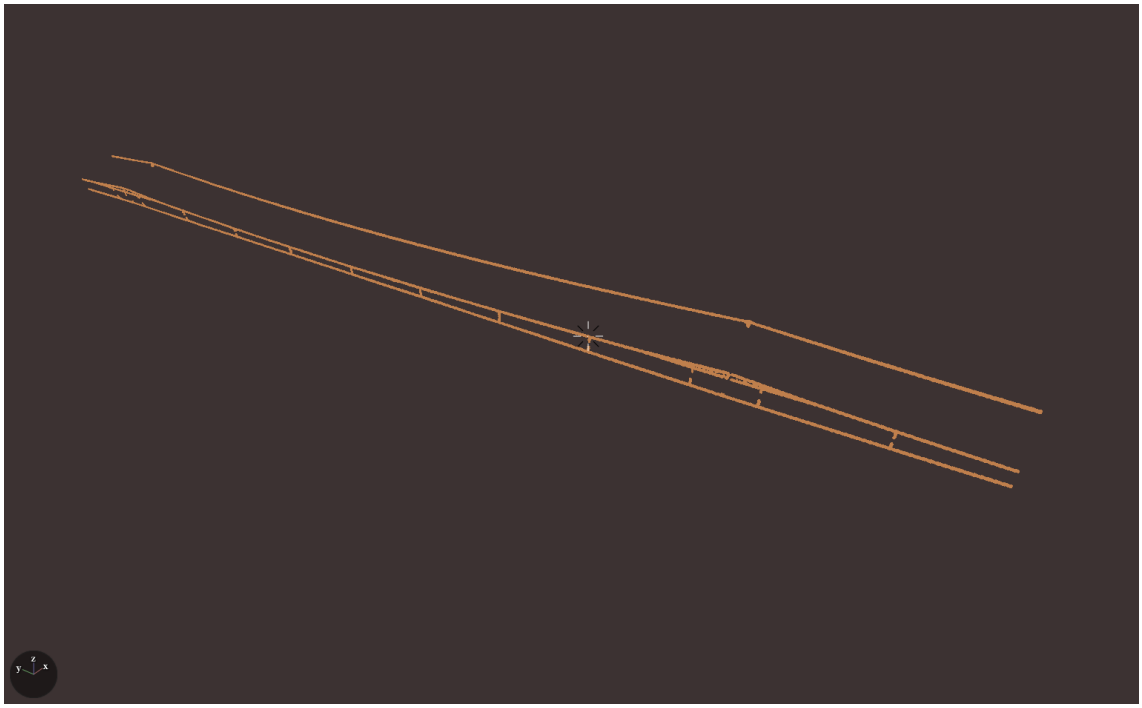


Figure 3.10: Region growing method result point cloud

Chapter 4

Results

4.1 System specification

The methods were implemented using C++ 11 and the PCL library. The used PCL version was 1.8.1, g++ version was 7.4.0 on elementary OS 5.1 Hera (a Debian-based OS). The tests were performed on a 100 meter segment of the point cloud. The validation performance is done using an Intel Core i7-8550U CPU at 1.8 GHz and 16 GB of RAM.

4.2 Implementation

The implementation was made as part of an already implemented framework called *railroad*, described in [26].
The source code of the program is an attachment of the thesis; but can also by viewed online at the `https://github.com/mcserep/railroad` GitHub repository.

4.3 Test analysis

The analysis represents the runtime and the accuracy. The power lines were successfully detected in both cases. The runtime of the RANSAC algorithm depends on the number of iterations, which can differ because of the random point selection, thus the data in Table 4.1 represents the average runtime. These false positive and

negative percentages are the proportions of the false negative and positive values in the resulting point cloud. The results are show, that the most accurate method is RANSAC, also, it has the lowest runtime. Although the region growing algorithm is the most stable, since it can perform well even in longer areas. Figure 4.1, 4.2 and 4.3 visualizes the results. Figure 4.4 and 4.5 shows the outcome for a 300 meter long area. Table 4.2 shows the runtime for the longer segment.

Method	Runtime (sec)	Filtered points	Result size	False negative	False positive
RANSAC	0.18	7283701	32597	0 %	23.9194 %
Hough transform	2.18	7278007	38291	0 %	35.2244 %
Region growing	2.66	7250572	24121	3.00403 %	0.327339 %

Table 4.1: Results of the 100 meter segment tests

Method	Runtime (sec)	Filtered points	Result size
RANSAC	0.678	22577523	104366
Region growing	48.19	22609865	72024

Table 4.2: Runtime for the 300 meter segment

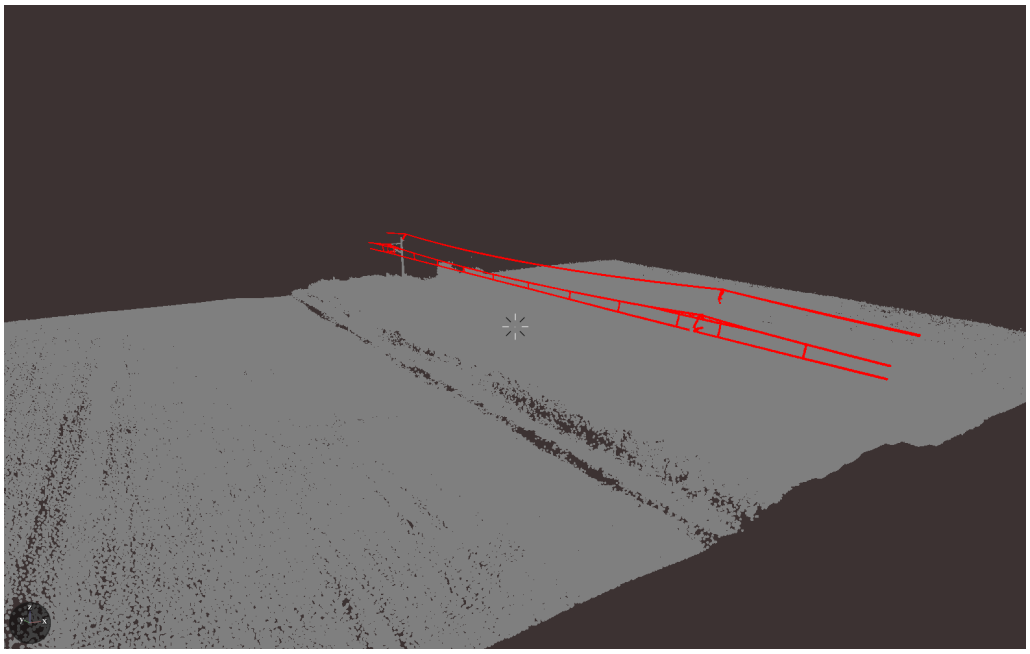


Figure 4.1: The result of the RANSAC algorithm

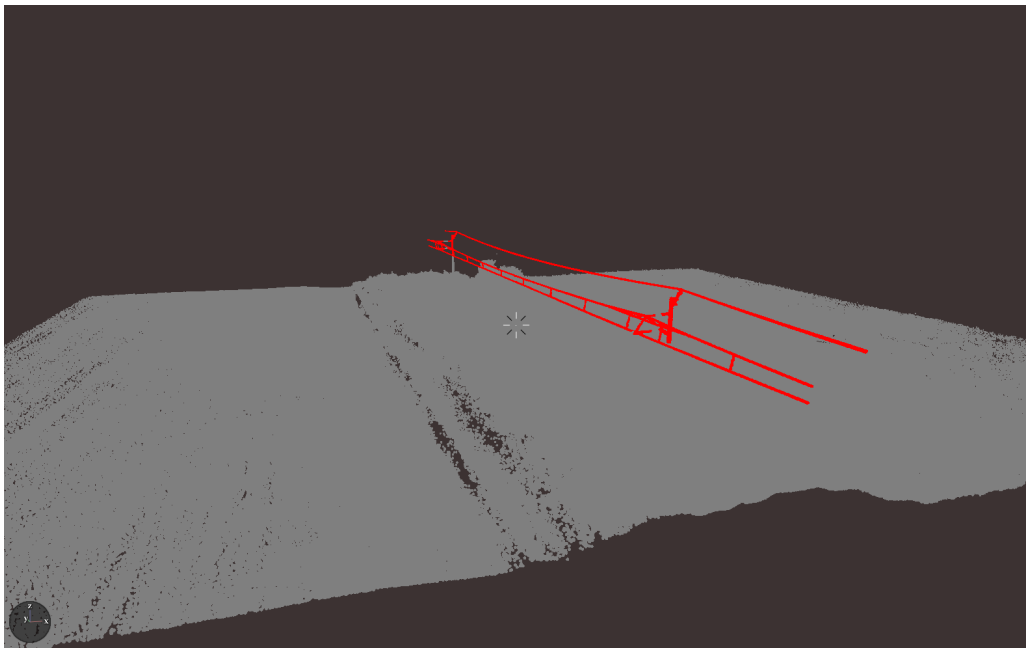


Figure 4.2: The result of the Hough transform

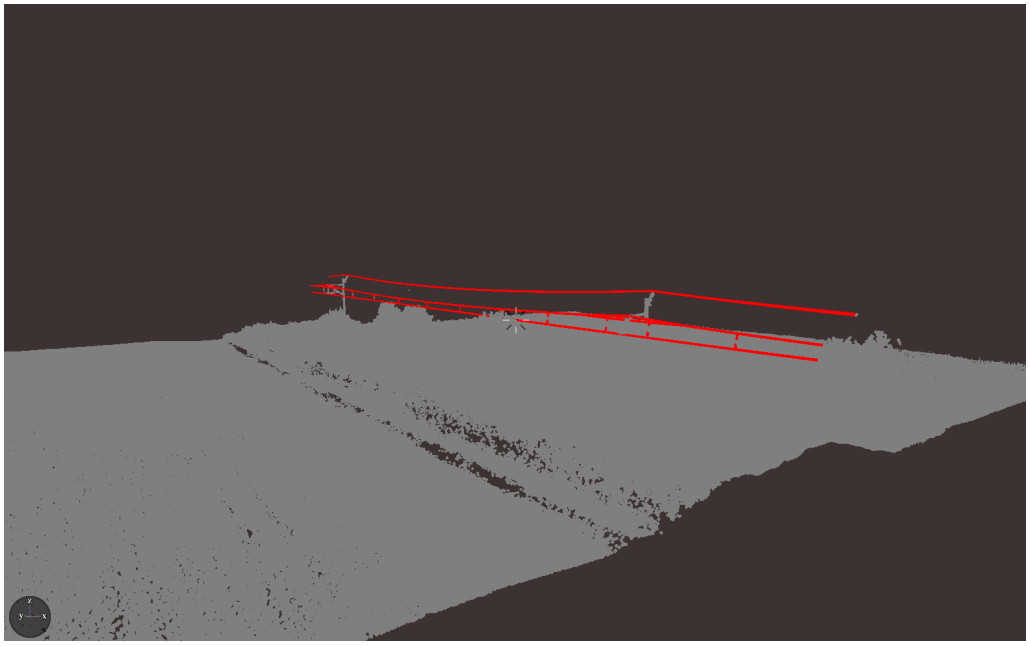


Figure 4.3: The result of the region growing algorithm

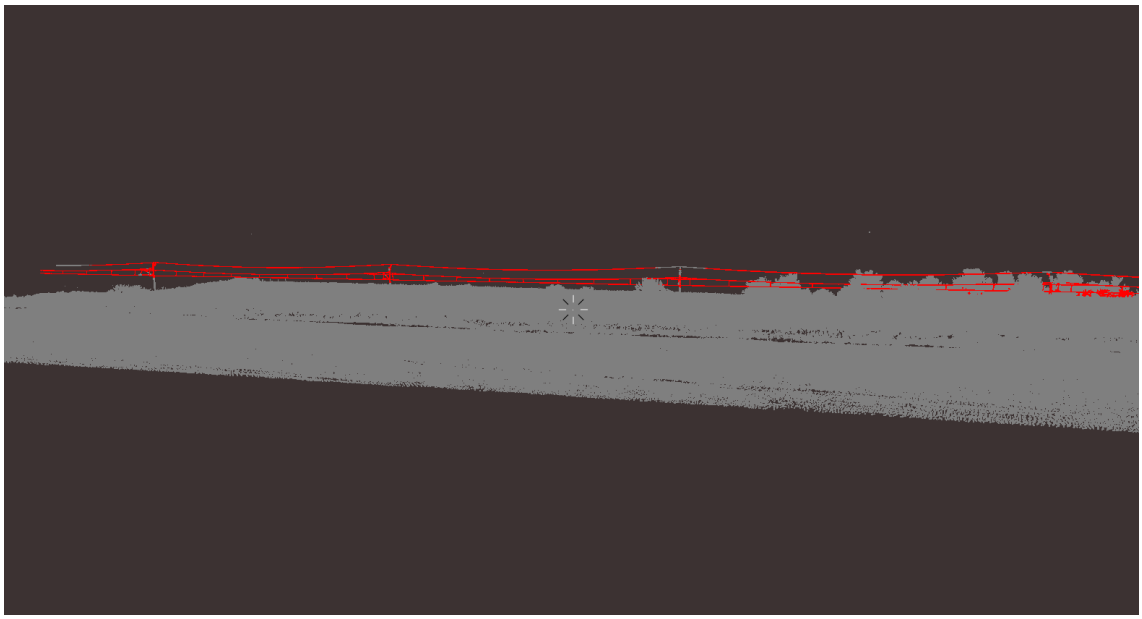


Figure 4.4: The result of the RANSAC algorithm

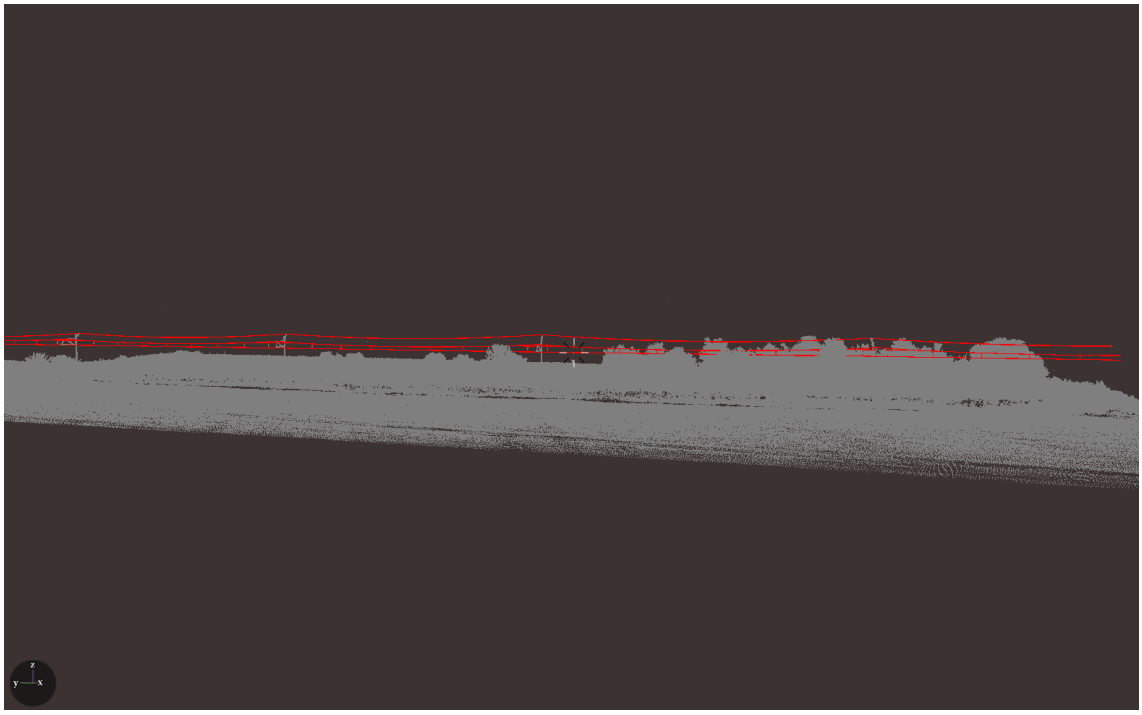


Figure 4.5: Region growing algorithm result

Chapter 5

Conclusion and discussion

My thesis goal was to examine that if we are able to create and implement efficient, stable and fast methods with the help of a seed point cloud. During this work the focus was on three algorithms: RANSAC, Hough transform and a self-adaptive region growing method. The main conclusion of this work is that we can create fast and efficient methods using a seed point cloud for the problem of finding power cables in a dataset. The seed can help in preprocessing and can be used for region growing algorithms. Overall, the most stable method is the region growing algorithm. In case of RANSAC the result are worse when we're using a more than 100m long sections, since it – as it was stated in Section 3.3.2 – assumes that all of the measurements have the same influence, thus if we have long areas with bushes and trees it can misbehave and find a line with lots of inliers which isn't belong to a cable. The 3D Hough transform fails to perform well when it is used for long sections. The overall accuracy in case of false negatives for RANSAC 100%, for Hough transform 100%, for the region growing method 97%. In the same order the accuracy of false positives are 76%, 65% and 99,7%. According to these data the most promising is the region growing method, so in the future we should work on that to make it better and more optimal. The region growing results show that the proportion of falsely recognized points are only around 3,3%. RANSAC has the smallest runtime, it's significantly faster, than the region growing method for the 300 meter long segment.

5.1 Future work

The methods work in straight rail track segments, thus in the future it would be subservient to carry out a preprocessing algorithm which can detect curves and cut the point cloud along these regions. In this case the algorithms should run in parallel on more threads. This procedure also could eliminate memory allocation problems.

All of the result sets contains some false positives. A postprocessing method should be introduced, which can deal with this issue.

Since it is assumed which axes are used during preprocessing and in the methods, it would be necessary to introduce user parameters, which can control this behaviour and make the algorithms more universal.

Acknowledgement EFOP-3.6.3-VEKOP-16-2017-00001: The research behind this master thesis is supported by the Hungarian Government and co-financed by the European Social Fund. Special thanks to Máté Cserép who helped this work as a supervisor with his professional knowledge.

Bibliography

[1] Juntao Yang and Zhizhong Kang. "Voxel-Based Extraction of Transmission Lines From Airborne LiDAR Point Cloud Data". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 11 (Oct. 2018), pp. 3892–3904. DOI: 10.1109/JSTARS.2018.2869542.

[2] B. Lohani. "Introducing airborne LiDAR to surveying students". In: (1999).

[3] Erik Heinz et al. "Analysis of Different Reference Plane Setups for the Calibration of a Mobile Laser Scanning System". In: Apr. 2017, pp. 131–146.

[4] Ian Dowman. "Integration of LiDAR and IFSAR for mapping". In: *International Archives of Photogrammetry and Remote Sensing* 35 (Jan. 2004).

[5] Binbin Xiang et al. "Segmentation-based classification for 3D point clouds in the road environment". In: *International Journal of Remote Sensing* 39.19 (2018), pp. 6182–6212. DOI: 10.1080/01431161.2018.1455235. eprint: https://doi.org/10.1080/01431161.2018.1455235. URL: https://doi.org/10.1080/01431161.2018.1455235.

[6] S..J. Walsh et al. "9.13 - A Beach Vulnerability Framework for the Galapagos Islands: Fusion of WorldView 2 Imagery, 3-D Laser Scanner Data, and Unmanned Aerial Vehicles". In: *Comprehensive Remote Sensing*. Ed. by Shunlin Liang. Oxford: Elsevier, 2018, pp. 159 –176. ISBN: 978-0-12-803221-3. DOI: https://doi.org/10.1016/B978-0-12-409548-9.10438-5. URL: http://www.sciencedirect.com/science/article/pii/B9780124095489104385.

[7] Diana Krupnik et al. "Study of Upper Albian rudist buildups in the Edwards Formation using ground-based hyperspectral imaging and terrestrial laser

scanning". In: *Sedimentary Geology* 345 (Sept. 2016), 154–167. DOI: 10.1016/j.sedgeo.2016.09.008.

[8] ASPRS Board of Directors. *LAS Specification*. Tech. rep. 1.4 – R13. American Society for Photogrammetry and Remote Sensing, July 2013. URL: https://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf.

[9] Martin Isenburg. "LASzip: lossless compression of LiDAR data". In: *Photogrammetric engineering and remote sensing* 79.2 (2013), pp. 209–217.

[10] Hui Huang et al. "L1-medial skeleton of point cloud". In: *ACM Transactions on Graphics* 32.4 (July 2013).

[11] Bertrand Kerautret et al. "3D Geometric Analysis of Tubular Objects Based on Surface Normal Accumulation". In: June 2015. DOI: 10.1007/978-3-319-23231-7_29.

[12] Thorsten Schmidt et al. "Modeling of Sparsely Sampled Tubular Surfaces Using Coupled Curves". In: *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*. Springer. July 2012, pp. 83–92. DOI: 10.1007/978-3-642-32717-9_9.

[13] Rahul Kumar et al. "Blood Vessel Segmentation and Centerline Tracking Using Local Structure Analysis". In: Jan. 2015, pp. 122–125.

[14] Elzbieta Pastucha. "Catenary System Detection, Localization and Classification Using Mobile Scanning Data". In: *Remote Sensing* 8 (Sept. 2016), p. 801. DOI: 10.3390/rs8100801.

[15] Shanxin Zhang et al. "AUTOMATIC RAILWAY POWER LINE EXTRACTION USING MOBILE LASER SCANNING DATA". In: vol. XLI-B5. June 2016, pp. 615–619. DOI: 10.5194/isprs-archives-XLI-B5-615-2016.

[16] Mostafa Arastounia. "Automated Recognition of Railroad Infrastructure in Rural Areas from LIDAR Data". In: *Remote Sensing* 7 (Nov. 2015), pp. 14916–14938. DOI: 10.3390/rs71114916.

[17] Joris Ravaglia, Alexandra Bac, and Richard Fourner. "Extraction of tubular shapes from dense point clouds and application to tree reconstruction from laser scanned data". In: *Computers & Graphics* 66 (June 2017). DOI: 10.1016/j.cag.2017.05.016.

[18] Ana Sánchez Rodríguez et al. "Automated Inspection of Railway Tunnels' Power Line Using LiDAR Point Clouds". In: *Remote Sensing* 11 (Nov. 2019), p. 2567. DOI: 10.3390/rs11212567.

[19] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[20] Marta Martinez-Camara et al. "A robust method for inverse transport modeling of atmospheric emissions using blind outlier detection". In: *Geoscientific Model Development Discussions* 7 (Oct. 2014), pp. 3193–3217. DOI: 10.5194/gmdd-7-3193-2014.

[21] Konstantinos G. Derpanis. "Overview of the RANSAC Algorithm". In: 2005.

[22] Richard O. Duda and Peter E. Hart. "Use of the Hough Transformation to Detect Lines and Curves in Pictures". In: *Commun. ACM* 15.1 (Jan. 1972), 11–15. ISSN: 0001-0782. DOI: 10.1145/361237.361242. URL: https://doi.org/10.1145/361237.361242.

[23] Neeti Taneja, Er. Mohammad Shabaz, and Vinayak Khajuria. "Iris Detection Using Segmentation Techniques". In: *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING* 6 (Sept. 2018), pp. 442–444. DOI: 10.26438/ijcse/v6i9.442444.

[24] Christoph Dalitz, Tilman Schramke, and Manuel Jeltsch. "Iterative Hough Transform for Line Detection in 3D Point Clouds". In: *Image Processing On Line* 7 (2017), pp. 184–196. DOI: 10.5201/ipol.2017.208.

[25] S. Hojjat and J Kittler. "Region Growing: A New Approach". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 7 (Feb. 1998), pp. 1079–84. DOI: 10.1109/83.701170.

[26] Máté Cserép, Péter Hudoba, and Zoltán Vincellér. "Robust Railroad Cable Detection in Rural Areas from MLS Point Clouds". In: July 2018. DOI: 10.7275/z46z-xh51.

List of Figures