EFFECTIVE RAILROAD FRAGMENTATION AND INFRASTRUCTURE RECOGNITION BASED ON DENSE LIDAR POINT CLOUDS

Máté Cserép[a,*], Adalbert Demján [a], Friderika Mayer [a], Balázs Tábori [a], Péter Hudoba [b]

[a] Department of Software Technology and Methodology, ELTE Eötvös Loránd University, Budapest, Hungary
[b] Department of Computer Algebra, ELTE Eötvös Loránd University, Budapest, Hungary
[*] mcserep@inf.elte.hu

KEY WORDS: LiDAR, point clouds, railroad, object recognition, segmentation, fragmentation

ABSTRACT:

Monitoring the condition of railway infrastructure is essential for maintaining safety standards and preventing accidents. The regular inspections required for this task are still typically carried out in many countries with costly and time-consuming on-site human inspections. LiDAR point clouds collected by mobile laser scanning (MLS) already proved to be suitable for recognizing important railroad infrastructure elements, such as cables and the rail tracks. However, the computational requirement for processing large data sets like these often extremely dense point clouds is still a challenge nowadays, resulting in longer execution time than practically applicable. In our research, we have implemented and comparatively analyzed railroad fragmentation and object segmentation algorithms with the focus on robustness and high effectiveness: prioritizing automation and prerequisite reduction (e.g. the spatial relationship between the position of the railway track and the overhead contact line). These aspects also enable the easy parallelization for the processing of larger railroad segments.

1. INTRODUCTION

Railroad transportation is one of the most popular methods both for passenger traveling and cargo shipment. Public railroad transportation provides annually around 8 billion unlinked passenger trips and over 400 billion passenger-kilometers in the EU (EuroStat, 2021b), together with around 390 billion tonne-kilometers in railway freight transport (EuroStat, 2021a). Regular monitoring and surveillance of the railroad infrastructure is crucial for safety concerns and accident prevention. This task is still carried out by expensive and time consuming manual visual inspections in many countries nowadays.

Automated detection of railroad infrastructure has been addressed based on LiDAR point clouds acquired both by mobile terrestrial laser scanning (MLS) (Arastounia, 2015), (Jwa and Sonh, 2015) or low-altitude aerial laser scanning (ALS), obtained usually from helicopters (Zhu and Hyyppa, 2014), (Jeon and Kim, 2019). Beside the generalized approaches, specialized algorithms on some characteristic of the surrounding environment have also been developed, optimizing their results in rural environments (Arastounia, 2015) (Cserép et al., 2018) or in urban environments (Arastounia and Oude Elberink, 2016). Either the powerline cable or the rail recognition in these studies depends on the previously calculated results, their position. Auxiliary data sources, either laser pulse return intensity (Yang and Fang, 2014) or high resolution ortho-imagery for RGB data could also be involved (Neubert et al., 2008), (Beger et al., 2011).

These state of the art methods can provide precise results, but their evaluation time is usually considerably high (magnitude of 5-10 minutes) even for relatively small railroad segments of a few hundred meters due to the heavy computational load. Developing concurrent algorithms can boost evaluation time resulting from the extensive size of the datasets (Arastounia, 2017).

[*] Corresponding author

Our research compares existing algorithms and contributes to the development and comparison of automated data-driven methods based on LiDAR point clouds for railroad fragmentation and infrastructure recognition. The proposed solution of our study focuses on the robustness and automation of the algorithm, through minimizing the assumptions (spatial relations between the cables and rail, flatness of the ground, known trajectory of the train and thus the rail tracks, etc.) Results are evaluated by their computational efficiency and the accuracy of the segmented objects.

The rest of the paper is organized as follows: Section 2 describes the utilized sample LiDAR datasets for the paper. Then Section 3 introduces the proposed methodology of our research and explains the processing steps. Section 4 presents the visual and numerical results and also contains a verification against a manually annotated point cloud. Finally, Section 5 concludes the paper and discusses the future work.

2. DATASET

The sample LiDAR datasets used in this study were collected by the Hungarian State Railways with a *Riegl VMX-450* high density mobile mapping system (MMS) mounted on a railroad vehicle (shown in Figure 1), operating at 60 km/h. The imaging unit was composed of two 360° field of view laser scanners and two high-resolution cameras. The navigation unit consisted of an integrated global navigation satellite system (GNSS) and an inertial navigation system (INS). The sensor was capable of recording 1.1 million points / sec with an average 3 dimensional range precision of 3 mm and a maximum threshold of 7 mm. Average positional accuracy was 3 cm with a maximum threshold of 5 cm. The acquired point clouds contain the georeferenced[1] spatial information (3D coordinates) with intensity and RGB data attached to the points.

[1] In the Hungarian national spatial reference system EPSG:23700.

Figure 1. For technical reasons, the *Riegl VMX-450* MMS sensor was mounted on a car, which was placed on a carriage.

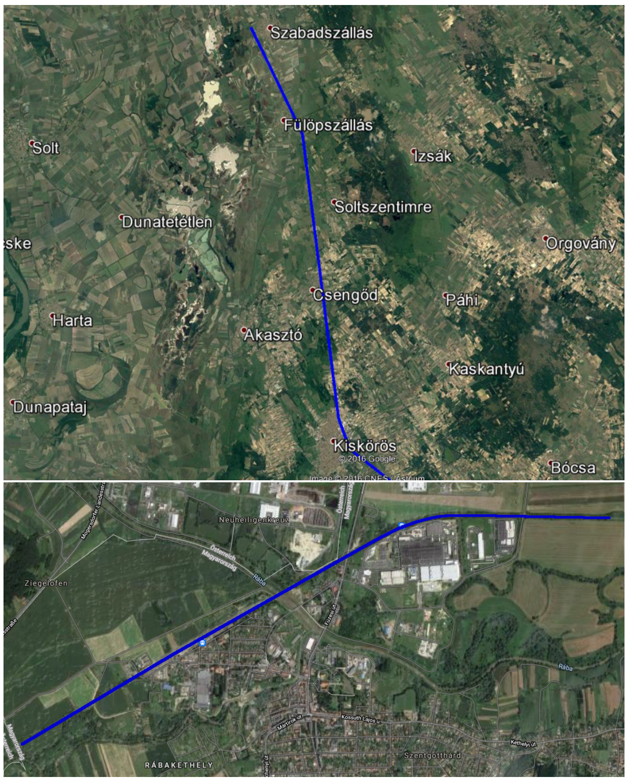

Figure 2. Satellite view of the sample datasets.

Two datasets from different topographical regions of Hungary were selected and used in our research. The satellite view of the locations are depicted in Figure 2.

Dataset 1 is the *Szabadszállás - Kiskőrös* dataset, which covers an approximately 29 km long and 130 m wide rural railroad segment in Southern-Central Hungary and contains ca. $2.5 * 10^9$ points. This area is generally flat with minimal to no slopes on the rail tracks.

Dataset 2 is the *Szentgotthárd neighborhood* dataset, which covers an approximately 5 km long and 90 m wide, partially rural, partially suburban railroad segment in Western-Hungary and contains ca. $0.8 * 10^9$ points. Here, at the foothills of the Alps, topography is more varied and the sample contains slopes.

The complete datasets used in this research are proprietary, but the selected segments used for results and verification are made publicly accessible in the *Data availability* section at the end of the paper.

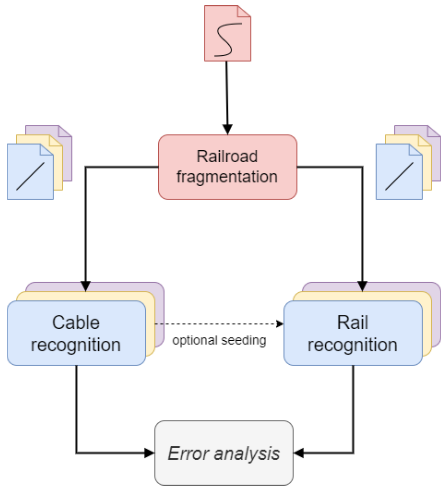

Figure 3. Workflow diagram of the processing steps.

3. METHODOLOGY

The proposed methodology of our research contains of 3 major processing steps: i) *railroad fragmentation* receives a single large input point cloud and fragments it at the curves of the rail track. Hence the later processing steps, ii) *cable recognition* and iii) *rail recognition* will receive multiple smaller inputs, containing a mostly straight segment of the railroad. Cable and rail recognition can be evaluated independently and can be optimized to be executed parallelly. When required by the applied specific algorithm, cable recognition might depend on the result of rail recognition (or vice versa). This optional dependency disables the direct parallel execution of cable and rail detection algorithms for the same area. However, in case of a large amount of input fragments, where the complete dataset cannot be analyzed at once due to its size, this will not hinder the parallelization of the entire process. A possible follow-up, iv) processing step is the *error analysis* of the railroad infrastructure, which will not be addressed in detail in this paper. The described workflow of the methodology is depicted in Figure 3.

The following subsections 3.1, 3.2 and 3.3 will introduce these processing steps.

3.1 Railroad fragmentation

The fragmentation consists of the following parts:

1. A 2D projection of the input point cloud is generated. This 2D digital elevation model (DEM) is constructed from the point cloud along the Z axis, however instead of the usual inverse distance weighting (IDW) algorithm, the maximal Z coordinate in each grid cell is used as its value.

2. Vegetation is filtered through contour detection, since it can be a problem at the edge of the railway track: in some cases, the algorithm will not only be inaccurate, but it may even result in false splitting points.

3. The curve of the rail track is detected using one the following methods:

   Contour finding by first performing an Otsu thresholding (Otsu, 1979), followed by the contour finding with Suzuki's algorithm (Suzuki and Abe, 1985).

   Hough transformation (Duda and Hart, 1972) preceded by a Canny-edge detection (Canny, 1986).

Generalized Hough transformation or its Ballard-defined version (Ballard, 1981) to be more specific. It is a modification of the normal Hough transformation so that it can recognize arbitrary shapes. However, this method is not completely automated: while it recognizes the precise occurrence of the searched shape, it is not able to rotate or resize the pattern during the search.

4. Finally, the point cloud is split based on the curve of the trajectory, resulting in one or more output point clouds.

3.2 Cable recognition

There are multiple types of cables to detect above the rail track (contact cables, catenary cables, return current cables), international and national legislation regulating their relative position to each other and to the rail tracks. In our study we aim to detect all kind of cables, but with no expectation to distinguish them. We present 3 algorithms in this subsection we have implemented and compared to achieve this goal.

3.2.1 Search from above with 2D Hough transform The computational load usually grows with the dimension of the space, thus we used an algorithm that achieve point count reduction based on a 2 dimensional projection of the original point cloud (Cserép et al., 2018). Similarly like in Section 3.1, a 2 dimensional DEM is constructed from the point cloud along the z axis, with the maximal z coordinate in each grid cell as its value.

In order to reduce the noise, the projection grid is filtered by clearing all cells that have less than half of the maximum value. Afterwards we run a probabilistic Hough line detection on the projection first with permissive and then with strict parameters. Finally a cleaning phase of the algorithm goes through all the points and counts the cells around the actual cell with a similar value – a difference lower than a small threshold. In case this count falls below a given threshold, the cell must be removed, since on a continuous cable, cells with similar height should be located around it. The disadvantage of this approach would be the incapability to detect cables below each other. To address this issue, after the first run the selected points are removed from the cloud and the algorithm can be evaluated again to find the lower level cables also. Then, the detected cables from consecutive runs can be merged into a single result set.

Figure 4 shows the main steps of the algorithm. In the first column the first run of the inner algorithm is displayed, and can be observed how the line detection initially finds the cables and the trees also, but then the cleaning step removes the false positive parts. Since our sample datasets contained three cables (with two below each other), the second run of the algorithm was deemed necessary. The second column of the subimages presents these results and how the additional cable was located correctly.

3.2.2 Hough transform for 3D line detection This approach is based on the work of Dalitz and his colleagues (Dalitz et al., 2017). They proposed a new scheme based on Roberts' minimal and optimal line representation (Roberts, 1988) 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. The method works well in case of outliers.

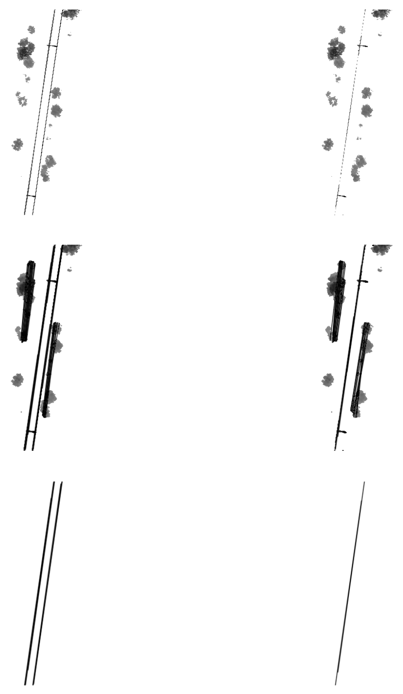
Figure 4. Mid-steps of the 2D Hough transform method.

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.

3.2.3 Region growing algorithm 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 (Hojjat and Kittler, 1998).

The region growing approach is based on Zhang's and his colleagues method (Zhang et al., 2016). The original paper assumes that the trajectory of the train – and thus the rail track and the cables – are known. Since this information is not necessarily provided (e.g. for airborne laser scanning), we replaced this information with a small seed point cloud of the power-line cable as more robust solution, from which the trajectory can be calculated at the beginning of the algorithm with the RANSAC algorithm (Fischler and Bolles, 1981). Since the paper was not detailed enough some steps were changed in our implementation. Our version of the algorithm is summarized in Algorithms 1 and 2.

Algorithm 1 Self-adaptive region growing method, step 1

```
Func 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 contained by the grids
```

Algorithm 2 Self-adaptive region growing method, step 2

```
Func Extract cables ($boxLength, maxPointsPerBox$)
 1: Select an initial seed point
 2: Create initial bounding box with size $boxLength$
 3: while Max Y value of cable $<$ max Y value of point cloud
    do
 4:   Select points with biggest Y value from bounding box
 5:   Create new bounding box around the center of these
      selected points
 6:   if Y value of new center is $\geq$ center of 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:     Decrease Y value of the seed point by $boxLength$
11:     Calculate average of X last 100 cable points
12:     If a point is further by 0.5 meter than the average
        (on X axis), remove this point
13:   end if
14:   if number of points $>$ $maxPointsPerBox$ then
15:     Reduce $boxLength$ by its quarter
16:     Find points which are inside the reduced bounding box
17:     if New number of points $<$ 2 then
18:       Use the new bounding box
19:     else
20:       Remove points with biggest X values from original
          bounding box
21:     end if
22:   end if
23:   Add 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 contained by the grids
```

3.3 Rail recognition

Our solution is an adapted and optimized version of Arastounia's proposed algorithm (Arastounia, 2017). The original algorithm assumed that the trackbed is mainly flat, with very little variance, which we found not to be the case in our datasets. The developed algorithm was enhanced with proper slope detection and handling. The algorithm consists of three main parts.

1. **Locating the trackbed within a small subset of the data**

   (a) First, the railway direction axis and the start coordinate are computed. The initial step of the algorithm requires to cut out a small portion of the dataset, in which we detect the rail pairs. The problem emerges, that without directional data – which is not necessarily at our disposal –, it would not be defined where to cut the dataset.

   (b) A course classification on a subset of the cloud is performed based on the heights of the points in the cloud portion. In a railway environment, the object with the most points in it should always be the trackbed, so the height of the trackbed is determined by searching for the most common height in our subset within a tolerance threshold of $0.75m$.

2. **Detecting the rail pairs in that subset**

   (a) Candidate seed points for the rails are selected. Since rail tracks by definition are narrow and relatively high objects, our aim is to locate points the trackbed, which are outliers in their respective local neighborhoods. Given p is point of the trackbed, this task can be achieved with the following algorithm:

      i. Calculate p's local neighborhood, N_p.

      ii. Calculate N_p's covariance matrix, C.

      iii. Apply eigendecomposition to C.

      iv. Classifying candidate rail seed points. The smallest eigenvalue of a local neighborhood without a rail piece should be below a low threshold close to zero, as the trackbed is usually constructed to have the smallest height variation possible in the longitudinal direction due to safety regulations.

   (b) Lines are detected with 2D Hough transformation. In our algorithm, first the 3D point cloud of candidate rail seed points are converted into a 2D image. For this purpose we use the projection filter introduced and implemented in our previous work (Cserép et al., 2018). Since the Hough line transformation is dependent on the threshold given, there is a high chance that the same threshold will not provide appropriate results for two different datasets. To resolve this potential issue, the developed algorithm works as follows:

      i. Set the threshold to a high number.

      ii. Run the Hough transformation on the image.

      iii. If the Hough transformation did not give at least two lines, lower the threshold.

      iv. Repeat steps $ii.$ and $iii.$ until at least two lines are found or the threshold reaches zero.

   When the Hough transform was executed successfully we now convert the 2D image back to 3D.

   (c) Rail pairs can be recognized through their matching direction and the their predefined distance from each other, called the track gauge. Let d_1 and d_2 be the direction vectors of the lines calculated from the start and end points given by the Hough transform, and the following criteria can be constructed:

$$\angle \vec{d_1}\vec{d_2} \leq 5^\circ \tag{1}$$

$$|DistanceBetweenLines - Gauge| \leq 0.05m \tag{2}$$

3. **Growing the rail pairs throughout the rest of the data.** An iterative algorithm was implemented which fully grows its input rail pair. Each point has to meet two criteria in order to become a candidate rail point. These are the following:

$$|H_{rail} - H_p| \leq 0.05m \tag{3}$$

$$\angle \vec{v}_{raildirection}\vec{v}_p \leq 5^\circ \tag{4}$$

H_{rail} depicts the average height of the current segment we grow, H_p is the height of the point, $v_{raildirection}$ is the direction vector of the rail and v_p is the vector connecting the point to the current rail segment. To grow a rail segment, first we calculate the local neighborhood N_p for each p point with the radius being our grow size, then recognize candidate rail seed points from N_p.

The flowchart in Figure 5 depicts the main steps of the algorithm.

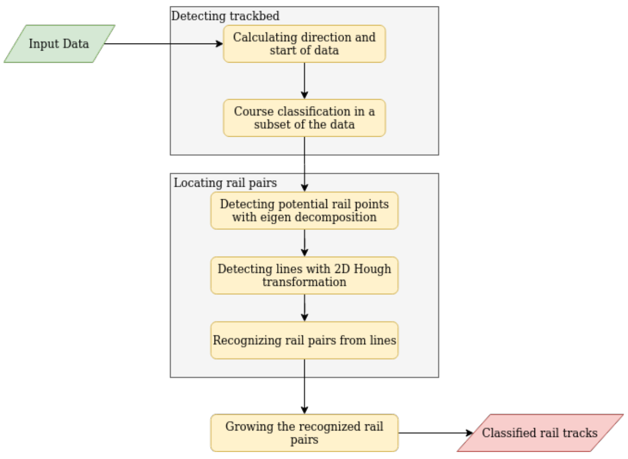
Input Data
Detecting trackbed
Calculating direction and start of data
Course classification in a subset of the data
Locating rail pairs
Detecting potential rail points with eigen decomposition
Detecting lines with 2D Hough transformation
Recognizing rail pairs from lines
Growing the recognized rail pairs
Classified rail tracks

Figure 5. Flowchart of the developed rail recognition algorithm.

4. RESULTS

4.1 Fragmentation results

A curved rail track segment was selected from both sample datasets described in Section 2 to evaluate the railroad fragmentation. These test areas are shown in Figure 6 and 7.

The value of the maximum allowed path curve was 10°, with this value the implemented methods in the framework worked properly. The execution time of each method for a given sample data can be found in Table 1.

The computed splitting locations of the methods are visualized in Figure 8. Each method is marked with a different color as denoted in the caption. In addition, the manually determined locations of the maximum trajectory of 10° were also marked

Method	Dataset 1 area	Dataset 2 area
Contour finding	2 m 52 s	9 m 10 s
Hough trans.	2 m 36 s	8 m 59 s
Gen. Hough trans.	3 m 11 s	13 m 7 s

Table 1. Runtime results of the rail fragmentation with different curve detection methods.

to assess the accuracy of the methods. In both cases, the Hough transformation produced the best splitting locations (closest to the manually determined locations).

4.2 Object recognition results and verification

To evaluate and also verify the result and the accuracy of the object recognition algorithms, we annotated manually both the cables and the rails for a $100m$ long segment consisting of 7,316,298 points from Dataset 1 and tested the algorithms on it. This railroad segment is shown in Figure 9. The following metrics were examined: i) the runtime (without parallel execution), ii) the number of remaining points, iii) the number of false negatives and iv) the number of false positive detections[2]. The results are shown in Table 2.

Among the cable detection algorithms, the *region growing* produced the best accuracy, however it had the benefit of receiving a small seed of the cable as an additional input, as discussed in Section 3.2.3. The *2D Hough transform* algorithm for cable detection and the rail recognition method also produced a fairly good accuracy. The execution time for all evaluated methods are outstanding, since other novel approaches like (Arastounia, 2017) required over 5 minutes to process a $100m$ railroad segment even with concurrency. Unfortunately, concrete source code implementations and tested datasets are rarely made publicly available in the related literature, hindering the opportunity of a more precise comparison of results.

Figure 10 shows the combined visual output of the best cable and rail track detection result.

5. CONCLUSION AND FUTURE WORK

Both MMS and low-altitude ALS point clouds of the railroad infrastructure are typically dense and therefore large point clouds, to guarantee that enough points are located on the important objects (e.g. cables) to recognize them. Therefore the automatic surveillance and monitoring of railroad infrastructure requires not only reliable, but also computationally efficient algorithms.

[2] The percentage of false negative detections were calculated against the size of the verification point cloud, while the percentage of false positive detections were calculated against the number of remaining points.

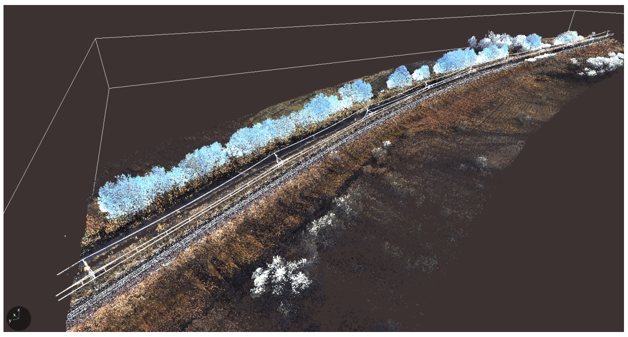

Figure 6. Area from Dataset 1, ca. 600 m, $51.8 * 10^6$ points.

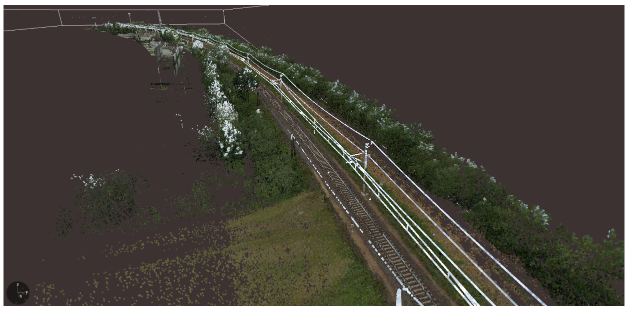

Figure 7. Area from Dataset 2, ca. 1500 m, $58.6 * 10^6$ points.

Algorithm	Object	Runtime	Remaining points	False Negative	False Positive
Hough 2D	cable	3.11 s	23,397	7.77 %	2.24%
Hough 3D	cable	2.38 s	38,291	0%	35.23 %
Region growing	cable	0.16 s	24,121	3.06 %	0.33 %
Rail track	rails	67.18 s	67,368	3.16 %	1.52 %

Table 2. Accuracy of the object recognition algorithms.

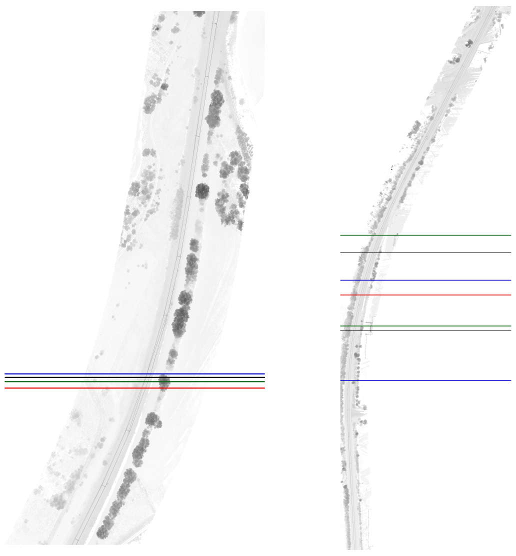
Figure 8. Curve detection result. Blue: contour finding, Green: Hough trans., Red: Gen. Hough trans., Black: manual

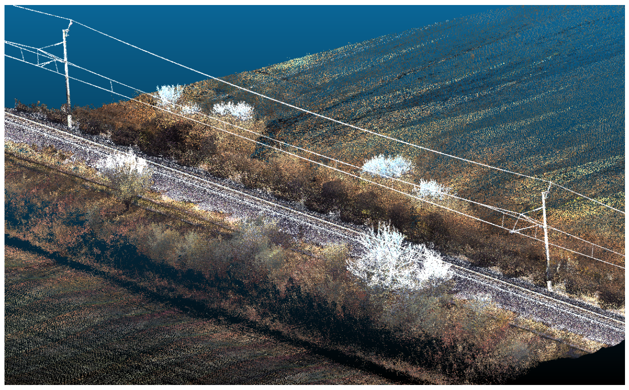
Figure 9. Verification area for cable and rail object recognition.

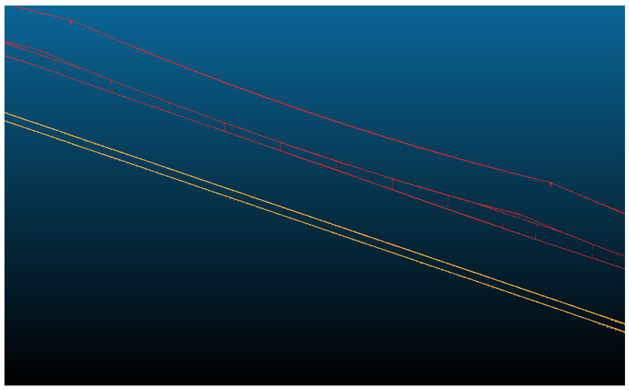
Figure 10. Combined visual result of the cable and rail track detection.

In our research we developed a software framework capable of detecting the most important railroad infrastructure, cables and rails in a large input file through a series of 3 processing steps. First, the trajectory of the rail tracks are detected and the input point cloud is fragmented into parts containing a straight segment of the rail track. By dividing the original input file into multiple fragments, this step already provides a high-level parallelization for future steps. After the fragmentation, the cable and rail recognition steps are performed, which could also be parallelized with each other. The study considered multiple algorithms for these steps and carried out comparative examination on their runtime and accuracy.

In our further work we will focus on the omitted fourth processing step mentioned in Section 3: the automated detection of possible errors and anomalies in the railroad infrastructure and its surrounding. Typical issues could be i) the improper height of overhead contact cable, ii) the horizontal deviation of the cables, iii) the dangerously close vegetation, iv) the deformation of the railway bedding or v) the sinking of the railway sleepers. We also aim to extend the involved algorithms with further available attributes of the points beside their position, like laser pulse return intensity or RGB data.

COMPUTER CODE AVAILABILITY

An open source prototype implementation for the discussed and compared algorithms were carried out in standard C++11 as part of our railroad infrastructure detection framework. Source code is available on GitHub, released under the BSD-3 license, at `https://github.com/mcserep/railroad`. The project was tested to build and run on Ubuntu Linux 20.04 LTS.

DATA AVAILABILITY

Datasets used in Sections 3 and 4 to reproduce results can be found at `http://dx.doi.org/10.17632/ccxpzhx9dj.1`, an open-source online data repository hosted at Mendeley Data (Cserép, 2022).

ACKNOWLEDGMENTS

”Application Domain Specific Highly Reliable IT Solutions” project has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the Thematic Excellence Programme TKP2020-NKA-06 (National Challenges Subprogramme) funding scheme.

REFERENCES

Arastounia, M., 2015. Automated recognition of railroad infrastructure in rural areas from LiDAR data. *Remote Sensing*, 7(11), 14916–14938.

Arastounia, M., 2017. An Enhanced Algorithm for Concurrent Recognition of Rail Tracks and Power Cables from Terrestrial and Airborne LiDAR Point Clouds. *Infrastructures*, 2(2), 8.

Arastounia, M., Oude Elberink, S., 2016. Application of template matching for improving classification of urban railroad point clouds. *Sensors*, 16(12), 2112.

Ballard, D., 1981. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2), 111-122.

Beger, R., Gedrange, C., Hecht, R., Neubert, M., 2011. Data fusion of extremely high resolution aerial imagery and LiDAR data for automated railroad centre line reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6), S40–S51.

Canny, J., 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), 679-698.

Cserép, M., 2022. Hungarian MLS point clouds of railroad environment and annotated ground truth data. Mendeley Data. DOI: 10.17632/ccxpzhx9dj.1.

Cserép, M., Hudoba, P., Vincellér, Z., 2018. Robust railroad cable detection in rural areas from mls point clouds. *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*, 18, 8.

Dalitz, C., Schramke, T., Jeltsch, M., 2017. Iterative Hough Transform for Line Detection in 3D Point Clouds. *Image Processing On Line*, 7, 184–196.

Duda, R. O., Hart, P. E., 1972. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Commun. ACM*, 15(1), 11–15.

EuroStat, 2021a. Railway freight transport statistics. Technical report.

EuroStat, 2021b. Railway passenger transport statistics. Technical report.

Fischler, M. A., Bolles, R. C., 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6), 381–395. https://doi.org/10.1145/358669.358692.

Hojjat, S., Kittler, J., 1998. Region Growing: A New Approach. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, 7, 1079-84.

Jeon, W.-G., Kim, E.-M., 2019. Automated Reconstruction of Railroad Rail Using Helicopter-borne Light Detection and Ranging in a Train Station. *Sensors and Materials*, 31(10), 3289–3302.

Jwa, Y., Sonh, G., 2015. Kalman Filter Based Railway Tracking from Mobile LiDAR Data. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2.

Neubert, M., Hecht, R., Gedrange, C., Trommler, M., Herold, H., Krüger, T., Brimmer, F., 2008. Extraction of railroad objects from very high resolution helicopter-borne LiDAR and ortho-image data. *Int Arch Photogramm Remote Sens Spat Inf Sci*, 38, 25–30.

Otsu, N., 1979. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62-66.

Roberts, K. S., 1988. A new representation for a line. *Proceedings CVPR'88: The Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 635–636.

Suzuki, S., Abe, K., 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1), 32-46.

Yang, B., Fang, L., 2014. Automated extraction of 3-D railway tracks from mobile laser scanning point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(12), 4750–4761.

Zhang, S., Wang, C., Yang, Z., Chen, Y., Li, J., 2016. Automatic railway power line extraction using mobile laser scanning data. XLI-B5, 615–619.

Zhu, L., Hyyppa, J., 2014. The use of airborne and mobile laser scanning for modeling railway environments in 3D. *Remote Sensing*, 6(4), 3075–3100.