Automatic Rail Extraction in Terrestrial and Airborne LiDAR Data

Mustafa Muhamad1  Kresimir Kusevic2  Paul Mrstik2  Michael Greenspan1,3

1School of Computing, Queen’s University, Kingston, Canada
2GeoDigital International Inc., Ottawa, Canada
3Dept. Electrical & Computer Engineering, Queen’s University, Kingston, Canada

Abstract

Datasets of stretches of railway tracks are collected using both Airborne and Terrestrial LiDAR scanners having varying density, resolution and provide different views of the railway track. Manual feature extraction from such datasets is tedious and labour intensive. Therefore, automatic extraction of desired features is highly desirable. In this work, we propose a technique to extract the rails from these two types of datasets. Our rail extraction technique models the railway track as a dynamic system of local pairs of parallel line segments and uses the Kalman filter to predict and monitor the state of the system. The system’s state is composed of the two centroids of the parallel line segments as well as their common direction. Additionally, we augment the Kalman filter process to deal with special cases such as missing railway track segments, sensor noise, and data sparseness. Our technique is effective on both types of data sets as we achieve a precision of 91% and a recall of 99% on the high resolution Terrestrial dataset and a precision of 94% and a recall of 91% on the Airborne dataset.

1. Introduction

Due to technological advances in recent years, 3D data acquisition of large environments such as urban, rural and railway track scenes has become more popular and less expensive. At the same time, automatic analysis of this data has become more attractive due to the resources and person hours that it can save. In addition, future applications such as autonomously driving cars or trains can benefit from a full object recognition pipeline of 3D data [7] and the first step of this pipeline is the extraction of the driving pathway. On the railway track, other applications such as geo-tagging specific features of the track such as switches, signals, and railway track crossings require the extraction of rails as the initial step. Since these datasets tend to capture very long stretches of railway track the ability to be able to extract the rails automatically (or with minimal human interaction) becomes very important.

Two popular ways of collecting 3D data is using Airborne LiDAR [1] [12] [4] [8] and Mobile Terrestrial LiDAR [10] [6] [3]. Airborne LiDAR is collected using a laser scanner which is mounted on an airplane. The mobile scanner platform velocity is very high while the data acquisition rate is slow resulting in sparse scans of the top view of the environment. Terrestrial LiDAR is collected by having a LiDAR scanner mounted on a vehicle which scans the ground and the right and left directions as the vehicle drives through a city road or on top of railway track. Terrestrial LiDAR has a much slower mobile scanner platform velocity and a faster data acquisition rate resulting in data that is more dense. The data also contains much more detail because the scanner is at a much closer distance to the objects of interest resulting in much less dispersion in the scanner beams and therefore higher resolution. In addition, Airborne data can miss large portions of the scene of interest due to structures such as bridges.

In this work, we propose a technique to automatically extract railway tracks based on the Kalman filter. The technique works by iteratively fitting lines to local portions of the railway track using RANSAC while using the Kalman filter to guide the entire process. The technique is tested on both Airborne and Terrestrial LiDAR data and shown to be effective on both types of data.

2. Related Work

Previous work on automatically extracting structures from 3D LiDAR data focused on a variety of application. One area that is well studied is bare-earth and building structure and feature extraction [13], [1], [12], [10]. Road extraction, which is more related to our work is studied less extensively in [3],[4], [6]. We only found one work which deals with the problem of rail extraction in LiDAR data [8], described below.

Most of the work dealing with bare-earth and building structure extraction deals with Airborne data. As outlined in the review by Vosselman [13], the techniques used include
scan line segmentation, surface growing, and 3D Hough transform to directly extract parametrized shapes such as planes, circles, and cylinders. For example, Alharthy and Bethel [1] use smooth surface fitting to distinguish building rooftops from other extraneous objects such as trees and cars but does not deal with complex roof structures. On the other hand, Verma et al. [12] deals with the problem of detecting and modelling complex roof structures by fitting several simple shapes to complex rooftop structures and using a graph-based approach to cluster the roofs of different buildings. Further segmentation of the building into windows, doors and walls is performed in [10] for a terrestrial LiDAR dataset.

In the work of [4] the authors take advantage of both 3D and intensity data. They perform fully automatic road extraction by performing a point based classification based on height, intensity value, and local point density. This is followed by a connected components stage in which small connected components are removed leaving classified road points. In the work of Boyko and Funkhouser [3] the road network is extracted from a large Terrestrial LiDAR urban dataset. Their system is initialized by manually picking a single point at every road intersection or by mapping a 2D street map to their 3D point cloud. The mapping from 2D to 3D is accomplished by optimizing cardinal splines that ensure continuity and smoothness of the map. Next an attractor map is constructed that gives high values to potential road boundaries (curbs). Finally an active contour (also called a snake) is fitted to the attractor map. The active contour aims to minimize an energy function that ensures the contour respects road boundaries and road shape. Finally, [6] used a Kalman Filtering approach to automatically extract the road given a manually initialized small patch of the road from scans of cities gathered using Terrestrial LiDAR. Our approach for rail extraction is is most similar to this work.

In [8], the authors extract rails by filtering points based on their height and then mapping the point cloud onto a 2D grid. This step is followed by applying both line and circle RANSAC to each cell to find parallel rail segments separated by a fixed distance. Finally, overlapping cells are merged together to reconstruct the railway tracks. The authors claim to have extracted 90% of the rails in their dataset. However their result is based on visual inspection of their output and not on comparison to ground truth.

3. Rail Extraction

We break the problem of extracting a railway track into two stages. The first is initialization and the second is following the railway track which we call railway track growing. In the initialization stage, the goal is to detect a section of a railway track represented as two parallel line segments. Once initialization is accomplished our algorithm grows the extracted initial segments in the direction of the rail. The growing algorithm is inspired by the fact that changes in rail structure parameters such as curvature and elevation happen gradually [9]. Due to this property, an extracted section of railway track can be used to estimate and therefore extract the next section of railway track. In order to accomplish this goal, the railway track is modelled as a discrete dynamic process that uses a Kalman filter to estimate the parameters of the next rail section based on the previous rail section.

3.1. Rail Track Initialization

In order to distinguish railway tracks from other ground points a ground extraction step is first performed. Due to the fact that the ground is not flat, a simple passthrough filter applied to the $z$-coordinate of the points is not a suitable solution. Instead we use a method very similar to what is proposed in [11]. First, every point in the pointcloud is mapped onto a 2D grid and the minimum $z$-coordinate for each cell in the grid is computed. Next all points whose $z$-distance to the minimum $z$-coordinate of the cell is below a threshold are removed. In our case we set the cells size to be $0.2m \times 0.2m$ and the $z$-distance threshold to be $0.15m$ across all three datasets. Figure 1 shows an example of a pointcloud before and after ground extraction. We have found that this method effectively removes the railway ties and bedding, while preserving the tracks which are now nicely isolated.

Once the rails were isolated the next step was to locate a
A discrete dynamic process is governed by the linear stochastic difference equation [2]:

\[ x_k = F_{k,k-1} x_{k-1} \]

where \( x \) is the state vector, \( F \) is the transition matrix between states, and \( k \) is the state.

Since \( F \) cannot be accurately determined, \( x_k \) will be a noisy prediction based on the previous state. In addition, in a discrete dynamic process the value of the state can be directly measured. However, the measured state value, denoted \( z_k \), is also noisy. The process (prediction) noise covariance matrix, \( Q \), and the measurement noise covariance matrix, \( R \), are parameters that are used to describe the expected error in both the prediction and measurement of the state vector. Given \( Q \) and \( R \), the Kalman filter re-computes the state vector using the following formula:

\[ x_k := x_k + K(z_k - x_k) \]  

where \( K \) is the Kalman gain.

The equation above controls the significance given to the difference between the measured and the predicted state vector. For instance, if the measurement noise is very high relative to the process noise then \( K \) tends towards zero. The value of \( K \) is optimized such that the estimated state vector minimizes the covariance matrix of the estimation error [5].

At each step the Kalman filter is used to compute the optimal estimation for the state parameters. In the rail track growing algorithm we seek to grow two parallel rail segments in the direction of the actual rail track in the 3D cloud. In order to achieve this, we need to use the Kalman filter to estimate the following state parameters: the two centroids of the two parallel line segments of the current state and the common unit direction vector of the two rail segments. For each state parameter, there is a predicted value and a measured value. We denote the predicted centroids as \( p^l_k \), \( p^r_k \), and the predicted direction as \( d_k \). The values of \( p^l_k \) and \( p^r_k \) are calculated from the addition of the previous rail segment centroid and the previous direction vector multiplied by an arbitrary scalar factor \( s \):

\[ p^l_k = p^l_{k-1} + s d_{k-1}, \quad j = 1, 2 \]

The value of \( s \) must be chosen carefully. If the value of \( s \) is too large then our next prediction might go off the rail track. On the other hand, choosing an \( s \) that is too small means that the direct measurement of the state parameters will be based on a small set of points and could therefore have a high level of inaccuracy. Both scenarios could lead to jeopardizing the accuracy of the Kalman filter. Therefore we simply set \( s \) to equal the length of the previous extracted segment of rail. The current direction vector, \( d_k \), is predicted by applying the rotation matrix, \( R \), between the two preceding direction vectors, \( d_{k-2} \) and \( d_{k-1} \), to the previous direction vector, \( d_{k-1} \):

\[ d_k = Rd_{k-1} \]

When computing the first state, the value for \( d_{k-2} \) is not available and therefore \( R \) is simply set to the identity matrix \( I \). In fact, \( R \) can be set to \( I \) for the whole model to simplify it. However, this simplification makes the dynamic process highly dependent on the previous state which might negatively impact the accuracy of the filtering process. On the other hand, computing \( R \) at each step as described keeps track of the changing curvature of the railway track making the filtering process more accurate. The complete railway track dynamic process can now be expressed as:
The measured direction of each of the two rails can be expressed as:

\[ a = \mathbf{p}_k^j + \frac{s}{2} d_k \]

where \( B \) is a constant set to 5 metres. Next, we extract all points whose Euclidean distance to the segment \( \mathbf{a}\mathbf{b} \) is within a factor, \( f \), of the constant rail thickness, \( r \). We set \( f = 4 \) and \( r = 0.2m \). Essentially, we are extracting a patch of points where we expect the next section of rail to be. We further refine this patch by finding only the points that fall on the rail. We do this by applying a line RANSAC to find the line with the strongest support using an error threshold of \( \frac{s}{2} \). All points falling on the RANSAC extracted line are considered rail points. We then set the value of \( p_{2k}^j \) equal to the centroid of the extracted line segment. We also measure the length of this line segment and use it to update the value of \( s \) as previously described. In order to calculate \( \mathbf{d}_{2k} \), we express the measured direction of each of the two rails as:

\[ d_{2k}^j = (p_{2k}^j - p_{2k-1}^j), \quad j = 1, 2 \]

and then measure \( \mathbf{d}_{2k} \) as:

\[ d_{2k} = \frac{1}{2} d_{2k}^1 + \frac{1}{2} d_{2k}^2 \]

5. Re-extract the current line segment based on the updated state parameters.

6. Repeat from step 1 or stop if next patch has less than 2 points.

### 3.3. Improving the Robustness of Railway Track Growing

Although it is sufficient to use the Kalman filter to follow a single rail instead of a pair of parallel rails to extract each railway track, using both rails increases the accuracy and robustness of the growing algorithm. The data dealt with in this work can be very sparse in certain regions and can contain missing rail sections; this is especially true for the Airborne dataset. In addition, due to sensor noise the ground extraction step can sometimes leave off ground points which lie very close to the rails which negatively impacts the state measurements. This implies that one half of a parallel rail segment pair can be a better representation of the true track state than the other. Therefore when measuring the \( k^{th} \) state we apply checks to each rail segment before accepting its measured values. The checks are:

1. The angle between the measured direction of a rail, \( \mathbf{d}_{2k}^j \), and the previous direction, \( \mathbf{d}_k \) does not exceed a threshold, \( \theta \).

2. The ratio of the number of points of the less dense segment to the number of points of the more dense segment is above a threshold, \( \alpha \).

In our experiments, we set \( \theta = 10^\circ \) and \( \alpha = 0.5 \). If one of the two measured rail segments is invalid then \( \mathbf{d}_{2k} \) is equated to the direction of the valid rail segment. In addition, the centroid of the valid rail segment is projected onto the invalid rail segment. If both rail segments are invalid then we attempt to re-initialize the Kalman filter by searching for a new pair of adjacent rail segments. We do this by continuing to extract patches of points in the direction of last valid state and checking for two parallel lines that are specific distance apart. The patches are of length \( B \) and width \( 3 \times w \) where \( w \) is the pre-determined rail gap between two rails which is set to 1.4m in our case. This process continues until we have found a valid pair of rail segments or we exceed the point cloud boundaries.

### 4. Experimental Results

We ran our algorithm on three subsets of three different datasets. The first dataset was collected using an Airborne LiDAR scanner. The second dataset is a terrestrial LiDAR scan of a 40 km stretch of railway near the city of Algoma in Northern Ontario, Canada. The Algoma data was collected using a TITAN 3D scanner which uses four Riegl Q120 lasers pointing in four different directions. The
third dataset was collected at an undisclosed location using the StreetMapper system which uses two VQ 250 lasers that scan 360 degrees in a circular manner around the lasers symmetrical axis. The data varies in resolution; the Airborne data is very sparse and contains missing patches of railway track as well as large missing sections of rail due to structures such as bridges blocking the viewpoint of the Airborne scanner. The TITAN data is more dense than the Airborne data and the StreetMapper data has a higher resolution than the TITAN data, see Figure 3. In each dataset the data was partitioned into several tiles (i.e. pointclouds), and the algorithm was manually initialized on each tile and allowed to grow until a stoppage condition was met. Here we report results for applying our technique on subsets of the three different datasets.

In order to evaluate our algorithm, we compare the railway tracks extracted using our technique with ground truth extraction and report both recall and precision as defined below:

\[
\text{Precision} = \frac{tp}{tp + fp} \quad (9) \\
\text{Recall} = \frac{tp}{tp + fn} \quad (10)
\]

Where \( tp \) is true positives, \( fp \) is false positives, and \( fn \) is false negatives.

The ground truth extraction was achieved by running a modified version of our algorithm where multiple manual re-initializations of the Kalman filter were allowed. In this way, if the algorithm ever diverges from the track or stops growing then the manual re-initialization would allow the growing process to continue until the railway track was fully extracted. Manual ground truth extraction was performed for the Airborne and StreetMapper datasets only. In order to calculate precision and recall we needed to compute the number of true positives, false positives, and false negatives. An extracted point was considered a true positive if it fell within a distance \( \epsilon = 0.4 \text{m} \) from the a ground truth point; otherwise it is a false positive. All points in ground truth that do not fall within \( \epsilon \) from an extracted point are considered false negatives. A visual comparison between the original cloud without ground and the extracted railway tracks is shown for Airborne (Figure 4, 5, 6, and 7) and TITAN (Figure 8, 9). Results for StreetMapper dataset are not visually shown due to confidentiality reasons. Table 1 provides a summary of the precision recall results for the Airborne and StreetMapper datasets.

By visual inspection it can be seen that we achieve very good results for these different kinds of datasets. We note here that the TITAN results actually show two parallel rails but due to the size of the TITAN tiles we had to zoom out significantly to show the entire original cloud and thus the pair of rails seem like a single rail. We achieve an average precision of 92.5% and an average recall of 95%. As the
rail constraints that prevent the growing process from deviating from the track which increases precision. At the same time, these rail constraints could cause the growing process to stop too early. We alleviate this issue by searching for a new pair of parallel line segments to initialize a new Kalman filter process and this explains our high recall rate.

5. Conclusion and Future Work

We described a technique that automatically extracts railway tracks from a variety of datasets. The technique models railway tracks as a dynamic system of local pairs of parallel rail segments and uses the Kalman filter to monitor the state of the dynamic system. On average, our technique successfully extracted 95% of the data with an accuracy of 92.5%. Although we have reported excellent results, we think there is still room for improvement in our algorithm. For example, we could pre-process the data by applying morphological filters to improve the connectivity of rails and remove ground noise prior to running the railway track growing algorithm. Furthermore, an automatic initialization of the growing algorithm is also desirable. Finally, we wish to test our technique on a larger subset of the datasets described here.

6. Acknowledgments

We would like to thank FedDev and Geodigital Inc. for their financial support along with access to the railway datasets.

References

