The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Automated Safety Inspection of Grade Crossings

Pradeep Ranganathan and Edwin Olson

***Abstract*— A grade crossing is a crossing of a railway line and a motor road. In 2009 alone there were 248 deaths and 682 injuries at grade crossings in the United States. Factors like the elevation profile of a crossing or the environment and foliage around the crossing can render it unsafe. Often, vehicles with low ground clearance bottom out on a crossing with a humped elevation profile. Excessive foliage around the crossing can obstruct the visibility of an approaching train, reducing the time a driver has to stop. Hence ensuring safety requires regular monitoring and timely maintenance of grade crossings across the country.**

In this paper, we describe our method for automatically inspecting grade crossings. Our work employs principled machine learning methods to detect grade crossings from sensor data and then reconstructs the profile of that rail-road intersection. We then show how traffic simulation on the reconstructed profile can be used to determine whether the crossing is unsafe.

I. Introduction

As of 2008, there were 137659 public, 85176 private and 1963 pedestrian grade crossings in the United States. One must realize that the maintenance of grade crossings is not a one-time effort since the conditions of the environment around a crossing change over time. Foliage grows obstructing the visibility around the crossing and the road surface can sink as the earth under it settles, causing a change in the elevation profile of the crossing.

Grade crossings are a frequent location for rail-road accidents. One of the most common incidents is a truck or other vehicle with low clearance bottoming out at a sufficiently "humped" grade crossing. This results in a rail/road traffic hangup. In the worst case this may even result in the train smashing into the stranded vehicle causing loss of life and property.

Current standards call for bi-yearly inspection of grade-crossings or around 400 grade crossings per day. The sheer number of crossings (224,798) makes automation of this process critical.

Ranging sensors like the Velodyne HDL 64-E [1] enable robust sensing of the environment around railway tracks. As a result, it is now possible to survey the area around a grade crossing efficiently and accurately. This also opens up the possibility of automatically analyzing this data to recover information relevant to the maintenance of grade crossings.

In this paper we describe a system that automatically inspects a grade crossing. The central contributions of this paper are:

This work was supported by FRA grant #DTFR53-09-G-00046

The authors are with Department of Computer Science and Engineering, Univeristy of Michigan, Ann Arbor, USA {rpradeep,ebolson}@umich.edu

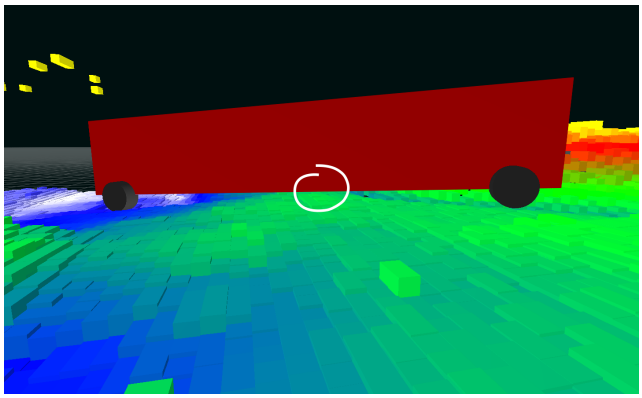

Fig. 1. An unsafe grade crossing. Grade crossings occur with diverse profiles. Crossings that are "humped" can cause heavy motor vehicles passing over them to bottom-out. While some crossings are visibly risky, some appear benign from the point of view of the truck driver. The proposed method reconstructs the profile of the grade crossing from sensor data and simulates traffic over the crossing to detect defects (highlighted) in the profile of the crossing.

- A supervised learning method that automatically detects grade crossings from a stream of 3-D points.
- An outlier rejection method based on Markov random fields to identify the extent of the road surface.
- An automatic safety analysis system based on traffic simulation on the reconstructed grade crossing profile.

All the components of our system operate in real-time, without any deferred processing of data.

In the next section, we review related work. We describe our method in the Section III. In Section IV, we evaluate our method on real world data and present the results of our evaluation, showing that our method can detect unsafe grade crossings in real-time.

II. Previous work

LIDARs are common sensors employed to sense the environment and acquire range data electronically. The Velodyne HDL 64-E is a scanning LIDAR system that uses an array of LIDAR sensors to acquire three dimensional points clouds of the environment around the scanner in real-time. The accuracy, robustness and resolution of LIDAR solutions make them viable for surveying.

Several teams participating in the DARPA Urban Challenge used Velodyne scanners to perceive the environment around an autonomous vehicle [1], [2]. The MIT team reports the use of data from a Velodyne to detect obstacles and recognize traversable ground [1]. Moosman et al. [3] describe another technique to segment 3-D range data to extract ground surfaces.

Fig. 2. Modified truck used for data gathering. The Velodyne HDL is mounted facing the ground at an angle to improve resolution of the ground surface immediately in front of the truck.

Others have reported the use of LIDAR scanners for modeling 3-D environments. ENSCO reports the use of a LIDAR scanner to map the environment around railways tracks [4]. The resulting data was then used for analysis of the crossing. However their system is not fully automated and requires manual intervention for recognizing crossings from data and for subsequent processing. Elashker and Bethel [5] and You et al. [6] describe methods that generate models of buildings using range scanners. Haala and Brenner [7] describe the generation of 3-D city models from aerial range scan data.

Sheikh et al. [8] present a trespass detection system that can track pedestrians and vehicles at a grade crossing. Their work addresses another aspect of safety at grade crossings and is complementary to our work.

III. Method

A. Scan acquisition

Data was gathered along rail-road intersections using a truck modified to travel on railway tracks (see Fig. 2). The truck has a Velodyne HDL mounted in front, facing the ground at 45° in order to improve the resolution of the surface immediately in front of the truck. The truck also contains an Applanix IMU unit to track and record the pose of the truck as it travels on the track. Images of the path through which the truck traveled were captured at regular intervals to aid visualization and labeling of the data.

B. Intersection detection

In order to automate the task of surveying grade crossings, one must first automate the task of detecting a grade crossing. The exact locations of grade crossings are not known a priori. Existing databases of grade crossing locations are not accurate because of the lack of guidelines regulating the process of obtaining grade crossing locations. For example, manual tagging of grade crossings from aerial imagery can introduce errors of 100 meters or more. Hence a more accurate approach for detecting a grade crossing is required.

We approach this as a pattern recognition problem: recognize patterns that correspond to intersections, from a stream of 3-D point cloud data. Support Vector Machines (SVM) [9], [10] are a popular supervised machine learning method that can learn to classify new data based on training data. We used LIBSVM [11], a popular implementation of SVM classifiers.

The first step is to extract features from the data that can aid in classification. We begin by creating an occupancy grid of the 3-D point cloud. The space around the sensor is divided into tiles of equal area along the plane of the ground and points that fall into each of the tiles are recorded. In our implementation we chose a tile size of $0.5\ m \times 0.5\ m$.

Intuitively, the intersection is recognizable mostly by features on the ground rather than objects above the ground. Hence while constructing feature vectors, we ignore features that begin more than M meters above the ground in front of the train. This construction prefers features supported by the ground (tree-trunks and shrubs) while ignoring features that are free floating or vertically unsupported (tree canopies). In our implementation we choose $M = 5$ meters. The exact value of this parameter does not affect the accuracy of the resulting classifier. The only constraint on this parameter is that it should be large enough to separate objects supported by the ground and that are vertically unsupported.

Due to noise in the Velodyne, the returns from the sensor do not represent exact ground points [1]. We compute at an estimate of the actual level by taking the point at the 80^{th} percentile of the points falling on a given tile. The resulting occupancy grid is visualized in Fig. 3. Hereafter, we will refer to this value of the point at the 80^{th} percentile of the points falling on a tile as the μ-statistic of that tile.

As an effect of ignoring the contribution of points above a certain height from the ground, there can be tiles that have no data points associated with them. This can happen if the entire contribution to that tile comes from objects high above the ground. The SVM classifier cannot handle feature vectors with missing values. So missing tiles were assigned statistics from the nearest tile in the previous time step. Because of the low frequency nature of most terrain, this scheme works quite well in practice.

It is difficult to identify a crossing from a single Velodyne scan. Instead, we would like to construct features vectors that use a history of Velodyne scans. Hence feature vectors for classification are constructed from a contiguous subset of tiles from the occupancy grid. Each feature vector consists of tile statistics W meters on either side of the sensor collected over H time steps. If the tile size is $T \times T$, each feature vector will then contain $N_f = (2W \times H)/T^2$ feature elements (Fig. 4). By construction, each feature vector will overlap with the feature vector constructed in the previous time step.

Velodyne scans were obtained from six different intersections and feature vectors were constructed from the 3-D point cloud data as described. Each feature vector was labelled manually using the images taken along with the Velodyne scans. We then used the labelled feature vectors obtained as

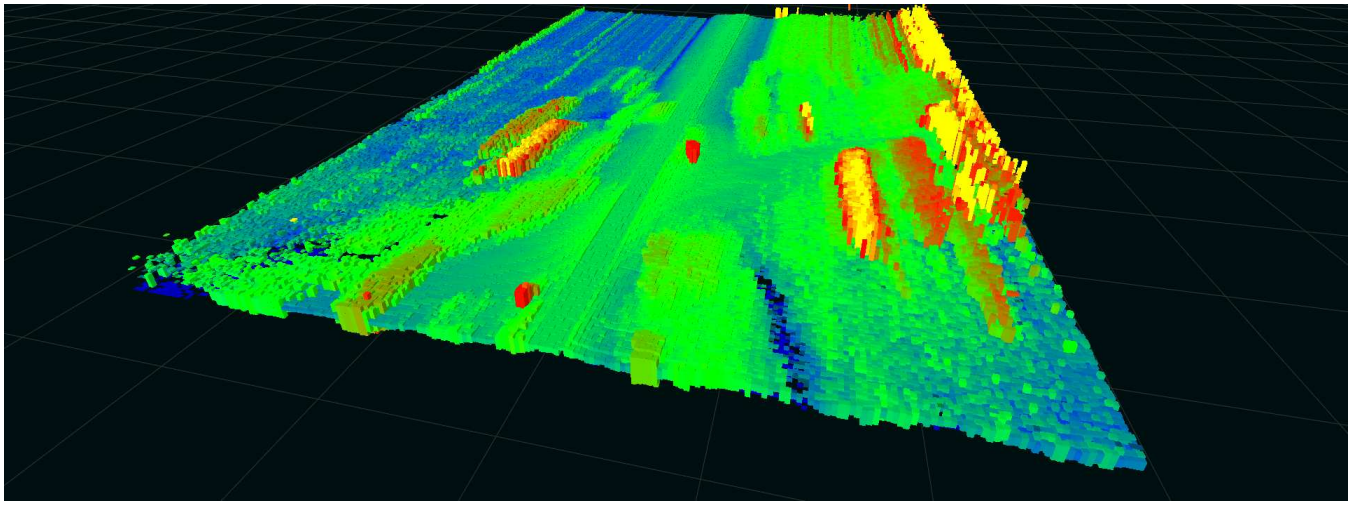

Fig. 3. This visualization shows μ-statistics on the occupancy grid. Each tile of the occupancy grid contains a small three dimensional bar whose height is equal to the μ-statistic computed at that tile. The μ-statistic corresponds to the ground height estimate at a tile of the occupancy grid.

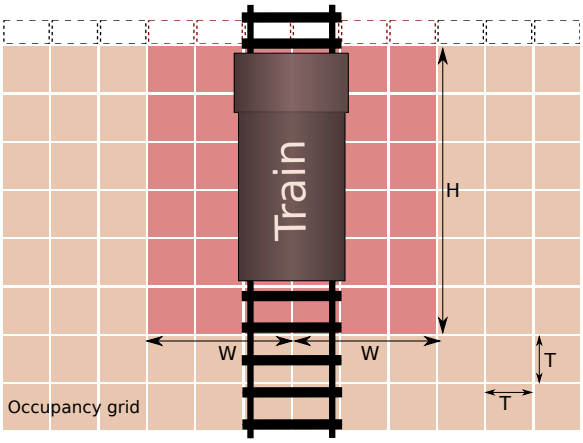

Fig. 4. Feature vector construction. Feature vectors for classification (shaded red) are obtained from a subset of the data in the occupancy grid. One feature vector is generated at each time step using statistics from tiles that are W meters on either side of the current position of the truck and H meters behind it. The size of each tile of the occupancy grid is $T \times T$. The entire occupancy grid is shown in Fig. 3. Each feature vector shares $2W/T$ elements with the feature vector generated at the previous time step.

training data for a SVM classifier with a radial basis kernel

$$K(u, v) = e^{-\gamma|u-v|^2}$$

The parameter γ was set to $\frac{1}{N_f}$, where N_f is the number of features in the feature vector. All other parameters were set to their respective defaults provided by LIBSVM. The trained classifier was then used to classify new feature vectors obtained from data sets collected on other routes. Henceforth, we refer to these locations on the track that the SVM classifier labels as an intersection as *intersection points*. The SVM classifier detects a series of intersection points for each grade crossing. The number of intersection points is roughly correlated with the width of the road at the crossing.

C. Road surface isolation

The next step is to separate the surface of the road from the surrounding area. For this and subsequent stages, we construct an occupancy grid with tile size $0.25\ m \times 0.25\ m$ and compute the μ-statistics for the tiles. We limit processing to a rectangular section of the occupancy grid whose boundary is at a perpendicular distance of 15 meters from reported intersection points (the length and breadth of this rectangular section is at least 30 meters). This length is chosen in order to contain a strip of road long enough to analyze the passage of a semi-tractor trailer whose typical length is 11 to 13 meters.

One characteristic of the road is its prominent edge with respect to the adjacent ground and vegetation. As a consequence, it is possible to separate the road surface from the surrounding area by detecting these edges.

We detect these edges by transforming the existing occupancy grid data into an image and then applying standard image processing techniques. The data in an occupancy grid of dimensions $L \times B$ can be transformed into a grayscale image of corresponding dimensions by quantizing the information in each grid tile over the possible number of intensity values. We have already computed the μ-statistic for each grid tile. We then compute the intensity of a pixel in the image corresponding to a grid tile using a quantized linear transformation:

$$L_{ij} = \left\lfloor \frac{\mu_{ij} - \mu_{min}}{\mu_{max} - \mu_{min}} \times 255 \right\rfloor,$$

Where μ_{ij} is the μ-statistic computed for each grid tile and μ_{min}, μ_{max} are the minimum and maximum values that the statistic takes over the whole occupancy grid.

The resulting image of the terrain around the grade crossing is noisy due to small artifacts that occur on natural terrain e.g. stones, pits and pot-holes on roads. Using this image directly for detecting edges will lead to detection of spurious edges. Hence we apply a Markov random field based noise reduction method to smooth out the image and remove noise.

A Markov Random Field (MRF) [12] consists of a set of nodes connected together by edges according to a pre-defined neighborhood scheme. Each node is assigned a label f from a possible set of labels $F = \{f_1, f_2, f_3, \ldots\}$. The problem to be solved is then modelled as an MRF optimization problem in which one assigns new labels to each node such that the new labels are similar to the observed labels (label costs) and nearby nodes have similar labels (edge costs).

For our problem the nodes are the grid tiles and the neighborhood scheme is based on adjacency of tiles. This is because adjacent nodes are correlated due to their physical proximity and are thus likely to have similar height values. The set of labels is the set of possible heights that can be assigned to the grid tiles. The noise reduction problem is then transformed into an MRF optimization problem where one assigns new heights to nodes subject to the truncated quadratic label cost function

$$V(f_i - f_j) = \min\left((f_i - f_j)^2, d_v\right),$$

and truncated quadratic edge cost function

$$D(f_j) = \lambda \ \min\left((I(j) - f_j)^2, d_e\right),$$

where d_v and d_k are thresholds beyond which the cost functions are truncated.

Because of the large number of variables and edges, solving the resulting problem exactly is computationally infeasible. Instead, we use loopy belief propagation to incrementally improve our posterior estimate. Computationally, our method is identical to that used by Felzenswalb [13], though the semantic meaning of the labels is different in our application. For the parameters, we use $\lambda = 0.05$, $d_v = 200$, $d_e = 10000$. The neighborhood edge length was varied from 1 to 5 and for each setting of the edge length, five iterations of loopy belief propagation were done. For a detailed explanation of this method, we refer the reader to [13].

We then apply the Robert's cross operator [14] on the smoothed image to extract edges. The Robert's cross operator detects edges by sequentially convolving two matrices

$$R_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$R_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix},$$

with the image. The Robert's cross edge detection operator is computationally inexpensive but is sensitive to noise in the image. However for our application, use of the Robert's cross operator to detect edges is effective because we have already removed noise in the image. This operator computes an edge intensity for each pixel. We then adaptively threshold these images at the 90^{th} percentile of the range of edge intensities to finally declare pixels as edges (see Fig. 5).

However, this edge detection phase does not separate the track surface from the road surface because there is no prominent edge separating the track from the road (see Fig. 5). This is because, in the vicinity of the grade crossing, the railway track is at the same height as the road. As a consequence, when we use these edges as barriers for the flow of traffic during simulation in the next stage of our method, the simulated traffic might pass through the track surface. This is clearly undesirable. This apparent shortcoming of the edge detection phase is mitigated by adding appropriate constraints in the next stage of our method.

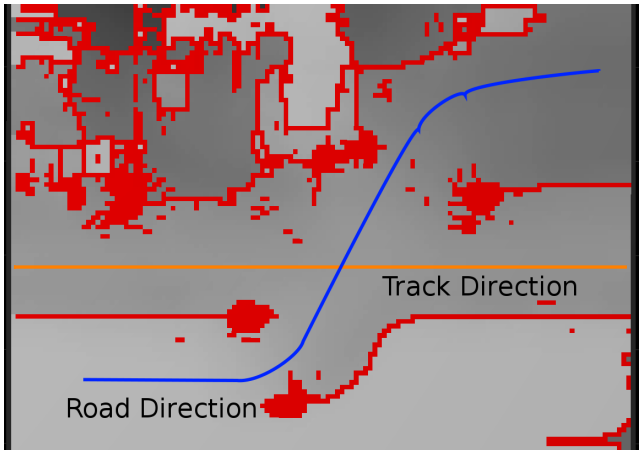

Fig. 5. Edges detected using the Robert's cross operator after removing noise. This intersection is the same as the one shown in Fig. 6. The road and track have been labeled manually for reference.

D. Traffic simulation

From the previous stage, we have an estimate of the road surface. Next, we want to determine whether a truck would bottom out while traveling over that road. For this analysis, we use a sampling based method to simulate traffic over the road surface that was extracted.

We have a set of candidate points for the road surface from the previous stage of our method. In this stage, we add more constraints to refine this candidate set such that the road surface is clearly separated from the track surface. Traffic simulation is then done over this refined set. First two points are randomly choosen from this refined set, such that they correspond to the point of contact of the wheels of a truck on the road. We then search along the line connecting these two points for projections that exceed the ground clearance of the truck. If any such projections are found the crossing is declared dangerous. We now give a detailed description of this simulation procedure in the following paragraphs.

The sample space is a subset of the occupancy grid tiles that are within a radius of 15 meters from the first and last intersection points identified by the intersection detection phase. We randomly pick two points from this subset of the occupancy grid, sampling uniformly over all elements from the subset. We then declare that the two chosen points (x_1, y_1) and (x_2, y_2) represent a plausible line of traffic flow if they satisfy the following criteria:

1) The distance between two points is between 11 meters and 13 meters. These values are based on the average length of a truck.

2) The line segment connecting the points passes through the strip of railway track containing intersection points.
3) None of the points that are labeled as an edges fall on the line segment connecting the two selected points.
4) The line segment connecting these points does not present an angle less than 30° or more than 150° to the direction of the train through the grade crossing. This constraint discards traffic flow directions that are along the direction of the track.

Once a plausible line of flow of traffic is identified, we check for sufficient clearance by comparing the deviation of the μ-statistic computed for the grid tiles along the line segment, with the expected height at a point along the same line segment. The expected height z_{xy} along a line segment from (x_1, y_1, z_1) and (x_2, y_2, z_2) dimensions is given by the following equation:

$$\frac{x - x_1}{x_1 - x_2} = \frac{y - y_1}{y_1 - y_2} = \frac{z_{xy} - z_1}{z_1 - z_2}$$

Here $z_1 = \mu_{x_1 y_1}$ and $z_2 = \mu_{x_2 y_2}$. μ_{xy} is the observed height of ground at a point (x, y) along the line segment. The condition for insufficient ground clearance along this line segment is then

$$z_{xy} + C \leq \mu_{xy},$$

where C is the ground clearance of the truck. Informally this amounts to ensuring that the vertical clearance between the two chosen points is sufficient for a truck with low ground clearance.

Thus, if the deviation exceeds the expected value by more than the ground clearance C of the truck, the point of failure is noted and the crossing is declared dangerous. This sampling procedure is repeated until a sufficient density of points on the road surface have been sampled and analyzed. The number of iterations was set to 1000 after empirical observations. The crossing is declared safe if none of these traffic flow lines present any danger.

IV. Experimental results

The data used for this study was obtained from six grade crossings in Virginia. Each grade crossings has a different elevation profile. Data was collected over a length of 30-40 metres around the grade crossing. This gave us an average of 150 Velodyne scans per grade crossing. Feature vectors were then constructed from these Velodyne scans and each feature vector was labeled manually using ground truth from images collected along with the scans.

A. Accuracy of intersection detection

The following table shows the accuracy of the SVM classifier that was trained as described in section III B. We report the number of false positives and false negatives when data from each intersection was presented to the classifier for cross-validation.

Location	test cases	false +	false -
Old State 1	93	9	4
Old State 2	123	0	3
Old State 3	147	4	17
Green Mountain Rd.	226	0	6
Meyers Town	146	0	9
Pine Grove	172	6	10

The trained classifier detects the presence of grade crossings on all test cases accurately. The false positives and false negatives reported in the table above are for intersection points. The overall accuracy of the classifier for intersection points was 90%.

Traffic simulation works well even in the presence of a few false negatives on the intersection. Only false positives sufficiently far away from an intersection create a problems. The trained classifier does not produce these kinds of false positives that are far away from an intersection.

The method for constructing feature vectors for the classifier has two free parameters: W and H. These determine the number of features in the feature vector as described in section III B. Optimal values for these parameters were learnt by optimizing the classifier performance as a function of these parameters. The optimal values of the parameters W and H for the intersection detector were found to be 2 meters and 8 meters respectively.

B. Traffic simulation and analysis

Fig 6 shows the different stages in processing sensor data from a single faulty grade-crossing.

C. Performance

Grade crossing detection and analysis can be done on the fly on the sensor vehicle without any post-processing. For this application it would suffice to identify and process data from an intersection such that the processing requirements are bounded by a few seconds. The detection of an intersection, followed by road surface isolation and safety analysis takes about 3 seconds on a 2.3 GHz processor. This is easily completed before the next grade crossing. Only the μ-statistic data for each tile on the occupancy grid is required for traffic simulation. This data ($<$ 1MB) is small enough to fit in the main memory of the system and can be discarded once the simulation is complete.

V. Conclusion

We have described a system that automatically surveys grade crossings using ranging sensor data and generates safety reports for them. Our system uses principled machine learning methods to detect crossings. It then performs simulation based analysis on a non-parametric model of the grade crossing to detect faults in the crossing. We also showed how the system can be engineered to perform this analysis in real-time, without any deferred processing.

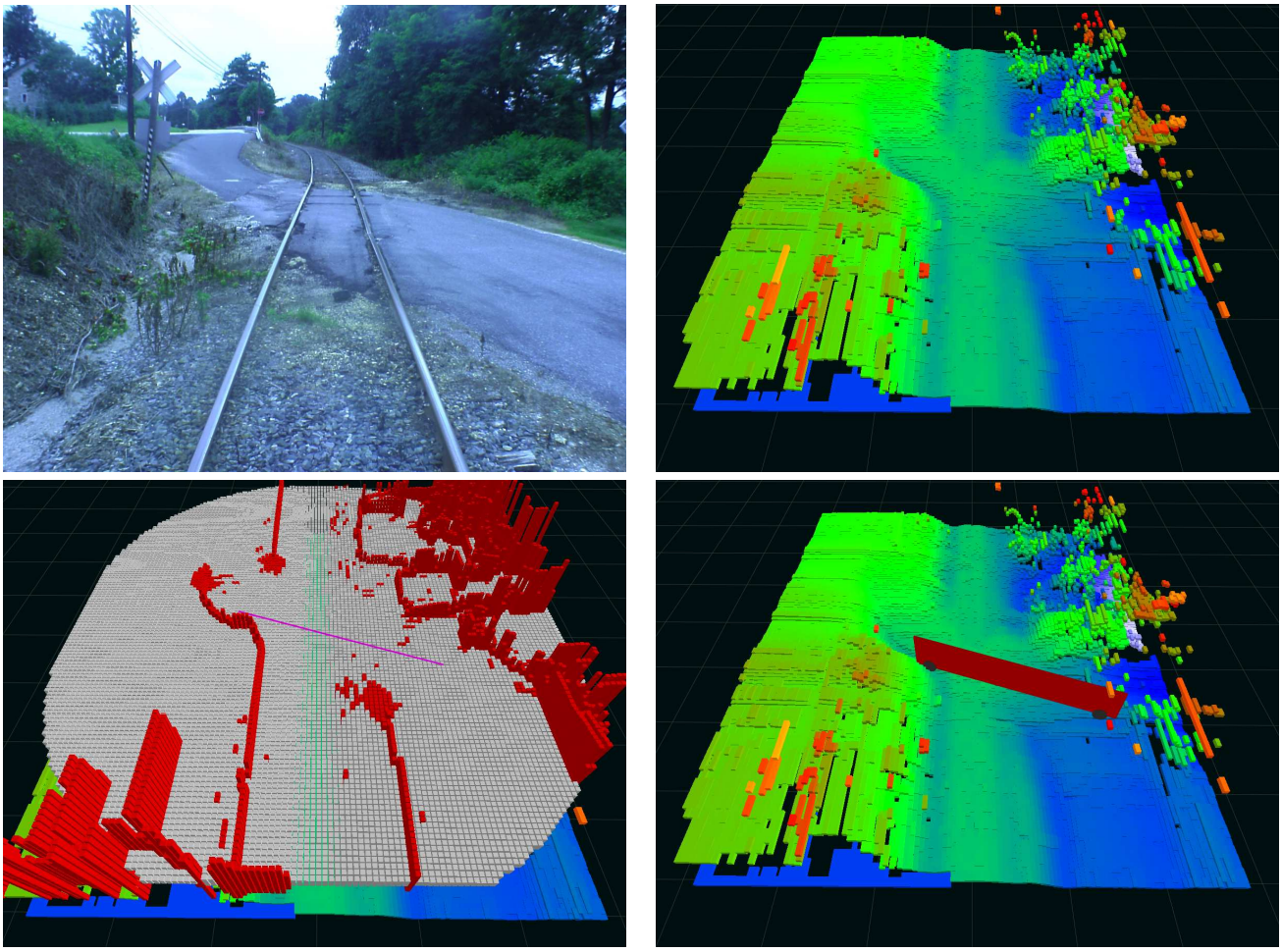

Fig. 6. Different stages in our method. (top-left) Photograph of the grade crossing being analyzed. (top-right) Model of the crossing reconstructed from sensor data. (bottom-left) Edge detection phase where edges are detected on a subset of the tiles on the occupancy grid. Edges are shown along with their intensities plotted as heights. A traffic flow line that is dangerous is shown in magenta. (bottom-right) The orientation in which a truck could bottom-out at this grade crossing.

VI. Future work

We are actively working to develop methods for extracting other safety parameters such as the distance from which a car can see an approaching train and mapping assets such as sign posts and railway gates.

VII. Acknowledgements

We would like to thank ENSCO for providing us with real-world sensor data from grade crossings.

References

[1] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams., "A perception driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, September 2008.

[2] Montemerlo, Michael, Becker, and et al, "Junior: The Stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[3] O. P. Frank Moosmann and C. Stiller, "Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion," in *Intelligent Vehicles Symposium*, 2009.

[4] P. V. Arman, "Automated technology for rail-based grade crossing surveying," ENSCO, Tech. Rep. DOT-FR-08-08, 2008.

[5] A. F. Elaksher and J. S. Bethel, "Building extraction using lidar data," in *ASPRS-ACSM Annual Conference and FIG XXII Congress*, 2002, pp. 22–26.

[6] S. You, J. Hu, U. Neumann, and P. Fox, "site modeling from lidar," in *In Proc. 2nd Intl Workshop Computer Graphics and Geometric Modeling (CGGM*, 2003, p. 588.

[7] N. Haala and C. Brenner, "Generation of 3d city models from airborne laser scanning data," in *Proceedings EARSEL workshop on LIDAR remote sensing on land and sea*, 1997, pp. 105–112.

[8] Y. A. Sheikh, Y. Zhai, K. Shafique, and M. A. Shah, "Visual monitoring of railroad grade crossing," in *SPIE Defense and Security Symposium*. SPIE Press, 2004, pp. 654–660.

[9] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*. New York, NY, USA: ACM, 1992, pp. 144–152.

[10] D. Meyer, F. Leisch, and K. Hornik, "The support vector machine under test," *Neurocomputing*, vol. 55, no. 1-2, pp. 169 – 186, 2003.

[11] C.-C. Chang and C.-J. Lin, "Libsvm - a library for support vector machines," 2001.

[12] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Springer, 2009.

[13] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 1, pp. 261–268, 2004.

[14] L. Roberts, "Machine perception of 3-d solids," 1963.