Robust Fastener Detection for Autonomous Visual Railway Track Inspection

Xavier Gibert, Vishal M. Patel, and Rama Chellappa
UMIACS, University of Maryland
College Park, MD 20742-3275, USA
gibert,pvishalm,rama@umiacs.umd.edu

Abstract

Fasteners are critical railway components that maintain the rails in a fixed position. Their failure can lead to train derailments due to gage widening or wheel climb, so their condition needs to be periodically monitored. Several computer vision methods have been proposed in the literature for track inspection applications. However, these methods are not robust to clutter and background noise present in the railroad environment. This paper proposes a new method for fastener detection by 1) carefully aligning the training data, 2) reducing intra-class variation, and 3) bootstrapping difficult samples to improve the classification margin. Using the histogram of oriented gradients features and a combination of linear SVM classifiers, the system described in this paper can inspect ties for missing or defective rail fastener problems with a probability of detection of 98% and a false alarm rate of 1.23% on a new dataset of 85 miles of concrete tie images collected in the US Northeast Corridor (NEC) between 2012 and 2013. To the best of our knowledge, this dataset of 203,287 crossties is the largest ever reported in the literature.

1. Introduction

Monitoring the condition of railway fasteners is essential to ensure train safety. Fasteners maintain gage by keeping both rails firmly attached to the crossties. According to the Federal Railroad Administration (FRA) safety database[1], in 2013, out of 651 derailments due to track problems, 27 of them were attributed to gage widening caused by defective spikes or rail fasteners, and another 2 to defective or missing spikes or rail fasteners. In the United States, regulations enforced by the FRA[2] prescribe visual inspection of high speed rail tracks with a frequency of once or twice per week, depending on track speed. These manual inspections are currently performed by railroad personnel, either by walking on the tracks or by riding a hi-rail vehicle at very low speeds. However, such inspections are subjective and do not produce an auditable visual record. In addition, railroads usually perform automated track inspections with specialized track geometry measurement vehicles at intervals of 30 days or less between inspections. These automated inspections can directly detect gage widening conditions. However, it is preferable to find fastening problems before they develop into gage widening conditions. The locations and names of the basic track elements mentioned in this paper are shown in Figure 1.

Recent advances in CMOS imaging technology, have resulted in commercial-grade line-scan cameras that are capable of capturing images at resolutions of up to 4,096$\times$1 and line rates of up to 140 KHz. At the same time, high-intensity LED-based illuminators available in the market, whose life expectancies are in the range of 50,000 hours, enable virtually maintenance-free operation over several months. Therefore, technology that enables autonomous visual track inspection from an unattended vehicle (such as a passenger train) may become a reality in the not-too-distant future. Now that the systems integration challenge is already solved, we expect that there will be a surge in applications in the near future.

This paper shows that, by applying computer vision techniques, it is possible to inspect tracks for missing and broken components using only grayscale images with no additional sensors. Figure 2 shows the types of defects that our algorithm can detect. The detectors have been tested on concrete ties, but the framework can easily accommodate other types of fasteners and ties.

This paper is organized as follows. In Section 2, we review some related works on this topic. Details of our approach are given in Section 3. Experimental results on 85 miles of concrete tie images are presented in Section 4. Section 5 concludes the paper with a brief summary and discussion.

[1] http://safetydata.fra.dot.gov
[2] 49 CFR 213 – Track Safety Standards

Table 1. Taxonomy of automated visual railway component inspection methods.

Authors	Year	Components	Defects	Features	Decision methods
Marino *et al.* [13, 8]	2007	Fasteners	Missing	DWT	3-layer NN
Gibert *et al.* [9, 3]	2007	Joint Bars	Cracks	Edges	SVM
Babenko [1]	2008	Fasteners	Missing/Defective	Intensity	OT-MACH corr.
Resendiz *et al.* [16]	2013	Ties/Turnouts	–	Gabor	SVM
Li *et al.* [11]	2014	Tie plates	Missing spikes	Lines/Haar	Adaboost
Gibert *et al.* [10]	2014	Concrete ties	Cracks	DST	Iterative shrinkage

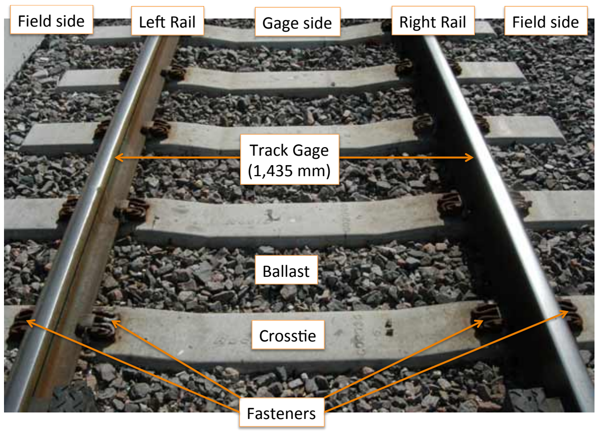


Figure 1. Definition of basic track elements.

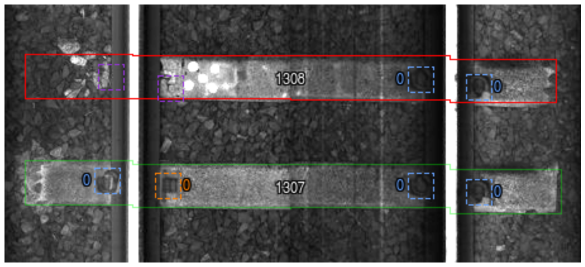


Figure 2. Example of defects that our algorithm can detect. Blue boxes indicate good fasteners, orange boxes indicate broken fasteners, and purple boxes indicate missing fasteners. White numbers indicate tie index from last mile post. Other numbers indicate type of fastener (for example, 0 is for e-clip fastener).

2. Prior Work

Since the pioneering work by Cunningham *et al.* [6, 18] in the mid 1990's, machine vision has been gradually adopted by the railway industry as a track inspection technology. Those first generation systems were capable of collecting images of the railway right of way and storing them for later review, but they did not facilitate any automated detection. As faster processing hardware became available, several vendors began to introduce vision systems with increasing automation capabilities.

In [13, 8], Marino *et al.*, describe their VISyR system, which detects hexagonal-headed bolts using two 3-layer neural networks (NN) running in parallel. Both networks take the 2-level discrete wavelet transform (DWT) of a 24×100 pixel sliding window (their images use non-square pixels) as an input to generate a binary output indicating the presence of a fastener. The difference is that the first NN uses Daubechies wavelets, while the second one uses Haar wavelets. This wavelet decomposition is equivalent to performing edge detection at different scales with two different filters. Both neural networks are trained with the same examples. The final decision is made using the maximum output of each neural network. In [9, 3], Gibert *et al.*, describe their VisiRail system for joint bar inspection. The system is capable of collecting images on each rail side, and finding cracks on joint bars using edge detection and a Support Vector Machine (SVM) classifier that analyzes features extracted from these edges. In [1], Babenko describes a fastener detection method based on a convolutional filter bank that is applied directly to intensity images. Each type of fastener has a single filter associated with it, whose coefficients are calculated using an illumination-normalized version of the Optimal Tradeoff Maximum Average Correlation Height (OT-MACH) filter [12]. This approach allowed accurate fastener detection and localization and it achieved over 90% fastener detection rate on a dataset of 2,436 images. However, the detector was not tested on longer sections of track. In [16], Resendiz *et al.* use texture classification via a bank of Gabor filters followed by an SVM to determine the location of rail components such as crossties and turnouts. They also use the MUSIC algorithm to find spectral signatures to determine expected component locations. In [11], Li *et al.* describe a system for detecting tie plates and spikes. Their method, which is described in more detail in [17], uses an AdaBoost-based object detector [19] with a model selection mechanism that assigns the object class that produces the highest number of detections within a window of 50 frames. The problem of crack detection on concrete ties was addressed on our previous work[10], where we used iterative shrinkage over dictionaries of shearlets and wavelets to separate the correlated texture from crack edges.

Table 1 summarizes several methods for inspecting track components described in the literature. In addition to the

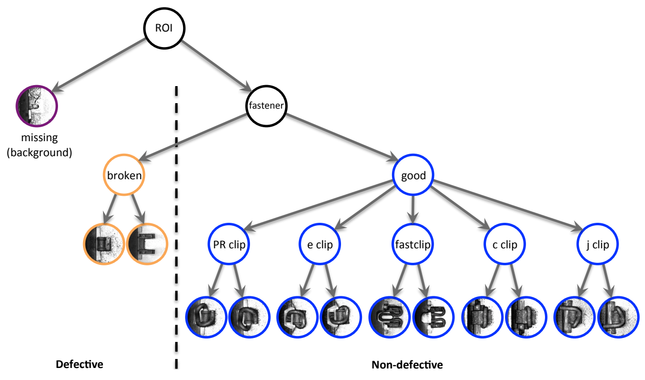


Figure 3. Object categories used for detection and classification (from coarsest to finest levels).

works described in this section, there are other commercial vendors that offer automated visual track inspection systems, but they have not disclosed the techniques that they use nor their detection performance. More details about these and other methods can be found in the surveys by Molina and Edwards [4], and Podder [15].

3. Proposed Approach

In this section, we describe the details of our proposed approach to automatic fastener detection.

3.1. Overview

Due to surface variations that result from grease, rust and other elements in the outdoor environment, segmentation of railway components is a very difficult task. Therefore, we avoid it by using a detector based on a sliding window that we run over the inspectable area of the tie. The detector uses the well-know descriptor based on the Histograms of Oriented Gradients [7] (HOG), which was originally designed for pedestrian detection, but it has been proven effective for a variety of object detection tasks in unconstrained environments. Although, most of the time, fasteners are located very close to the rail, we need to search over a much broader area because on turnouts (switches and frogs) fasteners are positioned farther away from the rail, with more varied configurations.

3.2. Classification

Our goal is to simultaneously detect, within each predefined Region of Interest (ROI), the most likely fastener location and then classify such detections into one of three basic conditions: background (or missing fastener), broken fastener, and good fastener. Then, for good and broken fastener conditions, we want to assign class labels for each fastener type (PR clip, e-clip, fastclip, c-clip, and j-clip). Figure 3 shows the complete categorization that we use, from coarsest to finest. At the coarsest level, we want to classify fastener vs. unstructured background clutter. The background class also includes images of ties where fasteners are completely missing. We have done this for these reasons: 1) it is very difficult to train a detector to find the small hole left on the tie after the whole fastener has been ripped off, 2) we do not have enough training examples of missing fasteners, and 3) most missing fasteners are on crumbled ties for which the hole is no longer visible. Once we detect the most likely fastener location, we want to classify the detected fastener between broken vs. good, and then classify it into the most likely fastener type. Although this top-down reasoning works for a human inspector, it does not work accurately in a computer vision system because both the background class and the fastener class have too much intra-class variations. As a result, we have resorted to a bottom-up approach.

Since we use inner products, our detector may resemble the correlation-based approach used in [1], but there are three key differences that sets us apart: 1) our input is a HOG feature vector rather than raw pixel intensities, 2) we use a linear SVM to learn the coefficients of the detection filter, 3) we use a second classifier to reject misclassified fastener types.

To achieve the best possible generalization at test time, we have based our detector on the maximum margin principle of the SVM. Although SVM is a binary classifier, it is straightforward to build a multi-class SVM, for example, by combining several one-vs-rest or one-vs-one SVM classifiers, either by a voting scheme or by using the DAG-SVM framework [14]. Our approach uses the one-vs-rest strategy, but instead of treating the background class as just another object class, we treat it as a special case and use a pair of SVMs per object class. For instance, if we had used a single learning machine, we would be forcing the classifier to perform two different unrelated tasks: a) reject that the image patch that does not contain random texture and b) reject that the object does not belong to the given category. Therefore, given a set of object classes $\mathcal{C}$, we train two detectors for each object category. The first one, with weights b_c, classifies each object class $c \in \mathcal{C}$ vs. the background/missing class $m \notin \mathcal{C}$, and the second one, with weights f_c classifies object class c vs. other object classes $\mathcal{C} \backslash c$. As illustrated in Figure 4, asking our linear classifier to perform both tasks at the same time would result in a narrower margin than training separate classifiers for each individual task. Moreover, to avoid rejecting cases where all f_c classifiers produce negative responses, but one or more b_c classifiers produce strong positive responses that would otherwise indicate the presence of a fastener, we use the negative output of f_c as a soft penalty. Then the likelihood that sample x belongs to class c becomes

$$L_c(x) = b_c \cdot x + \min(0, f_c \cdot x), \tag{1}$$

where $x = HOG(I)$ is the feature vector extracted from a given image patch I. The likelihood that our search region contains at least one object of class c is the score of the union, so

$$L_c = \max_{x \in \mathcal{X}} L_c(x), \tag{2}$$

where $\mathcal{X}$ is the set of all feature vectors extracted within the search region, and our classification rule becomes

$$\hat{c} = \begin{cases} \arg\max_{c \in \mathcal{C}} L_c & \max_{c \in \mathcal{C}} L_c > 0 \\ m & \text{otherwise.} \end{cases} \tag{3}$$

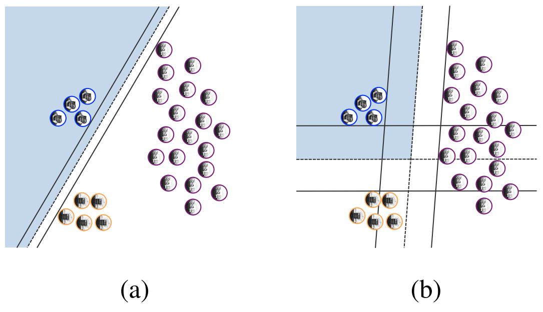


Figure 4. Justification for using two classifiers for each object category. Shaded decision region corresponds fastener in good condition, while white region corresponds to defective fastener. Blue circles are good fasteners, orange circles are broken fasteners, and purple circles are background/missing fasteners. (a) Classification region of good fastener vs. rest (b) Classification region of intersection of good fastener vs. background and good fastener vs. rest-minus-background. The margin is much wider than using a single classifier.

3.3. Score Calculation

For the practical applicability of our detector, it needs to output a scalar value that can be compared to a user-selectable threshold τ. Since there are several ways for a fastener to be defective (either missing or broken), we need to generate a single score that informs the user how confident the system is that the image contains a fastener in good condition. This score is generated by combining the output of the binary classifiers introduced in the previous section.

We denote the subset of classes corresponding to good fasteners as $\mathcal{G}$ and that of broken fasteners as $\mathcal{B}$. These two subsets are mutually exclusive, so $\mathcal{C} = \mathcal{G} \cup \mathcal{B}$ and $\mathcal{G} \cap \mathcal{B} = \emptyset$. To build the score function, we first compute the score for rejecting the missing fastener hypothesis (i.e, the likelihood that there is at least one sample $x \in \mathcal{X}$ such that $x \notin m$) as

$$S_m = \max_{c \in \mathcal{G}} L_c \tag{4}$$

where L_c is the likelihood of class c defined in Eq. 2. Similarly, we compute the score for rejecting the broken fastener hypothesis (i.e, the likelihood that for each sample $x \in \mathcal{X}$, $x \notin \mathcal{B}$) as

$$S_b = -\max_{c \in \mathcal{B}} \max_{x \in \mathcal{X}} f_c \cdot x, \tag{5}$$

The reason why the S_b does not depend on a c-vs-background classifier b_c is because mistakes between missing and broken fastener classes do not need to be penalized. Therefore, S_b need only produce low scores when x matches at least one of the models in $\mathcal{B}$. The negative sign in S_b results from the convention that a fastener in good condition should have a large positive score. The final score becomes the intersection of these two scores

$$S = \min(S_m, S_b). \tag{6}$$

The final decision is done by reporting the fastener as good if $S > \tau$, and defective otherwise.

3.4. Training Procedure

The advantage of using a maximum-margin classifier is that once we have enough support vectors for a particular class, it is not necessary to add more inliers to improve classification performance. Therefore, we can potentially achieve relatively good performance with only having to annotate a very small fraction of the data. To generate our training set, we initially selected ∼30 good quality (with no occlusion and clean edges) samples from each object category at random from the whole repository and annotated the bounding box location and object class for each of them. Our training software also automatically picks, using a randomly generated offset, a background patch adjacent to each of the selected samples. Once we had enough samples from each class, we trained binary classifiers for each of the classes against the background and tested on the whole dataset. Then, we randomly selected misclassified samples and added those that had good or acceptable quality (no occlusion) to the training set. To maintain the balance of the training set, we also added, for each difficult sample, 2 or 3 neighboring samples. Since there are special types of fasteners that do not occur very frequently (such as the c-clips or j-clips used around joint bars), in order to keep the number of samples of each type in the training set as balanced as possible, we added as many of these infrequent types as we could find.

3.5. Alignment Procedure

For learning the most effective object detection models, the importance of properly aligning the training samples cannot be emphasized enough. Misalignment between different training samples would cause unnecessary intra-class variation that would degrade detection performance. Therefore, all the training bounding boxes were manually annotated, as tightly as possible to the object contour by the same person to avoid inducing any annotation bias. For training the fastener vs. background detectors, our software cropped the training samples using a detection window centered around these boxes. For training the fastener vs. rest detectors, our software centered the positive samples using the user annotation, but the negative samples were re-centered to the position where the fastener vs. background detector generated the highest response. This was done to force the learning machine to learn to differentiate object categories by taking into account parts that are not common across object categories.

4. Experimental Results

To evaluate the accuracy of our fastener detector, we have tested it on 85 miles of continuous trackbed images. These images were collected on the US Northeast Corridor (NEC) by ENSCO Rail's Comprehensive Track Inspection Vehicle (CTIV) (see Figure 5). The CTIV is a hi-rail vehicle (a road vehicle that can also travel on railway tracks) equipped with several track inspection technologies, including a Track Component Imaging System (TCIS). The TCIS collects images of the trackbed using 4 Basler sprint (spL2048-70km) linescan cameras and a custom line scan lighting solution[2].

The sprint cameras are based on CMOS technology and use a CameraLink interface to stream the data to a rack-mounted computer. Each camera contains a sensor with 2 rows of 2,048 pixels that can sample at line rates of up to 70KHz. The cameras can be set to run in dual-line mode (high-resolution) or in "binned" more, where the values of each pair of pixels are averaged inside the sensor. During this survey, the cameras were set up in binned mode so, each camera generated a combined row of 2,048 pixels at a line rate of 1 line/0.43mm. The sampling rate was controlled by the signal generated from a BEI distance encoder mounted on one of the wheels. The camera positions and optics were selected to cover the whole track with minimal perspective distortion and their fields of view had some overlap to facilitate stitching.

The collected images were automatically stitched together and saved into several files, each containing a 1-mile image. These files were preprocessed by ENSCO Rail using their proprietary tie detection software to extract the boundary of all the ties in each image. We verified that the tie boundaries were accurate after visually correcting invalid tie detections. We downsampled the images by a factor of 2, for a pixel size of 0.86 mm. To assess the detection performance under different operating conditions, we flagged special track sections where the fastener visible area was less than 50% due to a variety of occluding conditions, such as protecting covers for track-mounted equipment or ballast accumulated on the top of the tie. We also flagged turnouts so we could report separate ROC curves for both including

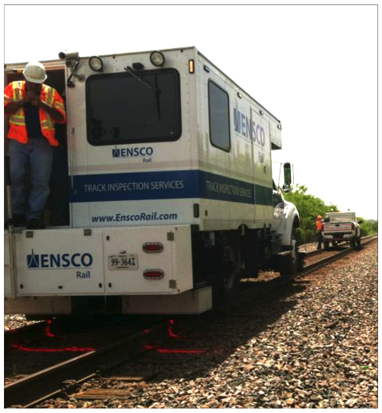


Figure 5. CTIV platform used to collect the images.

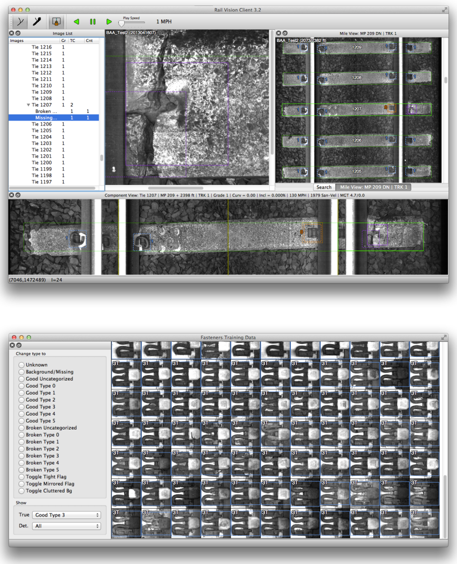


Figure 6. GUI tool used to generate the training set and to review the detection results.

and excluding them. All the ties in this dataset are made of reinforced concrete, were manufactured by either San-Vel or Rocla, and were installed between 1978 and 2010.

Due to the large size of this dataset, we have implemented a customized software tool that allows the user to efficiently visualize and annotate the data (see Figure 6 for a screenshot). This tool has been implemented in C++ using the Qt framework and communicates with the data repository through the secure HTTPS protocol, so it can be used from any computer with an Internet connection without having to set up tunnel or VPN connections. The tool allows the user to change the threshold of the defect detector and select a subset of the data for display and review. It also has the capability of exporting lists of detected defects as well as summaries of fastener inventories by mile.

4.1. Fastener Categorization

On our dataset, we have a total of 8 object categories (2 for broken clips, 1 for PR clips, 1 for e-clips, 2 for fast clips, 1 for c-clips, and 1 for j-clips) plus a special category for background (which includes missing fasteners). We also have 4 synthetically generated categories by mirroring non-symmetric object classes (PR, e, c, and j clips), so we use a total of 12 object categories at test time. The HOG features are extracted using a 160×160 pixel sliding window with a strap of 8×8. We use the HOG implementation in the object detection module of OpenCV using default parameters. For classification, we use the linear SVM implementation in the machine learning module of OpenCV (which is derived from *LIBSVM*[5]) with a soft margin ($C = 0.01$).

For training our detectors, we used a total of 3,805 image patches, including 1,069 good fasteners, 714 broken fasteners, 33 missing fasteners, and 1,989 patches of background texture. During training, we also included the mirrored versions of the missing/background patches and all symmetric object classes. To evaluate the feasibility of the algorithm, we performed 5-fold cross-validation on the training set, where we classified each patch into one of the 9 basic object categories (we excluded the 4 artificially generated mirrored categories). Figure 7 (a) shows the resulting confusion matrix. We only had 14 misclassified samples (0.37% error rate). If we consider the binary decision problem of finding defective fasteners (either missing or broken), we have a detection rate of 99.74% with a false alarm rate of 0.65%. This is an encouraging result, since as explained in section 3.4, our training set has been bootstrapped to contain many difficult samples.

In addition to the proposed method described in Section 3, we have also implemented and evaluated the following alternative methods:

- **Intensity normalized OT-MACH:** As in [1], for each image patch, we subtract the mean and normalize the image vector to unit norm. For each class c, we design an OT-MACH filter in the Fourier domain using $h_c = [\alpha I + (1-\alpha)D_c]^{-1}\overline{x}_c$ with $\alpha = 0.95$, where I is the identity matrix, $D_c = (1/N_c)\sum_{i=1}^{N_c} x_{ci}x_{ci}^*$, and N_c is the number of training samples of class c.

- **HOG features with OT-MACH:** The method in [1], but replacing intensity with HOG feature. Since HOG features are already intensity-invariant, the design of the filters reduces to $h_c = \overline{x}_c$.

- **HOG features with DAG-SVM:** We run one-vs-one SVM classifiers in sequence. We first run each class against the background on each candidate region. If at least one classifier indicates that the patch is not background, then we run the DAG-SVM algorithm [14] over the remaining classes.

- **HOG features with majority voting SVM:** We run all possible one-vs-one SVM classifiers and select the class with the maximum number of votes.

For the first and second methods, we calculate the score using the formulation introduced in sections 3.2 and 3.3, but with $b_c = h_c$ and $f_c = h_c - \sum_{i \neq c} h_i/(C-1)$. For the third and last methods, we first estimate the most likely class in $\mathcal{G}$ and in $\mathcal{B}$. Then, we set S_b as the output of the classifier between these two classes, and S_m as the output of the classifier between the background and the most likely class.

We can observe in Figure 8 (a) that the proposed method is the most accurate, followed by the HOG with OT-MACH method. The other methods are clearly inferior. Figure 7 shows the confusion matrix of our method and the second best method. This method had an error rate of 2.23% (6 times greater than our proposed method). The detection rate was 98.86% with a false alarm rate of 4.02%. We can see that j-clips and c-clips are the most difficult types of fasteners. These 2 types of fasteners contain more intra-class variation than other types because they are placed next to joint bars, so some of them are slightly rotated to accommodate the presence of joint bar bolts.

4.2. Defect Detection

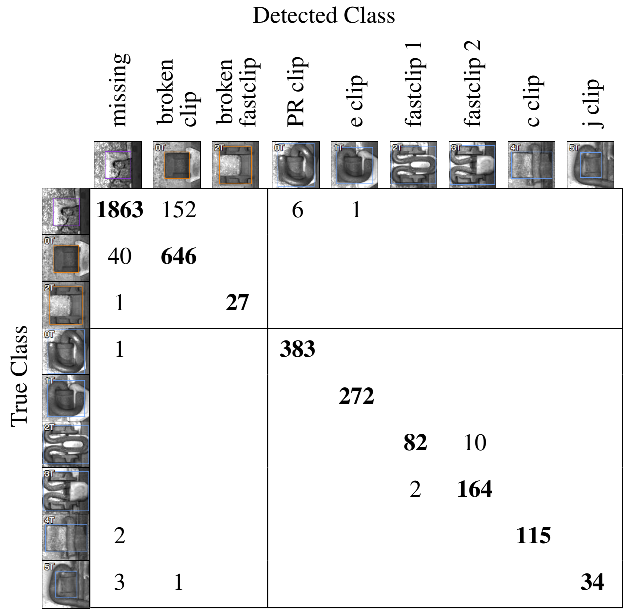
To evaluate the performance of our defect detector, we divided each tie into 4 regions of interest (left field, left gage, right gage, right field) and calculated the score defined by (6) for each of them. Figure 8 shows the ROC curve for crossvalidation on the training set as well as for the testing set of 813,148 ROIs (203,287 ties). The testing set contains 1,051 ties images with at least one defective fastener per tie. The total number of defective fasteners in the testing set was 1,086 (0.13% of all the fasteners), including 22 completely missing fasteners and 1,064 broken fasteners. The number of ties that we flagged as "uninspectable" is 2,524 (1,093 on switches, 350 on lubricators, 795 covered in ballast, and 286 with other issues).

We used the ROC on clear ties (blue curve) in Figure 8 (b) to determine the optimal threshold to achieve a design false alarm rate of 0.1% ($\tau = 0.1614$). Using this sensitivity level, we ran our defective fastener detector at the tie level (by taking the minimum score across all 4 regions). Results are shown in Table 2.

Table 2. Results for detection of ties with at least one defective fastener.

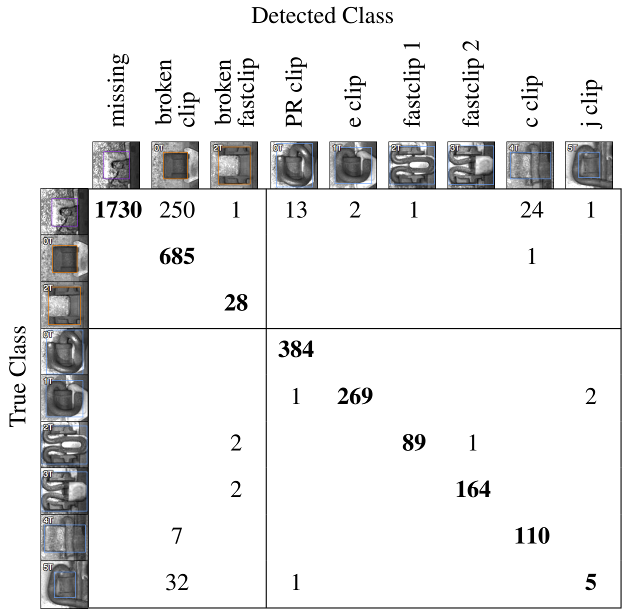
Subset	Total ties	Defective	PD	PFA
clear ties	200,763	1,037	98.36%	0.38%
clear + switch	201,856	1,045	97.99%	0.71%
all ties	203,287	1,051	98.00%	1.23%

Our protocol has been to mark the whole tie as unin-



True Class \ Detected Class	missing	broken clip	broken fastclip	PR clip	e clip	fastclip 1	fastclip 2	c clip	j clip
	1863	152		6	1				
	40	**646**							
	1		**27**						
	1			**383**					
					272				
						82	10		
						2	**164**		
	2							**115**	
	3	1							**34**

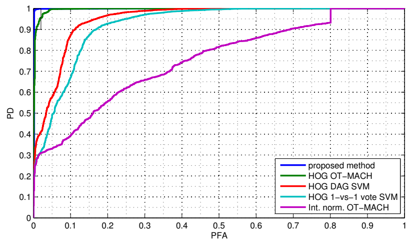
(a)



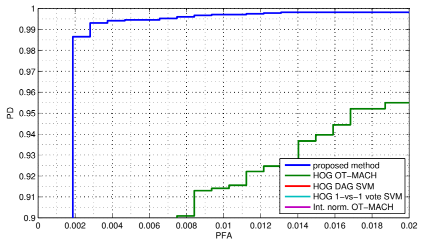
True Class \ Detected Class	missing	broken clip	broken fastclip	PR clip	e clip	fastclip 1	fastclip 2	c clip	j clip
	1730	250	1	13	2	1		24	1
		685						1	
			28						
				384					
				1	**269**				2
			2			**89**	1		
			2				**164**		
		7						**110**	
		32		1					**5**

(b)

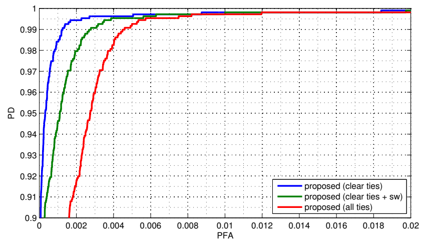
Figure 7. Confusion matrix on 5-fold cross-validation of the training set using (a) the proposed method (b) the method described in [1] with HOG features.



(a)



(a) detail



(b)

Figure 8. ROC curves for the task of detecting defective (missing or broken) fasteners (a) using 5-fold cross-validation on the training set (b) on the 85-mile testing set.

spectable if at least one of the fasteners is not visible in the image. This is not ideal as there are situations where parts of the tie are still inspectable, for example when the field side of the rail is covered with ballast, but the gage side is inspectable (this explains the 6 additional defective ties when including uninspectable ties).

5. Conclusions and Future Work

In order to advance the state-of-the-art in automated railway fastener inspection, our design has been driven by the fundamental principle of projecting the samples into a representation that minimizes intra-class variation while maximizing inter-class separation. To achieve minimum intra-class variation, we use the HOG features, which have built-in intensity normalization, while preserving the distinctive distribution of edges. We have also implemented a sophisticated graphical user interface that facilitates accurate alignment of the fastener locations to avoid intraclass variations due to misalignment. To achieve maximum inter-class separation while maintaining the principle of parsimony, we resort to the maximum margin formulation and simplicity offered by linear SVMs. We further enforce intra-class separation during the sampling of the training data. For the fastener-vs-background classifiers we bootstrapped difficult samples when we built the training set. For the fastener-vs-rest classifiers, we aligned the negative samples to the most confusing position, so the learning machine could focus on the best way to separate classes on the most distinctive parts of the object.

The detector discussed in this paper is based on inner products between feature vectors extracted from image patches and a set of templates. Therefore, the computation cost is the cost of calculating the feature vector plus performing the inner products with each the 2 template vectors of each class. We have chosen the HOG as our feature vector, but other (probably simpler) alternative representations are possible, and may help to dramatically speed-up the computation time without significantly degrading the detection performance. Alternatively, we could speed-up the computation of the inner products by reducing the dimensionality of the feature vector by using Principal Component Analysis (PCA).

Although the approach described here works most of the time and can handle a wide variety of track conditions, including mud splashes, heavy grease, and partial occlusions due to small branches, leaves, pieces of ballast or cables, there is still room for improvement. Currently, the decision is based on an image by image basis, disregarding the statistical dependencies of fastener location, and fastener type between adjacent ties. Adding such dependencies through a Markov model would probably help reduce spurious detection and classification errors. Moreover, specific models for the arrangement of fasteners around switches and other special track structures could be used to reduce the uncertainty in fastener detection that our solution has under such scenarios. In the future, we plan to address some of these issues and extend this framework by adding more types of fasteners and using more robust matching methods. Nevertheless, we believe that the system described here is a big step towards automated visual track inspection and will help railroads maintain their tracks in optimal conditions.

Acknowledgements

The authors thank Amtrak, ENSCO, Inc. and the Federal Railroad Administration for providing the data used in this paper. This work was supported by the Federal Railroad Administration under contract DTFR53-13-C-00032.

References

[1] P. Babenko. *Visual inspection of railroad tracks*. PhD thesis, University of Central Florida, 2009. 2, 3, 6, 7

[2] Basler AG. Success story: ENSCO deploys Basler sprint and ace GigE cameras for comprehensive railway track inspection. http://www.baslerweb.com/linklist/9/8/3/6/BAS1110_Ensco_Railway_Inspection.pdf, Oct 2011. 5

[3] A. Berry, B. Nejikovsky, X. Gibert, and A. Tajaddini. High speed video inspection of joint bars using advanced image collection and processing techniques. In *Proc. of World Congress on Railway Research*, 2008. 2

[4] L. M. Camargo and J. R. Edwards. Emerging condition monitoring technologies for railway track components and special trackwork. In *ASME/ASCE/IEEE Joint Rail Conf. & Internal Combustion Engine Spring Tech. Conf.*, 2011. 3

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 6

[6] J. Cunningham, A. Shaw, and M. Trosino. Automated track inspection vehicle and method, May 2000. US Patent 6,064,428. 2

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR), 2005 IEEE Computer Society Conference on*, volume 1, pages 886–893, Jun 2005. 3

[8] P. De Ruvo, A. Distante, E. Stella, and F. Marino. A GPU-based vision system for real time detection of fastening elements in railway inspection. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2333–2336. IEEE, 2009. 2

[9] X. Gibert, A. Berry, C. Diaz, W. Jordan, B. Nejikovsky, and A. Tajaddini. A machine vision system for automated joint bar inspection from a moving rail vehicle. In *ASME/IEEE Joint Rail Conference & Internal Combustion Engine Spring Technical Conference*, 2007. 2

[10] X. Gibert, V. M. Patel, D. Labate, and R. Chellappa. Discrete shearlet transform on GPU with applications in anomaly detection and denoising. *EURASIP Journal on Advances in Signal Processing*, 2014(64):1–14, May 2014. 2

[11] Y. Li, H. Trinh, N. Haas, C. Otto, and S. Pankanti. Rail component detection, optimization, and assessment for automatic rail track inspection. *Intelligent Transportation Systems, IEEE Trans. on*, 15(2):760–770, April 2014. 2

[12] A. Mahalanobis, B. V. K. V. Kumar, S. Song, S. R. F. Sims, and J. F. Epperson. Unconstrained correlation filters. *Appl. Opt.*, 33(17):3751–3759, Jun 1994. 2

[13] F. Marino, A. Distante, P. L. Mazzeo, and E. Stella. A real-time visual inspection system for railway maintenance: automatic hexagonal-headed bolts detection. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Trans. on*, 37(3):418–428, 2007. 2

[14] J. C. Platt, N. Cristianini, and J. Shawe-taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Systems*, pages 547–553. MIT Press, 2000. 3, 6

[15] T. Podder. Analysis & study of AI techniques for automatic condition monitoring of railway track infrastructure. Master's thesis, Dalarna University, Computer Engineering, 2010. 3

[16] E. Resendiz, J. Hart, and N. Ahuja. Automated visual inspection of railroad tracks. *Intelligent Transportation Systems, IEEE Trans. on*, 14(2):751–760, Jun 2013. 2

[17] H. Trinh, N. Haas, Y. Li, C. Otto, and S. Pankanti. Enhanced rail component detection and consolidation for rail track inspection. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 289–295, 2012. 2

[18] M. Trosino, J. Cunningham, and A. Shaw. Automated track inspection vehicle and method, Mar 2002. US Patent 6,356,299. 2

[19] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition (CVPR), 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001. 2