ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ADDIS ABABA, ETHIOPIA

Automatic Railway Track Element Detection for East-West Route of Addis Ababa LRT

August 2015

Melaku Tegegne

A Thesis Submitted to School of Electrical and computer engineering in Partial Fulfillment of the Requirements of the Degree of Master of Science in electrical and computer engineering

Certificate of Approval

We hereby declare that this thesis is from the student's own work and effort, and all other sources of information used have been acknowledged. This thesis has been submitted with our approval.

Approved by Board of Examiners

_____	_____
Chairman	Signature
_____	_____
Advisor	Signature
_____	_____
Internal Examiner	Signature
_____	_____
External Examiner	Signature

Declaration

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Where other sources of information have been used, they have been properly acknowledged.

Name: Melaku Tegegne Abitew

Signature: _____

Place: Addis Ababa, Ethiopia

Date: August 2015

I certified that the above statement made by the student is correct to the best of my knowledge and it has been submitted for examination with my approval as university advisor.

Name: Dr. Yalemzewd Negash

Signature: _____

Date: August 2015

Abstract

In railways, regular maintenance and inspection of railway track is quiet commonly used approach to maintain an acceptable level of safety. Such approach is currently done in most railway companies; ERC will not be a different, using a very laborious, lengthy and manual work by visually inspecting and visiting test sites with a help of trained operators. Another approach is to have an automatic train control system where a train vehicle is completely controlled by a central control system without the need of human driver intervention since computer-aided machines are less likely to make mistakes than humans once carefully designed and installed.

In this thesis, an automated system is presented which mitigates the said problem by detecting a railway track elements (such as fasteners) from a real railway track image taken by a digital camera that is mounted in the train. Based on the level of difficulty in the detection problem, suitable combinations of image processing and pattern recognition methods are applied to achieve high performance automated detection. The system is primarily based on the Viola-Jones object detection framework Viola and Jones, but it selects best new Haar wavelet-like features more convenient to the problem of Addis Ababa Light rail transit track fasteners detection. The performance of the system is improved by employing an appropriate image pre-processing and performance evaluation techniques. Accordingly, the system has managed to achieve a 94.6% detection rate of performance and has a failure rate of 0.11%.

Acknowledgements

I am very thankful to Dr. Yalemzew Negash for his generous support and continuous guidance and advice. I am also grateful to those who directly and indirectly put their contribution to the success of this dissertation.

Contents

List of Figures Page

List of Tables page

Chapter 1

1 Introduction

Key words: *Classifier, Haar wavelet-like, object detection, boosting, learning algorithm*

1.1 Problem identification

Railway safety is very essential component of a railway system. To maintain an acceptable level of safety, railway companies consider different kinds of approaches. Among all, quiet commonly used approach is to conduct regular maintenance and inspection of railway track. Because loose bolts and detached fasteners of the rail are potential dangers for derailments and other accidents. Such approach is currently done in most railway companies; ERC will not be a different, using a very laborious, lengthy and manual work where trained human operators maintain and inspect by visiting test sites.

Another approach is to have an automatic train control system where a train vehicle is completely controlled by a central control system without the need of human driver intervention. This is primarily motivated by the fact that computer-aided machines are less likely to make mistakes than humans once carefully designed and installed although costly and complex.

This thesis presents an automated system which mitigates the said problems in two ways. First, it presents a system which automatically detects the presence of railway track elements (for instance the presence of a fastener) and hence avoids the need for intensive labor work and subjectivity in maintenance. Secondly, object detection is one integral part of an automatic train control system. Therefore, the object detection scheme presented by the system will be facilitating the development of an automatic train control system as the introduction of such system is essential for ERC.

Object detection is one of the toughest problems in the area of computer science and engineering. To tackle the problem, many algorithms and methods has been developed by the computer vision society. Among the entire approaches, the Viola-jones object detection framework is proven to be robust, fast and suitable for real-time applications.

Hence, the system will highly depend on this framework to implement an automatic detection of railway line elements. Furthermore, an effective image processing techniques (filtering, image enhancement, de-noising, cropping and etc.) and pattern recognition approaches will be considered.

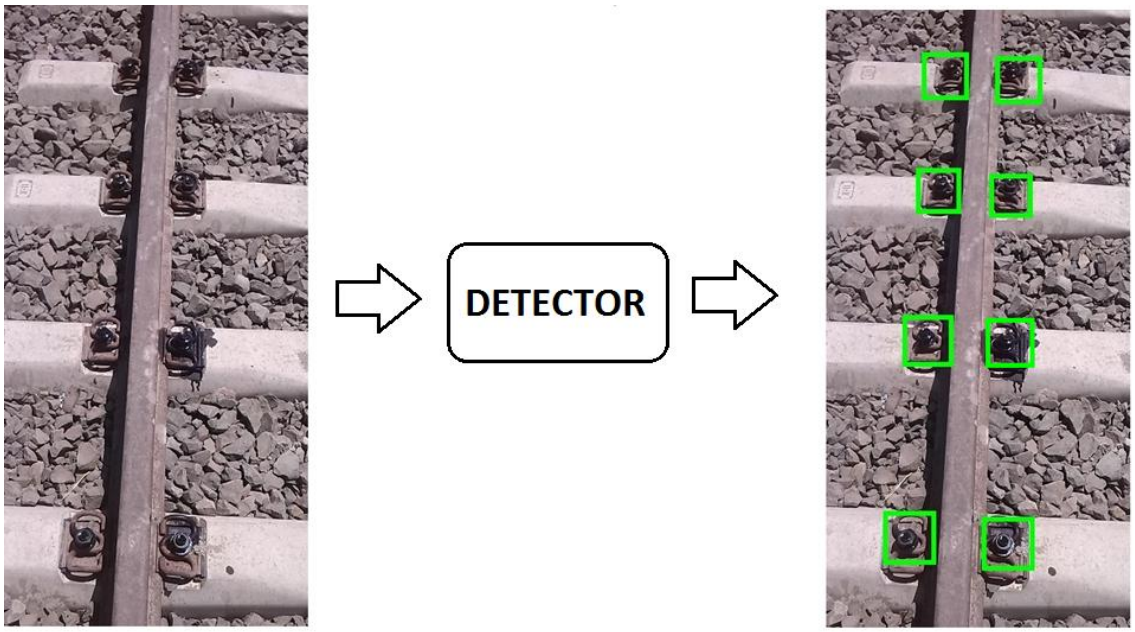
1.2 Objectives

In this dissertation, an automatic railway track element detector software system is developed. Its main objective is to detect the presence of rail track elements, such as fasteners, whether they are present or not in an image of railway track taken by a digital camera mounted on a train. This is done by scanning the image and look for elements of interest like fasteners, by applying a new learning algorithm, namely the Viola-Jones object detection framework Viola and Jones [1], which was proved to be fast and robust for face detection and can be suitably applied for a general object detection [1] like a railway track elements detection system. Hence the main objective of thesis is:

- Developing a software system which automatically detects the presence of a railway track element from captured railway track image

1.2.1 Specific objectives

- To mitigate the time and resource wasted in railway track maintenance and inspection
- To facilitate automatic railway track maintenance and inspection
- To facilitate the development of automatic train control system
- To implement and understand Viola-jones object detection framework



a) Input image b) Output image

Figure 1. 1: Sample detection of the system

Thus, the contribution of thesis is the development of a software system based on application of viola-jones object detection framework. The performance of this system is evaluated in terms of true positive and false positive rates based on test result that will be discussed in last chapter.

1.3 Thesis Scope

This thesis study is confined to development of a software system with capabilities of detecting the presence of railway tack element (i.e. fasteners), solely from a given railway track images. The images are taken using Huawei Ascend Y300 cellphone camera with a resolution of 8 Mega pixel (MP) for test and demonstration purposes.

The thesis addresses types of fasteners that are currently installed in Addis Ababa light rail transit (LRT) that runs from Ayat to Torhayloch, but they can be suitably applied to any type of objects to be detected in railway track images if sufficient training examples are considered.

Chapter 2

2 Literature review

This chapter reviews basic issues and concepts that have been around in the literature in the areas of image processing and general object detection methods, algorithm and applications.

2.1 Addis Abeba LRT existing railway track

On the design plan of Addis Abeba light rail transit, automatic train control and automatic railway track maintenance are not stated. Hence, it is expected that the LRT is going to be maintained using a trained human operators [2]. However, manual railway track maintenance has disadvantages due to it slowness, wastage of resource and human-made errors. It is useful to Ethiopian railway corporation (ERC) if an automatic railway track element detector is developed as it helps to detect rail defects and to increase the ability to detect defects and reduce the maintenance time.

Besides, railway track information is very crucial for different variety of control system in railway. For instance, signaling and control system, automatic train control systems are one of them. Hence, the developed detector can be feed significant information for the signaling system. It can also help and consolidates the development of automatic train control system because the introduction of such systems is inevitable for ERC.

Figure 2. 1: Addis Ababa LRT fastening element, Tension clamps

This dissertation targets to build an automatic railway tack element detection software system, which is able to automatically detect the presence of fasteners (particularly tension clamp fasteners (see figure 2.1 above) that are installed in Addis Ababa LRT) from railway track image. This system should be able to detect various objects such as fastening elements (bolts, clamps, clips, etc. if proper training data are available however in this thesis only tension clamp fasteners are considered) by scanning the images taken by a digital camera mounted under a train. Figure 2.1 shows Addis Ababa LRT fastening element, Tension clamps elements that will be considered.

In scientific community, object recognition from two-dimensional images is fundamental problem and has greatly been studied. Such a problem is tackled using two approaches, non-learning-based and learning-based approaches [3], non-learning based approaches employs a pre-designed rigid geometric models to represent the object to be detected. However, railways tracks are characterized by very rough environment and these methods are not effective in detecting objects of interest under different conditions [4]. Due to the challenging problems in great variety in lighting, viewing directions, sizes makes modeling these objects very difficult.

The second approach includes learning-based techniques. These techniques employ properly pre-labeled training sets to automatically learn a function that will be able to classify image sub-windows and therefore detect the desired objects [3]. Such methods are effective for generic shapes because they don't require any geometrical model knowledge of the desired object. Hence, in the thesis this approach is applied for this reason.

2.2.3 Automated Detection of railway track elements

Automated visual inspection is non-destructive technique causing no damage to the material to be evaluated. Unlike destructive techniques, this does not need specific devices[4]. Thus, this technique relies to a big extent on classification algorithms in order to detect objects like fasteners. At present, automated visual inspection systems are typically used to measure rail profile (Magnus[6] , Bachinsky[7]).

Rail inspection cars have been created in order to automate the analysis of railroad data and to meet today's high mileage inspection needs. They are basically their own train with inspection equipment on board. The devices are mounted on carriages located underneath the inspection car. These inspection cars are loaded with high speed computers using advanced programs which recognize patterns and contain classification

information. Systems capable of recording track geometry have been developed for rail-road cars.

Automated visual inspection systems can operate at very high velocities, and offer an alternative to slow, expensive and erratic human inspection. Therefore, this technique is particularly suitable for high-speed tasks required by railway maintenance.

Furthermore, advances in technology have resulted in more powerful and cheaper image-analysis equipment, as well as more effective algorithms in pattern recognition, image processing and artificial intelligence areas. These advances permit automated visual inspection systems in production environments, and may meet industrial requirements.

The applicability of this technique is nevertheless restricted to the detection of surface features only. Therefore, other techniques (such as ultrasound inspection) must be used if internal defects are to be detected.

2.3 Fundamentals of object detection

A digital image can be considered as a spatial two-dimensional mathematical signal (f(x, y)) representation of some physical phenomenon [5]. Where x and y are spatial coordinates whereas the value of the function represents light intensity values. Images are acquired by a digital camera. Object detection algorithms and pattern recognitions system play around and process these values to achieve their goal of object detection.

2.3.1 Scanning, feature extraction and classification

In object detection, the main task is to identify, in given image, image areas sub-windows that contain the patterns to be detected. For instance, an image of typical human, the task is to locate the face, eye or other parts of the body. To reach this goal, a basic method consists in exhaustively sliding a subwindow on a captured image. This is termed as scanning. Pixel values contained in each scanned subwindow are processed with a feature extraction algorithm, and then provided to a classifier [5].

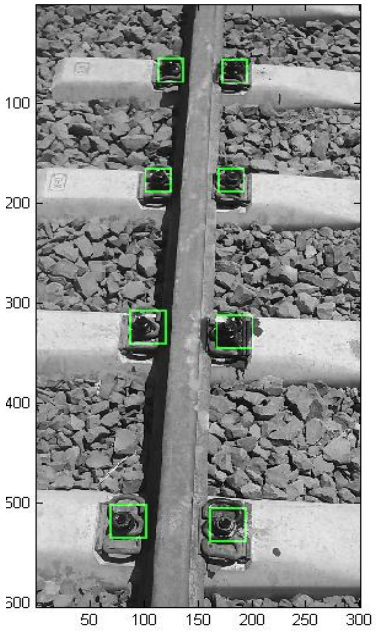


Figure 2. 2: Detected fasteners and their locations in a railway track image.

Figure 2.2 shows a real image of AA LRT railway track taken by a digital camera around hayahulet. The green-marked subwindows show the locations of detected fasteners.

Feature extraction is extracting a relatively representative set of data from data (the sub-windows) that will be processed by the classifier, while maintaining important information about these data. Classification is the problem of identifying to which of a set of categories a new sub-windows belongs, on the basis of features of the training set of data whose category membership is known. Therefore, different object detection algorithms can differ in the choice of a feature extraction algorithm and a classifier [3]. Object detection approaches are widely categorized into two groups namely, non-learning-based approaches and learning-based approaches.

2.3.2 Learning and Non-learning-based approaches

A Non-learning-based approach employs a pre-designed rigid geometric model to represent the object to be detected. These methods are hard to extend to complex objects, as it requires significant amount of prior information and domain knowledge to build up a geometric model. These systems are likely to suffer from restrictive

assumptions on the scene structure. They are difficult to apply to rail inspection because railways represent a rough environment in which variety in lighting and texture poses a challenging problem for object modeling.

However, the specificity of learning-based approaches is the use of a training set of data (subwindows), whose actual class (or label*)* is known a priori. This training set is presented to the classifier, which automatically adjusts its internal parameters to minimize some measure of the error between the estimated class and the actual label. Learning-based methods avoid difficulties in modeling objects by considering examples of that object under various conditions. Thus, a first human contribution is needed to build up a training set of data, in which each data item is assigned a label.

2.3.4 Learning-based object detection Techniques

Most learning-based general object detection frameworks have been widely developed for the specific problem of face detection. For instance, Papageorgiou and others [13] process data using a Haar wavelet-like representation, which is used as an input to a Support Vector Machine (SVM) classifier. Whereas Sung and Poggio [14] introduced processing and image using a mixture of Gaussian model which is used as an input to a Multilayer Perceptron (MLP) classification.

Rowley and others [15] proposed an image processing through an extensive stage like lighting correction and histogram equalization and before they are classified with retinally connected neural networks. Schneiderman and Kanade [16] use multi-resolution information for different levels of wavelet transform. A nonlinear face and non-face classifier is constructed using statistics of products of histograms computed from face and non-face examples using AdaBoost learning Freund and Schapir [17]. Although this computation is relatively slow it can detect profile views.

A very fast and robust object detection system in which AdaBoost learning is used to construct a nonlinear (‘strong’) classifier was proposed by Viola and Jones [1]. In this paper, an AdaBoost is used to construct weak classifiers based on simple scalar Haar wavelet-like features, and to boost them to construct a strong classifier. Viola and Jones made three fundamental contributions for fast computation of a large number of features. These include efficient image representation technique called image integral, a modified adaboost and an attention cascade of classifiers makes the computation even more efficient, allowing background regions of the image to be quickly discarded while spending more computation on object-like regions. The detector developed in this thesis is highly based on this method and techniques developed by Viola and Jones [1].

Chapter 3

3 Methodology

A digital camera mounted on a train is used to capture images of the railway track. These images are processed using the most versatile development environment MATALB centrally depending on viola-jones object detection framework. The developed detector scans and processes these images, and detects fasteners. To achieve this, many algorithms and routines have been developed. Considering the time they take and their level of difficulty some of them are written in C programming language and the remaining using MATALB. Hence using the MATALB C-interface, all have been intergraded a single detector. The output of the detector can be used for conditioning, decision and control purposes and it can also be provided for other control and signaling systems. The whole detector diagram is shown as in figure 3.1 blow.

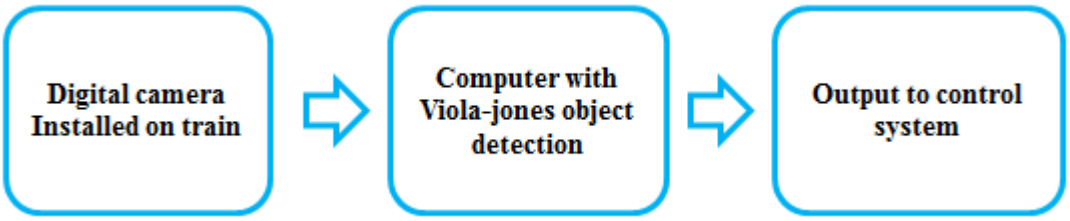


Figure 3. 1: Detector block diagram

However, the whole process of development of automatic railway track element detector can be seen as a composition of three main system components as illustrated in figure 3.2 below. These include an image pre-processing, main image processing and post-processing. In the following sections, each of the main detector components and stages will be descripted.

Figure 3. 2: Main detector components

3.1 Image pre-processing

The performance of the detector depends on the amount of computation time it spends on the captured raw input images. These raw input images contain both wanted and unwanted area of interest; in addition they are subjected to variation in lighting, scale, orientation and alignments. Hence, the major goal of this stage is to pre-process and prepare these images of railway track and make it ready and suitable for further processing so that the performance of the detector is improved.

The first step considered is an image cropping which involves the process of removing and cutting unwanted areas from the input images so that the detector would only waste its time on the wanted areas. Correction variations in lighting conditions have been practiced by many object detection algorithms for making the classification task easier and more accurate. This is accomplished by image standardization process. So image standardization is also considered.

Finally, preparing dataset is done as part of this process. In this process, positive and negative instances of the fasteners to be detected are properly extracted and labeled with correct output by hand. For instance, by cutting of the fastener image from the railway track giving them as the positive sample image and those that don't represent fasteners as negative sample images. These are called the training dataset. However, due care must be taken in preparing a very good training dataset as the performance of the classification highly depend on them.

3.2 Main image processing

In this stage, a pre-processed image is scanned through a fixed-size sliding window to determine every potential localizations of the instances of the object to be detected. It is subsequently fed to a pattern recognition scheme which applies the viola-jones object

detection framework. The pattern recognition scheme involves training and testing phases. In training phase, the training datasets which features are extracted from are used to train a classifier through learning algorithm. In testing stage, this classifier is applied to predict the class of test data whether it is fastener or not.

Hence, the main image processing comprises of two main stages: The first involves determining the entire possible localizations of the fasteners to be detected. Such localisations delimit instance images, called subwindows, which may represent instances of objects to be detected. This problem is generally solved by considering every possible location in the digital image. This process is called image scanning.

The second stage is the task of classifying each sub-window, effectively as an instance of objects to be detected or not. The second problem must be tackled with advanced image-analysis algorithms capable of assigning an output class (or label) to a given input instance. This thesis will address this problem with a learning-based algorithm, called pattern recognition algorithm, which generates a model (or classifier) based on train data set.

3.2.1 Image scanning

Image scanning is the process of locating each and every possible locations of a given size in a digital image. This is often done by sliding a fixed sub window on the digital image. Subsequent locations are obtained by shifting the sub window from left to right, and from top to bottom until all possible locations are reached.

Figure 3. 3: Illustrating the process of scanning an Image of rail

Figure 3.3 illustrates a scanning process on a rail track image. The size of the sliding subwindow is determined by the size of the objects to be detected within the digital image. However, the total number N of possible locations depends on both the size of the image and the size of the subwindow, is given as :

$$N = (W_i - W_s)*(H_i - H_s)$$

Where W_i is width of the image and W_s is width of the subwindow
H_i is height of the image and H_s is height of the subwindow

For instance, the camera used for this thesis has a pixel resolution of 1944×2592 pixels and assuming object instances of 100×100 pixels: the total number of possible locations will be: there are 4,595,248 possible subwindow locations in a given rail track image though prohibitively large. For each location, the subwindow is analyzed by a pattern recognition algorithm, which will classify it as an instance of fasteners to be detected or not.

3.2.2 Pattern recognition

Pattern recognition is a learning-based approach that maps input instances to output labels, based on empirical data. The problem of classification is addressed with a supervised pattern recognition algorithm, that is, in which a set of training data has been provided. The training dataset consists of a set of instances that have been properly labelled 'by hand' with the correct output. It contains both positive data, which are instances of objects to be detected, and negative data, which are instances of objects not to be detected. Figure 3.4 shows some examples of positive and negative examples in the context of rail inspection. The learning algorithm then processes certain characteristics (or *features*) of the training data to generate a classifier.

(a) Positive data. (b) Negative data.

Figure 3. 4: An example of labelling positive and negative data of rail fasteners.

The pattern recognition scheme comprises two main stages:

1. Training stage: is the task of forming a classifier from the training dataset.
2. Testing stage: consists in applying the generated classifier to a new input instance (or *test data*) in order to classify it, that is, predict its class (either positive or negative).

Thus, the classifier generated at the training stage will be used to actually classify every given input instance. Note that the training stage is performed only once, while the testing stage is performed for each input instance to be classified.

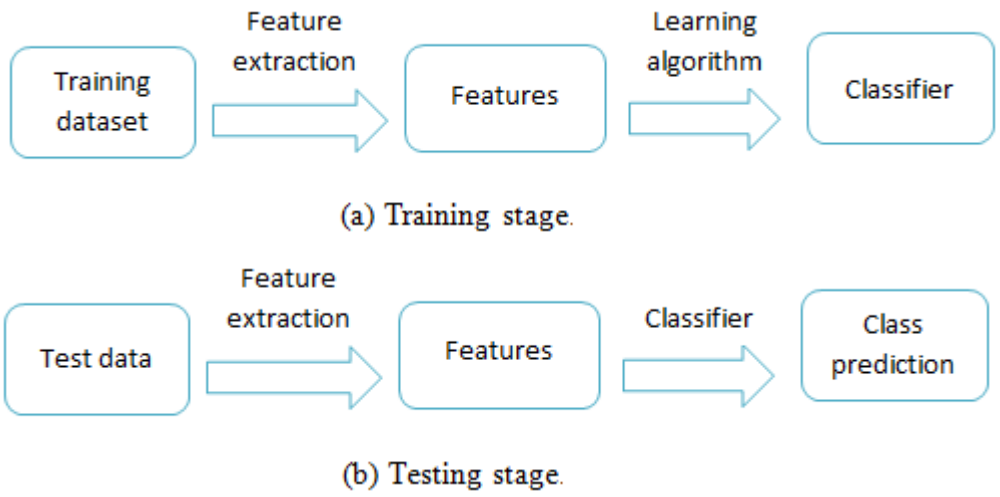


Figure 3. 5: Stages of pattern recognition

Figure 3.5a and Figure 3.5b shows respectively the training stage and the testing stage. During the training stage, a feature extraction algorithm is first applied to the training data in order to extract their features, which then will be processed by a learning algorithm to generate a classifier. During the testing stage, the same feature extraction algorithm is first applied to a new test data in order to extract its features, which then will be processed by the generated classifier to predict its class.

3.2.3 Viola-jones object detection principles

Viola and Jones [1] have made three main contributions to the pattern recognition approach by introducing the following techniques:

- Integral image technique to efficiently compute Haar wavelet-like features
- Boosting algorithm, called AdaBoost, as a learning algorithm,
- Cascading strong classifiers

In following section, each of these techniques will be discussed.

3.2.3.1 Feature extraction and haar wavelet-like features

Feature extraction consists in extracting certain properties of the training data that will be processed by the classifier. The main purpose of using features rather than the raw data (raw pixel values) lies in the fact that features can encode knowledge about the data, which is difficult to learn from the raw data. Thus, feature extraction helps to increase the generalization power [5] of the classification. The other motivation for using features is to reduce the size of the data that will be processed by the classifier, thus reducing the computation time of the classification.

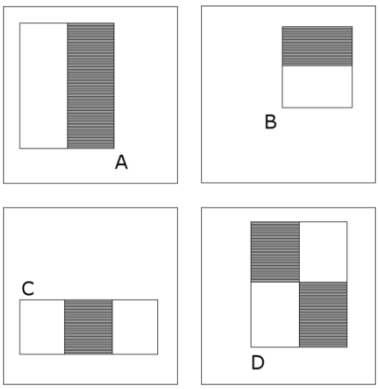


Figure 3. 6: Four Haar wavelet-like features

Four Haar wavelet-like features are shown relative to their enclosing subwindows. The sum of the pixels which lie within the white rectangles is subtracted from the sum of the pixels in the grey rectangles.

Viola and Jones [1] propose four basic types of scalar features, called Haar wavelet-like features. These features are reminiscent of Haar wavelets, which have been developed for basis functions to encode signals. The objective of these features is to collect local oriented intensity information at different scales and locations for representing image patterns.

These features can be represented by rectangular blocks located in sub-regions of a subwindow. These rectangular blocks can vary in shape, size, and location inside the subwindow. The value of a Haar wavelet-like feature is the difference between the sums of the pixels within rectangular regions of these blocks. Figure 3.6 shows some examples of the four basic Haar wavelet-like features.

Since Haar wavelet-like features provide a rich image representation, which supports effective generalization capabilities of the classification and can be efficiently computed using integral image technique; interesting for three reasons: they are also very important because they can be easily interpreted as local oriented intensity information at specific locations of an image.

3.2.3.2 Integral image technique

The integral image is a kind of linear transformation on an input image. It is actually an intermediate representation for the training data that is used to compute Haar wavelet-like features very rapidly. As defined by Viola and Jones [1], the *integral image* at location (*x, y*) contains the sum of the pixels above and to the left of (*x, y*):

$$II(x,y) = \sum_{x' \le x, y' \le y} I(x', y')$$

Where $II(x, y)$ is the integral image and $I(x, y)$ is the original training image.

The image can be computed in one pass over the original image using the following pair of recurrences:

$$S(x, y) = S(x, y-1) + I(x, y)$$
$$II(x, y) = II(x-1, y) + S(x, y)$$

Where S(x, y) is the cumulative row sum, $S(x, -1) = 0$ and $II(-1, y) = 0$.

Thus, any rectangular sum of pixels in the original image can be computed in four references in the integral image, as illustrated in Figure 3.7 This technique enables to avoid re-computing the sum of every pixel within a rectangular region of a Haar wavelet-like feature, for each rectangular region considered.

However, given a training image of resolution of only 24×24 pixels, the total number of Haar wavelet-like features is very large (approximately 160,000), far larger than the number of pixels in the raw training image. Therefore, the learning process must select only a small number of these Haar wavelet-like features. Viola and Jones [1] propose to perform this feature selection process in the learning algorithm, while training the classifier.

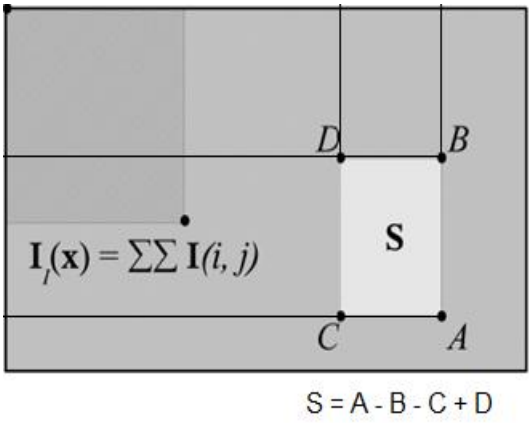


Figure 3. 7: Computing an integral image

The sum of the pixels (of the original image) within rectangle *S* can be computed as (*A* + *D*) − (*B* + *C*).

3.2.3.3 Adaptive Boosting as a learning algorithm

The AdaBoost learning procedure Freund and Schapire [17] consists of iteratively training weak classifiers, with respect to the distribution of the training data, and combining them into one strong classifier. A weak classifier refers to a simple learning algorithm that is not expected to classify the training data with a good performance (for example, even an accuracy of 51% may be acceptable). A strong classifier refers to the weighted combination of weak classifiers, and is expected to classify the training data with a good performance (for example with accuracy of 95%). The primary objective of

AdaBoost is therefore to boost the classification performance of a simple learning algorithm.

AdaBoost is a multi-stage learning procedure that incrementally builds a *strong* classifier by training each new weak classifier to emphasize the training data that previous weak classifiers misclassified. At each round, a new weak classifier is trained, then the training data are re-weighted (misclassified data gain weight and correctly classified data lose weight), so that future weak classifiers focus more on the data that previous weak classifiers misclassified.

Viola and Jones [1] use AdaBoost both as a feature selection process and as a powerful training algorithm. The idea is that only a very small number of important Haar wavelet-like features can be selected to form an effective classifier. The exhaustive set of Haar wavelet-like features is indeed too large to be computed by the learning algorithm.

To achieve the feature selection process, a weak classifier is constrained to depend on a single feature. For each feature, the weak classifier is defined as the optimal threshold classification function, such that the minimum numbers of examples are misclassified. Thus, a weak classifier $h_{f,\theta,p}(x)$, classifying an example x, consists of a feature f , a threshold θ and a polarity $p \in \{-1, +1\}$, such that:

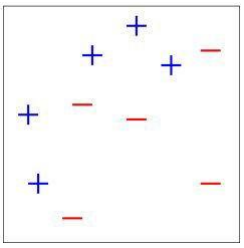
$$h_{f,\theta,p}(x) = p \cdot sign(f(x) - \theta)$$

At each round of the AdaBoost learning algorithm, the single weak classifier (and so the single corresponding feature) that best separates (with the lowest weighted classification error) the positive and negative examples is selected.

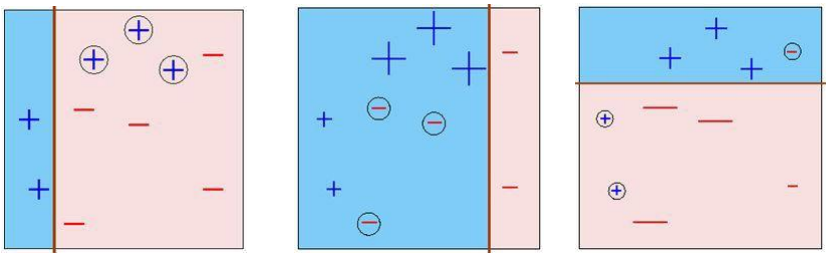
No single weak classifier can usually achieve a low error. However, after linearly combining several weak classifiers into a single strong classifier, the latter can effectively reach a very low error. Schapire and others [18] formally demonstrate the ability of a strong classifier to achieve low training error and good generalization performance. Figure 3.8 illustrates the AdaBoost learning procedure.

3.2.3.4 Cascading strong classifiers

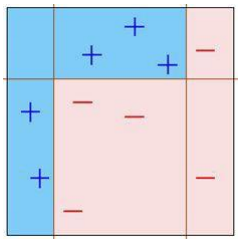
A boosted strong classifier effectively eliminates a large portion of negative subwindows while maintaining a high detection rate. Nonetheless, a single strong classifier may not meet the requirement of an extremely low false positive rate.



a) Initial distribution



a) Round 1. (b) Round 2. (c) Round 3.



(d) Strong classifier.

Figure 3. 8: AdaBoost learning procedure.

The training data are distributed according to two features (horizontal and vertical axes). Positive data are in blue (+), negative data are in red (-).

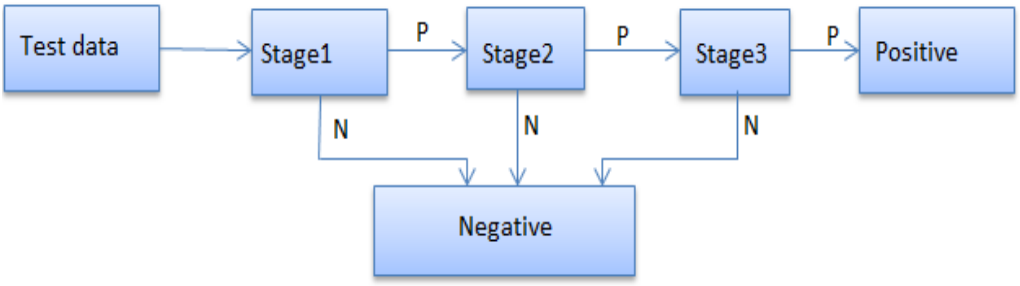


Figure 3.9: Cascading classifiers

Each stage corresponds to one strong classifier, which classifies the test data either as positive (P) or negative (N).

Several strong classifiers cooperate to provide a solution. This leads to the concept of a cascade of strong classifiers, as illustrated in Figure 3.9. Each test data that fails to pass a strong classifier is not further processed by subsequent strong classifiers.

The motivation behind the cascade of classifier is that simple classifiers at early stage can find out most negative examples efficiently, and stronger classifiers at later stage are necessary only when dealing with instances that are likely to be positive. Taking into account that the vast majority of the subwindows in a real image are negative, this strategy can significantly speed up the detection and reduce false positives, with a little sacrifice of the true positive rate.

Chapter 4

4 Detector implementation

The detector is developed using the most versatile development environment MATALB centrally depending on viola-jones object detection framework [1]. To achieve this, many algorithms and routines have been developed. Considering the time they take and their level of difficulty some of them are written in C programming language and the remaining using MATALB. Hence using the MATALB C-interface, all have been intergraded a single detector.

Hence, this chapter outlines about the details of implementation of the detector. Although many routines and functions have been developed, only the algorithms, methods and techniques related to main components of the detector software will be addressed. Hence the following key components will be discussed:

- Image cropping
- Image standardization
- Labeling dataset,
- Generating negative and positive dataset
- Haar feature designer
- Haar Feature listing
- Haar features computation
- Haar adaptive boosting weak learner
- Cascading strong classifiers

Before applying all these process, a digital image has to be firstly read into MATlab. A digital image can be considered as a spatial two-dimensional mathematical signal (f(x, y)) representation of some physical phenomenon [5]. Where x and y are spatial coordinates whereas the value of the function represents light intensity values. Images are acquired by a digital camera. Similary matlab considers digital images a two-dimensianl matrix each entry representing the intensity values. The built in function used to read images in marlab is as follows:

nvariable = Imread(“imfilename.exe”)

where nvariable is a matlab variable(two dimensional variable) and imfilename is imput image file name and extension type. Imread is a built-in matlab function to read images.

For example the following input image:

Show MATLAB's image representation of the above' image after read with MATLAB and scaled to 24x24 pixel

184	203	225	196	156	131	102	107	129	122	126	125	121	130	131	127	135	143	118	126	84	52	31	40
183	206	129	119	128	120	103	103	100	102	101	102	107	111	122	125	117	112	105	123	136	125	55	37
73	105	112	98	79	50	37	32	32	26	25	25	23	23	26	30	55	65	126	39	95	146	105	30
83	119	113	110	125	141	145	154	159	162	169	175	174	170	174	174	161	132	60	27	63	120	99	29
110	139	139	140	155	144	156	157	143	159	151	160	156	157	145	154	156	164	187	27	73	145	99	30
105	125	154	143	139	146	143	157	160	162	156	144	162	158	147	144	154	163	181	25	80	133	94	29
73	140	144	139	138	144	148	148	175	157	124	115	126	94	85	55	56	63	128	21	63	92	75	31
65	82	131	141	145	141	93	74	53	31	43	51	64	46	82	61	53	63	62	46	76	48	36	40
167	33	48	82	107	84	36	50	76	93	71	74	82	102	70	69	58	88	60	82	112	97	38	38
167	172	108	59	39	37	56	106	63	56	64	55	59	57	108	81	39	53	59	36	107	124	93	26
165	170	140	45	44	41	101	84	59	116	117	89	47	65	43	133	74	255	51	20	69	123	123	22
160	160	31	54	40	38	93	82	80	150	109	74	43	55	55	90	23	83	64	56	39	129	134	18
166	96	48	52	43	34	39	109	88	53	58	43	52	78	85	23	48	37	120	52	107	132	129	21
178	32	56	48	43	39	34	34	51	86	89	78	85	58	29	38	41	42	87	60	119	136	103	22
167	132	32	73	58	42	37	35	34	27	28	29	28	32	38	55	92	83	75	62	104	103	31	32
150	143	117	138	131	81	56	42	42	47	43	46	42	40	42	43	38	41	37	84	126	39	30	34
96	123	135	103	91	87	80	73	63	57	74	74	49	43	46	47	48	103	170	159	129	77	74	33
117	136	126	128	160	168	171	178	171	164	154	176	95	63	69	73	62	75	125	104	41	111	137	26
137	136	146	156	161	164	170	162	156	154	153	157	153	98	80	70	108	143	141	98	30	95	116	31
144	145	142	167	159	139	122	165	157	161	157	156	158	160	159	153	167	165	168	117	19	100	136	31
127	125	108	128	159	139	111	161	147	161	160	148	145	144	146	140	133	125	128	89	83	110	134	39
67	124	144	110	111	109	94	80	81	58	34	41	41	42	39	62	78	92	113	118	128	111	123	30
44	57	95	131	133	140	148	138	142	143	63	74	75	73	70	136	148	128	118	120	108	80	41	39
56	53	47	41	53	66	63	77	81	90	91	68	56	59	91	98	78	65	61	56	33	35	37	39

All the images processing techniques will manipulate these images pixels to achieve the goals of detection:

4.1 Image cropping

Cropping an image is cutting out unwanted areas from the image. These unwanted areas are those parts of the image which certainly doesn't contain any of the fasteners to be detected. Hence, these unwanted areas are not required to be processed by the object detector so that it can only focus on the object of interest, the fasteners.

To remove the unwanted areas, first the horizontal position of the main real head is detected. Since the rail head is uniformly illuminated vertically than any region, it can be detected by computing median of columns of the image vertically. Then, left and right regions cointaining can be extracted from a constant horizontal distance d of the rail head that is pre-determined. See figure 4.1 and Table 4.1

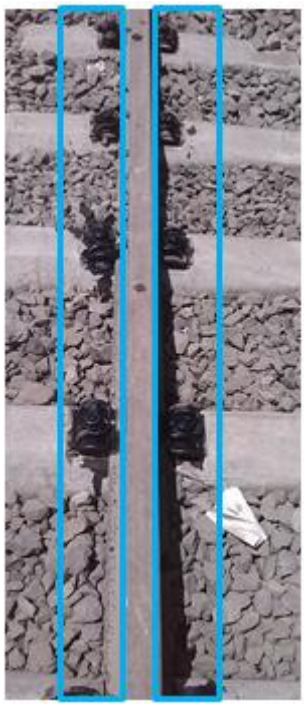


Figure 4. 1: Cropped regions of rail track image containing only fasteners

Input

1) Railway track image, I
2) Size of fastener (F_w , F_h)
3) Horizontal distance d of the fastener from the rail head

Do Find the column C with highest median value (rail head) Output Left region: I([C – d -1.5* F_w: C-d], :) Right region: I([C + d: C+d+-1.5* F_w], :)

Table 4.1: Algorithm for the image cropping

The method helps save the computation time of object detection; it can also be uses to minimize the number of false positive detections that may occur due to the inclusion of unwanted areas.
.

4.2 Image standardization

Correcting variations in lighting conditions have been practiced by many object detection algorithms [14] [15] for making the classification task easier and more accurate. This is accomplished by image standardization process. Standardising an image consists in performing the following operation:

$$I_{standard} = \frac{I - \mu}{\sigma}$$

Where $I_{standard}$ is the standardized image,
I is the original image, μ its mean, σ its standard deviation.

The integral image makes the computation of, the mean and the standard deviation of an image fast.

$$\begin{aligned} \mu &= \frac{1}{n} \sum_{x,y} I(x,y) \\ &= \frac{1}{n} II(x_{end}, y_{end}) \end{aligned}$$

$$\begin{aligned} \sigma &= \sqrt{\frac{1}{n} \sum_{x,y} I(x,y)^2 - \mu^2} \\ &= \sqrt{\frac{1}{n} II_2(x_{end}, y_{end}) - \mu^2} \end{aligned}$$

Where II is the integral image of the original image, II_2 is the integral image of the original image squared, n is the number of pixels in the image, (x_{end}; y_{end}) is the last (bottom right) pixel location.

4.3 Labeling dataset

In this process, positive and negative instances of the fasteners to be detected are properly extracted and labeled with correct output by hand. For instance, by cutting of the fastener image from the railway track giving them as the positive sample image and those that don't represent fasteners as negative sample images, see figure 4.2 illustrates the extraction of positive and negative examples from a rail track image. These properly labeled and extracted regions are called the training dataset. However, due care must be taken in preparing a very good training dataset as the performance of the classification highly depend on them.

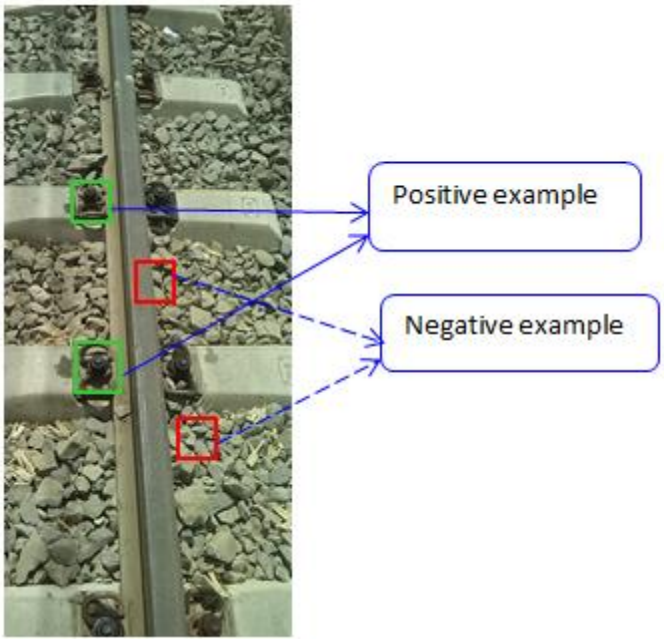


Figure 4. 2: Labelling a positive and negative examples from a railway track image

4.4 Generating negative and positive dataset

As the detector is a learning-based system, the amount of training data to be extracted is relatively significant. Furthermore in these systems, larger set of training examples normally leads to better performance. However, manually extracting and classifying so many examples is a lengthy and tedious process.

Input

1) Railway track image, I
2) Size of fastener (F_w , F_h)
3) Number N of fasteners to be extracted

Loop

For n=1, …,N

Compute random location(x,y)

Compute and save the training example:

$I(x - F_w/2 : x + F_w/2,\ y - F_h/2 : y + F_h/2)$

Table 4.2: Algorithm for extracting training examples

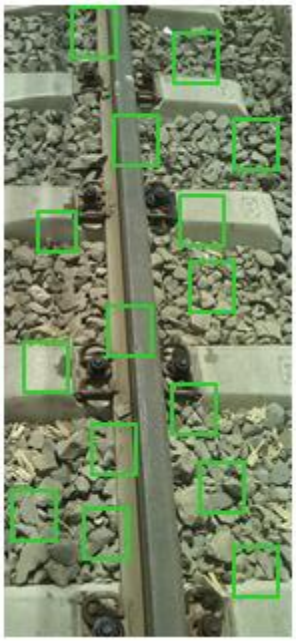


Figure 4.3: Automatically extracted negative training examples

An algorithm is developed which helps automate the said task. This algorithm, described in Table 4.2, extracts instances at several random locations in rail track images. These instances are visually inspected by a human expert, and appropriately labeled either as positive or negative examples.

It is to be noted that the railway track images contain much more negative instances than positive instances. Therefore, instances at several random locations in these images rarely represent positive instances. To increase the number of positive examples in the training dataset, I developed a functionality that extracts and classifies positive examples by manually selecting, via a GUI, the center of these examples in rail track images. Table 4.4 describes the algorithm used.

Input

1) Railway track image, I
2) Size of fastener (F_w , F_h)
3) Number N of fasteners to be extracted

Loop

For n=1, …,N
Compute random location(x,y)
Compute and save the training example:
$I(x - F_w/2: x + F_w/2,\ y - F_h/2: y + F_h/2)$

Table 4.3: Algorithm for the automatic extraction of training examples.

Input

1) Railway track image, I
2) Size of fastener (F_w , F_h)

Loop

Get mouse click location(x,y)
Compute and save the training example:
$I(x - F_w/2 : x + F_w/2,\ y - F_h/2 : y + F_h/2)$

Table 4.4: Algorithm for the extraction of positive examples.

Based on the algorithms given, we then build as many training datasets as pairs of positive/negative classes defined, and train a classifier based on each training dataset. A classifier can then specifically classify the types of instances that were used to train it.

Decreasing the intra-class variability of the training data by defining several couples of positive/negative classes, and therefore training several classifiers, increases the discriminative power of each trained classifier. However, it may also compromise their generalisation power. The objective is to maximize the accuracy of the classification by making a trade-off between its discriminative power and its generalization power.

4.5 Haar features designer

Feature extraction consists in transforming the training data into a set of features. Features must be carefully chosen so as to extract relevant information from the training data, thus reducing its representation. Here we discuss how to select appropriate Haar wavelet-like features in the context of a rail inspection system.

Another issue concerns the types of Haar wavelet-like features (or feature templates) to be considered. Increasing the number of types of features normally increases the performance of the classifier, since the learning algorithm will select the best features among all candidate features. However, it also increases the amount of candidate features, proportionally to the number of types of features. My objective is to devise appropriate feature templates to the current problem, so as to increase the performance of the classifier but without excessively increasing the amount of candidate features.

To assist in achieving this task, a functionality that enables to design, via a GUI, specific feature templates, has been developed. The feature templates can then be stored in dictionaries, in a format supported by the Haar wavelet-like feature descriptor. Its formats described below, and Figure 4.4 shows the use of this functionality in the GUI.

Taking into account the easy interpretability of Haar wavelet-like features, and the geometrical intrinsic properties (or patterns) of positive data, we can smartly design appropriate feature templates. Indeed, appropriate feature templates can be designed so as to fit these patterns.

- Index of the current feature template
- Size of the current feature template
- Total number of rectangles in the current feature template
- Index of the current rectangle
- Top-left coordinates of the current rectangle
- Size of the current rectangle
- Weight of the current rectangle

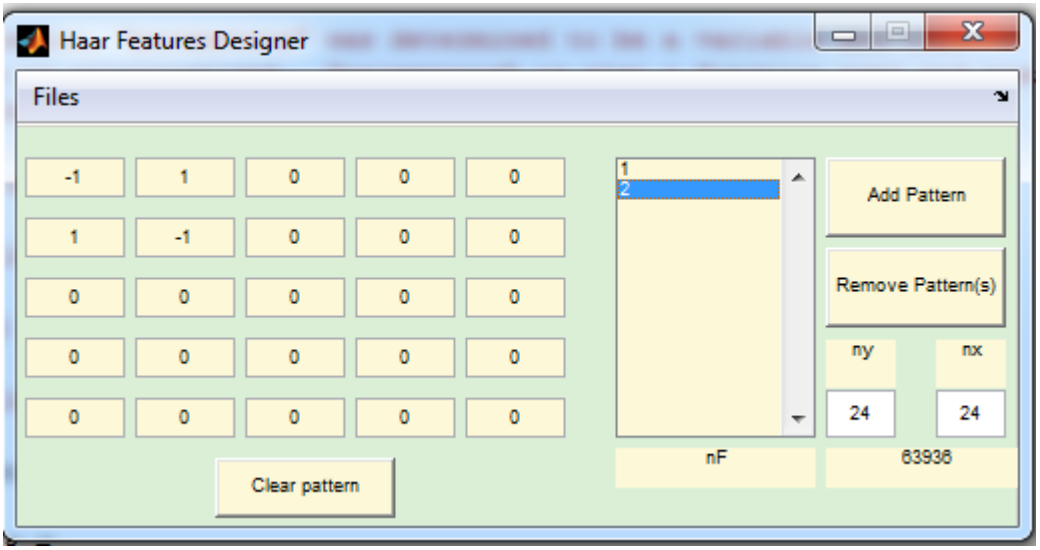


Figure 4. 4: Designing Haar wavelet-like features.

4.6 Haar Features listing

In this routine, the exhaustive list of the haar features of a given rectangle pattern is recorded in structure. The rectangle patterns are designed by haar feature template designer as mentioned earlier. A reasonable resolution for a training data is normally taken to be 24×24 pixels, for which a total of approximately 160,000 features can be constructed. Therefore, in this step based on the rectangle pattern and size of the image to be considered, index of the current feature, its coordinates and width and height are recorded for each feature.

The input and output of the feature listing is given below:

Input	Output
Number of rows of the pattern Number of rows of the pattern Features rectangles parameters	index of the current feature top-left coordinates of the current Haar's feature width and height of the current feature Haar's feature

Table 4.5: Haar feature listing: input and output

Feature rectangle parameters: is a structure containing the following parameters about the rectangle pattern:

- index of the current Haar's feature
- width of the current rectangle pattern defining current Haar's feature
- height of the current rectangle pattern defining current Haar's feature
- total number of rectangles for the current Haar's feature if
- index of the current rectangle in the current Haar's feature
- top-left coordinates of the current rectangle of the current Haar's feature
- width and height of the current rectangle of the current Haar's feature
- weights of the current rectangle of the current Haar's feature

4.7 Haar features computation

Once having the exhaustive list of the haar features, in these step values of features are computed. This is accomplished by computing the integral image of every rectangle in every feature followed by subtracting every sum of the white rectangle from the gray rectangle. This gives the feature value of that particular rectangle pattern and is stored in structure. The input and output of this routine is given as shown blow.

Input	Output
Set training examples of size (24x24) Haar feature listing Features rectangles parameters	Haar feature matrix

Table 4.6: Haar feature computation: input and output

4.8 Haar adaptive boosting weak learner

Given a training set of positive and negative data, and a set of feature templates, the learning algorithm can now be applied. The standard AdaBoost learning algorithm is given below.

Input

(1) Training examples $(x_1, y_1), ..., (x_N, y_N)$,
where x_i is a subwindow, and y_i its label $(y_i \in \{-1, +1\})$.

(2) The number T of weak classifiers to be combined.

Initialisation

$D_1(i) = 1/N$, $i = 1, ..., N$

Loop

For $t = 1, ..., T$

(1) Find the classifier h_t that minimises the weighted error ϵ_t:

$$h_t = \arg\min_{h_t \in H} \epsilon_t, \text{ where } \epsilon_t = \sum_{i=1}^{N} D_t(i)[y_i \neq h_t(x_i)]$$

(2) Choose the coefficient α_t:

$$\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$$

(3) Update the distribution D_{t+1}:

$$D_{t+1}(i) = D_t(i) \exp(-\alpha_t y_i h_t(x_i)),\ i = 1, ..., N,$$

and normalise to $\sum_{i=1}^{N} D_{t+1}(i) = 1$.

Output

Real output value: $V(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

Final strong classifier: $H(x) = sign\,(V(x))$

$(H(x) \in \{-1, +1\})$

Table 4.7: Standard adaboost algorithm

At each round, AdaBoost finds the optimal weak classifier (and so the corresponding feature) that best separates the positive and negative data. A distribution of weight D_t indicates the importance of the data for the classification. These weights are updated at each round so that new weak classifiers focus more on the data previously misclassified. The weak classifiers are then linearly combined into a strong classifier (the linear coefficients α_t being related to the error of the weak classifiers ε_t).

4.9 Cascading strong classifiers

A cascade of classifiers is a sequence of several strong classifiers that will be applied to a test data. The structure of a cascade reflects the fact that the vast majority of instances in a rail track image are negative. The objective is to speed up the detection and reduce false positives, with a little sacrifice of the true positive rate.

Table 4.8 describes the algorithm used to cascade several strong classifiers. At any stage in the cascade, a test data that is classified as negative is immediately rejected, while a test data that is classified as positive is further processed by the subsequent strong classifier. A test data is considered negative when it is classified as negative by any strong classifier of the cascade. A test data is considered positive only when it is classified as positive by every strong classifier of the cascade.

```
Input
    N strong classifiers Hn

Loop

    For n =1 , … , N
   If Hn(x) = -1
      Output: Negative
   Else if Hn(x)=+1 and n=N
      Output: postivie
  End
```

Table 4.8: Algorithm for cascading classifiers.

The implementation of a cascade of classifiers requires adjusting three
Parameters:

- the number of classifier stages,
- the number of weak classifiers (or features) used to boost a strong classifier in each stage
- The threshold of the strong classifier in each stage.

In practice, we can manually define, at each stage, targets for false positive rate and true positive rate. The thresholds of each stage can then be tuned to meet these targets, by testing the classifier on a testing dataset, as done in the previous section.

Chapter 5

5 Detector performance, results and conclusion

In this chapter, a performance evaluation and some test results of the detector will be presented. Variation in the system parameters affects the detector performance in different degrees. Understanding and evaluating of these system variations as function of different parameters helps to improving detector performance, troubleshooting and maintenance. A proper analysis of these parameters is given in following sections.

5.1 Measuring performance

The performance of the detector can be measured in terms of a detection rate, also called true positive rate (TPR). It is given by dividing the number of positive instances that have been correctly classified (thus called true positives) by all positive instances considered in the test, whereas the false positive rate FPR is defined as the number of negative instances that have been incorrectly classified (thus called false positives) divide to all negative instances available in the test. In the following tables, both the detection and failure rates are mean value of a run of 8 rounds of each test and they are put as percentage.

5.2 Performance results

The following tests show the dependency of the performance of a learning-based detector on the on the amount and type of training data, type and amount of feature templates and image pre-processing. In each test, a certain parameter is varied by keeping the other constant i.e. taking system default values.

No of negative instances	No of positive instances	Detection rate	Failure rate(mean)
100	40	92.8	0.45
400	40	93.6	0.36
400	90	94.1	0.36
700	90	94.4	0.21
700	200	94.6	0.13
1000	200	94.6	0.11

Table 5.1: Detection and failure rates as function of amount of training data

The test in table 5.1 illustrates Detection and failure rates as function of amount of training data used. It is clearly shown that increasing training examples enhances system performance although the contribution of increasing negative example is more important than that of increasing positive examples.

No of feature template	Detection rate (Mean)	Failure rate(mean)
2	93.8	.32
4	94.2	.21
7	94.6	.19
11	94.6	.11
12	94.7	.11

Table 5.2: Detection and failure rates as function of the number of feature templates.

Number of feature templates is one the factors which affect the detection capability of the system, as it is shown in Table 5.2, the performance of the systems is improved as the number of feature templates are increased.

Although including image cropping process don't bring about improvement in detection rate but it reduces the failure rate i.e. it reduces the number of false positive rates. Hence it is helpful to consider this process. In contrast, the standardizing the input image cause improvements both in detection rates and the failure rates of the detector.

Detector parameter	Detection rate(mean)	Failure rate(mean)
With image cropping	94.6	.19
Without image cropping	94.6	.11

Table 5.3: Detection and failure rates as function of image cropping

Detector parameter	Detection rate	Failure rate
With image standardization	93.9	.12
Without image standardization	93.1	.18

Table 5.4: Detection and failure rates as function of the image standardisation.

5.3 Recommendation for Future Work

The detector developed in this thesis can be improved and extended to increase its detention performance. Hence, I make the following suggestion for future work.

Since the performance of the detect depends on amount and type of training data, this work can be enhanced by considering example images from unusual railway track, like those at junctions and by repairing a relatively large training examples. It is also recommended to consider other feature templates like extended haar rectangular wavelets Lienhart and Maydt [19] since only haar rectangular wavelet-like feature considered. Even though AdaBoost has proved to be very effective, there are a number of other learning algorithms that could be used.

5.4 Conclusion

In this thesis, a kind of object detector is developed that processes railway track images to automatically detect the presence of fasteners in these images. The detector is primarily based on the Viola-Jones object detection framework (Viola and Jones, 2001) [1]. However it conveniently selects a new Haar wavelet-like features that are found to be more appropriate to the problem of fasteners detection in railway track images. While the detector has managed to achieve a detection rate of 94.6% but it also incurs a failure rate of .11%.

References

[1] P. Viola, M. Jones, “Rapid object detection using a boosted cascade of simple features,” Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition.

[2] Addis Ababa LRT project E-W line project design, China railway group limited

[3] The MIT Encyclopedia of the Cognitive Sciences, “learning and non-learning based approaches in machine learning”

[4] Yohann Rubinsztejn, “Automatic Detection of Objects of Interest from Rail Track Images,” University of Manchester

[5] Rafael C. Gonzalez , Digital Image Processing, book third edition 2007

[6] D.L. Magnus, “Non-contact technology for track speed rail measure-ment (ORIAN)," Proc. SPIE, Nondestructive Evaluation of Aging Rail-roads, vol. 2458, pp. 45-51, 1995.

[7] G.S. Bachinsky, “The electronic BAR gauge (a customized optical rail pro_le measurement system for rail grinding applications)," Proc. SPIE, Nondestructive Evaluation of Aging Railroads,

[8] W. Ebersohn, E.T. Selig, “Use of geometry measurements for mainte-nance planning," Transportation Research Record No. 1470, Railroad Research Issues.

[9] A.M. Zarembski, W.T. McCarthy, “Development of nonconventional tie and track structure inspection systems," Transportation Research Record No. 1489, Railroad Transportation Research

[10] Cybernetix Group (France), “IVOIRE: A system for rail inspection," internal documentation, [Online]. Available: http://www.cybernetix.fr

[11] Benntec Systemtechnik Gmbh, “RAILCHECK: Image processing for rail analysis," internal documentation, [Online]. Available: http://www.benntec.com

[12] M. Singh, S. Singh, J. Jaiswal, J. Hempshall, “Autonomous Rail Track Inspection using Vision Based System," IEEE Int. Conf. Computational Intelligence for Homeland Security and Personal Safety,

[13] C. Papageorgiou, M. Oren, T. Poggio, “A general framework for object detection," Proc. IEEE Int. Conf. Computer Vision,
[14] K.K. Sung, T. Poggio, “Example-based learning for view-based human face detection," IEEE Trans. Pattern Analysis and Machine Intelligence,

[15] H. Rowley, S. Baluja, T. Kanade, “Neural network-based face detection," IEEE Trans. Pattern Analysis and Machine Intelligence

[16] H. Schneiderman, T. Kanade, “A statistical method for 3D object de-tection applied to faces and cars," Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition.

[17] Y. Freund, R.E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting," Computational Learning Theory

[18] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, “Boosting the margin: A new explanation for the e_ectiveness of voting methods," Proc. 14th Int. Conf. Machine Learning

[19] R. Lienhart, J. Maydt, “An Extended Set of Haar-Like Features for Rapid Object Detection," Proc. IEEE Int. Conf. Image Processing.