Machine Vision and Applications
DOI 10.1007/s00138-011-0393-1

SPECIAL ISSUE PAPER

Creating robust high-throughput traffic sign detectors using centre-surround HOG statistics

Gary Overett · Lachlan Tychsen-Smith · Lars Petersson · Niklas Pettersson · Lars Andersson

Received: 30 October 2010 / Revised: 31 July 2011 / Accepted: 15 November 2011

Abstract In this paper, we detail a system for creating object detectors which meet the extreme demands of real-world traffic sign detection applications such as GPS map making and real-time in-car traffic sign detection. The resulting detectors are designed to detect and locate multiple traffic sign types in high-definition video (high throughput) from several cameras captured along thousands of kilometers of road with minimal false-positives and detection rates in excess of 99%. This allows for the accurate detection and location of traffic signs in geo-tagged video datasets of entire national road networks in reasonable time using only moderate computing infrastructure. A key to the success of the methods described in this paper is the use of extremely efficient classifier features. In this paper, we identify two obstacles to achieving the desired performance for all target traffic sign types, feature memory bandwidth requirements and feature discriminance. We introduce our use of centre-surround histogram of oriented gradient (HOG) statistics which greatly reduce the per-feature memory bandwidth requirements. Subsequently we extend our use of centre-surround HOG statistics to the color domain, raising the discriminant power of the final classifiers for more challenging sign types.

Keywords Sign detection · Geo-location · Color

1 Introduction

Traffic sign detection is an important application for car navigation and intelligent vehicle systems. While traffic signs are designed as geometrically consistent, distinct objects, the task of detecting them remains a challenging problem when the volume of input video is exceedingly large and computing resources are divided among multiple target sign types (see Fig. 1). Furthermore, the required computing power increases rapidly as classifiers are pushed to achieve higher detection rates.

At the current time several companies perform geo-tagged video data collection on road networks across the globe. Typically, traffic signs within the data are manually detected within the video by humans and added to maps using basic software tools. Despite the significant cost of this human approach, detection rates are usually around 80–85%. The aim of this research is to significantly improve on the detection rate, reduce the cost of map creation, minimise the time taken in creating maps, and raise the number of different object types that can be feasibly 'mined' from the data. Ultimately, we hope to replace all manual markup with automated traffic sign (and other road asset) detection software and eventually perform the detection using an in-car real-time system.

Given the importance of traffic sign detection in several applications there is a significant amount of previous

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

G. Overett (✉) · L. Tychsen-Smith · L. Petersson · N. Pettersson · L. Andersson
NICTA, Locked Bag 8001, Canberra, Australia
e-mail: gary.overett@nicta.com.au

L. Tychsen-Smith
e-mail: lachlan.tychsen-smith@nicta.com.au

L. Petersson
e-mail: lars.petersson@nicta.com.au

N. Pettersson
e-mail: niklas.pettersson@nicta.com.au

L. Andersson
e-mail: lars.andersson@nicta.com.au

Published online: 10 December 2011

Fig. 1 Typical traffic signs. Signs vary in their use of shape, text and color. While some sign types will occur with a higher frequency (e.g. speed signs) many sign types are exceedingly rare across video frames of a national road network (e.g. railway crossing). A desirable detection methodology will yield robust detections across a large number of different sign styles with minimal time-to-decision

literature dedicated to the subject [1,2,4,11,15,20] though little of this research deals with very high volumes of video data exposed to real-world problems. Broggi et al. [4] present a real-time road sign detector using color segmentation, shape recognition and neural network learning. More recently, Timofte et al. [20] detailed a full sign detection algorithm which automatically acquires a 3D localisation (geo-location) of the detected signs. While this method does commit to some sign-specific shape-based techniques it is shown to produce excellent results across a range of sign types and circumstances.

The widespread use of color in traffic sign design also means that color has often been used in sign detection literature. This includes the use of color segmentation in a support vector machine [5] based classifier using shape information [11], and the use of Haar features adapted to measure color information [2]. In the case of Bahlmann et al. [2] this requires the creation of the precomputed datatype, the integral image [22], for each colour space used (seven in total). This raises the issue of the time cost of referencing multiple precomputed datatypes during classifier evaluation, an issue we will deal with in this paper. Results in [2] indicated that color information allowed for the reduction of false-positive rates by one order of magnitude. More recently, Paulo and Correia [15] have used red and blue color information as an initial cue for a sign detection system. Signs are then further classified using shape information into several broad subcategories such as 'danger' and 'information'.

The work presented in this paper aims to improve an existing object detection framework which is actively employed in

Fig. 2 A successful sign detection as presented by the project detection software

traffic sign detection for commercial applications. This existing detector creation strategy is described in Sect. 2. Results on several sign types in numerous countries indicate that this preexisting process is successful for a large proportion of sign types but unfortunately not all. Typically, when classifier creation fails for a given sign, its training process has either *stalled* (failed to achieve reductions in error-rates with the selection of additional features) or *overgrown* (it requires too many features to achieve the desired error-rate performance) yielding a slow, resource hungry classifier. Figure 2 shows a successful sign detection as presented by the detection software along with estimated location, height, size and other details.

This paper makes three contributions. Firstly, we provide a basic outline of the creation of detection classifiers using synthetic data and discuss some of the issues of this approach. Secondly, we introduce the use of a centre-surround image statistic to minimise memory bottlenecks in classifier evaluation. Finally, we extend the use of centre-surround HOG statistics to the color domain.

2 Detector creation

The experimentation undertaken in this paper is designed to closely match the usual process of detector creation which is used in our commercially deployed traffic sign detectors. It differs somewhat from the classifier creation processes used in some academic literature and therefore warrants description and justification. Figure 3 outlines the basic process of detector creation for a given sign.

The aim of our detector creation process is ideally to yield a three to five stage classifier with each individual cascade stage achieving a 0.1–0.5% per-window[1] false-positive rate

[1] All false-positive rates indicated in this paper are calculated per-classifier-inspected-window. A typical single frame of video may be inspected up to a million times as the detector is run over multiple scales and locations in the frame.

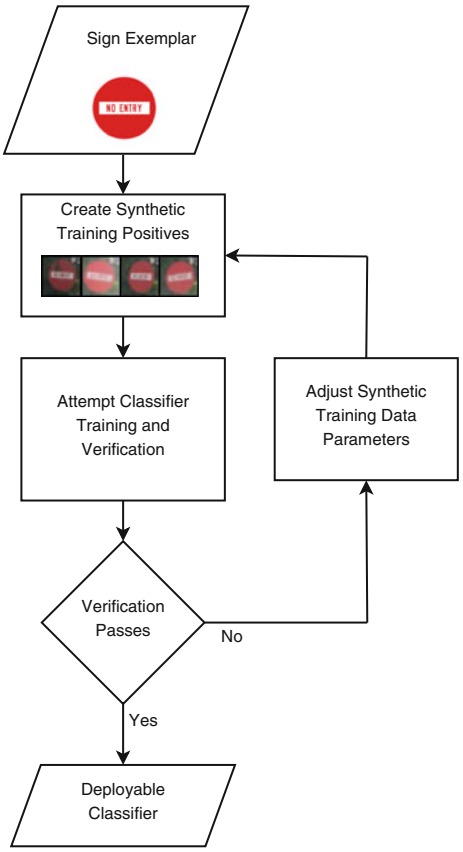

Fig. 3 Classifier creation using synthetic training data. The degree of desirable distortion which must be added to synthetic training data must be adjusted to yield a sufficiently general sign detector

in tandem with a 0.1–0.3% false-negative rate. The final classifier should aim to achieve a 99% detection rate with a false-positive rate below 10^{-9}. If the false-positive rate is much higher the detector will often produce more false positives than true positives due to the highly asymmetric nature of detection problems (especially the traffic sign detection problem where the in-car video may process several kilometers without encountering a specific sign). Some applications require false-positive rates as low as 10^{-11} to 10^{-12}.

2.1 Synthetic training data

The first aspect of our detection system which warrants explanation is the use of synthetic training data. Collecting labelled examples of various sign types in multiple countries under multiple illumination conditions is an expensive and time-consuming task. Therefore we have adopted the use of synthetic training data (see Fig. 4).

A key issue with synthetic training data is adding the 'right' degree of distortion to the data. If too much distortion is added the training process will stall as the classification problem becomes too difficult. Alternatively, if too little

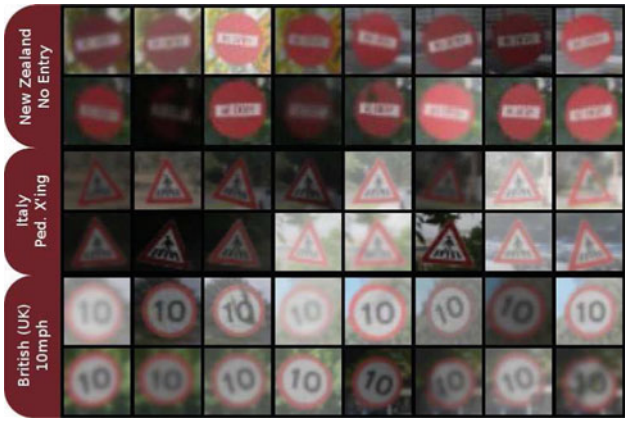

Fig. 4 Synthetic training data for the New Zealand no entry, Italian pedestrian crossing, and British 10 mph signs. For most sign types, these synthetic data prove sufficient for training real-world sign detectors. To generate such data we apply geometric distortions, synthesised changes in illumination, and minor occlusions to a digital sign exemplar. The results are superimposed on real-world background imagery

distortion is added the resulting classifier will fail to generalise its model of the given sign which leads to poor performance on real data.

The question of how best to adjust synthetic training data parameters between iterations (see Fig. 3) remains largely unsolved. In brief, our approach is to start with a well-tested but ambitious set of parameters which maximize variance across attributes such as geometric distortion, shading, noise levels, blur, and occlusions. This produces a dataset which is generally *more difficult*[2] to classify than most signs encountered in the real world. Critical to the success of this initial synthetic set is that it must be an *overcomplete* set, of which the real-world cases will form a subset. That is, if the resulting classifier detects 99.9% of the synthetic data it should equal or outperform this result when given real-world examples.

For difficult signs where verification fails during the classifier creation loop, we proceed by 'dialing' back the variance. Usually, this begins by reducing variance on well-understood properties such as geometric distortion. For example, in a first attempt at synthetic data we may apply random horizontal-plane rotations of ±55°. In a second attempt at generating data we may reduce this to a more conservative 40° which has the known effect of reducing the classifiers ability to detect a sign at wider angles. Since most signs will appear in multiple video frames at various viewing angles this is not usually a problem. For text-only signs which lack simpler (low frequency) pictographic distinctions, we commonly reduce the amount of random blurring applied in training data

[2] More difficult from the viewpoint of our feature set. It is possible to imagine feature sets which might easily distinguish our synthetic training data from real-world imagery on the basis of tell-tale signs in both sets.

since this can make these signs unrecognisable even to human eyes. In practice even minor reductions in variance greatly reduce the difficulty of the training task still allowing for a high degree of confidence that the final synthetic training data are overcomplete rather than deficient. For commonly occuring signs this can usually be verified. For rare signs where real-world verification is beyond practical reach it is often possible to compare synthesis parameters to a more common sign with similar attributes (shape, color, text, size etc), this serves to provide reassurance that training data have at least been created by a 'verified' formula.

Another problem with synthetic training data is that it may mimic some statistical properties of real-world data differently to others. Since different features measure different image properties, synthetic data does not work equally well for all feature types. For example, Haar features are more sensitive to luminance statistics present in training data and consequently require synthetic data to more accurately mimic real-world luminance properties. HOG [7,28] features, on the other hand, pay attention to aliasing effects, fringes, proper scaling and various edge properties. Similarly, training data for color features require a proper synthesis of the various color properties. As we observe in Sect. 8 we may not yet have the best 'recipe' for synthetic color training data.

Each of these aspects adds complexity to the task of creating synthetic training data where synthetic and real-world validation tasks perform with comparable results.

2.2 The short cascade structure

A popular detector structure found in literature is the cascade of classifiers [22] which has been used in a large number of published object detectors [14,23,24,26–28]. A requirement of all such cascade structures is that suitable termination criteria must be determined for selecting the number of features to be included in a given stage and setting appropriate decision thresholds at each stage. For example, the detectors of [21] and [14] employ a target false-negative rate of 0.2% with a corresponding false-positive rate of 65%. For high accuracy sign detectors such relaxed termination criteria are unsuitable. Under such criteria it would require nearly 50 cascade stages to achieve a false-positive rate below 10^{-9}, reducing the detection rate to around 90%. Even if the per-stage false-negative rate was greatly reduced, improving the overall detection rate, the resulting classifier would be very long. In contrast, we have adopted an approach of short, low error, cascades, using just 3–5 stages for a given detector. This has three important advantages.

Firstly, when very high final detection rates (in excess of 99%) are desired, accurately thresholding long cascade classifiers is difficult and requires a very large population of validation data. Consider the example of a 20-stage classifier with a 99% detection rate. In this case, each stage could afford only a false-negative rate of 0.05% (1 in 2,000 positives). Therefore, even with a validation population of 10,000 positive samples, only 5 samples would be rejected in each stage. Statistically 5 samples simply do not form a large enough population for guaranteeing accurate thresholding. Therefore the selected thresholds could not be relied upon to behave as desired in the production classifier.

Secondly, between the creation of cascade stages one must perform bootstrapping[3] to find the negative training samples which pass through the previous cascade stage. As classifiers achieve very low false-positive rates, this requires bootstrapping over very large amounts of video data captured over tens of thousands of kilometers. A human operator must then remove any true positives[4] from the bootstrapped negatives. It is highly desirable to minimise this required human input to the training system. Therefore, fewer cascade stages are desirable.

Thirdly, the final commercial detection jobs will be farmed out to a number of servers, some of which use a GPU implementation for the detector. GPU implementations generally require that all GPU cores execute the same function at a given time. In our implementation, each sub window of the input image is assigned to an individual GPU thread (core). When evaluating the first stage in a cascaded classifier, all the GPU cores will be running the same instructions on a regular grid, and full GPU core utilisation is easily achieved. After evaluating the first stage in a cascaded classifier, some sub-windows can be rejected right away, while others need to evaluate additional stages in the cascade to reach a decision. In our implementation, this leads to a "fragmentation" of the grid where some cores will be done with their sub-window and simply stay idle while other cores continue evaluating stages. The severity of this fragmentation depends on the particular GPU. In our configuration, if all GPU cores in a small input image neighbourhood (i.e 32 sub-windows/patches) are able to reject their sub-window, that particular neighbourhood can instantly be finalised and the GPU cores allocated to another neighbourhood. The waste of GPU cycles mainly occur when cascade evaluation diverges within a small image neighbourhood. Thus, as a rule of thumb, fewer, more powerful cascade stages will yield faster GPU implementations since they will generate less stage divergence (GPU grid fragmentation).

Lastly, it is worth noting that short cascade structures do have at least one disadvantage. One such disadvantage is that the average number of features which must be evaluated by

[3] Bootstrapping: A process whereby an incomplete cascade classifier is run over in-car video data. The aim is to acquire non-sign image data which passes through to the end of the currently incomplete cascade detector.

[4] True positives occur because the in-car video data contain images of traffic signs. Of course, we do not have labelling of the sign locations for our in-car video data as this is precisely what we are searching for.

the classifier is larger. This is another strong motivation for the fast-yet-discriminant features presented in this paper.

3 Technical considerations for high-throughput sign detection

In this section, we introduce two concepts which motivate our classifier design.

3.1 Memory wall

The term "memory wall" [25] is used to describe the growing disparity between processor speeds and bandwidth to off-processor memory. Wulf and McKee [25] predicted that as processor speeds increased more quickly than memory speeds one would eventually reach a point where the majority of processing time would be wasted as the processor waited on memory.

This effect is particularly apparent in numerous computer vision applications, such as object detection, due to the data-heavy nature of video and image information. Pettersson et al. [16] observe that for a typical object detection feature, such as the popular Haar features used in [22], the majority of the feature evaluation time is spent waiting on memory accesses to either the image data or a precomputed datatype such as the integral image [22,6] or integral histogram [17].

Consider the example of any of the rectangle-based features; Haar [22], RHOG [7,28], or region covariance [21]. Typical computing architectures will stream consecutive memory regions with great efficiency, however, the above features require memory access spread out over their respective precomputed datatypes. This causes a high rate of cache misses, forcing the CPU to idle. Therefore, while many precomputed datatypes, including the integral image [21,22] and the integral histogram [17], yield constant time feature evaluation, the final system may be limited by memory performance rather than the number processor instructions to be executed.

Pettersson et al. [16] address this problem via a novel precomputed image which they called the *histogram image*. This is an "image" where each 32-bit word represents a histogram of oriented gradients within a 4×4 pixel area in the input image. The histogram image is used in a similar way to the precomputed images for other features, but has a major advantage in terms of speed. A feature based on the histogram image requires only a *single read* of a 32-bit word for feature evaluation, as opposed to several semi-random reads per feature.

Notwithstanding a basic change in the operation of most computing hardware, a most apparent solution to this problem is to redesign the way in which a specific object detection algorithm relies on memory. We demonstrate this in Sect. 4 using our centre-surround HOG statistics.

3.2 Image precomputed datatypes and multi-class detectors

Many popular features for object detection do not directly reference the image itself but rather rely upon a precomputed datatype. Examples include Haar features [22] and rectangular histogram of oriented gradient (RHOG) features [7,28]. For many detection applications, the time taken to create this precomputed datatype must be considered when calculating the expected time-to-decision of a given classifier. However, when multiple classifiers each containing large numbers of features all reference the same precomputed datatype the overheads often become negligible.

In the case of the sign detection task being considered in this paper we find that the time taken to create the precomputed datatypes is indeed negligible. This remains the case even when a selected feature's precomputed datatype takes considerably more time to compute than an alternative. Consequently, we find that greater effort toward the construction of powerful precomputed datatypes to support more discriminant features is a favourable approach for multi-class many-feature detection problems.

We also note that the desirable characteristics of features which will be evaluated early in the cascade structure are different to the desirable characteristics at the tail-end of the detector. For example, front-end features must evaluate in minimal time since they will be evaluated on most of the image data. In contrast, tail-end features must be able to 'dig deep' to achieve a reasonable discriminance on more challenging image data which have passed through the earlier stages of the cascade. Therefore, precomputed datatypes for tail-end features can often be far more computationally intensive.

4 A centre-surround HOG statistic

In this section we present the fundamental principles of a memory efficient centre-surround image statistic, specifically we provide the background to the centre-surround HOG statistic first presented in [16].

The notion of a centre-surround image statistic comes about in response to observations about the bottlenecks involved in feature evaluation. Of particular concern is the issue of a memory wall as described in Sect. 3.1. At the current time many image features have low feature complexity relative to their memory bandwidth requirements (see Fig. 5).

By reducing the feature evaluation memory requirement to a single reference of adjacent bytes, we greatly improve the time-to-decision of the final classifier. In this paper we present a single example of a centre-surround image statistic, however, other features taking advantage of the same feature design properties can be developed.

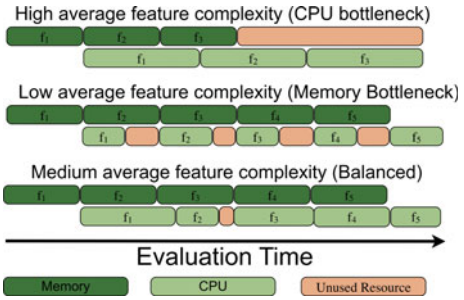

Fig. 5 Bottlenecks in the evaluation of a feature. Both CPU computation and fetching memory may combine to form bottlenecks in the evaluation of a feature. Many popular features used today have relatively low computational overheads compared to their associated memory requirements. We anticipate that as the availability of computing power continues to outgrow memory bandwidth, that low memory overhead image features such as centre-surround statistics may play an increased role in computer vision

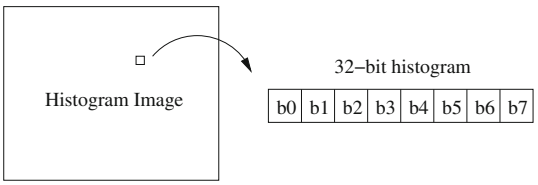

Fig. 6 Each 'pixel' in the histogram image encodes a histogram of the orientations in a 4×4 neighbourhood. Each orientation is represented by just 4 bits

4.1 The histogram image

The process of computing the histogram image is outlined in [16]. However, since most readers may not be familiar with its form, we provide an outline of the content of each histogram image pixel.

A normal RHOG implementation [7] creates a histogram over an arbitrary rectangular region [7]. In order to concentrate all the information required to evaluate a feature into a single histogram image pixel, binning is only performed over each 4×4 image patch for just 8 gradient orientations. Since each gradient within the 4×4 region is subject to thresholding, the maximum value for any histogram bin is just 16. By reducing the occasional value of 16 down to 15, it is possible to capture a histogram bin count in just 4 bits. This reduces the total memory requirement for a single histogram image pixel to just 4×8 bits $= 32$ bits. See Fig. 6.

5 The LiteHOG and HistFeat features

In this section, we provide the details of two feature types which both make use of the same histogram image precomputed datatype, the HistFeat and LiteHOG features. Both features have extremely fast evaluation times when compared to other popular object detection features.

5.1 HistFeat

The initial evaluation of the HistFeat feature can be found in [16]. It involves taking two orientation bins from a single histogram image pixel (see Fig. 6). The highly simplified evaluation of the HistFeat feature means that despite its minimal memory requirement it still leaves the CPU waiting on memory, though not to the same severe degree as with Haar and RHOG features. This suggested the formulation of an improved feature that used the idle computational resources. This is because the CPU is free to do additional computations during waits for memory, providing these computations only require memory already in the cache. The LiteHOG feature uses this spare CPU time using *all* 8 orientation bins not just 2 as in the case of HistFeat.

5.2 LiteHOG

The original presentation of the LiteHOG feature and the LiteHOG+ extension can be found in [13]. Here we provide an extended explanation of the features design and implementation.

Creating a robust eight-dimensional model with very fast lookup is unfortunately extremely challenging. Our solution is to find a one dimensional linear projection from the 8 orientation bins of the single histogram image pixel. This is done using the canonical variate of Fisher's linear discriminant (FDA) [8],

$$w = S_w^{-1}(m_1 - m_2) \tag{1}$$

where w is the N dimensional projection matrix, S_w is the within class scatter matrix and m_1, m_2 are the means of the positive and negative classes, respectively. We consider two feature variants, LiteHOG which uses all eight dimensions, and an extended feature set, LiteHOG+ which uses any subset of the eight dimensions. That is, for LiteHOG $N = 8$ while in the case of LiteHOG+ $N \in [1, 8]$. Note, also that the boosting algorithms employed, RealBoost [19] and LogitBoost [9], produce a weights vector which is used in the calculation of the projection matrix w, i.e. we are using *weighted* FDA. This means that new projections may be found for each boosting training round, uncovering new information from the *same* histogram image pixel according to the boosting algorithm's weighted priorities.

At this point we must deal with a 'special problem' which arises in the histogram image. A combination of the gradient magnitude thresholding (Sect. 4.1) and the low frequency edges found in negative data (due to sky, road, walls etc.) means that a common response of any given histogram image 'pixel' is straight zeros. That is, all 8 directional bins counted no edges. Over a typical video sequence up to 40% of the histogram image 'pixels' may contain straight zeros. At first sight this may seem an indication of wasted information,

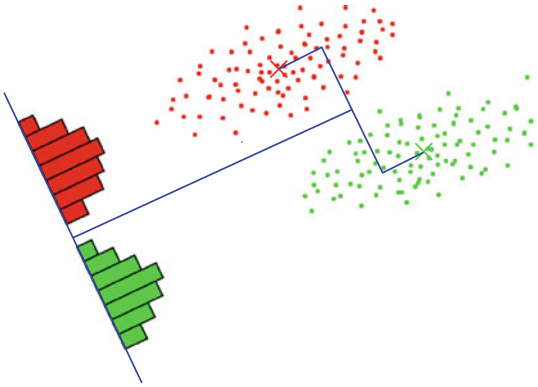

Fig. 7 Fishers linear discriminant. A projection w is found which divides the two class populations C_+ and C_-. When the underlying distributions are Gaussian the projection w provides the optimal separation. See [3] Sect. 4.1.4

however, since many video frames contain a significant proportion of either sky or road it is not unexpected. The issue for Fishers Linear Discriminant is that it is only optimal for a Gaussian distribution. The actual distribution at hand is eight dimensional, concentrated at the origin and strictly positive. Thus, we apply Fishers linear discriminant only to those points which are not at the origin to make the distribution appear more Gaussian. No projection is needed for these points and we want the projection calculation to 'focus' on the remaining data.

Let,

$$\hat{C}_i \subseteq C_i \quad \text{such that} \quad \forall x_j \in \hat{C}_i, x_j \neq \mathbf{0} \tag{2}$$

so that $\hat{C}_i$ is the subset of the class data C_i without the points at the origin. Let the within-class scatter matrix $\hat{S}_w$ be defined using the positive and negative class subsets,

$$\hat{S}_w = \sum_{n \in \hat{C}_1} (x_n - \hat{m}_1)(x_n - \hat{m}_1)^T \tag{3}$$

$$+ \sum_{n \in \hat{C}_2} (x_n - \hat{m}_2)(x_n - \hat{m}_2)^T \tag{4}$$

using the subset means $\hat{x}_1$ and $\hat{x}_2$.

This gives the final projection of,

$$\hat{w} = \hat{S}_w^{-1}(\hat{m}_1 - \hat{m}_2). \tag{5}$$

Figure 7 shows a graphical explanation the normal behaviour of Fisher's Linear Discriminant, while Fig. 8 shows how FDA is used on the modified populations $\hat{C}_+$ and $\hat{C}_-$.

Once this projection is applied, the LiteHOG+ feature response is able to be dealt with in a manner similar to other scalar feature responses such as in the case of Haar-features. In this way we can pair the basic LiteHOG+ feature with any of the weak learners which operate on scalar feature responses. In this paper, we combine all features with the smooth response binning weak learner as presented in [12].

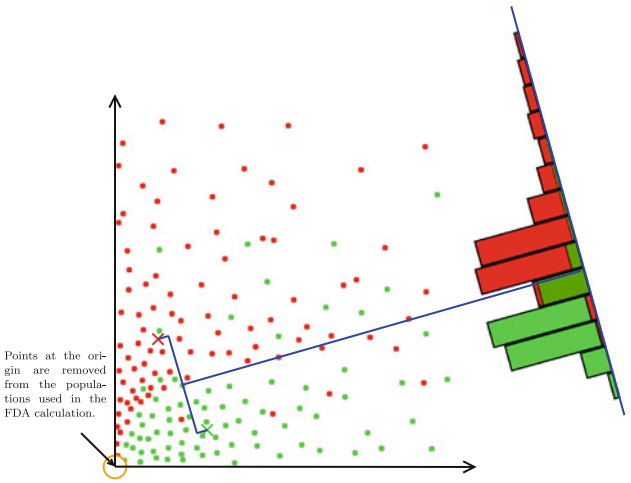

Fig. 8 Fishers linear discriminant for the LiteHOG+ feature. A projection $\hat{w}$ is found which divides the two class populations $\hat{C}_+$ and $\hat{C}_-$ which have had points at the origin removed. The underlying distributions are not Gaussian and therefore the projection $\hat{w}$ does not necessarily provide the optimal separation. However, experiments show that the projection $\hat{w}$ provides excellent separation and is relatively simple to calculate. Results are shown in Sect. 6

Thus the evaluation of the LiteHOG+ feature involves N multiplications, and $N - 1$ additions to compute the projection, see Eq. (6).

$$R = w \cdot x \tag{6}$$

where x contains the N selected bins from the histogram image. Since the LiteHOG feature space always requires an eight-dimensional projection, it is slower to compute than the LiteHOG+ feature space which may ignore several dimensions. Additionally, the projections within the LiteHOG+ feature space are often more discriminant, making LiteHOG+ both faster to compute and more powerful on an averaged per-feature basis. In this paper we will limit our experimentation to the use of the LiteHOG+ feature rather than the original LiteHOG feature.

Figure 9 shows the basic flow of LiteHOG+ feature evaluation.

6 Time-error feature performance

In this section, we provide a time–error comparison of the LiteHOG+ and HistFeat features to the Haar and RHOG features. The results presented here take a closer look at the single-stage stop sign experiments presented in [13]. Readers wishing to see similar analysis using alternative sign types and a pedestrian dataset are referred to [13].

Our experimental setup in [13] did not employ our most recent method of generating training data synthetically but rather created it from a small population of real-world sign examples. In the case of the stop sign results of Figs. 11 and 12, an initial population of stop sign images was expanded

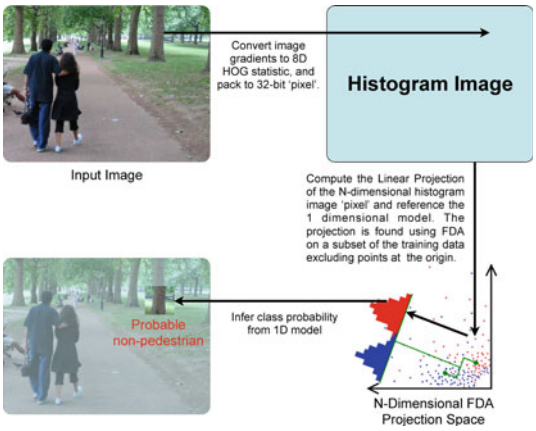

Fig. 9 LiteHOG+ feature evaluation

Table 1 Number of images used as positive training and validation data

Type	Raw train	Raw valid	Dist train	Dist valid
Stop sign	129	32	10,000	2,500

Distortions are applied to the raw images to create a larger, more generalised dataset

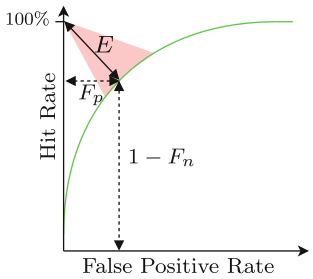

Fig. 10 The minimum total error or distance to the knee. We combine the false-positive and false-negative rates to form a single error rate. This is plotted alongside time-to-decision in the time–error plot shown in this Fig. 11

by applying basic distortions to the imagery. Table 1 gives the exact numbers of training and validation data used.

For the stop sign population listed in Table 1, the RealBoost [19] learning algorithm was used to build 1,000 different strong classifiers consisting of 1–1,000 features. For each of the strong classifiers, we calculated the *ROC curves* and the *scan time* on a typical video sequence using an AMD64 2.2GHz machine.

By pairing the ROC and scan time data we produced the time–error curves shown in Fig. 11. In order to produce a two dimensional time–error plot we combine the false-positive and false-negative error rates into a single error measure, the *minimum total error*, which samples the location of the ROC curve closest to the *'knee'*. Comparison with other combined error measures such as the area under the curve reveal equivalent results for most classifiers. Figure 10 and Eq. (7) provide the details.

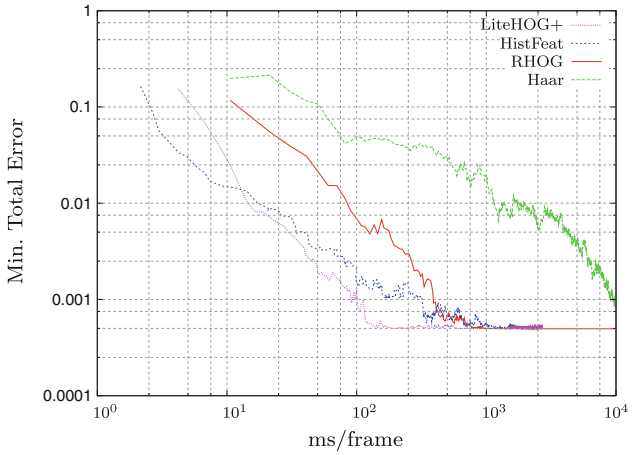

Fig. 11 First-stage time–error plots for LiteHOG+ feature versus Other (stop sign). The ROC performance comparison at time-to-decision =50 ms is shown in Fig. 12. LiteHOG+ is the dominant performer for classifiers in the range 15–1,000 ms

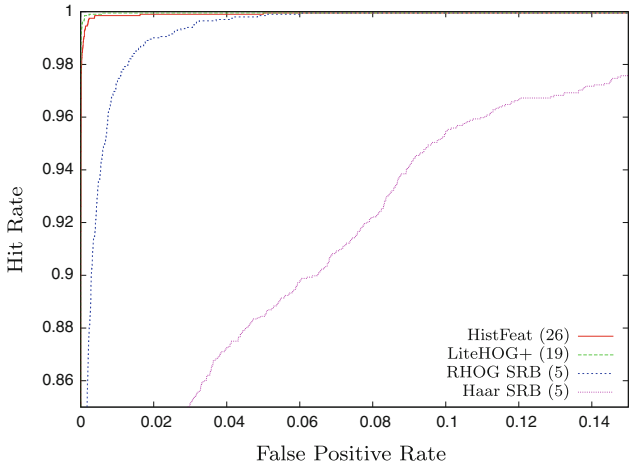

Fig. 12 First-stage ROC Performance at 50 ms (20 Hz) of computational time (stop sign). Again, the LiteHOG+ feature has the highest classification performance in this CPU time budget with half the error rate of its nearest rival, the HistFeat classifier. In this case we see a dramatic difference between the HistFeat and LiteHOG+ classifiers and the Haar and RHOG rivals. Specifically, the LiteHOG+ feature achieves a reduction in minimum total error of 10-fold over the RHOG feature and 50-fold over the Haar feature. The *number in brackets* next to the feature key is the number of features in the classifier. This very large improvement in the time–error performance is a primary motivation for the use of efficient centre-surround statistics as used in the HistFeat and LiteHOG+

The minimum total error E on the ROC curve is taken to be at the 'knee' of the curve, where the ROC curve is closest to perfect classification.

$$E = \min(\sqrt{F_p^2 + F_n^2}) \tag{7}$$

where E is the minimum total error, F_p is the false-positive rate and F_n is the false-negative rate

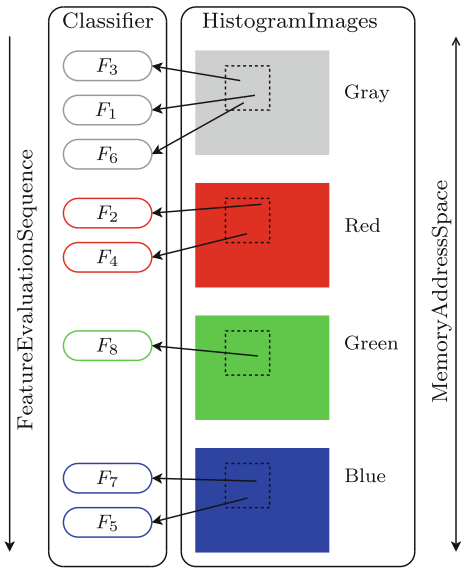

Fig. 13 Feature reordering allows for memory to be *streamed* efficiently from the precomputed datatype. Note, for features such as the rectangular Haar features (integral image) or HOG features (integral histogram [17]) no ordered evaluation can be created as individual features rely on *multiple* and potentially distant locations in memory

Figure 11 shows the resulting time–error plot for various features including HistFeat and LiteHOG+. The differences between Haar and RHOG features and our centre-surround HOG statistic-based features, HistFeat and LiteHOG+ become even more apparent when considering their ROC validation performance for a CPU budget of 50 ms. See Fig. 12.

It is important to note that all features compared in this plot use precomputed datatypes to achieve *constant time* feature evaluation, however, centre-surround features are much faster to evaluate because of their lower memory requirements and ability to stream memory consecutively (see Sect. 7 and Fig. 13). By comparison, if we had used an integral histogram [17] the time-to-decision for these classifiers would be about four times slower and similar to that of our RHOG classifier due to the need to reference four locations in memory for each feature evaluation. One circumstance under which the integral histogram would be required is if the target application needed the ability to evaluate HOG features of arbitrary scale to achieve the desired detection rate.

7 Extending the use of centre-surround HOG statistics

In this section, we extend our preexisting use of the centre-surround HOG statistic to the color domain. Clearly many traffic signs make extensive use of color and therefore this is a natural extension of our grayscale detection capabilities. Furthermore, we note that some sign types contain different colors whose grayscale intensities may be very similar. Degrading the information available for detection.

The color implementation of the LiteHOG+ and HistFeat features follows naturally from the original implementation. We construct histogram images using the grayscale, red, green and blue color channels in turn. This yields four precomputed datatypes for a single frame of video or training image and is similar to the approach of [2]. The complicating factor which must be remembered is that the classifier will now be referencing four adjacent sections of memory depending on the 'color' of a selected feature within a classifier stage. To minimise the negative effects this has on evaluation speeds, we sort the selected features of a classifier according to their 'color' and reference location in the precomputed datatype (the reference location is a consequence of the features x and y coordinates within the detection window). Figure 13 gives visual representation of how reordering features in a classifier can 'serialise' the order in which memory is referenced so that memory can be efficiently 'streamed' to the processor.

8 Results: color versus grayscale features

In this section, we provide experimentation and analysis of prospective color and grayscale classifiers. For our initial cascade stage we created color and grayscale classifiers of up to 35 features using the RealBoost [19] training algorithm. This training process was repeated for the three sign types shown in Fig. 4 using synthetic positive training data. Training data consisted of 100K synthetic positive training images and 200K real-world negative training images. Of these training populations 20% was set aside within the training process to report internal validation statistics. In each training round, 4,300 features were made available for selection by RealBoost with the feature pool being renewed after each feature selection (this is found to be a better approach than evaluating a very large but static feature pool for multiple training rounds).

As stated in Sect. 2 we do not normally adopt a one-size-fits-all approach to setting classifier stage termination criteria. Rather we customise the stage thresholds and number of features according to the reported detection performance of prospective stages. Under ordinary circumstances this involves allowing the classifier to grow (add features) until such time as it is able to achieve a false-positive rate at least lower than 0.5% with a false-negative rate of around 0.1–0.3%. The operator then sets the classifier threshold to achieve whatever appears to be the best performance within this range. If a given feature type does not achieve this performance range before the maximum number of features (in the first stage we set this to 35 features) is selected, the classifier is considered failed. If none of our available features can achieve a reasonable performance, we must seek out better

Table 2 Stage 1 classifier lengths (#features) and timings

Feature	#Grayscale	#Color	Time-to-decision (ms)
Haar	35	30	342
HistFeat	35	18	53
LiteHOG+	35	33	102

features or extensions to existing features such as the color extension presented here.

However, for academic comparison we must alter this approach somewhat to yield the best color versus grayscale comparison possible. In the first instance we proceed with *grayscale* classifier creation according to the above approach. However, after setting a grayscale classifier threshold we then create a *color* classifier containing just enough features to provide the given stage with *equivalent* time-to-decision as its counterpart grayscale classifier. Table 2 shows the evaluation times for Haar, HistFeat and LiteHOG+ features. HistFeat, the fastest feature has the greatest difference in terms of color versus grayscale evaluation times. This makes sense when one considers that such low-complexity features are memory bandwidth limited and referring to multiple color histogram images would require a greater memory bandwidth overhead (see Fig. 5).

8.1 Synthetic validation

Figure 14 shows the ROC detection performance for the prospective first stages of a cascade using the LiteHOG and HistFeat features using only grayscale features or a combination of colors. This result shows very significant improvement in performance using color classifiers, however, it is important to bear in mind that this experiment employs the use of *synthetic data*. Indeed, we find that color performance gains weaken significantly when considering real-world validation data (see Sect. 8.2 next).

8.2 Real-world validation

Validation of traffic sign classifiers requires that one has a large population of real-world positive traffic sign instances. For most traffic sign types we simply do not have such datasets. However, our previous commercial sign detection work does provide us with a few large sign datasets which have been assessed for accuracy by our customers. Of these datasets we have selected, the New Zealand no entry sign dataset containing 3,274 automatically detected no entry signs from in-car video data. Therefore, we will use the no entry signs collected by this *baseline* classifier to determine the likely detection rate of our prospective color and grayscale stages.

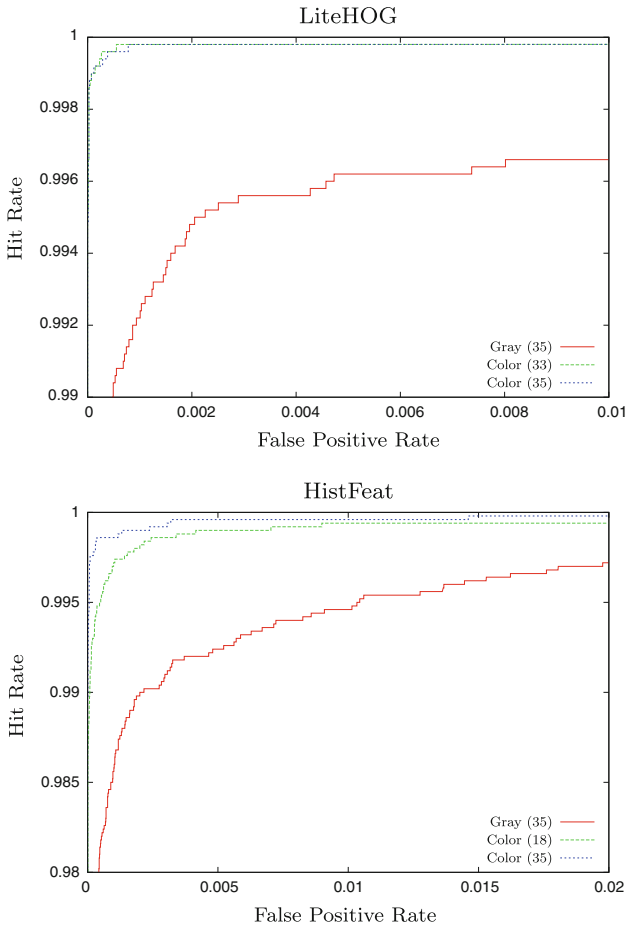

Fig. 14 ROC Performance for prospective first stages of a HistFeat or LiteHOG+ cascade (New Zealand no entry sign). Note change in scale. Classifier performance is shown for each classifier type using all 35 features selected. We also show the performance of a truncated color stage which will evaluate in equal computation time (see Table 2). These results reflect the detection performance on *synthetic data*

The baseline classifier in question comprised of grayscale features. This introduces the possibility of verification bias, since no entry signs which are difficult to detect without color cues may not be present in the dataset. However, examination of the baseline classifiers detection rate indicated that this is quite unlikely and in any event this bias would only have the potential to favour grayscale classifiers during validation. Therefore any improvement in color classification is likely to be at least as good as (possibly better) than the verification process indicates.

Our verification process proceeds as follows:

1. Hits from the commercially used baseline classifier are grouped into clusters of hits which are likely to belong to the same single sign instance. The mean location and scale of the overlapping hits are taken to indicate the sign's true location and scale (this is generally a reasonable assumption over the verification population). This results in 3,274 true-positive sign instances.

Table 3 Comparison synthetic versus real-world validation data

Classifier	Synthetic		Real world		Computation time (ms)
	False-positive rate (%)	Hit rate (%)[a]	Misses	Hit rate (%)[a]	
Grayscale HistFeat	0.17	98.96	14	99.57	53
Color HistFeat	0.20	99.90	3	99.91	53
Gray LiteHOG	0.23	99.52	13	99.60	102
Color LiteHOG	0.20	99.98	2	99.94	102
Gray Haar	0.44	98.54	1,086	66.83	342
Color Haar	0.18	99.98	1	99.97	342

[a] Hit rates for real-world and synthetic data are not equivalent measures as the real-world hit rate is calculated by matching hits to the approximate locations found using a baseline classifier while synthetic hit rates are calculated per synthetic training image

2. The resulting 3,274 sign instances are extracted from their original video along with some additional boundary context to yield patches containing instances of New Zealand no entry signs in the real world.
3. Next, the prospective single-stage classifier is run at various scales and locations over the patches yielding a list of hits (or lack thereof) for each patch.
4. Finally, we compare the hit locations found by the prospective single-stage classifier against the mean location and scale determined by the hit cluster of the baseline classifier. A sign is considered "missed" if *no* hit is produced which is centred within a 20% (relative to the sign width and height) margin of error of the baseline cluster location with a corresponding scale within 20% of the cluster scale i.e. the single-stage classifier fails to fire within a reasonable location and scale of the sign location obtained in step 2.

The resulting hit rate comparisons can be found in Table 3. Comparing grayscale classifier hit rates with color classifier hit rates (following row), we observe a clear improvement in detection performance for the color classifiers.

8.3 Exposing the difficulties with synthetic data

The results in Table 3 reveal the hazards of using synthetic data. For each feature and each color domain (grayscale or color), we observe quite varied correspondences between the calculated hit rate using synthetic validation data and the real-world hit rate. Nonetheless, the greater pattern is clear. Color classifiers more easily achieve higher detection rates and more importantly, *color classifiers achieve higher detection rates under equivalent time budgets*.

Unfortunately, color classifiers will not see adoption within our main classifier training work flow (Sect. 2) until we find a reliable method of avoiding the large disparity between synthetic and real-world classifier performance. Currently, we rely upon reliable synthetic validation data to guide us toward the creation of a successfully deployable classifier.

Several feasible solutions exist. Firstly, it may be possible to update our synthetic data generation to better synthesise the color properties of real-world images. However, the issue of color in computer vision is an open problem [10,18]. Accurately producing a truly realistic model of the manner in which sign exemplar colors change under different illumination conditions is most likely a *very* difficult problem. An alternative approach is to collect in incomplete pool of real-world signs using an initial classifier trained using synthetic data run over a subset of real-world video. Subsequently, the incomplete pool of real-world signs may be used to train a stronger classifier designed for the final commercial and complete collection of signs. This second approach is considered in the next section where we test real-world training data obtained using a classifier trained using synthetic data.

8.4 Real-world training and validation data

In this section, we consider the use of real-world training and validation data for complete cascade training. To obtain a real-world training and validation dataset, we rely on previously trained sign detectors which have been used on a very large video dataset from the New Zealand national road network. For a convenient division of training and validation data without overlap, we select data from the north island to be our training data with the south island data making up a smaller validation dataset. Table 4 provides the details of the dataset sizes obtained in this manner for three selected sign types.

Next we train a four-stage cascade classifier for each of the three sign types using both the HistFeat and LiteHOG+ feature sets in their grayscale and color variants. To render comparable results, we attempt to align the *false-positive rates* and *time-to-decision* of each color classifier with its grayscale variant. The divergence between color and grayscale variants is left to be borne out in the false-negative rate of the

Table 4 Number of positive training (north island) and validation (south island) images

Type	North/train	South/valid
No entry	2,574	699
Giveway/roundabout	40,020	13,362
Speed	57,377	21,500

Table 5 Cascade structure

	#Feats	False-positive rate	
		Per stage	Total
Stage 1	35[a]	0.1%	0.1%
Stage 2	200	0.1%	10^{-6}
Stage 3	400	1%, 5%[b]	10^{-8}, 5×10^{-7} [b]
Stage 4	600	0–100%[c]	NA

[a] Grayscale only. Color classifier stage lengths are varied to yield a stage with the equivalent first-stage time-to-decision as its grayscale counterpart

[b] No entry sign classification requires more relaxed threshold due to classification difficulty

[c] We do not commit the final stage to a given threshold but rather produce an ROC error performance curve across a range of thresholds (see Fig. 15)

final classifier. This yields the classifier structure outlined in Table 5. For grayscale classifiers, we always employ an initial stage of 35 features while color classifiers have varied initial stage lengths in order to normalise the time-to-decision between grayscale and color counterparts.[5]

Between the training of each classifiers four cascade stages we must bootstrap negative training data which passes through the previous cascade stages. For this we use a large library of in-car video data from various locations around the world. Since these data are not previously tagged for the existence of signs, the bootstrapped negatives must have any true-positive sign instances removed manually. In the early stages, the bootstrapping yields predominantly non-sign instances (false positives) and therefore this manual task is relatively simple. However, as false-positive rates approach 10^{-6} (after two stages are trained) the bootstrapping yields great many true-positive instances (which must be manually removed). Furthermore, the volume of in-car video data that must be searched by the classifiers' increases greatly. Indeed as false-positive rates pass 10^{-9}, bootstrapping must be performed over several hundred Gigabytes of in-car video, a process which requires significant computational resources and far outstrips the resources required to actually perform training once the negative training data have been collected. Generally, we aim to have around 200,000 bootstrapped negative training examples available for each training stage. However in some cases, the manual removal of true-positive instances means that as few as 140,000 negative (non-target) training examples are available.

Experience showed that the classifiers constructed using the LogitBoost [9] boosting algorithm outperformed that of the RealBoost [19] trained classifiers. Therefore the results in this section report the performance of LogitBoost trained classifiers. Figure 15 shows one of the results.

The ROC classifier performance for each of the three sign types across all 4 stages yields a total of 12 ROC plots for the HistFeat and LiteHOG+ classifiers. Since the results for each classifier essentially shows a similar trend across its four stages, we have selected only the ROC performance plots for the LiteHOG+ speed sign classifier to show in Fig. 15. While some other classifiers showed a greater relative improvement through the use of color, the LiteHOG+ color classifier is invariably the strongest performer. Furthermore, the speed sign class has the largest validation population (see Fig. 4) which also warrants its selection. The greatest reduction in error rate across the experiments was for the LiteHOG+ grayscale no entry classifier versus the corresponding LiteHOG+ color classifier. In this case a 75% reduction in false-negative rates was observed. This result however, must bear in mind the relatively small south island 'no entry' sign validation sets.

9 Conclusion

In this paper, we have detailed a system for creating traffic sign object detectors for use in very high-throughput applications. The issues involved in constructing such detectors with very low false-positive rates are presented. In support of creating stronger, faster classifiers, we present our use of centre-surround HOG statistics in the HistFeat and LiteHOG+ feature type. Additionally, we expand their use along with Haar features to include color information.

Time-error analysis shows the very significant benefit of using centre-surround HOG statistics in creating improved classifiers with minimal time-to-decision. For a given CPU budget of 50 ms, the LiteHOG+ feature reduces the error by an order of magnitude relative to an RHOG feature implementation while maintaining high detection rates. Our time-to-decision evaluation also shows that color classifiers will perform consistently slower than their grayscale counterparts but that the slowdown is not extreme. This slowdown is attributed to memory overheads caused by the referencing of multiple precomputed datatypes. Despite this slowdown, color information still yields better performance in comparable time-to-decision. We also note that synthetic training data should be used with caution when constructing high performance classifiers for the real world.

[5] For classifiers with a very low first-stage false-positive rate, the time-to-decision is found to be almost entirely dependant on the first-stage classifier timing.

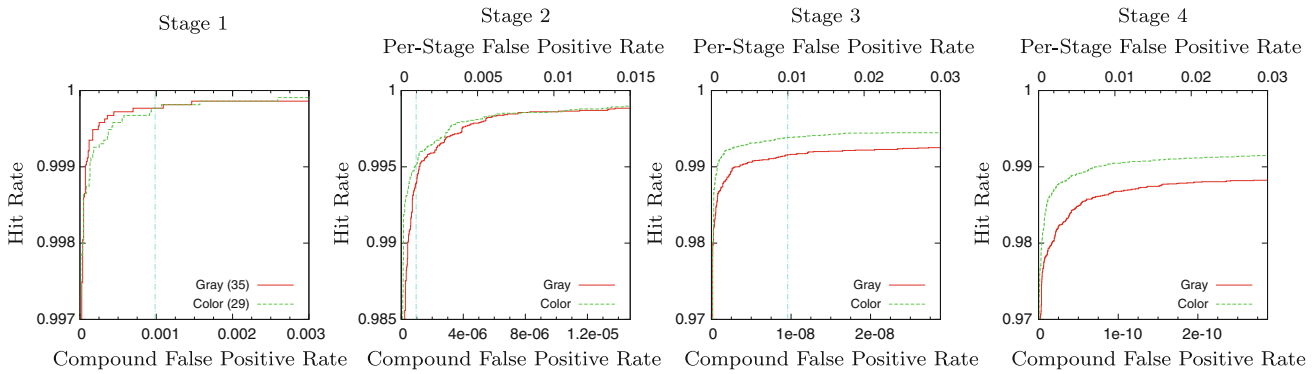

Fig. 15 Grayscale and color LiteHOG+ speed sign classifier ROC performance. As the color classifier is extended to contain more stages it begins to clearly outperform its grayscale counterpart. In stage 4 the **color LiteHOG+** classifier succeeds in achieving a 99% detection rate in conjunction with a $\mathbf{10^{-10}}$ false-positive rate. *Note 1* The *vertical line* shown in stages 1–3 represents the selected false-positive thresholding point specified by Table 5. *Note 2* Lower x-axis is labeled with the compound false-positive rate while the upper x-axis shows the per-stage false-positive rate of the data passing through the current stage only

Finally, we have demonstrated the construction of a LiteHOG+ traffic sign cascade classifier able to achieve a hit rate of 99% with a false-positive rate of just 10^{-10} when using color information.

10 Future work

The use of centre-surround image statistics, such as that used in the LiteHOG+ feature, allow for the creation of powerful (low false-positive rate) initial classifier stages for a large variety of traffic signs. Analysis shows that such low false-positive rates in the initial stages that significantly increasing the computational complexity of subsequent stages does not greatly affect the *average* time-to-decision of the overall classifier. However for some sign types (such as no entry signs), the subsequent stages struggle to reduce the false-positive rate to an acceptable level. While the inclusion of color information closes this gap significantly there is still a great need for more powerful classifier features to drive the success of the tail-end classifier cascade stages.

References

1. Alefs, B., Eschemann, G., Ramoser, H., Beleznai, C.: Road sign detection from edge orientation histograms. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV) (2007)
2. Bahlmann, C., Zhu, Y., Ramesh, V., Pellkofer, M., Koehler, T.: A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), pp. 255–260 (2005)
3. Bishop, C., et al.: Pattern recognition and machine learning. Springer, New York (2006)
4. Broggi, A., Cerri, P., Medici, P., Porta, P., Ghisio, G.: Real time road signs recognition. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), pp. 981–986. IEEE (2007)
5. Burges, C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov. **2**(2), 121–167 (1998)
6. Crow, F.: Summed-area tables for texture mapping. In: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques, pp. 207–212. ACM, New York (1984)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 886–893 (2005)
8. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification (2nd edn.). Wiley-Interscience, New York (2000)
9. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. **28**(2), 337–407 (2000)
10. Kawakami, R., Tan, R.T., Ikeuchi, K.: Consistent Surface Color for Texturing Large Objects in Outdoor Scenes. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), vol. 2 (2005)
11. Lafuente-Arroyo, S., Gil-Jimenez, P., Maldonado-Bascon, R., Lopez-Ferreras, F., Maldonado-Bascon, S.: Traffic sign shape classification evaluation I: SVM using distance to borders. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), pp. 557–562 (2005)
12. Overett, G., Petersson, L.: On the importance of accurate weak classifier learning for boosted weak classifiers. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV) (2008)
13. Overett, G., Petersson, L.: Fast features for time constrained object detection. In: Feature Detectors and Descriptors: The State Of The Art and Beyond (CVPR Workshop) (2009)
14. Paisitkriangkrai, S., Shen, C., Zhang, J.: Fast pedestrian detection using a cascade of boosted covariance features. IEEE Trans. Circuits Syst. Video Technol. **18**(8), 1140–1151 (2008)
15. Paulo, C.F., Correia, P.L.: Automatic detection and classification of traffic signs. In: International Workshop on Image Analysis for Multimedia Interactive Services, p. 11 (2007)
16. Pettersson, N., Petersson, L., Andersson, L.: The histogram feature – a resource-efficient weak classifier. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV) (2008)
17. Porikli, F.: Integral histogram: a fast way to extract histograms in cartesian spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1 (2005)
18. Rosenberg, C., Hebert, M., Thrun, S.: Color constancy using kl-divergence. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 239–246 (2001)
19. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Mach. Learn. **37**(3), 297–336 (1999)
20. Timofte, R., Zimmermann, K., Van Gool, L.: Multi-view traffic sign detection, recognition, and 3D localisation. In: Applications

of Computer Vision (WACV), 2009 Workshop on, pp. 1–8. IEEE (2010)
21. Tuzel, O., Porikli, F., Meer, P.: Pedestrian detection via classification on Riemannian manifolds. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1713–1727 (2008)
22. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, p. 511. IEEE Computer Society (2001)
23. Viola, P., Jones, M.: Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade. Adv Neural Inf. Process. Syst. (ANIPS) **2**, 1311–1318 (2002)
24. Wu, J., Mullin, M., Rehg, J.: Linear asymmetric classifier for cascade detectors. In: Machine Learning, vol. 22, p. 993 (2005)
25. Wulf, W.A., McKee, S.A.: Hitting the memory wall: implications of the obvious. SIGARCH Comput. Archit. News **23**(1), 20–24 (1995)
26. Zhang, C., Viola, P.: Multiple-instance pruning for learning efficient cascade detectors. In: Advances In Neural Information Processing Systems (ANIPS) (2007)
27. Zhang, L., Wu, B., Nevatia, R.: Pedestrian detection in infrared images based on local shape features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
28. Zhu, Q., Yeh, M.C., Cheng, K.T., Avidan, S.: Fast human detection using a cascade of histograms of oriented gradients. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1491–1498. IEEE Computer Society (2006)

Author Biographies

Gary Overett received his Ph.D. in Computer Vision and Bachelor of Software Engineering from the Australian National University in Canberra. He now works as a Research Engineer at NICTA improving object detection algorithms for automatic map creation for the AutoMap project. His research interests include pattern recognition and computer vision with a focus on fast and efficient algorithms for large video datasets.

Lachlan Tychsen-Smith is in his final year of completing a combined bachelor degree in Systems Engineering and Science majoring in Mechatronics and Theoretical Physics at the Australian National University. Following the successful completion of a Summer Scholarship in 2009/2010, he has been employed at NICTA as a part time Research Assistant in the Automap team. In this role he has worked on projects in the fields of 3D point cloud visualization and processing, data mining, and machine learning.

Lars Petersson has a M.Sc. in Engineering Physics with a focus on semi conductors and optics, and a Ph.D. in Computer Vision and Robotics, both from the Royal Institute of Technology in Stockholm, Sweden. Lars has been with NICTA for 8 years and has in that time worked on Advanced Driver Assistance Systems as a project leader for the NICTA Smart Cars project as well as automatic map creation through video analysis in the more recent AutoMap project. Prior to joining NICTA, he was a researcher with the Australian National University.

Niklas Pettersson completed his M.Sc. in Engineering Physics at Chalmers University of Technology, Gothenburg, Sweden in 2004. He continued on to work for seven years with NICTA in Canberra pursuing his interests in computer vision.

Lars Andersson completed his M.Sc. in Engineering Physics at Chalmers University of Technology in Gothenburg, Sweden. He has been working as a Research Engineer at NICTA in Canberra, Australia. Lars is focusing on design and implementation of high performance computer vision algorithms in general, and development of a distributed framework for training and evaluating object detection classifiers in particular.