Object Classification in Visual Surveillance Using Adaboost

John-Paul Renno, Dimitrios Makris and Graeme A. Jones
Digital Imaging Research Centre
Kingston University, Penrhyn Rd, Surrey,
KT1 2EE, United Kingdom.
{j.r.renno, d.makris, g.jones}@kingston.ac.uk

Abstract

In this paper, we present a method of object classification within the context of Visual Surveillance. Our goal is the classification of tracked objects into one of the two classes: people and cars. Using training data comprised of trajectories tracked from our car-park, a weighted ensemble of Adaboost classifiers is developed. Each ensemble is representative of a particular feature, evaluated and normalised by its significance. Classification is performed using the sub-optimal hyper-plane derived by selection of the N-best performing feature ensembles. The resulting performance is compared to a similar Adaboost classifier, trained using a single ensemble over all dimensions.

1. Introduction

Object Classification in the context of Visual Surveillance is mainly used to annotate events for indexing/retrieval purposes [16][19], although some times it is used as part of the tracking process [1].

Within visual surveillance applications, the classes sought are usually confined to pedestrians/non-vehicles [16,18], groups as well as combinations of derivatives from the hierarchy of vehicular types [16,18,21,22]. A typical surveillance system is usually comprised of motion detection followed by motion tracking [3,6,9,18,20,22]. Motion detection provides the object cues from which the tracker depends on to: maintain the temporal association of its tracked objects and hypothesise or destroy objects. Due to their temporal stability, object classification is usually applied to the descriptors obtained from the tracked objects.

The focus of this paper is object classification within for visual surveillance systems, with an emphasis on feature selection. The goal of classification is to determine the class of the trajectories maintained by a blob-based tracker. The classifier is developed using the Adaboost method discussed in [13] and trained in a supervised manner. The structure of this paper is comprised of a review of the features and classification techniques discussed in the literature, followed by a discussion of our surveillance system and proposed classifier. The final sections consist of the preliminary results which detail classifier performance followed by our conclusions.

2. Review

The goal of object classification is to assign a segmented object with some descriptive label. Given an object's feature vector representation, this is achieved by looking for the cues in a given feature space indicative of an objects' class. This is typically achieved by determining the decision boundaries between the classes represented within a labelled training set. Training is based either on manually labelled data (supervised) or on unlabelled data (unsupervised). Automatic training usually assumes an underlying distribution for the each of classes, labelling each sample accordingly. However, a common issue in both cases is the selection of the features that will be used for object classification

A simple classifier proposed in [20] uses assumptions about the expected repetition within the foreground mask to infer class. A heuristic is applied that assumes any repetitive motion within specific regions of a bounding box model to be the result of people or groups, i.e moving arms and legs. Unfortunately, the implied threshold is defined subjectively and too few features are used to ensure scalability across multiple video datasets. Similarly in [16], the common feature aspect ratio is used again with subjective threshold, i.e – people exhibit an aspect ratio less than one. They extend their methodology to incorporate a solution for running people by counting corners within the target, assuming vehicles exhibit more edges than pedestrians. Again, this method may fail in differing scenarios, especially if the camera has significant yaw, pitch or roll angles. In [21], classification is directed towards classifying vehicle classes from traffic scenes. They classify using features representative of the physical

1-4244-1180-7/07/$25.00 ©2007 IEEE

dimensions of objects in 3D, but, assume the presence of a calibrated ground-plane coordinate system.

Adaboost [13] provides a framework for training a strong classifier by combining many weak classifiers. In Computer Vision, Adaboost is used for binary pixel-level classification [25]. In [11] a tracker, based upon Adaboost, is proposed and adopts the work of [24] as an online procedure for evaluating multiple features. Their choice of feature-vector comprises a selection of 11 features: the local orientation histogram (8) and pixel colours (3) and apply the work in [24] to determine an online significance for each. In this paper, we adopt a similar philosophy to classification and feature selection. We find their ideas of using and evaluating a large number of features attractive and develop our methodology to encompass some aspects of this. Specifically, Adaboost is used to train an object feature-classifier ensemble per feature, where each has an assigned weighting determined a-priori, indicative of importance.

3. Visual Surveillance Architecture

A typical surveillance system comprises a motion detection module followed by some type of motion tracker [2,3,6]. These modules provide the functionality for the detection and subsequent tracking of objects around some scenario. Although informative, these systems do not provide an implicit expression of the type of object being detected and tracked, however, such types can be implied through analyzing movement correlations such as routes, pathways and entry exit zones[14]. These modules are the focus of the following sections giving some context for the classification algorithm/module.

3.1. Motion Detection.

The goal of the motion detection module is to determine the location and the spatial extent of the foreground objects within a video sequence. Figure 1 summarises the processes and data flow that form the motion detection module used in this work. Typically, during a background subtraction process a video frame is compared against a reference image (*background*) - *i.e*, a representation of the scene devoid of all active foreground objects. Acquiring a background representation is a common research problem for which many approaches have been developed [2,3,4,6].

The differences are analysed during the segmentation phase during which a pixel's class: *foreground* or *background* is determined and represented in a segmentation mask; from this subsequent region analysis (*connected components*) groups the active pixels into moving regions (*blobs*), the output of the module. Depending upon the update methodology, the segmentation mask may be used in the update procedure to aid in maintenance of the background model such that it reflects the current appearance of the scene.

The motion detection technique used in this paper relates closest to that of Bowden et al [7] which itself is an extension of the work addressed in [3]. Their methodology assumes a multi-modal background distribution and as such adopts multiple models to represent both foreground and background. This enables the algorithm to be robust in the presence of changing conditions within the background, *i.e* – swaying leaves and lighting changes. Each of the models represents a particular class and ordered according to their relative significance. Background subtraction is performed much in the same way as discussed above, except that each model is compared in the order of significance. Upon a statistical association to one of the models, background subtraction terminates and segmentation assumes the label of the matched model. After all pixels are classified, the segmentation mask is updated accordingly, with the blobs being determined in the usual manner. Finally, the segmentation mask is used within the update procedure to influence either the: 1) creation or update a foreground/background distribution or 2) promotion of a foreground model to background; this represents an assumption that a foreground pixel unchanged for a significant period of time has become static and part of the scene, *i.e* - a stopped vehicle.

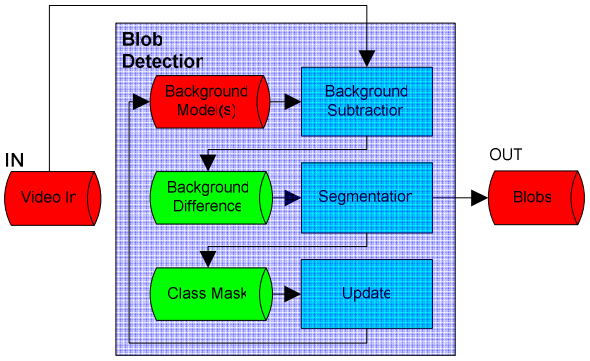


Figure 1: Motion Detection Module

3.2. Motion Tracking

In this work, the focus of tracking is to maintain the temporal association between corresponding blobs. Figure 2 shows an overview of the module adopted in this paper which focuses on blob based tracking using trajectory information. Blobs are passed into an observation parser

whose role is to extract features and represent blobs in the measurement space used in this module. In parallel, each object's measurement state is predicted by the motion-model used in the estimator process. It is the responsibility of the data-association process to determine the correct association between the blob and the predicted target measurements. The outcomes of the association process are used to either update the existing trajectory models or to hypothesize the possible existence of new objects.

Usually, blob trackers utilise either the a: Kalman filter [8,9] or a Particle filter [10,12] to fuse measurements with the associated target state information. Kalman filtering is based on Gaussian densities which, being uni-modal, cannot represent simultaneous alternative hypotheses [12], but due to its simplicity and robustness in mid to long range scenarios, it is adopted in this work. The tracking methodology discussed by Xu and Ellis in [8] uses Kalman tracking and is adopted in this paper and mapped by the module in figure 2. Their work is ideally suited because of its blob based approach; its novelty in the data-association process and its ability to resolve tracking during periods of dynamic occlusion. In this work the data-association process of [8] is extended to handle cases of blob fragmentation.

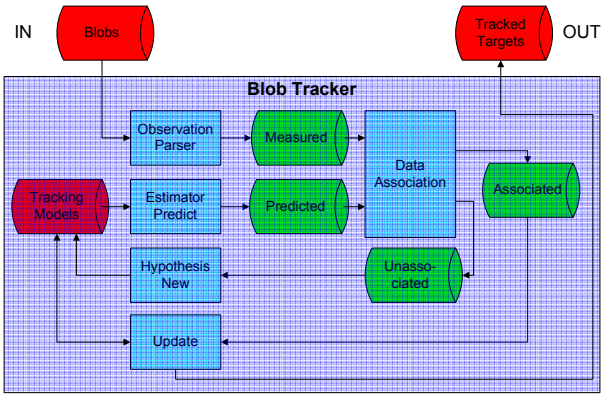


Figure 2: The tracking module

3.3. Object Classification using Adaboost

Classification uses the output of the previous module. The main objective of this work is to classify tracked targets into the classes: *person* or *vehicle*. Additionally, we want to determine the best feature space for object classification for different scenes. The classification methodology comprises an initial training phase followed by a quantitative analysis in-order to determine feature significance. A final normalization of the classifier ensembles yields the classifier: a weighted linear combination of Adaboost ensembles.

3.3.1 The Feature Classifier

The classifier used in this work is based upon Adaboost classifier and is comprised of a weighted linear combination of these to achieve its final classification result. Adaboost itself trains an ensemble of weak-learners to form a *strong classifier* which is guaranteed perform at least as well as an individual weak-learner. In this work, each Adaboost ensemble models one particular feature or dimension, where each feature represents an observable or derivable quantity associated with a tracked target, i.e {*width, height, area*, *aspect-ratio}* etc.

- *The Weak Learner*

The goal of Adaboost is to determine an ensemble of weak-learners which at worst exhibits a performance equal to that of an individual weak-learner. The process is known as boosting, and depending on the application can significantly improve performance. Adaboost relies upon an iterative procedure of re-weighting its training data such that subsequent weak-learners will best represent the training data that is poorly represented by the weak-learner ensemble. Within the context of re-weighted data, weighted least squares (*WLS*) enjoys the ability to handle regression situations in which the data points are of varying quality. Given its suitability toward weighted data, *WLS* is an ideal choice of weak-learner trainer and as such is adopted in this work. In [11], *WLS* is used in by Adaboost to develop an ensemble of 11-dimensional weak-classifiers. In this paper a similar approach is adopted, except that each feature is considered independent and modelled within its own feature ensemble. Later, a method for combining the individual ensembles is developed. The general form of an Adaboost weak-classifier discriminant function is given by equation (1):

$$h(\mathbf{x}) = sign\,(\mathbf{h}^{\mathrm{T}}\mathbf{x}), \quad h(\mathbf{x}) : \Re^{d} \rightarrow \{-1,+1\} \qquad (1)$$

where $\boldsymbol{h}$ represents the discriminating hyper-plane. Considering each feature independently, *i.e* $d = 1$, our discriminant $\boldsymbol{h}_{\mathrm{k}}$ of the weak-classifier $h_k(\mathbf{x}_k) : \Re^1$ for dimension k is given by:

$$\mathbf{h}_k = \left(\mathbf{A}_k{}^{\mathrm{T}}\mathbf{W}\mathbf{A}_k\right)^{-1}\mathbf{A}_k{}^{\mathrm{T}}\mathbf{W}\mathbf{y}, \qquad \mathbf{h} \in \Re^{d=1} \qquad (2)$$

$\boldsymbol{x}$	=	The feature-vector consisting of k dimensions.
$\boldsymbol{x}(k)$	=	The feature value of the k^{th} dimension of $\boldsymbol{x}$.
$\boldsymbol{x}_k$	=	The feature value of the k^{th} dimension augmented with a 1, *i.e* $\boldsymbol{x}_k = [\boldsymbol{x}(k), 1]$
$\boldsymbol{A}_k$	=	A matrix of feature values augmented with a 1 for the k^{th} dimension, *i.e* - the l^{th} row $\boldsymbol{A}_l = [\boldsymbol{x}_k, 1]$
$\boldsymbol{W}$	=	A square matrix whose diagonal contains a weight representative of samples quality.
$\boldsymbol{y}$	=	A list of class labels for each feature value in $\boldsymbol{A}_k$.

Table 1: The variable definitions of equations (1&2)

- *Training the strong classifier*

During an offline process, a database containing object trajectories is perused to obtain a suitable training set $\boldsymbol{X}=[\boldsymbol{X}_1,..,\boldsymbol{X}_b,..,\boldsymbol{X}_L]$ of length L. For each track $\boldsymbol{X}_l$ in the set $\boldsymbol{X}$ there exists: n_l samples of each feature k, *i.e* $\boldsymbol{X}_{k,l} = [\boldsymbol{x}_{k,1},\ldots,\boldsymbol{x}_{k,nl}]^T$ *(see*-table 1*)* and their corresponding class labels denoted by the vector $\boldsymbol{Y}_l$. Using this notation, $\boldsymbol{A}$, $\boldsymbol{y}$, and $\mathbf{W}$ are expressed as follows:

$$\mathbf{A}_k = \left[\mathbf{X}_{k,1} \;,...,\; \mathbf{X}_{k,l} \;,...,\; \mathbf{X}_{k,L}\right]^T \qquad \mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & 0 & 0 & 0 & 0 \\ 0 & . & 0 & 0 & 0 \\ 0 & 0 & \mathbf{w}_l & 0 & 0 \\ 0 & 0 & 0 & . & 0 \\ 0 & 0 & 0 & 0 & \mathbf{w}_L \end{bmatrix}$$
$$\mathbf{y} = \left[\mathbf{Y}_l \;,...,\; \mathbf{Y}_l \;,...,\; \mathbf{Y}_L\right]^T \qquad (3)$$

where $\boldsymbol{W}_l$ represents a square diagonal matrix of the weighting for each feature sample of track $\boldsymbol{X}_l$. Initially each weight has equal an importance within the context of its class and all weights sum to one[1], *i.e* - $Tr(\boldsymbol{W}) = 1$. Training proceeds by: iteratively solving (2) to obtain a new weak classifier $h_{k,j}(\boldsymbol{x}_k)$ and its relative weight α_j. Following this, $\boldsymbol{W}$ is adjusted and normalised to reflect the training data's representation by the ensemble of weak-classifiers. Training terminates when a subsequent new weak-classifier performs no better than chance, *i.e* accuracy less than 0.5. At this point the *strong classifier* $H_k(\boldsymbol{x})$ for the k^{th} feature is determined by the J weak-learners and their relative weights, shown by equation (4):

$$H_k(\mathbf{x}) = \sum_{j=1}^{J} \alpha_{k,j} h_{k,j}(\mathbf{x}_k)$$
$$H_k(\mathbf{x}) : \Re^1 \rightarrow [-1,+1] \qquad (4)$$

where in this work $\sum_{j=1}^{J} \alpha_{k,j} = 1$.

[1] $Tr(\boldsymbol{X})$ represents the trace of matrix $\boldsymbol{X}$.

- *The Strong Feature Classifier Ensemble*

In this paper a weighted linear combination of strong feature classifiers is used to formulate the final classifier. Combining classifiers in this way resolves issues relating to feature selection, ensuring that all features may be reflected in the final classification result. Relegating the discussion of determining a feature's significance w_k to the next section, the form of the resulting ensemble of feature classifier ensembles $G(x)$ is given as:

$$G(\mathbf{x}) = \sum_{\forall_k \in \mathbf{x}} w_k H_k(\mathbf{x}) \qquad G(\mathbf{x}) : \Re^k \rightarrow [-1,+1] \qquad (5)$$

and the weights w_k representative of feature significance are normalised such that:

$$\sum_{\forall_k \in \mathbf{x}} w_k = 1 \qquad (6)$$

3.3.2 Feature Selection/Significance

In the previous section the individual Adaboost classifiers where modelled upon an individual feature of the training set and then combined to form the final ensemble of classifier ensembles. The motivation for segregating the feature classifiers is to get a handle on the significance of particular features with respect to a given scenario, *i.e* - to determine a feature's suitability for the dataset to hand. In this section a technique for ranking the feature classifiers is proposed.

- *Assessing Feature Classifier Performance*

The significance for each of the features classifiers is assessed through quantitative evaluation of its performance. This is determined on a second independent training dataset drawn at random and hand labelled. Each of the feature classifiers $H_k(\boldsymbol{x})$ encapsulated by the feature classifier ensemble $G(\boldsymbol{x})$ are applied to the samples of this training set. Each feature classifier is subsequently evaluated with a view to determine its error rate e_k. The error rate is representative of the proportion of the training set assigned an incorrect label by the k^{th} feature classifier and is used to determine a feature classifier's relative significance within the ensemble $G(x)$.

$$w_k = (1 - e_k) / \sum_{\forall_k \in \mathbf{x}} (1 - e_k) \qquad (7)$$

As well as identifying the significance of a particular feature, ranking in this way provides flexibility in the formulation of the final classifier $G(x)$, *i.e* - use all of the feature classifiers or the N best? Ultimately a flexible methodology is suggested which considers features on

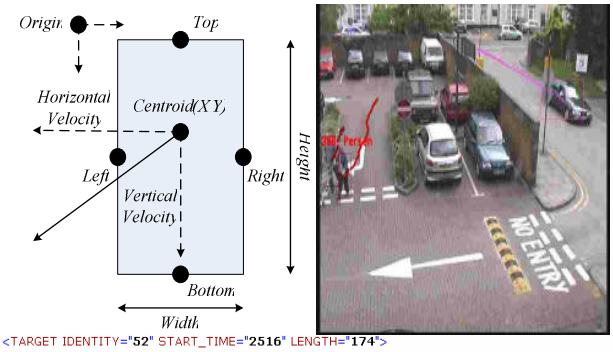
their own merit, setting aside problems of feature selection and allowing for the testing of a plethora of features.

3.4. The Features and their Metrics

The metrics relate to the method of computation of the features within the feature vector $\boldsymbol{x}$. The choice of features is usually determined through an offline analysis of their class separability in the feature-space. This work diverts attention from this by considering all possible features: measurable and derivable, from a small set, *i.e* those maintained by a tracker. An example of features maintained by our tracker can be seen in the in Figure 3. Under each XML node named 'TRAJECTORY', a list of feature values are shown at a particular time. Ten features are used, each maintained within the tracking model, from which a further 11 are derived. The list of the features used in this paper can be seen below in Table 1, including their identifying key and computation metric. Figure 3 provides the key that embodies the relationship between the features and the trajectory information. Normalisation refers to the scaling of a feature's metric according to its vertical position, since from observation, it can be seen that objects become smaller/larger as they move up/down the image. To gauge the impact of scale, the feature metric's are given in both their normalised and un-normalised forms.

ID	Feature Name	Norm-alised	Metric
1	Aspect Ratio	No	Height / Width
2	Width	Yes	Width / Bottom
3	D-Right	Yes	(Right - CentroidX)/Bottom
4	D-Left	Yes	Left – CentroidX.
5	Area	Yes	Height * Width / Bottom
6	D-Right	No	Right – CentroidX
7	Width	No	Width
8	D-Left	No	(Left – CentroidX)/Bottom
9	Dispersedness	No	$4x[Width+Height]^2$/ [Width * Height]
10	Area	No	Width * Height
11	D-Top	Yes	(Top – CentroidY)/Bottom
12	Velocity Horizontal	Yes	VelocityX / Bottom
13	Velocity Horizontal	No	VelocityX
14	Velocity Vertical	No	VelocityY
15	Velocity Vertical	Yes	VelocityY/Bottom
16	D-Bottom	No	Bottom – CentroidY
17	Height	No	Height
18	D-Top	No	Top – CentroidY
19	Height	Yes	Height / Bottom
20	Dispersedness	Yes	$(4x[Width+Height]^2$/ [Width * Height]) / Bottom
21	D-Bottom	Yes	(Bottom – CentroidY) / Bottom

Table 2: List of the features and their associated metrics




```
<TARGET IDENTITY="52" START_TIME="2516" LENGTH="174">
- <INSTANCE NUMBER="0">
  - <TRAJECTORY>
      <POSITION X="595.4747" Y="192.5942" Z="0" />
      <BOUNDINGBOX LEFT="578" TOP="203" RIGHT="623" BOTTOM="186" />
      <VELOCITY VX="0" VY="0" VZ="0" />
    </TRAJECTORY>
  </INSTANCE>
- <INSTANCE NUMBER="1">
  - <TRAJECTORY>
      <POSITION X="595.2095" Y="192.5296" Z="0" />
      <BOUNDINGBOX LEFT="577.3674" TOP="202.9677" RIGHT="622.8674" BOTTOM="185.9677" />
      <VELOCITY VX="-0.1323239" VY="-3.223151E-02" VZ="0" />
    </TRAJECTORY>
  </INSTANCE>
```

Figure 3 – Top-left: Tracking Model Key, Top-right: The Video Data – MPG2 - 25fps - 12mbs – 720x576, Bottom: Example of the Xml tracking schema for a tracked target.

4. Results

In this section the proposed motion detection and tracking modules are used to track targets around a scene. Each of the target trajectories are stored during an offline process and labelled for both the training and testing sets. The training sets are used to both determine each feature's significance and develop the feature classifier ensembles. The proposed classifier is subsequently applied to the test dataset to assess its performance. Finally, a comparison of performance is made between the proposed classifier and a similar classifier trained using Adaboost applied to the multi-dimensional feature vector $\boldsymbol{x}$.

4.1. Experimental: The Scenario

To test the robustness of the proposed classifier, an offline detection and tracking of a car-park scenario was performed. Using the fore-mentioned motion detection and tracker modules, the scene shown in *figure 3* was monitored for a period of one day. During this time, any visible objects were tracked and stored within a XML file. The features of each track are stored iteratively over time and represent those used within the tracking model - see *figure 3*. In-order to assess classifier performance, three sample sets were selected at random. The first two sets represent those required to perform the training and quantitative analysis of the feature classifier ensembles; the third set is used for testing. All samples within these sets were hand labelled and used to:

1) Train each of the Adaboost Feature Classifier's – *see* equation (4). The training dataset consisted of 10 people and 10 vehicle tracks totaling 4716 samples of the feature-vector.

2) Quantify significance to determine each of the Feature Classifier's relative weight – see equations (5&6). Again, the analysis dataset comprised of 10 people and 10 vehicle tracks comprising of 4215 samples.

3) Assess the performance of the weighted vs the un-weighted feature classifier ensembles in the following configurations:

   - *All Ensembles* – This configuration represents the classifier when all of feature classifier ensembles are used.

   - *N-Best Ensembles* – This configuration represents the classifier when the *N* most significant feature ensembles are used – equation (6) still applies.

4) For comparison purposes, assess the performance of a classifier trained using the traditional Adaboost algorithm discussed in [13]. The configurations are:
   - *Multi-Dimensional* – Adaboost is used to determine a classifier trained on the entire feature vector. This is the typical manner in which Adaboost is used. This results in one ensemble of classifiers representative of the entire feature vector.

For testing purposes, a comparatively large set of 200 targets was randomly selected which comprised of 75 people and 125 vehicles objects, totaling 44509 samples.

4.2. Plots of the features in the Testing Data

The data plots for 10 features in the testing dataset are presented in Figure 4. Each feature value is plotted against image height and un-normalised. With the exception of the 'aspect-ratio', both a high dependency between the feature value and the vertical image position as well as significant overlap between class distributions can be observed. The 'aspect ratio' feature appears to be relatively invariant to vertical position and exhibits a large separability between the class distributions. From this observation and within the context of this dataset, we can assume that this feature should achieve a high significance value.

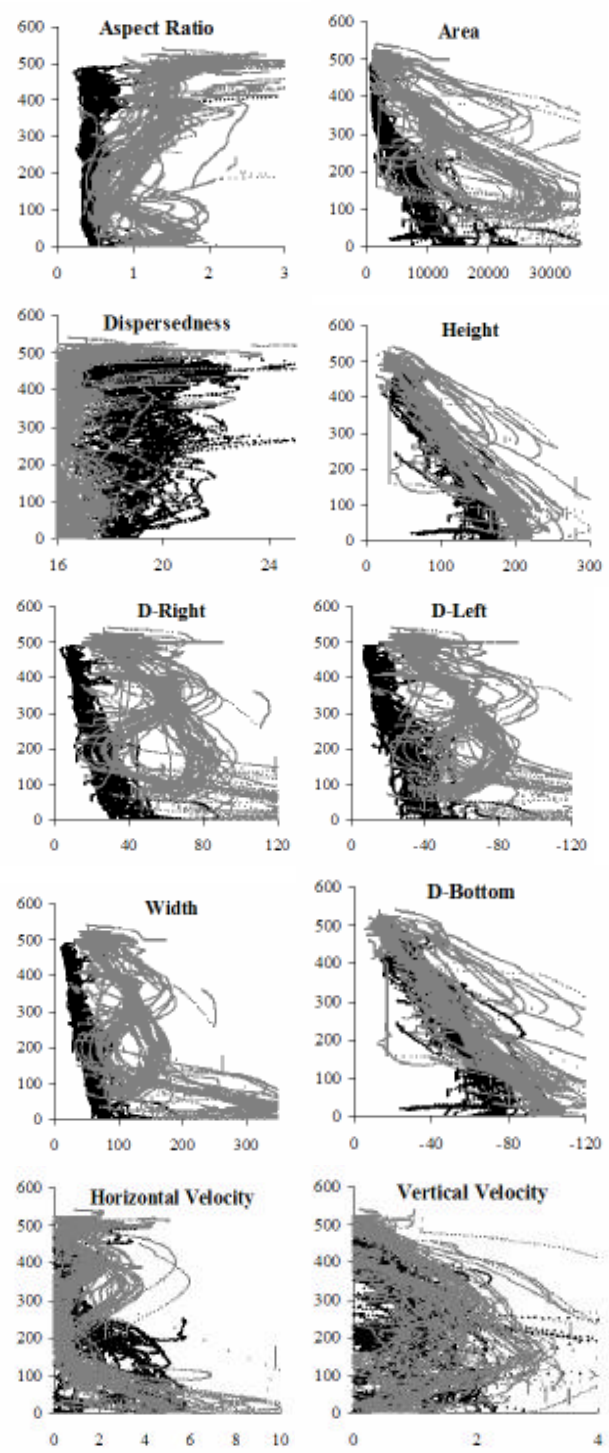


Figure 4: The plots of each un-normalised feature value (*x-axis*) vs its vertical image (*y-axis*) position measured from the testing dataset. The darker and lighter points represent people and vehicle samples respectively.

4.3. Ranking of the Feature vectors

In this section the outcomes of the feature classifier boosting and subsequent quantitative analysis are presented. The errors of the feature classifiers before and after boosting are shown in the graph of figure 5. The graph is quite reveals the following: 1) a high degree of variability within the error distributions of the feature classifiers, 2) the poorer performing feature classifiers are less likely to benefit from boosting and 3) the importance of normalisation; for those feature classifiers achieving an error rate less than 10%, those that are normalised perform marginally better than their un-normalised counterparts. The weak-learner for D-Bottom' was unable to perform better than that of chance and as a consequence was dropped from the feature vector. Aside from the observations mentioned previously, the graph gives an indication of each feature classifier's significance, subsequently used to determine their relative weighting according - *see* equation (7). As observed in [19], the 'aspect ratio' feature exhibits the largest accuracy.

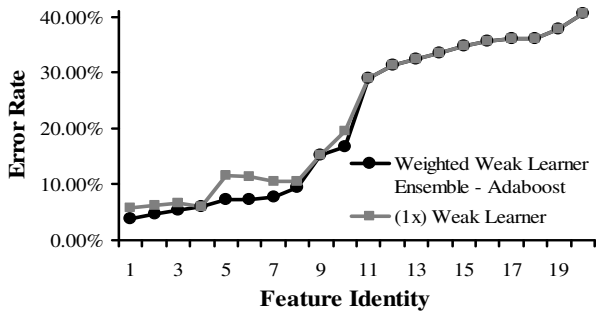


Figure 5: A plot of the ranked feature classifiers according to their error rate during classification of the 2nd training dataset.

4.4. Performance on the Testing set

In this section the various configurations of the feature classifiers are assessed in their ability to correctly label targets as well as their samples. For completeness a hierarchical evaluation is performed consisting of two categories: *object* and *sample* based evaluation. Sample based evaluation measures the classifiers ability to correctly label the individual track instances. Similarly, object evaluation measures the classifiers ability to correctly label an object. In this work an object's label is assumed to be the class with the largest vote given all classified instances for each track. To assess the significance of boosting, each of the hierarchies is further sub divided into the: boosted and un-boosted cases. The outcomes of the evaluation are presented in the graph of Figure 6. It shows the impact upon performance when increasing the number of feature classifiers within the ensemble $G(\boldsymbol{x})$ according to the N-Best criteria. The end points for each plot represent the performance of the ensemble when all feature classifiers are used. What is evident in these results is the importance of having prior knowledge relating to feature significance. In both the object and sample cases of uniformly weighted feature classifier ensembles, the accuracy is be seen to be dropping off significantly as additional weaker feature classifiers are added.

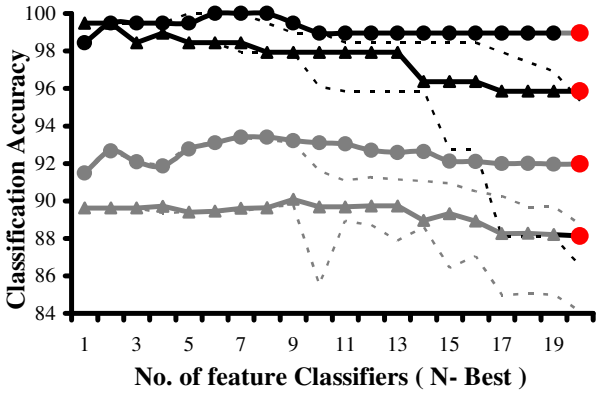


Figure 6: Weighted Object (circles) and sample (triangles) based evaluation represented in both: boosted (black) and un-boosted (grey) setups. The highlighted end points represent the performance when $G(\boldsymbol{x})$ encompasses all feature classifiers. The dotted plots represent the case of uniform weighting in $G(\boldsymbol{x})$.

To determine the relative performance of the proposed classifier, its accuracy is compared to that of a classifier trained using standard Adaboost. This entailed training an ensemble of weak-learners on the feature vector to determine its hyper-plane discriminant function. Table 2 compares the performances of the proposed and fore-mentioned classifier's on the testing dataset.

Classifier configuration (d=21)	**Accuracy (%)**	
	Sample	**Object**
All ensembles – *Boosted*	*91.98*	*98.96*
All ensembles	*88.12*	*95.85*
d-dimensional – *Boosted*	*94.46*	*97.92*
d-dimensional	*90.01*	*94.45*

Table 3: d-dimensional Adaboost compared to the proposed ensemble approach.

5. Conclusions & Future work

The work presented in this paper presented a novel use of the Adaboost algorithm to classify tracked objects to pedestrians or vehicles. A method for determining a feature selection criterion was proposed and subsequently used to infer a feature classifier's significance in the form of a weight; normalisation was implemented in the form of a weighted linear combination of the feature classifiers. The proposed method was applied to a testing dataset and

was shown to perform well, especially after boosting. The proposed technique was evaluated against a similar classifier trained using Adaboost on the entire feature vector, (*d=21*). After training, this classifier was composed of 28 weak-learners giving an error rate of only 1% on the training set. As well as drastically cutting the training times, our proposed method configured for all dimensions showed to marginally outperform its rival in the object based evaluation, although the *d*-dimensional Adaboost classifier achieves a marginal victory in the sample based evaluation. This indicates that the distribution of erroneous samples is not uniformly distributed across object tracks, possibly the result of over-fitting of the decision boundary to the training data.

This work represents an initial platform from which to develop a multi-class classification system for use in learning a model of scene semantics. In the future we intend to further evaluate the technique on additional larger datasets using yet more features. Other possible extensions include: experimentation with other weak-learner's and feature selection for *d*-dimensional classifier training.

References

[1] P. Remagnino, A. Baumberg, T. Grove, D. Hogg, T. Tan, A.D. Worrall, K. Baker, "An Integrated Traffic and Pedestrian Model-based Vision System", British Machine Vision Conference, 2, BMVA, 9/8, UK, pp. 380-389. (1997).

[2] J. Aguilera, D. Thirde, M. Kampel, M. Borg, G. Fernandez and J. Ferryman, "Visual Surveillance for Airport Monitoring Applications", 11th Computer Vision Winter Workshop, 2006.

[3] C. Stauffer, W.E.L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking", IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149) Part Vol. 2, 1999.

[4] R. Cucchiara, C. Grana, M. Piccardi and A. Pratti, " Detecting Objects, Shadows and Ghosts in Video Streams by Exploiting Color and Motion Information", ICIAP '01: Proceedings of the 11th International Conference on Image Analysis and Processing, 2001.

[5] P.L. Rosin and T. Ellis. "Image Difference Threshold Strategies and Shadow Detection". In Proceedings of the 6th British Machine Vision Conference (BMVC'95), volume 1, pages 347{356, University of Birmingham, Birmingham, September 11-14 1995.

[6] N.T. Siebel and S.J. Maybank. "Real-time Tracking of Pedestrians and Vehicles". In Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, Hawaii, December 2001.

[7] P. KaewTraKulPong and R. Bowden, "An Improved Adaptive Background Mixture Model for Real-Time with Shadow Detection", In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems-AVBS01 – Sept 2001.

[8] M. Xu and T. Ellis, "Partial Observation vs Blind Tracking through Occlussion", British Machine Vision Conference, 2002.

[9] J. Black, M. Xu and T. Ellis, "Multi-View Image Surveillance and Tracking", Proc of the Workshop on Motion and Video Computing, 169-174, 2002.

[10] S.K. Zhou, R. Chellappa and B. Moghaddam, "Visual Tracking and Recognition using Appearance-Adaptive Models in Particle Filters", IEEE transactions on Image Processing, Vol13, No. 11 (1491-1506), Nov 2004.

[11] S. Avidan, "Ensemble Tracking", IEEE Transactions on Pattern Alalysis and Machine Intelligence, Vol29, No.2 (261-271), 2007.

[12] M. Isard and A. Blake, "Condenstation – conditional density propagation for visual tracking", International Journal of Computer Vision, Vol29, No.1 (5-28), 1998.

[13] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Computational Learning Theory, Eurocolt 95, (23-37), 1995.

[14] D. Makris and T. Ellis, "Finding Paths in Video Sequences", British machine vision conference, 2001.

[15] J. Black, T. Ellis, D. Makris, "A Hierarchical Database for Visual Surveillance Applications", IEEE International Coference on Multimedia and Expo (ICME2004), Taipei Taiwan, pp. 1571-1574, June 2004.

[16] Q. Zang and R.Klette, "Object Classification and Tracking in Video Surveillance", In Proc. Computer Analysis of Images and Patterns, pp 198-205, Berlin 2003.

[17] C.Stauffer, "Factored Latent Analysis for far-field Tracking data",

[18] R. Visser, N. Sebe and E. Bakker, "Object Recognition for Video Retrieval", International Conference on Image and Video Retrieval, Vol.2383, pp 250 – 259, 2002.

[19] C.O. Conaire, N. O'Connor, E.Cooke and A.F. Smeaton, "Multispectral Object Segmentation and Retrieval in Surveillance Video", In Proceedings of IEEE International Conference on Image Processing (ICIP), Atlanta US, October 8-11, 2006

[20] J-L Landabaso, L-Q Xu, M. Pardas, "Robust Tracking and Object Classification towards Automatic Video Surveillance", International Conference on Image Analysis and Recognition ICIAR(2), pp 463-470, 2004

[21] A. Lai, G. S.K. Fung, N.H.C. Yung, "Vehicle Type Classification from Visual-Based Dimension Estimation", In Proc Intellignet Transportation Systems, 2001.

[22] J-W. Hseih, S-H. Yu, Y-S Chen and W-F Hu, "Automatic Traffic Surveillance System for Vehicle Tracking and Classification", IEEE Trans on Intelligent Transpotation Systems, Vol 7 Issue 2, pp 175-187, 2006.

[23] S. Nene and S. Nayer, "A Simple Algorithm for nearest neighbour search in high dimensions", IEEE Trans on Pattern Analysis and Machine Vision, vol.19, 1997.

[24] R.T. Collins, Y. Liu and M. Leordeanu, "On-Line Selection of Discriminative Tracking Features", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 10, pp. 1631-1643, Oct., 2005.

[25] P.Viola and M.Jones, "Real-time Object Detection", Cambridge Reseacch Laboratory, Technical Report Series, 2001.