Boosting with Cross-Validation Based Feature Selection for Pedestrian Detection

Kenji NISHIDA and Takio KURITA

Abstract— **An example-based classification algorithm to improve generalization performance for detecting objects in images is presented. The classifier integrates component-based classifiers according to the *AdaBoost* algorithm. A probability estimate by a kernel-SVM is used for the outputs of base learners, which are independently trained for local features. The base learners are determined by selecting the optimal local feature according to sample weights determined by the boosting algorithm with cross-validation. Our method was applied to the MIT CBCL pedestrian image database, and 54 sub-regions were extracted from each image as local features. The experimental results showed a good classification ratio for unlearned samples.**

I. INTRODUCTION

According to government statistics[1], 6,871 people were killed and 1,156,633 were injured in 933,828 traffic accidents in 2005 in Japan. Therefore, preventing traffic accidents is one of the most urgent issues in our society.

Drivers make decisions according to what they recognize in their driving environment, mainly on the basis of visual information, and failures in recognition of the environment may cause an accident. Actually, these statistics indicate that about 70% of these accidents were caused by failures in recognition of the driving environment. In particular, failures in the recognition of pedestrians may cause serious injuries or death. Therefore, we focused on pedestrian detection in images from an on-board camera and constructed a driver assistance system.

Pedestrian detection is more challenging than detecting other objects such as cars and faces because since people have a variety of shapes and sizes, and defining a single model that captures all of these possibilities is diffcult. Therefore, classifiers for pedestrian detection must have high generalization performance. Previous studies tried to resolve this problem.

Gavrila[4] used hierarchical template matching to find pedestrian candidates from incoming images. His method provides in advance multiple templates that are outline edge images of typical pedestrians, and dissimilarities (or similarities) between the edge feature of incoming images were measured by the chamfer distance. The variety of shapes and sizes of pedestrians was accommodated with a variety of templates, which bound system performance.

Viola et al.[5] presented a pedestrian detection system that integrates image intensity information with motion information. Their detection algorithm scans a detector over two consecutive frames of a video sequence, and the detector was trained using *AdaBoost* to take advantage of both motion and appearance information. They achieved a high detection speed (about four frames/second) and a very low false positive rate, while combining two different modalities of information in one detector.

Although they showed the advantage of integrating motion information, applying their algorithm to an on-board pedestrian detection system is still difficult because of the difficulty of canceling out the movement of the camera only from visual information. Therefore, we focused on pedestrian detection from static images to achieve our example-based object detection method.

Mohan et al.[3] applied an adaptive combination of classifiers (ACC) to pedestrian detection. Their system consists of two-stage hierarchical classifiers. The first stage is structured with four distinct example-based classifiers, which are separately trained to detect different components of pedestrians, such as the head, legs, and the right and left arms. The second stage has an example-based classifier that combines the results for the component detectors in the first stage to classify the pattern as either a "person" or a "non-person." A support vector machine (SVM)[6][7][8] is useed for each classifier. Their results indicated that a combination of component-based detectors performed better than a full-body person detector. The components in their system were determined in advance, and they were not exactly optimal to classify the examples.

One of the most important problems in determining classifiers is how to avoid overfitting a set of training samples while preserving classification accuracy. *AdaBoost* empirically has such property when each base learner is weak enough, although the reasons are not clear yet. Some previous studies showed that random feature selection improves the generalization ability of ensemble classifiers [14], [16], and some other studies indicated the effect of the margin of each base learner. Shapire tried to represent the bounds of generalization error using margin distribution of classifiers in [12], and Breiman tried to give a sharper bound of generalization error by introducing the maximal margin error of ensemble classifiers [15]. These considerations give good interpretations for generalization error of ensemble classifiers in most cases. Therefore, we determined that maximizing the margin of base learners in boosting would be effective. The original *AdaBoost* algorithm determines the base learners according to the error rate for training samples (training error). However, it does not help maximizing the margin of base learners. Breiman [15] and Lu [13] proposed minimizing

Kenji NISHIDA and Takio Kurita are are with the Neuroscience Research Institute of the National Institute of Advanced Industrial Science and Technology (AIST), Central 2, 1-1-1 Umezono, Tsukuba 305-8568, Ibaraki, Japan (email: {kenji.nishida, takio-kurita}@aist.go.jp).

978-1-4244-1821-3/08/$25.00©2008 IEEE

cross-validation error instead of training error to determine large margin classifiers for the base learners.

In our previous study, we examined the effect of boosting soft-margin SVM with feature selection [9], and it showed fairly good classification performance. However, we had to hand-tune the soft-margin parameter to attain an adequate weakness for base learners, and when base learners were determined to be too weak, more boosting steps were required to attain a good classification accuracy. Therefore, a systematic algorithm to determine the weakness (or margin) of base learners was necessary.

In this paper, we used cross-validation error for local-feature selection instead of training error to improve the margin of base learners. The SVM parameters, such as soft-margin cost and gaussian width, were also determined through a grid search to minimize the cross-validation error. The local-features were limited to some position variations of the pedestrian components to avoid over-fitting by selecting inappropriate local regions. Boosting determines the classifier weights through step-wise optimization. Therefore, they are not exactly optimal. We also examined the effect of overall optimization of classfier weights after all the base-learners were determined.

Our experimental results show that our algorithm achieves a good classification ratio for unlearned samples. Our method can be applied to any object composed of distinct identifiable parts that are arranged in a well-defined configuration, such as cars and faces. We describe our object detection method in the next section, and the experimental results are presented in the final section.

II. Feature Selection based on a Cross-Validation Estimate

Our key idea is introducing cross-validation for feature selection while boosting to improve the generalization performance (margin) of base learners to preserve their classification accuracy. We previously examined a soft-margin SVM for base learners of boosting[9] to adopt a certain weakness. However controlling the generalization performance of the strong (integrated) classifier was difficult, and it required large computation. In our method, we computed the probability estimate of a pedestrian image for 54 sub-regions (selected as position variations for pedestrian components) in advance, and the collection of the results was treated as a feature vector for each sample. We then applied feature selection on the feature vectors to ensemble their results. boosting determines the classifier weights by a step-wise optimization procedure, the weights were not exactly optimal. Therefore, we also examined the overall optimization of the classifier weights using a linear SVM.

In this section, we first briefly describe an SVM that outputs the probability estimate and the boosting algorithm, because our method used these algorithms for determining an ensemble classifier. We then present sample images and their local features. At the end of this section, we describe our method, in which base learners threshold the probability estimates of sub-regions based on cross-validation while selecting optimal features.

A. Support Vector Machine

When the classification function is given as

$$y = \mathrm{sign}(\boldsymbol{w}^T \boldsymbol{x} - h), \tag{1}$$

where $\boldsymbol{x}$ stands for an input vector, $\boldsymbol{w}$ stands for a weight vector of the input, and h stands for a threshold. Function $sign(u)$ is a sign function that outputs 1 when $u > 0$ and outputs -1 when $u \leq 0$. A SVM determines a separating hyperplane with maximal margin (distance), which is the distance between the separating hyperplane and the nearest sample. If the hyperplane is determined, there exists a parameters to satisfy

$$t_i(\boldsymbol{w}^T \boldsymbol{x_i}) \geq 1, \;\; i = 1, \ldots, N. \tag{2}$$

This means that the samples are separated by two hyperplanes H1: $\boldsymbol{w}^T \boldsymbol{x}_i - h = 1$ and H2: $\boldsymbol{w}^T \boldsymbol{x}_i - h = -1$, and no samples exist between them. The distance between the separating hyperplane and these hyperplanes is defined as $1/\|\boldsymbol{w}\|$.

A soft-margin SVM allows some training samples to violate hyperplanes H1 and H2. When the distance from the H1 (or H2) is defined as $\xi_i/\|\boldsymbol{w}\|$ for the violating samples, the sum

$$\sum_{i=1}^{N} \frac{\xi_i}{\|\boldsymbol{w}\|} \tag{3}$$

should be minimized. Therefore, a soft-margin SVM is defined as an optimization problem of the following evaluation function

$$L(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{N} \xi_i \tag{4}$$

under a constraint

$$\xi_i \geq 0, \;\; t_i(\boldsymbol{w}^T \boldsymbol{x}_i - h) \geq 1 - \xi_i, \;\; (i = 1, \ldots, N), \tag{5}$$

where t_i stands for the correct class label for input vector x_i, and C stands for a **cost** parameter for violating hyperplane H1 (or H2). By solving this problem with an optimal solution $\boldsymbol{\alpha}^*$, the classification function can be redefined as

$$\begin{aligned} y &= \mathrm{sign}(\boldsymbol{w}^{*T} \boldsymbol{x} - h^*) \\ &= \mathrm{sign}(\sum_{i \in S} \alpha_i^* t_i \boldsymbol{x}_i^T \boldsymbol{x} - h^*). \end{aligned} \tag{6}$$

The samples are grouped with α_i^*; sample x_i is classified correctly when $\alpha_i^* = 0$, when $0 < \alpha_i^* < C$ sample x_i is also classified correctly, and it locates on the hyperplane H1 (or H2) as a support vector, if $\alpha_i^* = C$ sample x_i becomes a support vector but it locates between H1 and H2 with $\xi \neq 0$.

The kernel-trick, which drastically improves the performance of the SVM, can also be applied to a soft-margin SVM. In the kernel-trick, the input vectors are transformed by non-linear projection $\phi(\boldsymbol{x})$ and linearly classified in the projected space. Because the SVM depends on the product of two input vectors, the product of input vectors in projected

space can be used instead of computing the non-linear projection of the each input vector, such as

$$\phi(\boldsymbol{x}_1)^T\phi(\boldsymbol{x}_2) = K(\boldsymbol{x}_1, \boldsymbol{x}_2). \quad (7)$$

K is called a *kernel function*, and usually a simple function, a Gaussian

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(\frac{-||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2}{2\sigma^2}\right) \quad (8)$$

is selected for instance. The classification function can be redefined by replacing input vectors with kernel functions, as follows

$$\begin{aligned} y &= \text{sign}(\boldsymbol{w}^{*T}\phi(\boldsymbol{x}) - h^*) \\ &= \text{sign}(\sum_{i\in S}\alpha_i^* t_i \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}) - h^*) \\ &= \text{sign}(\sum_{i\in S}\alpha_i^* t_i K(\boldsymbol{x}_i, \boldsymbol{x}) - h^*). \end{aligned} \quad (9)$$

Two parameters, σ and C, affect the classification (and generalization) performance of the kernel-SVM. These parameters are usually determined through a grid search before the classification task is executed.

The SVM can output binary classification results such as equation (9), and a continuous probability estimation can also be output when $\sum_{i\in S}\alpha_i^* t_i K(\boldsymbol{x}_i, \boldsymbol{x}) - h^*$ is normalized between -1 to +1. Therefore we used the probability estimation for our base learners.

B. Boosting

When an ensemble classifier that consists of M base learners for a sample set $\boldsymbol{x}$ containing N samples, the boosting algorithm for two-class classification is defined by the following algorithm:

1) Initialize the observation weight $D_i = 1/N$, $i = 1, 2, \ldots, N$
2) For $m = 1$, to M:
   a) Fit a classifier $G_m(\boldsymbol{x})$ to the training samples with respect to the observation weight, D_i
   b) Compute the weighted error ratio $err_m = \frac{\sum_{i-1}^{N} D_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} D_i}$.
   c) compute a classifier weight for G_m $\alpha_m = \log((1 - err_m)/err_m)$.
   d) Update the observation weight $D_i \leftarrow D_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.
3) Output $G(x) = \text{sign}[\sum_{m=1}^{M}\alpha_m G_m(x)]$.

The current classifier, $G_m(\boldsymbol{x})$, is determined according to the observation weight at line 2a. The weighted error is computed in line 2b. Line 2c computes a classifier weight for each boosting step, and observation weights are updated in line 2d for next boosting step. The final classification result is computed in line 3, which combines the results of base learners through a weighted vote.

The classifier weights, α_m, is not exactly optimal, because boosting is based on an additive optimization model, which does not modify the classifier weights of the previous boosting steps. Therefore, we considered that optimizing the classifier weights would improve the performance of ensemble classifier. Since the collection of results of the base learners can be treated as feature vectors for the sample set, we used linear SVM to determine the optimal classifier weights.

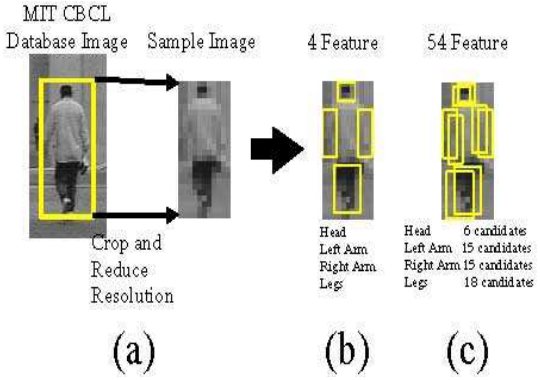

Fig. 1. Sample Images and Local Features

C. Sample Images and Local Features

We used the MIT CBCL database for the sample data, which contained 926 pedestrian images with 128×64 pixels, and we collected 2,000 random non-pedestrian images. We reduced the resolution of all the samples to 64×32 before we applied them to our system.

Figure 1 shows the original image and the locations of sub-regions in a pedestrian image. We first extracted the region for pedestrians from database images and reduced their resolution to 24×11 (figure 1a). Figure 1b shows the regions for four features, which consist of the head, left arm, right arm, and legs. Figure 1c shows the example of regions for 54 features, which include six candidate positions for the head, 15 positions for the left arm, 15 positions for the right arm, and 18 positions for the legs.

We selected 700 pedestrian and 4900 non-pedestrian images for training and 200 pedestrian and 1400 non-pedestrian images to test the generalization error.

D. Determining Base Learners

A support vector machine (SVM) was independently trained for each sub-region. While we used probability estimates for the SVM results, a 54 dimensional feature vector was determined for each sample. Each element of the feature vector contains the probability of each sub-regions as either a pedestrian or a non-pedestrian. The performance of the SVM is affected by two parameters such as soft-margin cost and Gaussian width. We determined them through a grid search tool in LIBSVM.

Our base learner is based on thresholding the results of the SVM probability estimate for local features, such as

$$G_i = \text{sign}(P_i - T), \quad (10)$$

where G_i and P_i stand for the classification output and probability estimate for the ith training sample, and T stands for the threshold. Threshold T is defined to minimize the accumulation error E, such as

$$E = \frac{1}{N}\sum_{i=1}^{N} w_i I(G_i \neq y_i), \tag{11}$$

where w_i stands for the sample weight, and y_i stands for the target value (label) for each sample. Although Adaboost usually adopts E for the training sample set, we used minimized cross-validation error, Ec, to enhance the generalization performance of the base learner.

The training sample set was split to K parts for K-fold cross-validation. For the kth ($k = 1 \ldots K$) part, the classification function was fitted for the other $K-1$ parts of the training sample set.

When $G^{-k}(x_i)$ denotes the classification results of a sample x_i, which belongs to the kth part, with the fitted function for the sample set where the kth part of the training samples is removed, the cross-validation error is defined as

$$Ec = \frac{1}{N}\sum_{i=1}^{N} w_i I(G^{-k}(x_i) \neq y_i). \tag{12}$$

E. Overall Algorithm for Proposed method

The overall algorithm for the boosting with cross-validation based feature Selection is defined as follows.

---

1) Let N be the number of samples, M be the number of boosting steps, and L be the number of sub-regions.
2) Train SVM independently for each local feature and obtain probability estimate P_l, where l from $l = 1, 2, \ldots, L$.
3) Initialize the observation weights $D_i = 1/N$, $i = 1, 2, \ldots, N$.
4) For $m = 1$, to M:
   a) l from $l = 1, 2, \ldots, L$.
      i) Minimize the cross-validation error Ec^l depending on the sample weight D_i.
      ii) fit the classifier f_m^l for the lth feature.
      iii) Determine classification result $G_m(x_i^l) = \mathrm{sign}(P_i^l - T^l)$
   b) Set err_m with the smallest cross-validation error Ec^l, $l = 1, 2, \ldots, L$.
   c) Set $G_m(x) \leftarrow G_m^l(x^l)$ with l in the above step.
   d) Compute $\alpha_m = \log((1 - err_m)/err_m)$.
   e) Set $D_i \leftarrow D_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.
5) Output $G(x) = \mathrm{sign}[\sum_{m=1}^{M} \alpha_m G_m(x)]$.

TABLE I
ERROR RATIO FOR HIERARCHICAL SVM CLASSIFIER

	Training Error (%)	Test Error (%)
Raw Data	0.43	2.44
4 Feature	0	1.25
54 Feature	0	1.88

The C (cost) and g (Gaussian width) parameters are determined in advance using a grid search tool in LIBSVM[11], and the SVM is trained independently for each local feature to determine the probability estimate for the local feature.

Every boosting step starts with selecting the SVM results for a local feature (probability estimate) with the lowest cross-validation error depending on the sample weight of the training samples, thereby the local feature L_m is selected. Then, the optimal threshold, T_m, is determined for the SVM results. The classification results for the base learner are determined by applying T_m to the selected SVM results for the local feature, thereby the classification result, G_m, and the classifier weight, α_m, for the boosting step are determined.

The final classification result is defined by weighted voting,

$$G(x) = \mathrm{sign}[\sum_{m=1}^{M} \alpha_m G_m(x)], \tag{13}$$

in boosting.

We used the linear-SVM for an overall optimization of the classifier weights. The results of the base learners $G_m m = 1, \ldots, M$ are treated as input vectors for a linear-SVM, and the optimal classifier weights are computed through the classification procedure of the SVM.

III. EXPERIMENTAL RESULTS

The experimental results are described in this section.

We first examined the single SVM for raw (intensity) data and the hierarchical SVM for four feature data and 54 feature data. Table I shows the results for pedestrian detection. The results of four local features indicate the performance of [3] for our sample set. The four feature data had an optimal classification ratio better than that of the 54 feature data implying over-fitting occurred with 54 feature data.

The reason for over-fitting in the 54 feature hierarchical SVM is that some of the local feature regions have an inadequate position to classify all the training samples. Figure 2 shows examples of this situation. The region of the left arm of four features in Figure 2a seems to be adequate to classify all the samples because it is very typical position for the left arm. However, the left arm position showed in fig. 2b, seems inadequate to classify sample i because it does not contain the left arm of sample i. However, it would be effective to classify sample ii and iii.

In the second experiment, we examined the effect of boosting with feature selection for the classification results for the local features. The best (one) local feature wss selected by a cross-validation estimate in each boosting step.

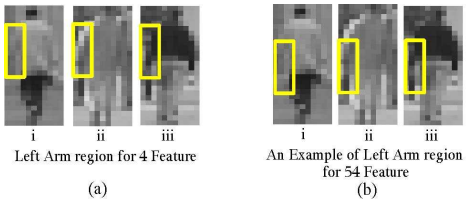

Fig. 2. Example of Local Feature

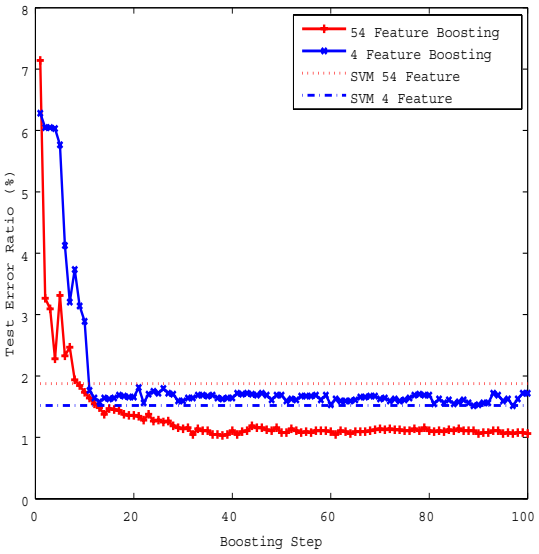

Fig. 3. Test Error Ratio for Boosting

The results were averaged through four trials. Figure 3 shows the error ratio for the test samples. Although the results of the hierarchical SVM classifier for 54 feature showed over-fitting against four feature, boosting for 54 feature showed better generalization performance than that of against four feature, and it had the lowest test-error ratio of 1.12%.

Figure 4(a) shows the classification results for the training samples while boosting. The black dots indicate mis-classification. Figure 4(b) shows weighted classification results, and figure 4(c) shows the sample weights; white dots indicate higher sample weights. Because the SVM is one of the most strongest classifiers even when soft-margin was used, our base-learners classified training samples fairly well, and most of the training samples were classified correctly throughout the boosting stages. However, some samples were frequently mis-classified, and these samples wrer considered marginal (hard to classify). The emphasized area (i, ii, iii, iv) in figure 4 shows the behavior of such samples. Mis-classification in earlier stages of boosting causes the increase in the sample weights in the later stages. The increased sample weights improve the classification of the samples.

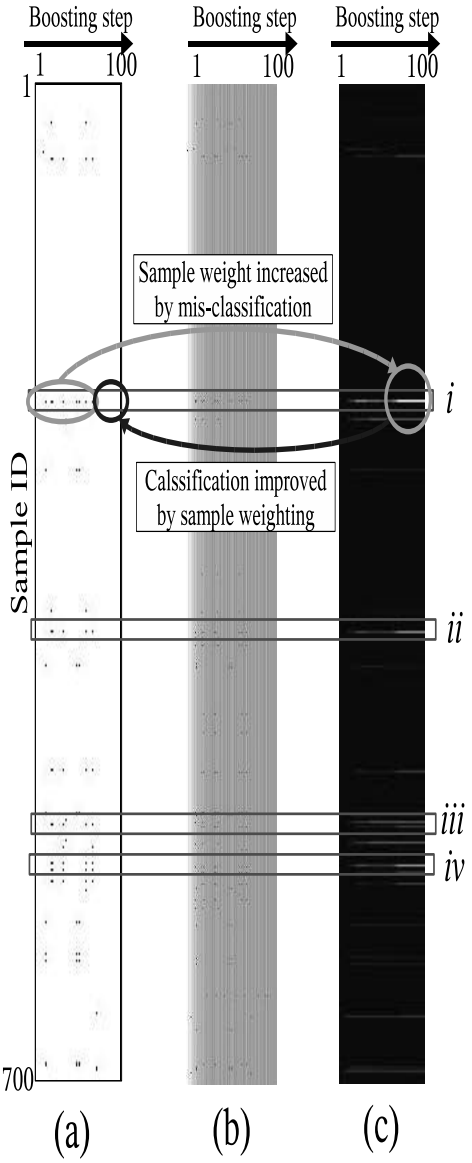

Fig. 4. Classification Error and Sample Weight during Boosting

Figure 5 is the histogram of the selected features, and table II shows the selected frequency of pedestrian components such as the head, legs, right arm, and left arm. Thirty-five regions were selected out of 54 regions, during 100 boosting steps. The most selected component was the legs, of which the selected frequency was 46, about half of the boosting steps. The location of the head and the legs should have relatively small variations because the pedestrian images were aligned in the database. However, the legs should have a wider variation in shapes. Therefore, selecting the variations of legs was effective for the classification.

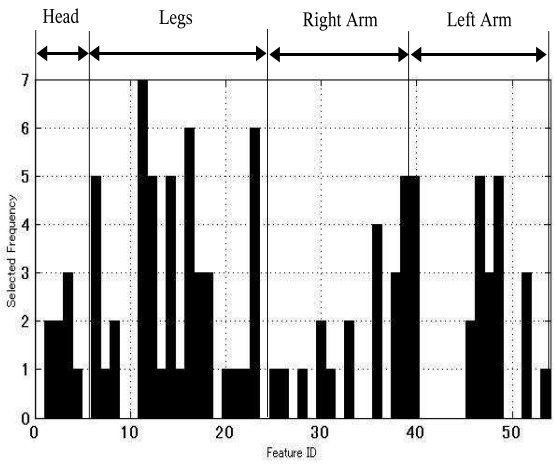

Fig. 5. Histogram for Selected Features

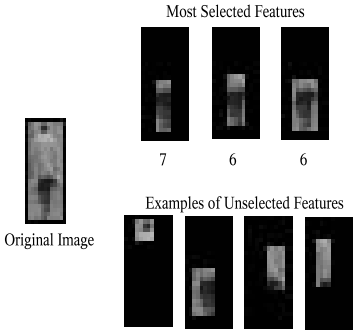

Fig. 6. Most Selected Features and Unselected Features

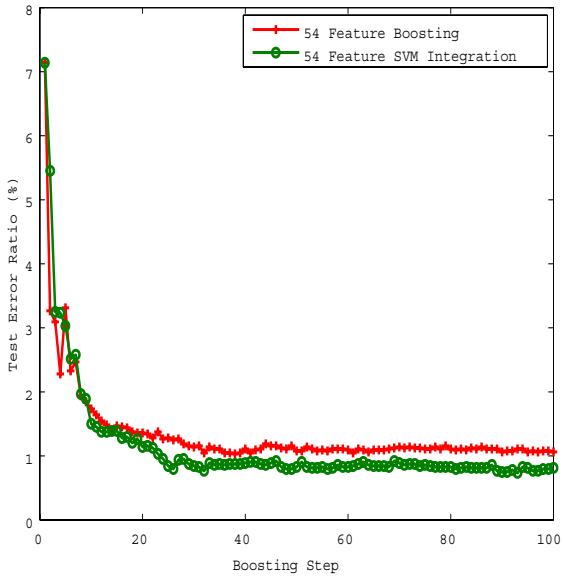

Fig. 7. Test Error for Boosting and SVM Integration

TABLE II
SELECTED FREQUENCY OF PEDESTRIAN COMPONENTS

	Selected Region	Total Region	Frequency
Head	4	6	10
Legs	15	18	46
Right Arm	9	15	20
Left Arm	7	15	24
Total	35	54	100

Figure 6 shows the examples of selected features (regions) and unselected features. The top three selected features were legs, and their locations were adequate to represent the variations. However, the locations for unselected features were inadequate to represent their components.

Boosting provides a wider variation of base learners by sample weighting. However, the classifier weights for base learners are not exactly optimal, since the weights are determined step by step. Therefore, we integrated the base learners determined in the third experiment using the linear-SVM. Figure 7 compares the test error of boosting and linear SVM integration for the base learners. The results show that the optimal classifier weights further improve the generalization performance with the lowest test-error ratio of 0.81%.

IV. CONCLUSION

We presented an object detection method that was created using on ensemble classifier with feature selection based on a cross-validation estimate. In this paper, we focused on pedestrian detection using a probability estimate of the kernel SVM for base learners, and the results of the base learners were integrated using linear-SVM and step-wise optimization through boosting. The experimental results showed that feature selection with ensemble classifiers improved the generalization performance of an object detection. We also examined the effect of optimal classifier weighting, which further improved the generalization performance over that of step-wise optimization by boosting.

We had to limit the number of sub-regions to 54 in this paper, because we had limited computational time to find the optimal parameters for base learners. We are planning to evaluate our system with a larger number of small sub-regions to improve the generalization ability of pedestrian detection.

REFERENCES

[1] Ibaraki Pref. Police Department, Japan. Statistics on Traffic Accidents (in Japanese), http://www.pref.ibaraki.jp/kenkei/kikaku/tokei/geppou/14.pdf 2006.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning - Data Mining, Inference, and Prediction*, Springer-Verlag, USA, 2001.

[3] A. Mohan, C. P. Papageorgiou, and T. Poggio, "Example-Based Object Detection in Images by Components," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 23, No. 4, pp. 349-361, 2001.

[4] D. M. Gavlira, "Pedestrian Detection from a Moving Vehicle," *Proc. of the European Conference on Computer Vision*, pp. 37-49, 2000.

[5] P. Viola, M. J. Jones, and D. Snow,"Detecting Pedestrians Using Patterns of Motion and Appearance," *Proc. of the Int'l Conf. on Computer Vision*, pp. 734-741, 2003.
[6] V.N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, USA, (1998).
[7] B. Scholkopf, C. J. C. Burges, and A.J. Smola, *Advances in Kernel Methods - Support Vector Learning*, The MIT Press, USA, 1999.
[8] N. Cristianini and J. S. Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, 2000.
[9] K. Nishida, T. Kurita, "Boosting Soft-Margin SVM with Feature Selection for Pedestrian Detection," *Proc. of the 6th International Workshop on Multiple Classifier Systems MCS 2005*, pp. 22-31, 2005.
[10] H. Schwenk and Y. Benjio, "Boosting Neural Networks," *Neural Computation*, Vol. 12, pp. 1869-1887, 2000.
[11] C. C. Chung and C. J. Lin, "LIBSVM: a library for support vector machines," *Software available at* `http://www.csie.ntu.edu.tw/`, 2001.
[12] R. E. Shapire, "The Boosting Approach to Machine Learning: An Overview," *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.
[13] J.Lu, K.N. Plataniotis, A.N. Venetsanopoulos, and S.Z. li, "Ensemble-based Discriminant Learning with Boosting for Face Recognition," IEE Trans. on NN, Vol. 17, No. 1, pp. 166-178, 2006.
[14] L. Breiman, "Random Forests," *Machine Learning*, Vol. 45, pp. 5-32, 2001.
[15] L. Breiman, "Prediction Games Arcing Algorithms," Tech. Report 504, UC Berkley, 1997.
[16] T.K. Ho, "Random Subspace Method for Constructing Decision Trees," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 8, pp. 832-844, 1998.