Proceedings of the 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems

Adaptive Object Detection From Multisensor Data

Yong-Jian Zheng and Bir Bhanu
College of Engineering
University of California
Riverside, CA 92521-0425

Abstract

This paper focuses on developing self-adapting automatic object detection systems to achieve robust performance. Two general methodologies for performance improvement are first introduced. They are based on optimization of parameters of an algorithm and adaptation of the input to an algorithm. Different modified Hebbian learning rules are used to build adaptive feature extractors which transform the input data into a desired form for a given object detection algorithm. To show its feasibility, input adaptors for object detection are designed and tested using multisensor data including SAR, FLIR, and color images. Test results are presented and discussed in the paper.

1 Introduction

Automatic object detection is of great importance for many vision based real-world applications. An automatic object detection system should be able to locate objects of interest in the input images produced by different sensors such as CCD cameras, infrared sensors, radars and multispectral scanners. Although many automatic object detection systems have been developed, their performance is still limited [8]. This paper is motivated by the increased demand for new theories and methodologies to improve system performance [3, 8, 9] and to minimize the effort needed for the development of robust object detection systems. The original contribution of this paper is the idea that the performance of a given algorithm can be improved by adding an adaptor between the input data and the algorithm. This is an input adaptive process and is based on the observation that most algorithms would perform well if the desired input data can be provided to them. Different kinds of Hebbian-like learning rules are introduced and applied to developing such adaptors. The feasibility of this methodology for performance improvement is demonstrated by experimental results using multisensor data.

*This work was supported by ARPA, AFOSR grants F49620-93-1-0624, F49620-95-1-0424 and MDA972-93-1-0010. The contents of the information do not necessarily reflect the position or the policy of the U.S. Government.

2 Parameter Optimization Versus Input Adaptation for Performance Improvements

The first methodology is based on the consideration that some algorithms and systems have certain controllability and their performance can be improved by tuning their parameters [1]. To find the best parameter set for the given input data a learning and optimizing process is usually required. This methodology is, therefore, parameter optimizing oriented. As shown in Figure 1, parameter optimizing based methodology employs different parameter set for different input data in order to obtain the optimal output. However, this methodology suffers from some inherent shortcomings: (a) It is driven by both the input data and the output data. It has to have an off-line learning phase. This leads to the difficulty of sample collection because some input situations are not predictable. Besides, the off-line learning process is usually time consuming. (b) In order to use the trained algorithm, information about the possible category of the input data is needed before the appropriate parameter set can be switched on. This means that the trained algorithm works only with an additional input identifier which triggers the corresponding parameter set. Certainly the design of such an identifier is as hard as that of the algorithm itself. (c) The performance of an algorithm cannot be always improved by optimizing the parameter set because the gradients of objective functions of some algorithms with respect to their parameters are too small. So not all algorithms can be improved by using this methodology.

The second methodology for performance improvement is based on the observation that most algorithms would perform well if their input data are "friendly", as discussed above. Thus, the performance of almost all commonly used algorithms can be improved by adding an adaptor between the input data and the algorithm (see Figure 2). An ideal adaptor should automatically judge the input data, provide the desired input data to an algorithm, and learn something from this process in order to improve itself in the future.

In comparison with the parameter optimizing based methodology, the input adapting methodology presented

0-7803-3700-X/96 $5.00

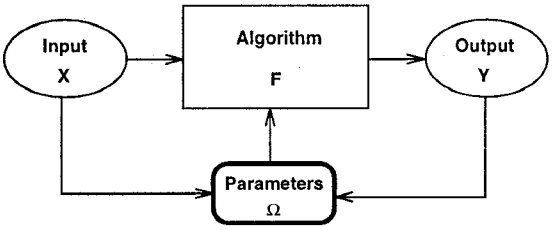

Figure 1: Parameter optimizing methodology for performance improvement.

in this papaer has some positive features such as: (a) It is suitable for almost all algorithms because the desired input data (not always the perfect input data) always exists for a given algorithm. (b) It is driven only by the input data. So it can work both on-line and off-line. This is very important for real-world and real-time applications. (c) This methodology makes it possible to combine some simple, ready-made available algorithms to build vision systems that exhibit high level performance. Without adding adaptors, these simple algorithms may be unreliable for practical applications, although they have simple structures and are not time consuming.

3 Representations Versus Salient Features

Most commonly used algorithms can show good performance only if their input representations have some "friendly" characteristics. To keep their performance high even if the input representations are not so "friendly", adaptors are needed which transform the input representations to some salient features. Thus, an adaptor can also be regarded as a salient feature extractor. The key issue in input adapting methodology for performance improvement is how to design an adaptor or feature extractor for each algorithm at each stage of the representation transformation.

3.1 Optimal Feature Extraction

From a mathematical viewpoint, feature extraction is a transformation from a m-dimensional input representation $\mathbf{x}$ to a n-dimensional output representation $\mathbf{v}$, so that $n \leq m$ and for each $v \in \mathbf{v}$ the expected value of $\rho(v)$ is minimized:

$$E(\rho(v)) = \int_{-\infty}^{+\infty} \rho(v)p(v)dv \rightarrow \min, \qquad (1)$$

where $\rho(\cdot)$ is a "loss" function, $E(\cdot)$ is the risk (the expected value of the loss), and $p(\cdot)$ is the probability density function of v. This means that the transformed representation $\mathbf{v}$ should be less redundant (because of

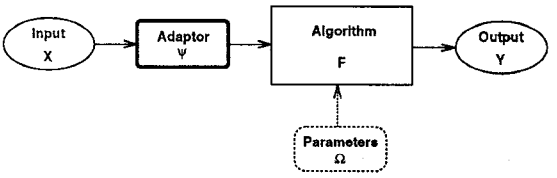

Figure 2: Input adapting methodology for performance improvement.

$n \leq m$) and salient (because of $E(\rho(v)) \rightarrow \min, v \in \mathbf{v}$). Thus $E(\cdot)$ is a measure of saliency which depends on the loss function $\rho(\cdot)$.

A simple example of the representation transformation is the linear mapping $\mathbf{W}$ which transforms the m-dimensional input representation $\mathbf{x}$ to n-dimensional output representation $\mathbf{v}$ by using

$$\mathbf{v} = \mathbf{W}\mathbf{x} = (\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m)^\top \mathbf{x}. \qquad (2)$$

In this case, $\mathbf{W}$ is a feature extractor if $\mathbf{v}$ has some nice properties. The feature extractor $\mathbf{W}$ can be realized by using a single-layer linear feed-forward network.

As can be seen, the basic unit of this network is a m to 1 mapping

$$v = \mathbf{w}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{w}, \quad v \in \mathbf{v}. \qquad (3)$$

The basic unit can also be nonlinear. In this case the m to 1 mapping is formulated by

$$v = \tau(\mathbf{w}^\top \mathbf{x}) = \tau(\mathbf{x}^\top \mathbf{w}), \quad v \in \mathbf{v}, \qquad (4)$$

where $\tau(\cdot)$ is nonlinear function. The mapping (3) or (4) is salient or interesting if $E(\rho(v))$ is minimized. The key issue of constructing a feature extractor is thus the design of the loss function.

Before the loss function can be designed, the question of which v is "salient" or "interesting" should be first defined. Certainly, no universal agreement on this question can be expected. Two general definitions about the saliency of v that we have are:

- **Expressiveness**: v is salient if it is *expressive.*
- **Discrimination**: v is salient if it is *discriminating.*

In the following the problem of how to extract these features is addressed.

3.2 Expressive Feature

Let us consider a set of m-dimensional input representations $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\}$ which builds a "cloud" of points in the m-dimensional space. It is clear that each point $\mathbf{x} \in \mathcal{X}$ can be projected onto a direction determined by the vector $\mathbf{w}$ by using Equation (3) or (4) and the result of this projection is v. Figure 3 just shows

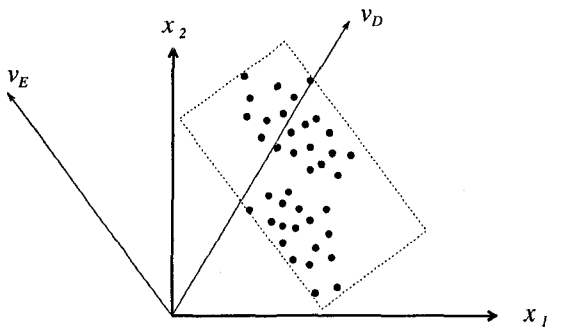

Figure 3: A point cloud in a 2-dimensional space.

a case of $m = 2$. Now the problem is which projection direction is interesting.

As shown in Figure 3 the first interesting direction is v_E because the projections of all points onto this direction have the maximal variance and v_E is, therefore, *expressive*. It can be proved that v_E is determined by that $\mathbf{w}$ which is the largest eigenvector associated with the largest eigenvalue of the correlation matrix

$$\mathbf{Q} = E(\mathbf{x}\mathbf{x}^\top). \tag{5}$$

Let us first define a loss function

$$\rho_G = \frac{1}{2}v^2 \tag{6}$$

The risk $E(\rho_G)$ can be calculated by

$$E(\rho_G) = \frac{1}{2}E(v^2) = \frac{1}{2}\mathbf{w}^\top\mathbf{Q}\mathbf{w}. \tag{7}$$

Minimizing $E(\rho_G)$ requires

$$\Delta\mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \mathbf{Q}\mathbf{w} = \mathbf{0}. \tag{8}$$

This leads to famous *plain Hebbian learning* rule

$$\Delta w_i = \eta v x_i, \tag{9}$$

where w_i and x_i are the i^{th} component of $\mathbf{w}$ and $\mathbf{x}$ respectively, and η controls the learning rate as usual. It can be seen that Hebbian learning is controlled by both the input (through x_i) and the output (through v).

Equation (8) tells that $\mathbf{w}$ is an eigenvector of $\mathbf{Q}$ with eigenvalue 0. But this could never be stable, because $\mathbf{Q}$ necessarily has some positive eigenvalues; any fluctuation having a component along an eigenvector with positive eigenvalue would grow exponentially. It might be suspected that the direction with the largest eigenvalue of $\mathbf{Q}$ would eventually become dominant, so that $\mathbf{w}$ would gradually approach the eigenvector corresponding to the largest eigenvalue with a increasingly huge norm. Certainly, $\mathbf{w}$ does not settle down in any case. There are only unstable fixed points for plain Hebbian learning procedure (9).

Let us modify the loss function (6) to

$$\rho_E = \frac{1}{2}\left[v^2 - E(v^2)\mathbf{w}^\top\mathbf{w}\right]. \tag{10}$$

The risk $E(\rho_E)$ can be calculated by

$$\begin{aligned} E(\rho_E) &= \frac{1}{2}(E(v^2) - E(v^2)\mathbf{w}^\top\mathbf{w}) \\ &= \frac{1}{2}(\mathbf{w}^\top\mathbf{Q}\mathbf{w} - E(v^2)\mathbf{w}^\top\mathbf{w}). \end{aligned} \tag{11}$$

Since the risk is continuously differentiable, the optimization of (11) can be achieved, via a gradient descent method, with respect to $\mathbf{w}$:

$$\Delta\mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \mathbf{Q}\mathbf{w} - E(v^2)\mathbf{w} = \mathbf{0}. \tag{12}$$

Clearly, an equilibrium can be reached if $\mathbf{w}$ is the eigenvector associated with one eigenvalue, say the largest one, of $\mathbf{Q}$ and $E(v^2)$ is just the eigenvalue.

Equation (12) leads to the learning rule suggested by Oja [6]. According to this rule, each input $\mathbf{x} \in \mathcal{X}$ is applied to adapt the weight $\mathbf{w}$ by using

$$\Delta w_i = \eta v(x_i - v w_i), \tag{13}$$

where w_i and x_i are the i^{th} component of $\mathbf{w}$ and $\mathbf{x}$ respectively, and η controls the learning rate. The learning rule (13) is a generalized version of Hebbian learning rule (9) which has been so widely applied to develop unsupervised learning networks.

Comparing the loss function (10) with (6) shows that these Hebbian-like learning rules are based on second order statistics as they usually use second order polynomials for measuring the interest and they lead to extraction of *principal components* (PC) of the input data [5, 6, 7]. It can be proved [5] that minimizing the risk (11) is equivalent to maximizing the information content of the output representation in situations where that has a Gaussian distribution.

The direction v_E found in this way allows faithful representation of the input data and the projection of the point cloud onto this direction can also show interesting structure if the cloud contains a few clusters and the separation between clusters is larger than the internal scatter of the clusters. However, the direction v_E can lead us astray if the cloud shows too many isotropically distributed clusters or if there are meaningless variables (x_i's) with a high noise level. In these two cases, the output representation v_E doesn't allow discrimination between clusters (see the example in Figure 3). This means that second order polynomials are not sufficient to characterize the important features of an input distribution and all second order statistics based feature extractors cannot provide features which are discriminating enough for recognizing the structure in the input representation.

3.3 Discriminating Feature

As shown in Figure 3, the second interesting direction is v_D because the projections of all points onto this direction can enable us to better distinguish the interesting structure (clusters) presented in the cloud and v_D is, therefore, *discriminating*.

In order to find this direction, a measure sensitive to distributions which are far from Gaussian is needed. As already discussed, second order polynomials such as shown in (10) cannot be used for measuring deviation from normality. To emphasize bi- or multi-modality of the projected distribution, higher order polynomials are required and care should be taken to avoid their oversensitivity to small number of outliers.

Let us define a loss function

$$\rho_D = v^2 \left[\frac{E(v^2)}{4} - \frac{|v|}{3} \right] = v^2 r(v), \tag{14}$$

where $r(v)$ can be regarded as a weighting function. The loss function ρ_D is small if v is close to zero or to $3E(v^2)/4$. Moreover, it remains negative for $v > 3E(v^2)/4$. Thus, ρ_D as an index can exhibit the fact that bimodal distribution is already interesting, and any additional mode should make the distribution even more interesting.

Actually, any radial basis function (see [9]) can be used as the weighting function in Equation (14) to design interesting loss functions. The advantage of $r(v)$ used in Equation (14) is its connection to the Bienenstock, Cooper, and Munro (BCM) theory of visual cortical plasticity [2, 4].

The expected value of ρ_D is given by:

$$\begin{aligned} E(\rho_D) &= \frac{1}{4}E^2(v^2) - \frac{1}{3}E(v^3) \\ &= \frac{1}{4}E(v^2)\mathbf{w}^\top \mathbf{Q}\mathbf{w} - \frac{1}{3}E(v^3). \end{aligned} \tag{15}$$

To achieve $E(\rho_D) \rightarrow \min$, the equation

$$\Delta \mathbf{w} = \frac{\partial E}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \left[\frac{1}{4}E^2(v^2) - \frac{1}{3}E(v^3) \right] = \mathbf{0} \tag{16}$$

should be satisfied. This leads to a learning rule

$$\Delta w_i = \eta \left[v^2 - E(v^2)v \right] x_i, \tag{17}$$

where x_i is the i^{th} component of $\mathbf{x}$ and η controls the learning rate.

The difference between the learning rule (17) and Hebbian learning rule (9) is the fact that the influence of the output on the learning process (the feedback) has been changed from v in (9) to $v^2 - E(v^2)v$ in (17). This enables the learning rule (17) to discover bimodal distributions as Δw_i in (17) (unlike in (9)) has opposite value depending on if v is larger or smaller than $E(v^2)$.

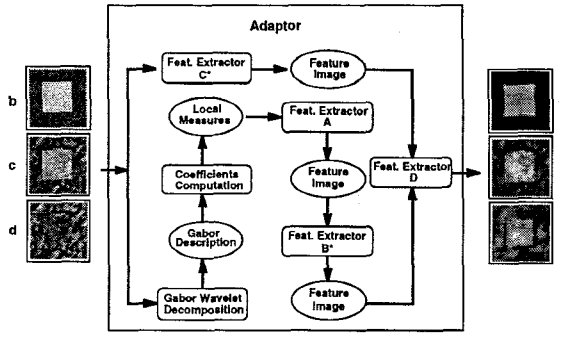

Figure 4: Input adapting for the image thresholding algorithm.

Unfortunately, the learning rule (17) has the same divergence problem like Hebbian learning rule (9) and in any case $\mathbf{w}$ does not settle down. One way to prevent the divergence of $\mathbf{w}$ is to constrain the growth of $\mathbf{w}$ by modifying the loss function (14):

$$\rho_Z = v^2 \left[\frac{E(v^2)}{4} - \frac{|v|}{3} \right] - E(v^2)\mathbf{w}^\top \mathbf{w}. \tag{18}$$

This leads to a new learning rule obtained by adding a weight decay to the learning rule (17):

$$\Delta w_i = \eta \left[v^2 - E(v^2)v \right] (x_i - v w_i). \tag{19}$$

So far four different learning rules have been introduced. They are based on four different loss functions (see Table 1) and can be applied to extracting expressive and discriminating features. In Table 1, EF denotes "expressive features" and DF denotes "discriminating features".

4 Adaptive Object Detection

4.1 Adaptor Design

Figure 4 shows an adaptor which is designed for the image thresholding algorithm. The key idea for designing this adaptor is to decompose the input image into some local measure images and then to adaptively extract salient features from these local measure images based on the modified Hebbian learning rules presented above. In order to derive local measures for each pixel in the input image, the quadrature Gabor filter kernels

$$\begin{aligned} G_+(\omega, \phi) = \exp &\left[-\frac{\lambda^2 \omega^2 (x^2 + y^2)}{4\pi} \right] \\ &\cdot \cos[\omega(x\cos\phi + y\sin\phi)] \end{aligned} \tag{20}$$

$$\begin{aligned} G_-(\omega, \phi) = \exp &\left[-\frac{\lambda^2 \omega^2 (x^2 + y^2)}{4\pi} \right] \\ &\cdot \sin[\omega(x\cos\phi + y\sin\phi)] \end{aligned} \tag{21}$$

Table 1: Different Learning Rules For Feature Extraction

Type	The Loss Function	The Learning Rule	Suitability
I	$\rho_G = \frac{1}{2}v^2$	$\Delta w_i = \eta v x_i$	EF
II	$\rho_E = \frac{1}{2}\left[v^2 - E(v^2)\mathbf{w}^\top\mathbf{w}\right]$	$\Delta w_i = \eta v(x_i - v w_i)$	EF
III	$\rho_D = v^2\left[\frac{E(v^2)}{4} - \frac{\lvert v\rvert}{3}\right]$	$\Delta w_i = \eta\left[v^2 - E(v^2)v\right]x_i$	DF
IV	$\rho_Z = v^2\left[\frac{E(v^2)}{4} - \frac{\lvert v\rvert}{3}\right] - E(v^2)\mathbf{w}^\top\mathbf{w}$	$\Delta w_i = \eta\left[v^2 - E(v^2)v\right](x_i - v w_i)$	DF

are applied to decompose the input image $I(x, y)$ by using

$$I_+(x, y, \omega, \phi) = G_+(\omega, \phi) * I(x, y),$$
$$I_-(x, y, \omega, \phi) = G_-(\omega, \phi) * I(x, y), \qquad (22)$$

where ω and ϕ are the modulation (center) frequency and orientation, respectively, of the Gabor filter kernel; λ is the ratio of the channel bandwidth and the modulation frequency; and $I_+(x, y, \omega, \phi)$ and $I_-(x, y, \omega, \phi)$ are Gabor space image descriptions. From these descriptions it is easy to derive some local measures. In Figure 4 the power

$$p(x, y, \omega, \phi) = I_+^2(x, y, \omega, \phi) + I_-^2(x, y, \omega, \phi) \qquad (23)$$

is used as a local measure of the input image $I(x, y)$. So far m power images can be obtained and m depends on the quantization of ω and ϕ. This means a local measure vector with m elements is associated with each pixel of the input image.

Each element of the local measure vector is a representation to describe the local property of the input image but it is not just the right feature to discriminate clusters depicted in the input image. The most discriminating feature should be found in the m-dimensional local measure space based on the structure presented by all local measure vectors in the input image. This requires a m to 1 feature extractor **A** which is trained by using the learning rule (17) or (19) as described above.

To reduce high frequency components in the input data two feature extractors **B**$*$ and **C**$*$ are introduced into the adaptor shown in Figure 4. They are actually two convolution kernels with $n \times n$ elements which should be trained by using the learning rule (9) or (13) as described above.

After the convolution using **B**$*$ and **C**$*$ two feature images can be produced which should be integrated by the feature extractor **D** in order to supply desired images for the thresholding algorithm. The feature extractor **D** performs a 2 to 1 transformation and is trained by using the learning rule (17) or (19).

Now the adaptor is able to produce desired input images for the thresholding algorithm (see Figure 4). It is obvious that the output images in Figure 4 are better than the input images in Figure 4 to be used as the input images for the thresholding algorithm because the object in the center is better discriminated from the background. Thus, the performance of the thresholding algorithm can be improved by using the adaptor.

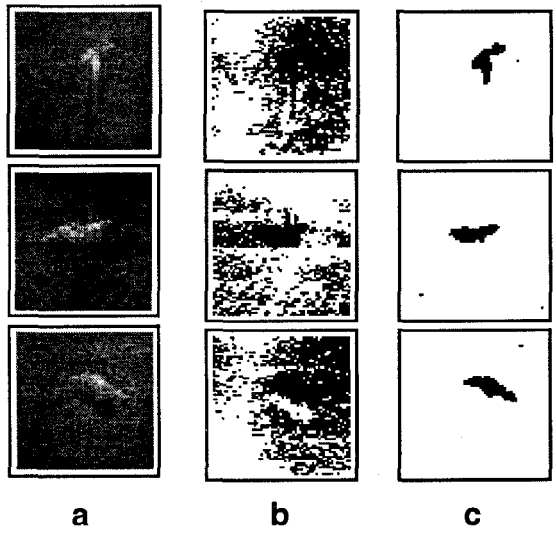

Figure 5: Test Result Using SAR Images.

4.2 Experimental Result for Adaptive Target Detection

Figure 5 shows the test result of target detection system using SAR image data. The column **a** shows the input images. The column **b** shows the test results using the thresholding algorithm. The column **c** shows the test results using the thresholding algorithm plus the input adaptor. It can be seen that even a simple algorithm can perform well if its input data are properly prepared by an input adaptor. This means that adding an input adaptor can enlarge the dynamic range of an algorithm and improve its performance.

Figure 6 shows another example of target detection in a FLIR (Forward Looking Infrared) image by using the same system. Again, the image **a** is the input image. The

image **b** and **c** show the test results using the thresholding algorithm without and with the input adaptor. As can be seen, the performance of the system is satisfied even when the input image has different properties as used before.

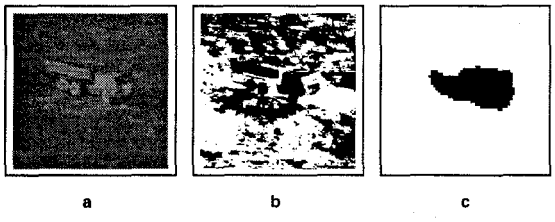

Figure 6: Test Result Using a FLIR Image.

4.3 Detection of Colored Objects

The first step to design a system for object detection from a color image is the specification of an object. An object, for instance, can be defined as a connected region in a color image which has a special shape such as circle or rectangle. It can also be defined as a connected region which has a given color topology such as a red region surrounded by a yellow region. In this paper, an object in a color image is defined as a connected region which is small and well colored. Figure 7 shows a sample image. The scene is photographed approximately every 15 minutes over a four hour period by using a fixed JVC GXF700U color video camera. A total of 20 image frames are obtained in this way and only four of them are selected for the experiment. The time and the weather condition of these four color images are: Frame 1, 1:20pm, Sunny; Frame 5, 2:15pm, Sunny; Frame 9, 3:15pm, Sunny; and Frame 13, 4:45pm, Sunny.

The colors of the car and traffic sign in Frame 13 are subdued since they are located under the shadow of the trees when Frame 13 was taken. However, these objects are well colored in Frame 1, because there was no shadow at 1:20 pm when Frame 1 was taken. This can be seen if all the pixels of both Frames are mapped into the RGB color space (see Figure 8). The R, G, and B values of all pixels in Figure 8 are normalized in the range $[-0.5, 0.5]$. As shown in Figure 8, all pixels in Frame 13 are located along the line segment between the point $[-0.5, -0.5, -0.5]$ and the point $[0.5, 0.5, 0.5]$. This line can be thought of as a colorless line. This means that the saturation or the color of all pixels in Frame 13 is relatively low because they are located close to the colorless line.

On the contrary, some pixels in Frame 1 are located away from the colorless line and the saturation of these pixels is relatively good. These pixels are well separated from the pixel group around the colorless line and can be regarded as outliers of this pixel group. Thus, they can

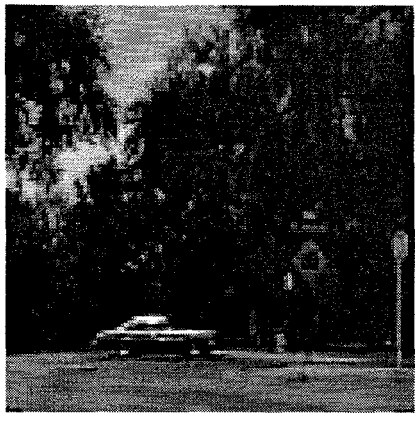

Figure 7: An outdoor scene with a car and a yellow traffic sign near the car.

Table 2: Weights after training by using the learning rule II

Image Used for Training	w_1	w_2	w_3
1	0.576884	0.592581	0.562207
5	0.585867	0.590048	0.555533
9	0.583240	0.567947	0.580755
13	0.585854	0.577843	0.568225

be defined as well colored pixels which build regions of interest in Frame 1. In the following we describe how to develop an adaptive system to find these outlier pixels.

It is first interesting to know what happens if all pixels in a color image are applied to train the 3 to 1 feedforward network by using Hebbian-like learning rules shown in Table 1. The weights of this 3 to 1 feedforward network obtained after training by using the learning rule II and III in Table 1 are shown in Table 2 and Table 3, respectively. The trained network can then be used to map a color image into a gray scale image. Figure 9 shows the gray scale images mapped for Frame 1 and Frame 13. The first row in Figure 9 shows the images mapped by using the weights listed in Table 2, while the second row in Figure 9 shows the images mapped by using the weights listed in Table 3.

If the two mapped images shown in the second row of Figure 9 are compared, we can see that the car and the traffic sign are much better separable from the background in Frame 1 than in Frame 13. This means that, although Frame 13 has three color channels R, G and B, the color information encoded in this frame is so weak that this frame can be regarded as almost colorless. In

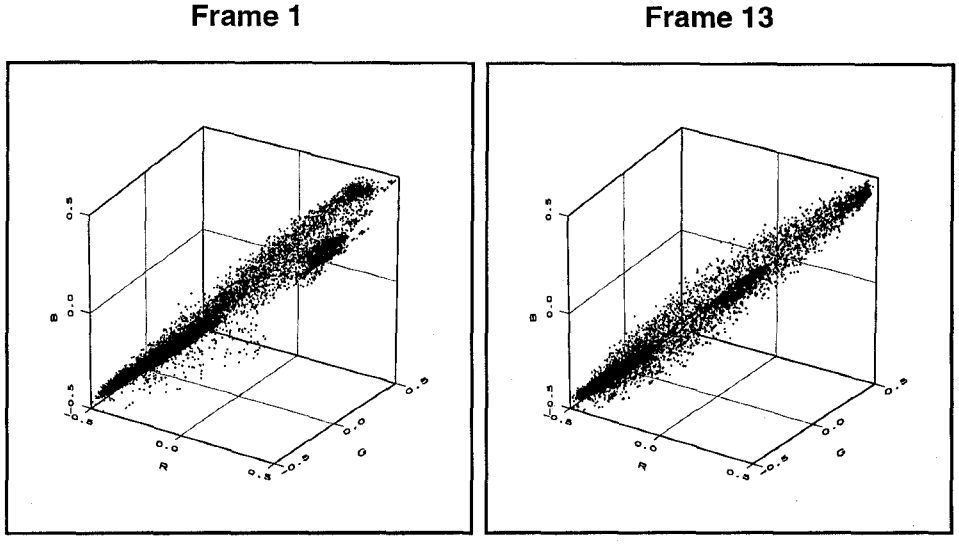

Figure 8: Image pixels are mapped into the RGB color space.

Table 3: Weights after training by using the learning rule III

Image Used for Training	w_1	w_2	w_3
1	0.691818	-0.716642	0.088382
5	0.685157	-0.724709	0.073194
9	0.727544	-0.685147	-0.035408
13	0.657560	-0.749287	0.078640

fact, most of the information in Frame 13 is encoded in a gray scale image which is the right image in the first row of Figure 9, because it was obtained by using the most expressive mapping and this mapping was determined by using all the pixels in Frame 13 as the training data and the modified Hebbian learning rule II in Table 1.

The well colored objects such as the car and the traffic sign in this image (the left image of second row shown in Figure 9) are separable from the background. To understand this, all selected frames are transformed by using such mapping (see the first row of Figure 10) and compared with their saturation (see the second row of Figure 10). It is clear that the adaptive mapping obtained by using the modified Hebbian learning rule III or IV in Table 1 discriminates well-colored objects from the background. The left image of Figure 11 shows the object detection result from Frame 1 after thresholding the results shown in the top left image of Figure 10. This image can be further used for post-processing. The right

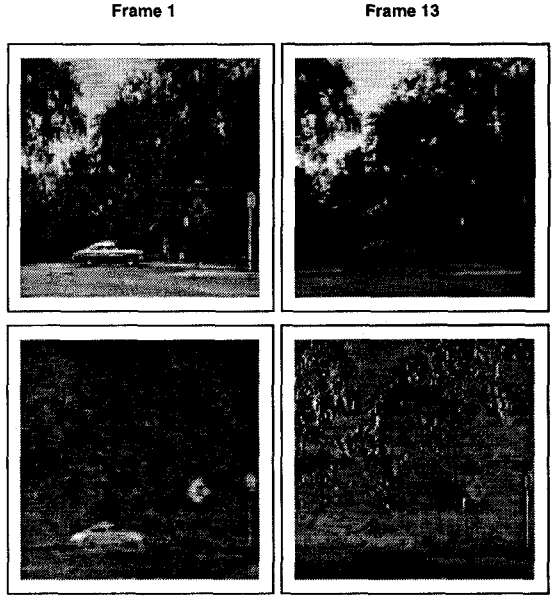

Figure 9: Grey scale images obtained by adaptive mapping.

image of Figure 11 shows the post-processing result using morphological filtering followed by color based filtering.

5 Conclusions

In this paper, the attention was paid on how to to improve the performance of object detection systems by adding the adaptability to ready-made available algorithms without changing their internal structure. The

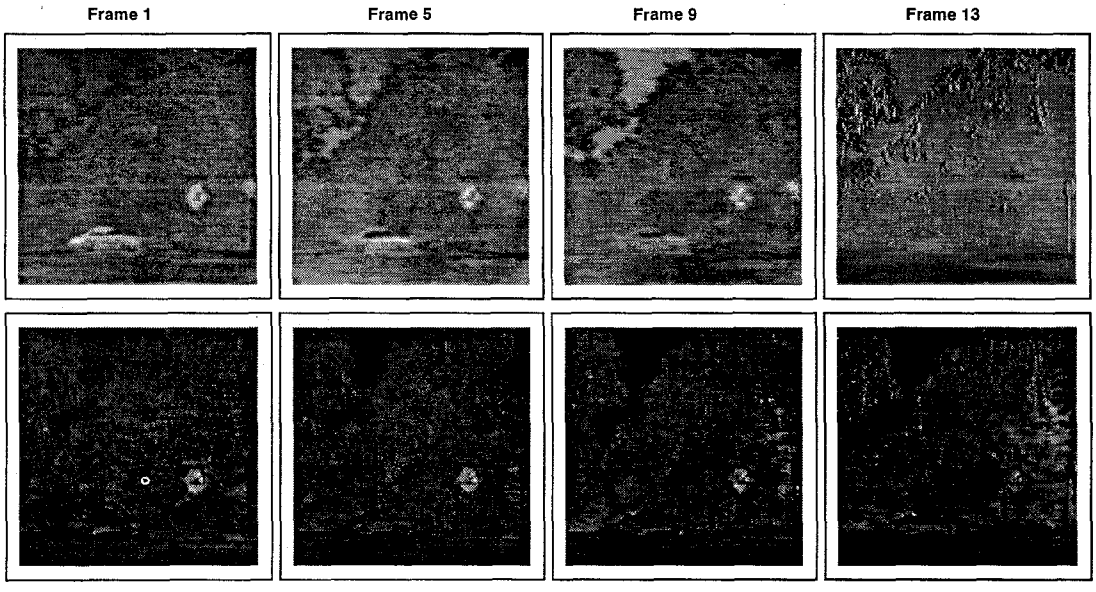

Figure 10: The discriminating mapping used to color images and compared with the saturation mapping

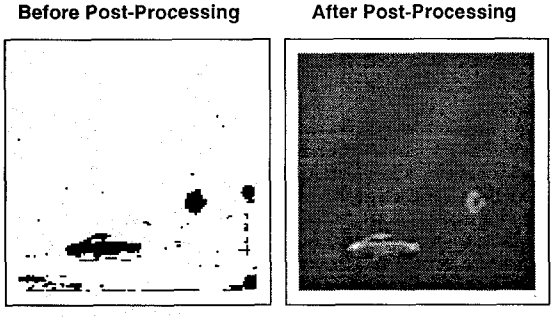

Figure 11: The object detection from Frame 1, before and after post-processing

input adapting based approach presented here provides a promising solution to improve the performance of pattern recognition and computer vision algorithms and systems to meet requirements of real-world applications.

References

[1] B. Bhanu and S. Lee. *Genetic Learning for Adaptive Image Segmentation.* Kluwer Academic Publishers, 1994.

[2] E. L. Bienenstock, L. N. Cooper, and P. W. Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal Neuroscience*, 2:32–48, 1982.

[3] R. M. Haralick. Performance characterization protocol in computer vision. In *Proc. Performance Versus Methodology in Computer Vision*, pages 26–32, Seattle, WA, June 1994.

[4] N. Intrator and L. N. Cooper. Objective function formulation of the BCM theory of visual cortical plasticity: Statistical connections, stability conditions. *Neural Networks*, 5:3–17, 1992.

[5] R. Linsker. Self-organisation in a perceptual network. *Computer*, 21(3):105–117, 1988.

[6] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[7] T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Network*, 2:459–473, 1989.

[8] Y.-J. Zheng. Feature extraction and image segmentation using self-organization networks. *Machine Vision and Applications*, 8(5):262–274, 1995.

[9] Y.-J. Zheng, W. Ritter, and R. Janssen. An adaptive system for traffic sign recognition. In *Proc. Intelligent Vehicles Symposium*, pages 165–170, October 1994.