Radar-Vision Fusion for Object Classification

Zhengping Ji and Danil Prokhorov
Technical Research Department
Toyota Technical Center - TEMA
Ann Arbor, MI 48105 USA
dvprokhorov@gmail.com

Abstract— **We propose an object classification system that incorporates information from a video camera and an automotive radar. The system implements three processes. The first process is attention selection, in which the radar guides a selection of a small number of candidate images for analysis by the camera and our learning method. In the second process, normalized attention windows are processed by orientation-selective feature detectors, generating a sparse representation for each window. In the final process, a Multilayer In-place Learning Network is used to distinguish sparse representations of different objects. Though it is more flexible in terms of variety of classification tasks, the system currently demonstrates its high accuracy in comparison with others on real-world data of a two-class recognition problem.**

Keywords: automotive, radar, camera, attention selection, sparse representation, MILN.

I. Introduction

Future automotive vehicles will have many more sensors and systems dedicated to various driver support, semi- and fully-autonomous functions. As one type of active sensors, the radar system has shown a reasonable performance of target detection in relatively simple environments (e.g., highways). It provides reasonably accurate measurements of object distance and velocity in various weather conditions. However, the vehicle radars have insufficient resolution for extraction of object features. Video cameras, called passive sensors, do provide sufficient (lateral) resolution in a suitable range of distances. The cues of object shapes and appearances may provide sufficient characteristics for classification of different objects. Considering the complementary properties of the two sensors, we can combine them in a single system for improved performance.

The fusion of data from radar and vision has been widely discussed for driver-assistance tasks. Coue et. al [2] described a Bayesian framework for fusion based on target information (position and range) from multiple modalities. In Jochem and Langer's work [6], a system is described where a purely radar-based obstacle detection and tracking is performed. The optical lane recognition is used to assign vehicles to different lanes. Grover et al. [3] performed fusion of low-level features, such as blobs, detected in both radar and vision data. With a single radar map and a single night-vision image, the fusion is performed in polar coordinates and based on angular position. In [7], an approach is proposed to perform fusion at the target level. By analyzing image samples of the locations given by the azimuth values from the radar, pattern recognition techniques were used for detecting vehicles or pedestrians. Shunji [8] et al. presented a range-window algorithm augmented by correlation-based pattern matching for detection and tracking of objects up to 50 m in front of the vehicle.

Instead of generic object classification architecture, most of work is dedicated to detection of specifically vehicles or pedestrians. In the proposed system, we took the advantage of radar-vision integration to achieve a reasonable attention selection on candidate targets and employ a general-purpose classification and regression network, called Multi-layer In-place Learning Network (MILN), originally proposed by J. Weng et al. [5]. We transformed the object representation from the original appearance space to the sparse coded space, using the best set of features derived from images of nature called natural images. Since natural images are very different and completely independent from images captured during the typical system operation, our proposed system is extendable to different driving environments and different classification tasks. By using MILN, the learning of coded representation can be implemented both prior to the in-vehicle deployment of the system and in real time. Its in-place learning mechanism holds comparatively low operational complexity even for large networks.

In what follows, we first describe the problem statement and present the operation architecture of our system. In Sec. IV, the learning algorithms is discussed. The experimental results and conclusions are reported in Sec. V and Sec. VI, respectively.

II. Problem statement

Our goal is to recognize objects in the path of a properly equipped, moving vehicle. The system should be designed to separate objects into two classes: "vehicles" and "non-vehicles".

The operation architecture of the proposed system is shown in Fig.1. Two external (outward looking) sensors are employed: a video camera and a radar. Table I & II specifies sensor parameters for our system.

A. Radar and camera integration

As shown in Fig. 2 (right), a group of target points in 3D world coordinates can be detected from the radar, which scans in the horizontal field of 15°, with the detection range up to 150 meters. A red/green bar is associated with each radar return, and its length and direction indicate the relative speed of each target object.

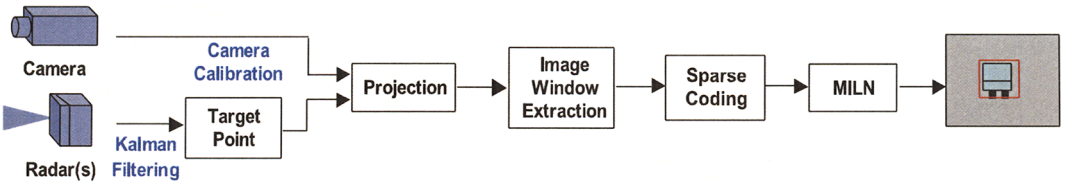

Fig. 1. Operation architecture of the proposed system.

TABLE I
SENSOR SPECIFICATIONS OF RADAR SYSTEM – F10 MM-WAVE RADAR

Key Parameters	Specification
Refreshing Rate	10 Hz
No. of targets	max. of 20 targets
Distance	$2 \sim 150\text{m} \pm \mathbf{max}(5\%, 1.0\text{m})$
Angle	$15^\circ \pm \mathbf{max}(0.3^\circ, \text{range of } 0.1\text{m})$
Speed	$\pm 56\text{m/s} \pm 0.75\text{m/s}$

TABLE II
SENSOR SPECIFICATIONS OF VISION SYSTEM – B&W CAMERA

Key Parameters	Specification
Refreshing Rate	15 Hz
View of fields	45°
Resolution	320×240

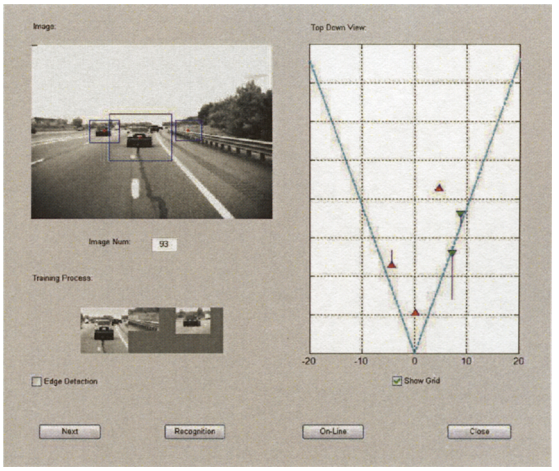

Fig. 2. A screenshot of the system development interface showing radar returns (red if going away and green if approaching, with relative speed shown as length of the line segments in the subplot on the right) and their projections on the image (upper left).

The radar-centered coordinates are projected into the image reference system using the perspective mapping transformation. The transformation is performed using calibration data that contain the intrinsic and extrinsic parameters of each camera. Given a 3D radar-returned target point projected on the image plane, an attention/radar window is created within the original 320×240 image, taking into account the expected maximum height and width of the vehicles.

For each radar window, the associated pixels are extracted as a single image. Each image is normalized in size to 56 rows and 56 columns of pixels, and the pixel intensities are normalized from 0 to 1. To avoid stretching small images into the fixed-sized window, each small image is placed in the upper left corner of the size-normalized image, and the other pixels are set to intensities of 0.5. Sometimes more than one object may be captured in each radar window, but for the purpose of classification, the radar window is assigned with only one label, either "vehicle" or "non-vehicle". We are not concerned about accurate labeling of multiple radar returns in the 56×56 window because it is quite rare that objects with different labels end up in the same window due to design specifics of our system.

By our creating a *set* of attention windows for each original image from the camera, the required mapping is greatly simplified. It allows us to learn detection and recognition of multiple objects within the same captured image, as long as there is a radar return from each object.

Our sensor integration leads to two advantages: (a) instead of manually segmenting target objects in an image, the radar system automatically provides areas of attention for the objects, obviating the need for potentially time-consuming human interpretations; (b) in each image from the camera, the desired output is simplified from a number of labels for objects in the original image to the individual label for each of the radar windows.

III. SPARSE CODING

The normalized pixels from each attention window form the appearance space, which may contain some information unrelated to the object, such as properties of a surface behind the object or noise. We map the raw representation from the appearance space to the coded space, wherein the coded representation of an object can be used for higher-level processing such as recognition or categorization. The details of this transformation are referred to as the coding process.

A variety of experimental and theoretical studies indicate that the human visual system encodes the retinal activations efficiently using sparse coding [10], [9] such that there are only a few active responses out of a large number of possible responses for any given stimulus. Theoretically, sparse coding leads to lower mutual information between neuron representations than the information in the case of non-sparse coding, meaning that the generated sparse codes will be more independent. This also improves memory capacity.

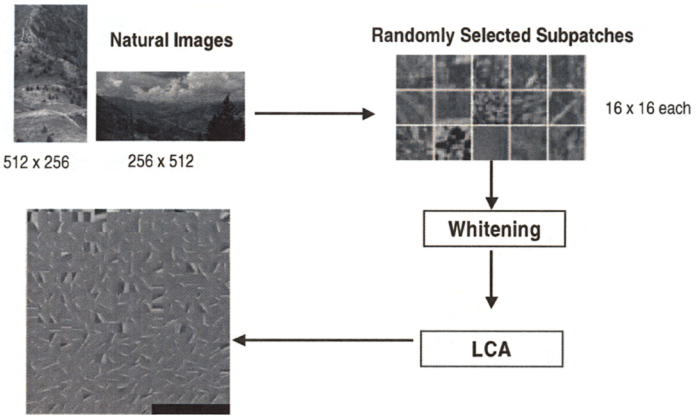

Fig. 3. The generation of orientation-selective filters by the lobe component analysis (LCA)[12].

In our proposed system, we implemented sparse coding by using orientation-selective filters, generated from images of nature henceforth called natural images through the Lobe Component Analysis (LCA) algorithm [12]. To do so, 16×16 image patches were incrementally selected from many random locations in 13 natural images [1]. Let $\mathbf{x}_i = [x_{i,1}, x_{1,2}, ...x_{i,d}]^T$ denote a column vector for the ith image patch, such that $d = 16 \times 16$. $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_n]$ is the set of randomly selected image patches, where n is the number of the patches. The whitening matrix $\mathbf{W}$ is generated by taking the matrix of principal components $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_d]$ and dividing each by its standard deviation. The matrix $\mathbf{D}$ is a diagonal matrix where the matrix element at row and column i is $\frac{1}{\sqrt{\lambda_i}}$ and λ_i is the eigenvalue of v_i. Then, $\mathbf{W} = \mathbf{VD}$. To whiten, we multiply the response vector(s) by this matrix.

$$\mathbf{Y} = \mathbf{WX} = \mathbf{VDX}. \tag{1}$$

where matrix $\mathbf{Y}$ contains a whitened input in each column. Thus, $dim(\mathbf{Y}) = 256 \times n$.

We train one layer of neurons, connected to a whitened input column vector $\mathbf{y}$, by the LCA algorithm which incrementally updates c such neurons (cells) represented by the column vectors $\mathbf{v}_1(t), \mathbf{v}_2(t), ..., \mathbf{v}_c(t)$ from whitened input samples $\mathbf{y}(1), \mathbf{y}(2), ..., \mathbf{y}(n)$. Each whitened input is a column vector extracted from the matrix $\mathbf{Y}$. At time t, the output of the layer is the response vector $\mathbf{z}(t) = [z_1(t), z_2(t), ..., z_c(t)]$.

Fig. 3 shows the result when using LCA on $c = 512$ neurons and $n = 1,500,000$ whitened input samples. Each neuron's weight vector is identical in dimension to the input, so it can also be displayed in each grid as a 16-row and 16-column image. (However, it must first be dewhitened in order to display properly, i.e., $\mathbf{x} = \mathbf{VD}^{-1}\hat{\mathbf{x}}$.) In Fig. 3, the component with the most wins is at the top left of the image grid, and it progresses through each row until the one with the least wins, at the bottom right. We discarded neurons with too low update totals (wins or cell age), and kept 431 neurons.

Each neuron's weight vector shows localized orientation patterns, therefore we call them orientation-selective feature detectors. Let $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_{c'}]$ denote a set of orientation-selective feature detectors, where c'=431. Each $\mathbf{f}_i = [f_{i,1}, f_{1,2}, ...f_{i,d}]^T$ is a (non-negative) feature detector with $d = 16 \times 16 = 256$.

It must be emphasized that such orientation-selective feature detectors are generated by *problem-independent* real-world natural images. This makes our system less dependent on problem domain experts, and does not require ad hoc image processing methods.

A. Local Receptive Fields

Hard-coded square (16×16) initial receptive fields are spread over the entire 56×56 attention window with overlap. Fig. 4 shows the arrangement. We used a stagger distance of 8 pixels, resulting in 36 total local receptive fields for the 56×56 image. Each receptive field is represented by a column vector $\mathbf{l}_i$ with dimension 256. Thus, for every window image, we have a non-negative matrix $\mathbf{L} = [\mathbf{l}_1, \mathbf{l}_2, ...\mathbf{l}_m]$, where $m = 36$.

B. Sparse Representation

The generated sparse representation is

$$\mathbf{S} = \mathbf{F}^T \times \mathbf{L}$$

The matrix $\mathbf{S}$ is non-negative and reshaped to be a vector with $431 \times 36 = 15,516$ dimensions. Thus, it maps the raw pixel representation (56×56) to a higher-dimensional, encoded space (15,516), which leads to a sparse representation of inputs (see Fig. 5).

IV. Multilayer In-place Learning Networks

The generated sparse representation are fed into our classifier – Multilayer In-place Learning Network (MILN). (Note that the LCA employed in Sec. III is a single-layer, unsupervised version of the MILN.) In in-place learning, each neuron

[1] from http://www.cis.hut.fi/projects/ica/data/images/ via Helsinki University of Technology

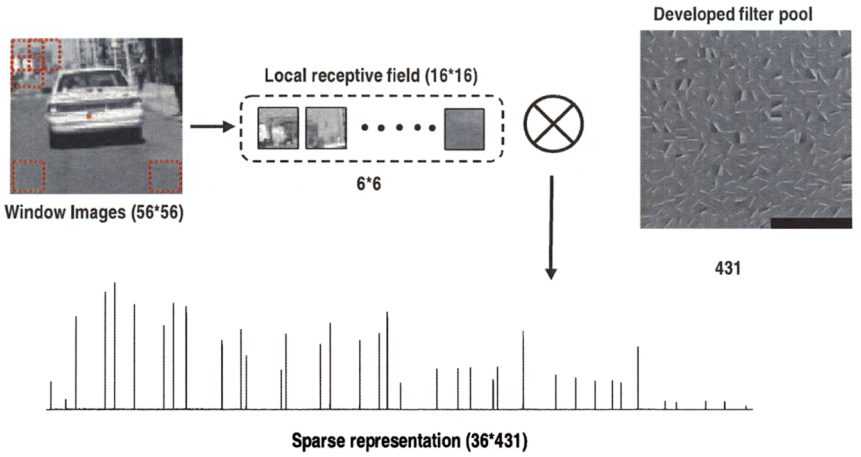

Fig. 5. The generated sparse representation by developed filters and extracted local receptive fields.

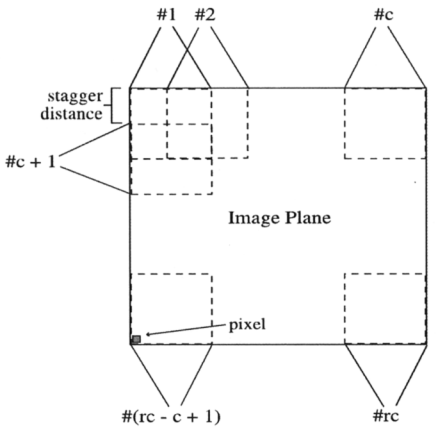

Fig. 4. Receptive field boundaries with rc squares of 16×16 pixels. The neighboring receptive fields overlap by the stagger distance both horizontally and vertically.

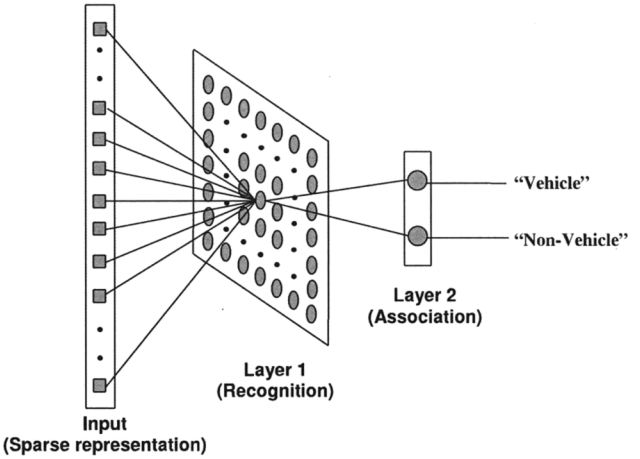

Fig. 6. In MILN, neurons are placed on different levels in the end-to-end hierarchy – from sensors to decision makers. A neuron in the hidden layer has both feedforward and feedback weights.

is not allowed to utilize partial derivatives (as in the popular backpropagation) or covariance matrices. Each neuron learns on its own, i.e., using only the data it receives from its inputs which may include feedback.

Unlike the LCA, the MILN has multiple layers. In this application, the sparse representations are considered to be on the bottom (input layer) and the class labels on the top (layer 2). Each MILN neuron has a bottom-up (excitatory) weight vector $\mathbf{w_b}$ that links input lines from the previous layer. Neurons in layer 1 also have top-down (excitatory or inhibitory) weights $\mathbf{w_t}$ that link the output from the neurons in the next layer, achieving the supervision by the top-down projection from the decision makers (layer 2). Note that the top-down weights of neurons in the layer 1 act as the bottom-up weights of neurons in the layer 2. Each linked weight pair (i, j) shares the same value, i.e., $\mathbf{w_t}_i^1 = \mathbf{w_b}_j^2$.

Fig.6 illustrates MILN in the proposed system. Algorithm 1 is the MILN learning algorithm $\mathbf{z}(t) = \text{MILN}(\mathbf{s}(t), \mathbf{m}(t))$, where $\mathbf{s}(t)$ is the input vector (sparse representations) at time t, and $\mathbf{m}(t)$ is the corresponding target vector (supervision signal). The vector $\mathbf{z}(t)$ contains responses of neurons in every layer. Note that Algorithm 1 may learn continuously ($n \to \infty$ in step 2), if required.

V. Experiments and Results

We used a properly equipped test vehicle to capture real-world image and radar sequences for training and testing. Our preliminary dataset consists of 400 images – stretches of roads in different places of city or highway environment, under different viewpoint variations, illuminations and scales (see Fig. 7 for a few examples of different environments). The original 320×240 images were captured with 0.5-second sampling. There were 499 samples in the vehicle class and 264 samples in the other object class. (Each radar window was size-normalized to 56×56 and intensity-normalized to

Algorithm 1 MILN

For $l = 1, 2$, set: $c_l = 100$, $c_l = 2$, $\alpha_1 = 0.3$, $\alpha_2 = 0.0$, the output of the layer l at time $t = 0$ to be $\mathbf{z}_l = \mathbf{0}$.
for $t = 1, 2, ..., n$ **do**
  $\mathbf{y}(t) = \mathbf{s}(t)$;
  for $l = 1, 2$ **do**
    for $i = 1, 2, ..., c_l$ **do**
      Compute pre-response of neuron i from bottom-up and top-down input connections:

$$b_i = (1 - \alpha_l) \frac{\mathbf{w}_{\mathbf{b}_i}^l(t) \cdot \mathbf{y}(t)}{\|\mathbf{w}_{\mathbf{b}_i}^l(t)\| \|\mathbf{y}(t)\|},$$

      if $l = 1$ **then**

$$\hat{z}_{l,i} = g_i \left(b_i + \alpha_l \frac{\mathbf{w}_{\mathbf{t}_i}^l(t) \cdot \mathbf{m}(t)}{\|\mathbf{w}_{\mathbf{t}_i}^l(t)\| \|\mathbf{m}(t)\|} \right),$$

      else

$$\hat{z}_{l,i} = g_i(b_i),$$

      end if
      where g_i is the standard neural network sigmoidal function.
    end for
    Simulating lateral inhibition, decide the winner: $j = \arg\max_{1 \le i \le c_l} \{\hat{\mathbf{z}}_{l,i}(t)\}$.
    If $l = 1$, the 3×3 neighboring cells are also considered as winners and added to the winner set $\mathbf{J}$ for the subsequent updating of the weights $\mathbf{w_b}$, otherwise $\mathbf{J} = \{j\}$.
    The winner set $\mathbf{J}$ may still contain neurons with zero pre-responses $\hat{\mathbf{z}}_l$. Define a sub-set $\mathbf{J}' \subseteq \mathbf{J}$, such that the response $z_{l,j} = \hat{z}_{l,j}$ if $\hat{z}_{l,j} \neq \mathbf{0}$, $\forall j \in \mathbf{J}'$.
    Update the number of hits (cell age) n_j ($j \in \mathbf{J}'$): $n_j \leftarrow n_j + 1$, and compute $\mu(n_j)$ by the amnesic function:

$$\mu(n_j) = \begin{cases} 0 & \text{if } n_j \le t_1, \\ c(n_j - t_1)/(t_2 - t_1) & \text{if } t_1 < n_j \le t_2, \\ c + (n_j - t_2)/r & \text{if } t_2 < t, \end{cases}$$

    where plasticity parameters $t_1 = 20, t_2 = 200, c = 2, r = 2000$ in our implementation.
    Update the winners $j \in \mathbf{J}'$ if $l = 1$, or all neurons if $l = 2$:

$$\mathbf{w}_{\mathbf{b}_j}^l(t) = \begin{cases} w_1 \mathbf{w}_{\mathbf{b}_j}^l(t-1) + w_2 z_{l,j} \mathbf{y}(t), & \text{if } l = 1, \\ w_1 \mathbf{w}_{\mathbf{b}_j}^l(t-1) + w_2 m_j \mathbf{y}(t), & \text{if } l = 2, \end{cases}$$

    where the scheduled plasticity is determined by its two age-dependent weights w_1,w_2:

$$w_1 = (n_j - 1 - \mu(n_j))/n_j, w_2 = (1 + \mu(n_j))/n_j,$$

    with $w_1 + w_2 \equiv 1$.
    For all $1 \le i \le c$, $i \notin \mathbf{J}'$, $\mathbf{w}_{\mathbf{b}_i}^l(t) = \mathbf{w}_{\mathbf{b}_i}^l(t-1)$.
    $\mathbf{y}(t) = \mathbf{z}_l(t)$;
  end for
end for

{0 1}.)

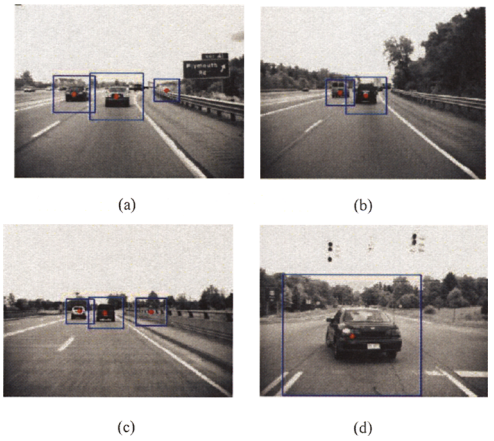

Fig. 7. Examples of images containing radar returns which are used to generate attention windows (blue borders). This figure shows examples of different road environments in our dataset.

A. Incremental Learning

Our learning method incrementally updates the network weights using one piece of training data at a time. After the network initialization with $10 \times 10 = 100$ samples $\mathbf{w}_{\mathbf{b}_i^1} = s(i), \forall i = 1, 2, ..., 100$, the rest of data is used for the incremental learning by the network. The network can be trained quite quickly if each training sample is presented only once (see Fig. 8 (a)). Even quicker and more accurate training results from relearning already presented samples, i.e., when a background training process keeps training the MILN on samples already seen by the system (see Fig. 8 (b)). The performance variability is illustrated in Fig. 8 (c) for 10 experiments of the incremental learning with randomly chosen samples.

We also carried out an incremental learning experiment with higher frequency of time sampling ("oversampling"), i.e., when the images are much more strongly correlated from one time step to another. We simulated the 0.1-second sampling by training the MILN on each image sample five times before proceeding to the next image sample. Our results indicate that the performance is in between the performances of Fig. 8 (a) and (b) in terms of speed of training, as expected.

B. Performance Evaluation and Comparison

To test the effectiveness of orientation-selective filters (resulting in sparse representations of samples), we compare the performance with the network without filters, where the input layer of the MILN is represented directly by the pixel intensity values. On average, the detection rate is improved by 5% for vehicle samples and 7% for non-vehicle samples by using the filters.

Figs 9 and 10 show the Receiver Operating Characteristic (ROC) curves for the MILN using sparse input representation and the ROC curves for the MILN with the pixel input. The

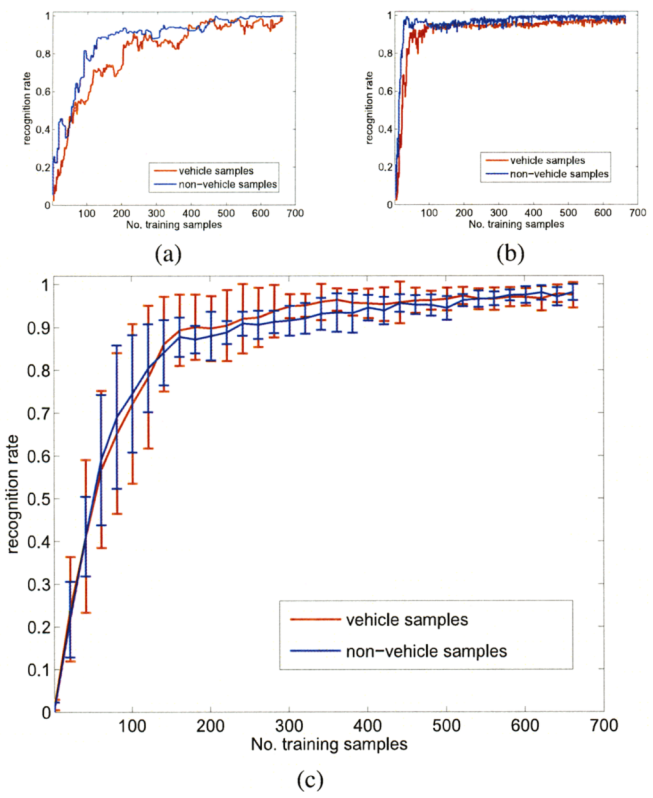

Fig. 8. (a) Efficiency of incremental learning: the number of training samples required to achieve the appropriate recognition rate. (b) Same as in (a) except that retraining on already presented samples is carried out. (c) Spread of performance for 10 experiments as in (a) (the error bars show standard deviation).

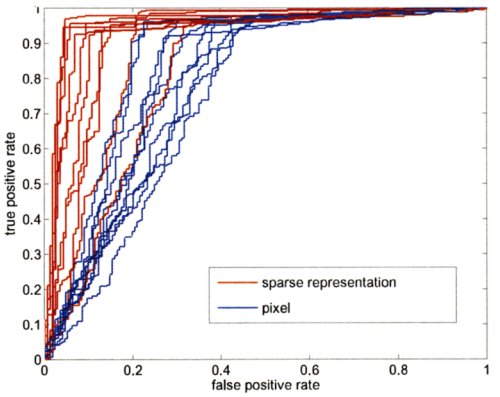

Fig. 9. The ROC graphs for the MILN with sparse input representation (red) and the one with the pixel input (blue); 10 experiments in both categories. We used 100 randomly chosen samples for initialization, 165 samples for training, and 498 for testing.

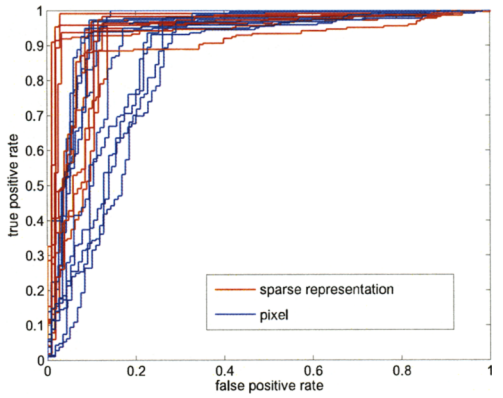

Fig. 10. Same as in Fig. 9, except 331 samples used for training and 332 for testing.

ROC curves are obtained according to the algorithm described in [13]. Performance characteristics of the MILN with the sparse input are more attractive on average than those of the MILN with the pixel input, although there exists a considerable spread due to the data size limitation.

Four different algorithms were compared for the purpose of their performance evaluation in the proposed sensor fusion system, where an efficient (memory controlled), real-time (incremental and fast), autonomous (without turning the system off to change or adjust), and extendable (the number of classes can increase) architecture is sought. We tested the following classification methods: k-Nearest Neighbor (NN), with $k = 1$ and $L1$ distance metric for baseline performance, incremental Support Vector Machine (I-SVM)[1][2], Incremental Hierarchical Discriminant Regression (IHDR) [11] and our method MILN. We used a linear kernel for I-SVM as suggested for high-dimensional problems [4]. (We experimented with several settings for an RBF kernel but did not observe as good a performance as with the linear kernel.)

Inputs to all systems in this comparison were two types. First, inputs were from the non-transformed – appearance or "pixel" – space with input dimension of $56 \times 56 = 3136$. Second, inputs were taken after transformation into the sparse-coded space (with dimension of $36 \times 431 = 15,516$).

The results of 10-fold cross validation are summarized in Tables III and IV (the numbers after $\pm$ sign indicate standard deviations for 10 repetitions). NN demonstrated the worst performance, and it is also prohibitively slow. IHDR combines the advantage of NN with an automatically developed overlaying tree structure that organizes and clusters the data. It is useful for extremely fast retrievals. IHDR performs better and faster than NN, and it can be used in real time. However, IHDR typically takes a lot of memory. It allows sample merging, but in our case it saved every training sample, not using memory efficiently. I-SVM performed well with both types of input, and it uses the least memory, in terms of the number of support vectors automatically determined by the data, but training time is the worst. Another potential problem with I-SVM is its lack of extendability to situations when the same data may be later expanded from the original two to more than two classes. As general purpose regressors, IHDR and MILN are

[2]Available @ http://www.biology.ucsd.edu/~gert/svm/incremental/

TABLE III

AVERAGE PERFORMANCE & COMPARISON OF LEARNING METHODS FOR PIXEL INPUTS

Learning Method	Overall Accuracy	"Vehicle" Accuracy	"Other Objects" Accuracy	Training Time Per Sample	Test time Per Sample	Final # Storage Elements
NN	$93.89 \pm 1.84\%$	$94.32 \pm 1.42\%$	$93.38 \pm 2.04\%$	N/A	432.0 ± 20.3 ms	621
I-SVM	$94.38 \pm 2.24\%$	$97.08 \pm 1.01\%$	$92.10 \pm 6.08\%$	134.3 ± 10.0ms	$\mathbf{2.2 \pm 0.1}$ ms	$\mathbf{44.5 \pm 2.3}$
IHDR	$\mathbf{95.87 \pm 1.02}\%$	$96.36 \pm 0.74\%$	$\mathbf{95.62 \pm 2.84}\%$	$\mathbf{2.7 \pm 0.4}$ms	4.7 ± 0.6 ms	689
MILN	$94.58 \pm 2.34\%$	$\mathbf{97.12 \pm 1.60}\%$	$91.20 \pm 5.31\%$	17.1 ± 2.0ms	8.8 ± 0.6ms	100

TABLE IV

AVERAGE PERFORMANCE & COMPARISON OF LEARNING METHODS FOR SPARSE CODED INPUTS

Learning Method	Overall Accuracy	"Vehicle" Accuracy	"Other Objects" Accuracy	Training Time Per Sample	Test time Per Sample	Final # Storage Elements
NN	$94.32 \pm 1.24\%$	$95.43 \pm 1.02\%$	$91.28 \pm 1.86\%$	N/A	2186.5 ± 52.5ms	621
I-SVM	$96.79 \pm 1.17\%$	$97.23 \pm 1.20\%$	$\mathbf{96.40 \pm 3.27}\%$	324.5 ± 22.4 ms	$\mathbf{7.6 \pm 0.3}$ ms	$\mathbf{45.2 \pm 2.6}$
IHDR	$96.54 \pm 1.83\%$	$96.79 \pm 1.04\%$	$96.31 \pm 2.05\%$	$\mathbf{12.2 \pm 1.3}$ ms	21.5 ± 1.7 ms	689
MILN	$\mathbf{97.14 \pm 1.27}\%$	$\mathbf{97.93 \pm 1.63}\%$	$95.46 \pm 2.54\%$	109.1 ± 3.2 ms	42.6 ± 0.4 ms	100

readily extendable.

Overall, MILN is very competitive for any comparison category, although it is not always the "best" in all categories, as currently implemented. NN is slower in testing, whereas I-SVM is slower in training and not extendable without full retraining. IHDR uses more memory and does not represent information efficiently and selectively. Our system in Figs 1 and 4 extends MILN to processing local receptive fields.

VI. CONCLUSION

Our results show promise for the development of object recognition and understanding systems based on the sensor fusion framework, in which the radar-directed attention mechanism reduces complexity of the analysis needed for each image frame. The information processing starts with orientation-selective feature detectors generated via the unsupervised LCA algorithm applied to problem-independent natural images. The attention windows are processed by orientation-selective feature detectors, generating a sparse representation for each window.

The incremental learning in our system is based on the MILN architecture. The sparse input representation helps the MILN to reach a performance better than its performance on the original pixel input. Though our system is designed for a specific automotive application, the general approach of combining different sensing modalities with the MILN is also suitable for other applications including those in the area of developmental robotics.

ACKNOWLEDGEMENT

We wish to thank Dr. Michael James for his input to this project.

REFERENCES

[1] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, volume 13, pages 409–415, Cambridge, MA, 2001.

[2] C. Coue, T. Fraichard, P. Bessiere, and E. Mazer. Multi-sensor data fusion using bayesian programming: An automotive application. In *International Conference on Intelligent Robots and Systems*, Lausanne Switzerland, 2002.

[3] R. Grover, G. Brooker, and H. F. Durrant-Whyte. A low level fusion of millimeter wave radar and night-vision imaging for enhanced characterization of a cluttered environment. In *Proceedings 2001 Australian Conference on Robotics and Automation*, Sydney.

[4] C. Hsu, C. Chang, and C. Lin. A practical guide to support vector classification. 2003.

[5] T. Luwang J. Weng, H. Lu and X. Xue. A multilayer in-place learning network for development of general invariances. *International Journal of Humanoid Robotics*, 4(2), 2007.

[6] T. Jochem and D. Langer. Fusing radar and vision for detecting, classifying and avoiding roadway obstacles. In *Procedings IEEE Symposium on Intelligent Vehicles*, Tokyo.

[7] J. Laneurit, C. Blanc, R. Chapuis, and L. Trassoudaine. Multisensorial data fusion for global vehicle and obstacles absolute positioning. In *Proceddings of IEEE Intelligent Vehicles Symposium*, Columbus.

[8] Shunji Miyahara et al. Target tracking by a single camera based on range-window algorithm and pattern matching. In *SAE 2006 World Congress and Exhibition*, Detroit.

[9] B.A. Olshausen and D.J. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14:481–487, 2004.

[10] B. A. Olshausen and D. J. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by VI? *Vision Research*, 37:3311–3325, 1997.

[11] J. Weng and W.S. Hwang. Incremental hierarchical discriminant regression. *IEEE Trans. on Neural Networks*, 2006.

[12] J. Weng and N. Zhang. Optimal in-place learning and the lobe component analysis. In *Proc. World Congress on Computational Intelligence*, Vancouver, Canada, July 16-21 2006.

[13] T. Fawcett. ROC graphs : Notes and practical considerations for researchers. Technical report, HP Laboratories, MS 1143, 1501 Page Mill Road, Palo Alto CA 94304, USA, March 2004. Also available online.