Learning Framework for Robust Obstacle Detection, Recognition, and Tracking

Vinh Dinh Nguyen, Hau Van Nguyen, Dinh Thi Tran, Sang Jun Lee, and Jae Wook Jeon, *Member, IEEE*

***Abstract*— This paper introduces a general framework for detection, recognition, and tracking preceding vehicles and pedestrians based on a deep learning approach. The proposed framework combines a novel deep learning approach with the use of multiple sources of local patterns and depth information to yield robust on-road vehicle and pedestrian detection, recognition, and tracking. The proposed system is first based on robust obstacle detection to identify obstacles appearing along the road that are likely to be vehicles and pedestrians, implemented as an efficient adaptive U-V disparity algorithm. Second, the results from the obstacle detection stage are input into a novel vehicle and pedestrian recognition system based on a deep learning model that processes multiple sources of depth information and local patterns. Finally, the results from the recognition stage are used to track detected vehicles or pedestrians in the next frame by means of a proposed tracking and validation model. The proposed framework has been thoroughly evaluated by inputting several vehicle and pedestrian data sets that were collected under various driving conditions. Experimental results show that this framework provides robust vehicle and pedestrian detection, recognition, and tracking with high accuracy, and also satisfies the real-time requirements of driver assistance systems.**

***Index Terms*— Deep learning, vehicle and pedestrian detection, unsupervised learning.**

I. Motivation and Main Contributions

RESEARCH into vision-based driver assistance systems (DASs) is an active field, and systems are currently under development in various car automotive companies and universities worldwide. Development continues due to the increasing requirements for safety and accident avoidance in DASs. A vision-based DAS consists of many modules, including modules for obstacle detection and recognition, obstacle tracking, and prediction of obstacle behavior (Fig. 1) [1]. Among these, obstacle detection, recognition, and tracking are the basic functions used by the DAS to understand traffic scenes. Most existing systems for detecting and recognizing obstacles are designed to take 2D images as an input and are trained on a large labeled dataset. However, the performance of such systems degrades considerably under conditions different from those represented in the labeled training dataset. In addition, the labeling of data is time-consuming and difficult, requiring the collection and manual labeling of a large number of training examples to capture numerous variations in object appearance. Recently, deep learning has often achieved the highest detection and recognition rates for tasks where large sets of data are available, such as face detection and speech recognition [13]–[15]. In this paper, we will develop a deep learning model based on multiple local patterns and depth information (from a stereo vision system) for accurate detection, recognition, and tracking of pedestrians and vehicles under various driving conditions. The proposed system has three main stages: pre-processing with multiple local patterns, un-supervised and supervised training. Unsupervised training contains K-means clustering, an auto-encoder, a tile convolutional neural network (TCNN), and a recursive tile convolutional neural network (R-TCNN). The soft-max regression is used in the supervised training stage. The proposed system improves the accuracy of the state-of-the-art recursive convolution neural network (RCNN) [13] from 96.75 % to 97.75 % and improves the processing time of the state-of-the-art RCNN from 5500 ms to 70 ms using the charge coupled device (CCD) stereo dataset under normal conditions [2]. In addition, the proposed system obtained much better results than RCNN using the Heidelberg Collaboratory for Image Processing (HCI) dataset under challenging conditions [23], improving the accuracy from 89 % to 96.25 %.

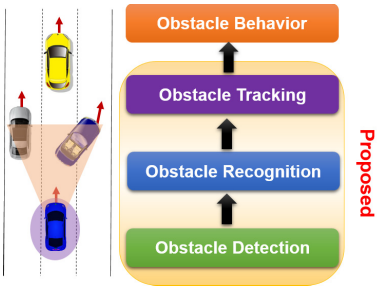

Fig. 1. Driver assistant system.

II. Related Work

Obstacle detection and recognition systems that detect and recognize the presence of pedestrians and vehicles play an

Manuscript received December 1, 2015; revised April 3, 2016, July 8, 2016, and September 12, 2016; accepted September 23, 2016. Date of publication October 19, 2016; date of current version May 29, 2017. This work was supported by the Samsung Research Funding Center of Samsung Electronics under Project SRFC-IT1402-12. The Associate Editor for this paper was Z. Duric.

The authors are with the School of Information and Communication Engineering, Sungkyunkwan University 16419, South Korea (email: vinhnd@skku.edu; haunguyen@skku.edu; dinhtt@skku.edu; vision.fpga@gmail.com; jwjeon@yurim.skku.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2016.2614818

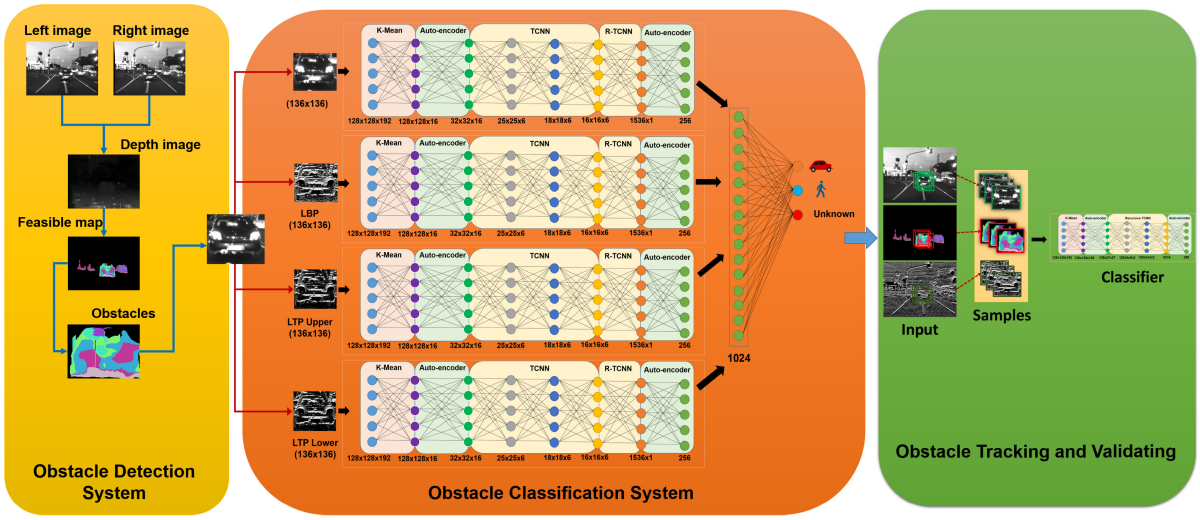

Fig. 2. General model of the proposed deep learning framework.

important role in DASs. They can greatly reduce the incidence of accidents due to careless driving. Therefore, the development of a robust and real-time obstacle detection and recognition system is heavily desirable. However, the development of such a method is still challenging to the research community due to various conditions in real-world driving.

In particular, on-road detection and recognition of obstacles has been a topic of great interest to researchers over the past decade [2]–[5]. Various methods have been proposed for on-road obstacle detection and recognition based on sensing data from radars, lidars, and cameras. The resulting obstacle detection and recognition data can be used to identify and classify on-road behavior. Millimeter-wave radar is widely used for the detection of obstacles on the road. However, vehicle-mounted radar sensors cannot provide a wide field-of-view. In recent years, the lidar's popularity has increased for use in on-road obstacle detection; however, it is more sensitive to weather conditions than radar and is still very expensive. Vision-based vehicle detection and recognition uses one or more camera as the primary sensor suite. Cameras provide a wide field of view, allowing for the detection, recognition and tracking of several obstacles across multiple lanes. Moreover, cameras are cheaper than other active sensors. Accordingly, in the present paper, we aim to develop a robust vehicle and pedestrian detection, recognition and tracking method that utilizes cameras. There are two main strategies for detecting and recognizing obstacles in various driving conditions: non-learning-based and learning-based methods. Non-learning-based obstacle detection and recognition is often designed for detecting and recognizing vehicles or pedestrians based on hand-designed features such as symmetry, color, shadow, texture, vehicle light, vehicle motion, and the relationships between obstacles' sizes and distances [5]–[8]. However, it is quite challenging to develop a robust hand-designed feature that can work well under varied driving conditions. Early non-learning based methods relied heavily upon single specific features, which are typically not robust enough in themselves to detect and recognize vehicles and pedestrians. Many methods based on learning strategies, such as neural networks, support vector machines (SVM) and AdaBoost, have yielded high performance [8]–[12]. In the early development of machine learning for obstacle detection and recognition, researchers focused only on how to design good features with a labeled dataset for supervised learning algorithms implemented in neural networks. Most current learning methods have shallow architectures (1, 2, and 3 levels).

Recently, deep learning has often achieved the highest detection and recognition rates for tasks where large sets of data are available, such as face detection and speech recognition [13]–[15]. Richard et al. have successfully developed an efficient indoor object classification by using RCNN [13]. They introduced a new learning model based on a fusion of convolutional and recursive neural networks for learning features and classifying images. RCNN combines convolution and pooling into one efficient, hierarchical operation. Their proposed method obtained state-of-the-art results relative to other deep learning-based indoor object classifications. However, the processing times of current deep learning-based object detection and recognition methods are quite long due to the increasing numbers of parameters and layers in their deep architectures. Motivated by the benefits of RCNN with a 3D feature [13], in the present work we developed a general deep learning framework for detecting, recognizing, and tracking pedestrians and vehicles under various driving conditions, without spending a great deal of time developing traditional hand-designed features. The proposed deep learning system can improve the performance of existing vehicle and pedestrian detection and recognition systems by automatically discovering meaningful structures from multiple sources of local patterns and depth information.

III. Proposed Deep Learning Framework

The proposed framework includes three main stages (Fig. 2). First, a stereo vision system [17] produces a robust disparity map under various driving conditions. The resulting disparity

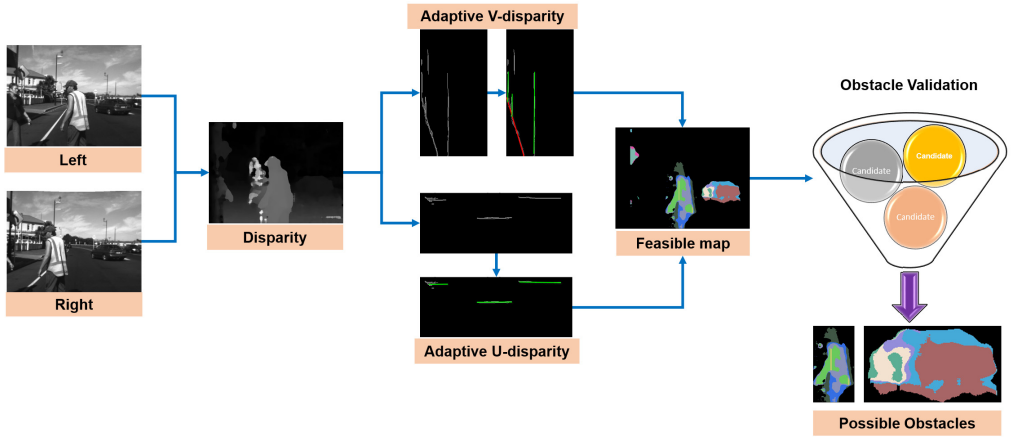

Fig. 3. Block diagram of the proposed adaptive U-V disparity algorithm to determine obstacle locations.

map is used as an input to the proposed adaptive U-V disparity algorithm, which then performs efficient obstacle detection. Second, detected obstacles are then classified into three categories, vehicles, pedestrians, and unknown obstacles (to remove false alarms from the detection stage), by means of the proposed deep learning system that considers multiple sources of information on local patterns and depth. Third, the results of the classification stages can be used to track detected vehicles or pedestrians in the next frame by means of a fast and robust tracking method.

A. Obstacle Detection Using Adaptive U-V Disparity

Most existing vehicle or pedestrian detection techniques are designed to use window-based methods to scan through entire images line by line to detect preceding cars or pedestrians. However, this approach has two main disadvantages: first, scanning the entire image is too time-consuming and multiple window sizes must be used to detect obstacles at various distances. Second, the accuracy of the proposed system is decreased because the detection and recognition schemes are applied to all sub-regions throughout the entire image. In this research, we develop an efficient obstacle detection algorithm that can provide accurate information on obstacles' locations, widths and heights based on stable disparity information obtained using a method reported by Nguyen *et al.* [17].

Stereo vision has been used in many applications, such as scene analysis, vehicle detection, and pedestrian detection [2], [3], [18], [19]. To isolate obstacles from road surfaces, Labayrade et al. proposed a V-disparity technique, where V corresponds to the vertical coordinate in the image coordinate system [20]. More recently, Zhang et al. developed an obstacle detection algorithm using the U-V disparity information [21]. However, their proposed algorithms fail to provide accurate U-V information when the original disparity map is noisy. Therefore, this paper introduces adaptive U-V disparity techniques to provide more stable obstacle information when noise is encountered in the original disparity image. Fig. 3 shows a block diagram of the proposed method to calculate U-V disparity information. V-disparity [20] and U-disparity [22] are often calculated by directly measuring the numbers of pixels having the same disparity level along rows or columns in the disparity image, respectively. However, this approach can fail when the original disparity image is noisy. In this paper, we introduce an adaptive threshold α to overcome this limitation. U-disparity (U_d) and V-disparity (V_d) are calculated as follows:

$$V_d(i,j)_{i\in \mathrm{H}, j\in\Delta} = \sum_{k\in W} \Psi_v(i,j,k)$$
$$\Psi_v(i,j,k) = \begin{cases} 1, & if\ |I_d(i,k) - j| \le \alpha \\ 0, & else \end{cases} \quad (1)$$
$$U_d(i,j)_{i\in\Delta, j\in W} = \sum_{k\in \mathrm{H}} \Psi_u(i,j,k)$$
$$\Psi_u(i,j,k) = \begin{cases} 1, & if\ |I_d(k,j) - i| \le \alpha \\ 0, & else \end{cases} \quad (2)$$

where $\Psi_v(i,j,k)$ and $\Psi_u(i,j,k)$ are binary functions that are used to threshold the intensity values of the V and U disparity images, respectively. H and W are the width and height, respectively, of the original disparity map, Δ is the maximum disparity level, and $I_d(i,k)$ is the intensity of pixel (i,k) in the original disparity map. α is used to measure the intensity difference within a window in the entire original disparity map as follows:

$$\alpha = \frac{1}{N}\sum_{W_p} \frac{1}{(W_p - 1)} \sum_{q\in W_p, q\neq p} \left| I_d(Z_p) - I_d(Z_q) \right| \quad (3)$$

where Z_p is the intensity value at center pixel P in the local region. Z_q is the intensity value at neighborhood pixel q in the local region. N is the total number of pixels in the disparity image and W_p is the window. Fig. 3 shows the results of the U-disparity, V-disparity, and feasible map prepared by mapping the U-V information to the original disparity map by using $\alpha = 1.35$; in the figure, a red line represents the road, and blue lines represent obstacles. There are many obstacles appearing along the road; however, we are interested in obstacles shaped similarly to vehicles or pedestrians. Therefore, an obstacle validation stage is introduced to reject other obstacles from consideration, such as roads, trees, and buildings.

Algorithm 1 Obstacle Validation

input : the original disparity image D, V-disparity information V_d, U-disparity information U_d
output: position of obstacle vehicle (location, width, height)

```
begin
    Build the lookup table L(distance, size) of the
    corresponding width and height of the car and the
    pedestrian at various distances from the camera
    system.
    Extract the road line and obtain Y position of all
    obstacles (N_y) using V_d
    for i ← 1 to N_x do
        Find the width W of the obstacle i using U_d.
        Find the height H of the obstacle i using the
        original disparity information.
        Compute the actual distance Γ of the obstacle i
        from the camera system using D
        if W in L(Γ) and H in L(Γ) then
            store the obstacle i
        end if
    end for
end
```

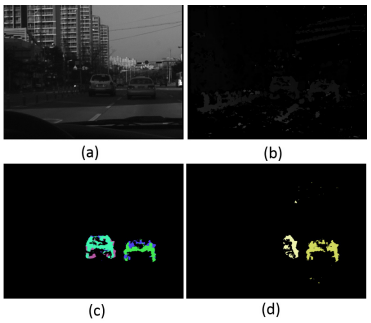

Fig. 4. Results of the proposed adaptive U-V disparity in comparison to the U-V disparity [21], [22] using the the CCD stereo dataset [2]. (a) left image. (b) disparity image. (c) The proposed method. (d) U-V disparity [21], [22].

The main work-flow of the validation process is shown in Algorithm 1.

Fig. 4 shows the results of the proposed adaptive U-V disparity in comparison to the U-V disparity [21], [22] using the CCD stereo dataset [2]. The proposed method obtained better performance than the U-V disparity algorithm [21], [22] when the original disparity image is noisy. Fig. 5 shows several results of adaptive U-V under various road driving conditions based on inputs from the HCI dataset [23] and the enpeda image sequence analysis test site (EISAT) dataset [24]. In these tests, the proposed system provided accurate locations of obstacles shaped similarly to vehicles or pedestrians.

However, there were still many obstacles that were not vehicles or pedestrians but had similar shapes. In the next section, we introduce a new deep learning model to accurately recognize vehicles and pedestrians based on the results of the obstacle detection stage. This obstacle recognition model can reject false detections made during the obstacle detection stage.

Fig. 5. Results of adaptive U-V disparity algorithm under various road driving conditions for the HCI and EISAT stereo datasets. Left, middle, and right columns are the left image, disparity image, and feasible map, respectively.

B. Obstacle Recognition (Classification)

Obstacle classification is one of the most important stages of a DAS. The results of obstacle classification allow scene analysis to be provided quickly to the driver, which might help to prevent accidents. Recently, Richard *et al.* [13] developed an object classification for indoor applications by using RCNN with a fixed tree model. Their proposed method obtained the highest performance for classifying the indoor objects under normal conditions. However, their proposed system performs worse in out-door conditions, where unknown factors, such as sunny, rainy, or snowy conditions might affect the input image. In addition, Le et al. have shown that RCNN only captures translational invariance and does not capture rotational invariance or out-of-plane rotation [14]. To overcome this limitation, Le *et al.* [14] proposed a tile convolution neural network based on topographic independent component analysis (TICA). Recently, the local binary pattern (LBP) and its variants have emerged as valid methods for improving performance for various applications, including texture classification, facial detection, and medical image processing [25]–[28]. LBP extracts the features of a local region by applying the sign function to the central pixel and its neighboring pixels. This method is simple to compute and is robust to noise, including noise caused by different lighting conditions. Therefore, motivated by the effective RCNN model of Richard *et al.* [13], the benefits of local patterns, and the advantages of tile convolution neural networks [14], herein we introduce a novel deep learning model for recognizing vehicles and pedestrians using recursive a tile convolutional neural network based on the use of multiple sources of information on local patterns and depth, including the use of an auto-encoder. Fig. 6 shows the block diagram of the proposed deep learning model for classifying vehicles and pedestrians. The proposed model includes three main stages: a pre-processing step to perform calculations of multiple local patterns, an un-supervised model that trains to discover meaningful features from multiple local patterns, and a supervised learning model that is trained to classify vehicles and pedestrians from among other obstacles.

1) Pre-Processing With Multiple Local Patterns: Ahonen *et al.* [25] proposed an LBP that is successful in

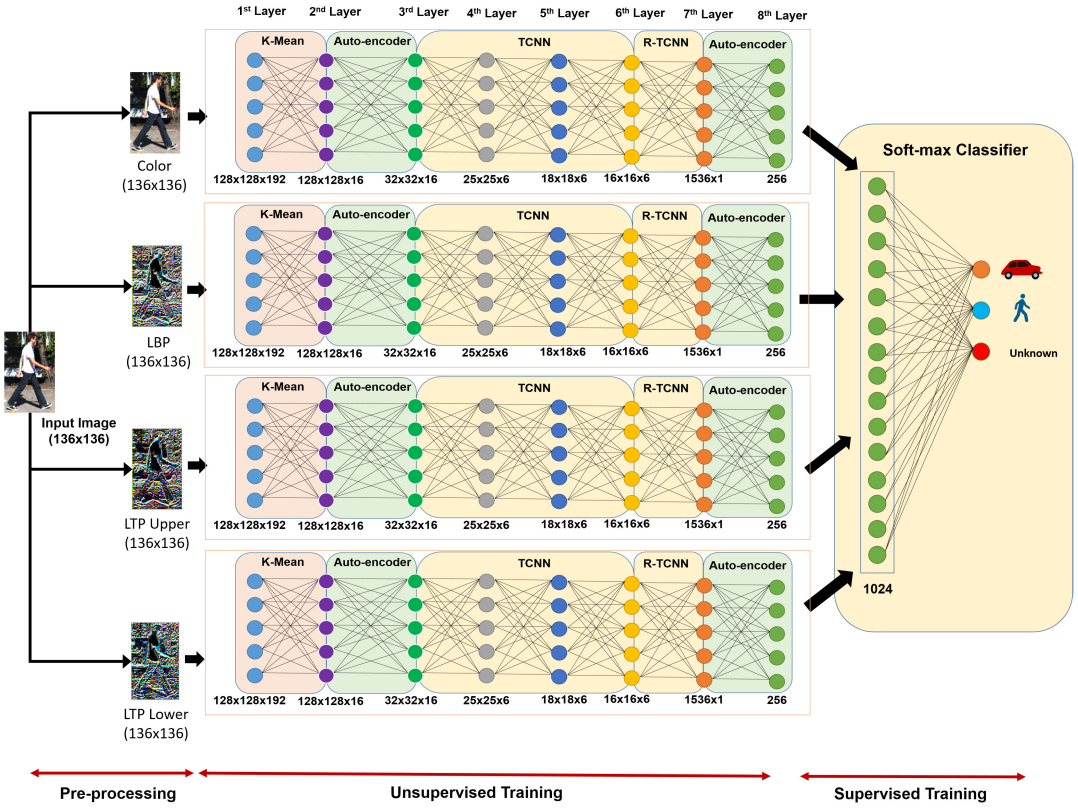

Fig. 6. General model of deep learning for classifying vehicles and pedestrians.

texture classification and image retrieval due to its simplicity and low computational complexity. The LBP method is a nonparametric method that captures the local structure of an image by thresholding the intensities of neighborhood pixels with the intensity of the center pixel. Given a center pixel $z_c = (x, y)$ in the image, the following LBP code is generated:

$$LBP_{N,R}(z_c) = \sum_{n=1}^{N} f(z_n, z_c) 2^n$$
$$f(z_n, z_c) = \begin{cases} 1, & I(z_n) \geq I(z_c) \\ 0, & I(z_n) < I(z_c) \end{cases} \quad (4)$$

where $I(z_c)$ and $I(z_n)$ are the gray values of the central pixel and the n-th neighbor pixel, respectively, and R is the radius of the neighborhood. Moreover, N neighborhood pixels can be located outside of the image grids. Their gray values can be calculated using bilinear interpolation, and the coordinates of z_n are determined as follows:

$$(x_n, y_n) = \left[x_c + R\cos\left(\frac{2\pi n}{N}\right), y_c - R\sin\left(\frac{2\pi n}{N}\right) \right] \quad (5)$$

However, the LBP is sensitive to random noise in uniform regions, or monotonic gray-level changes. Therefore, Tan and Trigg proposed a local ternary pattern (LTP) to overcome this problem, modifying the two-value encoding in the LBP to have three-value encoding instead [27], as follows:

$$LTP_{N,R}(z_c) = \sum_{n=1}^{N} f_{ltp}(z_n, z_c, \beta) 2^n$$
$$f_{ltp}(z_n, z_c, \beta) = \begin{cases} 1, & z_n \geq z_c + \beta \\ 0, & |z_n - z_c| < \beta \\ -1, & z_n \leq z_c - \beta \end{cases} \quad (6)$$

The LTP works well in the case of illumination changes, however finding the optimal threshold β for various conditions remains challenging. Therefore, this paper proposes an adaptive LTP that automatically estimates the parameter β as follows:

$$\beta = \beta_{id} + \beta_{dd}$$
$$\beta_{id} = \frac{1}{N}\sum_{W_p} \frac{1}{(W_p - 1)} \sum_{q \in W_p, q \neq p} \left| I(Z_p) - I(Z_q) \right|$$
$$\beta_{dd} = \frac{1}{N}\sum_{W_p} \sqrt{\frac{1}{(W_p - 1)} \sum_{q \in W_p, q \neq p} \left(\left| I(Z_p) - I(Z_q) \right| - \beta_{id} \right)^2} \quad (7)$$

where N is the total number of pixels in the input image. β is defined by using two main components: intensity difference (β_{id}) and distance difference (β_{dd}). β_{id} is used to measure the intensity difference within a window in the original image, and β_{dd} is used to measure the distance between intensity differences within a window with a given

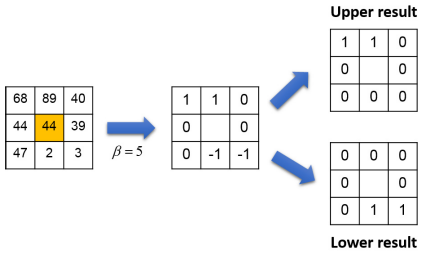

Fig. 7. Example of an LTP calculation using a fixed 3x3 window (LTP upper and LTP lower).

β_{id} value. The LTP results are divided into two LBP channels (LTP upper and LTP lower), as shown in Fig. 7. The LTP upper is generated by replacing the negative values in the original LTP by zeros. The LTP lower is generated by replacing the negative values in the original LTP by one and by replacing the positive values by zeros.

Local patterns might extract more discriminative structures from the local region, and are also less sensitive to noisy conditions. Most existing deep learning algorithms are only designed to consider color or depth features. Therefore, in this paper, we are the first to apply a deep learning model to discover and learn meaningful structures from multiple local patterns (such as LBP and LTP), which can help to increase the accuracy of the proposed classification system under difficult driving conditions. In the first layer, from the original input image (136x136 resolution), the proposed method performs window convolution, scanning over the entire input image to provide 128x128x192 input units with three channels, as illustrated in Fig. 6.

2) Unsupervised Training: In unsupervised training, a large amount of unlabeled dataset are used to learn abstract features in a manner that is less sensitive to noise. The main purpose of unsupervised learning is to help the neural network learn its own invariance features from an unlabeled dataset using multiple layers; this model is also known as deep learning.

In the second layer, the unsupervised training starts by performing K-means on multiple inputs, including LBP, LTP, and the raw image, independently. Based on the results of Coates *et al.* [29], our proposed system uses 16 K-means clusters to select 16 feature centers from large un-labeled datasets. It is worth noting that our proposed model includes color, LBP, LTP uppers, and LTP lowers (as described in Figure. 6). These four inputs are then used as inputs to independent K-means algorithms, which are easy to parallelize by using a graphics processing unit (GPU), thereby speeding up the processing. Thus, after performing K-Means clustering, we can convert the $128\times128\times192$ input units from the first layer to $128\times128\times16$ input units in the second layer.

In the third layer, an auto-encoder [30] is used to learn 16x32x32 hidden units from the 16x128x128 input units of the second layer. Our approach is different from that taken by Richard *et al.* [13], which used an average down-sampling technique including the Gaussian kernel to reduce the feature size. An auto-encoder is a two-layer neural network including an input-hidden-output layer that tries to learn an approximation of a density function by setting the input values equal to the output values. Given an unlabeled dataset $\left\{d_{unlabelled}{}^{(m)}\right\}_{m=1}^{M}$, an auto-encoder tries to learn a function $h\left(d_{unlabelled}{}^{(m)}; W, b\right) \approx \sigma\left(Wx^{(m)}+b\right)$ where $(Wx^{(m)}+b)$ is an approximation of $x^{(m)}$, and b is the bias value. σ is the activation function. We set the activation function σ to be the sigmoid function. An auto-encoder can capture meaningful features, such as a hierarchical grouping or a part-whole decomposition of the input image [15]. Algorithm 2 shows the main work flow of a stack auto-encoder algorithm using mini-batch training to learn a hidden parameter for training.

Algorithm 2 Auto-Encoder With Mini Batch Training

input : unlabelled dataset $\left\{d_{unlabelled}{}^{(m)}\right\}_{m=1}^{M}$
output: Weight W, and bias b

begin
  Define a fixed number of N patches for each iteration
  Define an accepted error ε to stop training
  for $m \leftarrow 1$ **to** M **do**
    Randomly select N sub-patches from M
    Compute and minimize the cost function
    $cf = \min_{W,b} \sum_{i=1}^{N} \left\| sig\left(W^T sig\left(Wx^{(i)}+b\right)\right) - x^{(i)} \right\|_2^2$
    using on N sub-patches
    Updated W, and b
    if $cf \leq \varepsilon$ **then**
      Compute the cost function cf using M patches
      if $cf \leq \varepsilon$ **then**
        Stop training and return W and b
      end if
    end if
  end for
end

In the fourth, fifth, sixth, and seventh layers, Richard *et al.* [13] used CNN to train multiple layers. However, CNN cannot learn complex translation invariant features. Therefore, to overcome this limitation, we proposed a recursive tile convolutional neural network taking advantage of the benefits of TCNN [14], as shown in Fig. 6 (this paper sets the number of recursive filters equal to 1 for the simplest case). TCNN tries to learn invariances, rather than relying on hard-coding, as in traditional CNN. TCNN is able to learn complex invariances (such as scale and rotational invariance) beyond translational invariance. A more detailed procedure describing the Recursive TCNN is given in Algorithm 3. In addition, supervised fine-tuning [31] is then applied to update the weight W in the single TCNN.

In the eighth layer, we applied an auto-encoder algorithm to reduce the feature size of the training input units by using 256 hidden units. Thus, the feature size input is decreased from 1536 to 256 units in this layer, as shown in Fig. 6. It is worth noting that color, LBP, and LTP inputs are trained independently to provide 256 units for each model. Thus, the total number of input units after un-supervised training is $256\times4 = 1024$.

Algorithm 3 Recursive Tile Convolutional Neural Network

input : unlabelled dataset $\{{d_{unlabelled}}^{(i)}\}_{i=1}^{M}$, tile size($ts$), reception field size($rf$), number of map($n$), number of input ($in$), number of hidden units (hid)
output: Weight W

begin
  Define number of recursive filters MAX_FILTER
  for $t \leftarrow 1$ **to** MAX_FILTER **do**
    Define maximum number of iteration MAX_ITER
    Define $\alpha_{(0)}^{(t)} = 1$
    Initialize $W_{(0)}$ and V
    Compute TICA cost [14] and gradient for linear search
    $$f_{(0)}^{t} = \sum_{i=1}^{M}\sum_{j=1}^{hid}\sqrt{\sum_{k=1}^{hid} V_{jk}\left(\sum_{l=1}^{in} W_{kl}^{(0)} x_l^{(i)}\right)^2}$$
    $$g_{(0)}^{(t)} = \frac{\partial(f_{(0)})}{\partial W^{(0)}}$$
    Perform a linear search until reaching maximum iteration or terminal condition
    for $a \leftarrow 1$ **to** MAX_ITER **do**
      Compute $W_{(a)}^{(t)} = W_{(a-1)}^{(t)} - \alpha_{(a-1)}^{(t)} g_{(a-1)}^{(t)}$
      Update $W_{(a)}^{(t)}$ using reception field rf
      Update $W_{(a)}^{(t)}$ using tile size ts
      Perform orthogonality [14] with local reception field on $W_{(a)}^{(t)}$
      Compute
      $$f_{(a)}^{t} = \sum_{i=1}^{M}\sum_{j=1}^{hid}\sqrt{\sum_{k=1}^{hid} V_{jk}\left(\sum_{l=1}^{in} W_{kl}^{(a)} x_l^{(i)}\right)^2}$$
      Set $\alpha_{(a)}^{(t)} = \frac{\alpha_{(a-1)}^{(t)}}{2}$
      if $f_{(a)}^{t} \leq f_{(a-1)}^{t}$ **then**
        Stop training and return $W_{(a)}^{(t)}$
      end if
    end for
  end for
end

3) Supervised Training Classifier: The results of unsupervised training are to learn invariant features or more abstract features from raw inputs. In supervised training, a smaller set of labeled data is used to learn and classify these features into specific categories. From the 1024 input units at the seventh layer, soft-max regression (also known as multinomial logistic regression) [38] is used to classify the input units into three output categories: vehicles, pedestrians, and other obstacles (Fig. 6). The final obstacle type is determined by selecting the output units that have the maximum activation value.

4) GPU Implementation to Speed up Training and Running Processes: We implemented the proposed model on a Tesla K40 GPU to reduce its processing time (Fig. 8). For the input image of 136x136, the processing time was considerably decreased from 5.5s on a PC (Core i7 4.0 GHz, 8.0 GB RAM) to 70 ms on the GPU. Thus, the GPU implementation of the proposed method can meet the real-time requirements of DASs. The processing time could be decreased further by reducing the input image resolution.

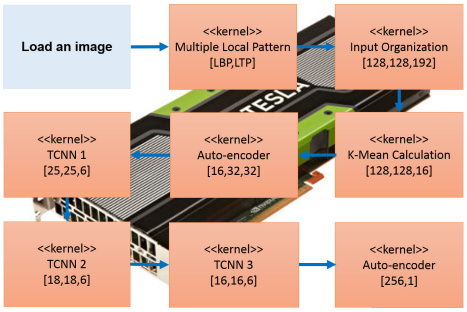

Fig. 8. Block diagram of the proposed method as implemented on a Tesla K40 GPU.

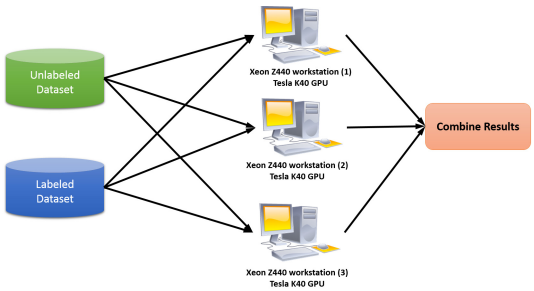

Fig. 9. Training model based on multiple workstations.

Because the training is also time-consuming, we designed a training model with a map reduction technique [16] to split the data set into several sub sets, and used a parallelism model to distribute the storage of parameters and gradient computations to different machines [15], as shown in Fig. 9. In each workstation, training on the data subsets was conducted by using the parallel GPU architecture introduced above. The training time was significantly reduced by using this double-parallel mechanism, whereby data was first divided among several workstations to perform training on the data in parallel, and each of these workstations used a GPU implementation to parallelize the training process. It is worth noting that we also applied a layer-wise strategy [30] in unsupervised training. For 20,480 training samples, the proposed system took approximately 90 minutes to train the entire system, while RCNN [13] took more than 11 hours to finish the training stage.

C. Obstacle Tracking and Validation

Once a vehicle or pedestrian is detected at the t^{th} frame, it can be tracked at next $(t+1)^{th}$ frame by using the information detected from t^{th} frame. There are several techniques that can be used to track moving obstacles, such as a Kalman filter or particle filters [32], [33]. Both of these methods have been used successfully to track preceding vehicles and pedestrians,

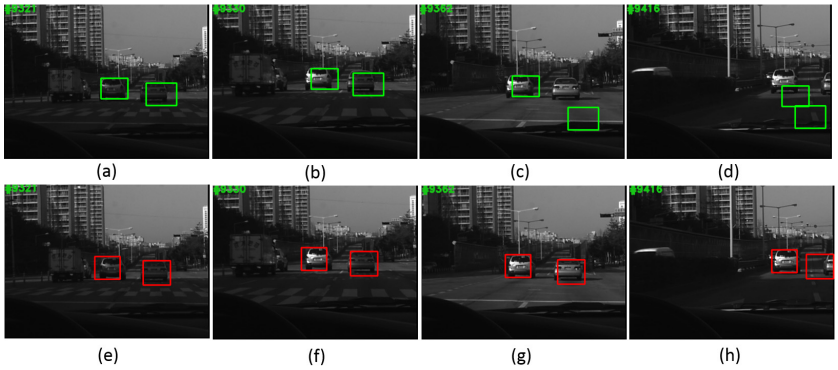

Fig. 10. Experimental tracking results of (a-d) CT [34] and (e-h) the proposed tracking method using the CCD stereo car dataset.

but they can fail due to drift problems caused by changes in appearance. Recently, Zhang *et al.* [34] proposed a state-of-the-art real-time comprehensive tracking (CT) algorithm that operates based on features extracted from a multiple scale image feature space in the compressed domain. However, CT directly uses the intensity of the original single image (left or right image). Therefore, it does not perform stably under challenging real-world illumination conditions, such as those encountered during nighttime driving or in other conditions with large fluctuations in illumination. The other main disadvantage of CT is its use of a fixed window-size to track a detected obstacle, which is not suitable for driving conditions in which the obstacle might move near or far from the camera system.

Fig. 10 (a)-(d) show the results of CT under the challenging driving conditions represented by the CCD dataset [2], in which the illumination changes significantly between continuous frames. CT failed under this test, as shown in frames 9362 (Fig. 10(c)) and 9416 (Fig. 10(d)). Therefore, we introduced an improvement to CT by combining the benefits of three features (original left, LBP, and adaptive U-V disparity) to overcome this limitation. The proposed tracking system integrates the adaptive U-V disparity, LBP, and raw RGB information. Therefore, in order to combine the three above features into a single feature for tracking, we need to define a threshold χ for the adaptive u-v disparity, κ for LBP, and ω for the raw RGB, as described in Algorithm 4. In our experiments, we found that $\omega < \kappa < \chi$ because the adaptive U-V disparity provides more robust feature than LBP, and raw RGB. We set $\chi = 0.5$, and $\kappa = 0.3$, and $\omega = 0.2$ throughout all experiments. Figs. 10(e)-(h) show the results of our proposed tracking model. The proposed method performs better than the CT in this test. In addition, CT uses a fixed window size to generate possible obstacle candidates. Therefore, if the tracked obstacle moves too near or too far from the system, CT fails to handle these situations. Fortunately, we can estimate the distance (in meters) from the proposed system to the preceding obstacle using the disparity information. Thus, the size of preceding vehicle (window size) can be estimated based on its distance to the camera system. Therefore, we proposed a tracking method that uses an adaptive window to track detected vehicles by

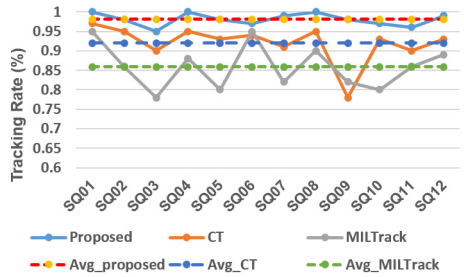

Fig. 11. Average percent tracking rates of CT [34] (Avg_CT), Miltrack (Avg_Miltrack) [37], and the proposed tracking method (Avg_Proposed), using the CCD stereo car dataset.

predicting their size using the U-V disparity information. The proposed method can handle an occlusion situation in which the detected vehicle is moving out of the image domain, as shown in Fig. 10(h); this is a situation in which the existing tracking methods might fail. Therefore, we employed a mechanism to track and validate the tracking obstacle within a given period. After detecting the obstacle's location at the t^{th} frame using our proposed tracking model, detected obstacles are validated by using the proposed classification method (see Section II.B), as shown in Algorithm 4. First, we generated a set of sample patches based on the position of an obstacle (i.e., the region of the interest; ROI) at the t^{th} frame. The classifier function [34] is then utilized with each sample to find the obstacle's location at $(t+1)^{th}$ frame. Next, the system updates the width and height of the tracking obstacle by using the U-V disparity. The classification process (II.B) is also periodically performed within a tracking interval T in order to remove false tracking. Finally, the classifier value is updated by using three features, color, LBP and adaptive U-V disparity. Fig. 11 shows the average percent success tracking rate of the proposed method (Avg_proposed) compared against the state-of-the-art tracking methods CT (Avg_CT) [13], and Miltrack (Avg_MILTtrack) [37] using the CCD stereo dataset with 3900 testing images in 11 scenarios [2]. The proposed tracking model yielded better results than CT and MILTTrack under various driving conditions. In these experimental results, the processing time of the proposed tracking model was approximately 8.5 ms/frame, demonstrating its suitability for real-time processing.

Algorithm 4 Proposed Tracking and Validating

input :
Location of a detected obstacle at $I^{(t)}$ frame, obstacle type
Left image ($I_L^{(t)}$), LBP results ($I_{LBP}^{(t)}$), adaptive U-V disparity image ($I_{AUV}^{(t)}$) at the t^{th} frame.
output: Location of the tracked obstacle at $I^{(t+1)}$ frame

begin
1. Determine a region of interest (ROI) at $I^{(t+1)}$ frame based on detected information from $I^{(t)}$ frame: $x^{(t)} - W^{(t} \leq x^{(t+1)} \leq x^{(t)} + 2W^{(t)}$ $y^{(t)} - H^{(t)} \leq y^{(t+1)} \leq y^{(t)} + 2H^{(t)}$
2. Generate a set of sample patches $S_{I_L^{(t+1)}}$, $S_{I_{LBP}^{(t+1)}}$, and $S_{I_{AUV}^{(t+1)}}$ of $I_L^{(t+1)}$, $I_{LBP}^{(t+1)}$ and $I_{AUV}^{(t+1)}$ within ROI, respectively.
3. Perform classifier function [34] on each sample and find the tracking location at $(t+1)^{th}$ frame with maximum classifier value.
4. Re-adjust the width and height of the tracking obstacle based on U-V disparity.
5. Check the tracking interval (T) to perform the validation process.
6. Perform classification process (II.B) on the tracked location at $(t+1)^{th}$ frame
7.**if** *detected type is obstacle type* **then**
    7.1 Generate two set of image patches, positive samples (near the current tracking position) and negative samples (far a way from tracking position).
    7.2 Calculate the feature value of two set image patches using integral technique with ratio function: $\omega S_{I_L^{(t+1)}} + \kappa S_{I_{LBP}^{(t+1)}} + \chi S_{I_{AUV}^{(t+1)}}$ $\omega + \kappa + \chi = 1$
    7.3 Update the classier function
end if
end

IV. Experimental Results and Discussion

A. Preparation of Unlabeled Training Data

Existing learning systems often apply a random selection technique to the original input image in order to generate a large unlabeled dataset. This approach might provide all possible training patches in every location in the original image, but many of these should not be included in the training process, such as patches of buildings and the sky. Therefore, we proposed a new approach by selecting unlabeled data for unsupervised training from possible obstacles that might appear along the road using our proposed obstacle detection method, as shown in Fig. 12. First, a disparity map is computed by using the stereo system [17]. Second, possible obstacles are identified by using the proposed obstacle detection method (see Section II.A). Finally, the unlabeled data is generated by mapping the obstacle results to the original left or right images.

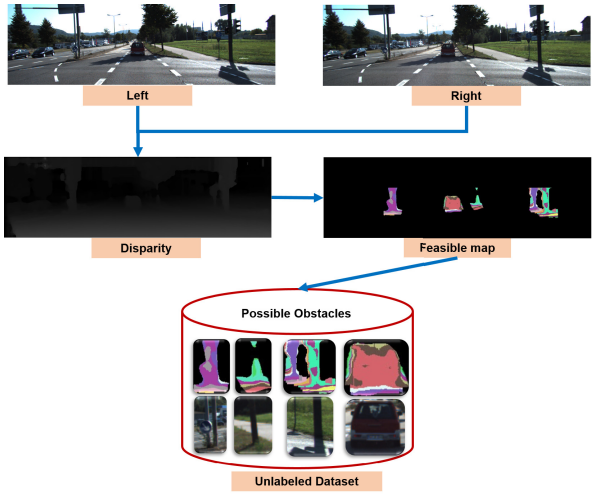

Fig. 12. Block diagram of the process used to automatically obtain an unlabeled data set by using the proposed model.

TABLE I
Parameter Settings of the Proposed Method

Parameter	Value	Parameter	Value
K-Means			
# filter	16	# iteration	150
Auto Encoder			
ρ	0.01	λ	0.00001
$\beta_{auto\_encoder}$	3	ϵ	0.01
TCNN			
ts	2	rf	1
n	6	# iteration	1
Soft-max			
δ	0.01	ϵ	10^{-6}

Thus, our proposed model can provide an unlabeled dataset that includes only feasible obstacles, which might help the un-supervised training algorithm learn abstract features of possible obstacle candidates more effectively.

B. System Configuration and Parameter Setting

To evaluate the processing time of the proposed system, we implemented the proposed method on a computer equipped with 8 Intel Xeon 3.5 GHz CPUs, 8GB of RAM, and a tesla K40 GPU accelerator to perform parallelization, as discussed previously. We compared the performance of the proposed method to various state-of-the-art algorithms: a feature-based stereo vision system [3], a motion-based stereo vision system [5], a stixel-based stereo vision [4], a genetic-based stereo vision system [2], and an RCNN [13]. Among these, the RCNN method performed most closely to the proposed method. Four datasets were investigated using each of these systems to evaluate their performance: the Daimler pedestrian dataset [35], the Karlsuhe Institute of Technology and Toyota Technology Institute (KITTI) Car dataset [36], the HCI dataset [23], and the CCD camera dataset [2]. The experimental parameter setting is shown in Table I.

C. Results

1) Daimler Pedestrian Dataset: For training, the Daimler dataset [35] is a pedestrian dataset including 15,560 pedestrian

TABLE II

ACCURACY (%) OF THE PROPOSED MODEL AND RCNN [13] USING VARIOUS DATASETS

Index	Training	Testing	RCNN [13]	The proposed
Daimler Pedestrian Dataset				
1.	2048	29072	93.21%	95.06%
2.	4096	29072	94.57%	96.29%
3.	6144	29072	95.14%	97.20%
4.	8192	29072	95.88%	97.49%
5.	10240	29072	96.15%	97.93%
KITTI Car Dataset				
1.	2048	24576	91.23 %	93.27%
2.	4096	24576	91.47%	93.99%
3.	6144	24576	94.08%	95.91%
4.	8192	24576	95.56%	97.16%
5.	10240	24576	96.25%	98.05%
CCD Stereo Dataset				
1.	2048	20480	92.20%	94.55%
2.	4096	20480	93.15%	95.17%
3.	6144	20480	94.61%	96.05%
4.	8192	20480	95.85%	97.46%
5.	10240	20480	96.14%	98.15%

samples (48x96 resolution) and 6,744 additional full images (640×480 resolution) not containing pedestrians for extracting negative samples. For testing, the Daimler dataset contains an independent sequence with more than 21,790 images with 56,492 pedestrian labels (fully visible or partially occluded). In addition, the Daimler pedestrian dataset is only a monocular image dataset. However, our proposed system can work well on both single and stereo images.

To evaluate the performance of the proposed method, for training, we randomly selected 10,240 samples from the Daimler training dataset (including 5120 positive samples from 15,560 pedestrian samples, and 5120 negative samples from 6,744 full negative images). For testing, we randomly selected 29,072 samples from the Daimler testing dataset. Thus, our selection approach is independent and random, and we do not control the selection of samples for training and testing. These testing and training samples do not overlap with each other. When 2,048 samples were used for training, the accuracy of the proposed method was 95.06% and that of the RCNN [13] was 93.21%, as shown in Table II. The accuracy of the proposed method increased when the size of the training data set was increased. With 10,240 training samples, the proposed method and the RCNN respectively yielded 97.93% and 96.15% accuracy. The proposed method performed much better than RCNN because of its ability to discover and learn meaningful features from multiple local patterns, along with its ability to use the tile-convolution neural network and an auto-encoder. To further evaluate the performance of the proposed method using an online-image, an image with a resolution of 640×480 was used; the proposed method performed better than the RCNN in all three test cases (Fig. 13). In this experiment, the RCNN produced more false alarms than the proposed method.

2) KITTI Car Dataset: Recently, the Karlsuhe Institute of Technology and Toyota Technological Institute (KITTI) [36] introduced a car dataset including 7,481 full training images (1242×375 resolution) and 7,518 full testing images

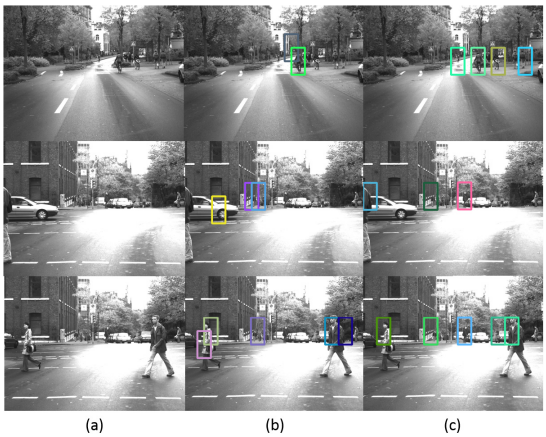

Fig. 13. Experimental results on the Daimler pedestrian dataset. (a) Input image. (b) Detected results from the RCNN [13]. (c) Detected results from the proposed method.

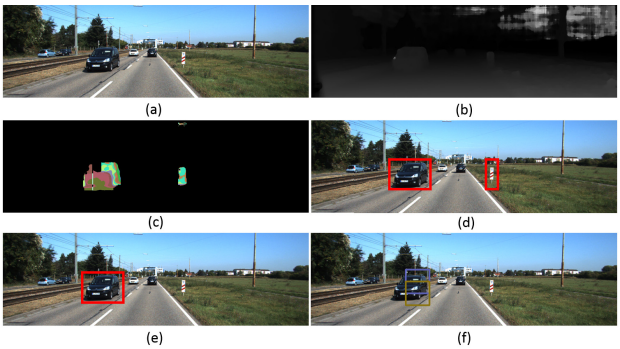

Fig. 14. Experimental results for the KITTI dataset. (a) Left image. (b) Disparity image [17]. (c) Results of the proposed obstacle detection method. (d) Mapping of the obstacle results to the left image. (e) Final results of the proposed classification method. (f) Results of RCNN [13].

(1242×375 resolution) under normal driving conditions, comprising a total of 80,256 labeled objects. First, we divided the original dataset into two sub-datasets: dataset A which contains 40,128 samples, and dataset B that contains 40,128 samples. We randomly selected 10,240 samples from dataset A for training, and 24,576 samples from dataset B for testing. Thus, the training and testing samples did not overlap with each other. The proposed method continued to perform better than RCNN during this test for each of the various training sizes, as shown in Table II. Fig. 14 shows the performance of the proposed method when using images with full resolution. The obstacle detection technique included in the proposed method provided accurate obstacle locations (Figs. 14(c)-(d)). Obstacle classification was then performed using the proposed method to accurately classify and validate the obstacles (Fig. 14 (e)). The RCNN produced many false alarms (Fig. 14 (f)).

To evaluate the performance of the proposed method in a more complicated situation, we conducted an another experiment, as shown in Fig. 15. In this experiment, the proposed method provided accurate obstacle locations (Figs. 15 (c)-(d)) and vehicle classifications (Fig. 15 (e)), whereas the RCNN failed by raising too many false alarms (Fig. 15 (f)).

TABLE III

AVERAGE DETECTION RATES(DR) AND FALSE ALARM RATES(FA) OF OUR PROPOSED ALGORITHM IN COMPARISON WITH FEATURE SYSTEMS [3], RCNN [13], STIXEL SYSTEM [4], MOTION SYSTEMS [5], AND SEA [2]

Seq.	Configuration	Illumination	# images	Feature system		RCNN[13]		Motion system		Stixel System		SEA		The proposed system	
				DR(%)	FA(%)	DR(%)	FA(%)	DR(%)	FA(%)	DR(%)	FA(%)	DR(%)	FA(%)	DR(%)	FA(%)
SQ01	b35,f16	Normal	110	92	8	98	0	93	3	95	1	100	0	100	0
SQ02	b60,f25	Normal	310	91	5	97	2	92	5	94	2	97	3	98	1
SQ03	**b60,f25**	**Low, changed**	**76**	**78**	**7**	**93**	**3**	**82**	**6**	**85**	**6**	**90**	**3**	**95**	**1**
SQ04	b60,f25	Normal	115	89	3	99	1	95	7	98	2	100	0	100	0
SQ05	b35,f25	Low	225	80	4	98	2	94	3	95	3	96	4	98	2
SQ06	**b60,f16**	**High, changed**	**85**	**82**	**8**	**94**	**2**	**85**	**4**	**88**	**4**	**91**	**4**	**96**	**2**
SQ07	b35,f16	Normal	197	88	4	98	2	96	2	98	2	98	2	98	1
SQ08	b35,f16	Normal	126	90	6	98	2	92	3	100	3	97	5	98	3
SQ09	**b35,f16**	**Low, changed**	**597**	**79**	**7**	**98**	**2**	**92**	**5**	**93**	**2**	**98**	**2**	**98**	**1**
SQ10	b35,f16	High	754	85	5	96	2	96	2	100	4	95	3	97	1
SQ11	b35,f16	High	800	87	11	95	2	92	6	98	5	94	4	97	2
SQ12	b35,f16	High	505	82	15	97	1	92	5	97	3	97	3	98	1
Average				**85.25**	**6.92**	**96.75**	**1.75**	**91.75**	**4.25**	**95.58**	**3.08**	**96.08**	**2.75**	**97.75**	**1.25**

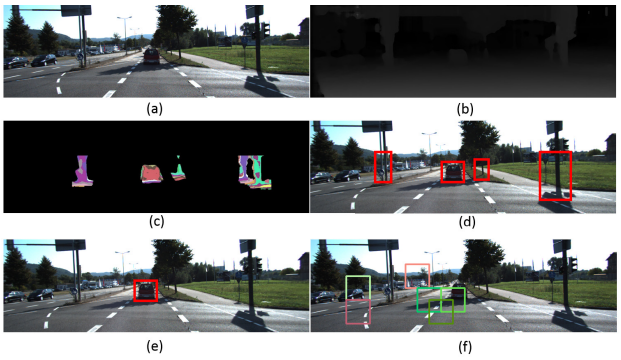

Fig. 15. Experimental results for the KITTI Dataset. (a) Left image. (b) Disparity image [17]. (c) Results of the proposed obstacle detection. (d) Mapping of the obstacle results to the left image. (e) Final results of the proposed classification. (f) Results of RCNN [13].

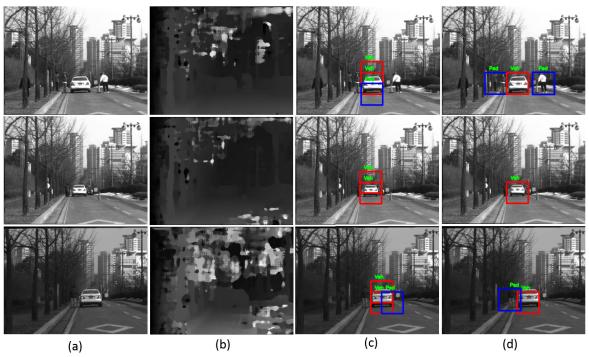

Fig. 16. Experimental results for the CCD dataset, where Veh denotes a vehicle, and Ped denotes a pedestrian. (a) Left image. (b) Disparity image [17]. (c) Results of the RCNN [13]. (d) Results of the proposed method.

3) Pedestrian and Car Based-CCD Camera Dataset: Nguyen *et al.* [2] have provided a CCD stereo image dataset using various base-lines and focal lengths. The dataset was acquired under various driving conditions including sunny, snowy, and cloudy weather conditions. However, this dataset provides only full-resolution images (320×240) originally intended for detecting vehicles using a genetic-based stereo vision algorithm [2]. Accordingly, in the present work, we created new training and testing samples from this dataset by labeling pedestrian and vehicle samples; in this way, we first labeled 50,100 samples from the CCD stereo dataset. The samples were then divided into subsets A and B. A contains 25,050 samples, and B contains 25,050 samples. We randomly selected 10,240 training samples from subset A, and 20,480 testing samples from subset B. The training and test samples are not overlapped with each other.This dataset was then used to compare the detection and classification results of the proposed system to those of the reference RCNN (Table II and Fig. 16). In this test, the proposed method accurately detected and classified pedestrians and vehicles, whereas the RCNN raised more false detections.

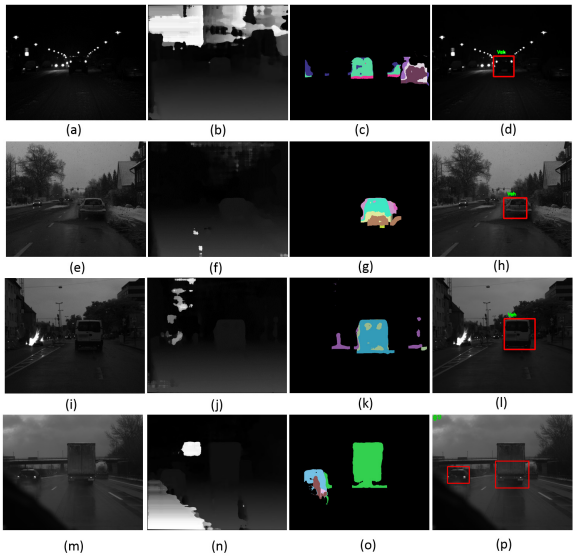

Fig. 17. Experimental results for the HCI dataset, where Veh denotes a vehicle, and Ped denotes a pedestrian. (a), (b), (c) and (d) are the left image, disparity image, proposed obstacle result, and proposed classification results for night and snow conditions, respectively. (e), (f), (g) and (h) are the left image, disparity image, proposed obstacle result, and proposed classification results for the flying snow condition, respectively. (i), (j), (k) and (l) are the left image, disparity image, proposed obstacle result, and proposed classification results for the rain flare condition, respectively. (m),(n),(o) and (p) are the left image, disparity image, proposed obstacle result, and proposed classification results for the wet Autobahn condition, respectively.

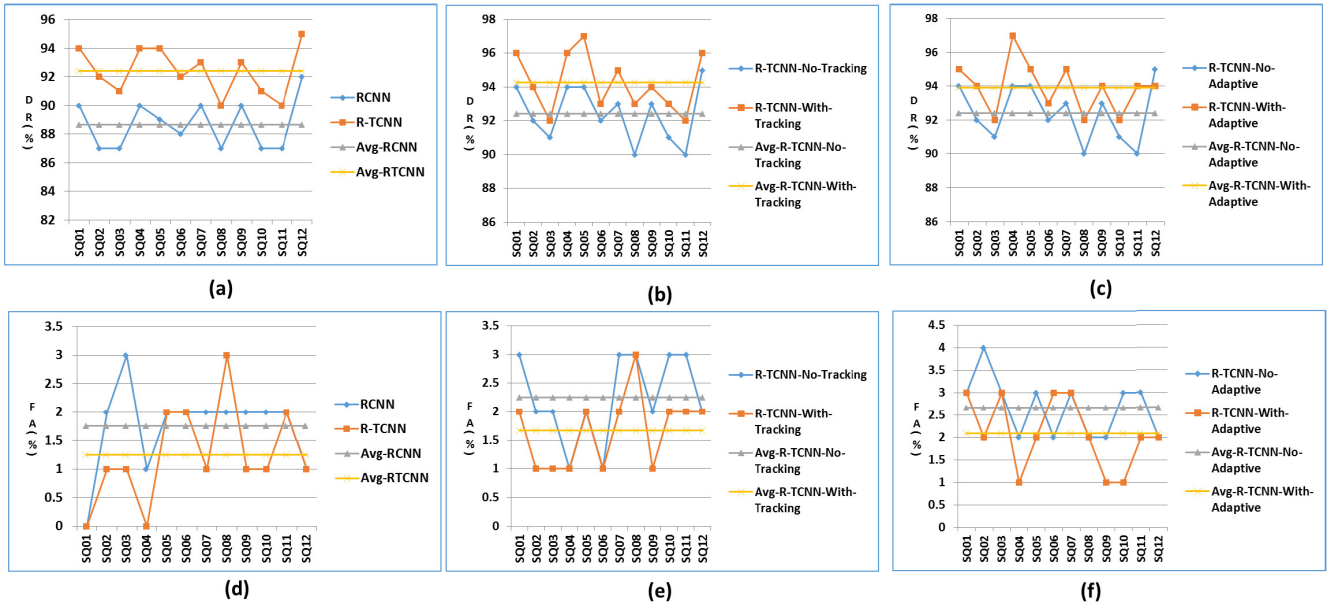

Fig. 18. Performance evaluation of the proposed system with and without a tracker and adaptive thresholds α, β. (a) and (d) are the DR and FA of the proposed system, respectively. (b) and (e) are the DR and FA of the proposed system with and without a tracker, respectively. (c) and (f) are the DR and FA of the proposed system with and without an adaptive threshold, respectively.

To further evaluate the performance of the proposed method compared to existing vehicle detection based on stereo vision methods, we conducted an experiment with 3,900 images, using various baselines, focal lengths, and driving conditions [2]. The proposed method performed the best in terms of detection rate (DR) (97.75%) and false alarm rate (FA) (1.25%) among the other approaches, as shown in Table III. Stereo vision and evolutionary (SEA) [2] yielded an average detection rate of 96.08% and an average false alarm rate of 2.75%. RCNN yielded an average detection rate of 96.74% and an average false alarm rate of 1.75%. The feature-based system [3] yielded an average detection rate of 85.25% and an average false alarm rate of 6.92%. The stixel-based system [4] yielded an average detection rate of 95.58% and an average false alarm rate of 3.08%. The motion-based system [5] yielded an average detection rate of 91.75% and an average false alarm rate of 4.25%.

4) HCI Dataset: We used the HCI dataset [23] to evaluate the performance of the proposed system under difficult driving conditions, such as those encountered in sequences with nighttime, snowy conditions, flying snow, rain flare, and wet Autobahn conditions. The HCI dataset contains 924 full images (656×541 resolution). We created new training and testing samples from the HCI dataset by labeling vehicle and non-vehicle samples. We labeled 2048 samples and 3072 samples for training and testing, respectively, from the HCI dataset. The training and test samples did not overlapped with each other. The proposed system obtained 96.25% accuracy, while RCNN obtained 89%. Fig. 17 shows the results of the proposed system when tested using these sequences. The proposed system accurately detected, classified, and tracked the preceding obstacles even under these difficult conditions. Table IV shows the comparison of RCNN and the proposed method in terms of DR and FA using the HCI dataset.

TABLE IV

AVERAGE DETECTION RATES (DR) AND FALSE ALARM RATES (FA) ON HCI DATASET

Seq.	Condition	# images	RCNN[13]		The proposed system	
			DR(%)	FA(%)	DR(%)	FA(%)
SQ01	Nighttime, snowy conditions	64	90	4	97	2
SQ02	Flying snow conditions	30	88	4	96	2
SQ03	Rain flare conditions	64	87	3	95	1
SQ04	Wet Autobahn conditions	30	91	2	97	1
Average			**89**	**3.25**	**96.25**	**1.5**

The proposed method obtained better performance than RCNN in this test. Overall, the robust experimental results proved that the proposed system works well under challenging road conditions.

5) Performance Evaluation With and Without the Proposed Approach: To evaluate the performance of the proposed system with and without a recursive tile convolution neural network (R-TCNN) (with no tracker and adaptive thresholds α and β), we conducted an experiment using the pedestrian and car based-CCD camera dataset [2]. Fig. 18 (a) and (d) show the performance of the proposed system R-TCNN and RCNN. The average detection rate (Avg) of R-TCNN and RCNN was 92.42% and 88.67 %, respectively. The average false alarm rate of R-TCNN and RCNN was 1.25 % and 1.75%, respectively. Thus, using the benefits of multiple local patterns and the tile convolution structure, the proposed R-TCNN obtained better performance than RCNN in terms of the detection rate (DR) and false alarm rate (FA). Hence, the proposed system improves the accuracy of RCNN from 88.67% to 92.42% by using the tile convolution mechanism.

To evaluate the performance of the proposed system with and without a tracker (with no adaptive thresholds α and β), we conducted experiments using the pedestrian and car

based-CCD camera dataset [2]. Fig. 18 (b) and (e) shows the performance of the proposed system with a tracker (R-TCNN-with-Tracking) and without a tracker (R-TCNN-No-Tracking). The average detection rate (Avg) of R-TCNN-with-tracking and R-TCNN-No-tracking was 94.25% and 92.42 %, respectively. The average false alarm rate of R-TCNN-with-tracking and R-TCNN-No-tracking was 1.67% and 2.25%, respectively. Thus, the accuracy of the proposed system was increased from 92.42% to 94.25% by employing a tracking mechanism.

To evaluate the performance of the proposed system with and without the adaption of α and β (no Tracker), we conducted experiments using the pedestrian and car based-CCD camera dataset [2]. Fig. 18(c) and (f) showss the performance of the proposed system with an adaption of α and β (R-TCNN-With-Adaptive) and without an adaption of α and β (R-TCNN-No-Adaptive). The average detection rate (Avg) of R-TCNN-Adaptive and R-TCNN-No-Adaptive was 93.92% and 92.42 %, respectively. The average false alarm rate of R-TCNN-With-Adaptive and R-TCNN-No-Adaptive was 2.08% and 2.25%, respectively. Thus, the accuracy of the proposed system was increased from 92.42% to 93.92% by using an adaption of α and β.

Overall, the tile convolution structure contributed the most to the proposed system when it increased the DR of RCNN from 88.67% to 92.42%, followed by a tracker that increased the accuracy from 92.42% to 94.25 %, and the adaption of α and β that increased the accuracy from 92.42% to 93.92%. Overall, the proposed system, by combining the benefits of the TCNN, a tracker, and the adaption of α and β, obtained the best performance in terms of DR 97.75 % and FA 1.25 %.

V. Conclusions

A general framework has been introduced for robust on-road pedestrian and vehicle detection, recognition, and tracking. The proposed framework combines the advantages of deep learning with those of using multiple local patterns and depth information, and can be operated efficiently. First, an efficient adaptive U-V disparity to detect feasible obstacles under various driving conditions was introduced. Second, a novel deep learning model was presented based on the use of a tile convolution neural network, an auto-encoder, multiple local patterns, and depth information to classify obstacles into vehicles and pedestrians. A simpler and robust tracking-based deep learning algorithm was developed to track detected obstacles in the next frame. Finally, the proposed method was parallelized by implementing it on GPU to enable its real-time use in DASs. By conducting various comprehensive experiments to evaluate performance, we demonstrated that the proposed framework yields a satisfactorily detection, recognition, and tracking for real-time driving assistance. Moreover, it is worth noting that the proposed frame could easy be extended to the detection and recognition of other obstacle types appearing along the road, such as traffic signs, traffic lights, and so forth. The present work was limited to the detection, recognition, and tracking of preceding obstacles. In future works, we plan to develop a full framework by including an obstacle behavior-based deep learning model for predicting the behavior and goals of detected obstacles on the road, including turning and acceleration. In addition, future research should transfer the current deep learning implementation to an embedded system (such as a Field-programmable gate array [FPGA])) mounted on a vehicle so it can be realized as a commercial product.

References

[1] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Trans. Intell. Transp. Sys.*, vol. 14, no. 4, pp. 1773–1795, Dec. 2013.

[2] V. D. Nguyen, T. T. Nguyen, D. D. Nguyen, S. J. Lee, and J. W. Jeon, "A fast evolutionary algorithm for real-time vehicle detection," *IEEE Trans. Veh. Tech.*, vol. 62, no. 6, pp. 2453–2468, Jul. 2013.

[3] G. Toulminet, M. Bertozzi, S. Mousset, A. Bensrhair, and A. Broggi, "Vehicle detection by means of stereo vision-based obstacles features extraction and monocular pattern analysis," *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2364–2375, Aug. 2006.

[4] F. Erbs, A. Barth, and U. Franke, "Moving vehicle detection by optimal segmentation of the dynamic stixel world," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2011, pp. 951–956.

[5] C. D. Pantilie and S. Nedevschi, "Real-time obstacle detection in complex scenarios using dense stereo vision and optical flow," in *Proc. 13th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2010, pp. 439–444.

[6] B. Southall, M. Bansal, and J. Eledath, "Real-time vehicle detection for highway driving," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recogn.*, Jun. 2009, pp. 541–548.

[7] C.-C. Chiu, M.-L. Chung, and W.-C. Chen, "Real–time front vehicle detection algorithm for an asynchronous binocular system," *J. Inf. Sci. Eng.*, vol. 26, pp. 735–752, May 2010.

[8] M. Nishigaki and Y. Aloimonos, "Moving obstacle detection using cameras for driver assistance system," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2010, pp. 805–812.

[9] M. Enzweiler and D. M. Gavrila, "A mixed generative-discriminative framework for pedestrian classification," in *Proc. IEEE Conf. CVPR*, Jun. 2008, pp. 1–8.

[10] S. Sivaraman and M. M. Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 267–276, Jun. 2010.

[11] C. Caraffi, T. Vojíř, J. Trefný, and J. Šochman, "A system for real-time detection and tracking of vehicles from a single car-mounted camera," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2012, pp. 975–982.

[12] C. Leistner, P. M. Roth, H. Grabner, H. Bischof, A. Starzacher, and B. Rinner, "Visual on-line learning in distributed camera networks," in *Proc. IEEE Int. Conf. Distrib. Smart Camera*, Sep. 2008, pp. 1–10.

[13] S. Richard, H. Brody, B. Bharath, D. M. Christopher, and Y. Andrew Ng, "Convolutional-recursive deep learning for 3d object classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 665–673.

[14] Q. V. Le, J. Ngiam, Z. Chen, D. Chia, P. W. Koh, and A. Y. Ng, "Tiled Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, NIPS, vol. 23, pp. 1279–1287, Jan. 2010.

[15] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Proc. Int. Conf. Mach. Learn.*, May 2013, pp. 8595–8598.

[16] Q. Yao, X. Liao, and H. Jin, "A map-reduce method for training autoencoders on xeon phi," in *Proc. IEEE Int. Conf. Comput. Inf. Technol. Ubiquitous Comput. Commun.*, Oct. 2015, pp. 1330–1337.

[17] V. D. Nguyen, D. D. Nguyen, S. J. Lee, and J. W. Jeon, "Local density encoding for robust stereo matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 12, pp. 2049–2062, Dec. 2014.

[18] A. Bensrhair, M. Bertozzi, A. Broggi, A. Fascioli, S. Mousset, and G. Toulminet, "Stereo vision-based feature extraction for vehicle detection," in *Proc. IEEE Intell. Vehicle Symp.*, Jun. 2002, pp. 465–470.

[19] C. G. Keller, M. Enzweiler, M. Rohrbach, D. F. Llorca, C. Schnorr, and D. M. Gavrila, "The benefits of dense stereo for pedestrian detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 1096–1106, Dec. 2011.

[20] R. Labayrade, D. Aubert, and J. P. Tarel, "Real time obstacle detection in stereovision on non flat road geometry through 'v-disparity' representation," *IEEE Intell. Vehicle Symp.*, vol. 2. Jun. 2002, pp. 646–652.

[21] M. Zhang, P. Liu, X. Zhao, X. Zhao, and Y. Zhang, "An obstacle detection algorithm based on U-V disparity map analysis," in *Proc. IEEE Int. Conf. ICITIS*, Dec. 2010, pp. 763–776.

[22] Z. Hu, F. Lamosa, and K. Uchimura, "A complete U-V-disparity study for stereovision based 3D driving environment analysis," in *Proc. Int. Conf. DIM*, Jun. 2005, pp. 204–211.
[23] S. Meister, B. Jähne, and D. Kondermann, "Outdoor stereo camera system for the generation of real-world benchmark data sets," *Opt. Eng.*, vol. 51, no. 2, p. 021107, Mar. 2012.
[24] R. Klette *et al.*, "Performance of correspondence algorithms in vision-based driver assistance using an online image sequence database," *IEEE Trans. Veh. Technol.*, vol. 60, no. 5, pp. 2012–2026, Jun. 2011.
[25] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.
[26] Z. Guo, L. Zhang, and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1657–1663, Jun. 2010.
[27] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," in *Analysis and Modeling of Faces and Gestures*, vol. 4778. Berlin, Germany: Springer, 2007, pp. 168–182.
[28] S. Murala, R. P. Maheshwari, and R. Balasubramanian, "Local tetra patterns: A new feature descriptor for content-based image retrieval," *IEEE Trans. Image Process.*, vol. 21, no. 5, pp. 2874–2886, May 2012.
[29] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," *Proc. J. Mach. Learn. Res. Track, AISTATS*, Jan. 2011, pp. 215–223.
[30] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Adv. Neural Inf. Process. Syst.*, 2007, pp. 153–160.
[31] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[32] R. Danescu, F. Oniga, S. Nedevschi, and M.-M. Meinecke, "Tracking multiple objects using particle filters and digital elevation maps," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2009, p. 88–93.
[33] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, p. 1131–1342, Dec. 2011.
[34] K. Zhang, L. Zhang, and M. H. Yang, "Real-time compressive tracking," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2012, pp. 864–877.
[35] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2179–2195, Dec. 2009.
[36] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recogn*, Jun. 2012, pp. 3354–3361.
[37] S. Babenko, M. H. Yang, and S. Belongie, "Robust object tracking with online multiple instance learning," in *Proc. PAMI*, 2011, pp. 1632–1649.
[38] K. L. Priddy and P. E. Keller, *Artificial Neural Networks: An Introduction*. Bellingham, WA, USA: SPIE, 2005, pp. 16–17.

Vinh Dinh Nguyen received the B.S. (*magna cum laude*) degree in computer science from Nong Lam University, Ho Chi Minh City, Vietnam, in 2007, and the M.S. and Ph.D. degrees in electrical and computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2012 and 2015, respectively. Since 2015, he has been with Sungkyunkwan University as a Researcher. His research interests include computer vision, image processing, and graphics processing unit computing.

Hau Van Nguyen received the B.S. degree in electronic physics from University of Science, Vietnam National University, Ho Chi Minh City, in 2006. He is currently working toward the Ph.D. degree with the School of Information and Communication Engineering, Sungkyunkwan University. His research interests include computer vision, image processing, and graphics processing unit computing.

Dinh Thi Tran received the B.S. degree in computer science from Nong Lam University, Ho Chi Minh City, Vietnam, in 2013. She is currently working toward the Ph.D. degree with the School of Information and Communication Engineering, Sungkyunkwan University. Her research interests include computer vision, image processing, and graphics processing unit computing.

Sang Jun Lee received the B.S. and M.S. degrees in electrical and computer engineering from Sungkyunkwan University, Suwon, South Korea, in 2007 and 2009, respectively, where he is currently working toward the Ph.D. degree with the School of Information and Communication Engineering.

His research interests include image/speech signal processing, embedded systems, and real-time applications.

Jae Wook Jeon (S'82–M'84) received the B.S. and M.S. degrees in electronics engineering from Seoul National University, Seoul, South Korea, in 1984 and 1986, respectively, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, in 1990. From 1990 to 1994, he was a Senior Researcher with Samsung Electronics, Suwon, South Korea. Since 1994, he has been with the School of Information and Computer Engineering, Sungkyunkwan University, as an Assistant Professor, where he is currently a Professor. His research interests include robotics, embedded systems, and factory automation.