

Real-time raindrop detection based on cellular neural networks for ADAS

Fadi Al Machot¹ · Mouhannad Ali¹ · Ahmad Haj Mosa¹ · Christopher Schwarzmüller¹ · Markus Gutmann¹ · Kyandoghere Kyamakya¹

Received: 17 August 2014 / Accepted: 10 February 2016 / Published online: 26 March 2016
© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract A core aspect of advanced driver assistance systems (ADAS) is to support the driver with information about the current environmental situation of the vehicle. Bad weather conditions such as rain might occlude regions of the windshield or a camera lens and therefore affect the visual perception. Hence, the automated detection of raindrops has a significant importance for video-based ADAS. The detection of raindrops is highly time critical since video pre-processing stages are required to improve the image quality and to provide their results in real-time. This paper presents an approach for real-time raindrops detection which is based on cellular neural networks (CNN) and support vector machines (SVM). The major idea is to prove the possibility of transforming the support vectors into CNN templates. The advantage of CNN is its ultra fast pre-processing on embedded platforms such as FPGAs and GPUs. The proposed approach is capable to detect raindrops that might negatively affect the vision of

the driver. Different classification features were extracted to evaluate and compare the performance between the proposed approach and other approaches.

1 Introduction

Video-based ADAS are expected to assist drivers in different weather conditions such as sunny weather conditions, rainy weather or even during heavy snow. In general, such systems work appropriately in good weather conditions whereas bad weather usually represents a big challenge for their applicability since influences such as rain and snow do lead to a significant degradation of accuracy and precision or even yield the full loss of practicability of the systems. One of the most common interfering effects leading to a falsified or even inoperative system are raindrops on a vehicle's windshield which occur during rainy or snowy conditions. These adherent raindrops occlude and deform some image areas. For example, raindrops will decrease the performance of clear path detection [38] by adding blurred areas to the image. Therefore, robust raindrop detection and corresponding noise removal algorithms can improve the performance of vision-based ADAS in rainy conditions. We believe that such algorithms should be integrated in all the modules that are responsible for removing noise in visual data or the handling of similar anomalies. In general, such modules are responsible for pre-processing the visual data before it is further analyzed and are therefore commonly referred to as “video pre-processors”.

Several approaches have been proposed throughout literature to deal with raindrop detection problems. However, most of these do not consider the following system requirements: high detection rate, real-time constraint and robustness under dynamic environments.

✉ Fadi Al Machot
fadi.almachot@aau.at

Mouhannad Ali
Mouhannad.Ali@aau.at

Ahmad Haj Mosa
Ahmad.HajMosa@aau.at

Christopher Schwarzmüller
Christopher.Schwarzlmuller@aau.at

Markus Gutmann
Markus.Gutmann@aau.at

Kyandoghere Kyamakya
Kyandoghere.Kyamakya@aau.at

¹ Institute for Smart System Technologies, Transportation Informatics, Alpen-Adria Universitaet Klagenfurt, Universitätsstrasse 65-67, 9020 Klagenfurt, Austria

The raindrop detection process requires the classification at pixel level. Therefore, it is computationally very expensive using high-resolution images. This paper suggests a real-time system architecture for object detection and classification since there is no clear definition of real-time raindrop detection in the literature [9, 32]. Cellular neural networks are powerful platforms for parallel computing and support vector machines are robust classifiers for object recognition. Our proposed CNN modification permits the combination between both approaches for a robust real-time object recognition. Our methodology is to illustrate a comprehensive modification of the support vectors that are given after SVM training using radial basis function kernel (RBF) to transform them into CNN control templates. Thus, we accelerate the SVM performance by significantly decreasing the classification time at pixel level.

This paper is organized as follows: In Sect. 2, we give an overview of the state of the art regarding raindrop detection and removal in images. Sections 3 and 4 provide an overview about CNN and SVM, respectively. Section 5 introduces an overall architecture of our proposed system. Section 6 describes the feature extraction for raindrops detection. The performance of our approach and a benchmarking evaluation are then presented in Sect. 7. Finally, the concluding remarks and an outlook are given in Sect. 8.

2 Related works

Several approaches have been proposed for detecting raindrops on a windshield and for generating the CNN templates. Regarding rain drop detection on a windshield [21], used raindrop templates known as eigendrop to detect raindrops on windshields. Related results show that the detection within the sky area is quite promising, but the named method produces a large number of false positive within non-sky regions of the image where raindrop appearance and background texture become less uniform.

Yamashita et al. [41] Proposed two methods of detecting and removing raindrops from videos taken by different cameras. One of these methods is based on image sequences captured with a moving camera. The second is based on a stereo measurement and utilizes disparities between stereo image pairs [40]. All these methods required an additional camera. Therefore, their solution is not the best for the practice.

Halimeh and Roser [15] Developed a geometric-photometric model to detect and remove raindrops from images taken behind a car's windshield. This approach performs well but requires a high computational time and does not consider that raindrops appear blurred when they are not in the scene focus of the camera. Another geometric-

photometric model proposed for a raindrop detection [32] was based on an approximation of a raindrop shape as a spheroid section. This model performs well but still needs too much computational time.

Roser and Geiger [26] Attempted to model the shape of adherent raindrops by a sphere crown and later, by Bezier curves [27]. However, these models are insufficient, since a sphere crown and Bezier curves can cover only a small portion of possible raindrop shapes.

Garg and Nayar [13] Studied the image appearance changes due to rain and proposed a photometric raindrop model that characterizes refraction and reflection of light through a stationary, spherical raindrop. However, this approach requires a stationary observer or a long exposure period, which are not appropriate for automotive driving applications with a moving camera to capture dynamic scenes.

You et al. [43] Proposed a method for raindrop detection based on the motion and the intensity temporal derivatives of the input video. The authors assume that the motion of no-raindrop pixels is faster than that of raindrop ones and the temporal change of intensity of non-raindrop pixels is larger than that of raindrop pixels. This method has a high detection rate. Unfortunately, it does not work with highly dynamic raindrops.

A slightly different approach was proposed by [39] who analyzed the texture, the shape and the color characteristics of raindrops in images to detect raindrops on a windshield. Despite of providing meaningful results when applied in low to medium rain scenarios the approach is not performing well in heavy rain scenarios. Another approach by [42] covers large obvious raindrops as well as those raindrops at the corners of the view and the trails. The approach facilitates the Hough transformation algorithm to detect raindrops in the context of a lane scene as well as the Sobel operator to detect raindrops in the context of a building scene. However, the approach is not applicable in a real-time scenarios since it requires the interaction of a human operator who processes the images.

Furthermore, another approach comes from [24] who presented an unprecedented approach to detect unfocused raindrops in spherical deformations which are the cause for a blurred vision through the windshield. This approach does not require any additional devices. Since the approach is in its experimental stage it lacks a completely proven and tested body of methods and concepts. Also the concept to measure the performance is not fully developed and needs further attention.

Regarding the generation of CNN templates, the goal is to find the optimal values of template elements that increases the overall performance of the CNN. During the last three decades, different methodologies have been proposed to generate CNN templates. For example, [20]

used a genetic algorithm for template learning. Others, [23], developed a learning algorithm based on back-propagation. Their convergence might be accepted but their system is not reconfigurable. Moreover [3], proposed an approach based on Simulated Annealing where it is difficult to find heuristically the desired template. Additionally [31], developed a method based on particle swarm optimization (PSO) which has the tradeoff problem between the balance of convergence speed and convergence accuracy. Also [37], combined a genetic algorithm and particle swarm optimization to generate CNN templates. However, that approach does not support classification templates. In this paper, we were inspired by the idea of [24] and we did improve it by providing a novel approach which is based on Cellular Neural Networks (CNN) to accelerate the Support Vector Machine (SVM) classification.

3 Cellular neural networks

A cellular neural network (CNN) is a system of fixed-numbered, fixed-location, locally interconnected, fixed-topology, multiple-input, multiple-output, nonlinear processing units [7]. The concept of CNN combines the architecture of artificial neural networks and that of cellular automata. In contrast to ordinary neural networks, CNN has the property of local connectivity [6]. In the context of neural networks, neural computing nonlinear processing units are often referred to as neurons or cells due to their analogy with their organic counterparts within the brain of an organism. Equation (1) [which is an ordinary differential equation (ODE)] represents the Chua-Yang model for the state equation of a cell [10].

$$\frac{dx_{i,j}(t)}{dt} = -x_{i,j}(t) + \sum_{k=-1}^1 \sum_{l=-1}^1 A_{k,l}y_{i+k,j+l} + \sum_{k=-1}^1 \sum_{l=-1}^1 B_{k,l}u_{i+k,j+l} + I \tag{1}$$

The variable $x_{i,j}(t)$ describes the actual internal state of the cell $C_{i,j}$. In addition to that, $y_{i+k,j+l}(t)$ represents the output of a cell and $u_{i+k,j+l}$ indicates the input. The input is constant over time, because in image processing the original image should be processed. The output of a cell is a variable over time, because it depends on its actual state. The coupling between cells is the feedback of another cell in the template matrices (A , B). Matrix A is denoted as feedback-template and matrix B as control template. Furthermore, every cell of the network is using the same templates. The variable I represents the bias value, which is the inaccuracy of a cell. The neighborhood of a cell $C_{i,j}$ is defined by Eq. (2).

$$N_{i,j}(r) = \{C_{k,l} : \max(|k - i|, |l - j|) \leq r, \text{ where } 1 \leq k \leq V, 1 \leq l \leq W\} \tag{2}$$

In the most common case $r = 1$, which means that the neighborhood of cell $C_{i,j}$ only consists of its direct neighbors (in this case each cell, except the border cells, has 8 neighbors). $V \times W$ is the size of the CNN array (commonly a two-dimensional array) we presume that V is equal to W . The border cells must be treated differently (border conditions) according to [5] because they do not have the same number of neighbors as the inner cells. The output of a cell is generated by a nonlinear function (in this case the Chua’s equation) as shown in Eq. (3):

$$y_{i,j}(t) = \text{nlf}(x_{i,j}(t)) = \frac{1}{2} (|x_{i,j}(t) + 1| - |x_{i,j}(t) - 1|) \tag{3}$$

This nonlinear function (nlf) maps the actual state of a cell $x_{i,j}(t)$ to a value within the interval $[-1, +1]$. In grayscale image processing $+1$ represents the color white and -1 represents the color black. The values in-between the maximum and minimum represent the grayscale spectrum.

4 Support vector machine (SVM)

A support vector machine (SVM) requires data for training to establish a model for classifying raindrops. The SVM is a classifier which separates a set of objects in classes such that the distance between the class borders is as large as possible [36]. The set of training data is defined by Eq. (4).

$$\{(x_1, y_1), \dots, (x_n, y_n) | x_i \in X, y_i \in \{-1, 1\}\} \tag{4}$$

where x_i represents a feature, y_i presents the class of the feature (here: -1 represents a non-raindrop, $+1$ represents a raindrop), X is the set of training data (with n elements), and $1 \leq i \leq n$. The idea of SVM now is to separate both classes with a hyperplane that the minimal distance between elements of both classes and the hyperplane is maximal. The hyperplane is given through a normal vector w and a bias value b which can be found in Eq. (5).

$$y_i(\langle w, x_i \rangle + b) \geq 1 \tag{5}$$

where $\langle \cdot \rangle$ is the scalar product. In general, training data are not always linearly separable due to noise or the classes distribution. Therefore, it is not possible to separate both classes with a single linear hyperplane. Due to this fact, the algorithm for SVM training is formulated as dual problem. This formulation is equivalent to the primal problem in terms that all solutions of the dual problem are solutions of the primal problem. The conversion is realized through the representation of the normal vector w which is defined as linear combination of examples of the training set [see Eq. (6)]:

$$w = \sum_{i=1}^m \alpha_i y_i x_i \quad (6)$$

where α_i is the lagrange multiplier coefficient and x_i is a support vector.

The derivation of the dual form can be found in [36] and the resulting classification function can be found in Eq. (7):

$$f(x) = \text{sgn} \left(\sum_{i=1}^m \alpha_i y_i K(x_i, x) + b \right) \quad (7)$$

where m is the number of support vectors, α_i is the lagrange multiplier coefficient, x_i is a support vector, x is a part of the original image (which is compared), $K(\cdot)$ is the kernel function, y_i is the label (to distinguish between raindrops and non-raindrops) and b is the bias.

The kernel function is needed because of the nonlinearity of the training data which transforms the data into a higher dimension. To reduce the computational cost, positive definite kernel functions are used, e.g., polynomial kernels and radial basis function (RBF). In the next section, more details regarding our kernel function will be presented.

5 The modification of CNN using RBF-SVM

In this section, our aim is first to prove why using support vectors as template is correct, when it is correct and what conditions or assumptions are made to use the support vectors as templates.

5.1 Using support vectors as CNN control templates

In CNN-based image processing, there are two approaches to design the CNN templates. The first approach uses optimization-based supervised learning techniques with input and output samples [3]. In the second approach, the control templates represent a predefined image-processing template (e.g., Sobel template or an averaging smoothing template), while the feedback templates are all zeros except the center element which represents the cell self-feedback. The goal of the second approach is to accelerate the related image-processing task (using its defined kernel as a CNN control template) via a CNN processor, which, due to its parallel paradigm, can operate much faster [5]. The latest mentioned predefined kernels have been suggested in Refs. [3, 5] for various pixel level image processing operations.

Furthermore, in the field of signal/image processing, the term Kernel is a function that gives the similarity or the correlation between its two inputs. For example, in the Sobel edge-detection technique, the convolution between the image space and the Sobel vertical or horizontal

template highlights the image regions that have a similar structure to the Sobel template. Like in edge detection, the kernel used in the SVM-based visual object detection also looks for the similarity between the features and the trained support vectors, using the convolution methods as well. This fact inspired us to use the SVM kernel as a CNN control template.

Thus, the aim of this combination is to take the advantage of the high detection rate of SVM and the high speed of CNN, which is realized using the pertained SVM kernel as a CNN control template.

The construction of the CNN templates starts from the SVM decision function which can be found in Eq. (7) and ends with the modified CNN using Eq. (1). As we have mentioned, our kernel function is the RBF kernel (see Eq. 15), we take the euclidian norm for the norm function, to be able to rewrite the equation.

The constant γ in the RBF-kernel function must fulfill the condition $\gamma > 0$.

The default value of γ is $\frac{1}{\text{Number of features}}$. The number of features is determined during SVM training in the off-line phase, see Sect. 6.2.

$$\begin{aligned} f(x) &= \text{sgn} \left(\sum_{i=1}^m y_i \alpha_i K(x_i, x) + b \right) \\ &= \text{sgn} \left(\sum_{i=1}^m y_i \alpha_i e^{-\gamma(x_i - x, x_i - x)} + b \right) \\ &= \text{sgn} \left(\sum_{i=1}^m y_i \alpha_i e^{-\gamma((x_i, x_i) - 2(x_i, x) + (x, x))} + b \right) \\ &= \text{sgn} \left(e^{-\gamma(x, x)} \sum_{i=1}^m y_i \alpha_i e^{-\gamma(x_i, x_i)} e^{2\gamma(x_i, x)} + b \right). \end{aligned} \quad (8)$$

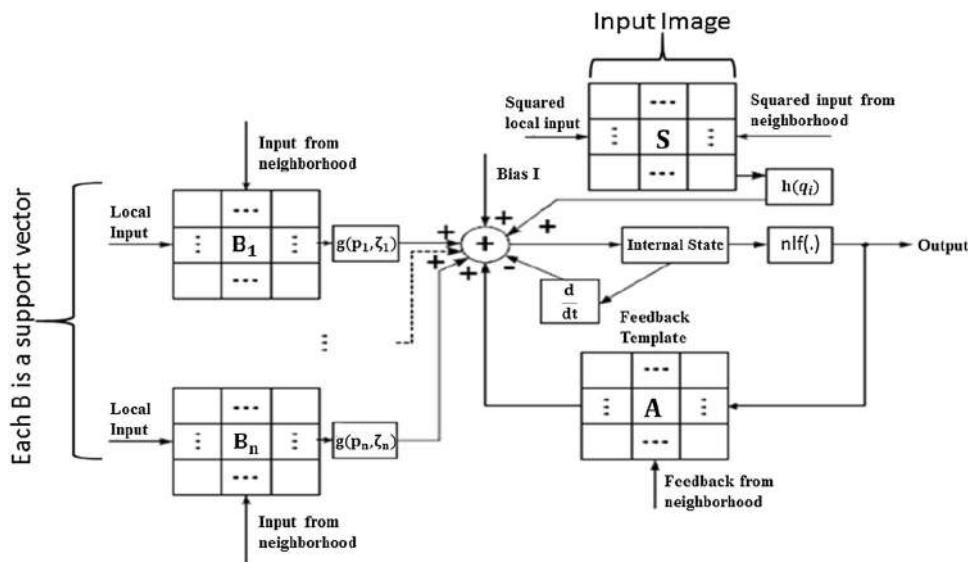
Equation (8) illustrates how to rewrite the decision function given in Eq. (7). Since most of the values are known after training the SVM, we are able to substitute $y_i \alpha_i e^{-\gamma(x_i, x_i)}$ with ζ_i which leads to Eq. (9):

$$f(x) = \text{sgn} \left(e^{-\gamma(x, x)} \sum_{i=1}^m \zeta_i e^{2\gamma(x_i, x)} + b \right). \quad (9)$$

Additionally, the templates of the CNN equation are then created according to Eq. (9) and the resulting support vectors.

After the substitution in the general CNN, Eq. (1), the sgn -function can be omitted, since by definition, the CNN output is within the interval of $[-1, 1]$. The final modified CNN state equation is presented in Eq. (10) which has m control templates. The size of the templates results from the length of the support vectors of the model. The best template size is determined empirically over various real raindrops images and found to be 15×15 . This selection

Fig. 1 The representation of the proposed CNN architecture using Eq. (10) and the support vectors for real-time raindrops detection



depends on the best correlation between the average rain-drop size and the template size.

$$\frac{dx_{i,j}(t)}{dt} = -x_{i,j}(t) + A * nlf(x_{i,j}(t)) + \sum_{i=1}^m g(B_i * u, \zeta_i) + h(S * u^2) + I \tag{10}$$

where $I = 0$.

The notation in Eq. (10) is identical to Eq. (1), except that the symbol “*” is the convolution operator. Figure 1 shows the final CNN block diagram where the index at the control templates B represents the number of support vectors, S is a matrix with the same size as the input image u , and I is the bias value of the CNN state equation. The functions $g(\cdot)$ and $h(\cdot)$ are defined in Eq. (11), and the initial value of S is the Input image pixels.

$$\begin{aligned} S &= (s_{ij})_{i=0\dots14,j=0\dots14} = 1 \\ g(p_i, \zeta_i) &= \exp(2 \cdot \gamma \cdot p_i) \cdot \zeta_i \quad \text{where } p_i = B_i * u \\ h(q_i) &= \exp((-\gamma) \cdot q_i) \cdot b \quad \text{where } q_i = S * u^2 \\ A &= (a_{ij})_{i=0\dots14,j=0\dots14} = \begin{cases} 1, & i = j = 7 \\ 0, & \text{else.} \end{cases} \end{aligned} \tag{11}$$

According to the previous discussion, considering the CNN state Eq. in (1) compared to Eq. (10), the third term in the right side of Eq. (1) does represent the control unit of CNN. This unit does a matrix dot product (replaced by a convolution of a whole image) between the control templates and the corresponding pixels. This operation is considered as a linear mapping of the CNN input. In many cases (i.e., edge detection), the linear mapping is sufficient to do the task (highlight the edges). However, in many other cases, the task is sophisticated, which requests a nonlinear mapping of the inputs.

On the other side, as is mentioned before, the RBF-SVM is also linear mapping. However, before the linear mapping,

RBF kernel transforms the input and the support vector into a higher dimension to be linearly separable. Hence, in this paper, we introduce the RBF kernel as CNN control unit which is more capable of dealing with high nonlinear problems, which is explained mathematically in the following.

The evidence of this transformation is supported by the following assumption, as long as the output function in Eq. (3) is $o_{i,j} \in \{-1, +1\}$, the previous assumption leads to Eq. (12):

$$\begin{aligned} o_{i,j}(t) &= nlf(-x_{i,j}(t) + (A * nlf(x_{i,j}(t))) \\ &+ \sum_{i=1}^m g(B_i * u, \zeta_i) + h(S * u^2) + I) \end{aligned} \tag{12}$$

where $I = 0$.

Thus, Eq. (13) holds the following inequality true:

$$\begin{aligned} o_{i,j}(t)(-x_{i,j}(t) + (A * nlf(x_{i,j}(t))) \\ + \sum_{i=1}^m g(B_i * u, \zeta_i) + h(S * u^2) + I) > 1 \end{aligned} \tag{13}$$

where $I = 0$.

Recalling Eqs. (5) and (13), it is evident that both are considered to be the same form.

5.2 The requirements of using support vectors as CNN templates

In our case, the use of our CNN in the field of image processing has to consider the single image pixel neighborhoods. Hence, the size and geometry of the CNN should be equal to the size of the “input image”.

Additionally, since the output of the CNN cells is either +1 or -1, the resulting pixels of the output image would be the same. Therefore, in real scenarios, the lowest value refers to the black color and the highest value refers to the

white color. To achieve that the pixel values have to be in grayscale and normalized to the interval $[0,1]$. Consequently, the result of the classification helps to allocate the raindrop pixels that can be marked easily in the original color image.

5.3 The use cases for using support vectors as CNN templates

In image processing, the pixels of an image can usually be divided into several distinct classes of similar pixels. For example, in outdoors scenarios, pixels could be divided into pixels belonging to a car or to the background, e.g., asphalt.

Consequently, if we have a set of training images in which the pixels have been assigned to a specific class label, we can extract image features that can distinguish between pixels of different classes.

Therefore, we believe that in any machine vision scenario where possible features, e.g., pixel intensity, edge strength or color based features, our proposed approach can be used to classify these classes in real-time. Thus, pixels features in a new image can be compared with the distribution of features in the training set and classified if they belong to a certain class. In other words, our system fits well to pixel-based image classification problems.

6 General description of the proposed approach

The overall architecture of the proposed approach consists of two phases: (a) an off-line phase which is the training of the support vector machine to construct a

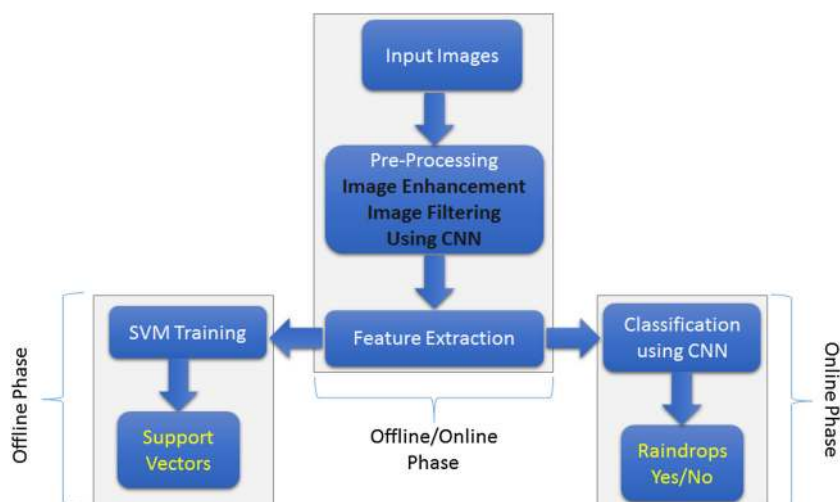
model for classifying raindrops; this phase is relevant only for the initialization of the system; (b) an online phase in which the real-time raindrop detection is performed.

A pre-processing step is required for both the online and off-line phases. It consists of image enhancement and image filtering. Finally, the classification of raindrops in real-time is applied (online) using a modified cellular neural network after the transformation of the resulting support vectors from the off-line phase into CNN templates, see Fig 2.

6.1 Pre-processing Phase

In computer vision systems, robust object detection requires a pre-processing step. The two major processing steps for raindrop detection are image enhancement and image filtering. The image enhancement is required to improve the contrast. The image filtering is used to remove weather noise. In our system, we use a partial differential equation (PDE)-based enhancement technique which improves the contrast and reduces the noise [12] at the same time. The CNN template for this PDE-based approach can be found in Eq. (14). The coefficient $\lambda \in R$ describes the ratio between smoothness and contrast enhancement. Values of $\lambda > 1$ are dominated by contrast enhancement while everything else is governed by smoothness. We already proved and compared the effectiveness of this algorithm in [30]. For this algorithm, the standard CNN state equation [6] is used. Additionally, the median filter based on CNN is applied to remove the rain dots from the image [17].

Fig. 2 The overall architecture of the real-time raindrops detection system



$$A = \begin{pmatrix} 0 & 0.25 & 0 \\ 0.25 & 0 & 0.25 \\ 0 & 0.25 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 & -\lambda & 0 \\ -\lambda & 4\lambda & -\lambda \\ 0 & -\lambda & 0 \end{pmatrix},$$

$$I = 0. \tag{14}$$

6.2 Off-line phase: SVM training

As an initial step, a set of images is captured while being in a rainy weather situation. These images are supposed to be of high quality with a resolution of 510×585 pixels. One raindrop is extracted as described in the following. We extract a complete quadratic bounding box, which surrounds the raindrop and stores it in a separate file. Additionally, non-raindrop images need to be created to establish a set of negative classes for training the SVM classifier. Diverse features are extracted from the raindrops with different natural highway backgrounds (see Sect. 4). The extracted features in the previous step are used to train the support vector machine [4, 35].

After testing different kernels for transforming the feature space [e.g. polynomial kernel, radial basis function (RBF) kernel and linear kernel], the RBF kernel [see Eq. (15)] provided by [29] was chosen because of its high performance—this kernel did provide the best results in our tests (compared to polynomial and the linear ones). Equation (15) shows the RBF kernel, x and x_i are represented as feature vectors. The RBF kernel plays the role of the dot product in the feature space. This technique avoids the computational impact of explicitly representing the feature vectors in the new feature space. The constant γ in the RBF-kernel function must fulfill the condition $\gamma > 0$. LIBSVM [4] library is used for the training of SVM.

$$K(x_i, x) = \exp(-\gamma \cdot \|x_i - x\|^2) \tag{15}$$



Fig. 3 The original input image

6.3 Online phase: real-time raindrop classification

In the online phase, after the learning process completed, the CNN templates can be modified using the resulted support vectors as we have discussed in Sect. 5.1.

Additionally, the proposed features in Sect. 7 are extracted from the given frame (image). Later on, the given frame will be used as input to the proposed CNN in Fig. 1. The result of this phase is a binary image which segments the raindrops as foreground and other discarded pixels as background. Moreover, the indices of the resulted raindrops can be used for the reconstruction of the original colored image (Fig. 3).

7 Raindrop detection: feature extraction

This section provides a brief overview about the detection of raindrops. Furthermore, we illustrate the extraction of those features that are used to train and test the system. The features are: edges of raindrops, color of raindrops, the histograms of oriented gradients and wavelets.

7.1 Using edge features for SVM

To extract edge features from images, the contrast enhancement and edge filtering techniques are applied. The resulting binary images consist of circular edges of raindrops and other edges. Those edges that do not belong to a raindrop must not be interpreted as raindrop features. Therefore, the oval shape of a raindrop is important. The center points of the raindrops are computed using the equations of contour moments [19] and the distances between the center and the borders of the raindrops are used as features for training the SVM (see Fig. 4).

After enhancing the image contrast (as discussed in pre-processing, see Sect. 6.1), a grayscale edge detector template [see Eq. (16)] [6] based on CNN is applied to find edges within all raindrops [18].

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} -0.11 & 0 & 0.11 \\ -0.28 & 0 & 0.28 \\ -0.11 & 0 & 0.11 \end{pmatrix}, I = 0. \tag{16}$$

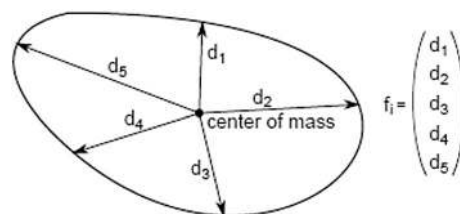


Fig. 4 Extraction of edge features for SVM

After extracting the edges, the center points of closed contours are calculated using the concept of contour moments. In this context moments are the weighted averages of the values of each pixel of an image. The definition of moments regarding the gray value function $g(x, y)$ of an object can be seen in Eq. (17) where p and q are the corresponding orders of the moment. In our case a discrete formula is used [see Eq. (18)], where the image is considered as a discrete function $f(x, y)$ with $x = 0, 1, \dots, M$ and $y = 0, 1, \dots, N$, [19].

$$\int \int x^p y^q g(x, y) dx, dy \quad (17)$$

$$m_{pq} = \sum_{x=0}^M \sum_{y=0}^N x^p y^q f(x, y). \quad (18)$$

This formula is calculated over the area of the pattern (raindrops). When we segment an object, we get a binary image whose pixels contain the value of either one or zero, where I_B is the image and (x, y) is the position of the pixel in the image I_B . The zero order moment is the sum of the pixel values of an image, it is defined as follows [see Eqs. (19), (20)]:

$$\mu_{00} = \sum_x \sum_y I_B(x, y) \quad (19a)$$

$$\mu_{10} = \sum_x \sum_y x I_B(x, y) \quad (19b)$$

$$\mu_{01} = \sum_x \sum_y y I_B(x, y). \quad (19c)$$

Because first order moments of this form do not have any physical interpretation, they need to be normalized:

$$\mu_{10} = \frac{\sum_x \sum_y x I_B(x, y)}{\mu_{00}} \quad (19d)$$

$$\mu_{01} = \frac{\sum_x \sum_y y I_B(x, y)}{\mu_{00}}. \quad (19e)$$

The normalized first order moments contain information about the center of gravity of the pattern. The normalized second order moments are associated with the orientation of the pattern:

$$\mu_{20} = \sum_x \sum_y x^2 I_B(x, y) \quad (20a)$$

$$\mu_{02} = \sum_x \sum_y y^2 I_B(x, y) \quad (20b)$$

$$\mu_{11} = \sum_x \sum_y xy I_B(x, y). \quad (20c)$$

After computing the center points using moments, the distance vectors between the center points and contours are

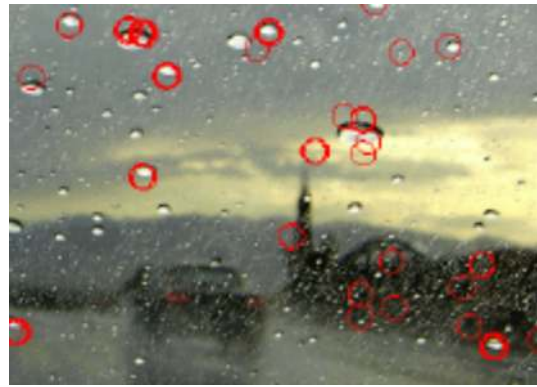


Fig. 5 Red circles indicate the identified raindrops (using edge features)

computed (see Fig.4). The number of vectors is reduced using principal component analysis. After reducing the number of vectors for every closed contour, the norms of these vectors are stored as features. Afterwards, the SVM is trained with this information. It is indeed a shortcoming of this approach that edge features appear not only with raindrops since the tests in complex environments were not as good as simple environments. Figure 5 shows the result of the raindrop detection using edge features. The image in Fig. 3 is the input test image.

7.2 Using color features for SVM

HSV (Hue, Saturation and Value) color space is used for color features. Hue represents the color, saturation indicates the range of gray in the color space, and value is the brightness of the color and varies with color saturation. The use of this color space comes with the advantage that operations on one layer give the same results as in the RGB color space (Red, Green and Blue) for all three layers. Furthermore, HSV is similar to the color model, which is used by humans [1, 28, 34]. Because the brightness is clearly a major characteristic of raindrops, the V channel is used to train the SVM. Figure 6 illustrates the result of the SVM using the V features and shows that the result is much better compared to raindrop detecting using edge features.

7.3 Using histograms of oriented gradient features for SVM

Dalal and Triggs [8] did describe Histograms of Oriented Gradient (HOG) descriptors which can be used in computer vision and image processing problems for detecting objects. HOG have proven to be very effective for the verification of a variety of objects. This feature is similar to that of SIFT descriptors [22], edge orientation histograms [11], and shape contexts [2], but HOG are

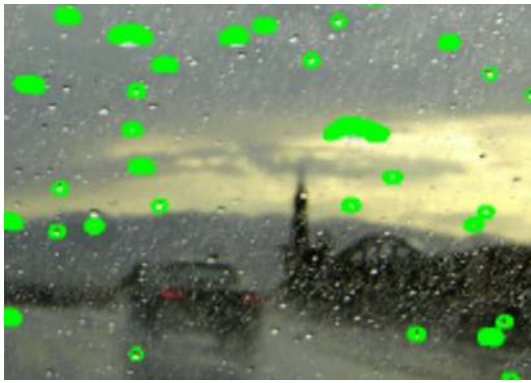


Fig. 6 The *green circles* indicate the identified raindrops (by using HSV gradient)



Fig. 8 The *blue circles* indicate the identified raindrops (using wavelets)

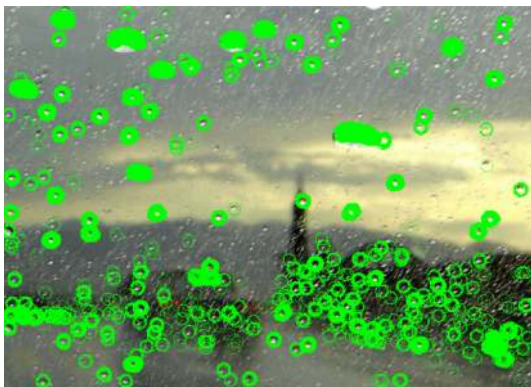


Fig. 7 The *green circles* indicate the identified raindrops (using HOG features)

computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy. HOG features are applied to build a training set to increase the performance of the classification. As a matter of fact, it led to the best results as compared with other features (see Fig. 7).

7.4 Using wavelets

Wavelet transformations are used widely nowadays, which makes it easier to compress, transmit, and analyze many images as compared for example to using the Fourier transform [14]). The transformation is based on small waves, so-called wavelets, of varying frequency and limited duration. In contrast, the Fourier transform is based on sinusoids. Wavelets are used to compress images or to suppress noise in images.

Tong et al. [33] showed that using ‘Haar’ wavelet transforms, the blurring coefficient of an image can be computed. ‘Haar’ wavelet transforms are the oldest and simplest orthonormal approach. According to linear algebra, it is essential that the ‘Haar’ functions are orthogonal

to each other, otherwise a unambiguous reconstruction is not possible. The idea now is to decompose an image with the ‘Haar’ wavelet, compare the levels and decide whether an image is blurred or not by counting the edges. The algorithm consists of the following steps:

1. Perform ‘Haar’ wavelet transform on the image with a decomposition level of 3.
2. Construct the edge map in each scale.
$$\text{Emap}_i(k, l) = \sqrt{LH_i^2 + HH_i^2 + HL_i^2} \quad i = 1, 2, 3.$$
 H means “high pass filter” and *L* means “low pass filter”.
3. Partition the edge maps and find local maxima in each window. The window size in the highest scale is 2×2 , the next more coarse scale is 4×4 , and the most coarse one is 8×8 . The result is denoted as $\text{Emap}_i \quad i = 1, 2, 3$.
4. Use the rules and the blur detection scheme of [33] to compute the blurred regions within the image.

Each recognized raindrop will be recorded and the surrounding coordinates (in radius *d* around the center point of the center point of the feature) will be marked (with a predefined color) and maintained in a copy of the original image. Figure 8 illustrates that the performance of this approach is not as good as the HSV feature approach.

8 Obtained results and experimental setup

This section outlines the experimental setup of the proposed approach. A camera was mounted under the rearview mirror, facing the windshield. In total, 315 traffic images (7054 raindrops) were taken with different traffic backgrounds in rainy weather conditions. 189 out of these images were used for training data set and 126 images for testing data set. These images have a dimension of 510×585 pixels. Raindrops that are well perceptible were extracted manually using GIMP 2.6.8. All of the extracted

raindrops offered the common characteristic that they were clearly recognizable as raindrops for human beings. In our experiments, different rescaling sizes were used for the final raindrops. The experiments yielded the result that a dimension of 15×15 pixels for a raindrop feature exhibits the best performance in the classification phase. Therefore, the SVM is trained with 15×15 images representing features of raindrops. Additionally, 15×15 images had to be created (are randomly chosen non-raindrop image regions) to establish a set of negatives (in terms of a training set for SVMs).

8.1 Performance and evaluation

After finishing the implementation of the final algorithm the system was tested under several conditions. The best

parameters to measure the performance of the system are the measures of sensitivity, specificity and accuracy. Sensitivity accounts for the proportion of the actual positives (i.e. those objects that are correctly identified) whereas specificity reflects the proportion of negatively classified objects (i.e. which are identified incorrectly). The accuracy is the degree of closeness of measurements of a quantity to that quantity's actual (true) value [16] (Fig. 9).

Table 1 represents the performance measures of HSV gradient, HOG, HSV and HOG and edge features and the computation time average of processing each image using these features (on GPU platforms). The related ROC curves are represented in Fig. 10. The best performance is given by HOG features approach which exhibits an accuracy value of 98.25%. Furthermore, the table shows that the computation time average for raindrop detection on GPU

Fig. 9 The detection results of various experiments where our method has been using different sets of features

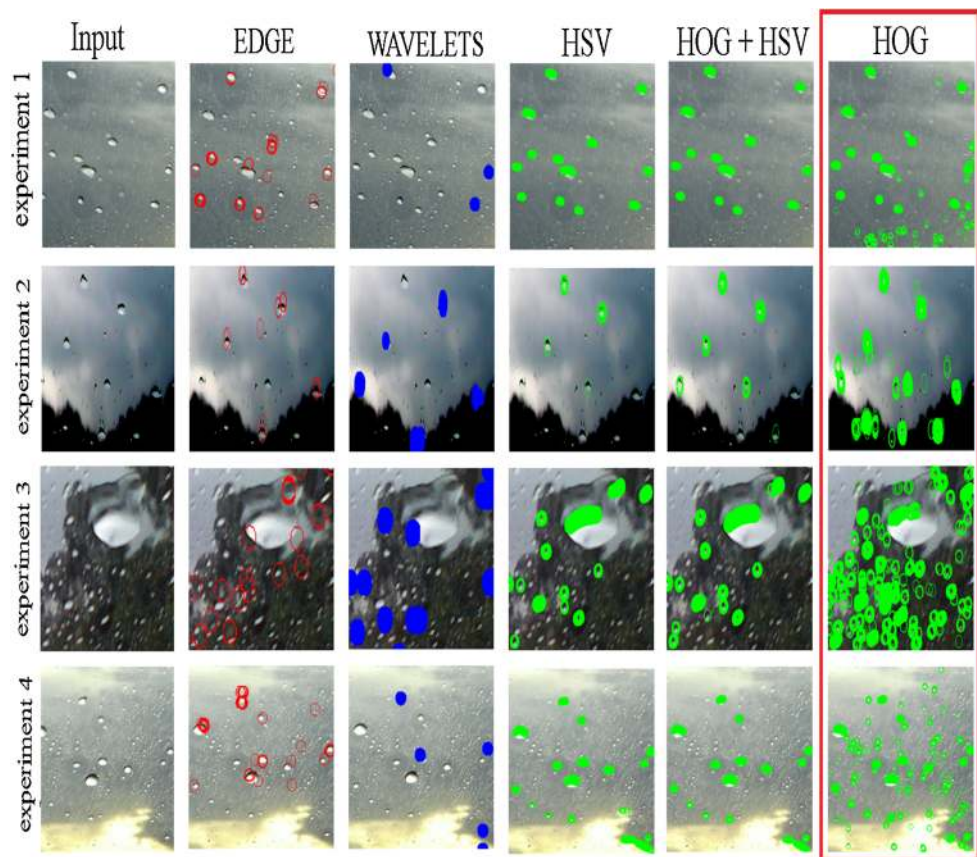


Table 1 Performance measures and the average computation time (510×585 pixels on GPU) of HSV gradient, HOG, HSV and HOG and edge features

Feature	Specificity (%)	Sensitivity	Accuracy (%)	Process time in s
HOG	98.68	94.32 %	98.25	0.565
HSV gradient	98.79	67.60 %	95.66	0.488
HSV and HOG	98.75	65.44 %	96.1	0.684
Wavelet	97.90	40.45	93.23	0.676
Edge features	96.83	56.43 %	93.54	0.563

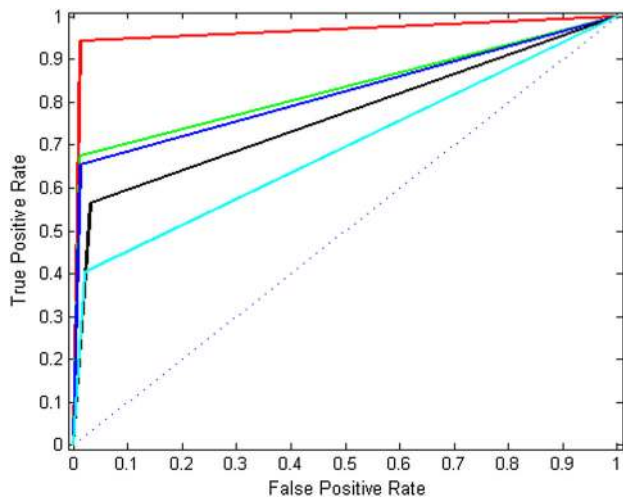


Fig. 10 The red line represents the ROC curve using HOG features, the blue line represents the ROC curve using HSV and HOG features, the black line represents the ROC curve using Edge features, the turquoise color represents the ROC curve using wavelet features and the green line represents the ROC curve using HSV features

(see Sect. 8.2) is between 0.488 and 0.676 s for 510×585 pixels.

Figure 9 shows some experiment results of our method using different sets of features. The input images are shown in the first column. Each line is representing an experiment (different input image). Columns 2 to 6 show the results of applying our method using Edge, Wavelets, HSV, HSV+HOG and HOG features respectively.

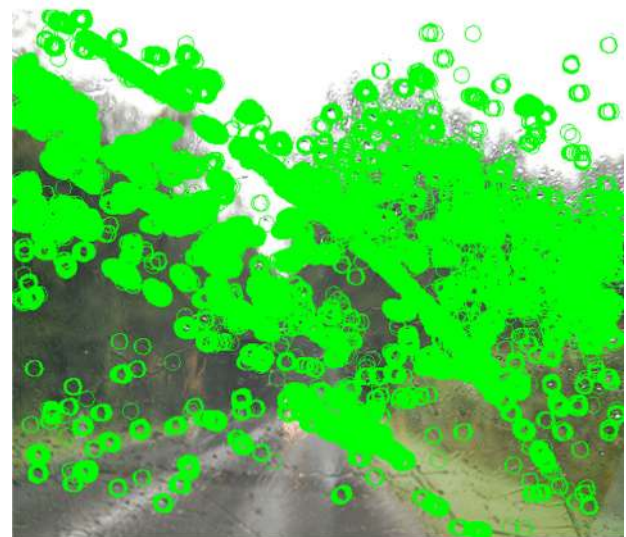
For more realistic scenarios, we captured different images of a windshield of a car traveling at 65 mph. In this case, the raindrops start to elongate and appear like streaks, especially, when the windshield wipers. Figure 11a demonstrates such a case. After applying our method, Fig. 11b proves the high performance of our approach where the water streaks are well detected.

8.2 Benchmarking regarding the computing/processing time

The proposed approach has been tested using two different platforms, namely a CPU as well as a GPU. In this section, we present the results obtained and the benchmarks used to evaluate the performance of the CNN processors system executed on both the GPU and the CPU. The resources available to evaluate the results have been Intel (R) Xeon (TM) CPU 3.60 GHz with 2GB RAM and 32-bit Windows operating system. The GPU used for this process is GeForce 9500 GT with 256 MB of memory. The hardware units used have four multiprocessors with 8 cores on each multiprocessor. In total these units provide 32 CUDA cores



(a)

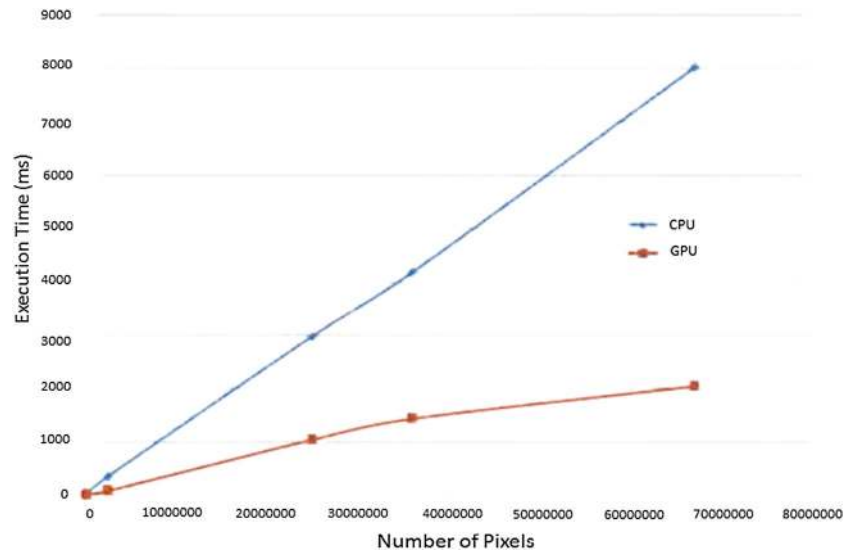


(b)

Fig. 11 Streaks Raindrops scenario. a Examples of raindrops on a windshield of a car traveling at 65 mph. b Raindrops detection using CNN and SVM

which are capable of running in parallel. The development environment was Visual Studio 2011. Figure 12 shows a comparison between the performance of the CPU with that of the GPU with respect to the number of pixels and the corresponding execution times. It can be seen that when the number of pixels is low the level of performance offered by the GPU is nearly equal to the performance level of the CPU. When the number of pixels to be processed is increased, the amount of parallelism obtained is high and hence the overall performance of the GPU becomes much better [25].

Fig. 12 A benchmark for CPU and GPU. The *blue line* is the performance of the CPU and the *red line* is the performance of the GPU



9 Conclusion and outlook

Improving the quality of images that have been degenerated through bad weather influences for the purpose of being further processed by ADAS is an important concern. This is due to the challenging fact that ADAS systems that are based on image processing are sensitive to noise and aberrations/anomalies. Therefore, it is important to provide an image pre-processing unit which is capable of working under real-time conditions and furthermore to improve the quality of images to increase the reliability of ADAS. Image processing in ADAS is important, since images deliver a broad spectrum of data which can be interpreted for different algorithms.

Related works which generally do not involve CNN have been found to perform less or do not consider clearly the real-time constraints. In addition, most of the existing systems are not easy to implement on hardware. We did overcome these shortcomings in our system which is based on pattern recognition and cellular neural networks. For pattern recognition, support vector machines were involved which provide a model for the classification phase. Different features are used for the training phase, where the HOG of the raindrop shape exhibit the best results. This model is then used for the configuration of CNN's which are capable to classify image regions in real-time.

10 Compliance with ethics requirements

This paper does not contain any studies with human or animal subjects and all authors declare that they have no conflict of interest.

Acknowledgements Open access funding provided by University of Klagenfurt.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Acharya, T., Ray, A.K.: Image processing: principles and applications. Wiley. com (2005)
2. Belongie, S., Malik, J.: Puzicha J Matching shapes. In: Computer vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, IEEE, vol 1, pp 454–461 (2001)
3. Chandler, B., Rekeczky, C., Nishio, Y., Ushida, A.: Adaptive simulated annealing in cnn template learning. IEICE Trans. Fundamentals Electronics Comm. Comput. Sci. **82**(2), 398–402 (1999)
4. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Trans. Intell. Syst. Technol. (TIST) **2**(3), 27 (2011)
5. Chua, L.O., Roska, T.: Cellular neural networks and visual computing: foundations and applications. Cambridge University Press (2002)
6. Chua, L.O., Yang, L.: Cellular neural networks: applications. Circ. Syst. IEEE Trans. **35**(10), 1273–1290 (1988)
7. Chua, L. O., Yang, L.: Cellular neural network. US Patent 5,140,670 (1992)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer vision and pattern recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, IEEE, vol 1, pp 886–893 (2005)
9. Eigen, D., Krishnan, D., Fergus, R.: Restoring an image taken through a window covered with dirt or rain. In: Computer vision (ICCV), 2013 IEEE International Conference on, IEEE, pp 633–640 (2013)

10. Ercsey-Ravasz, M., Roska, T., Nédá, Z.: Statistical physics on cellular neural network computers. *Physica D: Nonlinear Phenomena* **237**(9), 1226–1234 (2008)
11. Freeman, W.T., Roth, M.: Orientation histograms for hand gesture recognition. *Int. Workshop Automatic Face Gesture Recog.* **12**, 296–301 (1995)
12. Gacsádi, A., Grava, C.: Grava, A. Medical image enhancement by using cellular neural networks. In: *Computers in cardiology*, 2005, IEEE, pp 821–824 (2005)
13. Garg, K., Nayar, S.K.: Detection and removal of rain from videos. In: *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on, IEEE*, vol 1, pp 1–528 (2004)
14. Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*, 3rd edn. Prentice-Hall Inc, Upper Saddle River, NJ, USA (2006)
15. Halimeh, J.C., Roser, M.: Raindrop detection on car windshields using geometric-photometric environment construction and intensity-based correlation. In: *Intelligent Vehicles Symposium, 2009 IEEE*, IEEE, pp 610–615 (2009)
16. Hand, D.J.: Measuring classifier performance: a coherent alternative to the area under the roc curve. *Machine Learning* **77**(1), 103–123 (2009)
17. Karacs, K., et al.: Software library for cellular wave computing engines. http://cnn-technologyitkppkehu/Template_library_v31 (2010)
18. Kek, L., Karacs, K., Roska, T.: Cellular wave computing library v2. 1 (2007)
19. Kotoulas, L., Andreadis, I.: Image analysis using moments. 5th Int, pp. 360–364. *Conf. on Technology and Automation, Thessaloniki, Greece* (2005)
20. Kozek, T., Roska, T., Chua, L.O.: Genetic algorithm for cnn template learning. *IEEE transactions on circuits and systems 1, Fundamental theory and applications* **40**(6):392–402 (1993)
21. Kurihata, H., Takahashi, T., Ide, I., Mekada, Y., Murase, H., Tamatsu, Y., Miyahara, T.: Rainy weather recognition from in-vehicle camera images for driver assistance. *Intelligent Vehicles Symposium, 2005*, pp. 205–210. *Proceedings. IEEE, IEEE* (2005)
22. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Inte. j. Comput Vision* **60**(2), 91–110 (2004)
23. Mirzai, B., Cheng, Z., Moschytz, G.S.: Learning algorithms for cellular neural networks. In: *IEEE International Symposium On Circuits And Systems, Institute of Electrical Engineers Inc (IEEE)*, pp 159–162 (1998)
24. Nashashibi, F., de Charrette, R., Lia, A.: Detection of unfocused raindrops on a windscreen using low level image processing. In: *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on, IEEE*, pp 1410–1415 (2010)
25. Potluri, S., Fasih, A., Vutukuru, L.K., Al Machot, F., Kyamakya, K.: Cnn based high performance computing for real time image processing on gpu. In: *Nonlinear Dynamics and Synchronization (INDS) & 16th Int'l Symposium on Theoretical Electrical Engineering (ISTET), 2011 Joint 3rd Int'l Workshop on, IEEE*, pp 1–7 (2011)
26. Roser, M., Geiger, A.: Video-based raindrop detection for improved image registration. In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, IEEE*, pp 570–577 (2009)
27. Roser, M., Kurz, J., Geiger, A.: Realistic modeling of water droplets for monocular adherent raindrop recognition using bézier curves. In: *Computer Vision-ACCV 2010 Workshops, Springer*, pp 235–244 (2011)
28. Russ, J.C.: *The image processing handbook*. CRC Press (2006)
29. Sakr, G.E., Elhadj, I.H., Huijjer, H.S.: Support vector machines to define and detect agitation transition. *Affective Comput. IEEE Trans.* **1**(2), 98–108 (2010)
30. Schwarzmuller, C., Al Machot, F., Fasih, A., Kyamakya, K.: Adaptive contrast enhancement involving cnn-based processing for foggy weather conditions & non-uniform lighting conditions. In: *Nonlinear Dynamics and Synchronization (INDS) & 16th Int'l Symposium on Theoretical Electrical Engineering (ISTET), 2011 Joint 3rd Int'l Workshop on, IEEE*, pp 1–10 (2011)
31. Su, T.J., Cheng, J.C., Huang, M.Y., Lin, T.H., Chen, C.W.: Applications of cellular neural networks to noise cancelation in gray images based on adaptive particle-swarm optimization. *Circ. Syst. Signal Proc.* **30**(6), 1131–1148 (2011)
32. Sugimoto, M., Kakiuchi, N., Ozaki, N., Sugawara, R.: A novel technique for raindrop detection on a car windshield using geometric-photometric model. In: *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, IEEE*, pp 740–745 (2012)
33. Tong, H., Li, M., Zhang, H., Zhang, C.: Blur detection for digital images using wavelet transform. In: *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on, IEEE*, vol 1, pp 17–20 (2004)
34. Tönnies, K.D.: *Grundlagen der Bildverarbeitung*, vol 1. Pearson Studium (2005)
35. Vapnik, V.: *The nature of statistical learning theory*. springer (2000)
36. Vapnik, V., Golowich, S.E., Smola, A.: Support vector method for function approximation, regression estimation, and signal processing. *Adv. Neural Inf. Proc. Syst* pp 281–287 (1997)
37. Wang, W., Yang, L.J., Xie, Y.T., Yw, An: Edge detection of infrared image with cnn_dga algorithm. *Optik-Int. J Light Electron Optics* **125**(1), 464–467 (2014)
38. Wu, Q., Zhang, W., Chen, T., Kumar, B.V.: Camera-based clear path detection. In: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, IEEE*, pp 1874–1877 (2010)
39. Wu, Q., Zhang, W., Vijaya Kumar, B.: Raindrop detection and removal using salient visual features. In: *Image Processing (ICIP), 2012 19th IEEE International Conference on, IEEE*, pp 941–944 (2012)
40. Yamashita, A., Tanaka, Y., Kaneko, T.: Removal of adherent waterdrops from images acquired with stereo camera. In: *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on, IEEE*, pp 400–405 (2005)
41. Yamashita, A., Fukuchi, I., Kaneko, T.: Noises removal from image sequences acquired with moving camera by estimating camera motion from spatio-temporal information. In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, IEEE*, pp 3794–3801 (2009)
42. Yang, C.L.: A study of video-based water drop detection and removal method for a moving vehicle. (2011)
43. You, S., Tan, R.T., Kawakami, R., Ikeuchi, K.: Adherent raindrop detection and removal in video. *Environment* **21**(22), 18 (2012)