Neurocomputing 245 (2017) 31–45

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Efficient deep network for vision-based object detection in robotic applications

Keyu Lu*, Xiangjing An, Jian Li, Hangen He

College of Mechatronic Engineering and Automation, National University of Defense Technology, Changsha 410073, Hunan, China

ARTICLE INFO

Article history:
Received 17 June 2016
Revised 11 February 2017
Accepted 17 March 2017
Available online 24 March 2017

Communicated by Dr Zhang Zhaoxiang

Keywords:
Deep network
Object detection
Computer vision
Robotic application
MPGA

ABSTRACT

Vision-based object detection is essential for a multitude of robotic applications. However, it is also a challenging job due to the diversity of the environments in which such applications are required to operate, and the strict constraints that apply to many robot systems in terms of run-time, power and space. To meet these special requirements of robotic applications, we propose an efficient deep network for vision-based object detection. More specifically, for a given image captured by a robot mount camera, we first introduce a novel proposal layer to efficiently generate potential object bounding-boxes. The proposal layer consists of efficient on-line convolutions and effective off-line optimization. Afterwards, we construct a robust detection layer which contains a multiple population genetic algorithm-based convolutional neural network (MPGA-based CNN) module and a TLD-based multi-frame fusion procedure. Unlike most deep learning based approaches, which rely on GPU, all of the on-line processes in our system are able to run efficiently without GPU support. We perform several experiments to validate each component of our proposed object detection approach and compare the approach with some recently published state-of-the-art object detection algorithms on widely used datasets. The experimental results demonstrate that the proposed network exhibits high efficiency and robustness in object detection tasks.

1. Introduction

Object detection plays a critical role in a wide range of robotic applications such as service robot interaction [1], autonomous driving [2], and collision avoidance [3], which need to detect the presence of both stationary and moving objects in a specific area of interest around the host robots in order to perform corresponding actions such as interaction, braking and evading.

Vision sensors become particularly important in robot systems [4]. In the first place, compared with other popular object detection sensors such as LiDAR and millimeter-wave radar, vision sensors are able to obtain rich information from objects (such as luminance, color and texture) and to monitor wide areas [5], and these capabilities are of value to a range of robotic applications. Moreover, the cost of most vision sensors is much lower than that of LiDAR or millimeter-wave radar [6], and consequently vision sensors are a more feasible solution for robots employed for commercial use. Last but not least, vision sensors require much less power and space, and accordingly they are particularly suitable for integration into robot systems where power and space are limited.

Vision-based object detection has been addressed through a wide variety of methods in recent years. In the last decade, hand-craft solutions have proved extremely popular and have been applied successfully in several robotic applications. Early hand-craft solutions mainly focused on detecting objects using keypoints. Lowe proposed the SIFT algorithm [7] that is able to achieve scale and rotational invariance in keypoint detection, and this made keypoint-based approaches a popular solution to the object detection problem. Subsequently, numerous extensions (e.g., PCA-SIFT [8] and SUFR [9]) emerged that aimed at improving object detection performance in the context of robotic applications. In a different line of work, Viola and Jones [10] proposed a boosted object detection method for real-time object detection, using cascades strategy and Haar-like features to make efficient calculations through integral images. In the same vein, Dollár et al. introduced the Aggregated Channel Feature (ACF) [11], which obtains multiple channels of carefully tuned features through integral images and trains a robust object detector using AdaBoost [10] and efficient decision trees. Meanwhile, Dalal et al. [12] proposed an alternative hand-craft approach and popularized the use of the “HOG+SVM” paradigm in object detection. The main idea of this approach is to represent each image patch using histogram of oriented gradients and to classify the feature vector of each image patch using support vector machine (SVM) [13]. Later, a breakthrough in

* Corresponding author.
E-mail addresses: keyu.lu@nudt.edu.cn (K. Lu), anxiangjing@gmail.com (X. An), jianli@nudt.edu.cn (J. Li), hangenhe@yahoo.com (H. He).

http://dx.doi.org/10.1016/j.neucom.2017.03.050
0925-2312/

the application of the "HOG+SVM" paradigm to object detection was achieved in the form of the DPM (deformable part model) [14], which represents an image using a variant of HOG features [12] and takes both the whole and parts of an object into consideration by constructing a root model and multiple part models.

Another class of approaches for object detection is based on deep learning, and this approach has achieved tremendous success in recent years. Among them, convolutional neural networks (CNNs) [15] are a subtype of deep artificial neural networks which are inspired by the animal visual cortex organization [16]. CNNs have achieved state-of-the-art performance using local receptive fields, weight sharing and spatial pooling [17]. With the development of CNN-based models, they are increasingly prevalent in robotic applications. Alex et al. proposed AlexNet [15], that consists of several convolutional layers, max-pooling layers and fully-connected layers. This method achieved best performance in the ImageNet ILSVRC-2012 competition and shows significant gains over state-of-the-art hand-craft methods [10,12,14]. Thereafter, Google and Baidu have improved their image search engines according to this deep learning architecture which increases the searching accuracy markedly [16]. Nevertheless, deeper convolutional neural networks are often more difficult to train [18], and this causes a bottleneck in the development of CNNs. To overcome this limitation, He et al. introduced the deep residual network [18] that enables deeper CNN training through residual learning. With 152 layers, this deep residual network [18] won first prize in several tasks in the ImageNet ILSVRC 2015 and COCO 2015 competitions.

For object detection tasks, Girshick et al. proposed a R-CNN [19] framework that first generates a set of proposal bounding-boxes that are likely to contain objects, using region proposal methods such as Selective Search [20] and Edge Boxes [21]. Subsequently, CNNs are applied to these proposal bounding-boxes to achieve accurate object detection. In this way, the R-CNN framework has been successfully adapted for object detection tasks. However, as CNNs require a fixed-size input, R-CNN repeatedly has to resize proposal bounding-boxes to a fixed size and compute the convolutional features of each proposal bounding-boxes, and these processes consume valuable time in object detection. To eliminate this requirement, Girshick et al. later equipped the networks with ROI pooling layers and proposed the Fast R-CNN [22] algorithm, which reduces the computational complexity considerably and yet improves object detection performance. Nevertheless, the proposal generation method (Selective Search [20]) of Fast R-CNN remains inefficient and thus presents a bottleneck in efficient object detection. Later, the upgraded version "Faster R-CNN" [23] is proposed. Instead of Selective Search [20], Faster R-CNN utilizes region proposal networks (RPNs) to generate proposals. In this way, proposal generation and CNN-based classification/regression can be run under an unified framework and thus the detection speed can be boosted with the help of GPU [23].

In a different range of works, YOLO [24] opens the door to achieve real-time objects detection by regarding it as a regression problem. The method is able to regresses bounding-box of each object from the whole image at a extremely high frame rate. However, its performance on small objects detection and location accuracy are unsatisfactory [25]. Thereafter, SSD (Single Shot MultiBox Detector) [25] is proposed for regression-based object detection. With the help of multi-scale feature maps and default boxes techniques, SSD [25] has remarkable improvements in small objects detection and location accuracy compared with YOLO [24].

Despite the fact that CNNs have achieved excellent performance in object detection tasks, they still cannot effectively handle the problems of object rotation, within-class variability, and between-class similarity [26]. To address these issues, Cheng et al. proposed RIFD-CNN [26] that combines rotation-invariant CNN model and Fisher discriminative CNN model together to improve the performance of CNNs in object detection. To further improve the performance of CNN-based object detection, GBD-Net [27] is proposed to integrate local and contextual visual cues in CNN and has achieve remarkable success in the object detection task of ILSVRC 2016 competitions.

Although previous deep learning-based approaches have achieved considerable success in object detection, few of them could realistically be employed in robotic applications. In the first place, object detection in the real world is a challenging task, due to the near-limitless diversity of possible backgrounds, illumination, viewpoints and imaging conditions. In the next place, the behaviors of different moving objects (e.g., pedestrians and vehicles) are often extremely flexible and unpredictable, compounding the difficulties presented by robust object detection. Moreover, unlike the object detection tasks in the ILSVRC and COCO competitions, most robotic applications have strict constraints in run-time, power and space. Accordingly, in these robotic applications the computational complexities of object detection algorithms must

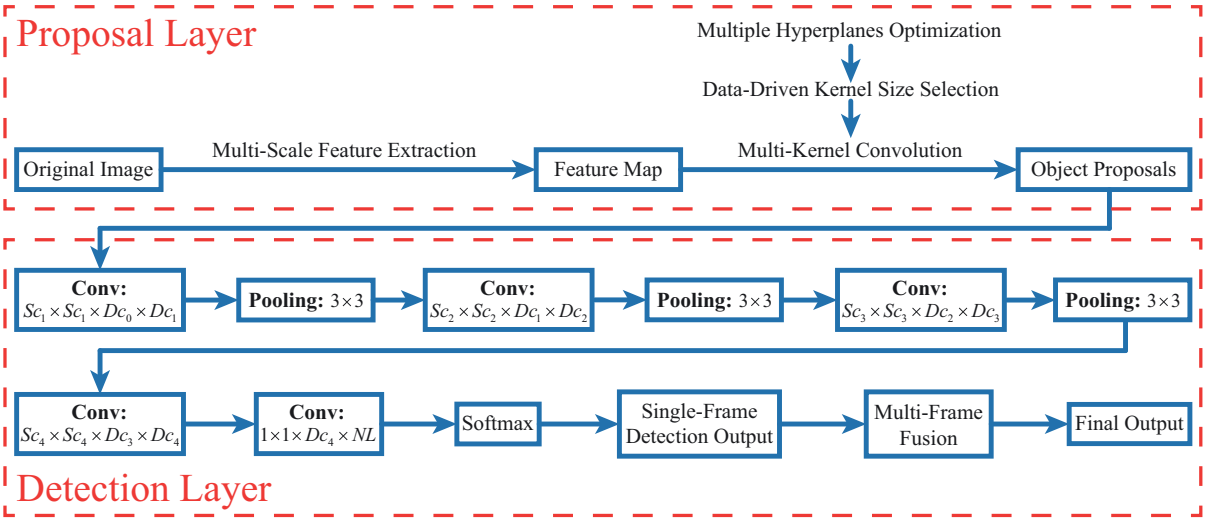


Fig. 1. Diagram of our efficient deep network.

be low enough that the applications can run efficiently without the support of a high performance server, workstation or even GPU. Only then could they be integrated into power and space constrained robot systems.

The motivation of this paper is to reduce the run-time of CNN-based object detection system and make it possible to be utilized in robotic applications without GPU support. All the on-line processes in our network are able to run efficiently and effectively without GPU support. In more detail, the proposed object detection architecture is depicted in Fig. 1, which consists of two main components:

(1) **Proposal layer:** This structure is proposed to efficiently generate potential object bounding-boxes, which could efficiently avoid exhaustive object searching across an image and yet improve object detection performance by reducing the complexity of the classification task for the detection layer. The proposal layer consists of multiple hyperplanes optimization, data-driven kernel size selection, multi-scale feature extraction and multi-kernel convolution. As multiple hyperplanes and data-driven kernel size selection do not need to run on-line, the proposal layer is able to work efficiently and effectively.
(2) **Detection layer:** This component runs on the proposal bounding-boxes generated by the proposal layer. We first propose a multiple population genetic algorithm-based convolutional neural network (MPGA-based CNN) module that is able to determine the number of feature maps at each layer and makes a trade-off between performance and computational complexity. Then, a TLD-based multi-frame fusion strategy is proposed to utilize temporal information and improve the detection performance.

Furthermore, we perform several experiments to validate each component of our proposed object detection approach and compare the results with some recently published state-of-the-art object detection algorithms on widely used datasets.

The rest of this paper is organized as follows. In Section 2, we introduce the proposal layer of our object detection network. In Section 3, we present the detection layer, including MPGA-based CNN and multi-frame fusion. Experimental results are presented in Section 4. Conclusions are drawn and future works proposed in Section 5.

2. Proposal layer

In this section, we detail our approach to generating a set of proposal bounding-boxes that are likely to contain objects. On the one hand, the proposal layer can avoid exhaustive object candidate searching across the given image and reduce run-time significantly; on the other hand, it is able to improve object detection performance by screening out a large number of negative samples, and thus reduces the complexity of the classification task to be performed by the detection layer. The proposal layer involves efficient on-line procedures (including proposal feature extraction and convolution-based proposal generation) and effective off-line procedures (including multiple hyperplanes optimization and data-driven kernel size selection).

2.1. Proposal feature extraction

Inspired by Dollár [11], we initially extract a set of gradient histograms as the proposal feature of the given image, which is computationally efficient and able to represent the edge and texture of an object. Instead of generating features from grey level images [12], we compute gradient histograms over RGB color images. The respective horizontal and vertical gradients of each color channel can be obtained by applying the Sobel filter $[-1, 0, 1]$ and $[-1, 0, 1]^{\mathrm{T}}$ on the image. For a pixel located at the ith row and jth column of the given image, the horizontal gradient $Gx(i, j)$ and the vertical gradient $Gy(i, j)$ are defined as follows:

$$\begin{cases} Gx(i,j) = \max(Grx(i,j), Ggx(i,j), Gbx(i,j)) \\ Gy(i,j) = \max(Gry(i,j), Ggy(i,j), Gby(i,j)) \end{cases}, \tag{1}$$

where (Gr, Gg, Gb) denotes the gradients of the three color channels. Having defined the gradient of each pixel, the magnitude $M(i, j)$ and orientation $Ang(i, j)$ of gradient can be written as:

$$\begin{cases} M(i,j) = \sqrt{Gx(i,j)^2 + Gy(i,j)^2} \\ Ang(i,j) = \arctan\left(\frac{Gy(i,j)}{Gx(i,j)}\right) \end{cases}. \tag{2}$$

$Ang(i, j)$ of each pixel is then quantized into k bins. Let $b(i, j)$ denote the bin that the pixel (i, j) belongs to; it can be obtained by:

$$b(i,j) = \left\lfloor \frac{k \cdot Ang(i,j)}{2\pi} \right\rfloor, \tag{3}$$

where $\lfloor \cdot \rfloor$ is the round-down operation.

To compute the gradient histogram of each location, we divide the image into small spatial regions. A larger size of the regions will result in a lower localization precision, while a smaller size of that will lead to a higher dimension of feature vector, which will increase the computational complexity or even cause the curse of dimensionality for SVM [28]. To balance the localization precision and dimension of feature vector, we use the region size of (4×4) for calculating the gradient histogram. For a region v, a histogram $\vec{x}_v = [h_1, h_2, \ldots, h_k]$ is defined. Each pixel in the region v casts a vote for the histogram $\vec{x}_v$ weighted by $M(i, j)$ according to $b(i, j)$. More specifically, the element h_v $(v = 1, 2, \ldots, k)$ in $\vec{x}_v$ can be formulated as:

$$h_v = \sum\nolimits_{i,j} [b(i,j) = v] \cdot M(i,j), \tag{4}$$

where the Iverson bracket function $[b(i, j) == v]$ is activated only if the bin number $b(i, j)$ equals the index v. The histogram $\vec{x}_v$ is regarded as the proposal feature vector of the spatial region v.

2.2. Convolution-based proposal generation

Convolution becomes a highly efficient operation in CPU-based image processing owing to the support of many proven algorithm libraries such as MKL [29], BLAS [30] and Caffe (CPU version) [31]. According to this fact, we attempt to achieve efficient proposal generation by transforming the dense sliding windows paradigm [10,14] to convolution operation.

As we know, convolution is similar to sliding windows paradigm. That is, each element in convolution result is correspond to a spatial region in input map. Let $\vec{x}_{i,j}$ denote the proposal feature vector of the bounding-box region which is correspond to the element located at the ith row and jth column of the gradient histogram feature space (center-aligned). The prediction of linear support vector machine (SVM) [13] can be formulated as the form of generic linear regression:

$$y = \vec{w} \cdot \vec{x} + b, \tag{5}$$

where "$\cdot$" denotes dot product. $\vec{w}$ and b are the weight matrix and bias of the SVM classifier respectively; y is the prediction value. The proposal feature vector $\vec{x}_{i,j}$ is comprised of k bins (denoted by $h_1(i, j), h_2(i, j), \ldots h_k(i, j)$). For a bounding-box with m rows and n

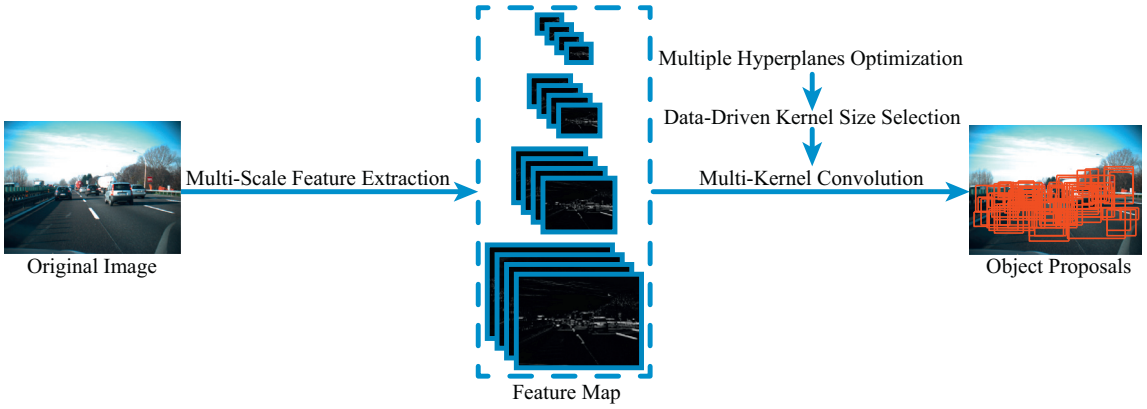


Fig. 2. Architecture of the proposal layer.

columns, Eq. (5) can be transformed to the following form:

$$\begin{aligned} y &= \sum_{i=1}^{m}\sum_{j=1}^{n} \vec{w}_{i,j} \cdot \vec{x}_{i,j} + b \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{p=1}^{k} w_p(i,j) \cdot h_p(i,j) + b \\ &= \sum_{p=1}^{k} \vec{w}_p \cdot \vec{h}_p + b, \end{aligned} \tag{6}$$

where $\vec{w}_p$ and $\vec{h}_p$ are the weight and feature matrix of the pth feature channel ($p = 1, 2, \ldots, k$), respectively. Let $\vec{\hat{w}}_p$ be the 180-degree rotated form of $\vec{w}_p$; that is, for the element at (i, j) of $\vec{w}_p$, $\vec{\hat{w}}_p$ satisfies:

$$\vec{w}_p(i,j) = \vec{\hat{w}}_p(m - i + 1, n - j + 1), \tag{7}$$

where m and n respectively represent the number of rows and columns of $\vec{w}_p$. Eq. (7) indicates that $\vec{\hat{w}}_p$ can be obtained by changing the order of elements in $\vec{w}_p$.

Having obtained $\vec{\hat{w}}_p$, Eq. (6) can be derived to:

$$\begin{aligned} y &= \sum_{p=1}^{k} \vec{w}_p \cdot \vec{h}_p + b \\ &= \sum_{p=1}^{k}\sum_{i=1}^{m}\sum_{j=1}^{n} \hat{w}_p(m - i + 1, n - j + 1) \cdot h_p(i,j) + b \\ &= \sum_{p=1}^{k} \vec{\hat{w}}_p \otimes \vec{h}_p + b, \end{aligned} \tag{8}$$

where $\otimes$ stands for two-dimensional convolution.

In summary, for a standard form of SVM, its weights can be regarded as the ensemble of several matrices which correspond to different feature channels. By performing a rotation of 180 degrees on these matrices, the form of SVM can be transformed to the paradigm of convolution. In Eq. (8), the 180-degree rotated matrix $\vec{\hat{w}}_p$ is called the kernel of convolution. We have trained a series of kernels for bounding-box at different sizes to fit objects in different scales and ratios. More specifically, we first classify each positive sample into a specific scale and ratio according to its size. At the same time, a set of negative samples of the same size are randomly generated from training sets. After that, the feature vectors of these samples are calculated through the method described in Section 2.1. Based on these feature vectors, SVM models in different scales and ratios could be trained. Finally, the SVM models are transformed to the kernels according to Eq. (8). Note that, for experiments in Section 4, as different object detection tasks are evaluated in different datasets, the convolution kernels for different objects are trained on each dataset separately. In this way, initial proposals can be obtained efficiently. Taking $k = 4$ as an example, Fig. 3 illustrates the pipeline of convolution-based proposal generation.

2.3. Data-driven Kernel size selection

As objects may appear in an image at different sizes (e.g., different scales and ratios), a series of convolution kernels at different sizes should be constructed to fit objects with various sizes. For the proposal feature described in 2.1, let W_{ker} and H_{ker} denote the width and height of a convolution kernel, W_{box} and H_{box} denote the width and height of the corresponding bounding-box. They satisfy:

$$\begin{cases} W_{ker} = \frac{W_{box}}{r \cdot c} \\ H_{ker} = \frac{H_{box}}{r \cdot c} \end{cases}, \tag{9}$$

where r and c are the number of rows and columns of each spatial region, which are used to compute the gradient histogram of each location. Eq. (9) indicates that the relationship between the size of convolution kernel and that of its corresponding bounding-box is linear. Thus, the problem of kernel size selection can be transformed to the problem of bounding-box size selection.

In most previous works [10–12,32], the sizes of bounding-boxes are selected in an empirical way. In this work, we propose to select the size of bounding-box using a data-driven approach, which is an off-line procedure and only needs to be performed once.

Taking the vehicle detection problem as an example, vehicles of various types (e.g., cars, trucks, and buses) at different distances from the vision sensor may have different sizes in the 2D image space. We illustrate the bounding-box size selection problem using the TME Motorway dataset [3], which is a challenging and widely-used dataset in vehicle detection. Let the width and height of the ground-truth bounding-box be the horizontal and vertical axes respectively. The distribution of the size of vehicle is depicted in Fig. 4.

We describe the distribution using a distribution map Ds. As shown in Fig. 5, the element $Ds(x, y)$ indicates the number of objects that have the width and height of x and y pixels, respectively.

As we know, an object of larger size is less sensitive to changes in the size of the bounding-box, which means that it is able to cover more bounding-box types. For example, an object of size 10 $\times$ 10 may be able to fit bounding-boxes ranging from 7 $\times$ 7 to 13 $\times$ 13, but the object of size 100 $\times$ 100 may be able to fit bounding-boxes ranging from 70 $\times$ 70 to 130 $\times$ 130. According to this fact,

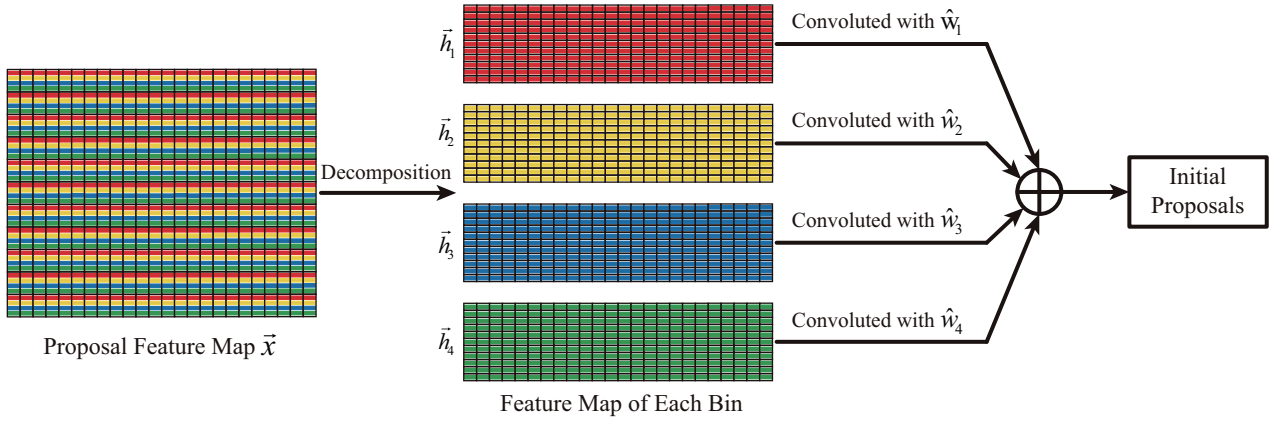


Fig. 3. Pipeline of convolution-based proposal generation (each color represents a histogram channel).

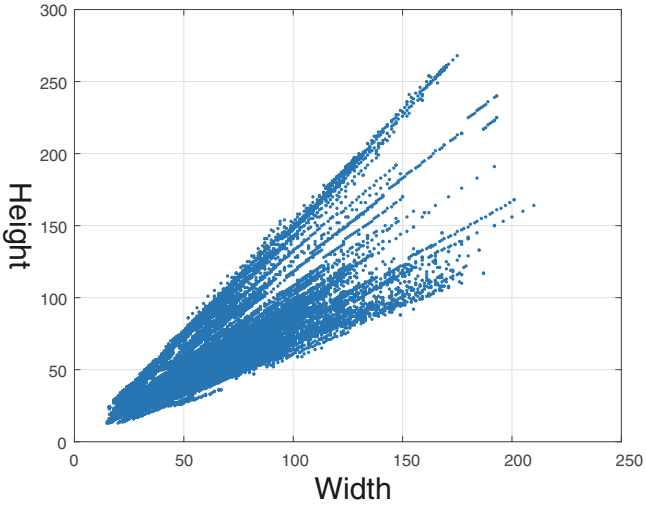


Fig. 4. The distribution of object size in TME Motorway dataset.

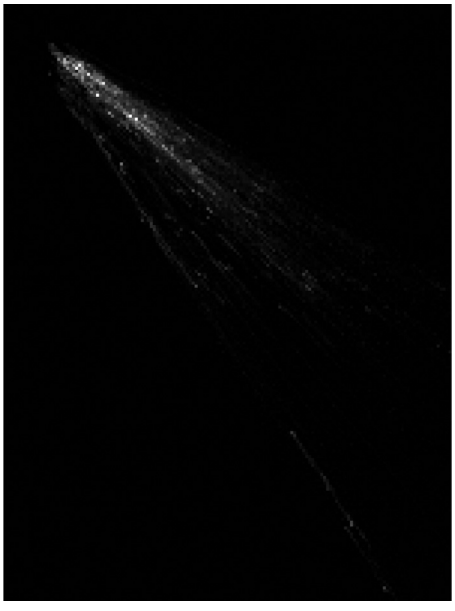


Fig. 5. The distribution map of object in TME Motorway dataset.

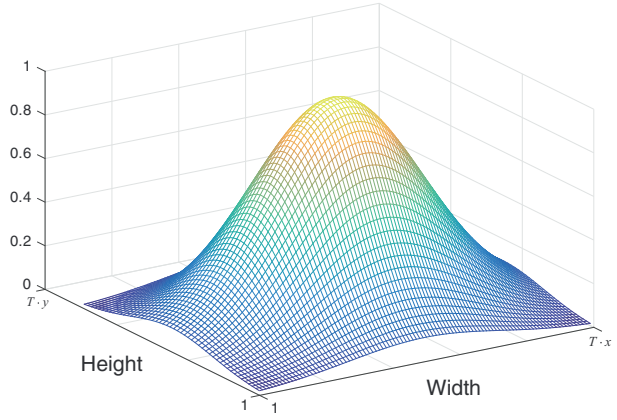


Fig. 6. 2D Gaussian filter $G_{x,\,y}(u,\,v)$ for location $(x,\,y)$ in distribution map.

we defined a 2D Gaussian filter $G_{x,\,y}(u,\,v)$ for each location $(x,\,y)$ to transform the distribution map to an importance map, which indicates the importance of selecting a bounding-box at each size. A location with larger x and y will be filtered by the $G_{x,\,y}(u,\,v)$ of larger scale. Let $u \in [1,\,T{\cdot}x]$ and $v \in [1,\,T \cdot y]$ be the column and row indices of $G_{x,\,y}(u,\,v)$, T is the factor used to control the scale of $G_{x,\,y}(u,\,v)$ ($T = 0.3$ in this work). The 2D Gaussian filter at $(x,\,y)$ can be written as:

$$G_{x,y}(u,v) = e^{-\left(\frac{(u-\mu_1)^2}{\eta_1}+\frac{(v-\mu_2)^2}{\eta_2}\right)}, \tag{10}$$

where:

$$\begin{cases}\mu_1 = \frac{1+T{\cdot}x}{2} \\ \mu_2 = \frac{1+T{\cdot}y}{2}\end{cases}. \tag{11}$$

In order to keep the weight at center close to 1 and the weight at boundary close to 0, η_1 and η_2 are defined as:

$$\begin{cases}\eta_1 = -\frac{(1-\mu_1)^2}{\log(\varepsilon)} \\ \eta_2 = -\frac{(1-\mu_2)^2}{\log(\varepsilon)}\end{cases}, \tag{12}$$

where ε is a small value that close to 0. We choose $\varepsilon = 0.0001$ in this work, which is small enough for the computation in Eq. (12). The 2D Gaussian filter $G_{x,\,y}(u,\,v)$ is illustrated in Fig. 6.

By applying the 2D Gaussian filter $G_{x,\,y}(u,\,v)$, the original distribution map Ds can be transformed to the importance map Es, which is shown in Fig. 7.

The element $Es(x,\,y)$ located at $(x,\,y)$ of the importance map represents the importance of selecting the bounding-box of width x and height y. Supposing that q types of bounding-box need to be

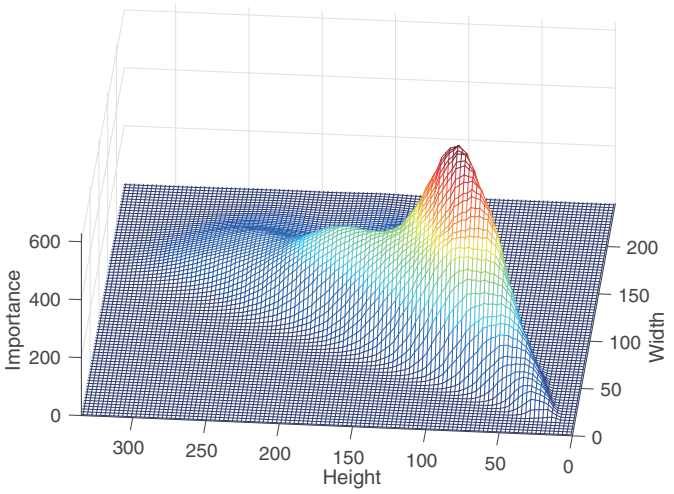


Fig. 7. Original importance map.

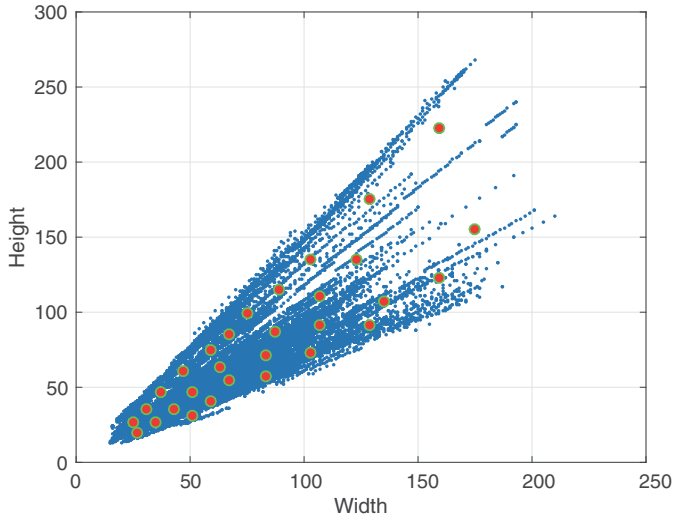


Fig. 8. The result of bounding-box selection. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

selected, we design q iterative procedures to address the problem. At each iteration, a bounding-box of size $xm \times ym$ is selected at the position (xm, ym), which has the maximum importance value in the importance map Es (see Eq. (13)).

$$(xm, ym) = \arg\max_{(x,y)} Es(x, y). \tag{13}$$

If a bounding-box size is selected, then it will be less important to select the same size in the next iteration. In other words, once a bounding-box size is selected in an iteration, the distribution of object sizes that need to be selected will change, and this leads to a reduction in the corresponding importance value. We achieve this by applying a suppression filter $Gs_{xm,\,ym}(u, v)$ on (xm, ym) of distribution map Ds. The suppression filter is defined as the reciprocal of $G_{x,\,y}(u, v)$: that is:

$$Gs_{x,y}(u, v) = \frac{1}{G_{x,y}(u, v)}. \tag{14}$$

After the corresponding position (xm, ym) is filtered by $Gs_{xm,\,ym}(u, v)$, the distribution map is transformed to an importance map by 2D Gaussian filter for the next iteration. In summary, the algorithm of bounding-box size selection is described in Algorithm 1, where Wm and Hm are the resulting width and height of the size selection, respectively.

Algorithm 1 Bounding-box size selection.

Input: Original distribution map Ds_0;
The number of bounding-box types q;
Output: Wm and Hm;
Initialization: $Wm = \varnothing$, $Hm = \varnothing$ and i=0;
Procedure:
1: Transform Ds_i to Es_i by applying $G_{x,y}$ on it;
2: Find the position (xm_i, ym_i) which has the maximum value in Es_i;
3: Update Wm and Hm: $Wm = Wm \bigcup \{xm_i\}$, $Hm = Hm \bigcup \{ym_i\}$;
4: If $i < q$ then calculate Ds_{i+1} by applying Gs_{xm_i,ym_i} at (xm_i, ym_i) of Ds_i;
5: Set $i \leftarrow i + 1$;
6: If $i \leq q$ then go to step 1, otherwise stop the iteration.

Taking $q = 30$ as an example, Fig. 9 depicts the change in the importance map with increasing iterations. Fig. 8 illustrates the result of bounding-box selection, with each red point in the figure representing a certain resulting bounding-box size. As shown in these figures, the proposed approach provides an effective solution to the problem of bounding-box size selection.

The corresponding convolution kernel sizes can be obtained by using Eq. (9). Note that in Fig. 8, the sizes of some resulting bounding-boxes are larger than 150 × 150. In this situation, the dimension of the corresponding feature vector will be larger than $(150/4)^2 \times 4 = 5625$, which is a relatively high dimension for the classification problem in this work where features are simple and classifiers are liner. Consequently, it may increase the complexity in classification or even cause the problem of over-fitting, which is so-called curse of dimensionality [28]. In order to avoid these problems, we downsample the original image to t sizes and assign each resulting bounding-box to a certain resized image (see Fig. 2). More specifically, let $Fr(i)$ denote the ith scale factor which is used to resize the original. $Wr_{i,\,j}$ and $Hr_{i,\,j}$ are the resized form of resulting width and height of the size selection respectively. They can be written as follows:

$$\begin{cases} Wr_{i,j} = Wm_j \cdot Fr(i) \\ Hr_{i,j} = Hm_j \cdot Fr(i) \end{cases}, \tag{15}$$

where $Fr(i) \in (0,\ 1]$, $i = 1, 2, \ldots, t$ and $j = 1, 2, \ldots, q$. Let $Iw \times Ih$ be the ideal size of bounding-box, which directly determines the dimension of each feature vector. In this work, we choose 60 × 60. In this way, the dimension of corresponding feature vector is $(60/4)^2 \times 4 = 900$, which is an ideal dimension for feature computation and SVM classification. Having defined the ideal size of bounding-box, the distance between $[Wr_{i,\,j}, Hr_{i,\,j}]$ and $[Iw, Ih]$ can be formulated as follows:

$$Sp_{i,j} = sqrt((Wr_{i,j} - Iw)^2 + (Hr_{i,j} - Ih)^2). \tag{16}$$

Then each type of bounding-box can be assigned an optimum scale factor Fp_j:

$$Fp_j = Fr(\arg\min_i Sp_{i,j}). \tag{17}$$

In this way, the proposal features of each type of bounding-box can be reduced to a proper dimension to avoid the curse of dimensionality [28].

2.4. Multiple hyperplanes optimization for proposal generation

Rather than accuracy or false positive rate, proposal generation is primarily concerned with recall rate (true positive rate) [21]. In other words, if recall rate meets the requirements, thousands of

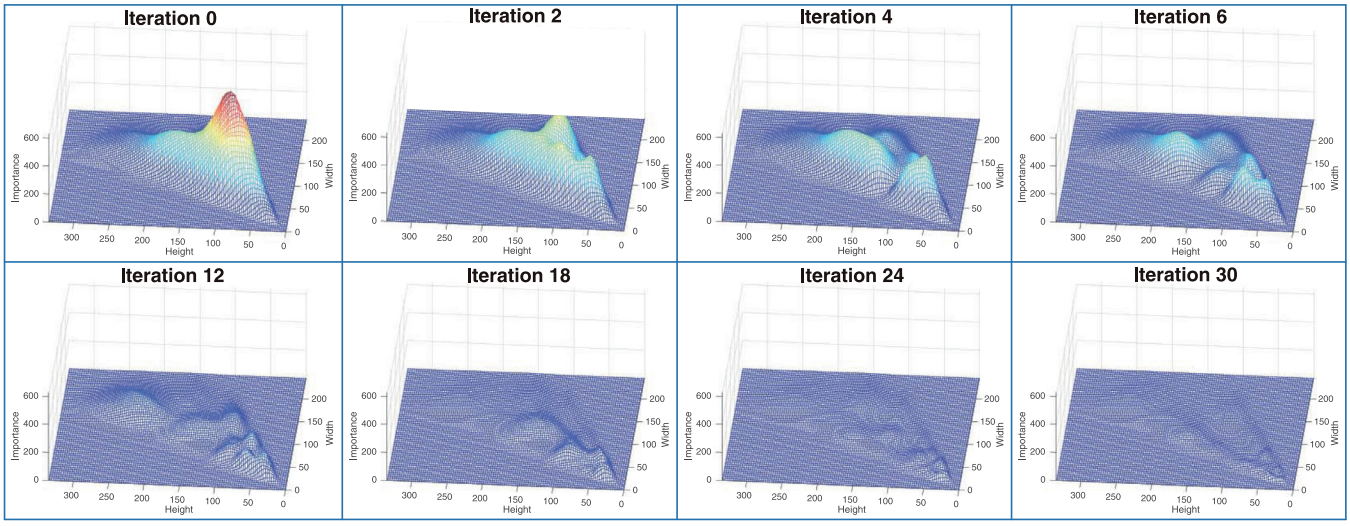


Fig. 9. The change of importance map with the increasing of iterations.

candidate windows are allowed. However, as a classifier designed for object detection, SVM [13] makes a trade-off between false positive rate and recall rate. In this work, we optimize the hyperplanes of multiple SVMs and make them more suitable for proposal generation with a high recall rate.

Assume that there are n positive samples in a training set, and m SVM classifiers (or convolution kernels) are constructed. In the proposal feature space, the distance between each positive sample and each SVM hyperplane can be defined as a matrix D:

$$D = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}, \tag{18}$$

where $d_{i,j}$ is the distance between the ith SVM hyperplane and the jth positive sample. We design a strategy to translate each hyperplane. Let $X = [x_1, x_2, \ldots, x_m]^{\mathrm{T}}$ denote the translation distances of all hyperplanes. After the translation, the original distance matrix D is transformed to $\hat{D}$:

$$\hat{D} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \tag{19}$$

$$= \begin{bmatrix} d_{11}+x_1 & d_{12}+x_1 & \cdots & d_{1n}+x_1 \\ d_{21}+x_2 & d_{22}+x_2 & \cdots & d_{2n}+x_2 \\ \vdots & \vdots & \cdots & \vdots \\ d_{m1}+x_m & d_{m2}+x_m & \cdots & d_{mn}+x_m \end{bmatrix}. \tag{19}$$

The maximum value of the jth column of $\hat{D}$ represents the maximum distance between the jth sample and all the SVM hyperplanes. If the maximum value is greater than zero, then the jth sample is classified as a positive sample by at least one hyperplane. Therefore, the maximum value of each column is able to indicate the recall rate, and we transform the maximum value to a continuous and differentiable function for further optimization. According to [33], for a set $\{z_1, z_2, \ldots, z_m\}$, the maximum function $F_m(Z) = \max\{z_1, z_2, \ldots, z_m\}$ can be smoothly approximated as:

$$\hat{F}_m(Z) = \frac{\sum_{i=1}^{m} z_i^{p+1}}{\sum_{i=1}^{m} z_i^{p}}, \tag{20}$$

where $p \geq 1$, $\sum z_i^p \neq 0$ and $z_i \geq 0$. The continuity and differentiability have been proofed by Yu [33]. As for the translated distance matrix $\hat{D}$, each element $d_{i,j}$ may not satisfy the condition of $d_{i,j} \geq 0$ (e.g., when the corresponding sample is classified as negative sample), and we refine Eq. (20) with natural exponential function. Thus, the maximum function of the jth column of $\hat{D}$ can be written as:

$$Fm_p(j) = \ln\left(\frac{\sum_{i=1}^{m} e^{(d_{ij}+x_i)\cdot(p+1)}}{\sum_{i=1}^{m} e^{(d_{ij}+x_i)p}}\right), \tag{21}$$

where p is used to control the approximation quality. A smaller p will result in a lower approximation precision, while a larger p is more likely to cause the problem of overflow in calculation. According to the reference [33], we choose $p = 40$ in this work. For more details on the tuning of parameter p, please refer to [33].

If $Fm_p(j) > 0$ for all samples $j = \{1, 2, \ldots, n\}$, then the recall rate would be 100%. However, the larger the value of $Fm_p(j)$, the larger the number of candidate windows, and therefore $Fm_p(j)$ should be optimized to a proper value. In this work, we design the loss function as:

$$J(x_1, \ldots, x_m) = \sum_{j=1}^{n} Fs_{\mu,\eta}(Fm_p(j)), \tag{22}$$

where:

$$Fs_{\mu,\eta}(x) = -e^{\frac{-(x-\mu)^2}{\eta}}, \tag{23}$$

where μ corresponds to the minimum of $Fs_{\mu,\eta}(x)$, and therefore we should set $\mu > 0$. For an one-dimensional case, the figure of $Fs_{\mu,\eta}(x)$ is illustrated in Fig. 10. The left part of μ (marked with blue) penalizes values of $Fm_p(j)$ less than 0, with the aim of improving the recall rate. The right part of μ (marked with red) penalizes large values of $Fm_p(j)$, with the aim of reducing the candidate windows. Moreover, unlike the parabola model, $Fs_{\mu,\eta}(x)$ is convergent to 0 when the maximum distance $Fm_p(j)$ is too large, which avoids the problem of overflow in the computer.

Having defined the loss function $J(x_1, \ldots, x_m)$, we optimize the model by gradient descent algorithm. As $\sum e^{(d_{ij}+x_i)p} \neq 0$, $J(x_1, \ldots, x_m)$ is differentiable for $x_i \in R$, we defined $Fc_p(j)$ as:

$$Fc_p(j) = \sum_{i=1}^{m} e^{(d_{ij}+x_i)p}. \tag{24}$$

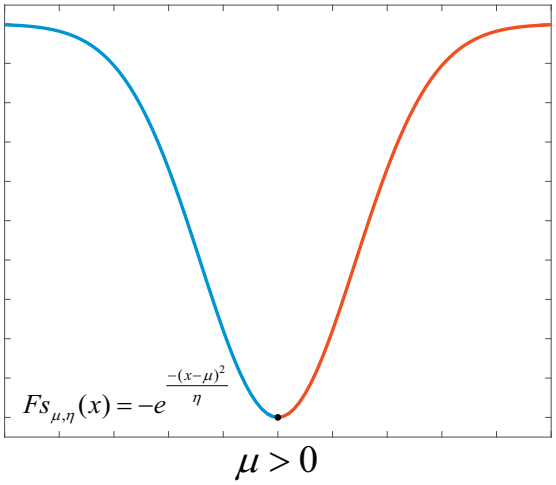


Fig. 10. One-dimensional case of $Fs_{\mu,\eta}(x)$. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

The gradient (differential) of J is denoted by:

$$\frac{\partial J}{\partial X} = \left[\frac{\partial J}{\partial x_1}, \frac{\partial J}{\partial x_2}, \ldots, \frac{\partial J}{\partial x_m}\right]^T, \tag{25}$$

where each element can be obtained:

$$\frac{\partial J}{\partial x_k} = \sum_{j=1}^{n} Fd_p(j) \cdot Fg_{p,\mu,\eta}(j), \tag{26}$$

where:

$$Fd_p(j) = \frac{e^{(d_{kj}+x_k)(p+1)} \cdot (p+1)}{Fc_p(j)} - \frac{e^{(d_{kj}+x_k)p} \cdot p \cdot Fc_{p+1}(j)}{Fc_p(j)^2}, \tag{27}$$

$$Fg_{p,\mu,\eta}(j) = \frac{-2Fc_p(j) \cdot Fs_{\mu,\eta}(Fm_p(j)) \cdot (Fm_p(j) - u)}{\eta \cdot Fc_{p+1}(j)}. \tag{28}$$

Having obtained the gradient (differential) of J, as shown in Eq. (29), for iteration step length γ, the translated distance X is updated at each iteration until convergence or until the maximum iterations have been reached; then the optimized translated distance $\hat{X}$ is obtained.

$$X_{t+1} = X_t + \gamma \frac{\partial J}{\partial X_t}. \tag{29}$$

In this way, multiple hyperplanes are optimized by adding the translated distance $\hat{X}$, which is able to achieve a high recall rate and an appropriate number of candidate windows. Further validation is given in Section 4.

Finally, a fusion step is needed for multiple overlapping proposals. For the same SVM classifier, the proposal score can be represented by the corresponding sample distance to the hyperplane. It can be assumed that the higher the proposal score, the higher the probability that the proposal window will be a true positive [34]. Thus, for the same SVM classifier (or convolution kernel set), we merge multiple overlapping proposals by non-maximum suppression [35]. However, the proposal scores are not comparable between different SVM classifiers. To solve this problem, we merge proposals from different classifiers by mean shift algorithm [34].

3. Detection layer

In this section, we describe our proposed detection layer, which works on the potential bounding-boxes generated by the proposal layer. In the detection layer, we first propose an MPGA-based convolutional neural network (CNN) module to balance the performance and computational complexity of deep convolutional neural networks. Then, we design a multi-frame fusion strategy to utilize temporal information and improve detection performance.

3.1. Constructing convolutional neural networks using MPGA

Recently, convolutional neural networks (CNN) [15] have been extensively explored and have achieved remarkable success in object detection tasks. However, the structures of CNN (e.g., the number of feature maps at each layer) are usually determined in an

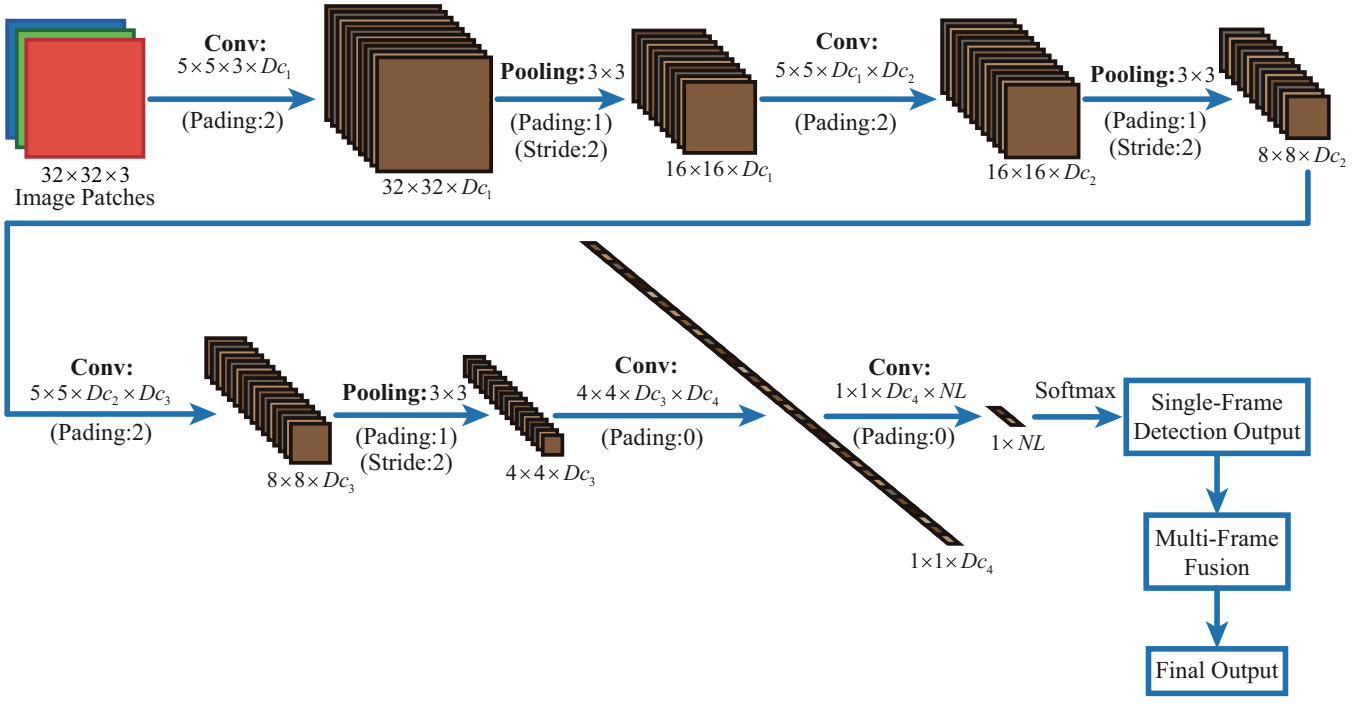


Fig. 11. Architecture of the detection layer.

empirical way: multiple CNNs with different structures are developed and the most appropriate is selected through high-volume performance comparisons [36], and this approach increases the cost of CNN development. In this work, we construct an adaptive CNN by using a multiple population genetic algorithm (MPGA) [37] which is able to determine the number of feature maps at each layer and makes a trade-off between performance and computational complexity.

In order to control the model complexity, we first construct a basic CNN framework, which will be completed by MPGA. As shown in Fig. 11, the basic CNN framework consists of an input layer, 5 convolution layers, 3 pooling layers and an output layer (softmax). The input layer is the RGB color image patch of 32 × 32 pixels, which is resized from object proposals.

In the convolution layers, the feature maps are convolved by a set of multi-dimensional filter banks. Let Dc_i denote the number of filter banks at the ith convolution layer ($i = 1, 2, 3, 4, 5$), and NL denote the number of output labels, where:

$$\begin{cases} Dc_0 = 3 \\ Dc_5 = NL \end{cases}. \tag{30}$$

Then the dimensions of the feature map at each layer are determined (see Fig. 11). The ith convolution layer computes the convolution of the input map $\vec{x}_{i-1}$ with the filter banks $\vec{w}_i$ and biases $\vec{b}_i$, where:

$$\begin{cases} \vec{x}_{i-1} \in \mathbb{R}^{W'_i \times H'_i \times Dc_{i-1}} \\ \vec{w}_i \in \mathbb{R}^{W''_i \times H''_i \times Dc_{i-1} \times Dc_i} \\ \vec{b}_i \in \mathbb{R}^{1 \times Dc_i} \end{cases}. \tag{31}$$

The output of the ith convolution layer can be formulated as:

$$\hat{x}_i(m', n', d') = b_i(d') + \sum_{m''=1}^{W''} \sum_{n''=1}^{H''} \sum_{d''=1}^{Dc_{i-1}} M_{m'',n'',d'',d'}, \tag{32}$$

where:

$$M_{m'',n'',d'',d'} = w_i(m'', n'', d'', d') \cdot x_{i-1}(m'' + m' - 1, n'' + n' - 1, d''). \tag{33}$$

After each convolution layer, we employ a rectified linear unit (ReLU) [38] to improve the performance of the CNN, which is defined as:

$$\tilde{x}_i(m', n', d') = \max\left\{0, \hat{x}_i(m', n', d')\right\}. \tag{34}$$

After obtaining feature maps from the convolution layer and ReLU, we apply a max-pooling operator on each input feature channel to reduce the size of the feature map and extract translation invariant features [39]. The max-pooling is defined as:

$$x_i(m', n', d') = \max_{m' \le m'' < m'+p, n' \le n'' < n'+p} \{\tilde{x}_i(m'', n'', d')\}, \tag{35}$$

where p controls the spatial pooling size. A larger pooling size will lead to more information loss in pooling stage. According to structures of other successful deep networks [15,18,19,40], we use $p = 3$ in this work.

Finally, a softmax function is applied on the feature map of the last layer, which is given by:

$$y_i(m) = \frac{e^{x_i(m)}}{\sum_{m'=1}^{NL} e^{x_i(m')}}. \tag{36}$$

In this basic CNN framework, the performance and computational complexity mainly depend on the number of filter banks Dc_i at each convolution layer, which also determines the dimensions of the output feature maps. In most previous works [15,39,40], the number of filter banks is determined by experience and large experiments. In order to balance the performance and computational complexity, we employ a multiple population genetic algorithm (MPGA) to seek the optimal Dc_i heuristically ($i = 1, 2, 3, 4, 5$).

MPGA is an optimization algorithm inspired by biology, whose object function is not required to be continuous or differentiable. Moreover, MPGA is more likely than traditional genetic algorithms to obtain global optimal solutions [41]. In MPGA, the variables that need to be optimized are first encoded to chromosomes, and then two or more populations that contain several chromosomes are generated. The populations evolve in parallel until they converge to a stable status. In this way, the optimal solutions are obtained.

For the optimization problem in this work, since Dc_0 and Dc_5 are determined (see Eq. (30)), we first transform $\vec{Dc} = [Dc_1, Dc_2, Dc_3, Dc_4]$ to binary codes which act as chromosomes of MPGA. Then Np populations are generated (each population is made up of Nc chromosomes). The fitness function $Fi(\vec{Dc}, e_r)$ of MPGA is defined as:

$$Fit(\vec{Dc}, e_r) = -\frac{Oc(\vec{Dc}) + Ks \cdot (e_r - e_p)^2}{1 + Ks}, \tag{37}$$

where e_r and e_p are the training and expected error respectively. $Oc(\vec{Dc})$ is the structure complexity, which is formulated as:

$$Oc(\vec{Dc}) = \sum_{i=1}^{4} Sc_i^2 \cdot Dc_{i-1} \cdot Dc_i, \tag{38}$$

where Sc_i is the size of the convolution filter of the ith convolution layer. Ks is the factor that controls the balance of structure complexity and performance of CNN.

At each iteration, each population experiences the processes of selection, reproduction and mutation via the fitness function [37]. Ultimately, the chromosome with maximum fitness value will be found. The optimal $\vec{Dc} = [Dc_1, Dc_2, Dc_3, Dc_4]$ can be obtained by decoding the optimized chromosome.

Meanwhile, in order to overcome the overfitting of CNN, inspired by Xu et al. [42], we corrupt training images to different degrees and in this way train the final CNN model. Corrupting methods include adding Gaussian noise and applying Gaussian smoothing filters to the training images.

3.2. Multi-frame fusion

The camera mounted on a robot can provide image sequence which contains rich temporal information. The goal of multi-frame fusion is to utilize this temporal information to improve performance in object detection. As we know, the probability of a non-object region giving a false positive in several consecutive frames is much lower than that in a single frame. Similarly, if an image region is detected to be positive in several consecutive frames, then there is a high probability that the image region is an object region. In other words, multi-frame fusion can effectively improve the true positive rate and yet reduce the false positive rate. Moreover, detection in different frames can be smoothed by multi-frame fusion.

Most previous works [3,43,44] utilize temporal information in object detection by employing a tracking procedure. Tracking-Learning-Detection (TLD) [44] is one of the most robust and efficient methods of object tracking [3]. It consists of a Lucas–Kanade (LK) local tracker [43], a randomized forest detector [45] and a P-N Learning algorithm [46].

For the multi-frame object detection problem, tracking is usually used to avoid exhaustive candidate searching. In this work, object candidates have already been efficiently obtained by the proposal layer (Section 2) and MPGA-based CNN (Section 3.1), and accordingly multi-frame fusion in this work focuses on validation rather than tracking. Consequently, we use the TLD method [44] as a multi-frame predictor which predicts the positions of candidates

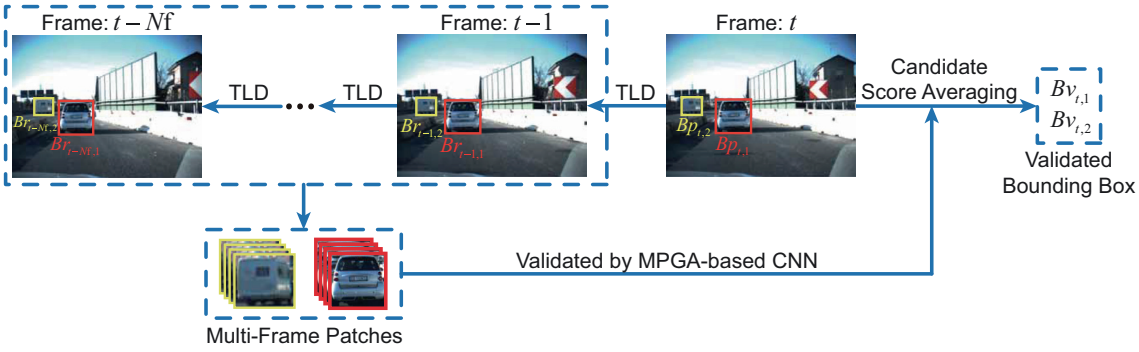


Fig. 12. Illustration of multi-frame validation.

in past N_f consecutive frames according to the candidate regions in current frame. The candidate regions are then validated by applying an MPGA-based CNN to corresponding candidates in past N_f frames. In this way, the multi-frame fusion model is able to tolerate changes of scale, illumination and background clutter due to the strong representation of the deep convolution neural network (MPGA-based CNN). In addition, as object locations are estimated in every frame, detections do not drift over a longer period or fail when objects disappear from the camera view. In this work, multi-frame fusion consists of multi-frame validation and bounding-box smoothing.

The procedure of multi-frame validation is illustrated in Fig. 12. For the tth frame (current frame), each object candidate is assigned a TLD [44]. In this work, the TLD is initialized according to the position of the candidate bounding-box, which is defined as:

$$\vec{Bp}_{t,k} = \begin{bmatrix} xp_{t,k} & yp_{t,k} & wp_{t,k} & hp_{t,k} \end{bmatrix}^{\mathrm{T}}, \tag{39}$$

where $(xp_{t,k}, yp_{t,k})$ and $(wp_{t,k}, hp_{t,k})$ respectively denote the center coordinate and size (width and height) of the kth object bounding-box in the tth frame.

As shown in Fig. 12, the bounding-boxes in past N_f frames are predicted by TLD. The kth object bounding-box in the $(t-i)$th frame is denoted by $\vec{Br}_{t,k-i}$, where $i = 1, \ldots, Nf$. Having obtained the bounding-boxes in past N_f frames, the corresponding multi-frame image patches are validated by an MPGA-based CNN, which returns a confidence score for each image patch. Let $Sp_{t-1,k} \sim Sp_{t-Nf,k}$ denote the confidence scores of the kth object candidate in past N_f frames, and $Sp_{t,k}$ denotes the confidence score in the tth frame (current frame). The validated confidence score $Sv_{t,k}$ is defined as:

$$Sv_{t,k} = \frac{1}{Nf+1} \sum_{i=0}^{Nf} Sv_{t-i,k}. \tag{40}$$

The procedure of multi-frame validation outputs each validated bounding-box $\vec{Bv}_{t,k}$ according to its confidence score $Sv_{t,k}$. That is, if $Sv_{t,k} > ST$ ($ST = 0.7$ in this work), then the bounding-box $\vec{Bv}_{t,k} = \vec{Bp}_{t,k}$ is output. Otherwise, $\vec{Bp}_{t,k}$ is rejected as the false positive. After that, a smoothing procedure is applied to the validated bounding-box according to its predicted bounding-box in past N_f frames. We construct a temporal matrix which consists of the validated bounding-box $\vec{Bv}_{t,k}$ and predicted bounding-boxes $\vec{Br}_{t-1,k} \sim \vec{Br}_{t-Nf,k}$. The temporal matrix can be written as:

$$\begin{aligned} \vec{Bm}_{t,k} &= \begin{bmatrix} \vec{Bv}_{t,k} & \vec{Br}_{t-1,k} & \cdots & \vec{Br}_{t-Nf,k} \end{bmatrix} \\ &= \begin{bmatrix} xv_{t,k} & xr_{t-1,k} & \cdots & xr_{t-Nf,k} \\ yv_{t,k} & yr_{t-1,k} & \cdots & yr_{t-Nf,k} \\ wv_{t,k} & wr_{t-1,k} & \cdots & wr_{t-Nf,k} \\ hv_{t,k} & hr_{t-1,k} & \cdots & hr_{t-Nf,k} \end{bmatrix}. \end{aligned} \tag{41}$$

We then define a smoothing weight vector $\vec{Mc}$, where each element denotes a weight for each frame:

$$\vec{Mc} = \begin{bmatrix} mc_1 & mc_2 & \cdots & mc_{Nf+1} \end{bmatrix}^{\mathrm{T}}. \tag{42}$$

As the predicted bounding-box in frames that are close to the current frame has a stronger relationship with the current object position compared with the bounding-boxes in frames that are further removed from the current frame, the weight for each frame is formulated as:

$$mc_j = \frac{(Nf - j + 2)^2}{\sum_{j=1}^{Nf+1} j^2}. \tag{43}$$

Finally, the smoothed bounding-box of the kth object candidate $\vec{Bs}_{t,k}$ can be obtained by matrix multiplication:

$$\vec{Bs}_{t,k} = \vec{Bm}_{t,k} \times \vec{Mc}, \tag{44}$$

4. Experimental results

The goal of this section is to validate the effectiveness and efficiency of the proposed object detection system in robotic applications. The experiments in this work mainly focus on vehicle and pedestrian detection, which is widely used in a range of robotic applications such as service robot interaction [1], autonomous driving [2], and collision avoidance [3]. As multi-frame fusion is an essential component of our proposed system, sequence-based datasets are needed. Accordingly, we evaluate the performance of the proposed approach on two widely used sequence-based datasets: the TME motorway dataset [3] and the ETHZ pedestrian dataset [47]. The former is composed of a large set of video sequences which are captured by a camera mounted on top of a vehicle windshield. These sequences are annotated by laser scanner and consist of "daylight" and "sunset" subsets, which allow the performance of the proposed system to be evaluated in various lighting conditions and traffic situations [3]. The latter dataset involves several video sequences collected from mobile platforms moving through busy pedestrian zones [47]. The sequences capturing pedestrians are manually annotated.

In the experiments, we first validate the performance of the proposal layer. This validation includes an evaluation of the importance of each component (e.g., multi-scale feature extraction, multiple hyperplanes optimization and data-driven kernel size selection) and a comparison with some state-of-the-art approaches to proposal generation. Then, the MPGA-based convolutional neural network (MPGA-based CNN) is evaluated, including its parameters settings, training details and classification performance. Finally, we validate the effectiveness and efficiency of the whole object detection system in comparison with some recently published state-of-the-art algorithms. Experiments in this work are performed on

Table 1
The evaluations of proposal layer.

Method	TME motorway dataset [3]			ETHZ pedestrian dataset [47]		
	Recall@2000	N@90%	Time	Recall@2000	N@90%	Time
Emp-Ker(15)	80.2%	4282	**0.1 s**	79.7%	3157	**0.1 s**
Emp-Ker(20)	82.1%	3418	**0.1 s**	82.5%	2439	**0.1 s**
Emp-Ker(25)	82.9%	2773	0.2 s	85.8%	2076	0.2 s
Emp-Ker(30)	84.7%	2165	0.2 s	87.1%	1634	0.2 s
Driven-Ker(15)	81.9%	3279	**0.1 s**	81.6%	2421	**0.1 s**
Driven-Ker(20)	84.6%	2425	**0.1 s**	83.2%	1863	**0.1 s**
Driven-Ker(25)	87.7%	1973	0.2 s	86.9%	1647	0.2 s
Driven-Ker(30)	90.8%	1695	0.2 s	89.3%	1265	0.2 s
Driven-Ker(30) + MHO(0.1)	93.2%	**1217**	0.2 s	92.7%	**1076**	0.2 s
Driven-Ker(30) + MHO(0.4)	93.8%	1934	0.2 s	93.3%	1856	0.2 s
Driven-Ker(30) + MHO(0.7)	94.3%	2776	0.2 s	93.8%	2749	0.2 s
Driven-Ker(30) + MHO(1.0)	**95.1%**	3853	0.3 s	**94.2%**	3558	0.3 s
Edge Boxes [21]	92.5%	1754	0.2 s	91.6%	1584	0.2 s
Selective Search [20]	92.7%	2276	7.3 s	92.0%	2362	8.2 s

a laptop which contains an Intel i7-4700MQ processor running at 2.4 GHz. Except for off-line training, all the detection processes in the experiments run without GPU support. For the following experiments, we use an intersection-over-union (IoU) threshold of 0.7 to determine the correctness of detection.

4.1. The proposal layer

As shown in Table 1, quantitative evaluations of the proposal layer are provided using three types of metrics: recall rate (Recall), average number of proposals per frame (N) and run-time per frame (Time). Let *TP* and *FN* be the number of positive samples that are correctly and incorrectly detected, respectively. Recall can be written as:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{45}$$

Eq. (45) demonstrates that recall rate directly reflects the proportions of positive samples that are detected correctly. As the fewer the proposals, the more efficient the subsequent image processing tasks will be, the average number of proposals per frame (N) is also an important metric in a proposal generation algorithm. Furthermore, run-time per frame (Time) represents the efficiency of a proposal generation algorithm.

In Table 1, Recall@2000 denotes the average recall using proposals at the number of 2000. N@90% represents the average number of proposals needed to achieve 90% recall per frame. All the recall rates in Table 1 are computed using the IoU threshold of 0.7.

In the first column of Table 1, different components of the proposal layer with different settings are evaluated. Emp-Ker(q) and Driven-Ker(q) stand for the traditional empirical kernel size selection approach and our data-driven kernel size selection approach respectively. q is the number of bounding-box types. MHO(μ) denotes the multiple hyperplanes optimization procedure with mid-value μ. Furthermore, we compare our method with two state-of-the-art proposal generation methods: Edge Boxes [21] and Selective Search [20]. Edge Boxes [21] detects proposals by edge-based object boundary estimation and has been widely used in vehicle detection. Selective Search [20] generates proposals by merging superpixels and has been widely used in many state-of-the-art object detection systems.

For both datasets, as can be seen from Table 1, the performance (Recall and number of proposals) of Emp-Ker and Driven-Ker is improved by increasing the number of bounding-box types q. This is because more object bounding-box types can be covered and fewer false positive detections arise when the number of bounding-box types rises. At the same time, the recall rate and the proposal count of Driven-Ker are respectively higher and lower than that of Emp-Ker, due to the fact that Driven-Ker selects the kernel size in a data-driven way that tends to cover more valid object sizes. Table 1 also indicates that performance could be improved dramatically by applying multiple hyperplanes optimization (MHO), whose optimization goal is to improve the recall rate and reduce the number of candidate windows. Although recall rate rises with the increase of the mid-value μ, the number of proposals grows, in this way, the efficiency of the subsequent image processing tasks would be decreased. Moreover, as data-driven kernel size selection (Driven-Ker) and multiple hyperplanes optimization (MHO) are both off-line procedures, they have no direct impact on run-time. Table 1 shows that the run-time of the proposal layer mainly depends on the number of bounding-box types (q), which determines the number of 2D convolutions. Furthermore, Table 1 demonstrates that the whole proposal layer (Driven-Ker + MHO) performs much better than the other two state-of-the-art methods [20,21] in terms of recall rate, number of proposals and run-time.

As types of vehicle bounding-box (car, bus, truck, jeep, and so on) are more diverse than types of pedestrian bounding-box, as can be seen from Table 1, the average number of proposals generated with the TME motorway dataset [3] is larger than with the ETHZ pedestrian dataset [47]. By the same token, the advantage of Driven-Ker is more obvious with the TME motorway dataset than with the ETHZ pedestrian dataset. As vehicles usually have rigid shapes while pedestrians do not, the edges of vehicle boundaries are more stable than those of pedestrian boundaries, and for this reason, the Edge Boxes algorithm [21] performs better with the TME motorway dataset than that with the ETHZ pedestrian dataset. In contrast, our proposed approach does not only rely on object boundaries and performs equally well with both datasets.

We make a trade-off amongst recall rate, average number of proposals and run-time, and use Driven-Ker(30) + MHO(0.1) in the experiments in Section 4.3. Experimental results in Table 1 show that these parameter are able to balance the performance and the number of proposals.

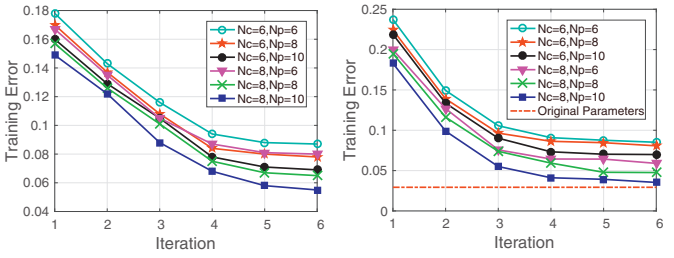
4.2. MPGA-based CNN

Our MPGA-based CNN aims at constructing an adaptive CNN by using a multiple population genetic algorithm (MPGA) [37], which is able to determine the number of feature maps at each layer and make a trade-off between performance and computational complexity.

In the experiments, positive samples are obtained from the ground-truth of TME motorway dataset [3] and ETHZ pedestrian dataset [47]. Negative samples are obtained according to their distance from the corresponding SVM hyperplane. More specifically,

Table 2
The training epochs of MPGA-based CNN.

Nt \ Np / Nc	2	4	6	8	10
4	40	80	120	128	200
6	48	144	108	288	240
8	80	192	144	256	480



(a) The structure of ours (b) The structure of AlexNet [15]

Fig. 13. The training error of MPGA-based CNN.

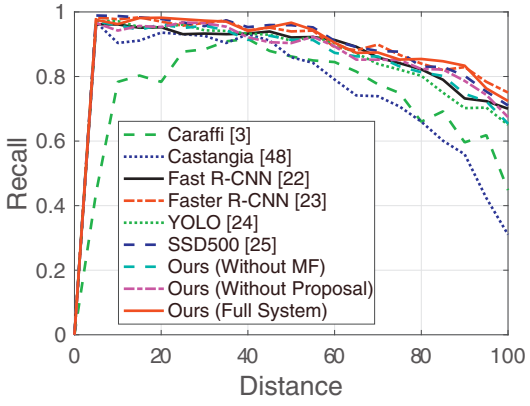


Fig. 14. Recall in function of vehicle distance on TME motorway dataset.

we first sort the negative proposals according to their distances from the corresponding SVM hyperplanes in descending order; then, negative samples are selected in that order. At the same time, a balance is kept between the numbers of positive and negative samples in order to avoid bias prediction.

We set $NL = 3$ to enable the MPGA-based CNN to output three types of labels (0: nothing, 1: vehicle, 2: pedestrian). To improve the efficiency of MPGA, the range of $Dc_1 \sim Dc_4$ is limited to:

$$Dc_i \in [4, 64] \ \wedge \ Dc_i \in N, \ i = 1, 2, 3, 4 \tag{46}$$

For a quantitative comparison, we denote the number of initial populations and the number of chromosomes in each population as Np and Nc respectively. Nt represents the number of CNN training epochs. The Nt of greedy searching method is:

$$Nt = (64 - 4 + 1)^4 = 13,845,841, \tag{47}$$

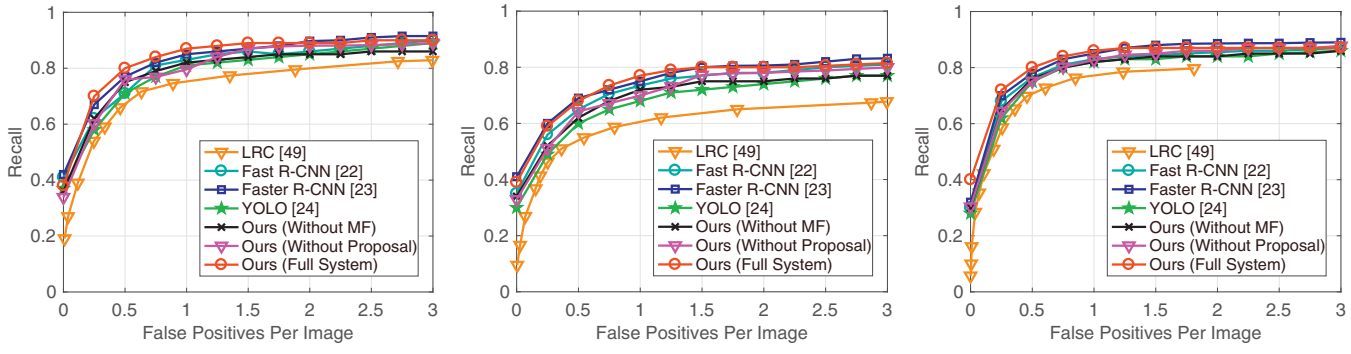
which is a huge amount of CNN training epochs and hard to achieve in GPU. We perform an experiment to validate the efficiency of our MPGA-based training strategy. As can be seen from Table 2, the number of CNN training epochs Nt of our proposed method is much lower than that of the greedy searching method (13,845,841). Consequently, the proposed method spends much less time in finding the optimum number of feature maps at each layer that balances the performance and computational complexity.

Fig. 13 illustrates the training error of MPGA-based CNNs with different settings when increasing the iterations of MPGA. For each iteration, the number of training epochs of the MPGA-based CNN is $Nc \times Np$. As shown in Fig. 13, the training errors are reduced with the increase of MPGA iterations. This is due to the increase in the number of repetitions of selection, reproduction and mutation which accompany increases in the number the iterations. Moreover, although increases in population size (Np) and in the number of chromosomes in each population (Nc) would result in an increase in training epochs of MPGA-based CNN (see Table 2), which would lead to a longer training time, the training error can still be reduced effectively (see Fig. 13). This is because local optimal solutions can be avoided by increasing the populations (Np) and chromosomes in each population (Nc).

4.3. Full system

We evaluate the performance of our object detection system in comparison with some state-of-the-art methods, which can be categorized into two types: deep learning based methods (Fast R-CNN [22], Faster R-CNN [23], YOLO [24], SSD500 [25] and our proposed method) and hand-craft methods (Caraffi [3], Castangia [48] and LRC [49]).

Fast R-CNN [22] employs Selective Search (fast mode) [20] to generate proposals and then runs a ROI pooling layer and convolution neural network (CNN) on these proposals to achieve high accuracy in object detection. Caraffi [3] is an efficient hand-craft vehicle detection method based on WaldBoost detectors [50] and TLD trackers [44]. Castangia [48] achieves real-time vehicle detection by employing integral channel features, AdaBoost soft-cascades and unscented Kalman filters. LRC [49] makes use of local response context descriptors to achieve robust pedestrian detection. In addition, we evaluate the importance of the proposal layer and multi-frame fusion in our object detection system. In Figs. 14, 15 and Table 3, "Ours (without proposal)" and "Ours (without MF)"



(a) ETH-01 (b) ETH-02 (c) ETH-03

Fig. 15. Performance evaluation on ETHZ pedestrian dataset.

Table 3
Run-time analysis on TME motorway dataset.

Method	Average run-time
Caraffi [3]	0.1 s
Castangia [48]	0.05 s
Fast R-CNN [22]	2.7 s
Ours (without proposal)	43 s
Ours (without MF)	0.23 s
Ours (full system)	0.25 s

Table 4
Run-time analysis of each on-line step on TME motorway dataset.

Step	Average run-time
Feature extraction	0.12 s
Multi-kernel convolution	0.08 s
CNN (path forward)	0.03 s
Multi-frame fusion	0.02 s
Total	0.25 s

denote our proposed system without the proposal layer and without the multi-frame fusion respectively. "Ours (full system)" represents our whole object detection system including proposal layer, MPGA-based CNN and multi-frame fusion.

As can be seen from Figs. 14, 15 and Table 3, although the average run-times of deep learning based methods (Fast R-CNN [22] and our proposed method) are longer than those of hand-craft methods (Caraffi [3], Castangia [48] and LRC [49]), the deep learning based methods in the experiment perform much better than these hand-craft methods in metrics of recall rate and false positive rate. Moreover, as shown in Figs. 14 and 15, the advantages of deep learning based methods are especially obvious in more challenging situations (e.g., with ETH-02 and long distance vehicles). These advantages are a result of the strong contribution of a deep convolution neural network.

Although there are several computational steps in our work, only four of them are on-line steps. Table 4 validates the efficiency of these on-line parts. Other parts such as multiple hyperplanes optimization, data-driven kernel size selection and MPGA-based training are performed off-line. Besides, the run-time of our proposed method is much less than that of Fast R-CNN [22] (see Table 3). Meanwhile, our full system exhibits high detection performance which is comparable with that of Fast R-CNN [22] (see Figs. 14 and 15). In other words, owing to the efficient proposal layer and MPGA-based CNN, our proposed method is able to balance detection performance and computational complexity, which is extremely important in many robotic applications.

The proposal layer can not only avoid exhaustive candidate searching across the given image and reduce run-time significantly (see Table 3), but also has a beneficial effect on detection performance (see Fig. 14). This is due to the fact that the proposal layer screens out a large number of negative samples, and consequently reduces the complexity of the classification task to be performed by the MPGA-based CNN.

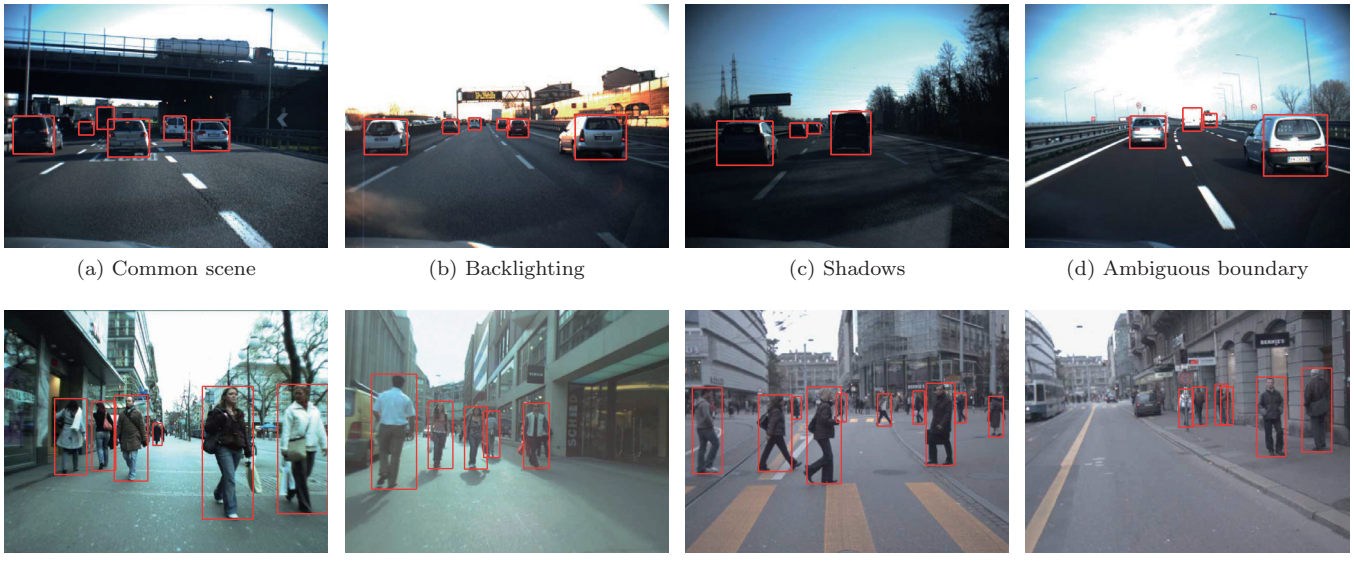
As shown in Figs. 14 and 15, multi-frame fusion in our work is effective in improving the recall rate and reducing the false positive rate. Since TLD [44] is an efficient method for multi-frame fusion, there is no obvious increase in run-time when integrating the multi-frame fusion procedure (see Table 3).

Besides, experimental results shown in Fig. 14 indicate that our proposed approach performs better in small objects detection compared with Fast R-CNN [22],YOLO [24] and SSD500 [25]. This is owing to our effective proposal layer which is based on dense sliding windows paradigm and is able to detect objects in low resolution.

Moreover, the qualitative results shown in Fig. 16 demonstrate that our proposed object detection system exhibits high robustness in challenging scenarios involving backlighting, shadows, different viewpoints, overlapping objects or ambiguous boundaries. In particular, object boundaries may be ambiguous when the appearance of an object border and its surroundings are similar. In this case, some grouping proposal methods (e.g., Selective Search [20]) are unsuitable since they rely on region segmentation and it is extremely difficult to segment object regions from their surroundings under this condition. To deal with ambiguous boundaries, our proposed approach generates proposals in a data-driven way, and thus is able to adapt the situation.

5. Conclusions and future works

In this paper, we have presented an efficient deep learning-based object detection system for robotic applications. To efficiently generate potential object bounding-boxes in an image, we first propose a novel proposal layer that consists of multiple



(a) Common scene (b) Backlighting (c) Shadows (d) Ambiguous boundary

(e) Common scene (f) Backlighting (g) Sideview (h) Overlapping objects

Fig. 16. Qualitative analysis of our proposed method.

hyperplanes optimization, data-driven kernel size selection, multi-scale feature extraction and multi-kernel convolution. As multiple hyperplanes optimization and data-driven kernel size selection are both off-line procedures, the proposal layer is able to work efficiently and reliably. Then, we construct a robust detection layer which works on the proposal layer. The detection layer involves an MPGA-based convolutional neural network (CNN) and a TLD-based multi-frame fusion procedure; the former is proposed to balance the performance and computational complexity of a deep convolutional neural network, and the latter utilizes temporal information to improve detection performance. Unlike most deep learning based approaches, which rely on GPU, all the on-line processes in our system are able to run efficiently without GPU support.

In the experiments, we validate each component of our proposed object detection system and compare the system with some recently published state-of-the-art object detection algorithms on widely used datasets. Even under challenging scenarios with shadows, backlighting, different viewpoints, overlapping objects or ambiguous boundaries, the proposed system is proved to be both efficient and effective in object detection.

Our future works will focus on implementing the proposed object detection system with FPGA (field programmable gate array) in robotic applications. As the on-line processes of our system, such as convolution-based proposal generation and MPGA-based CNN, can be run largely in parallel as each convolution is an independent computation, the system is suitable for implementation with FPGA using pipeline architecture. Using this approach, the system can achieve real-time performance without significant extra effort. Furthermore, compared with GPU based systems, FPGA based systems have much lower power and space consumption. Thus, when implemented with FPGA, our system will be easier to integrate into energy and space-constrained robot systems.

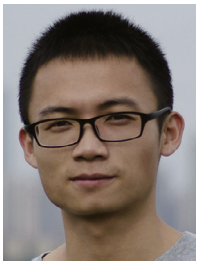
Acknowledgment

This work is supported by the National Natural Science Foundation of China (Grant No. 61473303).

References

[1] J.R. Terven, B. Raducanu, M.E.M. de Luna, J. Salasa, Head-gestures mirroring detection in dyadic social interactions with computer vision-based wearable devices, Neurocomputing 175 (2016) 866–876.
[2] A. Gepperth, B. Dittes, M.G. Ortiz, The contribution of context information: a case study of object recognition in an intelligent car, Neurocomputing 94 (2012) 77–86.
[3] C. Caraffi, T. Vojíř, J. Trefný, J. Šochman, J. Matas, A system for real-time detection and tracking of vehicles from a single car-mounted camera, in: Proceedings of the IEEE Intelligent Transportation Systems Conference, 2012, pp. 975–982.
[4] X. Zhong, X. Zhong, X. Peng, Robots visual servo control with features constraint employing Kalman-neural-network ltering scheme, Neurocomputing 151 (2015) 268–277.
[5] M. Bertozzi, A. Broggi, A. Fascioli, Vision-based intelligent vehicles: State of the art and perspectives, J. Robotics Autonomous Syst. 32 (1) (2000) 1–16.
[6] J. Tan, J. Li, X. An, H. He, Robust curb detection with fusion of 3d-Lidar and camera data, Sensors 14 (5) (2014) 9046–9073.
[7] D. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (60) (2004) 91–110.
[8] Y. Ke, R. Sukthankar, PCA-SIFT: a more distinctive representation for local image descriptors, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2004, pp. 506–513.
[9] H. Baya, A. Essa, T. Tuytelaarsb, L.V. Goola, Speeded-up robust features (SURF), Comput. Vis. Image Underst. 110 (3) (2008) 346–359.
[10] P. Viola, M.J. Jones, Robust real-time face detection, Int. J. Comput. Vis. 57 (2) (2004) 137–154.
[11] P. Dollár, R. Appel, S. Belongie, P. Perona, Fast feature pyramids for object detection, IEEE Trans. Pattern Anal. Mach. Intell. 36 (8) (2014) 1532–1545.
[12] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2005, pp. 886–893.
[13] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[14] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Trans. Pattern Anal. Mach. Intell. 32 (9) (2010) 1627–1645.
[15] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1–9.
[16] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing (2016) 0925–2312.
[17] T. Wang, D.J. Wu, A. Coates, A.Y. Ng, End-to-end text recognition with convolutional neural networks, in: Proceedings of the International Conference on Pattern Recognition, 2012, pp. 1051–4651.
[18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[19] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[20] J. Uijlings, K. van de Sande, T. Gevers, A. Smeulders, Selective search for object recognition, in: Proceedings of the International Journal of Computer Vision, 2013, pp. 154–171.
[21] C.L. Zitnick, P. Dollár, Edge boxes: locating object proposals from edges, in: Proceedings of the European Conference on Computer Vision, 2014, pp. 391–405.
[22] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[23] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Adv. Neural Inf. Process. Syst. (2015) 91–99.
[24] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 21–37.
[26] G. Cheng, P. Zhou, J. Han, RIFD-CNN: rotation-invariant and fisher discriminative convolutional neural networks for object detection, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 2884–2893.
[27] X. Zeng, W. Ouyang, B. Yang, J. Yan, X. Wang, Gated bi-directional CNN for object detection, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 354–369.
[28] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik, Feature selection for SVMS, Adv. Neural Inf. Process. Syst. 13 (2010) 668–674.
[29] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, Y. Wang, Intel math kernel library, in: High-Performance Computing on the Intel Xeon Phi, 2014, pp. 167–188.
[30] R. van de Geijn, K. Goto, Blas (basic linear algebra subprograms), in: Encyclopedia of Parallel Computing, 2011, pp. 157–164.
[31] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: ACM Proceedings of the 22nd ACM International Conference on Multimedia, 2014, pp. 675–678.
[32] P. Dollár, S. Belongie, P. Perona, The fastest pedestrian detector in the west, in: Proceedings of the British Machine Vision Conference, 2010, pp. 1–11.
[33] Z. Yu, An efficient method for solving nonlinear unconstrained min-max problem, Xi'An University of science and technology, Xi'An, China, 2011 Master's thesis.
[34] N. Dalal, Finding people in images and videos, Institut National Polytechnique de Grenoble, France, 2006 Ph.d. thesis.
[35] P. Dollár, Z. Tu, P. Perona, S. Belongie, Integral channel features, in: British Machine Vision Conference, 2009, pp. 7–10.
[36] Y. Zhang, D. Zhao, J. Sun, G. Zou, W. Li, Adaptive convolutional neural network and its application in face recognition, Neural Process. Lett. 43 (2016) 389–399.
[37] E. Cantu-Paz, D.E. Goldberg, Efficient parallel genetic algorithms: theory and practice, Comput. Methods Appl. Mech. Eng. 186 (2000) 221–238.
[38] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the International Conference on Machine Learning, 2010, pp. 807–814.
[39] H.-M. Moon, C.H. Seo, S.B. Pan, A face recognition system based on convolution neural network using multiple distance face, in: Soft Computing, 2016, pp. 1–8.
[40] Y. Zeng, X. Xu, Y. Fang, K. Zhao, Traffic sign recognition using deep convolutional networks and extreme learning machine, Intell. Sci. Big Data Eng. 9242 (2015) 272–280.
[41] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 839 Greene Street Ann Arbor, Michigan, 1975.
[42] Q. Xu, C. Zhang, L. Zhang, Denoising convolutional neural network, in: Proceedings of the IEEE International Conference on Information and Automation, 2015, pp. 1184–1187.
[43] B. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1981, pp. 674–679.
[44] Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-learning-detection, IEEE Trans. Pattern Anal. Mach. Intell. 34 (7) (2012) 1409–1422.
[45] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[46] Z. Kalal, J. Matas, K. Mikolajczyk, P-N learning: bootstrapping binary classifiers by structural constraints, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2010, pp. 49–56.

[47] B. Lucas, T. Kanade, Moving obstacle detection in highly dynamic scenes, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2009, pp. 56–63.
[48] L. Castangia, P. Grisleri, P. Medici, A. Prioletti, A. Signifredi, A coarse-to-fine vehicle detector running in real-time, in: Proceedings of the IEEE Intelligent Transportation Systems Conference, 2014, pp. 691–696.
[49] W.R. Schwartz, L.S. Davis, H. Pedrini, Local response context applied to pedestrian detection, in: Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, 2011, pp. 181–188.
[50] J. Šochman, J. Matas, Waldboost - learning for time constrained sequential detection, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2005, pp. 150–157.



Keyu Lu received the B.E. degree and the M.E. degree in control science and engineering from National University of Defense Technology (NUDT), Changsha, Hunan, P.R. China, where he is currently working toward the Ph.D. degree. He is also a visiting Ph.D. student at the University of British Columbia (UBC). His research interests include computer vision, robotics, pattern recognition and machine learning.

Xiangjing An received the B.S. degree in automatic control from the Department of Automatic Control, National University of Defense Technology (NUDT), Changsha, P. R. China, in 1995 and the Ph.D. degree in control science and engineering from the College of Mechatronics and Automation (CMA), NUDT in 2001. He has been a visiting scholar for cooperation research in the Boston University during 2009–2010. Currently, he is a Professor at the Institute of Automation, CMA, NUDT. His research interests include computer vision, mobile robot, image processing and machine learning.

Jian Li received the B.E., M.E. and Ph.D. degrees in control science and engineering from National University of Defense Technology (NUDT), Changsha, Hunan, PR China. He was a visiting Ph.D. student at Center for Intelligent Machines (CIM) in McGill University in 2012. He is currently a lecturer in the College of Mechatronic Engineering and Automation at NUDT. His research interests include computer vision, robotics, image processing, and machine learning.

Hangen He received the BSc degree in Nuclear Physics from Harbin Engineering Institute, Harbin, China, in 1968. He was a visiting Professor at the University of the German Federal Armed Forces in 1996 and 1999 respectively. He is currently a professor in the College of Mechatronics and Automation (CMA), National University of Defense Technology (NUDT), Changsha, Hunan, China. His research interests include artificial intelligence, computer vision, robotics and learning control. He has served as a member of editorial boards of several journals and has cochaired many professional conferences. He is a joint recipient of more than a dozen academic awards in China.