Received May 1, 2019, accepted May 18, 2019, date of publication May 22, 2019, date of current version June 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2918267

Place and Object Recognition by CNN-Based COSFIRE Filters

MANUEL LÓPEZ-ANTEQUERA[1,2], **MARÍA LEYVA VALLINA**[1], **NICOLA STRISCIUGLIO**[1], **AND NICOLAI PETKOV**[1]

[1]Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, 9712 AK Groningen, The Netherlands
[2]MAPIR Group, Instituto de Investigación Biomédica de Málaga (IBIMA), University of Málaga, 29016 Málaga, Spain

Corresponding author: Manuel López-Antequera (m.a.lopez.antequera@rug.nl)

This work was supported in part by the European Horizon 2020 Program, under the Project TrimBot2020 Grant 688007.

ABSTRACT COSFIRE filters are an effective means for detecting and localizing visual patterns. In contrast to a convolutional neural network (CNN), such a filter can be configured by presenting a single training example and it can be applied on images of any size. The main limitation of COSFIRE filters so far was the use of only Gabor and DoGs contributing filters for the configuration of a COSFIRE filter. In this paper, we propose to use a much broader class of contributing filters, namely filters defined by intermediate CNN representations. We apply our proposed method on the MNIST data set, on the butterfly data set, and on a garden data set for place recognition, obtaining accuracies of 99.49%, 96.57%, and 89.84%, respectively. Our method outperforms a CNN-baseline method in which the full CNN representation at a certain layer is used as input to an SVM classifier. It also outperforms traditional non-CNN methods for the studied applications. In the case of place recognition, our method outperforms NetVLAD when only one reference image is used per scene and the two methods perform similarly when many reference images are used.

INDEX TERMS COSFIRE filter, CNN, object recognition, place recognition.

I. INTRODUCTION AND RELATED WORK

The COSFIRE (Combination of Shifted Filter Responses) method as proposed in [3] is a brain-inspired computer vision technique that uses the relative arrangement of local patterns in an image. It has been applied to various problems, such as the localization of bifurcations in retinal fundus images, the localization and recognition of traffic signs and the recognition of handwritten digits [3], as well as the delineation of blood vessels in medical images [5], [21] or curvilinear structures in natural images [22]. The COSFIRE method has been designed taking inspiration from the function of a certain class of neurons in area V4 of the visual cortex. Such a neuron would respond to a curved segment or a vertex of some preferred orientation and opening [16], [17]. These neurons most likely receive their input from orientation selective neurons in areas V1 and V2 in visual cortex: combining the responses of groups of such neurons that are selective for the two orientations of the two legs of a vertex, a V4 neuron would be selective for the vertex. One way to implement this idea in a computational model and an image processing operator is to take orientation selective filters – Gabor filters were used in [3] – and combine their responses. As these filters respond in different positions of the pattern of interest, their responses need to be shifted in order to bring them to the same position where they can be combined by a point-wise image processing operation. This is the origin of the name of this method: Combination of Shifted Filter Responses, abbreviated as COSFIRE (Figure 1).

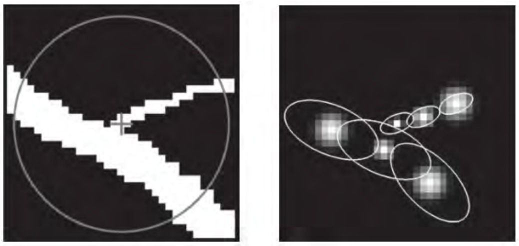

FIGURE 1. (left) A vessel bifurcation and (right) a schematic representation of a COSFIRE filter configured to detect it. The ellipses illustrate contributing Gabor filters of different orientations and support sizes and the locations from which the responses of these Gabor filters are taken. These responses are combined to produce the output of the COSFIRE filer.

The associate editor coordinating the review of this manuscript and approving it for publication was Varuna De Silva.

Although it was inspired by a specific type of curvature and vertex selective neuron in cortical area V4, the principle of a COSFIRE filter is not limited to the use of orientation selective contributing filters, such as Gabor filters. More generally, this principle involves the use of some filters that we will call in the following 'contributing' filters and the relative positions of their responses in the image plane. The output of the composite COSFIRE filter is computed as a function of the shifted responses of the contributing filters. Gabor filters are appropriate to use when the pattern for which a COSFIRE filter is configured is mainly defined by contours. This is for instance the case for a blood vessel bifurcation or a part of a handwritten digit. In other applications, different contributing filters may be more appropriate. This is for instance the case where the pattern of interest is defined by the local spatial distribution of different colors. The latter case was treated in [6] where Difference of Gaussians (DoG) color-blob detectors were deployed as contributing filters.

The main advantage of COSFIRE filters is that such a filter can be configured automatically by presenting a *single* training example. This aspect can be a major advantage over alternative approaches, such as (deep) convolutional neural network (CNN), in applications in which a relatively small number of training examples are available. Another advantage over Convolutional Neural Networks (CNNs) is that a COSFIRE filter can be applied to an input image of any size while a CNN for classification expects an image of a fixed size and needs to be applied on sliding windows for larger images.

The main limitation of the COSFIRE method so far was the use of only Gabor and (color) DoG filters as contributing filters. The use of these filters was inspired by their biological counterparts in the visual system of the brain, namely neurons in LGN and cortical areas V1 and V2 for which a lot is known from neuroscience research. The properties of neurons in deeper areas of visual cortex, such as TEO, are less known and no mathematical models are available. A multi-layer COSFIRE model of the ventral stream was presented and deployed for object recognition in [4]. However, that model is based on general knowledge about the architecture of the ventral system (V1 - V2 - V4 - posterior TEO - anterior TEO) rather than on detailed knowledge of the functions of neurons in the deeper layers.

In this paper we propose to use a much broader class of contributing filters, namely filters defined by intermediate CNN representations. This idea originates in the apparent similarity of the (2D-Gabor-filter-like) properties of some computational units in the first convolutional layer of a deep CNN with the properties of neurons in areas V1 and V2 of visual cortex [11]. While there are no mathematical models of visual neurons in the deeper layers of the ventral stream of the brain, such models of the units in the deeper layers of a CNN are available.

In the current paper we propose to use filters defined by intermediate CNN representations as contributing filters in the COSFIRE method. We consider a set of points in an image and the feature vectors associated with these points in a certain layer of a pre-trained CNN when a pattern of interest is presented. We use each such feature vector to define a filter: the filter output for a new presented image is computed as the inner product of the concerned feature vector with the 3D representation of the new image at the same layer of the pre-trained CNN.

To validate the proposed method we apply it to various problems: recognition of MNIST handwritten digits, localization and classification of butterflies, and place recognition in a garden.

Our contributions are as follows:

- We use intermediate CNN representations to define contributing filters in the COSFIRE approach.
- We show that the proposed approach allows to use the representations computed by pre-trained CNNs in different applications, without fine-tuning the network.
- We demonstrate the effectiveness of the proposed approach in applications with a few training samples.

The paper is organized as follows: in Section II we present the new method, in Section III we present the data sets and results, and in Section V we draw conclusions.

II. METHOD

A. OVERVIEW

A COSFIRE filter is constructed by combining the responses of so-called *contributing filters*. The contributing filters respond to given local patterns, while the COSFIRE filter responds to a larger pattern that is composed of the mentioned local patterns in a given geometric configuration. In a CNN-based COSFIRE filter, we define contributing filters using feature vectors extracted from an intermediate convolutional layer of a pre-trained CNN.

More specifically, we use a CNN to obtain a 3D intermediate representation of an input image at a given layer l of a pre-trained CNN. We input an image of size $w_0 \times h_0 \times d_0$ and obtain the $w_l \times h_l \times d_l$ representation in layer l. A pre-selected point of interest in the input image maps to a point in the $w_l \times h_l$ space of the concerned intermediate representation. We take the feature vector of length d_l associated with that point. We use this feature vector to define a filter that produces a $w_l \times h_l$ output image as follows. For a new input image, we extract its $w_l \times h_l \times d_l$ representation in the concerned layer l of the CNN. Then we compute the dot product of this representation with the above mentioned feature vector, resulting in a $w_l \times h_l$ filter response. We refer to such a filter as a contributing filter. We define a set of contributing filters using different points in a region of interest or in a whole image. The concerned points can be selected manually, according to their perceptual importance, or randomly and illustrations for both are given in the following. We use these contributing filters to compute the output of a COSFIRE filter. Since the contributing filters produce maximum responses in different points of the image,

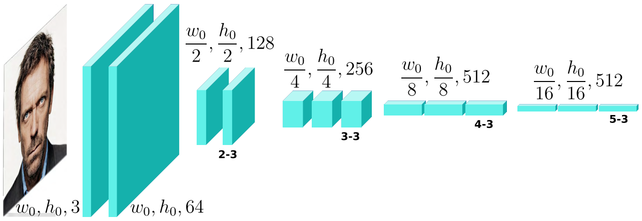
$\frac{w_0}{2}, \frac{h_0}{2}, 128$
$\frac{w_0}{4}, \frac{h_0}{4}, 256$
$\frac{w_0}{8}, \frac{h_0}{8}, 512$
$\frac{w_0}{16}, \frac{h_0}{16}, 512$
2-3
3-3
4-3
5-3
$w_0, h_0, 3$
$w_0, h_0, 64$

FIGURE 2. Intermediate representations of the VGG16 architecture. Each box represents the 3D array output of a convolutional layer and its subsequent ReLU layer. The bold text indicates the naming convention that we use for the layers.

we first shift them to one common point in which we want to have a maximum response of the composite COSFIRE filter. This common point is usually in the center of the region of interest. The shift vector is different for each contributing filter: it starts in the point used to define a given contributing filter and ends in the above mentioned common point. The shift vector is applied to the whole output of the concerned contributing filter. We combine the shifted responses of the different contributing filters in each image point using a multi-variate function, typically the geometric mean. As a result we obtain a scalar response map of the same size as the input image in which there is strong response in the above mentioned common point in the center of the pattern of interest used to configure the COSFIRE filter.

B. CONVOLUTIONAL NEURAL NETWORKS (CNNs)

We briefly review some aspects of CNNs that are relevant for the proposed method.

A typical CNN (Figure 2) produces a sequence of intermediate representations that are 3D arrays, each computed from the preceding one using a set of 2D convolutions followed by a non-linear activation function, such as half-wave rectification, and pooling (down-sizing). Typically, the first 3D array in this sequence is an RGB image with two indices corresponding to the 2D image coordinates and the third index corresponding to the color channel. In subsequent layers two of the indices retain their meaning of spatial coordinates but their extents are typically reduced by the factors used in pooling operations.[1] The third array index enumerates features that are computed (using different convolution kernels) and its extent usually increases in subsequent layers of the CNN.

More formally, given an input image $X \in R^{w_0 \times h_0 \times d_0}$, where w_0 and h_0 indicate its size and d_0 the number of channels ($d_0 = 3$ for RGB images), the output of the l-th layer of a CNN is the result of a transformation:

$$F_l = \Phi_l(X), \quad \Phi_l : \mathbb{R}^{w_0 \times h_0 \times d_0} \longmapsto \mathbb{R}^{w_l \times h_l \times d_l} \tag{1}$$

C. CNN-BASED CONTRIBUTING FILTERS

To illustrate our method let us consider the image X of a face shown in the top-left corner of Figure 3. In this image we have selected several points that are marked with red 'x' marks. For this illustration we select the concerned points manually. In the applications given in the results section we do not use manual selection. We use each such point for the definition of a so-called contributing filter.

First, we compute the intermediate 3D representation $\Phi_l(X)$ of X at layer l of a CNN. For this illustration we used the representation in layer relu3-3 of VGG16-net. A selected point with image coordinates (i, j) in the input image is mapped to a point with spatial coordinates (i_l, j_l) in layer l, according to the series of involved down-sampling (pooling) operations. $\Phi_l(X)$ is a $w_l \times h_l \times d_l$ 3D array. We extract from it a 1D segment, a vector $\Phi_l(X)_{i_l, j_l, :}$ that is associated with the spatial coordinates (i_l, j_l). We use this feature vector to define a contributing filter as follows.

[1] Other operations may reduce the resolution of the feature representation, such as convolutions with stride greater than one.

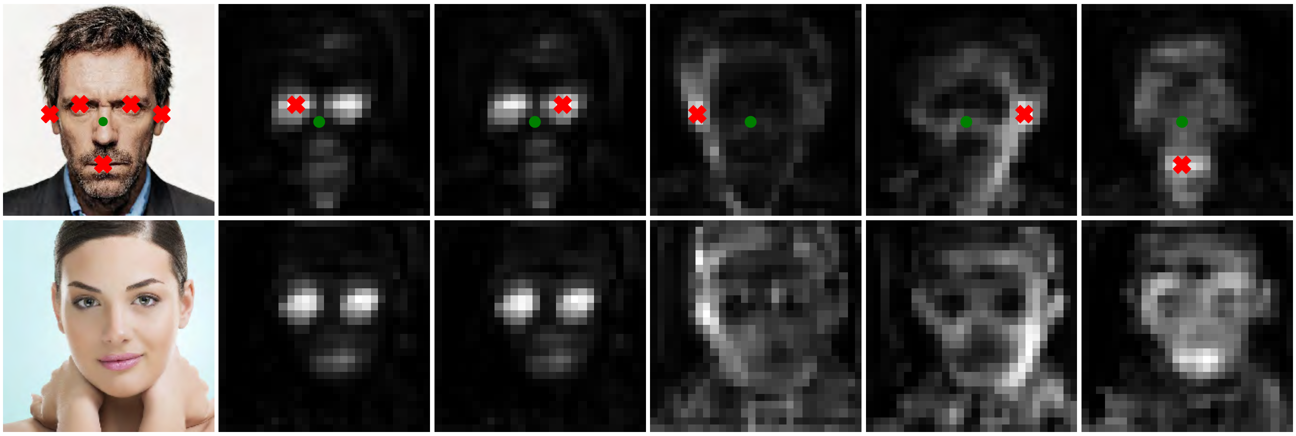

FIGURE 3. The top-left image is the image *X* we used for the configuration of CNN-based contributing filters. The red crosses in that image mark the positions manually selected to define contributing filters. The images in columns 2 through 6 show the filter responses that are re-sized (up-scaled) to the size of the input image. The response images in the first row contain also the marker of the position that was used to configure the corresponding contributing filter. The first row shows the responses to an image *X* that was used to define the contributing filters. The second row shows the responses to another image *Y*.

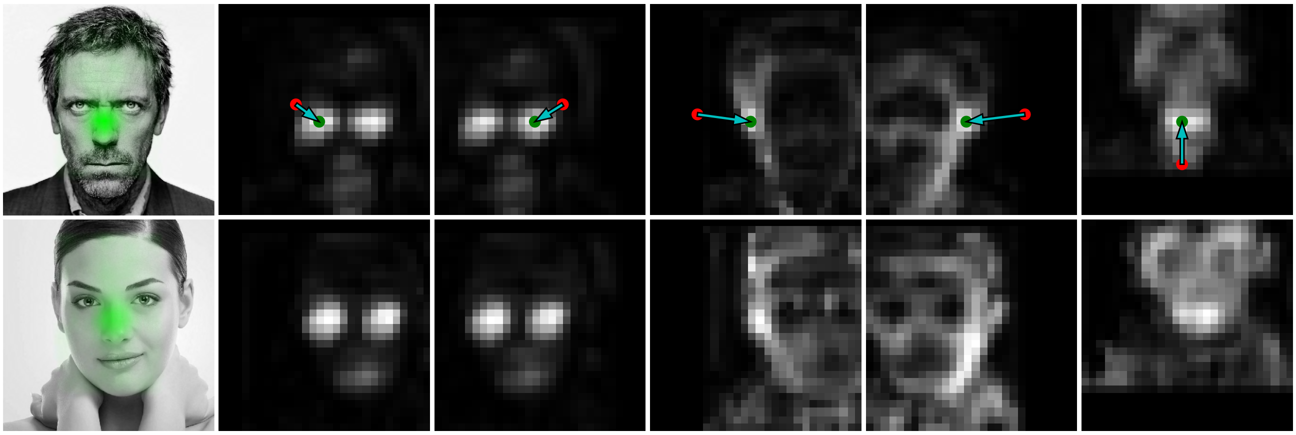

FIGURE 4. Columns 2 to 6 show the shifted responses of the contributing filters. As in Figure3, the first row corresponds to the image that we used to configure the COSFIRE filter. Note how the maximum response of each contributing filter moved from the red dot that marks the position used to define that filter to the green dot chosen as a center of the composite COSFIRE filter. The shift vectors are indicated by arrows. In the second row, we show the shifted response maps obtained on the test image in the second row of Figure 2. We combine the shifted responses of the contributing filters to yield the response of the COSFIRE filter that is rendered in green color superimposed on the input images as shown in the first column. The COSFIRE filter responds to the face pattern in the first face that we used for the configuration but it also responds to a different face in the second row.

The response map of a contributing filter to an input image Y is a $w_l \times h_l$ 2D array. To compute it, we first compute the representation $\Phi_l(Y)_{:,:,:}$ of the image Y at the concerned layer l which is a $w_l \times h_l \times d_l$ 3D array. Then we compute the inner product of $\Phi_l(Y)_{:,:,:}$ with the feature vector $\Phi_l(X)_{i_l,j_l,:}$ associated with point (i_l, j_l) of image X. We denote the resulting $w_l \times h_l$ 2D array by $C_l^{(X,i,j)}(Y)_{:,:}$ where we use the superscript (X, i, j) to indicate that C is the result of an application of a contributing filter configured in point (i, j) of an image X when this filter is applied to an image Y:

$$C_l^{(X,i,j)}(Y)_{k_l,m_l} = \Phi_l(Y)_{k_l,m_l,:} \cdot \Phi_l(X)_{i_l,j_l,:} \qquad k_l = 1\dots w_l, m_l = 1\dots h_l \tag{2}$$

Figure 3 shows two input images X and Y (first column) and the corresponding outputs of five different contributing filters (columns 2-6). The responses illustrate how each such filter is selective for the local image pattern around the point used for its definition. In the second row of Figure 3, a different image Y is used as input to the contributing filters defined on the image X.

D. COMBINING THE CONTRIBUTING FILTER RESPONSES

The contributing filters respond to local image patterns that are similar to the regions used to configure them. We use the responses of the contributing filters to compute the response of a COSFIRE filter. This COSFIRE filter aggregates the responses of the contributing filters in a specific geometric arrangement corresponding to the mutual arrangement of the regions used to configure the contributing filters. Since the contributing filters give maximum responses in different image positions, marked by red crosses in Fig.3, we first bring these maximum responses to one point in which we want the COSFIRE filter to give a maximum response. In Figure 3 this latter point is marked by a green spot. Bringing the maximum response of a contributing filter to that point is done by shifting the whole response map produced by that contributing filter by a shift vector defined by the corresponding (red) point used for contributing filter configuration and the (green) point in which we want the COSFIRE filter to give maximum response. The shift vector is thus specific for each contributing filter. After we shift the response maps of the contributing filters, we combine them using a point-wise multi-variate function, namely the geometric mean. The result is a response map of the COSFIRE filter that has a maximum in the above mentioned green point.

More formally, let us denote the coordinates of the 'green' point in the input image with $(\hat{i}, \hat{j})$ and the coordinates of the 'red' points with (i_c, j_c), $c = 1, \dots, n_c$, where n_c is the number of contributing filters. In the outputs of the contributing filters these points map to $(\hat{i_l}, \hat{j_l})$ and (i_{l_c}, j_{l_c}), $c = 1, \dots, n_c$, respectively. We define the shift vector of the c-th contributing filter as $(\Delta i_{l_c}, \Delta j_{l_c}) = (\hat{i_l} - i_{l_c}, \hat{j_l} - j_{l_c})$.

The results of the shift operations on the response maps of the contributing filters are displayed in Figure 4.

Finally, we compute the response of the COSFIRE filter as the pixel-wise geometric mean of the shifted responses of the contributing filters. The results for the two images X and Y are shown in the first column of Figure 4.

Formally, we compute the response of this COSFIRE filter to an image Y as follows:

$$R(Y)_{k_l,m_l} = \left[\prod_{c=1}^{n_c} C_l^{(X,i_c,j_c)}(Y)_{k_l-\Delta i_{l_c}, m_l-\Delta j_{l_c}}\right]^{\frac{1}{n_c}} \qquad k_l = 1\dots w_l; m_l = 1\dots h_l. \tag{3}$$

In the first column of Figure 4, we illustrate the response of this COSFIRE filter by re-sizing the output to the size of the input image and rendering it in green color superimposed on two images on which it is applied. It is evident that the filter can respond to a pattern of a face other than the one used for configuration.

E. CLASSIFICATION USING CNN-COSFIRE FILTERS

We deploy the proposed CNN-COSFIRE filters as feature extractors to form feature vectors, which we use in combination with a classifier. In a configuration phase, we use training images to configure N CNN-COSFIRE filters. For each such filter, we randomly select the location of its center. In a region around this center we then select randomly the centers of n_c contributing filters. Subsequently, we apply the configured N filters to a training image I, obtaining N two-dimensional response maps. We construct a feature vector $v(I)$ to represent the image I as:

$$v(I) = \left[\hat{R}_1(I), \hat{R}_2(I), \ldots, \hat{R}_N(I)\right] \tag{4}$$

where the i-th element

$$\hat{R}_i(I) = \max_{k_l, m_l}\{R_i(I)_{k_l, m_l}\} \tag{5}$$

is the global maximum of the i-th CNN-COSFIRE response map $R_i(I)_{:,:}$ computed on the image I according to Eq. 3. We compute such a feature vector for each image in the training set.

We use the CNN-COSFIRE feature vectors and the labels associated to the training images to train a SVM classifier. This choice of a classifier is independent of the proposed CNN-COSFIRE filters.

To classify a new image J, we compute the feature vector $v(J)$ using the CNN-COSFIRE filters configured in the training phase and use such vector as input to the classifier to predict the class label. In the next Section III, we provide results of classification experiments that employ CNN-COSFIRE feature vectors.

III. RESULTS

We carried out experiments on two data sets for classification, namely MNIST [12] and Butterflies [10], as well as on a novel TB-places8 data set for place recognition in garden scenes.

We consider the intermediate 3D representations in different layers of a VGGnet CNN pre-trained for the task of image classification on ImageNet [20] to define contributing filters. In the case of the experiments on garden place recognition, we used the response maps of the VGGnet trained for place recognition in the framework of NetVLAD. We chose the VGGnet for its simple implementation and popularity, but any CNN could be used. In all cases, the networks are used exclusively as feature extractors and are not fine-tuned for the tasks.

A. MNIST

The MNIST data set [12] is composed of 70k grayscale images of digits, of size 28 × 28 pixels, divided into 60k images for training and 10k images for testing. The images are organized in 10 classes. The MNIST data set has been widely used for benchmarking of object classification algorithms.

VGGnet is a large architecture trained for much more challenging data, however, we decide to use the same network for all of our experiments to show the effectiveness of our approach.

When using MNIST images (of size 28 × 28) as input to a VGG network, the resolution of intermediate response maps at deeper layers of the network collapses due to the max-pooling operations of stride 2 that halve the resolution. In order to produce feature maps with spatial resolution as required for the configuration and application of the proposed COSFIRE filters, we carried out experiments with a modified VGGnet architecture, with the aim of maintaining the resolution of the original images at all network layers.

Our modifications to the VGGnet architecture consist of using max-pooling layers with stride 1 instead of 2 and substituting the convolutions with dilated convolutions (with the same weights as the original convolutions). The max-pooling operation with stride equal to 1 avoids the exponential decrease of the resolution in deeper layers, while the use of dilated convolutions maintains the same spatial resolution of the filters in the original VGGnet. These modifications allow to compute feature maps at any layer that have the same resolution as the input image. It is worth pointing out that these modifications were only necessary to demonstrate the use of the proposed COSFIRE module on the limited resolution MNIST images. They are not needed for images of higher resolution.

We feed a training image into the modified VGGnet and extract its representation at layer relu4-3.[2] Then, we randomly select five points ($n_c = 5$) in the image and use them to define five contributing filters that we subsequently make part of a COSFIRE filter. We repeat this process for other training images and configure in total N COSFIRE filters. We performed experiments with different values of N : from 100 (10 per class) to 4000 (400 per class).

We apply the set of N COSFIRE filters to an image from the training or the test set and for each filter we keep only the maximum value in its response across all image points. In this way we obtain a feature vector of N elements, one for each COSFIRE filter, that we use as a representation of the concerned image. We use the feature vectors obtained from the images of the training set to train a 10-class linear SVM. Then, we deploy this SVM to classify the feature vectors obtained from the test images.

As is common in the literature, we use the accuracy to evaluate the performance of our method and to compare with others:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}},$$

[2] Through experimentation we discovered that layer relu4-3 provides the best features to be used as input to our method for this data set.

where the terms positive (P) and negative (N) refer to the classifier's prediction, and the terms true (T) and false (F) refer to whether that prediction is correct.

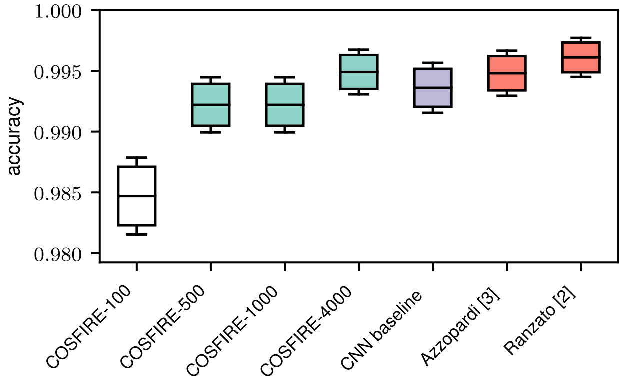

FIGURE 5. Accuracy on MNIST. Box height and whiskers span represent the 95% and 99% confidence intervals, respectively.

Figure 5 shows the accuracy of the results obtained with different values of the number N of COSFIRE filters. Using a relatively small number of COSFIRE filters N $= 100$ yields already a reasonable accuracy of 98.47%. The accuracy improves with an increasing number of filters: 99.22% with N $= 500$ and N $= 1000$. We obtain the highest accuracy of **99.49**% with the maximum number of filters with which we experimented N $= 4000$.

We also use the full representation in layer relu3-3[3] of the original VGGNet (i.e. with max-pooling of stride 2) that has size 12544 and shape $7 \times 7 \times 256$. We use this representation as an input to a 10-class linear SVM. We refer to this method as the 'CNN-baseline' classifier. (The idea of applying a SVM to intermediate CNN representations is due to [18]. The accuracy that we obtain with this CNN-baseline classifier is 99.36%.

The result 99.49% that we obtain with our approach is better than the CNN-baseline (99.36%) and the result 99.48% obtained by [3] with the same number (4000) of Gabor-based COSFIRE filters. The best result reported in the literature [2] is 99.61% but it has been obtained with an extended training set that includes elastically distorted training images. Furthermore, the differences between these results are not statistically significant.[4]

B. BUTTERFLY DATA SET

The butterfly data set [10] contains 619 RGB color images of butterflies divided into 7 classes. The sizes of the images are different, e.g. 737×553, 659×521, 390×339, etc. We use the training and test split included in the data set: 182 training images (26 per class) and 437 test images.

From each training set image we crop a region of interest (ROI) that is a bounding box containing a butterfly, (Figure 6). We feed the cropped region[5] into the VGGnet and extract its representation at layer relu5-3. We select randomly five points from the ROI and we use them to define five contributing filters ($n_c = 5$) that we subsequently make part of a COSFIRE filter. By selecting different sets of five points each in the same ROI we configure further COSFIRE filters, as many as we need. We repeat this process for ROIs obtained from other training set images and we configure in total N COSFIRE filters. We performed experiments with different values of N , from 7 (one per class) to 2800 (400 per class).

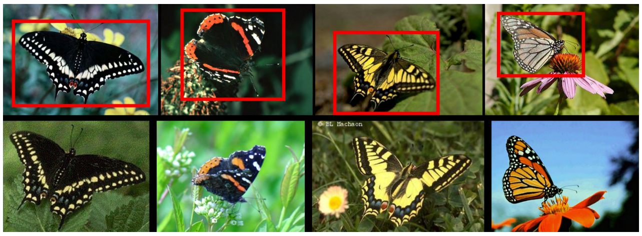

FIGURE 6. Example images from the butterfly data set. Top: Training images from four different classes; the bounding boxes define the regions of interest used to configure COSFIRE filters. Bottom: Test images from the same classes.

We apply the set of N COSFIRE filters to an image from the training or the test set and for each filter we keep only the maximum value of its response across all image points. In this way we obtain a feature vector of N elements, one for each COSFIRE filter, that we use as a representation of the concerned image. We use the feature vectors obtained from the images of the training set to train a 7-class linear SVM. Then we deploy this SVM to classify the feature vectors obtained from the test images.

Figure 7 shows the accuracy results obtained with different values of the number N of COSFIRE filters and with other methods. The accuracy improves with an increasing number of filters. We obtain the highest accuracy of 96.57% with N $=$ 700 but we obtain a comparable result with 2800 (94.97%) COSFIRE filters.[6]

We also use the full representation in layer relu5-3 as a baseline. For this purpose we first re-size the images to 224×224 resolution before feeding them to the VGGnet. This yields a 100352-dimensional representation (of shape $512 \times 14 \times 14$) that the VGGnet produces at layer relu5-3. We use this representation as an input to a 7-class linear SVM. We refer to this method as the 'CNN-baseline' classifier.

[3] We obtained the best results with layer relu3-3.

[4] The t-test comparison of our result 99.49% and the result 99.61% reported in [2] gives $t = 1.268$ and $p = 0.1024$, so that one cannot conclude that the latter method is better than the former with sufficient statistical significance.

[5] There are no restrictions on the size of the cropped region because we use an intermediate tensor representation that is obtained by applying a series of convolution, ReLU and pooling operations; we do not use any connected layers.

[6] The difference is not statistically significant to conclude that one method outperforms the other, as a one-tailed t-test yields relatively high p value of 0.12.

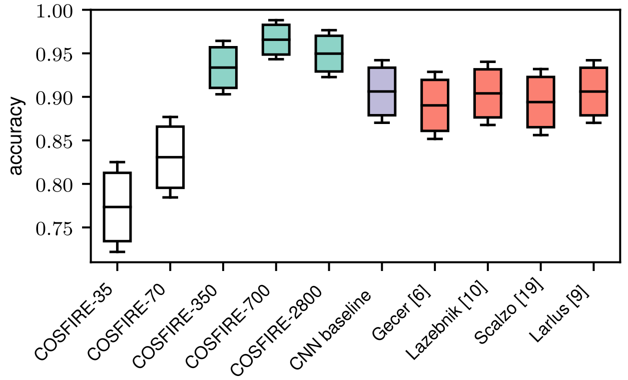

FIGURE 7. Accuracy on the butterfly data set with different methods, from left to right in %: 77.35, 83.07, 93.36, 96.57, 94.97, 90.62, 89.02, 90.40, 89.40, 90.61. The bars and whiskers represent the 95% and 99% confidence intervals, respectively, for the value of the accuracy that is obtained with a finite set of 437 test images.

The accuracy that we obtain with this CNN-baseline classifier is 90.62%.

The accuracy result of 96.57% that we obtain with our approach based on COSFIRE-700 filters is significantly better[7] than the results obtained with the CNN-baseline classifier (90.62%) and other previously deployed methods: 89.02% [6], 90.40% [10], 89.40% [19], 90.61% [9].

C. GARDEN PLACE RECOGNITION DATA SET

In visual place recognition one has to recognize a previously visited place based on visual cues only [14], [15]. Generally, a query image is compared with reference images of known places and a decision about the most similar place is taken. Algorithms that can effectively recognizing places can facilitate camera localization and visual navigation. Challenges are given by varying illumination conditions and changes of the viewpoint from where an image is taken. We constructed a new data set of 424 images (of size 224 × 350 pixels), which we called TB-places8. We recorded the data set in the experimental garden of the TrimBot2020 project,[8] whose aim is to develop the first outdoor gardening robot [23]. One of the tasks of the robot is to navigate the garden and localize itself by using camera sensors only. We recorded the image data by using a camera system on board of the robotic platform, which is composed of a rig of ten synchronized cameras [7]. For each image, we registered ground truth camera pose data by using a laser tracker and an intertial measurement unit (IMU). We provide more details about the recording hardware settings and the ground truth labeling in [13].

The 424 images of the TB-places8 data set are organized in eight classes, each of them containing images of a specific scene of the garden. We constructed the ground truth (i.e. a scene label for each image) using the camera poses associated with the image. In Table 1, we provide details about the composition of the data set. We show example reference and query images from the constructed data set in Figure 8.

TABLE 1. Number of images per scene class of the TB-places8 data set.

Place	0	1	2	3	4	5	6	7	*Total*
Train	5	5	5	5	5	5	5	5	*40*
Test	98	44	55	38	44	14	20	71	*384*
Total	103	49	60	43	49	19	25	76	*424*

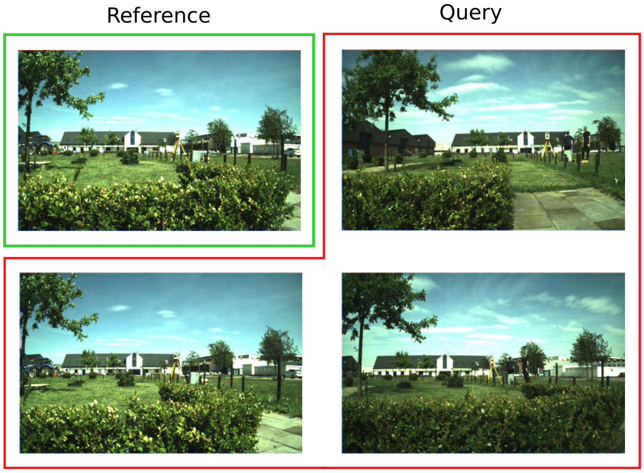

FIGURE 8. Examples of reference and query images from the TB-places8 data set. The top-left image is a reference image; the other images of the same scene are taken from different viewpoints and are used as queries.

For the configuration of the CNN-COSFIRE filters, we use as contributing filters the VGGnet part of the NetVLAD architecture [1] trained on the Pittsburgh30k data set. We use this specific version of VGGnet as it is trained for place recognition applications.

We use N_r (=1, 3 or 5) reference images from each class to configure CNN-COSFIRE filters and train a classifier, and keep 384 images for testing. Then, we randomly select N filter centers, uniformly distributed among classes, from the training images. For each of the filter centers, we select n_c (=3, 5, 7 or 9) contributing filters in a square ROI around the center of square with a side 100, 150 or 200 pixels. n_c and the ROI size are chosen randomly for each filter.

We apply the set of N filters to an image and we keep the maximum response of each filter to build a feature vector of the considered image. Then we use the feature vectors obtained from the set of training images to train a 8-class linear SVM classifier. Finally, we use this classifier to assign each test/query image to a class.

We considered the following values of N : 800, 1600, 2400, 3200 and 4000, for which we configured 100, 200, 300, 400 and 500 filters per class, respectively. We configured the filters on $N_r = 1, 3, 5$ reference/training images per class. Figure 9 shows the obtained accuracy for the different numbers of COSFIRE filters N deployed and the different number of reference images N_r per class used. We obtain better place recognition results when we use more than one reference

[7] A one-tailed t-test shows that COSFIRE-700 outperforms the best of these methods [9] with high statistical significance, corresponding to a very small p value of 0.00015.

[8] http://www.trimbot2020.org

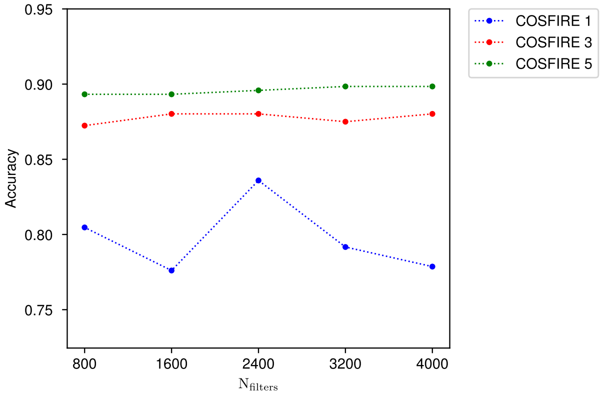

FIGURE 9. Accuracy on place recognition obtained with different numbers of COSFIRE filters N and different numbers N_r of reference images per class that were used for filter configurations. The latter number is shown in the legends of the methods. For instance, COSFIRE 5 means that $N_r = 5$ reference images per class were used to train COSFIRE filters. The feature vectors were presented to an SVM classfier.

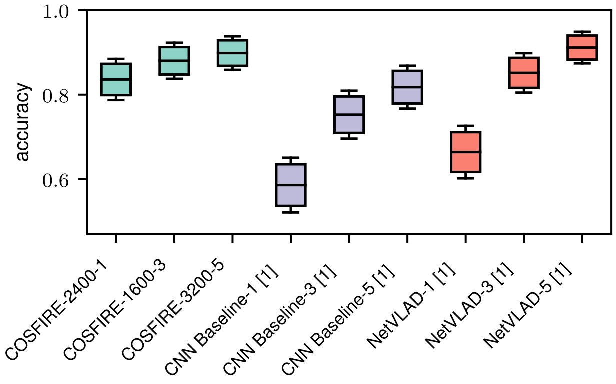

FIGURE 10. Accuracy on the TB-places8 data set with different methods, from left to right in %: 83.59, 88.02, 89.84, 58.59, 75.26, 81.77, 66.41, 85.15, 91.15. The digits 1, 3 and 5 in the names of the methods specify the number of used reference images per class. The bars and whiskers represent the 95% and 99% confidence intervals, respectively, for the value of the accuracy that is obtained with a finite set of 384 test images.

image per class. We obtain the best results with the largest number of reference images per class that we use $N_r = 5$. In this case, the total number of COSFIRE filters that we deploy N has lesser influence on the accuracy. We obtained the highest accuracy of 89.84% with N $= 3200$ COSFIRE filters using $N_r = 5$ reference images per class.

We compare the results we obtained with our CNN-COSFIRE method to those obtained by NetVLAD and by a CNN-baseline classifier that we construct using the full VGGnet relu4_3 layer representation as input to a 8-class linear SVM. For all three methods the results depend on the number N_r of training images used per class, Figure 10.

For all N_r values, our CNN-COSFIRE method outperforms the CNN-baseline classifier with a statistically significant difference.[9] Our CNN-COSFIRE method outperforms NetVLAD with a statistically significant difference when only one reference image per class is used, $N_r = 1$. For $N_r = 3$ COSFIRE still outperforms NetVLAD, but the difference is not statistically significant. For $N_r = 5$, NetVLAD is slightly better but the difference to CNN-COSFIRE is not statistically significant.[10]

IV. DISCUSSION

A. LOCAL RESPONSE OF CNN FEATURES

CNN features are robust to image characteristics that can be considered as noise for the purpose of classification: sufficiently deep CNN layers represent abstract semantic concepts (such as 'eye' or 'mouth') irrespective of changes in appearance seen in the training set. For this reason, CNN features are well-suited for the description of semantic keypoints in images. To illustrate this, we have performed a qualitative comparison with cross-correlation that can be seen in Figure 11. Notice how the response of the CNN-based contributing filters is strongly concentrated on the locations corresponding to the same semantic concepts, while the response of cross correlation is diffused over the whole image. Moreover, the CNN-based descriptors generalize better when applied to a new image, as indicated by the third and fourth rows of Figure 11.

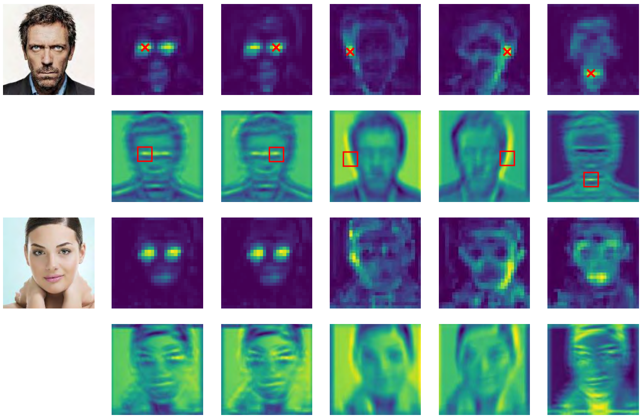

FIGURE 11. Qualitative comparison of the responses of CNN features with pixel-wise cross-correlation: The top left corner shows a reference image. The image is used as input to a CNN, obtaining a feature map. Each column in the top row represents the *response map* obtained by computing dot product of the feature vector at some particular location (marked by a red 'x' symbol) with the rest of the feature vectors extracted from the image. On the second row, a similar operation is performed by extracting pixel patches (indicated by red squares) and performing cross-correlation with the rest of the image. We evaluate the response of both techniques on a new image on the third and fourth rows: The extracted CNN feature vectors from the reference image are compared with the feature map extracted the test image on the third row. Finally, the pixel patches extracted from the reference image are cross-correlated with the test image, resulting in the response map shown in the last row.

B. BEYOND IN-PLANE ARRANGEMENT OF DESCRIPTORS

The proposed method deals with the explicit arrangement of features extracted using state of the art network architectures. Although the COSFIRE method deals with the 2D arrangement of features on the image plane, the concept can be

[9] A one-tailed t-test yields p values of $7.93 \cdot 10^{-15}$, $4.46 \cdot 10^{-6}$, 0.0013 for $N_r = 1$, 3 and 5 respectively.

[10] A one-tailed t-test yields p values of $2.97 \cdot 10^{-8}$, 0.24, 0.54 for $N_r = 1$, 3 and 5, respectively.

generalized to account for well-known phenomena related to the geometry of light projection into a camera: deformations seen due to effects such as the three-dimensional motion of the camera or the subject could be encoded using projective geometry. For example, the frame-to-frame pose change of a camera could be used to re-arrange the locations of the contributing filters of a subject that is being tracked using the response of a CNN-COSFIRE filter.

C. END-TO-END LEARNING

The proposed CNN-COSFIRE method opens up the possibility to train the whole pipeline end-to-end. Supervisory signals can be back-propagated through the arrangement of contributing filters to the feature extractor (originally trained to perform image classification) to fine-tune it for its use as input to the filter arrangement. Another possibility is the fine-tuning of the arrangement itself, that is, the relative positions $(\Delta i_{l_c}, \Delta j_{l_c})$ of the contributing filters with respect to the center of the CNN-COSFIRE filter. This is enabled by interpolation that is performed to extract feature values at non-integer coordinates [8].

V. CONCLUSIONS

In this paper we proposed an extension of the COSFIRE method based on the use of intermediate CNN representations. A CNN-COSFIRE filter's response is given by the combination of responses of contributing filters in a specific geometric arrangement on the image plane. A scheme for utilizing CNN features as contributing filters is proposed in this work. These features are highly invariant to common image perturbations such as illumination change or noise, as well as to intra-class appearance variations, due to them being trained to discriminate images for classification tasks.

Our CNN-COSFIRE outperforms a CNN-baseline method in which the concerned full intermediate representation is offered to a SVM classifier. It also outperforms traditional non-CNN methods for the studied applications. In the case of place recognition our method outperforms NetVLAD when only one reference image is used per scene and the two methods perform similarly when many reference images are used.

REFERENCES

[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. CVPR*, Jun. 2016, pp. 5297–5307.

[2] M. A. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient Learning of Sparse Representations with an Energy-Based Model," in *Proc. Adv. Neural Inf. Process. Syst.*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds. Cambridge, MA, USA: MIT Press, 2007, pp. 1137–1144.

[3] G. Azzopardi and N. Azzopardi, "Trainable COSFIRE filters for keypoint detection and pattern recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 490–503, Feb. 2013.

[4] G. Azzopardi and N. Petkov, "Ventral-stream-like shape representation: From pixel intensity values to trainable object-selective cosfire models," *Frontiers Comput. Neurosci.*, vol. 8, p. 80, Jul. 2014.

[5] G. Azzopardi, N. Strisciuglio, M. Vento, and N. Petkov, "Trainable cosfire filters for vessel delineation with application to retinal images," *Med. Image Anal.*, vol. 19, no. 1, pp. 46–57, 2015.

[6] B. Gecer, G. Azzopardi, and N. Petkov, "Color-blob-based COSFIRE filters for object recognition," *Image Vis. Comput.*, vol. 57, pp. 165–174, Jan. 2017.

[7] D. Honegger, T. Sattler, and M. Pollefeys, "Embedded real-time multi-baseline stereo," in *Proc. IEEE ICRA*, May/Jun. 2017, pp. 5245–5250.

[8] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 2017–2025.

[9] D. Larlus and F. Jurie, "Latent mixture vocabularies for object categorization and segmentation," *Image Vis. Comput.*, vol. 27, no. 5, pp. 523–534, 2009.

[10] S. Lazebnik, C. Schmid, and J. Ponce, "Semi-local affine parts for object recognition," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, A. Hoppe, S. Barman, and T. Ellis, Eds. Kingston, U.K., Sep. 2004, pp. 779–788.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[13] M. Leyva-Vallina, N. Strisciuglio, M. L. Antequera, R. Tylecek, M. Blaich, and N. Petkov, "TB-places: A data set for visual place recognition in garden environments," *IEEE Access*, vol. 7, pp. 52277–52287, 2019.

[14] M. Lopez-Antequera, R. Gomez-Ojeda, N. Petkov, and J. Gonzalez-Jimenez, "Appearance-invariant place recognition by discriminatively training a convolutional neural network," *Pattern Recognit. Lett.*, vol. 92, pp. 89–95, Jun. 2017.

[15] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 1–19, Feb. 2016.

[16] A. Pasupathy and C. E. Connor, "Population coding of shape in area V4," *Nature Neurosci.*, vol. 5, no. 12, pp. 1332–1338, Nov. 2002.

[17] A. Pasupathy and C. E. Connor, "Responses to contour features in macaque area V4," *J. Neurophysiol.*, vol. 82, no. 5, pp. 2490–2502, Nov. 1999.

[18] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, May 2014, pp. 512–519.

[19] F. Scalzo and J. H. Piater, "Adaptive patch features for object class recognition with learned hierarchical models," in *Proc. CVPR*, 2007, pp. 1–8.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, Sep. 2014.

[21] N. Strisciuglio, G. Azzopardi, M. Vento, and N. Petkov, "Supervised vessel delineation in retinal fundus images with the automatic selection of *B*-COSFIRE filters," *Mach. Vis. Appl.*, vol. 27, no. 3, pp. 1137–1149, 2016.

[22] N. Strisciuglio and N. Petkov, "Delineation of line patterns in images using B-COSFIRE filters," in *Proc. IWOBI*, 2017, pp. 1–6.

[23] N. Strisciuglio, R. Tylecek, M. Blaich, N. Petkov, P. Biber, J. Hemming, E. van Henten, T. Sattler, M. Pollefeys, T. Gevers, T. Brox, and R. B. Fisher, "TrimBot2020: An outdoor robot for automatic gardening," in *Proc. 50th Int. Symp. Robot.*, Berlin, Germany, 2018, pp. 1–6.

MANUEL LÓPEZ-ANTEQUERA received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Málaga, Spain, in 2009 and 2013, respectively. He is currently pursuing the Ph.D. degree with the Bernoulli Institute for Mathematics, Computer Science, and Artificial Intelligence, University of Groningen, and the Machine Perception and Intelligent Robotics Group, University of Málaga. He is currently a Computer Vision Engineer with Mapillary. His research interests lie at the intersection of machine learning and geometric computer vision for applications such as robot localization.

MARÍA LEYVA VALLINA received the M.Sc. degree in artificial intelligence from the Politèchnic University of Catalonia, in 2017. She is currently pursuing the Ph.D. degree with the Intelligent Systems Group, Bernoulli Institute, University of Groningen. Her main research interests include representation learning and computer vision.

NICOLA STRISCIUGLIO received the Ph.D. degree *(cum laude)* in computer science from the University of Groningen, The Netherlands, in 2016, and the Ph.D. degree in information engineering from the University of Salerno, Italy, in 2017. He is currently a Postdoctoral Researcher with the Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen. He has been the General Co-Chair of the 1st and 2nd International Conference on Applications of Intelligent Systems (APPIS), in 2018 and 2019, respectively. His research interests include pattern recognition, machine learning, signal processing, and computer vision.

NICOLAI PETKOV received the Dr. Sc. Techn. degree in computer engineering (informationstechnik) from the Dresden University of Technology, Dresden, Germany. Since 1991, he has been a Professor of computer science and the Head of the Intelligent Systems Group, University of Groningen. He has authored two monographs, and has authored or co-authored over 150 scientific papers. He holds four patents. His current research interests include pattern recognition, machine learning, data analytics, and brain-inspired computing, with applications in healthcare, finance, surveillance, manufacturing, robotics, and animal breeding. He is a member of the editorial boards of several journals.