Neurocomputing 183 (2016) 90–105

Contents lists available at ScienceDirect

Neurocomputing

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

Visual tracking with VG-RAM Weightless Neural Networks

Mariella Berger [a,*], Alberto F. De Souza [a], Jorcy de Oliveira Neto [a], Edilson de Aguiar [b], Thiago Oliveira-Santos [a]

[a] Departamento de Informática, Universidade Federal do Espírito Santo, Av. Fernando Ferrari 541, 29075-910 Vitória, ES, Brazil
[b] Departamento de Computação e Eletrônica, Universidade Federal do Espírito Santo, Rodovia BR 101 Norte km. 60, 29932-540 São Mateus, ES, Brazil

ARTICLE INFO

Article history:
Received 7 April 2014
Received in revised form
27 February 2015
Accepted 20 April 2015
Available online 1 January 2016

Keywords:
Weightless Neural Networks
VG-RAM
Visual Tracking
Saccade
Superior Colliculus

ABSTRACT

We present a biologically inspired long-term object tracking system based on Virtual Generalizing Random Access Memory (VG-RAM) Weightless Neural Networks (WNN). VG-RAM WNN is an effective machine learning technique that offers simple implementation and fast training. Our system models the biological saccadic eye movement, the transformation suffered by the images captured by the eyes from the retina to the Superior Colliculus (SC), and the response of SC neurons to previously seen patterns. We evaluated the performance of our system using a well-known visual tracking database. Our experimental results show that our approach is capable of reliably and efficiently track an object of interest in a video with accuracy equivalent or superior to related work.

1. Introduction

Humans are capable of visually track a large variety of objects efficiently even in the presence of challenging situations such as abrupt object motion, occlusions, changes in view point, changes in the background, and changes in the appearance of the object of interest, including non-rigid transformations. In spite of recent research advances [1–3], performing the same task with automatic systems is still challenging because specific algorithms have to be created to handle each one of these scenarios [1].

The visual tracking problem can be formulated as follows. Given a bounding box defining the object of interest in the first frame of a video, automatically determine the object's bounding box or indicate that the object is not visible in every frame that follows. The key challenges are that the object and background may change appearance after the initial frame, making it harder to detect the object later on. This problem is emphasized in long-term object tracking, since the chances of having large changes in appearance increase with the time. In addition, the object may be occluded in some parts of the video and reappear later in the sequence. Since objects might reappear in different locations, algorithms tailored to continuous tracking cannot be used and detection of the object is necessary. Object tracking has many practical applications including, but not limited to, robotic vision, human-computer interaction, automatic annotation of video, automated surveillance, traffic monitoring, and vehicle navigation. For reviews of the visual tracking literature, please refer to [1–3].

Algorithms for object tracking follow roughly two main approaches [2]: recursive tracking and tracking-by-detection. Recursive tracking methods estimate the current state of an object of interest by applying a transformation on the previous state based on measurements made on previous and current images. The recursive estimation depends on the state of the object in previous frames and is susceptible to error accumulation [2]. For instance, Lucas and Kanade [4] proposed a method for estimating optic flow within a window around pixels of the object of interest and Comaniciu et al. [5] proposed a real-time tracker based on *mean shift*. Tracking-by-detection methods estimate the object state considering measurements made on the current image only. This avoids error accumulation, but requires training an object detector beforehand. One example is the method proposed by Mustafa et al. [6] that generates synthetic views of an object by applying affine warping techniques to a single template and train an object detector on the warped images. Adaptive tracking-by-detection methods try to take advantage of the two approaches by updating the object detector online. We briefly describe some tracking-by-detection methods in the following.

The method presented by Avidan [7] integrates a support vector machine classifier into an optic-flow-based tracker. The technique proposed by Collins and Liu [8] treats tracking as a binary classification problem having either object of interest and background. Javed et al. [9] employ combination of discriminative and generative models in order to label incoming data and finally

* Corresponding author.
E-mail addresses: mberger@inf.ufes.br (M. Berger), alberto@lcad.inf.ufes.br (A.F. De Souza), jorcy@lcad.inf.ufes.br (J.d.O. Neto), edilson@lcad.inf.ufes.br (E. de Aguiar), thiago@lcad.inf.ufes.br (T. Oliveira-Santos).

http://dx.doi.org/10.1016/j.neucom.2015.04.127
0925-2312/

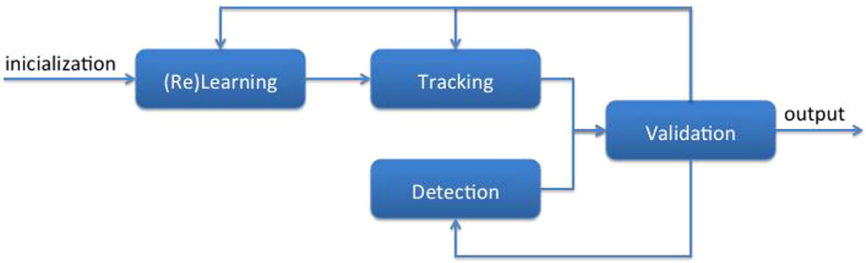


Fig. 1. Block diagram of the visual tracking neural system.

use it to improve an object detector. Ross et al. [10] incrementally learn a low-dimensional subspace representation and adapt it to changes in the appearance of the target. Adam et al. [11] propose *FragTrack,* a method that uses a static part-based appearance model based on integral histograms. Avidan [12] uses self-learning for boosting in order to update an ensemble classifier. Grabner et al. [13] employ a semi-supervised approach and enforces a prior on the first patch. Babenko et al. [14] apply Multiple Instance Learning (MIL) to object tracking. Stalder et al. [15] split the tasks of detection, recognition and tracking into three separate classifiers. Santner et al. [16] propose *PROST,* a cascade of a non-adaptive template model, an optical-flow-based tracker and an online random forest. Hare et al. [17] propose Struck, which generalizes from the binary classification problem to structured output prediction. Kalal et al. [18, 19] uses patches found on the trajectory of an optic-flow-based tracker in order to train an object detector; they named their technique Tracking-Learning-Detection (TLD). Updates are performed only if the discovered patch is similar to the initial patch. In contrast to adaptive tracking-by-detection methods, the output of the object detector is used only to reinitialize the optic-flow-based tracker in case of failure. This enables TLD to achieve superior results and higher frame rates. Recently, a variety of methods have been developed extending and improving the original TLD framework, for instance [20–22]. Although this class of methods has brought a new standard to tracking algorithms, they are still far from the performance achieved by humans during tracking problems. Therefore, there is still space for further improvements.

In this work, we take an alternative strategy for long-term object tracking and use a biologically inspired approach, based on Virtual Generalizing Random Access Memory (VG-RAM) Weightless Neural Networks (WNN) [23,24], for implementing an adaptive tracking-by-detection system. VG-RAM WNN (or VG-RAM for short) is a type of neural network that does not store weights in the synapses. Differently from standard neural networks, knowledge is kept within the neurons. It has been shown that this type of network has high performance for a variety of machine learning applications including face recognition [25,26], multi-label text categorization [27], stock return prediction [28], traffic sign detection and recognition [29–31], and tracking [32–34].

Our VG-RAM visual tracking system mimics the biological saccadic eye movement system. It models the transformations suffered by the images captured by the eyes in its way to the Superior Colliculus (SC) of mammalian brains, and models the response of SC neurons to previously seen patterns. Such biological model is incorporated into a Tracking-Learning-Detection algorithm to address all challenges presented by long-term tracking problems. We evaluate the performance of our system using the TLD database (available at http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html), and show that it is able to achieve, in average, equivalent or superior performance than TLD.

This paper is organized as follows. After this introduction, we describe the biologically inspired VG-RAM WNN architecture for visual tracking (Section 2). In Section 3, we describe our experimental methodology and, in Section 4, we present and discuss our experimental results. Our conclusions and directions for future work follow in Section 5.

2. Visual tracking system based on VG-RAM WNN

In this section, we present our biologically inspired visual tracking system based on VG-RAM WNN. The system aims at long-term tracking of an unknown object in a video considering a single sample of the object. Basically, a bounding box around the object of interest is defined in the first frame, and the tracking system determines the object's location in subsequent frames. In addition, the system indicates the presence or not of the object in the frames. The location of the object is given by a bounding box surrounding it.

Our system is made of several components whose existence is motivated by findings of research in the area of biological vision systems [35,36]. In particular, we tried to replicate the tracking performance achieved by the human visual system. In this context, three important aspects of the human and other primates' visual system need to be highlighted: (i) the saccadic eye movement, (ii) the transformation suffered by the images captured by the eyes in the way from the retina to the Superior Colliculus, and (iii) the response of the neurons of the Superior Colliculus to patterns of interest in the visual scene. Our system tries and mimics these three aspects of biological visual systems using VG-RAM WNN neurons.

Our visual tracking neural system comprises four modules: (Re) Learning, Tracking, Detection and Validation (see Fig. 1). The (Re) Learning module is responsible for training the system with the annotated object in the first frame, and for retraining the system with a detected object in order to reinforce the current learned description of the object in the system. The Tracking module is responsible for following the object from frame to frame considering the last location of the object. The Detection module is responsible for finding the object once it returns to the scene, after occlusion for example. The Validation module is responsible for (i) deciding if it is necessary to retrain the system, (ii) if the process of tracking should continue, or (iii) if the detection process should start. Based on the decision of Validation module, the system reports either the coordinates of the object's bounding box or the information that the object is not visible in the scene.

In the following subsections, we briefly explain how we modeled the biological visual system that is responsible for the saccadic eye movements using VG-RAM WNN. Thereafter, the details of each module – (Re)Learning, Tracking, Detection, and Validation – are presented, as well as how they interact with our VG-RAM saccadic system.

2.1. Modeling the biological saccadic system with VG-RAM WNN

The saccadic eye movement is present in the visual system of primates (and most mammalians) and it is responsible for pointing

the fovea towards objects of interest [35]. The fovea is the central region of the retina that has the highest density of receptors and; therefore, it is responsible for collecting information about the point of attention. The midbrain's Superior Colliculus (SC) receives information directly from the retina and from other areas of the brain, and is responsible for controlling the saccadic eye movements [35]. These movements, when driven by cognitive behavior, are guided by the memory of the object of interest. Information present in images captured by the eyes is transferred to the SC together with some form of representation of the memory of the object of interest (coming from the frontal eye field [35]), and neurons present in the SC fire when the object is in its receptive field. Based on the activation of these neurons, a target is selected in the visual field according to a winner-takes-it-all behavior, and the saccadic eye movement is performed [36].

In previous work [30], it was shown that saccadic movements can be mathematical-computationally modeled using VG-RAM WNN. In that work, a VG-RAM WNN was used to model the behavior of the SC, and the saccadic eye movements were performed considering the neuron with the highest output. Although good results have been achieved with that approach, it had the disadvantage of promoting competition among the active neurons, i.e. in that approach, each neuron tries to point the eye to its own location of interest. In this work, we use a similar VG-RAM architecture; however, we employ a different approach for computing the coordinates of the saccadic target. Our approach does not involve competition among neurons, but rather cooperation among them and competition among possible object centers. In our saccadic system, instead of competing for attention, neurons learn their position with respect to the center of an object, and contribute with a vote for the activation of that center. The best voted center is selected as the target of the saccadic movement. Note that the winner-takes-it-all behavior is still present, but it selects the winning object center that might have votes of many neurons, instead of selecting the location of a specific winning neuron as in [30].

2.1.1. *Model of the Superior Colliculus*

We modeled the SC with VG-RAM WNNs. These networks comprise a set of Neural Layers interconnected through a set of Synapses (Fig. 2). The output of a Neural Layer is represented by the activation values – labels in the case of VG-RAM WNNs – of each one of its neurons. The primary goal of a neuron in a VG-RAM WNN is to sample an Input Layer with its synapses, and to learn (during training) or to generate (during test) a label representing the non-binary pattern present within the neuron's Receptive Field, i.e. the region of the Input Layer sampled by the neuron's synapses (see Fig. 2). The conversion of non-binary values into binary values is achieved in VG-RAM WNNs through synapse mapping-functions. Such functions transform the non-binary values sampled by the set of synapses of each neuron into vectors of bits, one bit per synapse. VG-RAM WNN neurons learn input–output pairs, composed of these binary input vectors and associated output labels (set during training). These pairs are stored into lookup tables inside the neurons, which form the VG-RAM knowledge repository. This is in contrast with standard (weighted) neural networks, where the knowledge repository is formed by the weights of the set of synapses. During test, a VG-RAM neuron compares an input binary vector with all inputs in the input–output pairs learned – the neuron output is set to the label of the nearest pair according to the Hamming distance. See more about VG-RAM WNN in [23,24].

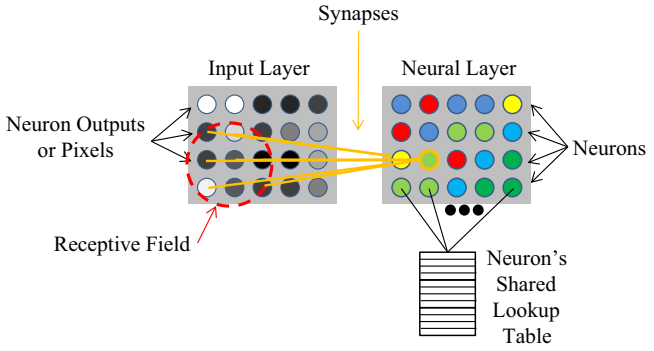


Fig. 2. Our SC model has a single VG-RAM Neural Layer. It comprises many neurons that are connected to an Input Layer by a set of Synapses. The set of synapses of a VG-RAM neuron sample the Input Layer and generates a binary input vector according to the neuron's Receptive Field and synapse mapping function. Each neuron shows an output activation value (Neural Layer colored circles) read from the Shared Lookup Table using its binary input vector as key. This label value corresponds to the output of the input–output pair (stored in the Shared Lookup Table during training) whose input is the nearest to the current binary input vector extracted from the Input Layer by the neuron's synapses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

To mimic the neurons of the SC, all neurons of our visual tracking system need to share the knowledge about the object of interest, since they have to signal its presence in their Receptive Field. It is not clear in the neurophysiology and neuroanatomy literature how the knowledge about objects of interest is conveyed to the SC, neither how it is represented in the SC, in the frontal eye field, or elsewhere in the brain. In our visual tracking system, all neurons have the same synaptical interconnection pattern and synapse mapping-function; therefore, the knowledge about the object of interest is represented in the same way in the lookup tables of all neurons of our system. Actually, all neurons of our system share the same knowledge, i.e. they share the same lookup table (they are shared-memory VG-RAM neurons). Therefore, the VG-RAM WNN we employ is, in fact, a convolutional WNN [37]. As in [38], the memory of the neurons is composed of look-up tables that store sets of learned input–output pairs. However, instead of having one private table for each neuron, our visual tracking system follows the architecture suggested in [30], which uses a Shared Lookup Table for all neurons (see Fig. 2).

To model the SC, we chose to use a VG-RAM WNN architecture similar to the one proposed by De Souza et al. for traffic sign detection [30]. Our architecture has the same main parameters, i.e. a single Neural Layer of size 65×48-neurons with 256 synapses each, directly connected to an Input Layer of size 201×201-pixels. In addition, our architecture follows the same synaptic mapping function used in [30], i.e. it uses *minchinton* cells [39] to transform non-binary pixel values from the Input Layer into the binary input vectors of each neuron. Minchinton cells compare the values of the target pixel of two synapses and returns 1 or 0 depending on which synapse is connected to the pixel of larger value. As in [30], the synaptic interconnection pattern of each neuron of our visual tracking system follows a random bi-dimensional Normal distribution (with standard deviation σ equals to 10 pixels) centered at the center of its Receptive Field. The center of the Receptive Fields is mapped to the Input Layer by an inverse log-polar function. This function creates the equivalent of a fovea for our system, since it augments the number of neurons with Receptive Field over the more central region of the Input Layer. Please refer to [30] for an extended description.

Although the VG-RAM WNN architecture of our system is similar to the one proposed in [30], it operates in a significantly different manner, as described below.

The Input Layer of our visual tracking system represents a cropped portion of a Gaussian filtered grayscale version of a frame of video (see Fig. 3). The cropping is performed considering the size of the Input Layer as well as the current Center of Attention,

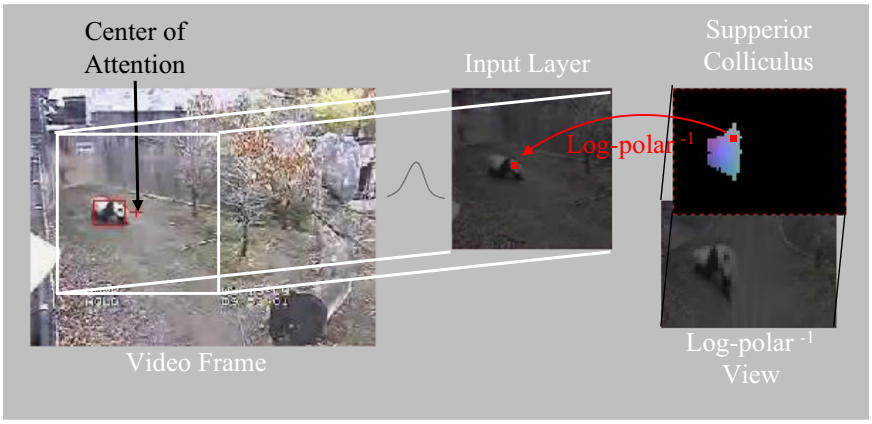


Fig. 3. Our model of the Superior Colliculus (SC). The neurons of the Neural Layer (top left), which represents the SC in our system, observe the Input Layer with their synapses. The center of the Receptive Field of these neurons, built by a set of synapses distributed according to a random Normal distribution, is mapped to the Input Layer according to an inverse log-polar function. This function creates the equivalent of a fovea for our system, since it augments the number of neurons with Receptive Field over the more central region of the Input Layer. The Input Layer, which represents the retina in our system, receives a Gaussian filtered copy of a squared region of each video frame. The center of this squared region is the Center of Attention of our system (its fovea). The neurons of the Neuron Layer fire (output a label with color different from black) when the object of interest is in their field of view. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

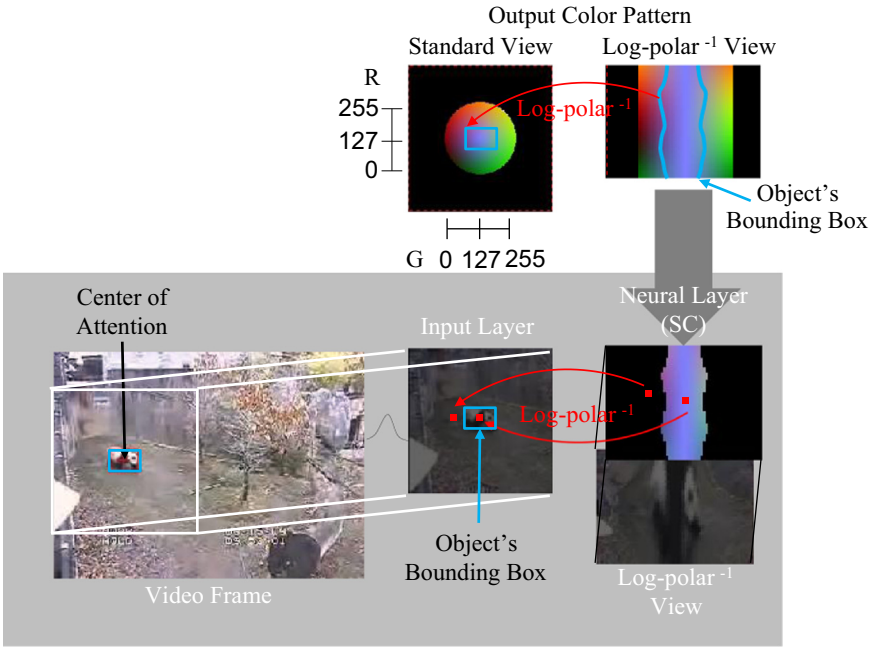


Fig. 4. Illustration of the training phase. The Center of Attention is firstly moved to the center of the object to be learned. Subsequently, the Neural Layer is painted with the expected output color of each neuron. Neurons falling inside the object's bounding box are considered activated and have a color different from black associated with them. A neuron falling outside the object's bounding box is considered deactivated and have the color black. Colors are organized so that an activated neuron has a color that tells what is this neuron's displacement from the center of the object of interest. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

i.e. the center of the Input Layer. Regions of the Input Layer falling outside the video frame are filled with black (zeroed) pixels. Although the input vector of each neuron of our system is computed in the same way as in [30], the output label is computed differently during training and treated differently during test (system recollection of the object of interest). Instead of storing and responding with grayscale label-values, each neuron stores and responds with color label-values (whose function is described in details below), i.e. four-byte integers with RGB values stored in the three least significant bytes.

During training, firstly, a video frame is given and the Center of Attention of our system is manually moved to the center of the bounding box of the object of interest. Subsequently, the Neural Layer is painted with the expected output color of each neuron that tells the position of the center of the Receptive Field of the neuron with respect to the center of the object's bounding box (see Fig. 4). Finally, the neurons are trained to output either black, if the center of their Receptive Field lies outside the object's bounding box, or to output the specific color that tells their displacement with respect to the center of the object's bounding box. Therefore, neurons painted black learn about the image background, while neurons painted with any other color learn about the object image.

The Neural Layer color pattern used for training the system is a RGB image with a special meaning for each one of the three color-channels: red channel, r_c; green channel, g_c; and blue channel, b_c (see Fig. 4). r_c goes from 0 to 254 and represents displacements in the vertical axis of the image pattern, where the 0 value indicates

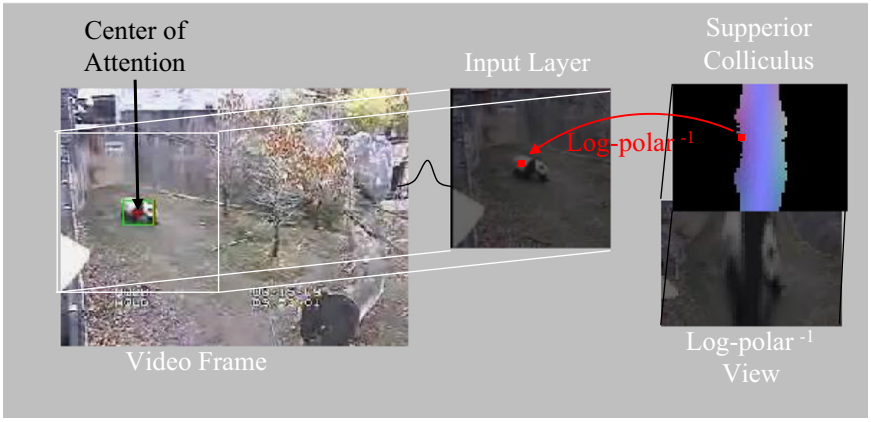


Fig. 5. Illustration of the test phase with the Center of Attention close to the center of the object of interest (panda). (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

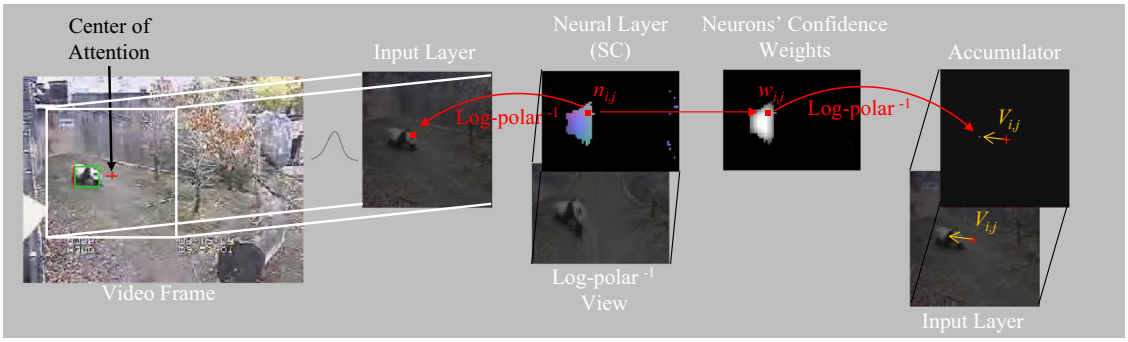


Fig. 6. Illustration of the architecture used for determining the target point for the saccadic movement. Each activated neuron, $n_{i,j}$, of the Neural Layer contributes with a vote for the activation of the probable location, $V_{i,j}$, of the center of the object of interest. Each vote is weighted by a measure of confidence of the response of a neuron, $w_{i,j}$. The votes are stored into an accumulator matrix, where its cell represents the possible spatial locations of the object of interest. The most voted location is chosen as target of the saccadic movement.

the lowest part of the pattern, the value 127 indicates the vertical center of the object of interest, and the value 254 indicates the highest part of the pattern. The value 255 is not used to make the number of elements odd and allow a specific center, 127. g_c goes from 0 to 254 and represents displacements in the horizontal axis of the image, where the 0 value indicates the left most part of the pattern, the value 127 indicates the horizontal center of the object of interest, and the value 254 indicates the rightmost part of the pattern. b_c channel goes from 0 to 255 and represents the inverse distance from the center of the object of interest in all directions, where the value 255 indicates the center of the object of interest and the value 0 indicates the furthest part of the image pattern.

This color organization enables the association of the output label of each neuron, $n_{i,j}$, with a specific displacement vector, $V_{i,j}=(x, y)$, that indicates the displacement between the system's Center of Attention and the center of the object of interest. This vector can be computed by the function, F, of the red and green channels of the neuron output, and the coordinates of the neuron in the Neuron Layer, as formulated by Eq. (1):

$$V_{i,j} = F(r_c, g_c, i, j). \quad (1)$$

During test, an input image is given and the Center of Attention is firstly moved to the region to be searched. Subsequently, the neurons respond according to the binary input vector sampled from their Receptive Fields. Neurons respond with black when the pattern seen in their receptive fields is similar to previously learned regions of background, and they respond with a color different than black when the pattern seen in their receptive fields is similar to previously learned regions of the object of interest. Activated neurons offer information about the precise location of the object of interest's center. This information is coded as the values of the red, r_c, and green, g_c, color channels. Fig. 3 shows an example of the test phase with the Center of Attention somewhat far from the center of the object of interest (panda), and Fig. 5 shows an example of the test phase with the Center of Attention close to the center of the object of interest (the panda bear).

2.1.2. The saccadic movement

In the previous section, it was showed how neurons learn about the center of an object of interest and respond to a given image. In this section, we show how the neurons collaborate to determine the center of a learned object, i.e. the target of a saccadic movement of our SC model.

In our model of the SC, each activated neuron $n_{i,j}$ contributes with a vote [40] for the activation of the probable location, $V_{i,j}$, of the center of the object of interest. Each vote is weighted by a measure of confidence, $w_{i,j}$, associated with each neuron $n_{i,j}$ (see details below). The votes are accumulated in an initially zeroed accumulator matrix. This matrix has the same dimensions of the Input Layer and represents all possible spatial locations that the center of the object of interest may be if it is inside the Input Layer. Finally, a winner-takes-it-all procedure is run over the accumulator matrix for choosing the most likely location of the center of the object of interest. In other words, the coordinates of the most voted cell of the accumulator matrix are selected as target coordinates of the saccadic movement. To measure the confidence of the saccadic movement (see below), the score accumulated in the winner cell is used. Fig. 6 shows an overview of the process of determining the target of the saccadic movement. In the following paragraphs, each step of this process is described in details.

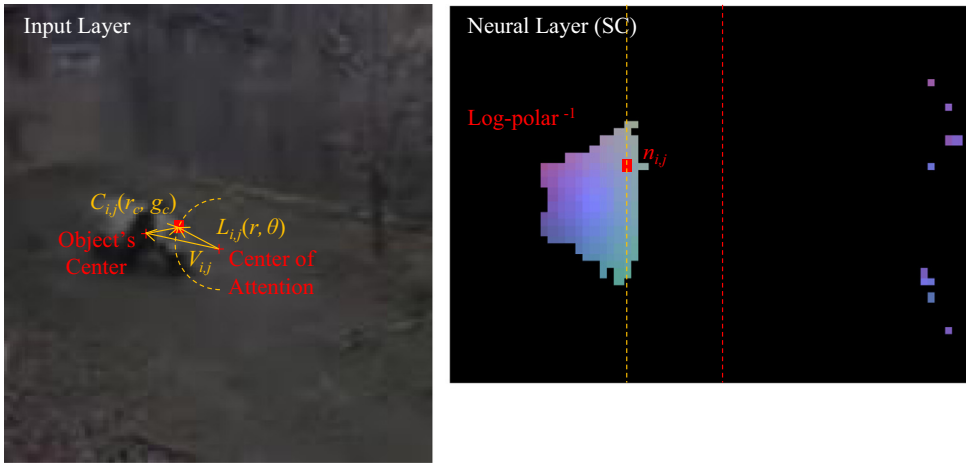


Fig. 7. The displacement $V_{i,j}$ is obtained by adding the vector $L_{i,j}(r,\theta)$ with the vector - $C_{i,j}(r_c, g_c)$. $L_{i,j}(r,\theta)$ corresponds to the location of the center of the receptive field of the neuron $n_{i,j}$ in the Input Layer. $C_{i,j}(r_c, g_c)$ corresponds to the displacement from center of the object of interest to the center of the receptive field of neuron $n_{i,j}$.

To be able to contribute with a vote for the probable location of the center of the object of interest, each neuron has to decide on a location in the Input Layer to vote for. Such a location is represented by the displacement vector, $V_{i,j}$, from the current Center of Attention (the center of the Input Layer). The displacement, $V_{i,j}$, of each active neuron, $n_{i,j}$, is computed combining the actual location of the neuron in the Neuron Layer with the output label of the neuron, i.e. with the learned displacement from the center of the object of interest, represented by the neuron output color. More precisely, $V_{i,j}$ is given by:

$$V_{i,j} = F(r_c, g_c, i, j) = L_{i,j}(r,\theta) - C_{i,j}(r_c, g_c) \quad (2)$$

where $L_{i,j}(r, \theta)$ is a vector representing the location of the center of the receptive field of the neuron $n_{i,j}$ in the Input Layer with respect to the Center of Attention; and, $C_{i,j}(r_c, g_c)$ is a vector representing the displacement from the center of the object of interest to the center of the receptive field of the neuron $n_{i,j}$ in the Input Layer (see Fig. 7).

$L_{i,j}(r, \theta)$ is, in fact, the inverse log-polar mapping from the coordinates (i, j) of the neuron $n_{i,j}$ of the Neuron Layer, N, of $m \times n$ neurons, to the coordinates (k, l) of the pixel $\varphi_{k,l}$ of the Input Layer, Φ, of $u \times v$ pixels, where (see [30] for details):

$$k = \frac{u}{2} + r \cdot \cos(\theta) \quad \text{and} \quad (3)$$

$$l = \frac{v}{2} + r \cdot sen(\theta); \quad (4)$$

$$r = \frac{u}{2} \cdot \left(\frac{\alpha^{\left|\frac{i-m/2}{m/2}\right|} - 1}{\alpha - 1} \right) \quad \text{and} \quad (5)$$

$$\theta = \begin{cases} \pi \cdot \left(\frac{3n}{2} - \frac{j}{n}\right) + \frac{\pi}{2n} \; ; & \text{if } k < \frac{m}{2} \\ \pi \cdot \left(\frac{3n}{2} + \frac{j}{n}\right) + \frac{\pi}{2n} \; ; & \text{if } k > \frac{m}{2}; \end{cases} \quad (6)$$

and α is the log-factor of the inverse log-polar mapping.

The vector $C_{i,j}(r_c, g_c)$ is the displacement that was learned during training, which can be calculated using the output color of $n_{i,j}$, as shown in the equation below (see Section 2.1.1):

$$C(r_c, g_c) = (r_c - 127, g_c - 127). \quad (7)$$

If the Center of Attention coincides with the center of the object of interest, $L_{i,j}(r, \theta) = C_{i,j}(r_c, g_c)$ and $V_{i,j} = (0, 0)$.

The weights of the neurons' votes, $w_{i,j}$, are calculated considering the proximity of two active neurons in N and the proximity of their targets in Φ. This assumption relies on the fact that, if two neighbor neurons, $n_{i,j}$ and $n_{i,j+1}$, have appropriate activation colors, they should both signal the same coordinates of the center of the object of interest, i.e., $V_{i,j} = V_{i,j+1}$. Expanding this assumption to all neighbors of the neuron $n_{i,j}$ and considering a neighborhood window of radius s (a window of size $2s+1^{*}2s+1$ pixels), the weight $w_{i,j}$ could be given by:

$$w_{i,j} = e^{-T_{i,j}/\left(2*\beta^2\right)}, \quad \text{and} \quad (8)$$

$$T_{i,j} = \sum_{a=-s}^{s} \sum_{b=-s}^{s} \left| V_{i,j} - V_{i+a,j+b} \right|, \quad (9)$$

where β^2 is a parameter of our model of the SC. With such a weight function, the neurons that respond in accordance with its neighbors will have higher confidence weights. In our implementation, s was empirically chosen to be equal to 4 and β to be equal to 10.

To compute the vote of each neuron $n_{i,j}$, and determine the target of the saccade, firstly $V_{i,j}$ is computed for all neurons. Then, $w_{i,j}$ is computed for all neurons. Subsequently, the cell of the accumulator matrix corresponding to the location indicated by the vector $V_{i,j}$ of each neuron is incremented with the corresponding weight $w_{i,j}$ of each neuron $n_{i,j}$ (see Fig. 8). Finally, the location of the most voted cell of the accumulator is selected as the winner displacement, V, and the saccadic movement is performed by displacing the current Center of Attention of V. The accumulated value of the winner cell is further used as the confidence of the saccadic movement.

2.2. (Re)Learning

The (Re)Learning module is responsible for training the Neural Layer of our SC model to detect the object of interest. There are two training moments. The first is the initial training (Learning phase), in which the system learns the manually annotated object in the first frame of the video. The second is the reinforcement training (Relearning phase), in which the system relearns the appearance of the object of interest in order to adapt to eventual changes, e.g. changes of viewpoint, illumination, geometry, etc. Given this two training moments and the natural behavior of the VG-RAM, the new appearance of the object slowly overwrites its old appearance over time since the memory entries of the VG-RAM neurons are randomly overwritten once the neuron reaches its maximum learning capacity.

The initial training of the system is performed as described in Section 2.1.1, i.e. the Center of Attention is manually moved to the center of the object of interest, and the neurons are trained. Each

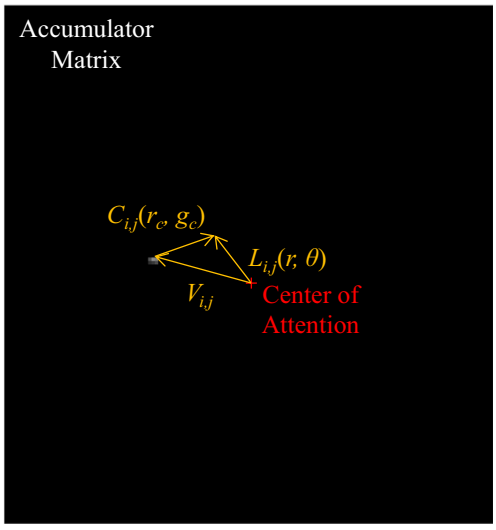


Fig. 8. Illustration of the accumulation of votes for the possible spatial locations of the object of interest. Each cell in the accumulator corresponding to the location indicated by the displacement vector $V_{i,j} = L_{i,j}(r,\theta) - C_{i,j}(r_c,g_c)$ of each neuron $n_{i,j}$ is incremented with the corresponding weight $w_{i,j}$ of $n_{i,j}$.

neuron adds its own description (input–output pair) of the current state of the Input Layer into the Shared Lookup Table of the Neuron Layer. The relearning phase on the other hand, is automatically triggered by the Validation Module after the system has been running for a while (Section 2.5), with no guarantee of a good alignment between the Center of Attention and the center of the object of interest. The reinforcement training relies on previous guesses of the system about the location of the object of interest (i.e. saccadic movements). Therefore, it is necessary to try to ensure that a good guess about the object's location is used to retrain the system. To accomplish that, a temporal window considering the three last frames is used to choose the best object center as soon as the Validation module fires the need for relearning. The video frame and saccadic target with best confidence in the temporal window is chosen for retraining the system. In addition, a fine search is performed around the saccadic target (considering a 3×3-pixels search window) of the chosen frame in order to avoid centralization problems. In this fine search, the Center of Attention is moved to each one of the pixels of the search window and accumulator matrix is recomputed for each pixel. The pixel with highest confidence value is chosen as the center of the object of interest for retraining the system.

In order to avoid retraining the system with improper images of the object of interest, some restrictions are imposed for retraining. The first restriction is the minimum confidence for retraining. The confidence achieved by the target pixel must be higher than a threshold. The second restriction is the minimum number of frames between retrains. The system can only retrain after 10 frames have passed from the last retraining. This avoids saturating the network memory with too many examples of the same object appearance. The last restriction is on the spatial location of the object. The object of interest cannot be too close to the edges of the input image during retrain. This avoids training the system with parts of the object of interest falling outside the input image.

The last important point about retraining is the size of the Shared Lookup Table of the Neuron Layer. In our model of the SC, we used a Neuron Layer, N, of size $m \times n = 65 \times 48$ neurons and use a Shared Lookup Table of size $65 \times 48 \times 32$. This amount of entries is equivalent to 32 lines per neuron in the Shared Lookup Table, i.e. our SC model has memory to hold the first 32 full representations of the object of interest. When the Shared Lookup Table is full and retraining is performed, each trained neuron replaces entries of Shared Lookup Table selected randomly. This allows graceful degradation of the previous contents of the Shared Lookup Table.

2.3. Tracking

The tracking module is responsible for following the object from frame to frame. One important restriction of this module is that the object of interest has to be visible in the previous frame. Basically, the location of the object in the previous frame is used as initial Center of Attention of the current frame. In order to be able to locate the object in the current frame, the object cannot move from the previous location to outside the field of view defined by the Input Layer.

2.3.1. Tracking steps

In general, the tracking procedure comprises three steps that are performed once the Center of Attention has been initialized in the current frame: (i) the first sequence of tracking saccades, (ii) the adjustment of scale, and (iii) the second sequence of tracking saccades.

The initial sequence of tracking saccades comprises more than a single saccade to allow for better alignment of the center of the object of interest with the center of the Input Layer. Once the center of the object of interest gets closer to the fovea region of our system (the Center of Attention), the log-polar mapping propitiates more accurate movements and, consequently, smaller and more precise saccades. The initial sequence of tracking saccades continues until there is no change in the position of the Center of Attention, or until a maximum of four movements is performed.

To ensure that the network is able to identify an object of interest correctly, its image must have a similar size inside the Input Layer throughout the entire video due to the color pattern learnt by the neurons (see Section 2.1.1). Therefore, the tracking process must ensure that the scale of the image of the object of interest is properly adjusted across time. With proper scale adjustment, the image of the object of interest has always the same size in the Input Layer. Details of the procedure of adjustment of scale are presented in the following subsection.

After the adjustment of scale, the second sequence of tracking saccades is performed to refine the location of the center of the object of interest. After this step of operation of the tracking module, it outputs, for the current frame, (i) the Center of Attention that best represents the center of the object of interest, (ii) the scale of the image of the object of interest, and (iii) the confidence of the last saccadic movement. The bounding box of the image of the object of interest has a fixed aspect ratio. This aspect ratio is given by the initial bounding box of the object of interest. The size of the bounding box of the object of interest in each frame is given by the current scale of the object.

2.3.2. Scale estimation

The standard way of selecting a proper scale for searching an object of interest in an image is simply to try the detection of the object with many different scales. Although such an approach brings good results, it requires running the detection process many times, which can be costly. To avoid trying various scales, we propose a closed-form solution to estimate the scale adjustment of an object from one frame to the other.

The proposed method follows the accumulator matrix idea employed for the saccadic movements. However, instead of computing a spatial displacement, a scale factor is computed. A one-dimensional accumulator vector is used to count votes for all possible scales of the object. The central position of the accumulator vector represents the object in the original training scale (scale factor equals to one) and positions above or below its center increase or decrease the scale factor of 0.02, respectively.

In general, each activated neuron $n_{i,j}$ of the Neural Layer contributes with a vote for the probable scale factor $z_{i,j}$ of the object.

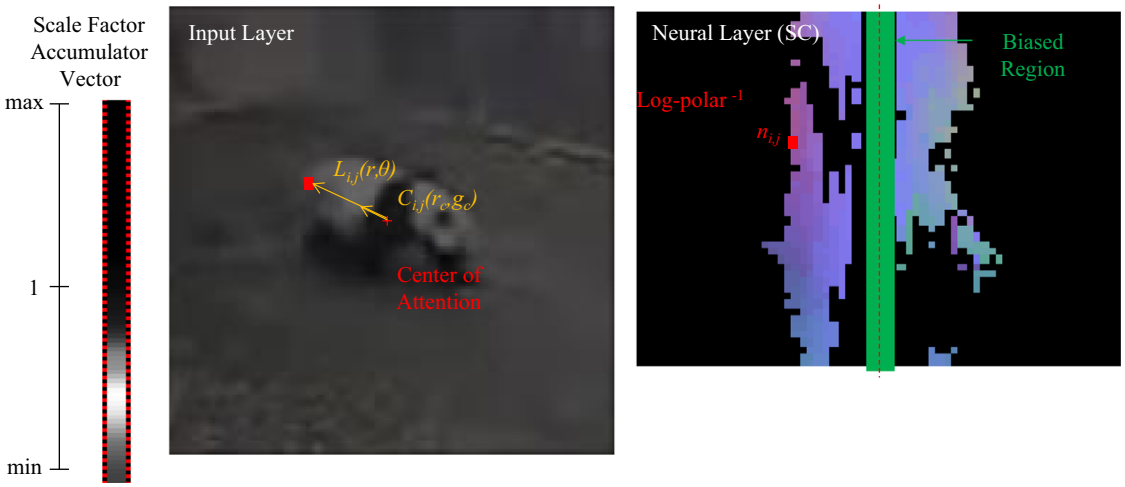


Fig. 9. Illustration of our automatic scale factor adjustment mechanism. Each position in the scale factor accumulator vector corresponds to the scale indicated by $z_{i,j}=|C_{i,j}(r_c, g_c)|/|L_{i,j}(r, \theta)|$. This position is incremented according to the corresponding weight $w_{i,j}$ of $n_{i,j}$. A Gaussian filter is used for smoothing out the data in the scale factor accumulator vector. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

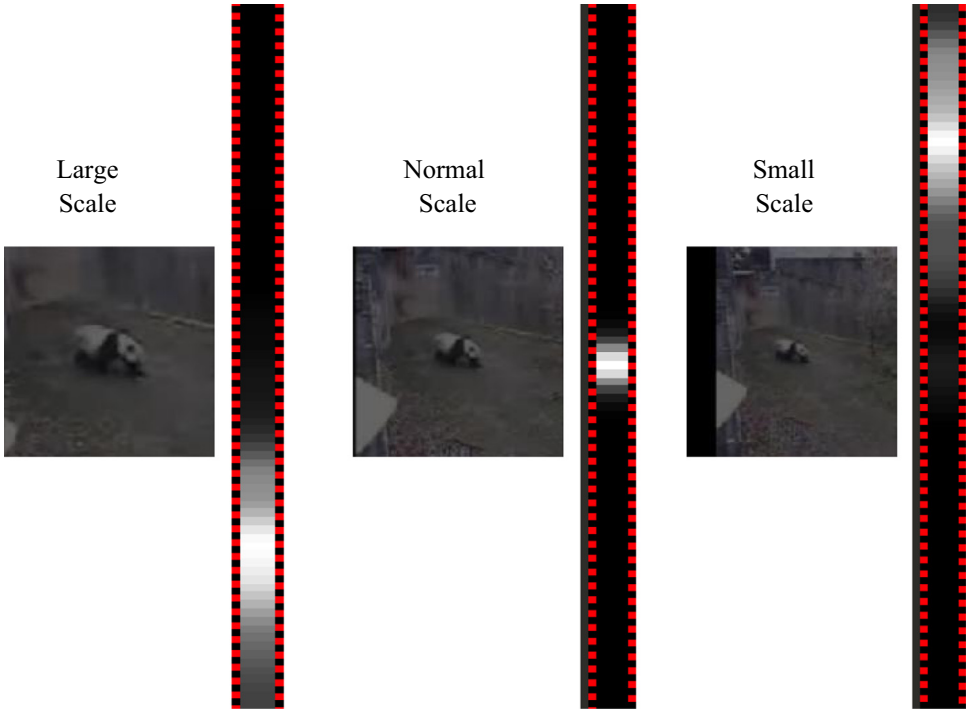
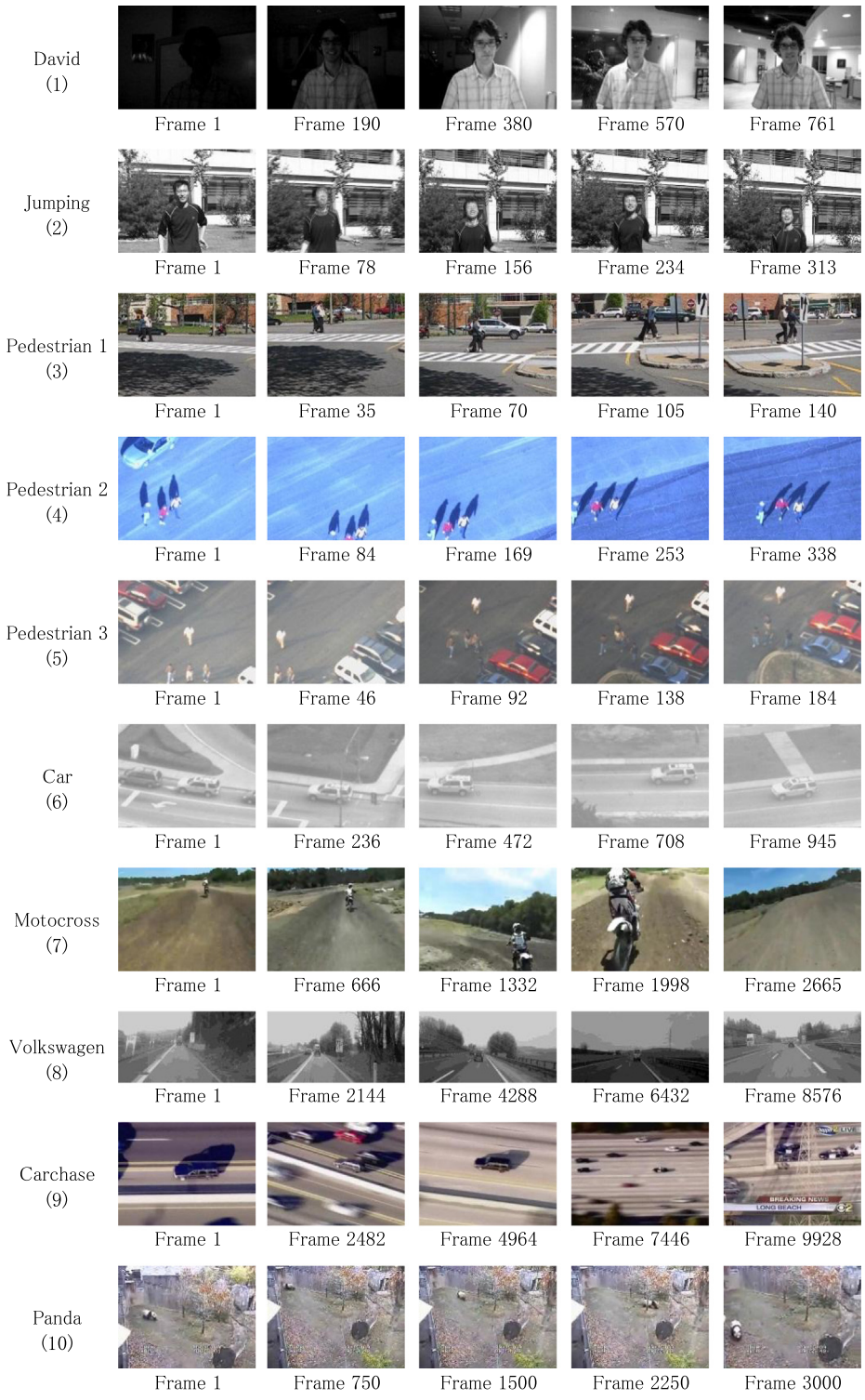


Fig. 10. Objects in different scales with their respective scale factor accumulator vector state.

Each vote is weighted by a measure of confidence of the response of its neuron $w_{i,j}$, as showed previously. The votes are stored in the scale factor accumulator vector. A Gaussian filter smooths the contributions to the accumulator vector. Finally, a winner-takes-it-all procedure is run over the scale factor accumulator vector for choosing the current scale of the object of interest. The selected scale factor is not directly used. It is firstly converted into either (i) not changed, (ii) 5% bigger, or (iii) 5% smaller than the previous scale factor. If the selected scale factor is within one vector position from the middle position of the accumulator vector, the scale factor is defined as not changed. For scale factors falling above this range, the increment or decrement of 5% of the current scale factor is applied depending on whether the accumulator vector position is above or below the middle position. Fig. 9 illustrates our mechanism for automatic scale factor adjustment.

The proposed method requires that the Center of Attention is already positioned in the center of the object, or is at least close to it. Considering this situation, one can assume that, if the color of an active neuron is in the correct geometric position in the Neural Layer – i.e. the same geometric position of this color during initial training –, the vector $L_{i,j}(r,\theta)$ should be equal to the vector $C_{i,j}(r_c, g_c)$ (see Section 2.1.2). This corresponds to a scale factor of one. Otherwise, one vector should be bigger than the other is and the scale factor should be either smaller or bigger than one. Therefore, a scale factor, $z_{i,j}$, suggested by the color and position of neuron $n_{i,j}$, could be estimated by dividing the magnitude of one vector by the

Table 1
Sample frames of the videos of the TLD dataset.



magnitude of the other as in the equation below:

$$z_{i,j} = |C_{i,j}(r_c, g_c)| / |L_{i,j}(r, \theta)|. \quad (10)$$

Since the central portion of the object is represented by the center of the log-polar mapping, changes in scale cannot be properly captured in this region due to too much amplification (or infinite amplification when $|L_{i,j}(r, \theta)| = 0$), biasing the resultant scale factor. To avoid such a bias, the contribution of the neurons located in the center on the Neural Layer (neurons with center of receptive field falling within 5 pixels radius from the Center of Attention) is not added to the accumulator vector (see green region in Fig. 9). Moreover, the proposed method depends on the

correct alignment of the center of the object of interest. Therefore, it should be limited to a certain range of scale factor magnitudes. Our system cannot handle abrupt changes in object size because the synaptic pattern of the receptive field of the neurons has fixed size.

Fig. 10 shows examples of the response of the accumulator vector for objects in different scales (large, normal, and small).

2.4. Detection

The detection module is responsible for finding the object once it goes out of scene, is occluded or cannot be identified, and then gets back to a state suitable for tracking. In order to find the object, it is necessary to perform a search across the frame. Since the field of view of the Input Layer may not cover the entire frame (its size varies with the scale factor), it may not be possible to locate the object with a single sequence of saccadic movements. Therefore, it is necessary to explore the entire frame looking for the object.

The image exploration is performed by dividing the frame in sub-regions to be covered by the field of view defined by the Input Layer. During exploration, the Center of Attention is placed in the middle of each sub-region, a sequence of saccadic movements is performed, and the confidence of each saccadic movement is stored. Since the object might reappear in a different scale, the same procedure is repeated for every possible object scale: in our experiments, from a bounding box of 20 pixels until a bounding box that is 5 times larger than the original, incremented by factors of 1.1. After analyzing all regions and all scales, the saccadic target with the highest confidence and its associated scale are selected. If its confidence is above a threshold, tracking continues with this target.

2.5. Validation

The Validation module is responsible for examining the results of the operation of the previous modules and deciding on the next step to be performed. The decisions taken by the Validation module are based on the confidence of the last saccadic movement, what was the last module which operated, and the current state of the system. The system has two states: *object is visible* and *object is not visible*. If the confidence of the last saccadic movement goes below a threshold Ψ, the state is changed to *object is not visible* and no bounding box is shown. Otherwise, the state is *object is visible* and a bounding box, centered in the Center of Attention and sized according to the current scale, is plotted.

For each new video frame, the modules of the system operate as follows. After the operation of the Tracking module, the Validation module can: activate the operation of the Tracking module again, in case of the *object is visible*; activate the Detection module, in case of the *object is not visible*; or activate the (Re)Learning module, in case the of *object is visible* and the confidence of the saccadic movement is below the retraining threshold, Ω. Note that the retraining threshold, Ω, is larger (easier to transpose) than the threshold Ψ. After the operation of the (Re)Learning module, the Validation module activates the Tracking module. After the operation of the Detection module, the Validation module can either activate the same module again, in case of the *object is not visible*, or activate the Tracking module, in case of the *object is visible*. This last transition requires a confidence higher than 2.5 times the threshold Ω. A 2.5 times higher confidence value is used in order to ensure that the object found is indeed the object of interest.

3. Experimental methodology

To evaluate the performance of our system, we used the TLD dataset (available at http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/tld.html). The TLD dataset contains 10 benchmarks, with videos showing various types of objects. The challenging conditions of the TLD dataset include abrupt camera motion, scale/pose/illumination changes, and occlusions.

The David video consists of 761 frames and shows a person (object of interest) walking from an initially dark setting into a bright room. The Jumping video has 313 frames and presents a person jumping rope. The Pedestrian1 (140 frames), Pedestrian2 (338 frames) and Pedestrian3 (184 frames) videos show pedestrians walking in different environments. The Car (945 frames) and Volkswagen (8576 frames) videos show moving cars. The Motocross video consists of 2665 frames of a running motocross. The Carchase video has 9928 frames of a car being chased by a police car and filmed by a helicopter. The Panda video consists of 3000 frames and presents a panda moving around a closed zoo environment. Table 1 shows some frames of each one of the TLD dataset videos.

We performed experiments with our visual tracking system on all videos from the TLD database. The ground-truth bounding box of the first frame was used to perform the initial training of our system. After processing each video, all occurrences of True Positives (TP), False Positives (FP), and Frames With Objects of Interest (FWO) were counted. True Positives are the frames in which the ground truth of the database indicates the presence of the object of interest and the system *correctly points to the object*. False Positives are: (i) frames in which the ground truth of the database indicates the lack of the object of interest and the system incorrectly responds with a bounding box; or (ii) frames in which the ground truth of the database indicates the presence of the object of interest and our system do not *correctly points to the object*. Frames With Objects of Interest are frames in which the database indicates the presence of the object of interest.

The decision about whether the system *correctly points to the object* or not, employed for counting the number of TP, was made using the Jaccard coefficient [41]. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets (Eq. (11)). It can be used to measure the overlap between two bounding boxes (the one found by our system and the ground truth provided with the database). In Fig. 11, *B1* is the area of the first bounding box, *B2* the area of the second bounding box, and **I** is the area of the intersection of the two bounding boxes. The Jaccard coefficient between the two bounding boxes is their intersection divided by the area of their union (Eq. (11)). As in [19], our system *correctly points to the object* if the Jaccard coefficient was larger than 0.25.

$$\text{jaccard} = \frac{\text{B1} \cap \text{B2}}{\text{B1} \cup \text{B2}} = \frac{I}{(\text{B1}+\text{B2})-I} \tag{11}$$

Based on the measures described above, the performance of our system was evaluated using three metrics: Precision (P), Recall (R), and F-measure (F). Precision is the number of True Positives divided by the number of all positive responses ($P=\text{TP}/(\text{TP}+\text{FP})$). Recall is the number of True Positives divided by the number of object occurrences that should have been detected ($R=\text{TP}/\text{FWO}$). F-measure is the harmonic mean of Precision and Recall ($F=2*P*R/(P+R)$).

4. Experimental results and discussion

Our VG-RAM WNN model of the SC has the following parameters: (i) Shared Lookup Table size; (ii) Neuron Layer dimensions, $m \times n$;

(iii) number of synapses per neuron; (iv) Input Layer dimensions, $u \times v$; (v) the standard deviation, σ, of the bi-dimensional Normal distribution followed by the synaptic interconnection pattern of the neurons Receptive Field; (vi) log-factor, α, of the inverse log-polar mapping; and (vii) the parameters s and β employed for computing the neurons' weights. These parameters can be chosen relatively easily in an ad-hoc manner by an expert, or they can be tuned with an automatic process. In our system, we have used the following parameters in our experiments, which were chosen in an ad-hoc manner: (i) memory size equals to $32 \times (65 \times 48$ neurons); (ii) Neuron Layer dimensions equal to 65×48; (iii) the number of synapses per neuron equals to 256; (iv) Input Layer dimensions equal to 201×201; (v) σ equals to 10; (vi) α equals to 2; and (vii) s and β equals to 4 and 10, respectively. The visual tracking system has two parameters: (i) the value of Ω that regulates the retraining step; and (ii) the value of Ψ that determines the object's presence in a scene. We have used (i) Ω equals to 55, and (ii) Ψ equals to 3. All parameters were fixed for all the experiments.

We compared our tracking system with TLD [19] – all metrics, common parameters, and database were the same. The TLD values presented below were taken directly from [19].

Table 2 shows the values for Precision, Recall and F-measure of TLD and of our system for each video in the database. The last row shows the mean performance, considering the weighted average by the number of frames of each video. As the table shows, our system achieved higher average performance than TLD for all metrics considered. The TLD performance in some of the videos is better than our approach, as individual results presented in Table 2 demonstrate; however, our system outperforms TLD in most of the videos.

To show the performance of the method as a separated tracking module, Fig. 12 summarizes the output results of the tracking system per frame of each video. Results are shown as boxplot of the jaccard coefficient per valid frame (i.e. frames with object present). The boxplot uses boxes to represent the average value with one standard deviation down and one up, and the lines to represent the maximum and minimum value achieved. In general, the results presented shown that our visual tracking system obtains good results for object tracking.

In addition to the quantitative evaluation, we have made a qualitative analysis of our results obtained with our system for each video. In the following, the output of our visual tracking system is presented as a green rectangle, while the manually annotated ground truth from the database is presented as a red rectangle. To allow the visual analysis of our results, the videos were regularly sampled for building the figures presented below.

The visual tracking results of frames 1, 190, 380, 570, and 761 of the David video are shown in Fig. 13. As it can be seen, the illumination variation did not hinder our visual neural tracking from correctly identifying the object of interest (face) in the scene. As shown in Table 2, the appearance changes resulted in few false negatives.

Fig. 14 presents the visual tracking results of frames 1, 78, 156, 234, and 313 of the Jumping video. As it can be seen, the camera movement and image blur did not hinder our tracking system

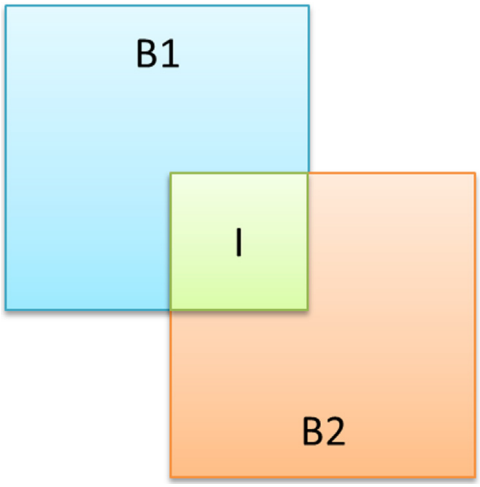


Fig. 11. Illustration of the Jaccard coefficient with 0.25 overlap between bounding boxes.

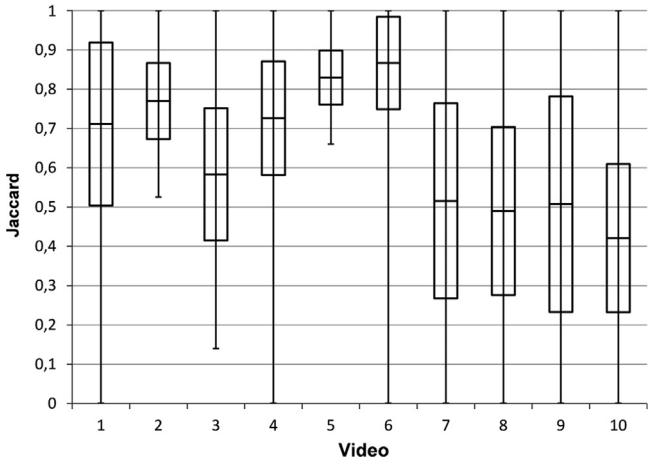


Fig. 12. Summary of the Jaccard coefficient per frame for all videos. The boxes represent the average value with one standard deviation down and one up, and the lines represent the maximum and minimum value achieved.

Table 2
Performances of TLD and of our visual tracking system.

Video	Frames	TLD			Our visual tracking system		
		Precision	Recall	*F*	Precision	Recall	*F*
David (1)	761	1.00	1.00	1	**1.00**	0.94	0.97
Jumping (2)	313	1.00	1.00	1	**1.00**	**1.00**	**1.00**
Pedestrian1 (3)	140	1.00	1.00	1	**1.00**	0.98	0.99
Pedestrian2 (4)	338	0.89	0.92	0.91	**1.00**	**0.95**	**0.97**
Pedestrian3 (5)	184	0.99	1.00	0.99	**1.00**	0.96	0.98
Car (6)	945	0.92	0.97	0.94	**0.92**	**0.95**	**0.94**
Motocross (7)	2665	0.89	0.77	0.83	0.84	**0.97**	**0.90**
Volkswagen (8)	8576	0.80	0.96	0.87	0.72	**0.85**	**0.78**
Carchase (9)	9928	0.86	0.70	0.77	**0.93**	**0.80**	**0.86**
Panda (10)	3000	0.58	0.63	0.60	**1.00**	**0.80**	**0.89**
Mean		**0.82**	**0.81**	**0.81**	**0.86**	**0.85**	**0.85**

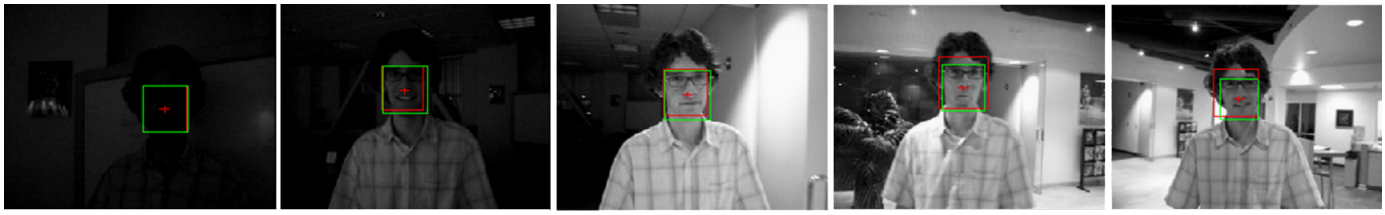


Fig. 13. Tracking results of selected frames of the David video (from left to right, Frame 1, Frame 190, Frame 380, Frame 570, and Frame 761). Our visual tracker system did not fail even in the presence of strong changes in illumination. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

Fig. 14. Tracking results of selected frames of the Jumping video (from left to right, Frame 1, Frame 78, Frame 156, Frame 234, and Frame 313).

Fig. 15. Tracking results of selected frames of the Pedestrian1 video (from left to right, Frame 1, Frame 35, Frame 70, Frame 105, and Frame 140).

Fig. 16. Tracking results of selected frames of the Pedestrian2 video (from left to right, Frame 1, Frame 84, Frame 169, Frame 253, and Frame 338).

Fig. 17. Tracking results of selected frames of the Pedestrian3 video (from left to right, Frame 1, Frame 46, Frame 92, Frame 138, and Frame 184).

from correctly identifying the object of interest (face). For this video, the system achieved the maximal possible score.

The visual tracking results of frames 1, 35, 70, 105, and 140 of the Pedestrian1 video are shown in Fig. 15. This dataset has constant abrupt appearance variations that affect the tracking Recall. The number of false negatives increases since the (Re)Learning module has a delay of 10 frames between retrains and is not able to update the representations of the object in the memory of the neurons in a sufficiently high rate. It can also be noticed that characteristics of the ground truth may have had some impact on our results. Frame 70 for example, shows a much wider bounding box ground-truth than the response given by our system.

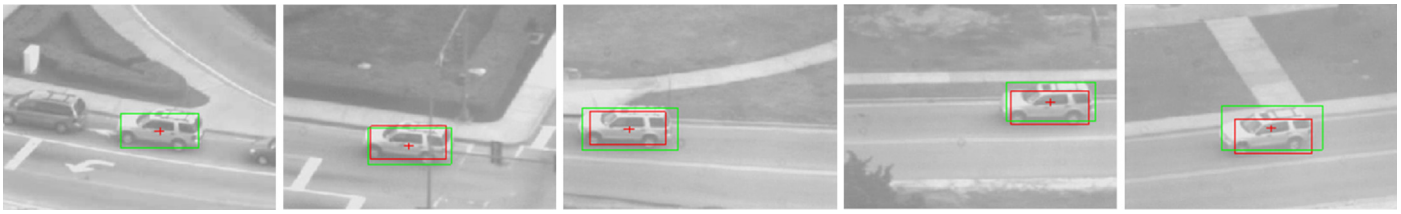


Fig. 18. Tracking results of selected frames of the Car video (from left to right, Frame 1, Frame 236, Frame 472, Frame 708, and Frame 945).

Fig. 19. Tracking results of selected frames of the Motocross video (from left to right, Frame 1, Frame 666, Frame 1332, Frame 1998, and Frame 2665).

Fig. 20. Tracking results of selected frames of the Volkswagen video (from left to right, Frame 1, Frame 2144, Frame 4288, Frame 6432, and Frame 8576).

Fig. 21. Tracking results of selected frames of the Carchase video (from left to right, Frame 1, Frame 2482, Frame 4964, Frame 7446, and Frame 9928).

Fig. 22. Tracking results of selected frames of the Panda video (from left to right, Frame 1, Frame 750, Frame 1500, Frame 2250, and Frame 3000).

However, the bounding box defined by our system circumscribes the person much better than the actual ground truth.

Fig. 16 shows the results of our system for frames 1, 84, 169, 253, and 338 of the Pedestrian2 video. Fig. 17 shows the results for frames 1, 46, 92, 138, and 184 of the Pedestrian3 video. Fig. 18 presents the result for frames 1, 236, 472, 708, and 945 of the Car video. Our visual neural tracking system performs satisfactory for all these cases (high Precision and Recall). Since in these videos the object goes out of scene, the system sustains some False Negatives, but increases the number of correct detections of the object of interest.

The visual tracking results of frames 1, 666, 1332, 1998, and 2665 for the Motocross video are presented in Fig. 19. False Positives occur due to abrupt scale changes. Our tracking system estimates the correct center of the object, but it does not estimate the scale correctly. As a result, the overlap between the bounding boxes (i.e. the Jaccard measure) is not sufficient for correct detection. It is interesting to note that our system correctly detects the object of interest in Frame 1998, but the ground truth does not indicate that the object is present in this frame. Frame 2665 shows an example of a False Positive.

Fig. 20 presents the results of frames 1, 2144, 4288, 6432, and 8576 for the Volkswagen video. Our system does not perform very well in this video since our Detector module identifies other objects with similar shapes as being the object of interest. After retraining with those objects, the system cannot recover the original object of interest anymore. An example of such situation can be seen in Frame 6432, in which a False Positive detection occurs.

Fig. 21 shows the visual tracking results of frames 1, 2482, 4964, 7446, and 9928 for the Carchase video, and Fig. 22 presents the results of frames 1, 750, 1500, 2250, and 3000 of the Panda video. As it can be seen from the images, the system tracks the objects of interest throughout the entire video.

Table 3
Discrimination of number of frames used by the different modules of the system. From left to right, the table shows for each video sequence: the total number of frames, the number of invalid, the number of frames in which the detection module was activated, and the number of frames used for retraining.

Video	Number of frames			
	Total	Invalid	Activation of detection	Retrains
David (1)	761	0	46	18
Jumping (2)	313	0	0	16
Pedestrian1 (3)	140	0	0	9
Pedestrian2 (4)	338	72	80	10
Pedestrian3 (5)	184	28	29	7
Car (6)	945	85	46	11
Motocross (7)	2665	1253	1337	88
Volkswagen (8)	8576	3435	3519	84
Carchase (9)	9928	1268	2347	218
Panda (10)	3000	270	366	96

Fig. 23. Frames 674, 773 and 787 of the car video. Frames where the ground-truth does not find the object.

In general, the results presented above show that our visual tracking system is able to obtain similar or superior results than the state-of-the-art technique for object tracking in video. Our system is able to track objects in long videos containing many challenges, including occlusions, camera movement, appearance changes, fast object movements, etc.

We have also measured the time performance of our system running experiments in a workstation with an Intel Core i7–4770 quad-core processor 3.4 GHz and 16 GB of RAM. The initial training of the system can be performed in 0.01 s, whereas the retraining of the system can be performed in 0.32 s. The higher retraining time, is due to the fact that a fine search (considering a 3×3 – pixels search window) is performed around the saccadic target in order to ensure the correct selection of the object to be trained. In this fine search, the Center of Attention is moved to each one of the pixels of the search window and the accumulator matrix is recomputed. The tracking procedure comprising three steps can be performed in 0.34 s on average: the first sequence of tracking saccades, performed in 0.12 s on average; the adjustment of scale, performed in 0.1 s on average; and the second sequence of tracking saccades, also performed in 0.12 s on average. The detection of an object is performed using around 20 scales times 4 saccades per frame, requiring 80 saccades of 0.12 s. This exhaustive approach was chosen to ensure the object would be found in the first frame possible. However, we are working on a randomic based search that can optimize this process when finding the object in the first frame possible is not mandatory.

The total number of activations of the detection module and the number of frames used for retraining for each benchmark test is shown in Table 3. When looking at the total number of activations of the detection module, we notice that it is in general not higher than the number of invalid frames, i.e. frames in which objects of interest were not present at the scene and therefore detection is necessary, except for the videos 1, 6 and 9. In the video 1, the object of interest revolves around its axis very quickly, not allowing the system to relearn the new appearance of the object of interest. Changing the state of the system to “the object is not visible” activates the detection module. In the video 6, the system correctly finds the object in frames where the ground-truth does not find, and therefore it activates the detection module a few more times than expected considering the number of invalid frames informed by ground-truth. Good examples are highlighted in Fig. 23. Finally, in the video 9, the object suffers occlusion and reappears several times with very different appearance and scale. Hence, in order ensure the detection of the correct object for such cases, the system requires a high confidence threshold of detection that consequently increases the number of times the detection module is activated. When looking at the number of frames used for retraining, we also notice that it is in general much lower than the number of valid frames, i.e. frames in which objects of interest were present at the scene. Such fact allows us to infer that the used neurons can memorize the object appearance along the video sequence even with very few samples of the object.

It is important to mention that our implementation can be optimized for taking advantage of hardware accelerators, such as GPU, FPGA or digital signal processors, improving the time performance. In addition, we are currently working on a new weightless neuron model that could further speed-up this application as much as 3 times.

5. Conclusions and future work

In this paper, we present a biologically inspired long-term object tracking system based on Virtual Generalizing RAM (VG-RAM) Weightless Neural Networks (WNN). Our VG-RAM WNN system models the biological saccadic eye movement system, i.e. the transformations suffered by the images captured by the eyes

from the retina to the Superior Colliculus, and the response of the neurons to previously seen patterns. Our experimental results showed that our approach is capable of efficiently track an object of interest in video with precision and recall equivalent or superior to the state-of-the-art methods. Our proposed machine-learning framework, in contrast to related techniques, is simple to be implemented and has close similarities with the biological saccadic system in several of its aspects.

As future work, we would like approximate our system to the biological system further by modeling other aspects of the human visual system, such as the smooth pursuit eye movements. In addition, we would like to perform a fine-tuning of the system parameters in order to improve tracking performance and investigate methods for improving the system runtime.

Acknowledgments

We would like to thank Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, Brazil (Grants 552630/2011-0, 308096/2010-0, and 314485/2009-0) and Fundação de Amparo à Pesquisa do Espírito Santo – FAPES, Brazil (Grant 48511579/2009, and 0690/2015) for their support to this research work.

References

[1] A. Yilmaz, O. Javed, M. Shah, Object Tracking: A Survey, ACM Comput. Surv. 38 (4) (2006).
[2] P. Fua, V. Lepetit, Monocular model-based 3D tracking of rigid objects, Comput. Graph. Vis. 1 (1) (2005) 1–89.
[3] H.M. Gomes, T. Zhang, Technology survey on video face tracking, Imaging and Multimedia Analytics in a Web and Mobile World 2014, 2014.
[4] B.D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision, in: Proceedings of International Joint Conference on Artificial Intelligence, 1981, pp. 674–679.
[5] D. Comaniciu, V. Ramesh and P. Meer, Real-time tracking of non-rigid objects using mean shift, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2000, p. 142–149.
[6] M. Ozuysal, P. Fua and V. Lepetit, Fast keypoint recognition in ten lines of code, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA, USA, 2007.
[7] S. Avidan, Support vector tracking, IEEE Trans. Pattern Anal. Mach. Intell. 26 (8) (2004) 1064–1072.
[8] R.T. Collins, Y. Liu, M. Leordeanu, Online selection of discriminative tracking features, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (2005) 1631–1643.
[9] O. Javed, S. Ali and M. Shah, Online detection and classification of moving objects using progressively improving detectors, in: Proceedings of 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, 2005, pp. 696–701.
[10] D. Ross, J. Lim, R.S. Lin, M.H. Yang, Incremental learning for robust visual tracking, Int. J. Comput. Vis. 77 (1) (2008) 125–141.
[11] A. Adam, E. Rivlin and I. Shimshoni, Robust fragments-based tracking using the integral histogram, in: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR06), vol. 1, 2006, pp. 798–805.
[12] S. Avidan, Ensemble tracking, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2) (2007) 261–271.
[13] H. Grabner, C. Leistner and H. Bischof, Semi-supervised On-Line boosting for robust tracking, in: Proceedings of the 10th Euro-pean Conference on Computer Vision, vol. 5302, Berlin, Heidelb, pp. 234–247.
[14] B. Babenko, M.-H. Yang and S. Belongie, Visual tracking with online multiple instance learning, in: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), 2009, pp. 983–990.
[15] S. Stalder, H. Grabner and L. Van Gool, Beyond semi-supervised tracking: tracking should be as simple as detection, but not simpler than recognition, in: Proceedings of IEEE International Conference on Computer Vision Workshops, 2009, pp. 1409–1416.
[16] J. Santner, C. Leistner, A. Saffari, T. Pock and H. Bischof, PROST: parallel robust online simple tracking, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 723–730.
[17] S. Hare, A. Saffar and P.H.S. Torr, Struck: Structured output tracking with kernels, in: Proceedings of IEEE International Conference on Computer Vision, 2011, pp. 263–270.
[18] Z. Kalal, J. Matas and K. Mikolajczyk, P-N learning: bootstrapping binary classifiers by structural constraints, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 49–56.
[19] Z. Kalal, K. Mikolajczyk, J. Matas, Tracking-Learning-Detection-learning-detection, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1) (2010) 1–14.
[20] P. Guo, X. Li, S. Ding, Z. Tian and X. Zhang, Adaptive and accelerated tracking-learning-detection, in: Proceedings of International Symposium on Photoelectronic Detection and Imaging, 2013.
[21] W. Cao, G. Xu, B. Lei, P. Yin and F. Dong, A multiple face detection and tracking system based on TLD, in: Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service, 2013, pp. 386–389.
[22] X.Q. Pan, H.F. Zhang, Research on target tracking based on TLD algorithm, Appl. Mech. Mater. 389 (2013) 819–822.
[23] I. Aleksander, From WISARD to MAGNUS: a family of weightless virtual neural machines, in: RAM-Based Neural Networks, J. Austin, Ed. Singapore: World Scientific, 1998, pp. 18–30.
[24] T.B. Ludemir, A.C.P.L.F. Carvalho, A.P. Braga, M.D. Souto, Weightless neural models: a review of current and past works, Neural Comput. Surv. 2 (1999) 41–61.
[25] A.F. De Souza, C. Badue, F. Pedroni, E. Oliveira, S.S. Dias, H. Oliveira and S. F. d. Souza, Face recognition with VG-RAM weightless neural networks, in: Artificial Neural Networks-ICANN, 2008, Berlin.
[26] J.L. Moraes, A.F. De Souza and C. Badue, Facial access control based on VG-RAM weightless neural networks, in: Proceedings of International Conference on Artificial Intelligence (ICAI 2011), 2011.
[27] A.F. De Souza, F. Pedroni, E. Oliveira, P.M. Ciarelli, W.F. Henrique, L. Veronese, C. Badue, Automated multi-label text categorization with VG-RAM weightless neural networks, Neurocomputing 72 (2) (2009) 2209–2217.
[28] A.F. De Souza, F.D. Freitas, A.G.C. d Coelho, Fast learning and predicting of stock returns with virtual generalized random access memory weightless neural networks, Concurr. Comput. 24 (8) (2012) 921–933.
[29] M. Berger, A. Forechi, A.F. De Souza, J.O. Neto, L. Veronese and C. Badue, Traffic sign recognition with VG-RAM Weightless Neural Networks, in: Proceedings of the 12th International Conference on Intelligent Systems Design and Applications (ISDA), 2012.
[30] A.F. De Souza, C. Fontana, F. Mutz, T.A. Oliveira, M. Berger, A. Forechi, J.O. Neto, E. Aguiar and C. Badue, Traffic sign detection with VG-RAM weightless neural networks, in: Proceedings of International Joint Conference on Neural Networks, 2013.
[31] M. Berger, A.F. De Souza, J.O. Neto, L. Veronese, V. Neves, E. Aguiar, C. Badue, Traffic sign recognition with WISARD and VG-RAM Weightless Neural Networks, J. Netw. Innov. Comput. 1 (2013) 87–98.
[32] M. Staffa, M. De Gregorio, M. Giordano and S. Rossi, Can you follow that guy?, in: Proceedings of 22th European Symposium on Artificial Neural Networks-ESANN, 2014.
[33] R.S. Moreira, N.F. Ebecken, A.S. Alves and F.M. França, Tracking targets in sea surface with the WiSARD weightless neural network, in: Proceedings of BRICS Conference on Computational Intelligence, IEEE, 2013.
[34] R. d Silva Moreira, N.F. Favilla Ebecken, Parallel wisard object tracker: a ram-based tracking system, Comput. Sci. Eng.: Int. J. 4 (2014) 1–13.
[35] E.R. Kandell, J.H. Schwartz, T.M. Jessel, S.A. Siegelbaum, A.J. Hudspeth, Principles of Neural Science, New York: McGraw-Hill, 2000.
[36] R.A. Marino, T.P. Trappenberg, M. Dorris, P. Munoz, Spatial Interactions in the Superior Colliculus predict saccade behavior in a neural field model, J. Cognit. Neurosci. 24 (2) (2012) 315–336.
[37] Y. LeCun, K. Kavukcuoglu and C. Farabet, Convolutional networks and applications in vision, in: Proceedings of Circuits and Systems (ISCAS), 2010.
[38] E. Aguiar, A. Forechi, L. Veronese, M. Berger, A.F. De Souza, C. Badue and T. Oliveira-Santos, Compressing VG-RAM WNN memory for real time applications and embedded systems, in: Proceedings of the 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, China, 2014.
[39] R.J. Mitchell, J.M. Bishop, S.K. Box, J.F. Hawker, Comparison of some methods for processing grey level data in weightless networks, in: J. Austin (Ed.), Ram-Based Neural Networks, 1998, pp. 61–70.
[40] J. Gall, A. Yao, N. Razavi, L. Gool, V. Lempitsky, Hough forests for object detection, tracking, and action recognition, IEEE Trans. Pattern Anal. Mach. Intell. 33 (11) (2011) 2188–2202.
[41] D. Winter, M. Levandowsky, Distance between sets, Nature 234 (5323) (1971) 34–35.

Mariella Berger was born in Colatina, ES, Brazil, on January 18, 1982. She received the B.Sc. degree in computer science in November 2004 and the M.Sc. degree in computer science in August 2007, both from the Universidade Federal do Espírito Santo (UFES), in Vitória, ES, Brazil. Nowadays, she is a Ph.D. student and member of the Laboratório de Computação de Alto Desempenho (LCAD – High Performance Computing Laboratory), both at UFES.

Alberto F. De Souza was born in Cachoeiro de Itapemirim, ES, Brazil, on October 27, 1963. He received the B. Eng. (Cum Laude) degree in electronics engineering and M.Sc. in systems engineering in computer science from Universidade Federal do Rio de Janeiro (COPPE/UFRJ), in Rio de Janeiro, RJ, Brazil, in 1988 and 1993, respectively; and Doctor of Philosophy (Ph.D.) in computer science from the University College London, in London, United Kingdom, in 1999. He is a professor of computer science and coordinator of the Laboratório de Computação de Alto Desempenho (High Performance Computing Laboratory) at the Universidade Federal do Espírito Santo (UFES), in Vitória, ES, Brazil. He has authored/co-authored one USA patent and over 90 publications. He has edited proceedings of four conferences (two IEEE sponsored conferences), is a Standing Member of the Steering Committee of the International Conference in Computer Architecture and High Performance Computing (SBAC-PAD), Senior Member of the IEEE, and Comendador of the Order of Rubem Braga.

Thiago Oliveira-Santos was born in Vitoria, ES, Brazil, on December 13, 1979. In 2004, he received the B.Sc. degree in computer engineering from the Universidade Federal do Espírito Santo (UFES), in Vitória, ES, Brazil. He received the M.Sc. degree in computer science from the same university in 2006. In 2011, he received a Ph. D. degree in Biomedical Engineering from the University of Bern in Swizerland, where he also worked as a post-doctoral researcher until 2013. Since then, he has been working as an Adjoint Professor at the the Department of Computer Science of UFES in Vitoria, ES, Brazil. His research activities are performed at the Laboratório de Computação de Alto Desempenho (LCAD – High Performance Computing Laboratory) and include the following topics Computer Vision, Image Processing, Computer Graphics, and Robotics.

Edilson de Aguiar was born in Vila Velha, ES, Brazil, on June 11, 1979. In March 2002, he received the B.Sc. degree in computer engineering from the Universidade Federal do Espírito Santo (UFES), in Vitória, ES, Brazil. He received the M.Sc. degree in computer science in December 2003 and the Ph.D. degree in computer science in December 2008, both from the Saarland University and the Max-Planck Institute for Computer Science, in Saarbrücken, Saarland, Germany. After that he worked as researcher from 2009 to 2010 at the Disney Research laboratory in Pittsburgh, USA. Since then, he has been with the Departamento de Computação e Eletrônica of UFES, in São Mateus, ES, Brazil, where he is an Associate Professor and researcher at the Laboratório de Computação de Alto Desempenho (LCAD – High Performance Computing Laboratory). His research interests are in the areas of Computer Graphics, Computer Vision, Image Processing and Robotics. He has been involved in research projects financed through Brazilian research agencies, such as State of Espírito Santo Research Foundation (Fundação de Apoio a Pesquisa do Estado do Espírito Santo – FAPES). He has also been in the program committee and organizing committee of national and international conferences in computer science.

Jorcy de Oliveira Neto was born in Linhares, ES, Brazil, on August 1, 1989. He received the B.Sc. degree in computer engineering in 2012 from Universidade Federal do Espírito Santo (UFES), in Vitória, ES, Brazil. Nowadays, he is working as an Electronic Equipment Engineer at Petrobras, Rio de Janeiro, RJ, Brazil.