Visual System Based on Artificial Retina for Motion Detection

Francisco Barranco, Javier Díaz, Eduardo Ros, and Begoña del Pino

Abstract—**We present a bioinspired model for detecting spatiotemporal features based on artificial retina response models. Event-driven processing is implemented using four kinds of cells encoding image contrast and temporal information. We have evaluated how the accuracy of motion processing depends on local contrast by using a multiscale and rank-order coding scheme to select the most important cues from retinal inputs. We have also developed some alternatives by integrating temporal feature results and obtained a new improved bioinspired matching algorithm with high stability, low error and low cost. Finally, we define a dynamic and versatile multimodal attention operator with which the system is driven to focus on different target features such as motion, colors, and textures.**

Index Terms—**Artificial retina, bioinspired vision, block matching, motion processing, multiscale motion estimation, rank-order coding, retinomorphic chip.**

I. Introduction

VISION IS one of the most useful and efficient sensory systems developed throughout evolution. Together with the other senses, its purpose is to provide animals with information about the world so that they can operate efficiently within a changing environment to achieve their ends and help ensure their survival. The visual system in humans is quite complex and structured in multiple processing layers that deal with different aspects of the visual input [1]. Motion processing is a key function for the survival of most living beings, and so, their visual systems have specific areas dedicated to this task [2]. Neurophysiological data [3] suggest that primary visual areas are modeled using spatiotemporal receptive filters [4]–[6] to compute motion.

Artificial processing architectures designed for tasks that biological systems solve with impressive efficiency can benefit considerably from mimicking computing strategies evolved in nature over millions of years. We have developed a visual model for motion estimation that integrates different bioinspired concepts. Simoncelli and Heeger modeled how the cortical areas (V1 and MT cells) can extract the structure of motion through local competitive neural computation [6]–[8]. We have developed this model following an engineering strategy. We integrate the multiresolution scheme carried out by the brain cells using a multiscale computation scheme [9], [10], as described by Willert [11], and use rank-order coding [12] as a natural way to choose the relevant information (salient maps according to a specific cost function). The processing scheme presented in this paper is based on Boahen's retinomorphic system, which translates visual stimuli using a population of cells that mimics retinal functions [13]–[16].

The first aim of this paper was to design and implement a bioinspired model based on artificial retinas for motion estimation using multiscale and rank-order coding computation. The system uses restricted-density saliency maps and is consequently of great interest for applications with strict bandwidth constraints.

By selecting responses in an intelligent way, we significantly improve the accuracy of the region-based matching model. As will be demonstrated in Section II, neurons that fire trains of spikes with the highest energy are the most reliable ones for region-based matching. We also implement new strategies for the matching algorithm integrating temporal information. To choose the best strategy, we define a cost function by comparing different ones in the search for that with the lowest average cost and the highest stability over different scheme parameters.

Finally, we define a versatile multimodal attention operator that focuses the matching algorithm on different features such as motion, colors, textures, etc., by preselecting the input responses for the model using rank-order coding biasing. The system scheme processing is shown in Fig. 1.

II. Bioinspired Motion Computation Based on Neural Population

Our novel development is an event-driven processing scheme based on the artificial-retina model described in Boahen's work [13], [14], [17]. These events are spikes fired by the encoding neurons when they tune different spatiotemporal features.

The retinomorphic front end uses four different kinds of neurons: sustained neurons (center-surround ON_OFF and OFF_ON units) and transient neurons (temporal INCREASING and DECREASING units). They can be seen as four output cells firing specific spikes in response to concrete stimuli. These spikes represent the input data for our model.

One of the main outcomes of this paper is the study of which kind of "retinal modality" (cell type) or group is more suitable for accurate motion estimation.

Manuscript received February 19, 2008; revised July 19, 2008. This work was supported in part by the VI EU research framework funds through European Projects DRIVSCO (IST-016276-2) and DINAM-VISION (DPI2007-61683) and Junta de Andalucía Project P06-TIC-02007. This paper was recommended by Associate Editor S. Sarkar.

The authors are with the Department of Computer Architecture and Technology, University of Granada, 18071 Granada, Spain (e-mail: fbarranco@atc.ugr.es; jdiaz@atc.ugr.es; eduardo@atc.ugr.es; bego@atc.ugr.es).

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

Digital Object Identifier 10.1109/TSMCB.2008.2009067

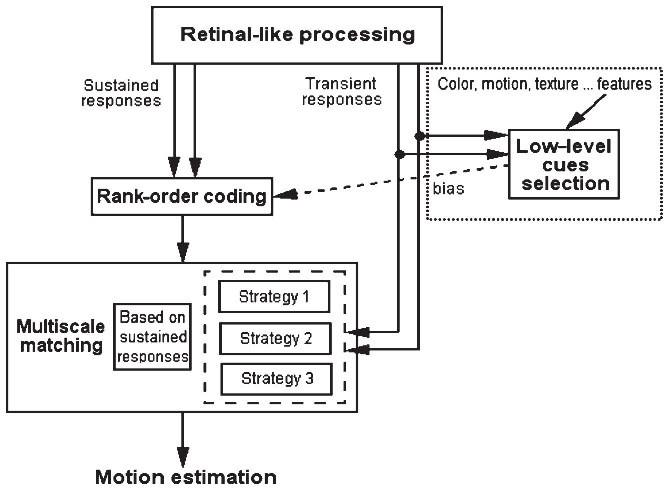


Fig. 1. System processing scheme. The dotted square is the part implemented using the attention operator described in Section IV. The cue-selection unit biases the rank order of the sustained responses according to other feature responses (e.g., transient responses, colors, textures, etc.). On the other hand, the dashed square is the unit for the different matching strategies that also integrate transient responses in order to incorporate temporal information into the matching computation.

A. Neuromorphic Chip Emulation: Sustained and Transient Neurons

We mimic Boahen's retinomorphic chip [13], [14], [17] using the following models.

1) Sustained neurons are modeled using difference of Gaussians (DoGs) with different inner and outer ratios or standard deviations to model a center-surround response model. Sustained ON–OFF neurons tune local regions with more intensity than surrounding regions; in this way, they fire a spike train with an instantaneous rate which depends on a Gaussian response. In a digital image, sustained ON–OFF neurons respond when a pixel has more intensity than the pixels in its neighborhood. Sustained OFF–ON neurons tune regions with less intensity than surrounding neighborhoods. In this case, they fire a spike train with an instantaneous rate which depends on a Gaussian response. In a digital image, sustained OFF–ON neurons respond when a pixel has less intensity than the pixels in its neighborhood. The calculus of all the sustained responses needs the definition of the different receptive fields, which are characterized by their size. The receptive field sizes for sustained neurons set the spatial features obtained or tuned (typically, we use the values of 7 and 11 pixels for the spatial center-surround Gaussian filters in our experiments).
2) Transient neurons model temporal changes with different filters. INCREASING transient neurons tune local regions where light intensity increases. DECREASING transient neurons tune local regions where intensity decreases. For transient neurons, we do not have spatial receptive fields but just temporal filters. To calculate transient responses, we use the previous and following frames, taking into account different weights for each of them. After evaluating different alternatives, we choose the one that uses the two previous, the current, and the two following frames with weights of -1, -2, 0, 2, 1, respectively.

TABLE I
PARAMETER CONFIGURATION USED FOR THE EXPERIMENTAL RESULTS. THE NUMBER OF RESPONSES IS NOT RESTRICTED BUT SET TO THE MAXIMUM NUMBER

Parameter	Value
Inner deviation (filter size)	1 (7 pixels)
Outer deviation (filter size)	1.5 (11 pixels)
Sustained threshold	5
Transient threshold	10
Temporal filter configuration	{-1, -2, 0, 1, 2}
No. of responses threshold	Maximum

In the image processing computations, errors are generated by noise, temporal aliasing, model limitations, and so on. Therefore, we need a threshold to reduce the impact of these errors. In our experiments, we used a threshold for the sustained responses with a value of 5 and another one for the transient responses with a value of 10. To produce a stable level of activation (rate of active cells), we also define a dynamic threshold. In this way we define the minimum number of responses our model needs and the other parameters are tuned to achieve it.

Only the cells with a stimulus suitably tuned to their receptive field fire a spike, and therefore, they produce a saliency map with a restricted density. We calculated the response density for different sequences and features to evaluate the activity rate (in percent) over the total number of pixels in the sequences. In standard images, using the parameters in Table I, the activity rate is around 11%–12% for each kind of response. An example is shown in Fig. 2.

Thus, our model is of potential interest for a wide variety of applications with strict bandwidth constraints. Nevertheless, activity depends strongly on the standard deviations of the DoGs used and also on the characteristics of the image.

B. Multiscale Motion Model for Spike Matching

Our motion-processing system computes a region-based matching method, as described in [19], in which we define the motion estimation as $v = (d_x = d, d_y)$ for the neighborhood of a specific cell as the best fit between the current image region and the previous one. In this way, we find the best matching region using a distance measure, the sum of squared difference (SSD) value, or the minimum mean square error (mse). In our experiments, to design the block-matching model, we implemented a full search using an exploration window of 14 pixels and a block size of 8 pixels. The block-matching example pseudocode is detailed in Appendix I-A.

The first processing stage computes the four neuron-cell responses emulating retinal processing. The second stage computes motion displacement by matching the responses produced by the sustained and transient neurons throughout every frame taken at different instants by mimicking the MT area of the cerebral cortex [6]. Bioinspired motion-estimation models are usually based on energy models due to their affinity to a neural-like description, but they are rather complex and require much

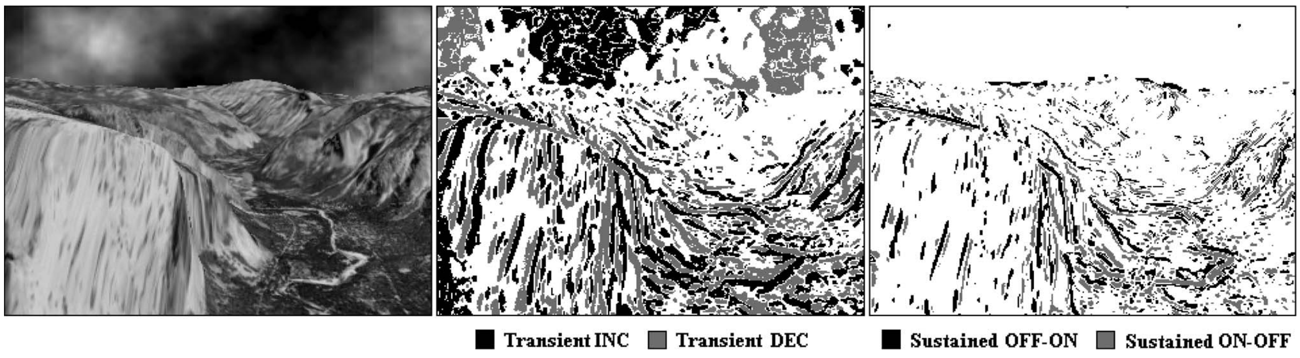


Fig. 2. Retina responses for a frame from the Yosemite sequence [18]. Activity varies from 11% to 12% for each type of neuron. (Left) Original image from the Yosemite sequence. (Center) Transient responses. (Right) Sustained responses.

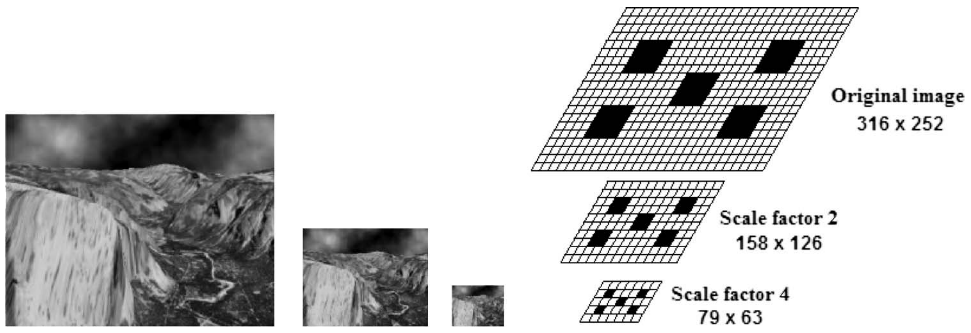


Fig. 3. (Right) Multiscale images for the Yosemite sequence. (Left) Generic scheme for multiscale with three scales and factor two, as in our tests.

more computational resources. As shown by Simoncelli [7], matching methods are equivalent to energy models but are much more efficient in terms of computational load.

For the matching process, we used a standard block-matching method, as described in [19] and [20], but relied on the neuron responses as input. We evaluated multiple matching cost functions combining the sustained and transient neuron responses. In this section, we focus on the use of sustained neurons for the matching process. In Section III, we will discuss other matching alternatives, including sustained and/or transient neurons.

We introduce here a new bioinspired element, multiscale processing [9], [10]. We use in our work three scales, one of them being the original sequence and the other two being obtained by subsampling with a factor of two. An illustrative example is shown in Fig. 3.

We apply our sustained neuron model to three well-known benchmark sequences to extract the responses (the Yosemite, the translation tree, and the diverging tree sequence). Different scales will tune better spatial features of different sizes.

At the smallest scale, the system computes the motion-estimation field, and at the next scale, this estimation is an input parameter for the new motion estimation, oversampled by a factor of two. The block matching in the next scale guides its own motion estimation using the previous motion estimation for fixing a search window of 3 pixels.

For instance, if the previous motion estimation is $v = (i, j)$ for the neuron n, as input parameter for the next scale, we will use $v = (2 \cdot i, 2 \cdot j)$ and oversample it by a factor of two (Fig. 4). This motion estimation guides the new motion-estimation computation by setting up a redefined search window for the algorithm and by exploring the region defined by (1)

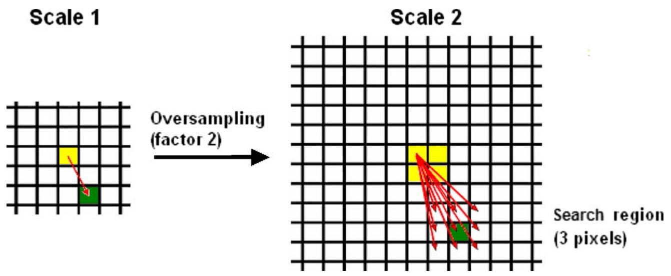


Fig. 4. Oversampling of a neuron motion estimation used for guiding the motion-estimation search at the next scale with a search window of 3 pixels, as in (1).

$$E_R = \{v = (2 \cdot i + \alpha, 2 \cdot j + \beta) : \alpha, \beta \in \{-1, 0, 1\}\} \quad (1)$$

$$E_R = \{v = (2 \cdot i + \alpha, 2 \cdot j + \beta) : \alpha, \beta \in \{-2, -1, 0, 1, 2\}\} . \quad (2)$$

The system fixes the best motion estimation for the new scale, and we follow the same process for the last scale, the original image, using a search window of 5 pixels instead of 3 pixels, as defined in (2).

The pseudocode for the multiscale approach is given in Appendix I-B. The minimum mse motion estimation updating uses a new threshold, which is set to 1.0 in our implementation.

One of the first results obtained from the last paragraphs is that if a sequence for the smallest scale contained a large object with a high contrast between it and the background, we would obtain the motion estimation for this region, but

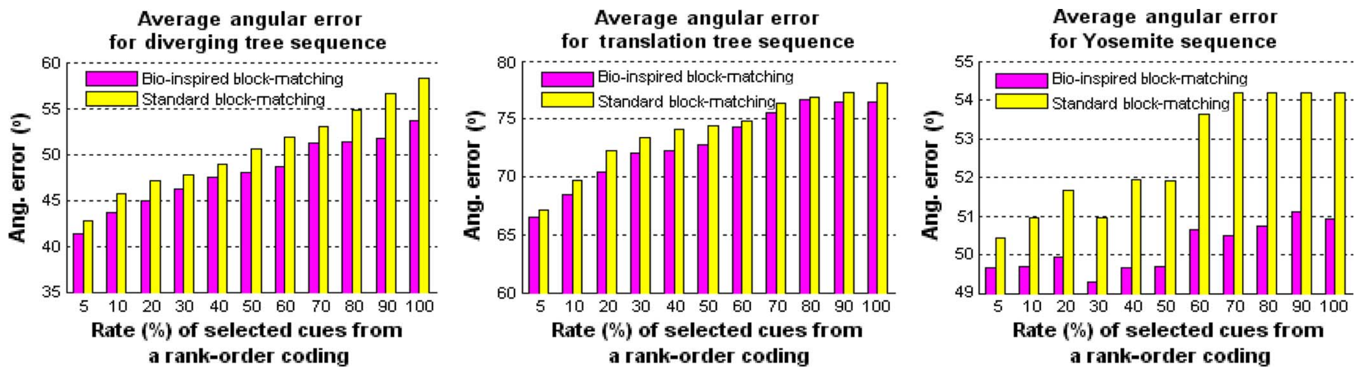


Fig. 5. Average angular error for test sequences. The rate of selected cues from rank-order coding is the percentage of responses chosen from sustained responses because of their local energy measure value (local structure support). This represents the optical flow estimation density.

if we had a small object with a low contrast, we would not detect it. This object will be invisible at the coarse scale due to the low-pass filter operations. As the motion-estimation field guides the following scale fields, we lose the smallest objects for the next scales and cannot retrieve them (this is caused by spatiotemporal aliasing).

To solve this problem, we could compute all the estimations at each scale, but the computational complexity would be very high, and there are other ways of solving these problems, as we propose in Section IV, where we explain the techniques used to reduce this computational complexity.

C. *Rank-Order Coding: Contrast Versus Accuracy Tradeoff*

Focusing on the use of sustained neurons, our first goal is to demonstrate that those neurons with larger firing responses produce more accurate motion estimations.

First of all, rank-order coding [12] consists of selecting only the most important responses or cues, sorting all the data or responses according to a measure or a criterion (cost function), the local energy measure for instance. The energy measure used for rank-order coding, in this case, is just the normalized sum of responses from every ON–OFF and OFF–ON sustained neuron (which, in a neural-like computing scheme, can be done by a collecting neuron). Therefore, we sort the responses by the energy values and select the highest ones according to a predefined rate threshold (in percent). The set of selected cues from a rank-order coding is the percentage of most important or most reliable responses that we are going to use in the block-matching algorithm according to the local light contrast. This is a concrete setup, but we also define this selection procedure in a different way, implementing a multimodal operator, as described in Section IV.

We use a set of well-known sequences to test our model: the Yosemite, the translation tree, and the diverging tree sequence [18]. The results are shown in Fig. 5. To facilitate the comparison of accuracy, the proposed motion-estimation model is compared with a standard block-matching method (which directly matches image luminance instead of neuron responses), as shown in Fig. 5.

In Fig. 5, there are three different plots, each of which has two different groups of columns. The first represents a bioinspired block matching, as can be seen in the legend, in which we use rank-order coding to select the most important responses and apply an adapted block-matching algorithm. The second column shows the results of a standard block-matching algorithm based on sustained neuron responses.

The metric used is the average angular error, which is the average of the angle difference between the computed motion vectors and the ground-truth ones, as defined in the following:

$$AAE = mean\left(\cos^{-1}(\hat{e} \cdot \hat{g})\right) \quad (3)$$

where AAE is the average angular error, $\hat{e}$ is the normalized estimated motion vector, and $\hat{g}$ is the normalized ground-truth vector.

An analysis of the average angular error clearly supports the bioinspired approach. As can be seen in every test, the average angular error increases concomitantly with an increase in the number of responses provided to the block-matching stage. At the top left, diverging tree results follow a stable increasing curve, while the percentage of responses taken (or rate (in percent) of selected cues from rank-order coding) is growing. Therefore, as we increase the number of cell responses provided to the block-matching stage, the system produces higher error. Furthermore, the same progression can be seen in all the tests. We also have to take into account the error difference between the two types of block-matching algorithms, and we see that the bioinspired algorithm based on cell responses always leads to the best results.

In conclusion, our working hypothesis is supported by the results: Neurons that provide the best results, i.e., neurons that fire spikes with higher energy due to their tuning regions with specific spatial and contrast features are the most reliable ones for estimating motion with block matching. Therefore, we use rank-order coding to choose the image areas with higher confidence. This is of critical importance for the subsequent processing layers.

III. Transient Neuron Integration Into the Motion Model

The transient neuron responses provide us with a way to incorporate temporal information into the matching process described in Section II. This alternative consists of integrating the increasing and decreasing (INC and DEC) transient response neurons with the cues given by the sustained neurons for our sparse block-matching algorithm.

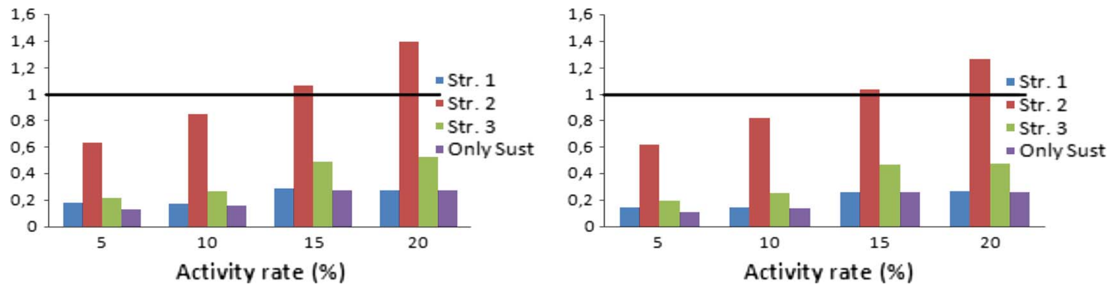


Fig. 6. CPU relative time graphics. The solid line represents the CPU time for the standard block-matching algorithm. This is compared with the CPU time consumed with each of the three strategies presented and the approach that relies only upon the sustained responses, using the Yosemite sequence. On the left, the exploration region used is of 3 × 3, and on the right, it is of 5 × 5. CPU time depends a great deal upon the ratio between the percentage of retinal responses and the resolution, i.e., upon the activity rate (several values of activity rate are explored along the x-axis).

We have implemented the following three different strategies.

1) Strategy 1 preselects dynamic local areas, i.e., local activation of transient response cells preselects areas of interest for motion estimation. The INC and DEC responses are used for the localization (definition) of regions where there is motion. In this case, we use local transient information only for finding these areas, thus providing sustained cell responses from these areas alone to the block-matching algorithm (the pseudocode is detailed in Appendix I-C). This alternative is based on a block-matching scheme that always focuses on areas where there is movement. If it is applied to a static background, there is no estimation provided. The scheme only generates motion estimation if temporal transitions exist in the processed scenario. This alternative significantly reduces the computations and helps to focus attention directly on the moving objects.
2) Strategy 2 is based on the idea of using all the retinal responses directly. In this case, the way to find the best matching region is the SSD, or the search of the region with which the current response neighborhood has the minimum mse, as described in Section II. Using transient and sustained responses, this strategy computes a new error measure to guide the exploration of the block-matching process. The new error measure, inspired by Simoncelli's work [6], [7], is defined in (4). It is not only a normalization; we define a new normalized mse to find the best matching in the region

$$E_n = \frac{S_n}{\sum_{n\in R} S_n + K} + \frac{T_n}{\sum_{n\in R} T_n + K} \tag{4}$$

where E_n is the error value for neuron response n, S_n is the mse for sustained response, T_n is the mse for transient response, R is the region or neighborhood where n belongs, and K is a constant. In addition, we use E_n to choose the right motion estimation in the matching algorithm search. The pseudocode is shown in Appendix I-D.

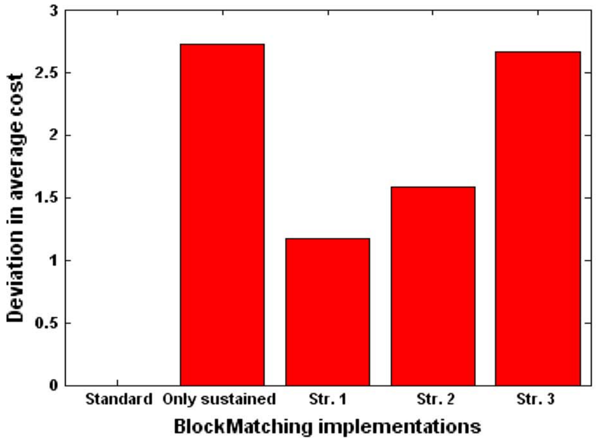
This strategy is based on energy models and represents a preselection of areas with higher energy for sustained and transient responses, thus normalizing their error.
3) Strategy 3 uses transient information only in specific situations. The algorithm uses the error measure defined in (4) in the exploration of the best region matching only when the decision is ambiguous, i.e., when the choice of a region is uncertain because two or more regions have a similar mse. The algorithm uses a threshold to decide whether to use mse or the new error measure (the threshold is set to 1.0). The fully detailed algorithm is shown in Appendix I-E.

The relative CPU times for the different strategies are shown in Fig. 6. The standard block-matching CPU time is defined by the line (1 in relative time) in the graphic. The graphic on the left depicts the relative time with an exploration region of 3 × 3, and on the right, it is of 5 × 5. The results support our hypothesis that our model is more efficient with lower activity rates than the standard block-matching algorithm. Strategies 1 and 3 achieve the best results, reducing CPU time by around 75% and 60%, respectively, in the worst case. On the other hand, strategy 2 needs more computational resources than the standard block matching, even with low activity rates.

In addition to this, we also studied the computational load (L) for the different strategies. These are detailed in (5)–(9), where n is the resolution in pixels for each frame; f_{sust} is the activity rate for sustained cells; f_{trans} is the activity rate for transient cells; $f_{\text{matching ambiguity}}$ is the likelihood that the minimum mse block search finds more than a single block; and A, B, and C are the computational loads for the different processing tasks of each pixel or response (note that $A < B, C$). The activity rates in benchmark sequences are tuned to be around 10%. The operations included in A, B, and C are indicated in Appendix I

$$L_{\text{standard}} = n(A) \tag{5}$$
$$L_{\text{sust}} = f_{\text{sust}}(n)(A) \tag{6}$$
$$L_{str1} = \left(f_{\text{tran}}(n) \cap f_{\text{sust}}(n)\right)(A) \tag{7}$$
$$L_{str2} = \left(f_{\text{tran}}(n) \cup f_{\text{sust}}(n)\right)(B) \tag{8}$$
$$L_{str3} = f_{\text{sust}}(n) \times \left(C + F_{\text{matching ambiguity}}\left(\frac{n}{block_{\text{size}}}B\right)\right). \tag{9}$$

Finally, we used a cost function to compare the different bioinspired block-matching strategies. The error estimation was calculated by the angular error using the three benchmarking



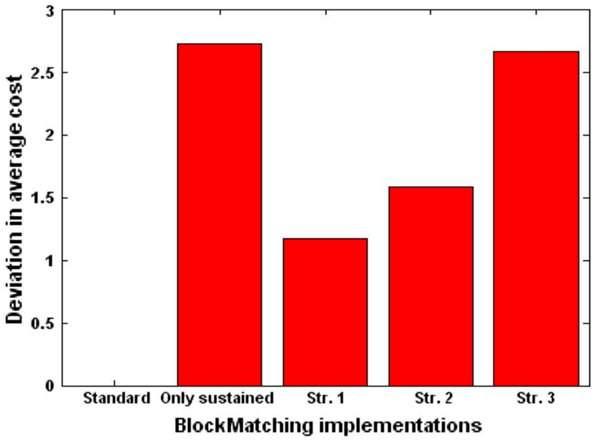


Fig. 7. (Left) Total average cost function when different scheme parameters are used for the various matching alternatives. The purpose is not to achieve an optimum solution but rather to evaluate the configuration and the role of each kind of response. (Right) Stability model with respect to parameter tuning, using the configurations shown in Table II.

TABLE II
COMBINATORY SCANNING OF PARAMETERS. THIS INVOLVED ANALYZING 480 CONFIGURATIONS FOR EACH STRATEGY AND SEQUENCE

Parameter	Values
Inner and Outer deviations (DOG)	1,2; 1,3; 1,5; 1,7; 3,5
Sustained threshold	5, 10, 20
Transient threshold	10, 15, 25
Temporal filter configuration	{-1,1}, {-1,0,1},{-1, -2, 0, 1, 2}
Exploration window	6,10,14
Block size	8, 16

sequences (comparing the obtained results with the known real ground-truth motion fields).

Angular error is not, however, a definitive evaluation estimation. It is important to emphasize that our strategies are sparse block-matching algorithms, and therefore, we need a cost function that selects the strategy with the smallest error and also with a higher number of responses (higher density).

The cost function (10) is defined as the ratio between the average angular error and the density of neurons that are active

$$Cost = \frac{Ang.Error}{Density}. \tag{10}$$

A comparison of the accuracy of the different matching strategies is shown in Fig. 7, in which five columns represent the standard block matching, the block matching using only sustained responses, and the other three strategies, which function according to the responses of the transient sensitive cells. We explored the space of the matching method parameters (modifying DoG sizes, search areas, block sizes, etc.) in order to determine the strategy with the highest accuracy and stability as far as parameter tuning was concerned. The optimization of the parameters involved a combinatory scanning, with the object being not to obtain an optimum solution but rather to evaluate the configurations and alternatives and a way of estimating the role of the different kinds of neuron. The multiparameter combinatory search scans inner and outer standard deviations for sustained responses, temporal filters for transient responses, and block-matching parameters (Table II).

Strategies 1 and 2 turned out to be best, not just on comparing average costs but also on analyzing their stability as far as different scheme parameters were concerned (Fig. 7). Strategies 3 and that consisting of just using sustained cells led to higher average cost and greater instability. On the other hand, high computational requirements were needed for strategy 2. After analyzing the results, we chose the first strategy as being a good tradeoff between accuracy, stability, and computational cost.

IV. ATTENTION MODELS

As demonstrated in Section II, it is possible to use multiscale and rank-order coding for the implementation of a system which is able to select a low percentage of sustained neuron responses to calculate optical flow fields very accurately. In that model, we did not use the transient neuron responses, but the information involved might be useful for attention models.

We define a versatile multimodal attention operator which is able to focus on different features such as motion, colors, textures, and so on. We apply rank-order coding by weighting the responses in the image that tune with a new feature, such as temporal events (transient cues), a specific color, or even a texture, and focus the system computing resources on them, even dynamically. The operator is defined in (11) and (12)

$$C_{\text{sel}} = R(\alpha \cdot S_F) \tag{11}$$

where C_{sel} is the result of the operator, i.e., the list of selected cues from rank-order coding, R is the operator, S_F is the sustained energy value for the frame F, and α is the multimodal factor defined in (11)

$$\alpha = 1 + \frac{\omega_T \cdot T_F}{\sum_{n\in F} T_n + k} + \frac{\omega_C \cdot C_F}{\sum_{n\in F} + k} + \frac{\omega_{Tx} \cdot Tx_F}{\sum_{n\in F} Tx_n + k} + \cdots \tag{12}$$

where T_F, C_F, and Tx_F are the transient, color, and texture energy values of the F frame, respectively, and ω_T, ω_C, and ω_{Tx} are the weights for each feature. The α factor is normalized by the sum of the values of the neuron responses (n) for each feature, and K is a constant (with a value of 1.1) used to avoid a null denominator. The multimodal operator (R) extracts the fixed rate of cues (in percent) from the sorted list. The

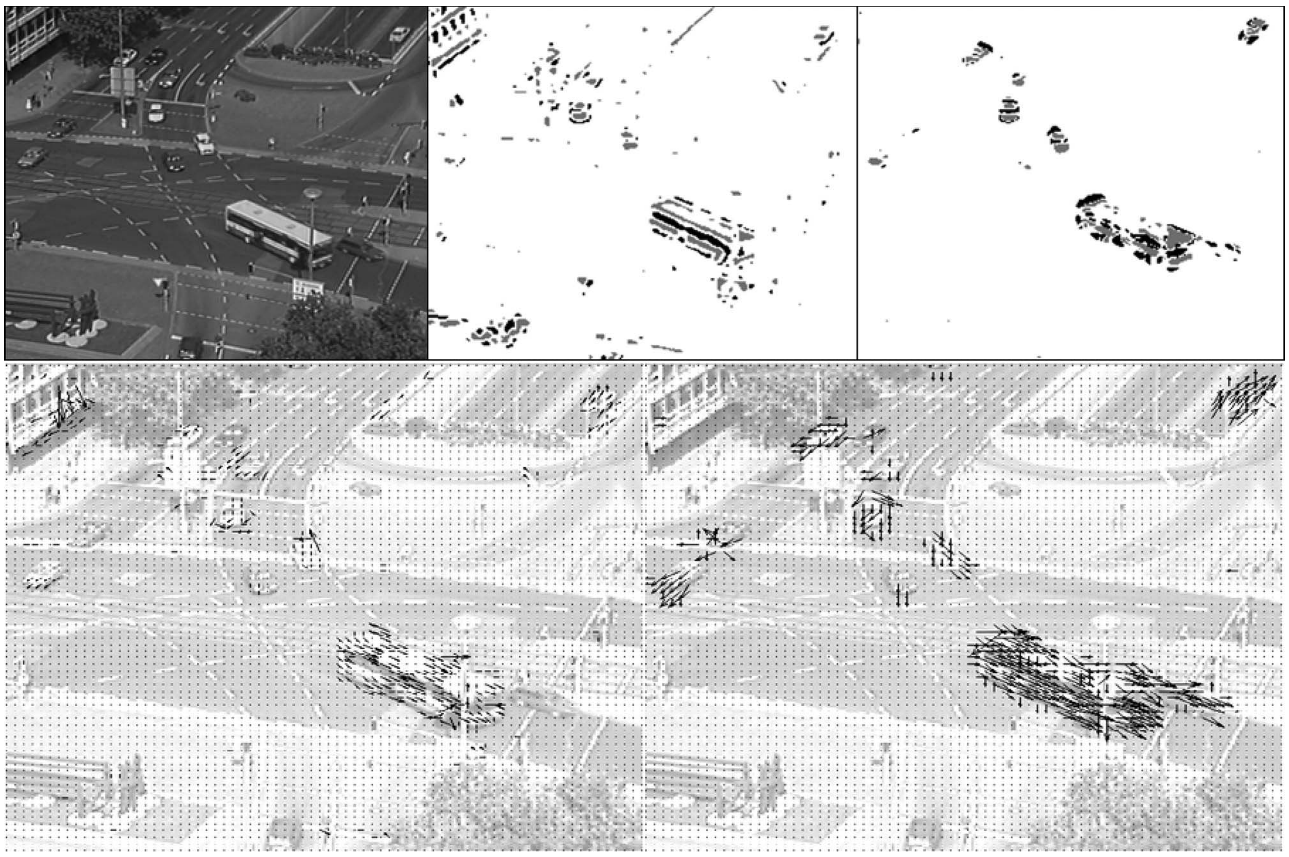


Fig. 8. Comparison between the block-matching algorithm and the block-matching algorithm using the attention operator focusing on motion over traffic sequences [21]. (From left to right) first row—original image, sustained responses and rank-order sustained responses; second row—flow field for the first algorithm and flow field for the second one overlapping the original image.

criterion for sorting depends on the sustained energy and on the multimodal factor α.

Rank-order coding can be used to reduce the number of computations, but if not carefully used, it produces undesirable effects. For instance, if a low-contrast object is moving in a high-contrast static environment, rank-order coding based on local energy rejects the moving object. This is so if the rank order only uses information from sustained neurons. Furthermore, it is important to point out that it is possible to use the multiscale approach, focusing the attention operator on each scale. For example, if we have a small object and a large object in motion, with our motion detection model, even with multiscale, it may be impossible to extract motion for the small object because the receptive fields do not tune its spatial features accurately because of its size. If we use only sustained neuron responses, we cannot improve the estimation, but if we use transient neuron responses, it is possible to extract the small object's motion by weighting its temporal features. In this way, an attention model based on motion will focus on the largest object's motion when they are both moving, but once the largest one becomes static, if the smallest one continues, its motion becomes salient.

An attention module can be implemented by defining a new energy measure, as shown in (11) and (12), where we weight the new features according to the application of our system. For example, if we need to locate objects in motion irrespective of their size, the last example would be a good solution. Nevertheless, if we need to extract the motion or to track a red object or one with a specific texture, we modify the attention operator to set higher weight upon responses that tune red objects or the specific target texture and raise them in the rank-order coding list to the highest status. As shown in Fig. 8, the attention operator focuses on motion, and the algorithm extracts more information for objects that are moving, although the contrast between them and the background is not particularly significant. Furthermore, the number of responses is similar for the algorithms; around 3% of the number of pixels and the estimations in the block-matching algorithm based only on sustained responses are the sparsest. It is assumed that the estimations for a block-matching algorithm that uses rank-order coding are the best because, where there is no motion, it estimates null velocity, and the system focuses on objects in motion, while the basic algorithm estimates different erroneous velocities in these cases. It can be seen in Fig. 8 how the system's response to moving objects is enhanced and the sparse erroneous responses to static objects in the basic model are neatly cleaned up (see the top row in Fig. 8).

An example where the attention operator is focused on the motion and on the red color is shown in Fig. 9, where a red car can be seen driving south. If we apply the standard algorithm, as the contrast with the background is not very significant, the sustained cells do not fire a high response. Therefore, the algorithm does not provide a motion estimation for the red car. On the other hand, if we use the attention operator focusing

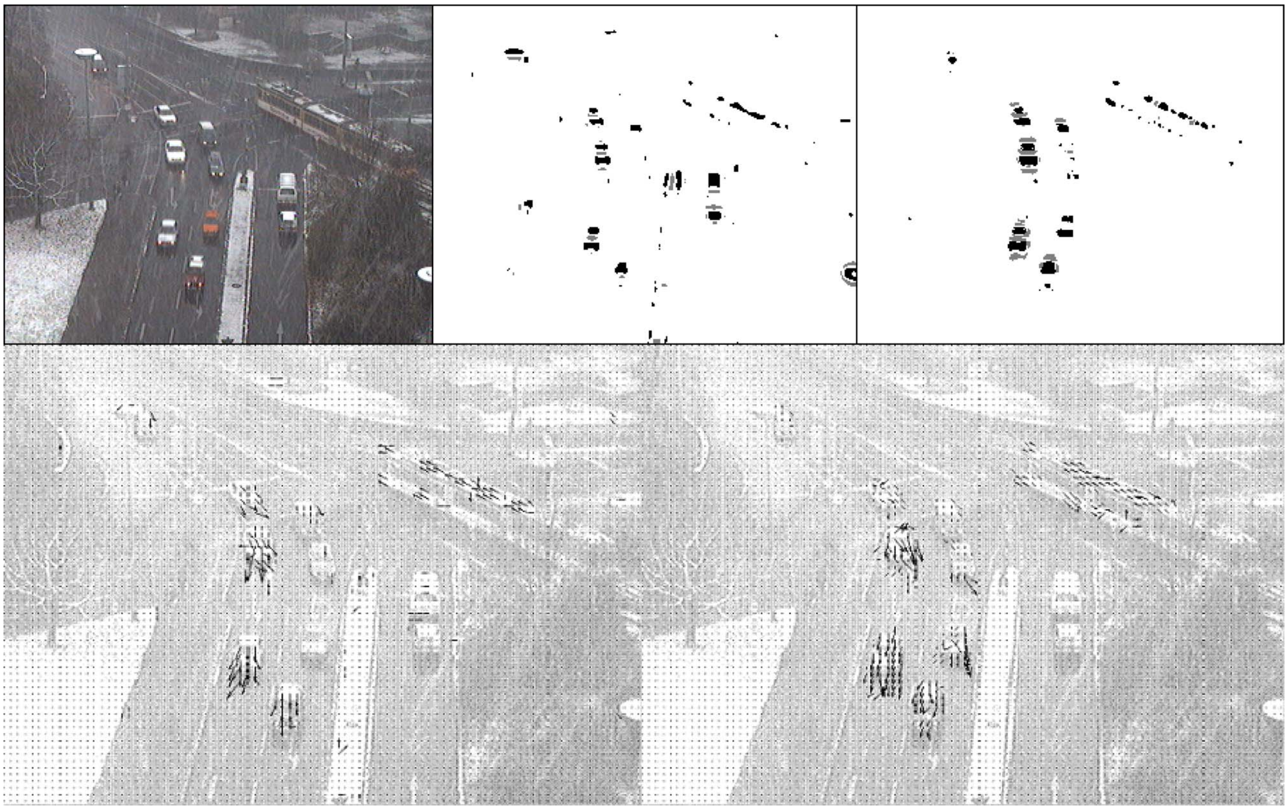


Fig. 9. Comparison between the block-matching algorithm and the block-matching algorithm using the attention operator focusing on color and motion over traffic sequences [21]. (From left to right) first row—original image, sustained responses, and rank-order sustained responses; second row—flow field for the first algorithm and flow field for the second one overlapping the original image. In this latter case, the red car (inside the dotted circle) attracts a larger number of estimations by the attention module.

on the red color, we reinforce the red car cues which emerge at the top of the rank-order coding list. In this way, we obtain the motion estimation for the red car. Furthermore, the operator is also focused on the motion, as shown in Fig. 8. Fig. 10 shows the same example, now focusing on the orange color. In Fig. 9, the focus was on the red car, but now, in Fig. 10, the orange trams provide the focus. The block-matching algorithm provides a sparse motion estimation for the tram in the upper left-hand corner, but it does not give any estimation for the other tram. Using the attention operator focused on the orange color, we achieve more responses for the two trams, and the algorithm provides a slightly denser motion estimation for both of them than does the former algorithm.

The difference between the two alternatives in each case is significant, showing more responses for the focus objects (red car in Fig. 9 and orange trams in Fig. 10) and estimating the motion.

Moreover, we can also define the multimodal attention operator dynamically. We are currently exploring schemes in which we first fix the operator to extract only objects in motion, and then, we modify the operator to extract objects of some colors and possibly objects that match a specific texture.

V. Conclusion

We have designed and implemented a bioinspired model based on artificial retinas for the detection of spatiotemporal features.

We have demonstrated that by choosing responses in an intelligent way, we have been able to improve the accuracy of our model significantly. We have shown that regions with higher local contrast lead to more accurate estimations. Furthermore, the average angular error decreases to around 32%, and the average improvement in accuracy is around 16% when preselecting the proper responses (before computing motion by a block-matching model). We have also implemented a new motion-estimation bioinspired model based on the standard block-matching algorithm integrating concepts such as multiscale and rank-order coding. The selection of the responses produces low-density saliency maps (about 11%–12% of the number of pixels in standard images for each kind of neuron), which is of interest for applications with strict bandwidth constraints.

We have designed a more stable improved block-matching algorithm which also has a lower cost by integrating transient neuron responses. We have implemented different strategies based on energy models and defined a cost function. We then selected the most stable and lowest cost strategy which preselects dynamic local areas for the matching processing. The cost-function tests show similar errors to standard block-matching algorithms and low deviations.

Finally, we define a versatile multimodal attention operator to extract other potential features by modifying the rank-order coding computation.

Our models allow features to be selected according to attention processes by using rank-order coding, and thus, we can

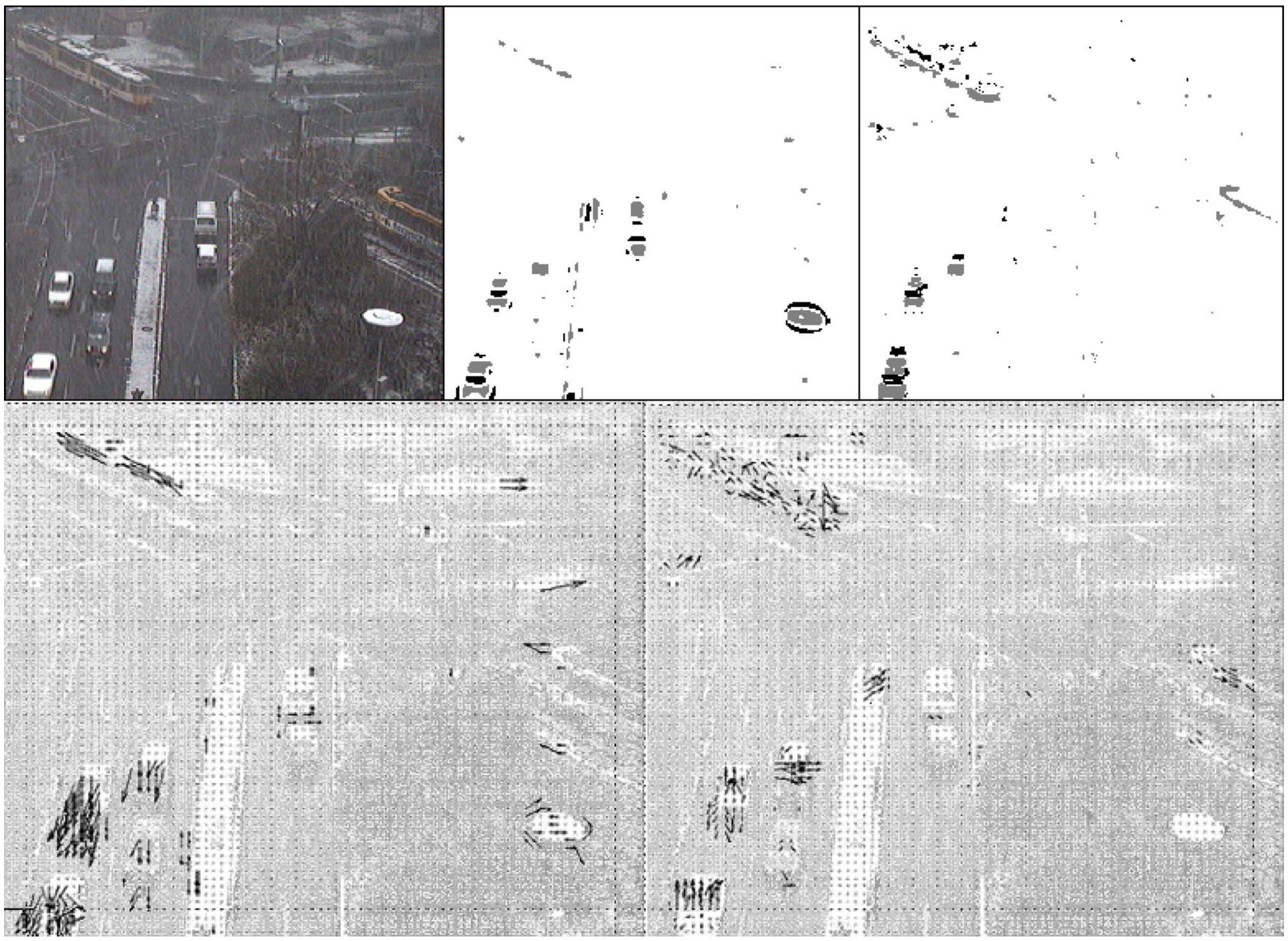


Fig. 10. Comparison between the block-matching algorithm and the block-matching algorithm using the attention operator focusing on color and motion in traffic sequences [21]. (From left to right) first row—original image, sustained responses, and rank-order sustained responses using motion and orange color to drive the attention operator; second row—flow field for the first algorithm and flow field for the second one overlapping the original image. In this latter case, the orange trams (inside the dotted ellipses) provide the focus of attention.

choose the image features that lead to more accurate motion estimations. This is highly relevant for efficient computation because only a few pixels of the image need to be processed. In tasks such as object tracking, for instance, we bias the rank order of the sustained neurons according to the transient neuron information by using a small number of high-confidence features.

This computing scheme is designed to be used with applications involving embedded processors, such as video surveillance [22]. For these devices, computing resources are very limited, but using the proposed method, we can produce motion estimations without losing the relevant features. We also plan to optimize the parameter search and integrate our model into a more general framework to study different vision schemes or even into real-time processing technologies.

APPENDIX I

A. Block-Matching Pseudocode

The input parameters are as follows: F_next is the next frame in the sequence, F_current is the frame for which we calculate the motion estimation (ME), Max_disp defines a square exploration window of (Max_disp · 2) × (Max_disp · 2) size, and B_size is the block size.

```
ME = BlockMatching(F_next, F_current, Max_disp, B_size)

ME = Initialise_ME();
B_half = B_size/2-1;

for i=1+B_half:ROWS-B_half
  for j=1+B_half:COLUMNS-B_half{

    %Extracting current block
    B_current = F_current(i-B_half:i+B_half, …
                                        … j-B_half:j+B_half);
    MSE_min = INFINITE;

    %Exploration window full search
    for u=-Max_disp:Max_disp
      %Checking image limits
      if(i-B_half+u >=1 && i+B_half+u<=ROWS)
        for v=-Max_disp:Max_disp
          %Checking image limits
           if(j-B_half+v >=1 && j+B_half+v<=COLUMNS){
%Extracting next block
  B_next = F_next(i-B_half+u:i+B_half+u, j-B_half+v:j+B_half+v);

  %Calculating MSE
  error = MSE(B_next, B_current);

 %Updating minimum MSE motion estimation
  if(error<MSE_min){
    me_x=u; me_y=v;
    MSE_min = error;
   } else
     %Undefined ME
     if(error==MSE_min)
     {
        me_x=NaN; me_y=NaN;
     }
  }% if

      ME(i,j)= {me_x, me_y};
} % for
```

B. Block-Matching Multiscale Pseudocode

The input parameters are as follows: F_next is the next frame in the sequence, F_current is the frame for which we calculate the motion estimation (ME), Max_disp defines a squared exploration window of (Max_disp · 2) × (Max_disp · 2) size, B_size is the block size, Region is a structure that defines the search region for the new motion estimation based on the previous estimation, and ME_old is the previous motion estimation oversampled.

The minimum mse motion estimation is updated in two cases: The new mse is significantly lower than the previous one (the mse of the new ME is DIST_THRES lower than the previous stored mse); the new mse and the Euclidean distance to estimation $\{0, 0\}$ are lower than the previous ones. The method is based on the distance because the lowest estimations are the most reliable ones in a multiscale approach, since the firing frequency is high.

```
const DIST_THRES = 1;
%For an exploration region of 3 pixels
Region.U = {1 1 1 0 0 0 -1 -1 -1};
Region.V = {1 0 -1 1 0 -1 1 0 -1};

%For an exploration region of 5 pixels
Region.U = {2 2 2 2 2 1 1 1 1 1 0 0 0 0 0 -1 -1 -1 -1 -1 -2 -2 -2 -2 -2};
Region.V = {2 1 0 -1 -2 2 1 0 -1 -2 2 1 0 -1 -2 2 1 0 -1 -2 2 1 0 -1 -2};

ME = BM (F_next, F_current, Max_disp, B_size, Region, ME_old)

ME = Initialize_motion_estimation();

for i=1+B_half:ROWS-B_half
  for j=1+B_half:COLUMNS-B_half{

      %Extracting current block
      B_current = F_current(i-B_half:i+B_half, j-B_half:j+B_half);

      if(! isempty(B_current)){
         MSE_min = INFINITE;

         %Calculating possible estimations based on previous ones
         disp(1) = Region.U + ME_old(i,j){1};
         disp(2) = Region.V + ME_old(i,j){2};
         me_x=0; me_y =0;

         %Exploration window full search
         for u=1:size(disp(1))
             if((i-B_half+disp(1,u) >=1) & …
                                         … (i+ B_half +disp(1,u)<= ROWS))
                for v=1:size(disp(2))
                   if((j- B_half + disp(2,v) >=1) & …
                                         …  (j+ B_half + disp(2,v)<= COLUMNS)){

%Extracting next block
B_next = F_next(i-B_size/2+disp(1,u):i+B_size/2+disp(1,u),
                   j-B_size/2+disp(2,v):j+B_size/2+disp(2,v));

%Calculating MSE
error = sum(sum(B_current - B_next));

%Updating minimum MSE using a threshold
if(error < MSE_min + DIST_THRES){
   if (abs(error-MSE_min)<=DIST_THRES){
      dist_min=me_x^2 + me_y^2;
      dist_new=disp(1,u)^2 + disp(2,v)^2;
      if(dist_new < dist_min){
         me_x=disp(1,u); me_y=disp(2,v);
         error=MSE_min;
      }
   }else{
      me_x=disp(1,u); me_y=disp(2,v);
      MSE_min= error;
   }
}

} %if((j-B_size/2 …

ME(i,j)= {me_x, me_y};

}%if(! Isempty …

}%for
```

C. Block-Matching Strategy-1 Pseudocode

The input parameters are as follows: FS_next is the next frame in the sequence of sustained responses, FS_current is the sustained response frame for which we calculate the motion estimation (ME), FT_next is the next frame in the sequence of transient responses, FT_current is the transient response frame for which we calculate the motion estimation (ME), Max_disp defines a squared exploration window of (Max_disp $\cdot$ 2) $\times$ (Max_disp $\cdot$ 2) size, and B_size is the block size. The highlighted code represents the computational load A in (4)–(6).

```
ME = BM(FS_next, FS_current, FT_next, FT_current, …
                                          … Max_disp, B_size)

ME = Initialise_motion_estimation();
B_half = B_size/2;

for i=1+B_half:ROWS-B_half
  for j=1+B_half:COLUMNS-B_half{
      %Extracting current transient & sustained block
      BT_current = FT_current(i-B_half:i+B_half, j-B_half:j+B_half);
      BS_current = FS_current(i-B_half:i+B_half, j-B_half:j+B_half);

      if(!isempty(BT_current) & !isempty(BS_current)){

        MSE_min = INFINITE;
        %Exploration window full search
        for u=-Max_disp:Max_disp
           %Checking image limits
           if(i-B_half+u >=1 && i+B_half+u<=ROWS)
              for v=-Max_disp:Max_disp
                 %Checking image limits
                 if(j-B_half+v >=1 && j+B_half+v<=COLUMNS){
                    %Calculating MSE
                    error = MSE(BS_next, BS_current);

%Extracting next sustained block
BS_next = FS_next(r0+u:r0+B_size+u-1, …
                                   … c0+v:c0+B_size+v-1);
%Calculating MSE

if(error<MSE_min){
   me_x=u; me_y=v;
    MSE_min = error;
}
else{
    %Undefined ME
    if(error==MSE_min)
    {
       me_x=NaN; me_y=NaN;
    }
}

}%if(v+c0 ...

ME(i,j)= {me_x, me_y};

}%if(!isempty ...

}%for
```

D. Block-Matching Strategy-2 Pseudocode

The input parameters are as follows: FS_next is the next frame in the sequence of sustained responses, FS_current is the sustained response frame for which we calculate the motion estimation (ME), FT_next is the next frame in the sequence of transient responses, FT_current is the transient response frame for which we calculate the motion estimation (ME), Max_disp defines a squared exploration window of (Max_disp $\cdot$ 2) $\times$ (Max_disp $\cdot$ 2) size, and B_size is the block size. The highlighted code represents the computational load B in (7) and (8).

```
const K = 1.1;

ME = BM(FS_next, FS_current, FT_current, FT_next, …
                                   …Max_disp, B_size)

ME = Initialise_motion_estimation();
B_half = B_size/2-1;

for i=1+B_half:ROWS-B_half
  for j=1+B_half:COLUMNS-B_half{

     %Calculating current transient & sustained blocks
     BT_current = FT_current(i-B_half:i+B_half, j-B_half:j+B_half);
     BS_current = FS_current(i-B_half:i+B_half, j-B_half:j+B_half);

    if(!isempty(BT_current) || !isempty(BS_current)){
     MSE_min = INFINITE;
     for u=-Max_disp:Max_disp
        %Checking image limits
        if(i-B_half+u >=1 && i+B_half+u<=ROWS)
           for v=-Max_disp:Max_disp
              %Checking image limits
              if(j-B_half+v >=1 && j+B_half+v<=COLUMNS){
                 %Calculating next transient & sustained blocks
                 BS_next = FS_next(i-B_half+u:i+B_half+u, …
                                        … j-B_half+v:j+B_half+v);
                 BT_next = FT_next(i-B_half+u:i+B_half+u, …
                                        … j-B_half+v:j+B_half+v);
                 %Calculating MSE
                 errorS(u+Max_disp+1, v+Max_disp+1) =
                                        … MSE(BS_next, BS_current);
                 errorT(u+Max_disp+1, v+Max_disp+1) =
                                        … MSE(BT_next, BT_current);
                 denS += errorS(u+Max_disp+1, v+Max_disp+1);
                 denT += errorT(u+Max_disp+1, v+Max_disp+1);
           }

%For subtained cells
errorS_norm=errorS/(denS+K);

%For transient cells
errorT_norm=errorT/(denT+K);

%Normalised error
errorN=errorT_norm + errorS_norm;


for u=-Max_disp:Max_disp
   if(i-B_half+u >=1 && i+B_half+u<=ROWS)
      for v=-Max_disp:Max_disp
         if(j-B_half+v >=1 && j+B_half+v<=COLUMNS){
            error = errorN(u+Max_disp+1, v+Max_disp+1);
            %Updating minimum MSE
            if(errorN<MSE_min){
               me_x=u; me_y=v;
               MSE_min = errorN;
            }
            else{
               %Undefined ME
               if(errorN == MSE_min)
               {
                  me_x=NaN; me_y=NaN;
               }
            }
         }

   ME(i,j)= {me_x, me_y};
 } %if (!isempty( ...
}
```

E. Block-Matching Strategy-3 Pseudocode

The input parameters are as follows: FS_next is the next frame in the sequence of sustained responses, FS_current is the sustained response frame for which we calculate the motion estimation (ME), FT_next is the next frame in the sequence of transient responses, FT_current is the transient response frame for which we calculate the motion estimation (ME), Max_disp defines a squared exploration window of (Max_disp $\cdot$ 2) $\times$ (Max_disp $\cdot$ 2) size, and B_size is the block size. The highlighted code in the left column represents the computational load B in (7) and (8), and in the right column, it represents the computational load C in (8).

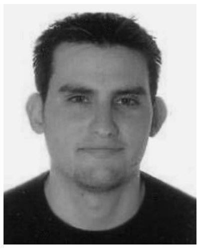
```
const K = 1.1; const AMB_THRES= 1.0;
ME = BM(FS_next, FS_current, FT_current, FT_next, …
                                  …Max_disp, B_size)
ME = Initialise_motion_estimation();
B_half = B_size/2-1;

for i=1+B_half:ROWS-B_half
  for j=1+B_half:COLUMNS-B_half{

     %Extracting current transient & sustained blocks
    BS_current = FS_current(i-B_half:i+B_half, j-B_half:j+B_half);

    if(!isempty(BS_current)){
     BT_current = FT_current(i-B_half:i+B_half, j-B_half:j+B_half);
     MSE_min = INFINITE;
     amb = 0;

     for u=-Max_disp:Max_disp
        if(i-B_half+u >=1 && i+B_half+u<=ROWS)
           for v=-Max_disp:Max_disp
             if(j-B_half+v>=1 && j+B_half+v<=COLUMNS){
              BS_next = FS_next(i-B_half+u:i+B_half+u, …
                                      … j-B_half+v:j+B_half+v);
              BT_next = FT_next(i-B_half+u:i+B_half+u, …
                                      … j-B_half+v:j+B_half+v);

              %Generating MSE
              errorS(u+Max_disp+1, v+Max_disp+1) = …
                                      … MSE(BS_next, BS_current);
             errorT(u+Max_disp+1, v+Max_disp+1) = …
                                      … MSE(BT_next, BT_current);
              if(errorS(u+Max_disp+1,v+Max_disp+1)< MSE_min …
                                      … & amb != 1){
                me_x=u; me_y=v;
                MSE_min = errorS(u+Max_disp+1,v+Max_disp+1);
           }
             else
                %Ambiguous situation
                if(abs(errorS(u+Max_disp+1,v+Max_disp+1)- …
                                      … MSE_min)<=AMB_THRES)
                      amb=1;
            }

if(amb == 1){

   %For subtained cells
   denS = sum(sum(errorS))+K;
   errorS_norm=errorS/denS;

   %For transient cells
   denT = sum(sum(errorT))+K;
   errorT_norm=errorT/denT;

   %Normalised error
   errorN=errorT_norm + errorS_norm;


   for u=-Max_disp:Max_disp
      if(i-B_half+u >=1 && i+B_half+u<=ROWS)
        for v=-Max_disp:Max_disp
           if(j-B_half+v >=1 && j+B_half+v<=COLUMNS){
              error = errorN(u+Max_disp+1, …
                                … v+Max_disp+1);
               %Updating minimum MSE
               if(errorN<MSE_min){
                 me_x=u; me_y=v;
                  MSE_min = errorN;
               }
               else{
                  %Undefined ME
                  if(errorN == MSE_min)
                  {
                   me_x=NaN; me_y=NaN;
                  }
               }

           }

        }
   ME(i,j)= {me_x, me_y};
  }%if(!isempty(...
}
```

ACKNOWLEDGMENT

The authors would like to thank A. L. Tate for improving the language of this paper.

REFERENCES

[1] J. T. Martin, *An Introduction to the Visual System*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[2] K. Nakayama, "Biological image motion processing: A review," *Vis. Res.*, vol. 25, no. 5, pp. 625–660, Nov. 1985.
[3] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interactions and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, pp. 106–154, 1962.
[4] E. H. Adelson and J. R. Bergen, "The extraction of spatio-temporal energy in human and machine vision," in *Proc. IEEE Workshop Motion: Representation Anal.*, 1986, pp. 151–155.
[5] D. J. Heeger, "Model for the extraction of image flow," *J. Opt. Soc. Amer. A, Opt. Image Sci.*, vol. 4, no. 8, pp. 1455–1471, Aug. 1987.
[6] E. P. Simoncelli and D. J. Heeger, "A model of neuronal responses in visual area MT," *Vis. Res.*, vol. 38, no. 5, pp. 743–761, Mar. 1998.
[7] E. P. Simoncelli, "Distributed analysis and representation of visual motion," Ph.D. dissertation, MIT, Cambridge, MA, 1993.
[8] S. Mota, E. Ros, J. Díaz, E. M. Ortigosa, and A. Prieto, "Motion-driven segmentation by competitive neural processing," *Neural Process. Lett.*, vol. 22, no. 2, pp. 125–147, Oct. 2005.
[9] E. H. Adelson and P. J. Burt, "The Laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. COM-31, no. 4, pp. 532–540, Apr. 1983.
[10] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Eng.*, vol. 29, no. 6, pp. 33–41, Nov./Dec. 1984.
[11] V. Willert, J. Eggert, J. Adamy, and E. Körner, "Non-Gaussian velocity distributions integrated over space, time, and scales," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 3, pp. 482–493, Jun. 2006.
[12] L. Perrinet, M. Samuelides, and S. Thorpe, "Coding static natural images using spiking event times: Do neurons cooperate?"*IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1164–1175, Sep. 2004.
[13] K. Boahen, "A retinomorphic chip with parallel pathways: Encoding ON, OFF, INCREASING, and DECREASING visual signals," *Analog Integr. Circuits Signal Process.*, vol. 30, no. 2, pp. 121–135, 2002.
[14] K. A. Zaghloul and K. A. Boahen, "Optic nerve signals in a neuromorphic chip I: Outer and inner retina models," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 657–666, Apr. 2004.
[15] E. Culurciello, R. Etienne-Cummings, and K. Boahen, "A biomorphic digital image sensor," *IEEE J. Solid-State Circuits*, vol. 38, no. 2, pp. 281–294, Feb. 2003.
[16] S. Granados, R. Rodríguez, E. Ros, and J. Díaz, "Visual processing platform based on artificial retinas," in *Proc. IWANN*, 2007, pp. 506–513.
[17] K. A. Zaghloul and K. Boahen, "Optic nerve signals in a neuromorphic chip II: Testing and results," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 667–675, Apr. 2004.
[18] *Optical Flow Synthetic Sequences With Ground-Truth Information*. [Online]. Available: ftp://ftp.vislist.com/SHAREWARE/CODE/OPTICAL-FLOW
[19] J. L. Barron, D. J. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43–77, Feb. 1994.
[20] E. Trucco, T. Tommasini, and V. Roberto, "Near-recursive optical flow from weighted image differences," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 35, no. 1, pp. 124–129, Feb. 2005.
[21] H. Nagel, *Institut für Algorithmen und Kognitive Systeme: Ettlinger-Tor*. [Online]. Available: ftp://ftp.ira.uka.de/pub/vid-text/image_sequences/dt/sequence.mpg
[22] L. Snidaro, R. Niu, G. L. Foresti, and P. K. Varshney, "Quality-based fusion of multiple video sensors for video surveillance," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 4, pp. 1044–1051, Aug. 2007.



Francisco Barranco received the M.S. degree in computer science from the University of Granada, Granada, Spain, in 2007.

He is with the Department of Computer Architecture and Technology, University of Granada. He is currently participating in an EU project related to adaptive learning mechanisms and conventional control. His main research interests include image processing architectures and embedded systems based on reconfigurable devices, real-time machine vision, general-purpose graphical programming devices, biologically processing schemes, and spiking neurons.

Javier Díaz received the M.S. degree in electronics engineering and the Ph.D. degree in electronics from the University of Granada, Granada, Spain, in 2002 and 2006, respectively.

He is currently an Assistant Professor with the Department of Computer Architecture and Technology, University of Granada. His main research interests include cognitive vision systems, high-performance image processing architectures, and embedded systems based on reconfigurable devices. He is also interested in spiking neurons, biomedical devices, and robotics.

Eduardo Ros received the Ph.D. degree from the University of Granada, Granada, Spain, in 1997.

He is currently an Associate Professor with the Department of Computer Architecture and Technology, University of Granada, where he is currently a Researcher for two European projects related to bioinspired processing schemes. His research interests include simulation of biologically plausible processing schemes, hardware implementation of digital circuits for real-time processing in embedded systems, and high-performance computer vision.

Begoña del Pino received the Ph.D. degree in computer science from the University of Granada, Granada, Spain, in 1999.

She is currently an Associate Professor with the Department of Architecture and Computer Technology, University of Granada. She has participated in different EU projects in the fields of visual rehabilitation, bioinspired processing schemes based on spiking neural networks, and real-time computer vision. Her main research interests include implementation of systems for visual rehabilitation, bioinspired processing systems, and reconfigurable hardware codesign for computer vision on-chip.