A Fast Biologically Inspired Algorithm for Recurrent Motion Estimation

Pierre Bayerl and Heiko Neumann, *Fellow*, *IEEE Computer Society*

Abstract—We have previously developed a neurodynamical model of motion segregation in cortical visual area V1 and MT of the dorsal stream. The model explains how motion ambiguities caused by the motion aperture problem can be solved for coherently moving objects of arbitrary size by means of cortical mechanisms. The major bottleneck in the development of a reliable biologically inspired technical system with real-time motion analysis capabilities based on this neural model is the amount of memory necessary for the representation of neural activation in velocity space. We propose a sparse coding framework for neural motion activity patterns and suggest a means by which initial activities are detected efficiently. We realize neural mechanisms such as shunting inhibition and feedback modulation in the sparse framework to implement an efficient algorithmic version of our neural model of cortical motion segregation. We demonstrate that the algorithm behaves similarly to the original neural model and is able to extract image motion from real world image sequences. Our investigation transfers a neuroscience model of cortical motion computation to achieve technologically demanding constraints such as real-time performance and hardware implementation. In addition, the proposed biologically inspired algorithm provides a tool for modeling investigations to achieve acceptable simulation time.

Index Terms—Motion estimation, computational models of vision, recurrent information processing, motion aperture problem, algorithms.

1 Introduction

Image velocity or image motion describes the gradual change of images in the presence of moving objects or a moving observer. Technical applications, such as robotic applications or human-computer interfaces require fast solutions to generate accurate estimations of image motion. Fast approaches often fail to disambiguate motion [1] and accurate solutions often do not run in real time (see [2] for an overview of different approaches). In particular, when dense motion representation has to be estimated from ambiguous motion configurations in the presence of only few trackable features, many approaches fail to generate fast and accurate results. Also, prior knowledge about what is expected to happen in an image sequence, e.g., based on computations from previous time steps or by top-down attention should easily be included in a visual motion estimation scheme. Biological systems share such properties, that instantaneous measurements are combined with recurrent information from previous time steps [3], [4], but with high-computational costs mostly caused by huge memory requirements or complex numerical computations. Similar arguments concerning the computational costs and the integration of prior knowledge also apply for Bayesian frameworks [5], [6]. Moreover, it is mandatory for the accuracy of extracted image motion to combine observations from different image locations to overcome ambiguities which inherently cannot be solved by purely local information (compare Fig. 1). Global integration is critical since it makes it impossible to distinguish between differently moving objects. Thus, some approaches use diffusion-schemes to propagate information in space and time [7], [8]. To segregate image motion from different regions with different motions, local discontinuities in the flow field can be utilized to steer the diffusion scheme [9]. Similar concepts of integration and segregation were introduced in [10] in a neural model of dense motion estimation and segregation. This model also shares the advantage of biological systems and Bayesian approaches to be able to easily include additional information or prior knowledge. In order to make that model suitable for technical applications, we present here a biologically inspired algorithm, which is based on the previously presented neural model, but with significantly reduced requirements to computational resources and near real-time capabilities.

1.1.1 Neural Model of Recurrent Motion Segregation

Here, we give a short overview of the neural model on which the proposed algorithm is based and introduce some terms related to neural modeling. Information concerning the function of **neural mechanisms** in the neural model are given later with detailed explanations as **algorithmic operations** (Section 2).

Visual information in the primate cortex is processed in two different streams, the ventral and the dorsal stream which mainly process form or motion information, respectively [11]. Furthermore, the cortex can be divided into different functional regions denoted as **cortical areas** with different functional properties. Each area, in turn, contains layers of neurons building a neural network with feedforward, lateral, and feedback connections to other layers within the cortical area and to other cortical areas [12]. Relevant for the neural model are areas V1 and MT along the dorsal stream which can be identified to play an important role for early visual motion processing [13], [14], [15].

• *The authors are with the University of Ulm, Department of Neural Information Processing, Oberer Eselsberg, D-89069 Ulm, Germany. E-mail: {pierre.bayerl, heiko.neumann}@uni-ulm.de.*

Manuscript received 29 Sept. 2006; revised 4 Apr. 2006; accepted 5 June 2006; published online 13 Dec. 2006.
Recommended for acceptance by M. Srinivasan.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0526-0905.

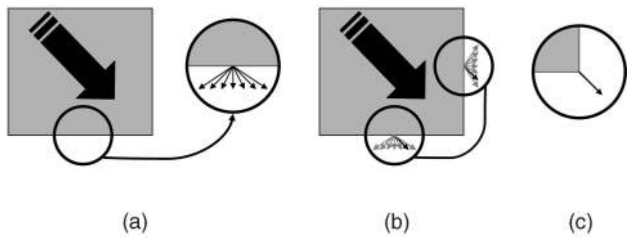


Fig. 1. Illustration of motion ambiguities showing that local information does not necessarily reveal the true object motion. The example shows a square moving to the lower right. The true object motion is indicated by the outlined arrow, detectable local motion cues are indicated by small arrows. (a) Local motion information at elongated contrasts contains inherent ambiguities which makes it impossible to identify the true object motion. (b) The combination of different ambiguous motion cues by, e.g., an intersection can reveal the true motion. (c) At locations with multiple contrast orientations such as corners the detected motion is already unambiguous.

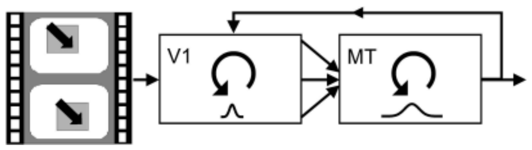


Fig. 2. Sketch of the neural model that serves as a framework to derive the fast algorithm. The model is divided into two modules which refer to visual cortical areas, namely, V1 and MT. Each of the modules is described by mechanisms of feedforward integration (bold arrows between module boxes), lateral normalization (circular arrows within module boxes), and feedback modulation (outlined arrow). The main difference between both modules is the spatial scale of the operations: A fine scale in module V1 and a coarse scale in module MT. As a consequence, the second module combines adjacent motion cues from the first module to solve ambiguities in a small neighborhood. Feedback modulation from the second module, in turn, emphasizes or selects individual motion cues from a set of ambiguous cues in the first module.

The major contributions of the previously presented neural model [10] are 1) the solution of the motion aperture problem and 2) the separation of areas of different motion. The motion **aperture problem** reflects the fact that motion can locally be measured only orthogonal to an extended contrast, the so-called normal flow [16] (compare Fig. 1a). The model solves this problem by propagating information along elongated contrasts and, thus, by combining motion cues over large areas. Moreover, the neural model clusters similar spatially connected velocities and, thus, **segregates regions of different motion** from each other. The model represents velocity locally by a full exhaustive coding and sampling of velocity space with **likelihoods** (also referred to as **neural activity**) for each represented velocity at each location. In neural terms, this representation of motion as likelihoods is called **population code**, where each individual likelihood corresponds to the activity of one individual model cell.[1] In the model, the represented velocity for one likelihood is called the preferred velocity of the corresponding cell, which, in neural terms, is coupled to the **tuning of the cell**. The spatial **receptive field** of a cell describes the image region which directly influences the activity of a cell. All activities of cells encoding information at one specific image location (sharing the same spatial receptive field) is called **local population code** and determines the estimated velocity at this location. Moreover, **inhibition** of neural activity plays an important role for the neural model and algorithm which means that the likelihoods representing neural activity are reduced by some amount. **Divisive inhibition**, in contrast to subtractive inhibition, means that the reduction of likelihoods is realized by a division and is often used to normalize activities [3], [18].

In a nutshell, the overall approach is that individual cells collect votes (or likelihoods) for the velocity they represent and that information from preceding time steps and from a spatial neighborhood are combined to generate the final motion estimation at each location. The model is divided into two modules denoted as *module V1* and *module MT*, according to the cortical areas V1 and MT of the visual system in primates (see [19] for an introductory overview and, e.g., [12] for a more detailed information). Each module performs the same basic operations, module V1 on a fine scale and model MT on a coarse scale. The operations consist of 1) integration, 2) normalization, and 3) feedback modulation of neural activity (sketched in Fig. 2). The basic idea is that context information from the coarse representation is utilized to solve ambiguities on the fine resolution. These improved estimations on the fine scale again are recombined in the coarse representation, which, in turn, is used to further disambiguate the representation on the fine scale in a recursive manner. Details about the individual mechanisms employed by the neural model are given in algorithmic terms later in Section 2.3. The input activities for the neural network represent the similarity of the structure at different locations at different time steps according to the corresponding velocity represented by a cell. This similarity is computed by the scalar product of bandpass filtered image regions (compare elaborated Reichardt detectors; [20]).

1.1.2 Limitations and Complexity of the Computational Implementation of the Neural Model of Motion Segregation

The computational simulation of the neural model leads to the following limitations making the approach unsuitable for many technical applications. 1) **Limited range of detectable velocities:** The explicit sampling of the velocity space limits the maximum speed to be represented and detected. 2) **Memory requirement:** The explicit representation of image motion leads to a huge amount of memory needed to represent the neural activity. The more velocities the model can detect, the more memory is required. Assuming an image sequence with 320×200 pixels 15×15 velocity tuned cells present at each location and single-precision floating point values (4 byte) to depict neural activity, the required memory to represent visual motion at a specific time step is approximately 56 MB. 3) **Runtime complexity:** As a direct consequence of 1) and 2), the runtime is also directly dependent on the maximum speed which can be represented by the model. In particular, the runtime complexity (and the required memory) grows with the square of the maximum speed to be detected.

1. In computational neuroscience approaches, formal neuron models are investigated at different levels, ranging from a detailed representation with the cell body, the structure of the dendritic tree (compartments), and the axon of individual cells to a coarse representation of a unit representing a cell or a cluster of cells without detailing the individual components [17]. In this case, a model cells unit is considered as single unitary compartment that is described by a membrane potential with changes as a function of the current input and the state other cells connected to the target cells. Such a single compartment model is utilized in [10].

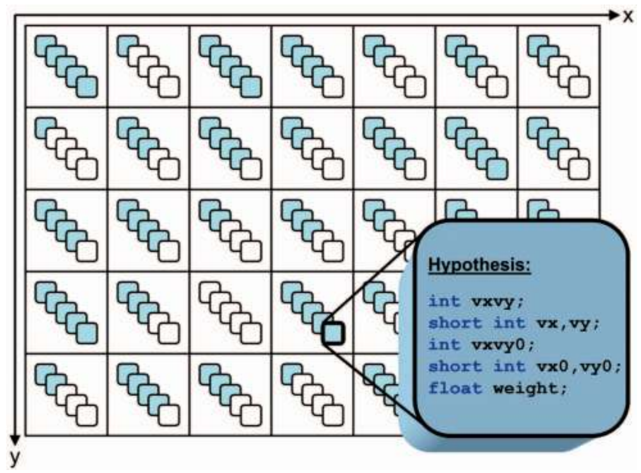


Fig. 3. Sparse motion representation. For each pixel, a maximum number of h_{max} hypotheses can be stored (here $h_{max} = 5$). Unused slots are left blank.

2 Biologically Inspired Algorithm of Recurrent Motion Estimation and Segregation

To overcome the drawbacks of the computational realization of the neural model, we introduce an algorithmic representation of sparse motion hypotheses to reduce the required memory. Then, we develop algorithmic versions of the neural mechanisms which work on the new representation. Moreover, the initial motion estimation is accelerated by introducing a simple similarity measure and a computational scheme to efficiently extract relevant motion correspondences.

2.1 Motion Representation and Data Access

Motion is represented by sets of hypotheses for certain velocities at each image location. A hypothesis thus contains data about its image location (x, y), an integer valued velocity vector (v_x, v_y), and a real valued weight w representing a likelihood for this hypothesis in relation to other hypotheses at the same location. The weight can also be interpreted as confidence value, belief, or as neural activity.[2] The spatial location is implicitly encoded by the position in the array where the hypotheses are stored. The array dimensions are arranged as in the input images (see Fig. 3). We constrain the amount of velocity hypotheses at each image location to a maximum of h_{max} hypotheses.

With each hypothesis, we further store additional information for data access and the preservation of information: 1) A combined velocity value $v_xv_y := v_x + v_y \cdot I_{width}$, which uniquely identifies the velocity in one single value and which is used as a key for fast access (see Section 2.3). The lists of hypotheses are sorted after this key, if not mentioned otherwise. Sorting is implemented by using the Intro-Sort algorithm that achieves $O(N \cdot \log N)$ complexity where N is the number of elements to be sorted [21] (implementation available through the C++ standard template library at http://www.sgi.com/tech/stl, 07/2005). Thus, sorting of hypotheses has a complexity of $O(h_{max} \cdot \log h_{max})$ at each image location. All hypotheses can be sorted in $O(n \log h_{max}) = O(n \log n)$, where n is the maximum number of all hypotheses at all locations. Such sorted arrays of hypotheses for each image location allow to quickly searching for the existence of a specific velocity at a given spatial location. Therefore, we use a binary search algorithm (e.g., [22], which is implemented in $O(\log h_{max})$). Moreover, if the hypotheses at each location are sorted according to their weight w instead of v_xv_y, we can directly select a subset of the m most likely hypotheses for further processing by simply using the first m hypotheses of the array. 2) Since hypotheses can also be represented on a subsampled spatial representation (see Section 2.3), we need copies of v_x, v_y, v_xv_y ($v_x^0, v_y^0, v_xv_y^0$) to store the velocity on the finest (original input) resolution. Consequently, for the **generation of a hypothesis**, we compute the combined value v_xv_y from v_x and v_y, we determine the values $v_x^0, v_y^0, v_xv_y^0$, and then store the hypothesis at the corresponding array location.

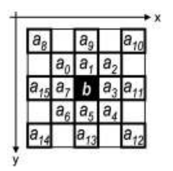
2. The weight thus corresponds to the activity of a neural cell tuned to the corresponding velocity. In contrast to the neural model, hypotheses are allocated dynamically at runtime and are not fixed at the beginning of the algorithm/simulation.

The data representation generated for velocities that range over the whole image requires a maximum of $I_{width} \cdot I_{height} \cdot h_{max}$ hypotheses. Thus, the representation needs $O(n) = O(n_p \cdot h_{max})$ bytes for the representation of the hypotheses describing the motion correspondences between two frames, where n_p is the total number of pixels of an input frame, h_{max} the maximum amount of hypotheses at each image location, and n the maximum number of all hypotheses. The constant factor is 20 bytes, since we utilize single-precision floating point variables (4 bytes) for the weights, integer variables (4 bytes) for the combined velocity values, and short integer variables (2 bytes) for the velocities. Assuming an image sequence with 320×200 pixels, the required memory to represent a maximum amount of, e.g., five hypotheses at each location at a specific time step is approximately 6.4 MB, independent of the maximum speed to be represented.

2.2 Algorithmic Realization of Initial Motion Estimation

The stage of initial motion estimation utilizes a discrete similarity measure between image features, particularly the Census transform [23]. The Census transform belongs to the class of rank-order approaches where the gray value at the center is compared with each value in a discrete neighborhood to generate the feature vector. Our approach is based on the algorithm proposed by [1], which uses the Census transform and Hashtables. Differences between their approach and our initial motion estimation are given in the discussion section. The novel aspect of our approach is that all motion correspondences are represented in an efficient way and no pre-selection based on local image structure heuristics is necessary for motion detection. As sketched in Fig. 4, the Census transform is applied to the input frames and the extracted feature vectors are converted to integer values (called *Census values*). Since these values contain information from a local neighborhood, each of them represents a compact description of local image structure.

For the **generation of hypotheses from Census transformed images**, we sort the Census values of two succeeding frames, which results in a compact representation of all motion correspondences (based on the correspondences of the Census values in the two frames). Sorting can be achieved in $O(n_p \log n_p)$, where n_p is the number of pixels in the image (see above). Further, the number of occurrences for all present Census values can be computed in $O(n_p)$. By linearly walking



$$c = \sum_{i=0}^{15} 3^i \cdot \begin{cases} 0 : a_i < b \\ 1 : a_i \approx b \\ 2 : a_i > b \end{cases}$$

Fig. 4. Census transformation at one specific location (black pixel in the center). We use 16 surrounding pixel values to compute the Census value c describing the local structure around the central pixel b. Each pixel value a_i is compared to the value of the central pixel. According to this comparison, individual bits of the resulting census value are changed (see equation in figure). A threshold of 6 (assuming 255 gray levels) is employed to determine if two luminance values are approximately equal. Additionally, the input images are blurred with a Gaussian $(\sigma = 1)$.The resulting Census value c lies within $0 \leq c < 43{,}046{,}721 = 3^{16}$.

through the sorted Census values of two frames, all motion correspondences between these two frames can be determined. To reduce the amount of memory required for the representation, we employ a selection process based on the input correspondences and knowledge from previous observations if available: 1) At locations with less than h_{max} motion correspondences, motion hypotheses are generated for each available correspondence. Such locations with only few motion ambiguities typically contain image structures like corners or line-endings, but our approach does not depend on the explicit detection of such image structures. 2) At locations with more than h_{max} motion correspondences the algorithm ignores all correspondences, except if they match expected motion hypotheses. Such expected hypotheses are generated from context information later in the algorithm and are fed back to the initial motion detection stage. The presence of an expected correspondence in the input can be tested in $O(1)$ by directly comparing the expected Census value with the Census value at the specific image location. We further ignore all correspondences representing a velocity with speed greater that s_{max} and completely ignore motion cues when more than h'_{max} correspondences are present ($h'_{max} > h_{max}$). Each selected motion correspondence is then transformed into a motion hypothesis with weight equal to 1.

In the first iteration, no expectations are present. As a consequence, only less ambiguous motion cues with less than h_{max} correspondences are selected to generate hypotheses to be represented explicitly by the algorithm. All other more ambiguous correspondences remain in the input signal as subthreshold motion information. Through iterative processing, such correspondences may be selected at a later time.

2.3 Biologically Inspired Algorithm for Motion Integration and Segregation

2.3.1 Overview

The algorithm is divided into two different modules, each representing motion hypotheses on specific spatial scales. The first module (corresponding to cortical area V1, referred to as *module V1*) represents motion hypotheses on the same scale/resolution on which they were detected. The second module (*module MT*, corresponding to cortical area MT) uses a coarser spatial resolution, where the accuracy of spatial location as well as velocity is reduced by the factor 5 which is in accordance with physiological findings [24]. The logic of the algorithm is that context information is assembled in module MT by forward integration of spatially adjacent local hypotheses from module V1. This context information is interpreted as expectation signal which is used to iteratively resolve ambiguities and to refine the motion estimation and segregation process.

In contrast to motion estimation approaches based on coarse-to-fine detection schemes, our approach has the ability to represent arbitrary movements of fine (small) structures. Such structures may not be represented in strong low-pass filtered versions of the input images and, thus, their corresponding image motion may not be extracted correctly on coarse image scales. In turn, image velocities of fine structures are typically overseen by coarse-to-fine approaches which rely on the initial guesses of image motion from coarser scales to compute the velocity on finer scales [25]. In our algorithm, the maximum loss of motion information caused by subsampling is defined by the constant rate 1:5 between module V1 and MT. Approaches with larger subsampling rates may oversee some important features and, thus, fail to detect the motion of small objects.

In a nutshell, the functionality of both interacting modules can be described as follows: raw motion estimations are detected at each time step in the first module (as described in Section 2.2). These motion cues typically are highly ambiguous at many locations caused by the aperture problem which leads to the presence of the true image velocity together with many other possible interpretations. In the second module, nearby motion hypotheses are combined by averaging their weights, when they represent the same velocities. Thus, assuming that motion of rigid objects is constant in a small neighborhood, the true image motion which has been measured at most locations of an object will generate the largest weight. More generally, due to the integration of weights from the previous stage we get more reliable motion estimations that are indicated by the motion hypotheses with the largest weights at each spatial location of the second module. From this representation, a prediction for the detected initial motion hypotheses at the next time step (or frame) in the first module is computed. By iteratively modulating the weights of these expected motion cues in the first module, the system resolves motion ambiguities over time. Furthermore, the predictive signal is utilized for the selection process for the initial motion detection. The detailed mechanisms of the iterative computation will be explained below.

2.3.2 Basic Operations and Mechanisms

1. **Recurrent modulation**. Recurrent excitatory modulation aims to enhance the likelihood of motion estimates that form the elements of a representation which match an expected signal, while leaving the likelihood of the rest of the representation unchanged. In our framework of sparse velocity cues, this is achieved by modulating the weight (w_{IN}) of the sparsely represented input velocities by utilizing a soft gating mechanism described in (1):

$$w_{IN} \cdot (1 + C \cdot w_{FB}), \qquad (1)$$

where w_{FB} is the expectation, or feedback signal which acts as a spatial reentry mechanism [4] (we set $C = 100$). The function of feedback modulation allows selective enhancement of the bottom-up activities (hypothesis weights w_{IN} in (1)) which

TABLE 1
Function and Effect of Excitatory Feedback Modulation

input (IN) \ feedback (FB)	0	b
0	0	0
a	a	$a + C \cdot ab$

Only if both input (a) and expectations (b) are in resonance, the feedback multiplication has an excitatory effect. Else, the input is left unchanged.

receive top-down same-velocity activity w_{FB} at the same location while leaving other inputs intact that receive no feedback enhancement (see Table 1). Because the feedback signal is of lower spatial resolution than the input signal, the weights are interpolated.[3] The complexity of the entire operation of recurrent modulation is $O(n_p \cdot h_{max} \cdot \log h_{max}) = O(n \cdot \log h_{max})$, since for each hypothesis $(n_p \cdot h_{max})$ the feedback signal has to be tested for a matching velocity (binary search, $O(\log h_{max})$).

2. **Weight normalization**. The weight normalization is necessary to guarantee that the weights of the hypotheses stay in certain bounds. This is realized by dividing the weight of each hypothesis at each location by the sum of weights of all hypotheses represented at the corresponding location. In other words, the total weight energy is conserved at each location. A small constant (0.001) is added to this sum of weights to prevent divisions by zero. We would like to highlight the role of the weight normalization. The conservation of energy leads to weakening the weights of ambiguous sets of hypotheses when multiple motions cues are present at one location. Consequently, an enhancement by, e.g., recurrent modulation, is generated at the cost of inhibiting the remaining weights. The complexity of this operation is $O(n)$.
3. **Subsampling**. Motion hypotheses represented in module V1 are spatially subsampled to build the representation in module MT. Therefore, hypotheses are pooled in a spatial neighborhood defined by a Hanning filter (e.g., [26]) of $4 \cdot \Delta s + 1$ pixel width with respect to the spatial resolution in module V1, where Δs is the spatial subsampling distance. The weights of the Hanning filter are normalized such that the coefficients sum to one. First, the subsampled output of the linear Hanning filter is computed. The result is a new list of weighted hypotheses at each subsampled image location, which may contain multiple hypotheses for the same velocity. After sorting, such multiple hypotheses are converted to single hypotheses by adding their weights. Because hypotheses from different locations are combined, the maximum number of hypotheses at each location may increase. By selecting a subset of hypotheses with maximum weights, the number of hypotheses is again reduced to a maximum of h_{max} velocities. The velocities of the resulting hypotheses are adjusted such that they represent rounded, integer valued subsampled velocities corresponding to the reduced resolution. These adjusted velocities are stored in addition to the velocity on the finest scale. The overall complexity of this operation is $O(n_s \cdot k_s \cdot h_{max} \cdot \log(k_s \cdot h_{max}))$, where k_s is the (constant) size of the kernel (the number of nonzero entries) and n_s denotes the number of subsampled locations. Note that $n_s \ll n_p$, where n_p equals the number of pixels of the input image.
4. **Velocity blur**. Velocity blur reduces the sharpness of the algorithmic response to a specific velocity by pooling over similar hypotheses. This is necessary to allow small motion variations within one segmented object. Without such operation, the slightest change in velocity at neighboring locations would lead to a sharp segregation of the motion signal into different spatial regions. The resulting velocities are dependent on the velocity of the input hypothesis and a filter kernel in the velocity domain. The output corresponds to the linear filtering (convolution) of the input in velocity space. Since the number of hypotheses may increase by this operation (similar as for the linear filtering in the subsampling operation), we select the h_{max} hypotheses with maximum weights. The complexity of this operation is $O(n_p \cdot k_v \cdot h_{max} \cdot \log(k_v \cdot h_{max}))$, where k_v is the (constant) size of the kernel in velocity space (the number of nonzero entries).
5. **Nonlinear enhancement**. This operation simply squares the weights of all hypotheses. The effect of the operation is that the relative differences of the hypotheses' weight coefficients are enhanced. The complexity of this operation is $O(n)$.
6. **Interpolation of subsampled hypotheses**. To extract the expected correspondences for the initial motion detection, the feedback signal from the second module has to be interpolated in order to expand the spatial resolution that matches the one of the first module (this operation is basically the same as in constructing, e.g., a Laplacian pyramid representation; [27]). Therefore, for each pixel in module V1 the interpolated hypotheses are generated by interpolation of the four nearest locations of the subsampled representation in module MT. The complexity of this operation is $O(n_p \cdot h_{max})$.
7. **Predictive shift**. The proposed algorithm is supposed to work with sequences of images with moving objects. Thus, the predictions from a previous time step have to be adjusted to match the input signal at a later time step. This is achieved by shifting the location of each hypothesis according to its velocity, i.e., a hypothesis that encodes a velocity of three pixels to the left is shifted by three pixels to the left in space. Because more than h_{max} hypotheses may be moved to a specific location, the algorithm combines hypotheses representing identical velocities and chooses the h_{max} hypotheses with maximum weights. The complexity of this operation is $O(n_p \cdot h_{max} \cdot \log(n_p \cdot h_{max}))$.

2.3.3 Algorithm

The algorithm is divided into two parts corresponding to modules V1 and MT, processed consecutively, and repeated iteratively (Fig. 5). As noted above, the differences between both modules are the spatial resolution of the representation and the different input signals. 1) **Module V1** computes initial motion estimates from two successive frames of the input

3. In order to achieve qualitatively good results at reasonable computational speed, we employed here a bilinear interpolation scheme.

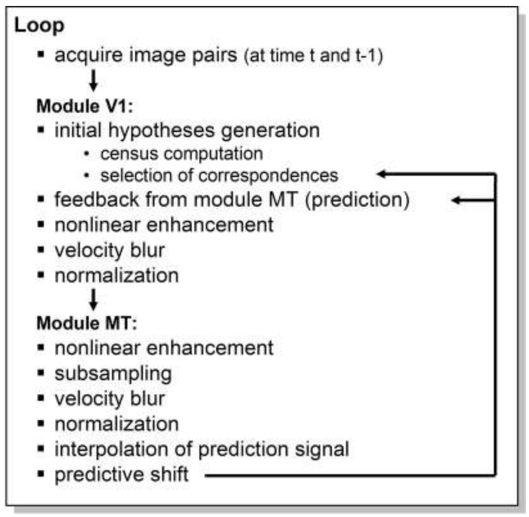

Loop
acquire image pairs (at time t and t-1)
Module V1:
initial hypotheses generation
census computation
selection of correspondences
feedback from module MT (prediction)
nonlinear enhancement
velocity blur
normalization
Module MT:
nonlinear enhancement
subsampling
velocity blur
normalization
interpolation of prediction signal
predictive shift

Fig. 5. Overview of the algorithm. The same operations (mechanisms) are employed in both modules. The only differences are their input signals and the spatial scale on which they operate.

sequence. Due to the spatially localized filtering employed for motion estimation, module V1 achieves a high spatial accuracy but generates low quality velocity estimations. Contrarily, 2) **Module MT** subsamples the output from module V1 which implies spatial pooling and which leads to larger apertures. Accordingly, the second module is less affected by the aperture effect and generates better velocity estimations, but with less spatial accuracy. After each iteration, the hypotheses in model MT are shifted in space according to their velocity to track the represented motion cue. Thus, the signal from module MT can be interpreted as predictor for the input expected in module V1.

After initial motion detection in module V1, the likelihoods of the generated hypotheses are modulated by the reentry, or expectation, signal received from module MT. As described in [10], this feedback modulation combines the high spatial accuracy of the first module with the improved velocity information of the second module by giving those hypotheses in module V1 a bias which match the feedback signal. Module MT does not receive any predictive modulation (therefore, in the current model implementation there is no feedback modulation step incorporated in the algorithm for module MT). Furthermore, the feedback signal from MT is employed to select subthreshold motion correspondences at locations with very ambiguous motion cues.

Within each module, the (modulated) input is nonlinearly enhanced, filtered, and normalized. The filtering step in module V1 consists only of a lowpass in velocity space, while module MT also applies a spatial subsampling which implies spatial lowpass filtering. The nonlinearity on one hand gives those hypotheses an extra advantage, which already have a higher likelihood compared to other hypotheses at a given location. The normalization on the other hand keeps the output in certain bounds and leads to flat likelihood distributions in the presence of ambiguities and to strong peaks for unambiguous motion cues where only a few hypotheses are likely to be present. The combined action of these mechanisms in both modules together with iterative feedback modulation implements a biased competition [28] and generates a context-dependent winner-takes-all behavior by gradually evolving a winning velocity hypothesis at each location.

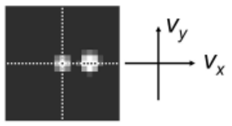


Fig. 6. Example of an extracted ambiguous population code; activities shown as luminance and arranged in velocity space. Here, a set of 25 hypotheses describes two blobs of possible velocities. A weaker blob around the zero-velocity (in the center) and a blob of stronger activities (weights) for rightward motion (on the right).

The main difference to the neural model described in [10] is the selection of expected motion correspondences in the initial motion estimation. This gives the algorithm an extra advantage in computational speed since very ambiguous motion cues are not explicitly represented and remain hidden in the input as subthreshold signal.

2.3.4 *Interpretation of Algorithmic Output*

At each location and at each time step, the estimated motion is represented as a set of weighted hypotheses. Such a set of hypotheses at one location can be interpreted as local neural population code representing distributed neural activity (average spike rates) by means of local likelihoods for individual velocities (see Fig. 6). Such interpretation is consistent with the tuning of cells in macaque area MT to different directions and speeds (e.g., [29]). One single velocity at each location is computed by the weighted sum of the velocity vectors of all hypotheses, weighted by the corresponding likelihoods (population vector estimator; [30]).

3 Results

The parameterization of the algorithm was held constant for all experiments if not mentioned otherwise. We used $h_{max} = 5$, $h'_{max} = 1000$, and $s_{max} = 30$; the Census value was computed as described in Fig. 4. The image sequences for the different experiments are described in the corresponding paragraphs. The algorithm was run on a standard PC (Intel Centrino 1.7Ghz with 1GB RAM). The estimated image velocities are shown using a color code and in some cases the direction is indicated by oriented lines (see, e.g., Fig. 7). For the color code, the direction of motion is encoded as hue, and the speed as saturation.

3.1 Results with Artificial Motion Sequences

Fig. 7 shows the results of the algorithm processing a synthetic scene with two non-overlapping objects (a rectangle and a bar) moving in different directions. The example illustrates that initially ($t = 0$) only unambiguous motion cues are selected to be represented by the algorithm, particularly in module V1. Iteratively, subthreshold correspondences implicitly encoded in the input pattern are selected by the expectation signal generated by module MT. This example demonstrates that the motion aperture problem can be solved by the algorithm in a scale invariant manner by recurrent information processing. Moreover, it shows that reliable motion cues near structures such as corners are estimated without the necessity to employ specific feature detectors to explicitly detect those structures. The runtime time of the algorithm for this sequence (120×120 pixel) was

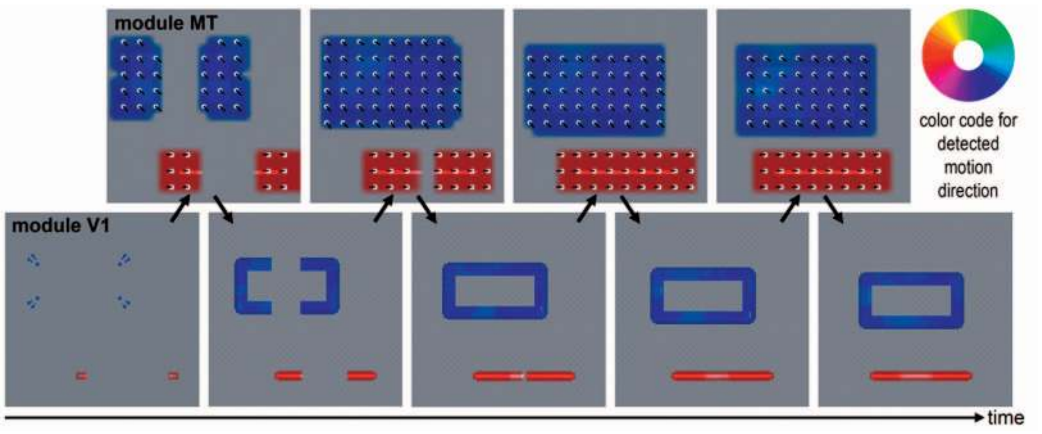


Fig. 7. Example processing a synthetic stimulus showing two moving objects: A box moving to the lower right and a bar moving to the left. Detected motion direction is indicated for both modules as colored modulation of the input frames (color code: see inlay). For module MT, the detected velocity is shown at selected locations (white discs) as black lines pointing in the direction of motion. The temporal development of motion estimations is shown from left to right. Initially, only unambiguous motion cues are selected at corners or line endings. Over time, motion information is propagated along object boundaries.

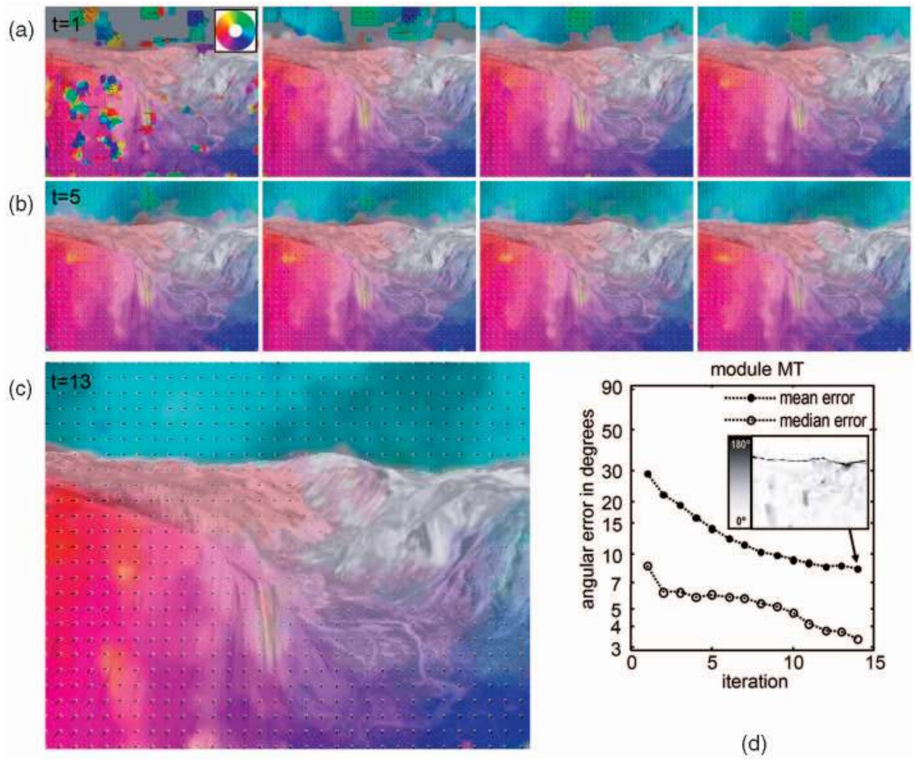


Fig. 8. Example processing the Yosemite sequence showing the propagation of motion cues. (a) The initial motion detection yields only very sparse unambiguous motion estimations in the image region covered by the sky. This high ambiguity in the sky is caused by the low contrast which leads to many identical Census values. (a), (b), and (c) The initially sparsely detected motion cues propagate over time and disambiguate motion information. Estimated velocities in module MT are shown for $t = 1, \ldots, 8$ and $t = 13$; color code: see inlay for $t = 1$). (d) Temporal development of the average and the median angular error of the entire flow field in module MT, excluding locations where no hypothesis are present (e.g., at some locations in the sky) or where the groundtruth or the detected velocity has zero speed. The inlay shows the angular errors at each image location for $t = 13$ which are utilized to compute the shown mean and median error (dark = large errors). The median error of 3.3° (after 13 iterations) illustrates the accuracy of the model excluding outliers with errors up to 180°, which typically occur at region boundaries along the horizon (compare inlay). The density of estimated flow in module MT is $> 99\%$ for $t \geq 3$.

between 100-200 ms/frame (with increasing number of selected hypotheses more time is required).

Fig. 8 illustrates the propagation of motion information in a more complex artificial sequence with real-world textures and shows a quantitative analysis of the detected optic flow (Fig. 8d). In module MT, we obtain a flow field of approx. 100 percent density with a median angular error of 3.3° after 13 iterations. The density in module V1 is smaller (about 43 percent) with higher errors (median error: 6.1° after 13 iterations; data not shown) due to outliers generated by the selection process. In the subsampling process in module MT such isolated erroneous motion cues are eliminated through competition since no coherent support is available from neighboring locations. Compared to technical approaches which yield optic flow estimation with 100 percent density the algorithm compares well [2]. Recent approaches [8], [31],

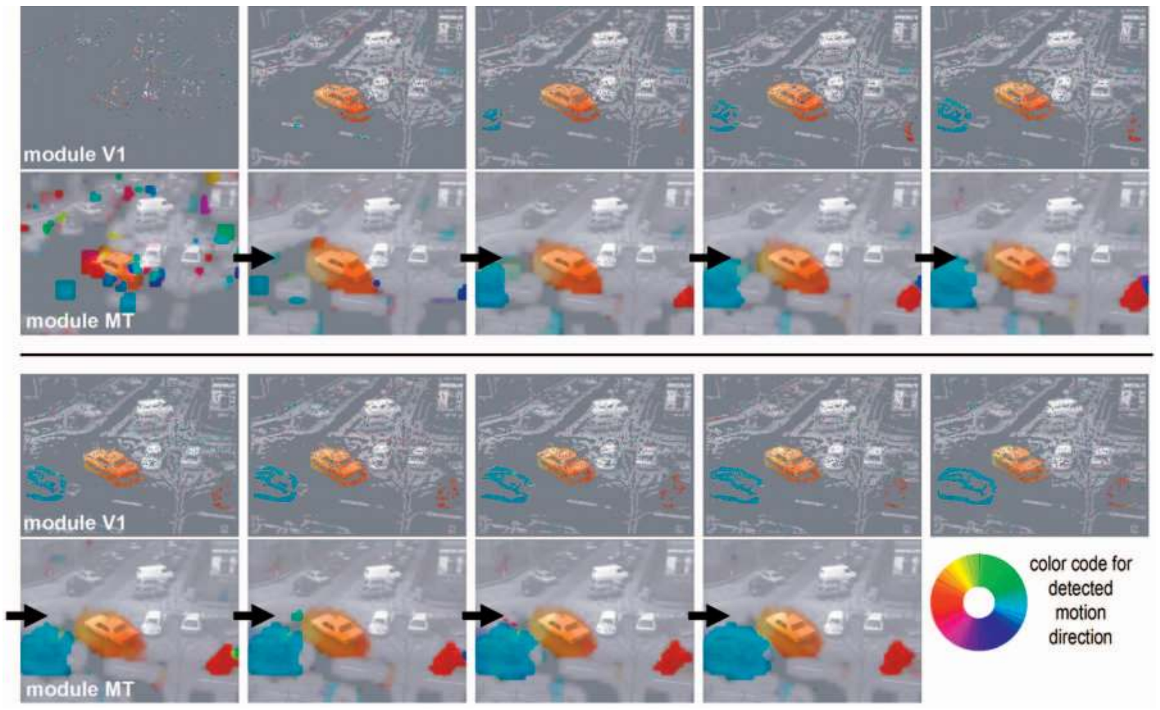


Fig. 9. Example processing a real-world sequence (the Hamburg Taxi Sequence; obtained from http://i21www.ira.uka.de (07/2005), H.-H. Nagel, KOGS/IAKS Universität Karlsruhe). Detected direction of motion is color coded (see inlay). The algorithm stabilizes after three iterations and tracks the three cars over time: Estimations from module V1 and MT shown at different time steps from left to right arranged in two lines. This example illustrates the different spatial resolutions of both modules. Spatial motion localization is very fine in module V1, while module MT shows only coarse blobs of motion.

however, perform better but do not show the advantages of the sparse voting scheme (sparse population coding) coupled with the ability of the neural model to easily include expectations by a modulatory expectation signal.

3.2 Results with Real-World Motion Sequences

With the same parameterization, the algorithm also processes real-world images. The Hamburg Taxi-Sequence (Fig. 9) shows a street with three cars: a taxi turning right into the street at the junction, and two cars coming from the left and the right, respectively, partly occluded by a tree. The initial motion estimations are very noisy and contain a lot of false motion estimates. After the third iteration, the motion signal is already stabilized and indicates the motion of the three cars. This example illustrates the different spatial resolutions in both modules: in module V1 the image structure is clearly visible in the motion signal, while in module MT there are large blobs indicating the presence of moving objects. The required computation time of the algorithm for this sequence (256×191 pixel) was 700-1,400 ms/frame.

In Fig. 10, we present the results of processing the Flower Garden sequence used by [32] with 360×240 pixel resolution. This sequence contains image motion strongly influenced by the motion parallax caused by an observer moving to the right. Objects near to the observer (e.g., a tree in the front) are strongly influenced by his translation which leads to fast image motion to the left while objects farer away (e.g., some houses) are less affected which leads to slower image motion to the left. Furthermore, the ground has continuously changing depth values in the vertical direction which generates a vertical speed gradient. The results show that the different speeds are successfully detected by the algorithm and that particularly the tree is segregated from the background. The illustration also shows that for representing a velocity or speed gradient more than five hypotheses are necessary in order to take advantage of the interpolation properties. These interpolation properties depend on blurring velocity space and, thus, on being able to represent the blurred and thus flattened hypotheses distributions. Fig. 10c demonstrates that 25 hypotheses in the second module are enough to generate smoothly interpolated speeds and, thus, to represent and detect the speed gradient induced by the smoothly varying depth of the ground.

Fig. 11 shows another real-world sequence that contains flow information from self motion of a moving car and a pedestrian crossing the street. We make similar observations concerning the spatial resolution and the temporal development as for the Hamburg Taxi sequence. The results demonstrate the motion separation capabilities of the algorithm by segregating the foot passenger and a traffic sign from the background differing with their motion from the background motion. The background motion itself is an expanding flow pattern induced by the self-movement of the car. The runtime of the algorithm for this sequence (320×200 pixel) was approximately 1,000-1,600 ms/frame. Fig. 12 shows further frames of the same sequence, when an independently moving car from the left comes into sight. This car is detected very fast, which demonstrates that our algorithm is able to react on novel motion patters not initially expected by the feedback signal. After only two cycles of iteration, module MT has built an updated representation and, in turn, enhances and stabilizes this object by modulatory feedback.

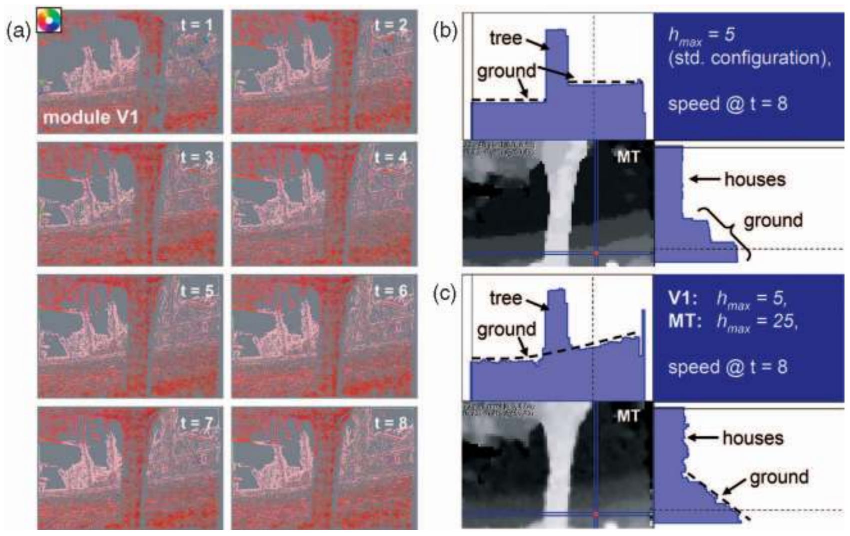


Fig. 10. Example processing a real world sequence (the Flowergarden Sequence; obtained from http://www-bcs.mit.edu/people/jyawang/demos/garden-layer/layer-demo.html (07/2005); Wang and Adelson, 1994). The sequence shows a tree in front of houses passing to the left caused by a translation to the right of the observer. The motion parallax leads to slower motions for objects farther away from the observer than for near objects. (a) Detected direction of motion in module V1 at different time steps is color coded (red = leftward motion). Different speeds are visible through the color saturation (pink is slower than fully saturated red). (b) and (c) Detected speed in module MT is shown as luminance ([black, white] is mapped on approx. [0, 7] pixel/frame) and as profile plots for the selected columns and rows. The dotted lines indicate position of the selected row/column in the column/row profile plot, respectively. The dashed lines in the profile plots illustrate the detected speeds from projections of the moving ground. (b) With $h_{max} = 5$ different regions of segregated motion are detected on the ground where the depth increases continuously (compare dashed lines). (c) With $h_{max} = 25$ for module MT (module V1 still uses $h_{max} = 5$), speed is smoothly interpolated in the second area which then is able to detect and represent continuous speed gradients (compare dashed lines).

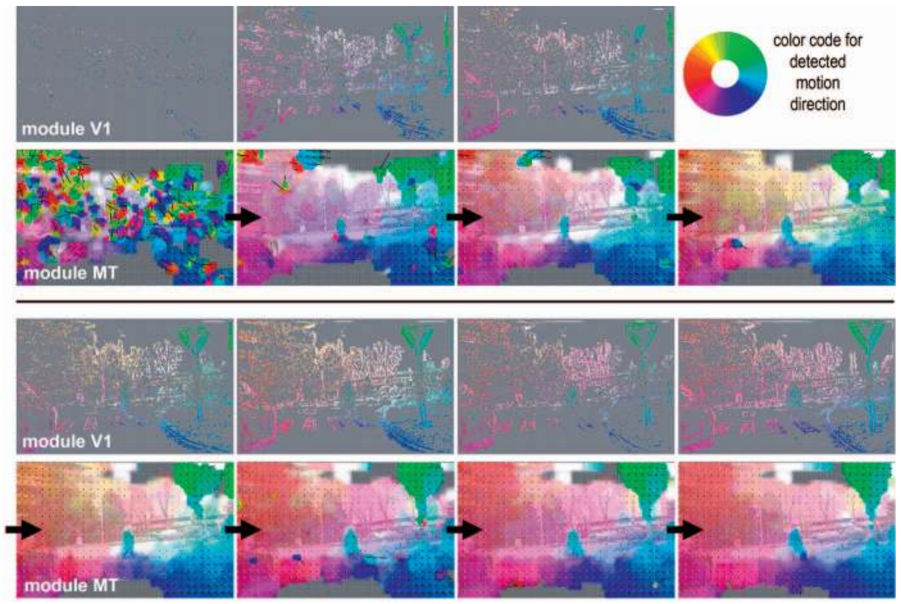


Fig. 11. Example processing a real world sequence showing the self-motion of a car with a pedestrian crossing the street. The image sequence has been gathered within project 23-7532-24-12-19/2 (see Acknowledgements). The isolated pedestrian is segregated from the expanding flow field. The expanding flow field is induced by self-motion and generates a gradual change of velocity over the entire image. The street panel also segregates from the background, because it differs in velocity since it has a significantly different image depth.

3.3 Interpretation of Algorithmic Population Codes

The algorithm represents motion as distributed sets of motion hypotheses and the operations on the hypotheses are based on neural mechanisms. Thus, the output of the algorithm can be interpreted as a neural population code where the likelihoods represent neural activity and no neural activity is assumed for velocities where no hypotheses are present. Such interpretation of the data can help visualizing the detected hypotheses and their likelihoods at individual locations. Fig. 13 shows the results of the algorithm processing a sequence of a textured rectangle slanted in space moving over a textured plane in the background. Here, we utilized a value of 25 for h_{max} in module MT to allow a larger amount of hypotheses to be represented at each location (module V1 still uses $h_{max} = 5$). The resulting optic flow pattern contains a motion gradient due to the perspective projection of the slanted object. The extracted population codes illustrate the detected motion

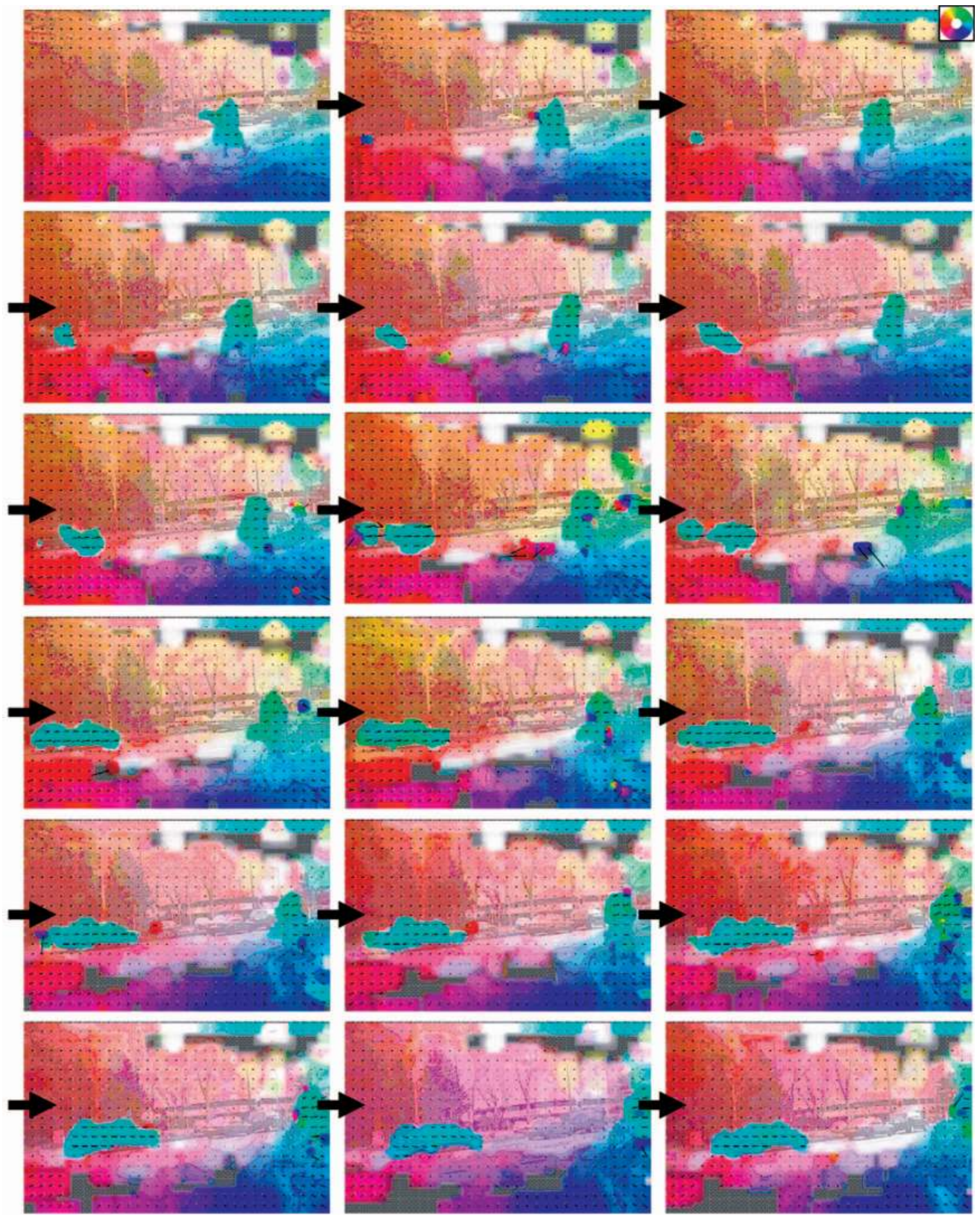


Fig. 12. Continued example processing the real world sequence shown in Fig. 11: Detected motion in module MT. The algorithm is able to detect the car newly appearing from the left of the sequence. The new object is detected from the second frame it appeared and completely stabilizes after eight frames.

hypotheses and clearly show the smooth gradient in the field of velocities. The increased number of hypotheses (h_{max}) in module MT hardly affects the computational performance because of the low resolution in module MT.

4 Discussion

We first discuss the differences of the biologically inspired algorithm and the underlying neural model [10] and to another neural model of motion integration [33], [34]. Then, we compare the proposed algorithm to the approach presented by [1] and to other related approaches.

4.1 Comparison of the Proposed Algorithm with Neural Models

We presented a biologically inspired algorithm of motion integration and segregation based on a sparse representation of motion hypotheses in velocity space and neurally inspired interactions between hypotheses. Here, we discuss the differences between the algorithm and the neural model proposed in [10] from which the algorithm is derived and to

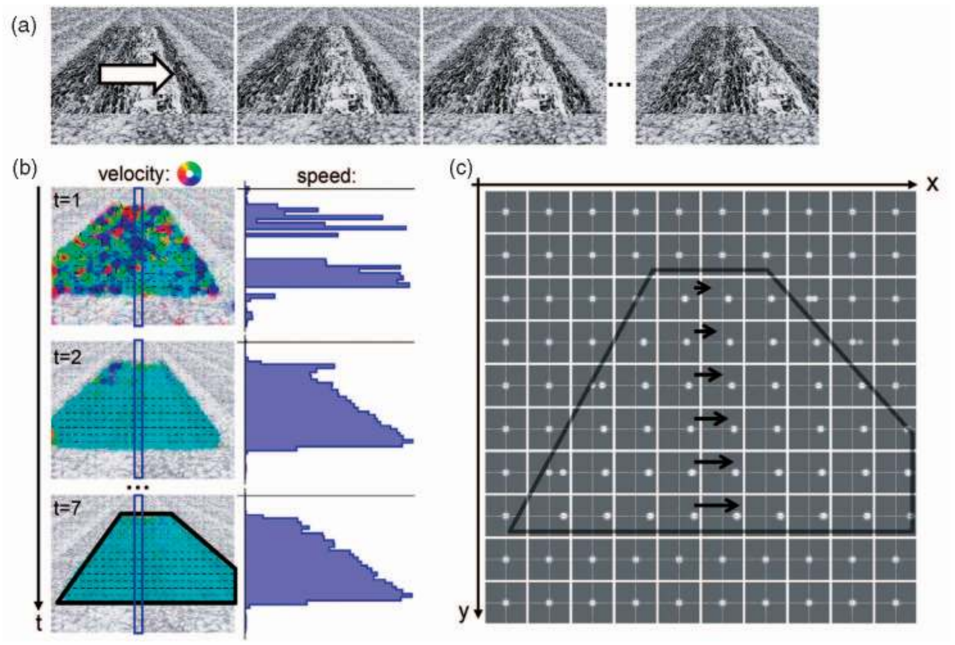


Fig. 13. Extracted population code. (a) Image sequence including a vertical velocity gradient (see text). The true direction of motion is indicated by the white arrow. (b) Color coded detected motion directions in module MT and the speed profiles of the selected column. (c) Extracted population codes at selected locations within the entire field of view. A velocity gradient is visible within the object (the object indicated by its outline). At object boundaries, multiple motion cues are detected which indicates an uncertainty at these locations.

the neural model proposed by [33], [34]. There exists a bunch of neural models which contain operations also utilized in [10] and in the proposed algorithm, such as, e.g., a stage of activity normalization [3], [18]. The approach of Nowlan and Sejnowski [33], [34] is of special interest in the context of the presented algorithm because it describes a selection process similar to the selection process implemented in our approach. A detailed discussion about differences of [10] to other neural models of motion perception is given in [10].

The **differences between the biologically inspired algorithm and the corresponding neural model implementation** are consequences of the sparse algorithmic representation of visual motion and its initial detection.

1. First, the input stage in the algorithm produces binary decisions whether certain velocities are present while the neural model generates gradual likelihoods/activities between 0 and 1 for individual velocities. A consequence of this binary similarity (match/nonmatch) is that ambiguities arising from transparent motion cannot be well detected in the input stage of the algorithm. This is different for the neural model where gradual similarity in the input image allows extracting meaningful likelihoods for transparent motion.[4]
2. Second, in the presence of high ambiguities the algorithm selects raw input velocity hypotheses prior to the feedback modulation using the hypotheses from the feedback signal as predictor (see Section 2.2). This leads to a much stronger influence of the feedback signal on the motion estimation process than in the neural model. The algorithm's initial motion detection stage only detects motion expected from prior observations or motion with ambiguities up to a maximum level of uncertainty. This uncertainty is expressed here by the number of different possible correspondences that represent different velocities. Contrarily, the neural model detects every velocity to which any of the model cells is tuned.
3. Third, the algorithm allows a maximum of h_{max} hypotheses to be present at each location which is achieved by a selection of the h_{max} most likely hypotheses (see, e.g., the subsampling operation in Section 2.3). This selection process is comparable to a dynamic threshold operation of neural activity, which is not contained in the neural model. Cutting off small activities in the neural model means to suppress ambiguous motion. This is a consequence of normalization which leads to small activities for ambiguous motion configurations.

To sum up, the maximum ambiguity which can be represented explicitly by the algorithm is limited by h_{max} and is generally much smaller than the maximum ambiguity which can be represented by the neural model. More ambiguous motion cues are represented as subthreshold signal in the input to the algorithm. Such ambiguous motion cues can only be selected and therefore enhanced if they are predicted from previous observations at neighboring locations. Moreover, the biologically inspired algorithm generally allows a much larger range of velocities to be detected than the neural model, requires less memory, and is computationally much faster. Altogether, this makes the algorithm feasible for applications in computer vision and at the same time provides a fast solution for early motion processing for neural modeling.

4. Moreover, the neural model as described in [10] aims to extract one winner at each location which is conflicting with the presence of multiple transparent motion cues. For processing transparent motion, additional mechanisms of mutual competition are needed to allow a nonconflicting local coexistence of multiple velocities as described in, e.g., [35], or as implemented as extension of our neural model in [36].

The major **differences between the motion algorithm** proposed in this paper and the approach presented by **Nowlan and Sejnowski** [33], [34] are two-fold and consist 1) of the motion detection and representation and 2) of the refinement of initial estimations. In [33], [34], the authors describe a model of motion integration that utilizes an explicit selection signal that is computed to determine the regions in the visual field where velocity estimations are most reliable. Motion-sensitive cells are then gated by this signal to produce the final estimate. Instead of using the computational operation of the soft-maximum as in Nowlan's model we employ a nonlinear enhancement of activities coupled with a divisive normalization. Although similar in the computational effect, our approach allows easily separating and, thus, understanding the effects of the nonlinear enhancement and the normalization. Moreover, similarly as for the model on which the algorithm is based [10], Nowlan's approach utilizes a dense coding of velocities limited in the maximum speed to be detected, whereas the proposed algorithm employs a sparse voting scheme in velocity space and an inital motion detection for arbitrary speeds. This allows our algorithm to process sequences with more different velocities present than neural implementations while using less computational resources. Compared to the selection process described in [34], the concept of iterative selection and feedback modulation proposed in this paper is fundamentally different. Nowlan's model is a strict feed-forward model, whereas our approach utilizes an iterative refinement over time taking into account the history of motion measurements from previous time steps. Instead of gating the initial hypotheses with some selection signal as in [34], our algorithm gradually modifies the weights of detected hypotheses to evolve the final estimate by means of recurrent modulation coupled with normalization. The recurrence of the algorithm presented in this paper allows combining information from distant locations without losing much of the spatial accuracy from the initial motion detectors.

4.2 Comparison to the Approach of Stein (2004)

The generation of the input signal for the presented algorithm is based on the **Census Transform** [23] and contains components also used in [1]. The differences to the presented algorithm are 1) that the approach of [1] utilizes a rule-based combination of motion hypotheses with a short temporal memory of 2-3 frames without spatial interaction. They refine the detected hypotheses using a temporal analysis of detected motion without combining information across space. Unlike [1], our algorithm realizes a biologically inspired approach also capable of solving ambiguous motion hypotheses by employing spatial coherence of detected hypotheses. We utilize information from previous time steps through feedback modulation and from a spatial neighborhood through spatial pooling. Moreover, decisions whether some hypotheses should be rejected or not are solved in a soft manner instead of a rule-based decision. On one hand, in the absence of evidence from prior hypotheses, a detected hypothesis may be regarded as significant if it represents an unambiguous estimation. On the other hand, if an expected hypothesis is not detected at one time, information from previous observations cannot generate (invent) illusory motion cues. Only if both, input correspondences and feedback hypotheses are in resonance, feedback modulation amplifies the expected motion patterns (see Table 1). 2) The approach of [1] employs a list of signatures (Census values) to classify potentially meaningful image patches based on their structural appearance. Basically, this list represents a fast approximation to a standard corner detector, such as [37], [38]. In contrast, our algorithm does not need to explicitly detect those structures. We extract the information about the relevance of motion cues solely based on the ambiguity which is reflected by the number of detected correspondences.

In addition, [1] uses a hash table (see, e.g., [22]) to efficiently store and retrieve the motion correspondences instead of the fixed array utilized in our approach. On one hand, we could also use a hash table, but the maximum gain in complexity would be $\log(h_{max})$. On the other hand, in order to get the advantage in complexity the hash table requires a large amount of memory. This amount depends of the number of possible census values (compare Fig. 4) which in the presented examples is much larger than the memory required by the array representation of the presented algorithm.

4.3 Comparison to Other Approaches

The selection of the hypotheses and the computation of the hypotheses' weights can be related to Maximum Likelihood methods (e.g., [39]), Bayesian approaches [5], [6], and to the Condensation algorithm [40].

The basic idea of the **Maximum Likelihood method** (e.g., [39]) is that some solutions are more likely to result from a given observation than others. An estimator which selects the most likely solution based on some given likelihood function is called a maximum likelihood estimator. In our algorithm, we select the h_{max} most likely solutions based on the similarity measure defined by the feedforward processing (initial motion detection and linear filtering in the spatial and velocity domain). Thus, the resulting hypotheses of our algorithm without iterative feedback (e.g., after the first iteration) can be interpreted as the result of a Maximum Likelihood estimator. The employed likelihood function of our model is given by the model operations described in Section 2.3.

The only difference between **Bayesian approaches** and the Maximum Likelihood method is that a posterior probability is utilized in addition to the observed likelihoods or probabilities. [6] argued that the integration of top-down knowledge (contextual priors) and bottom-up observations in the visual cortex can be described in a Bayesian framework. Concluding, the weights of the expected hypotheses generated in module MT of our algorithm act as contextual prior for the observations in module V1. The contextual prior is computed by $(1 + C \cdot$ expected likelihood) and is 1) uniformly distributed across velocity space if no expectations are present or, else, 2) contains increased likelihoods for predicted motion cues. However, this interpretation strictly holds only if observations at different time steps originate from independent random processes.

The proposed algorithm shares several properties with the **Condensation algorithm** [40]. Both approaches, the Condensation algorithm and the proposed biologically inspired algorithm, realize a hypotheses and test cycle to track and stabilize hypotheses. In contrast to Kalman filtering [41], both approaches represent sparse hypotheses together with corresponding likelihoods without any premises to the underlying tracked distribution of likelihoods. Here, we present the individual steps of the Condensation algorithm

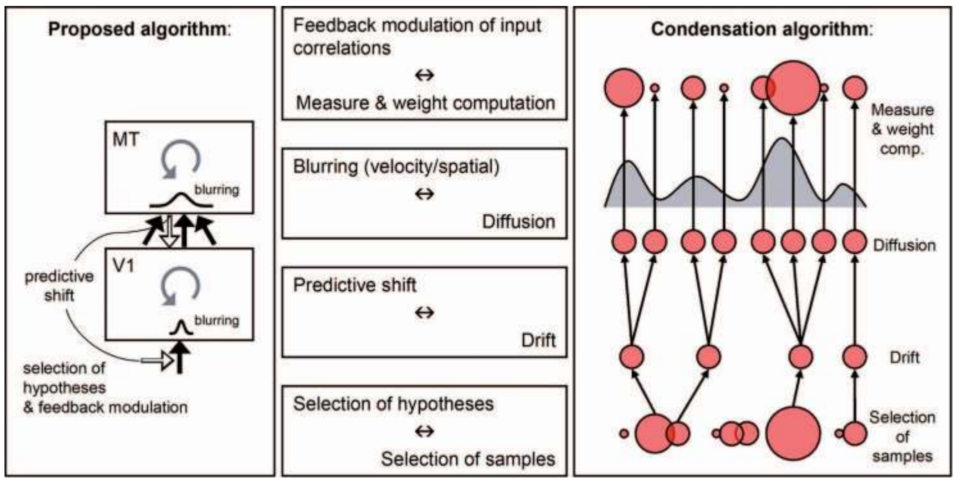

Proposed algorithm:
MT
blurring
V1
blurring
predictive shift
selection of hypotheses & feedback modulation
Feedback modulation of input correlations ↔ Measure & weight computation
Blurring (velocity/spatial) ↔ Diffusion
Predictive shift ↔ Drift
Selection of hypotheses ↔ Selection of samples
Condensation algorithm:
Measure & weight comp.
Diffusion
Drift
Selection of samples

Fig. 14. Comparison of the presented algorithm with the Condensation algorithm. The proposed approach shares many properties of the condensation algorithm. **Left**: Sketch of the presented algorithm. In addition to Fig. 2 and, thus, in addition to the neural model we included the selection of initial hypotheses. **Right**: The sketch of the Condensation algorithm, redrawn after [40]. **Center**: Operation/mechanism of the presented algorithm and algorithmic equivalent in the Condensation algorithm.

and discuss their similarities and correspondences to individual mechanisms implemented by the algorithm described in this paper (compare Fig. 14). The major difference between our approach and the Condensation algorithm is that our algorithm computes the diffusion of information and deterministically selects samples based on the combined bottom-up and top-down likelihoods, both in a deterministic fashion. On the contrary, the Condensation algorithm utilizes random processes to achieve these operations.

1. **Drift**. The drift of individual hypotheses in the Condensation algorithm is described by the given dynamic model (e.g., a shift describing image velocity). It realizes a prediction of the expected next observation based on the preceding observation. This operation directly corresponds to the predictive shift in our algorithm.
2. **Diffusion**. The diffusion described by the Condensation algorithm is realized as Brownian motion of discrete state parameters of individual hypotheses. Each parameter changes its value according to a normally distributed random sample. This diffusion process can be compared by the blurring operations in the spatial and the velocity domain described by our algorithm. Here, the combination of similar velocities within a certain neighborhood realizes an (isotropic) diffusion of information. In contrast to the Condensation algorithm the outcome of the diffusion step in the proposed approach is deterministic and directly computed by discrete sampling and combination of data instead of a stochastic diffusion based on random noise.
3. **Measurement**. The measurement step of the Condensation algorithm computes likelihoods for all individual hypotheses to occur under the given observations. This corresponds to the computation of weights of individual expected motion hypotheses in the proposed biologically inspired algorithm. Unlike the Condensation algorithm, we also generate unexpected sets of hypotheses at any location if they are unambiguous enough. Thus, our approach guarantees that all input signals are detected if they contain distinct motion information.
4. **Factored Sampling**. The sampling of hypotheses described in the Condensation algorithm differs from the selection process in the proposed approach. Instead of randomly choosing new hypotheses based on the previous likelihoods we select only those hypotheses which have the highest likelihoods. In contrast to the Condensation algorithm we do not allow to select identical hypotheses multiple times. Furthermore, we only allow integer velocities to be represented. Thus, the outcome of our algorithm contains always a minimum number of *different* hypotheses. This advantage ensures that the proposed algorithm can detect changes in the input velocities, which may be overlooked when the sampled hypotheses are all nearly identical.

Concluding, our algorithm shares several properties of the Condensation approach for detection and tracking of visual motion. The common properties further clarify the functional aspects of the individual mechanisms. Unlike the Condensation algorithm, the proposed architecture guarantees not to overlook any unambiguous input, even if it is not expected. Furthermore, the biologically inspired algorithm contains the normalization mechanism which goes beyond the capabilities of the Condensation algorithm and, together with the other mechanisms, realizes the disambiguation of ambiguous motion patterns and the segregation of regions of different motion.

5 Conclusion

We presented a simple and fast algorithm computing low-level motion processing in a neural fashion. The proposed algorithm iteratively solves ambiguities by spatiotemporal combination of available information. The outcome of the algorithm can be interpreted as neural population code and, moreover, the modular design of the framework allows for easy extension of the algorithm by other modules for further information processing. Furthermore, we identified similarities of the presented approach to the Condensation algorithm, which both realize a prediction and test cycle to find and track a hypothesis of enhanced likelihood.

The outstanding properties of the algorithm are that 1) computational complexity does not depend on the maximum speed to be detected, which often influences some kind of search range. 2) The proposed approach solves the aperture problem for objects of arbitrary size while preserving the spatial localization of motion cues to spatially segment the motion information. 3) The algorithm is demonstrated to react on changes in the input, such as novel objects entering the field of view.

Although the implementation does not run in real-time the algorithm represents a step towards it. We achieve computation speeds of 100ms-2,000ms per frame, depending on the complexity of the presented input information and, thus, the required representational load of the estimated motion responses. Additional spatial sparseness on one hand would highly increase the computation speed, but, on the other hand, would not generate dense optic flow estimations. Furthermore, the framework leaves enough space for further optimizations, such as, e.g., the heuristics utilized by [1] to select unambiguous motion cues. Our implementation is solely based on standard C and C++ libraries (using the GCC compiler). Thus, libraries taking advantage of hardware optimizations in CPUs (Intel Integrated Performance Primitives—IPP; http://www.intel.com, 07/2005) or GPUs (OpenVIDIA : GPU accelerated Computer Vision Library; http://openvidia.sourceforge.net, 07/2005) for signal processing would further increase the computation speed by a significant amount.

Acknowledgments

This research has been supported in part by a grant from the Ministry of Science and the Arts of Baden-Württemberg to Heiko Neumann (Az.: 23-7532-24-12-19/2) and by a grant from the European Union to Heiko Neumann (EU FP6 IST Cognitive Systems Integrated project: Neural Decision-Making in Motion; project number 027198).

The authors gratefully acknowledge Cornelia Beck for insightful comments on the manuscript and three reviewers for their constructive criticism that helped to improve the manuscript.

References

[1] F. Stein, "Efficient Computation of Optical Flow Using the Census Transform," *Proc. Symp. Die Deutsche Arbeitsgemeinschaft für Mustererkennung,* pp. 79-86, 2004.

[2] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, "Performance of Optical Flow Techniques," *Int'l J. Computer Vision,* vol. 12, no. 1, pp. 43-77, 1994.

[3] S. Grossberg, "How Does a Brain Build a Cognitive Code," *Psychological Rev.,* vol. 87, pp. 1-51, 1980.

[4] O. Sporns, J.A. Gally, G.N. Reeke, and G.M. Edelman, "Reentrant Signaling among Simulated Neuronal Groups Leads to Coherency in Their Oscillatory Activity," *Proc. Nat'l Academy Science,* vol. 86, pp. 7265-7269, 1989.

[5] Y. Weiss, E.P. Simoncelli, and E.H. Adelson, "Motion Illusions as Optimal Percepts," *Nature Neuroscience,* vol. 5, no. 6, pp. 598-604, 2002.

[6] T.S. Lee and D. Mumford, "Hierarchical Bayesian Inference in the Visual Cortex," *J. Optical Soc. Am. A,* vol. 20, no. 7, pp. 1434-1448, 2003.

[7] B.K.P. Horn and B.G. Schunk, "Determining Optical Flow," *Artificial Intelligence,* vol. 17, pp. 185-203, 1981.

[8] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High Accuracy Optical Flow Estimation Based on a Theory for Warping," *Proc. Eighth European Conf. Computer Vision,* pp. 25-36, 2004.

[9] H. Nagel and W. Enkelmann, "An Investigation of Smoothness Constraint for the Estimation of Displacement Vector Fields from Image Sequences," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 8, pp. 565-593, 1986.

[10] P. Bayerl and H. Neumann, "Disambiguating Visual Motion through Contextual Feedback Modulation," *Neural Computation,* vol. 16, no. 10, pp. 2041-2066, 2004.

[11] M. Mishkin, L.G. Ungerleider, and K.A. Macko, "Object Vision and Spatial Vision: Two Central Pathways," *Trends in Neuroscience,* vol. 6, pp. 414-417, 1983.

[12] D.C. Van Essen and J.L. Galant, "Neural Mechanisms of Form and Motion Processing in the Primate Visual System," *Neuron,* vol. 13, pp. 1-10, 1994.

[13] D.H. Hubel and T.N. Wiesel, "Receptive Fields and Functional Architecture of Monkey Striate Cortex," *J. Physiology,* vol. 195, pp. 215-243, 1968.

[14] J.A. Movshon, E.H. Adelson, M.S. Gizzi, and W.T. Newsome, "The Analysis of Moving Visual Patterns," *Pattern Recognition Mechanisms (Pontificiae Academiae Scientiarum Scripta Varia 54),* C. Chagas, R. Gattass, and C. Gross, eds., Vatican Press, Rome, (Reprinted in Experimental Brain Research, 11 (suppl.), pp. 117-151, 1986), pp. 117-151, 1985.

[15] T.D. Albright, "Direction and Orientation Selectivity of Neurons in Visual Area MT of the Macaque," *J. Neurophysiology,* vol. 52, pp. 1106-1130, 1984.

[16] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[17] P. Dayan and L.F. Abbott, *Theoretical Neuroscience.* MIT Press, 2001.

[18] E.P. Simoncelli and D.J. Heeger, "A Model of Neuronal Responses in Visual Area MT," *Vision Research,* vol. 38, pp. 743-761, 1998.

[19] E.I. Knudsen, S. du Lac, and S.D. Esterly, "Computational Maps in the Brain," *Ann. Rev. Neuroscience,* vol. 10, pp. 41-65, 1987.

[20] E. Adelson and J. Bergen, "Spatiotemporal Energy Models for the Perception of Motion," *Optical Soc. Am. A,* vol. 2, no. 2, pp. 284-299, 1985.

[21] D.R. Musser, "Introspective Sorting and Selection Algorithms," *Software Practice and Experience,* vol. 27, no. 8, pp. 983-993, 1997.

[22] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms.* McGraw-Hill, MIT Press, 1990.

[23] R. Zabih and J. Woodfill, "Non-Parametric Local Transforms forComputing Visual Correspondence," *Proc. Third European Conf. Computer Vision,* pp. 151-158, 1990.

[24] T.D. Albright and R. Desimone, "Local Precision of Visuotopic Organization in the Middle Temporal Area (MT) of the Macaque," *Experimental Brain Research,* vol. 65, no. 3, pp. 582-592, 1987.

[25] E.P. Simoncelli, "Coarse-to-Fine Estimation of Visual Motion," *Proc. Eighth Workshop on Image and Multidimensional Signal Processing,* http://www.cns.nyu.edu/ftp/eero/simoncelli93d.pdf (08/2005), 1993.

[26] S.W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing.* Calif. Technical Publishing, 1997.

[27] P.J. Burt and E.H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Comm.,* vol. 31, no. 4, pp. 532-540, 1983.

[28] R. Desimone and J. Duncan, "Neural Mechanisms of Selective Visual Attention," *Ann. Rev. Neuroscience,* vol. 18, pp. 193-222, 1995.

[29] J.H.R. Maunsell and D.C. Van Essen, "Functional Properties of Neurons in the Middle Temporal Visual Area of the Macaque Monkey. I Selectivity for Stimulus Direction, Speed and Orientation," *J. Neurophysiology,* vol. 49, no. 5, pp. 1127-1147, 1983.

[30] S. Deneve, P.E. Latham, and A. Pouget, "Reading Population Codes: A Neural Implementation of Ideal Observers," *Nature Neuroscience,* vol. 2, no. 8, pp. 740-745, 1999.

[31] M. Nicolescu and G. Medioni, "Layered 4D Representation and Voting for Grouping from Motion," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 4, pp. 492-501, Apr. 2003.

[32] J.Y.A. Wang and E.H. Adelson, "Representing Moving Images with Layers," *IEEE Trans. Image Processing Special Issue: Image Sequence Compression,* vol. 3, no. 5, pp. 625-638, 1994.

[33] S.J. Nowlan and T.J. Sejnowski, "Filter Selection Model for Motion Segmentation and Velocity Integration," *Optical Soc. Am., A,* vol. 11, no. 12, pp. 3177-3200, 1994.

[34] S.J. Nowlan and T.J. Sejnowski, "A Selection Model for Motion Processing in Area MT of Primates," *J. Neuroscience,* vol. 15, no. 2, pp. 1195-1214, 1995.

[35] N. Qian, R.A. Andersen, and E.H. Adelson, "Transparent Motion Perception as Detection of Unbalanced Motion Signals, III. Modeling," *J. Neuroscience,* vol. 14, no. 12, pp. 7381-7392, 1994.

[36] P. Bayerl and H. Neumann, "Attention and Figure-Ground Segmentation in a Model Motion Perception," *Proc. Fifth Ann. Meeting of the Vision Science Soc (VSS '05),* abstract 661, 2005.

[37] W. Förstner, "A Feature Based Correspondence Algorithm for Image Matching," *Int'l Archieves in Photogrammetry and Remote Sensing,* vol. 24, pp. 150-166, 1986.

[38] L. Harris and M. Stephen, "A Combined Corner and Edge Detector," *Proc. Fourth Alvey Vision Conf.,* pp. 147-151, 1988.

[39] J.W. Harris and H. Stocker, "Maximum Likelihood Method," *Handbook of Math. and Computational Science,* Springer-Verlag, p. 824, 1998.

[40] M. Isard and A. Blake, "Condensation—Conditional Density Propagation for Visual Tracking," *Int'l J. Computer Vision,* vol. 29, no. 1, pp. 5-28, 1998.

[41] R.E. Kalman, "A new Approach to Linear Filtering and Prediction Problems," *Trans. ASME—J. Basic Eng.,* vol. 82 (Series D), pp. 35-45, 1960.

Pierre Bayerl received the MS degree (German Diploma) in 2001 and the doctoral degreee in 2005, both in computer science, from the University of Ulm, Germany. He is currently working as research and teaching assistant in the Department of Neural Information Processing at the University of Ulm. His research interests include computer vision and neural modeling, in particular, biologically motivated solutions of motion analysis, and experimental psychophysical investigations of the human visual system.

Heiko Neumann studied computer science at Technical University of Berlin and received a doctoral degree in computer science at the University of Hamburg in 1988. He received the Habilitation degree in 1995 and was admitted as professor of computer science in the Department of Neural Information Processing at Ulm University in 1995. While at the University of Hamburg, he was a member of the Graduate School of Cognitive Systems. He spent several research sabbaticals at the Center for Adaptive Systems, Department for Cognitive and Neural Systems, at Boston University. He is cofounder of the competence center for Perception and Interactive Technologies (PIT) at Ulm University. His research interests include neural modeling in computational and cognitive neuroscience, biologically inspired computational vision, and various aspects of early and midlevel computational vision. He is a fellow of the IEEE Computer Society.