Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Invariant Object Recognition and Pose Estimation with Slow Feature Analysis

Mathias Franzius, Niko Wilbert, Laurenz Wiskott

2011

Preprint:

This is an accepted article published in Neural Computation. The final authenticated version is available online at: https://doi.org/[DOI not available]

LETTER **Communicated by Peter König**

Invariant Object Recognition and Pose Estimation with Slow Feature Analysis

Mathias Franzius
Mathias.Franzius@honda-ri.de
Institute for Theoretical Biology, Humboldt-Universität zu Berlin, 10115 Berlin, Germany, and Honda Research Institute Europe GmbH, Offenbach-Main, 63073, Germany

Niko Wilbert
mail@nikowilbert.de
Institute for Theoretical Biology, Humboldt-Universität zu Berlin, 10115 Berlin, Germany

Laurenz Wiskott
laurenz.wiskott@neuroinformatik.ruhr-uni-bochum.de
Institute for Theoretical Biology, Humboldt-Universität zu Berlin, 10115 Berlin, Germany, and Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany

Primates are very good at recognizing objects independent of viewing angle or retinal position, and they outperform existing computer vision systems by far. But invariant object recognition is only one prerequisite for successful interaction with the environment. An animal also needs to assess an object's position and relative rotational angle. We propose here a model that is able to extract object identity, position, and rotation angles. We demonstrate the model behavior on complex three-dimensional objects under translation and rotation in depth on a homogeneous background. A similar model has previously been shown to extract hippocampal spatial codes from quasi-natural videos. The framework for mathematical analysis of this earlier application carries over to the scenario of invariant object recognition. Thus, the simulation results can be explained analytically even for the complex high-dimensional data we employed.

1 Introduction

Sensory signals convey information about the world surrounding an animal. However, a sensory signal can change dramatically even when only a single object is slightly moved or rotated. The visual signal from the retina, for example, varies strongly when distance, position, viewing angle, or

 © 2011 Massachusetts Institute of Technology

lighting conditions change. A high-level representation of object identity in the brain should, however, remain constant or invariant under these different conditions (Booth & Rolls, 1998). How could the brain extract this abstract information from the highly variable stimuli it perceives? Furthermore, as it is unlikely that the visual brain is completely determined by genetic factors, how can this task be learned from the statistics of the visual stimuli in an unsupervised way? Objective function learning has proven an effective tool to tackle these questions (Olshausen & Field, 1996; Wallis & Rolls, 1997; Becker, 1999), as it can provide a level of abstraction for describing functional properties of neural systems that is compact and intuitive and yet allows the direct derivation of learning algorithms.

Invariant object recognition is only one task a vision system has to achieve to enable successful interaction with the environment. We do not only need to extract the identity of an object ("What is seen?") independent of its position and viewing angle; we also want to extract the position of an object ("Where is it?") independent of its identity or viewing angle. The relative rotational angle of a seen object can be just as crucial ("Does the tiger face me or go away?"). In principle, we might want a representation of any aspect (e.g., size, viewing angle, lighting direction) independent of all the others, and optimally, all these tasks should be solved with a single computational principle. In the following, we refer to the set of object position and angles relative to the viewer as the configuration or pose of an object. We call a 2D image of an object in a specific configuration a view.

How can a brain decide which view belongs to which object? Primates can perform invariant object recognition successfully in most situations of everyday life and behave accordingly, which requires that the information about object identity is available for higher brain areas in a sufficiently explicit format. Since it is hard to answer which specific format is best for later stages of the brain, we just postulate that a simple classifier should be able to distinguish between the object representations after a few training examples and that a simple linear regression should be able to extract pose information. A special case of such a representation would, of course, be an explicit population code where one set of neurons codes for object identity and a different set codes for object pose. Such a representation will be demonstrated for a scenario with reduced complexity.

A good vision system should also generalize to previously unseen configurations, that is, it should learn the relevant statistics of transformations rather than just memorizing specific views. It should also generalize to new objects; for example, it should successfully estimate the position and orientation of an object that was never shown before.[1]

[1]Generally there is no canonical 0 degree view of an object; thus, a random phase offset of absolute phase for a new object is to be expected.

In general, the process of extracting the configuration of an object from a view is very hard to solve, especially in the presence of a cluttered background and many different possible objects. In this work, we use high-resolution views of complex objects but restrict the problem to the cases of only one object present at a time and a static homogeneous background. Due to the high variability of views caused by changing configuration, there is an infinitely large space of possible views of even a single object. The manifold of all possible views of an object is embedded in a highly complex way within the visual (pixel) space, and two views of distinct objects are often closer to each other in pixel space than two views of the same object (DiCarlo & Cox, 2007). One approach to solving such complex computational problems is to break them up into a series of simpler computations. Functionally, each stage or layer should represent views belonging to a single object more compactly and views of distinct objects more separately, thus untangling the view manifolds. Furthermore, the dimensionality of the object representation should be sufficiently low on the highest layer such that object-specific behavior (or, in the simplest case, just an object classifier) needs only a few labeled training examples. Both properties, modularized computation and dimensionality reduction, can be implemented by a converging visual hierarchy.

Our hierarchical model is based on unsupervised learning with a global objective function (Becker & Zemel, 2003), specifically the principle of slowness. This principle has been applied for object recognition before (Becker, 1999; Stringer and Rolls, 2002; Einhäuser, Hipp, Eggert, Körner, König, 2005), and there is recent evidence for slowness-based adaptation in the inferior temporal cortex (Li & DiCarlo, 2008, 2010). However, our model goes beyond these earlier ones by not only extracting translation-invariant and view-invariant representations of object identity but also simultaneously information about position and viewing angles. Furthermore, we show meaningful generalization of the model to previously unseen objects. The structure of the resulting representation depends solely on the statistics of presentation of the training views. We show that in a restricted scenario, these representations are encoded independent of each other in an analytically predictable way. In a more complex scenario, the solutions tend to mix but are still very simple to decode by linear regression.

To optimize the slowness-based objective function, we use the slow feature analysis (SFA) algorithm (see section 2.2). Except for minor changes, the model used here is identical to that used earlier for the modeling of place cells, head direction cells, and spatial view cells in the hippocampal formation (Franzius, Sprekeler, & Wiskott, 2007). The complete mathematical framework of this publication carries over to the problem of invariant object recognition as presented here.

Our model is able to learn both view-invariant object-specific representations and position and rotational information from complex

high-dimensional data in a biologically plausible model of the ventral visual system. It has been used to extract information from quasi-natural stimuli leading to a large variety of behaviorally relevant representations (e.g., allocentric position and head direction, viewing direction, object identity, object position, and object angles). We speculate that slowness might serve as a general principle for sensory coding in the brain.

2 Methods

2.1 Stimuli. The model was trained and tested in distinct experiments with image sequences from three stimulus classes. For two of these classes, OpenGL was used to render the views of the objects as textured 3D models in front of a white homogeneous background. We initially used colored stimuli and found color to be a very strong cue, especially for object classification. In order to prevent the model from using this simple cue, we used only gray-scale views for the results presented here. The two rendered object classes are described below. For each object class, an instance of the model was trained with a number of different objects. In the testing phase, we also added new objects that the model had never seen during training. The complete set of models, example image sequences, and results can be viewed on the supplemental Web site (Franzius, Wilbert, & Wiskott, 2009). The stimulus set in the form of source code or data is available on request. Note that in contrast to many other models, the model was trained with image sequences consisting of tens of thousands of distinct views (see section 2.3.2).

We introduced the two dissimilar fish and sphere cluster stimulus sets to show that our results do not critically depend on one particular kind of stimulus. Using new stimulus sets instead of established ones like the COIL100 database (Nayar, Nene, & Murase, 1996) makes it difficult to compare the results with those of other models. Unfortunately, no freely available databases could give us a comparable amount of control over the object pose, which is critical for the evaluation of our model. Our implementation allows us controlled simultaneous rotation around multiple axes and specifically allows control over the temporal presentation order.

The HRI50 database (Kirstein, Wersing, & Körner, 2005, 2008) was used as a third stimulus class. It consists of recorded real-world views of 50 household objects freely rotated in hand. Views were rescaled to the size of the rendered images to be compatible with the model architecture. Because objects in the HRI50 database were presented with a back glove in front of dark background, no further preprocessing (e.g., segmentation) was applied. Object recognition results on this database are presented in section 3.4 to prove the efficacy of our model on real-world data. We cannot, however, quantify pose estimation on this database, as there are no ground truth pose data available for this database.

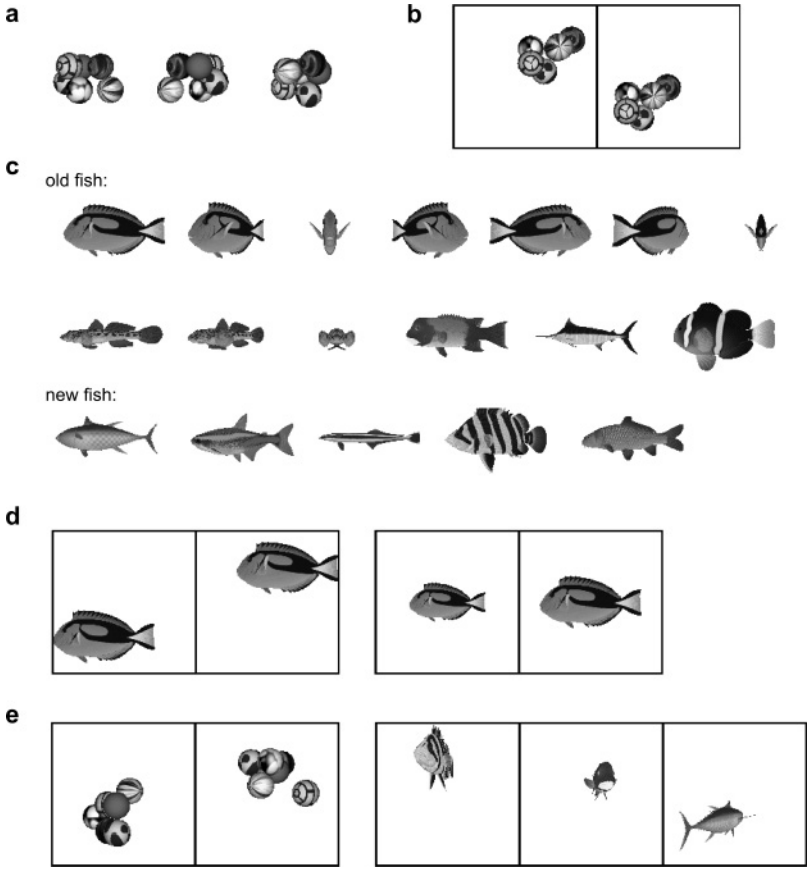

Figure 1: Objects used for stimulus generation. (a) The sphere cluster objects (each cluster of six spheres is one object). The first two views show the same object under different in-depth rotation angles, and the third view shows a different object. (b) The top right and bottom left corner positions for a single object to illustrate the translation range. (c) Some of the fish objects used for training and testing are shown with examples for the effect of in-depth rotation. The first row illustrates a rotation from 0 to 270 degrees. (d) The corner positions for a single fish and the smallest and largest scale. (e) Some of the images used for training and testing.

2.1.1 Stimulus Classes. In the first experiment, the objects were clusters of textured spheres as shown in Figure 1a, which provide a difficult but generic task. The same six textured spheres were used in different spatial arrangements for all the objects. For each object, the spheres were randomly fixed on a lattice (hexagonal close packing) with three layers of size 2×2.

Five such objects were used for training, and five additional were added for testing. As the examples in Figure 1a illustrate, identifying the rotation angles and identities for such objects is quite difficult even for human observers.

Using the same building blocks for all objects should force the model into using high-level configural features for object classification since the objects have the same low-level features. On the other hand, the common low-level features of the objects help the model generalize to untrained objects. Using spheres has the advantage that the outline does not give a simple clue for the in-plane rotation angle.

In the second experiment, models of different fish (see Figure 1b) were used to provide more natural stimuli from a single object class (all models taken from Toucan Corporation, 2005, with permission). We used 15 different fish for training and added 10 more for testing, raising the number of different objects to 25.

2.1.2 Object Configurations. For sphere cluster objects, the horizontal x-coordinate, the vertical y-coordinate, the in-depth rotation angle ϕ_y, and the in-plane rotation angle ϕ_z were chosen as configuration variables. x and y range from 0 to 1, ϕ_y and ϕ_z from 0 to 360 degrees. Another configuration variable was the object identity ranging from 1 to 10, with objects 1 to 5 being the training objects. So the transformations the system had to learn consisted of translations in the plane, rotations along the y- and z-axes (with the in-depth rotation coming first) and changes of object identity. The field of view used in the rendering of the sphere cluster objects was 10 degrees, so the views look almost like an orthographic projection.

For the fish objects, the configuration variables were x, y, ϕ_y, object identity, and size. Compared to the sphere cluster objects, we added changes in size and removed the in-plane rotations. The maximum size of the objects was 33% larger than the minimum size in each direction (see Figure 1d). A greater range of scale variations would have required larger images as well, since even for the smallest scale and difficult angles, all configuration variables should be extractable. Introducing scale transformations is also an additional safeguard against the model relying purely on the size of a fish for classification. The field of view was set to 45 degrees, leading to significant perspective projection distortions. This results in an additional position dependency of the fish object views, which is clearly visible in Figure 1e. A size or image area normalization of the different fish objects is not possible due to the configuration dependency of the effective object area in the image (which varies most for the in-depth rotation).

2.1.3 Random Walk Procedure for Generating Training Sequences. The configurations for the training sequences were generated by a random walk procedure. To generate a configuration in the sequence, we add a random term to the current spatial, angular, and scaling velocities of the object. The

Table 1: Parameters for the Random Walk Procedure.

	Spheres			Fish		
	Max. Vel.	Max. Acc.	Δ	Max. Vel.	Max. Acc.	Δ
x (in [0; 1])	0.025	0.005	0.008	0.03	0.005	0.008
y (in [0; 1])	0.025	0.005	0.008	0.03	0.005	0.008
size (in [0.75; 1])				0.005	0.001	0.014
ϕ_z in-plane)	0.04	0.01	0.007			
ϕ_y rad (in-depth)	0.04	0.01	0.007	0.04	0.01	0.007

Notes: All velocity values are given "per frame" and the angle values are in radiant measure. The "max. vel." parameter is the maximal velocity, "max. acc." is the maximal change in velocity per frame (i.e., acceleration). The Δ's are rounded empirical values from the actual image sequences.

random term is drawn from an interval with a uniform probability density and zero mean. The velocities are cut off at certain limits, and by adjusting these limits, one can effectively determine the transformation timescales. The position, angles, and scale are then updated according to these velocities. If an object reaches the position boundary, it is bounced back. All the parameter values are given in Table 1. The overall procedure yields flat configuration histograms (given enough time points), and the velocity profiles are independent of the configuration values.

We illustrate this procedure with an example for the x-position of a sphere cluster object. The allowed interval for these position values is $[0, 1]$. Assume that the current position is 0.99 and that the current velocity (change in position per frame) is 0.022, which is still below the maximum value of 0.025. The velocity update is now randomly picked from $[-0.005, 0.005]$, for example, 0.004. This brings the velocity value to 0.026, so it is cut off at 0.025. The new position would therefore be 1.015, but since this is beyond the boundary, the object bounces back, and the final new position is 0.985. To get a visual impression of the time structure we provide example sequences on the supplemental Web site (Franzius et al., 2009).

In each step, the object identity was changed with low probability ($p = 0.002$). A blank frame was inserted if a switch took place to avoid linking together different objects in identical poses in the stimulus, which would introduce an element of supervised training. The effect of the blank frame on the coding of the configuration variables is described in section 2.2.

Note that the timescales of the different configuration variables of the stimuli will resurface in the SFA output of our model. The SFA algorithm does not change the timescales of the features; these timescales are purely determined by the parameters of the random walk (see section 2.2.1).

2.2 Slow Feature Analysis. Slow feature analysis (SFA) solves the following learning task. Given a multidimensional input signal, we want to find instantaneous scalar input-output functions that generate output

signals that vary as slowly as possible but still carry significant information. To ensure the latter, we require the output signals to be uncorrelated and have unit variance. In mathematical terms, this can be stated as follows:

Optimization problem. Given a function space $\mathcal{F}$ and an I-dimensional input signal $\mathbf{x}(t)$ find a set of J real-valued input-output functions $g_j(\mathbf{x}) \in \mathcal{F}$ such that the output signals $y_j(t) := g_j(\mathbf{x}(t))$

$$\text{minimize } \Delta(y_j) := \langle \dot{y}_j^2 \rangle_t \tag{2.1}$$

under the constraints

$$\langle y_j \rangle_t = 0 \quad \text{(zero mean)}, \tag{2.2}$$

$$\langle y_j^2 \rangle_t = 1 \quad \text{(unit variance)}, \tag{2.3}$$

$$\forall i < j : \langle y_i y_j \rangle_t = 0 \quad \text{(decorrelation and order)}, \tag{2.4}$$

with $\langle \cdot \rangle_t$ and $\dot{y}$ indicating temporal averaging and the derivative of y, respectively.

Equation 2.1 introduces the Δ-value, a measure of the temporal slowness (or rather fastness) of the signal $y(t)$. It is given by the mean square of the signal's temporal derivative, so that small Δ-values indicate slowly varying signals. Constraints 2.2 and 2.3 avoid the trivial constant solution, and constraint 2.4 ensures that different functions g_j code for different aspects of the input. Because of constraint 2.4, the g_j are also ordered according to their slowness, with g_1 having the smallest Δ.

It is important to note that the temporal derivatives of the data are used only during the training phase. The resulting functions g_j are instantaneous functions of the input, and thus slowness cannot be achieved by low-pass filtering. Slow output signals can be obtained only if the input signal contains slowly varying features that can be extracted instantaneously by the functions g_j. Note also that for the same reason, once trained, the system works quickly, not slowly.

In the computationally relevant case where $\mathcal{F}$ is finite-dimensional, the solution to the optimization problem can be found by means of SFA (Wiskott, 1998; Wiskott & Sejnowski, 2002) and in a slightly different formulation in Berkes and Wiskott (2005). This algorithm, which is based on an eigenvector approach, is guaranteed to find the global optimum. We use the SFA implementation in the open source MDP library (Modular toolkit for Data Processing) (Zito, Wilbert, Wiskott, & Berkes, 2008). Biologically more plausible learning rules for the optimization problem, for both graded response and spiking units, exist (Hashimoto, 2003; Sprekeler, Michaelis, & Wiskott, 2007).

If the function space is infinite-dimensional, the problem requires variational calculus and is in general difficult to solve. Franzius et al. (2007) showed that the optimization problem for the high-dimensional visual input, as faced by the hierarchical model, can be reformulated for the low-dimensional configuration input (position, orientation, and object identity). In this case, the variational calculus approach becomes tractable and allows making analytical predictions for the behavior of the full model.

2.2.1 Theoretical Predictions. The theoretically derived optimal solutions for the infinite-dimensional function space provide very useful predictions for the SFA output of our model (even though it is based on the finite-dimensional SFA algorithm). Therefore, we now briefly describe these predictions. Detailed derivations were presented in Wiskott (2003) and Franzius et al. (2007) but are too lengthy to be replicated here. (The results in the second paper were derived by Sprekeler, 2009.)

The optimal solution for the slowness optimization problem is generally a function of the slowest configuration variable. For example, let us assume that $c_1(t)$ (e.g., the position x or y in section 2.1) is the configuration variable that varies on the slowest timescale (i.e., has the smallest Δ-value after normalization to unit variance). The configuration values are mapped to the input signal $\mathbf{x}(t)$, which corresponds to the raw pixel data in our model. Further assume that the values of c_1 are homogeneously distributed in the interval $[0; L]$ (with $L > 0$) and that the velocity profile does not depend on the value of c_1. As shown in Franzius et al. (2007), the slowest possible output signal $y_1(t) := g_1(\mathbf{x}(t))$ is then given by

$$g_1(\mathbf{x}(t)) = \sqrt{2}\cos\left(\pi\frac{c_1(t)}{L}\right). \tag{2.5}$$

So g_1 is a half cosine over the c_1-interval, that is, a monotonically increasing representation of c_1-position, which is invariant to all other configuration variables. If the timescale of c_2 is only slightly faster than that of c_1, and both are uncorrelated, then the second slowest solution $g_2(\mathbf{x}(t))$ is a similar half cosine of c_2. However, the decorrelation condition 2.4 is also satisfied by solutions of the form

$$g(\mathbf{x}(t)) = \sqrt{2}\cos\left(\pi\frac{nc_1(t)}{L}\right) \quad \text{with } 1 < n \in \mathbb{N}. \tag{2.6}$$

These higher-order harmonics are all uncorrelated, with their timescales being multiples of the original timescale. Similar harmonics exist for c_2. Another class of solutions consists of products of the solutions in c_1 and c_2 mentioned above. If the configuration variables are uncorrelated, then

the products of their solutions will be uncorrelated as well. All of these solutions are naturally ordered by their timescales (i.e., their Δ-value).

If two variables vary on equal timescales, then any normalized linear combination of their solutions will have the same Δ-value as well. Thus, any two uncorrelated solutions from the two-dimensional subspace spanned by the solutions of the individual variables form a valid set of solutions. We refer to this ambiguity as the problem of linear mixing and discuss its practical implications in section 2.4. Depending on their timescales, higher-order harmonics or products of solutions can also take part in linear mixtures.

Two other special cases are relevant for our model: cyclic configuration variables like ϕ_y or ϕ_z and variables coding for object identity. Cyclic variables arise if the input data $\mathbf{x}(t)$ consist of periodic functions of a hidden variable $\phi(t) \in [0; 2\pi]$,

$$\mathbf{x}(t) = \mathbf{x}(\phi, \ldots) = \mathbf{x}(\phi + 2\pi, \ldots). \tag{2.7}$$

Again we assume a uniform distribution of ϕ and an independent velocity profile. The optimal uncorrelated solutions based on the periodic $\mathbf{x}(t)$ are then

$$g_1(\mathbf{x}(t)) = \sqrt{2}\sin(\phi(t) + \phi_0) \quad \text{and} \quad g_2(\mathbf{x}(t)) = \sqrt{2}\cos(\phi(t) + \phi_0), \tag{2.8}$$

with an arbitrary phase offset ϕ_0. Because g_1 and g_2 vary on the same timescale, ϕ_0 simply reflects the linear mixing ambiguity.

The second special case is a variable that takes only discrete values, like the object identity $k(t) \in \{1, 2, \ldots, N\}$.[2] If the transitions between the values are all equal (e.g., switching from object 1 to object 3 is just as likely as switching to object 2), then there are $N - 1$ uncorrelated optimal solutions. Those solutions are different step functions of k (Berkes, 2005). Their exact shapes are determined by the decorrelation conditions.

We insert a blank frame whenever the object identity changes. Since SFA is instantaneous and deterministic, the SFA outputs always take the same value for this blank frame. Therefore, there is no penalty in the objective function if the SFA outputs become object dependent. It allows the SFA solutions for variables like position and angles to become object dependent without increasing their Δ-value (i.e., reducing their slowness). Therefore, the dimension of the space of Δ-optimal solutions increases for all variables apart from the identity. A basis of slowest solutions is now given by functions that are of the form 2.5 or 2.8 for a single object and are zero for

[2]When using a continuous time t, this implies nondifferentiable changes of the variable, leading to infinities in the analytical analysis. For practical applications of the SFA algorithm, this does not pose a problem, since the time t has to be discrete in any case.

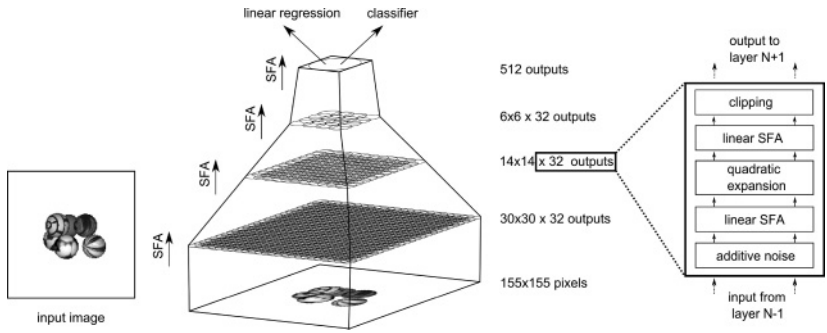

Figure 2: Model architecture and stimuli. An input image is fed into the hierarchical network. The circles in each layer denote the overlapping receptive fields and converge toward the top layer. The same set of steps is applied on each layer, which is visualized on the right-hand side (this is also called a node). For example there are 14 times 14 nodes in the second layer, and each produces a 32-dimensional output.

all others. So for one particular variable and N objects, there are now N uncorrelated optimal solutions instead of just one.

These optimal solutions assume an infinite-dimensional function space, whereas SFA itself is linear. The next section describes how a sufficiently large function space can be used for approximating the optimal solutions while keeping computational costs at a practical level.

2.3 Hierarchical Network Model. The computational model consists of a converging hierarchy of layers of SFA nodes (see Figure 2). Each SFA node finds the slowest features from its input according to the SFA algorithm and basically performs the following sequence of operations: linear SFA for dimensionality reduction, quadratic expansion, and another linear SFA step for slow-feature extraction. The network is implemented in Python, and all required elements (including the parallelization) are available in the MDP library (Zito et al., 2008).

2.3.1 Network Structure. The detailed network structure is shown in Figure 2. We first describe the network connectivity and then the internal structure of the SFA nodes.

The part of the input image that influences a node's output will be denoted as its receptive field. On the lowest layer, the receptive field of each node consists of an image patch of 10×10 gray-scale pixels (so the node input has a dimensionality of 100). The receptive fields jointly cover the input image of 155×155 pixels. The nodes form a regular (i.e., nonfoveated) 30×30 grid with partially overlapping receptive fields, resulting in an overlap of five pixels in each direction. The second layer contains 14×14

nodes, each receiving input from 4×4 layer 1 nodes with neighboring receptive fields, resembling a retinotopic layout (the overlap is two nodes in each direction). The third layer contains 6×6 nodes, each receiving input from 4×4 layer 2 nodes with neighboring receptive fields, again in a retinotopic layout (with 2 nodes overlapping in each direction). All 6×6 layer 3 outputs converge onto a single node in layer 4, whose output we call SFA output. The output of each node consists of the 32 slowest outputs, except for the top layer, where 512 dimensions are used. Thus, the hierarchical organization of the model captures two important aspects of cortical visual processing: increasing receptive field sizes and accumulating computational power at higher layers. The latter is due to the quadratic expansion in each layer, so that each layer computes a subset of higher polynomials than its predecessor. The SFA outputs in the top layer compute subsets of polynomials of degree $2^4 = 16$.

The first linear SFA stage in each node (see Figure 2) reduces the dimensionality to 32 in the first two layers, 42 in the third layer, and 52 in the fourth layer.[3] The quadratic expansion then increases dimensionality to 560, 945, and 1430, respectively.

Quadratic expansion means that the incoming data $x_1, \ldots, x_n$ are mapped with a basis of the space of polynomials with degree up to two. So in addition to the original data, all quadratic combinations like $(x_1)^2$ or $x_1 x_2$ are concatenated to the data block. The solutions of linear SFA on this expanded data are equivalent to those of SFA in the space of polynomials up to degree two.

The second linear SFA stage in a node reduces the dimensionality of the expanded signal to 32, except for the top layer, where the output is reduced to 512 dimensions. Compared to the number of features to be extracted, this number might seem high, but as described in section 2.2, we expect multiple solutions for each continuous configuration variable (depending on the number of training objects). We come back to this issue in later sections.

Additive gaussian white noise (with a variance of 10^{-6}) is introduced for numerical reasons to avoid possible singularities in the SFA step. After the second SFA stage, we apply a clipping at ± 4. This clipping removes extreme values that can occur on test data due to the divergence of the quadratic functions for larger values. However, both the additive noise and the clipping are just technical safeguards. As we verified in control experiments (see section 3.3), they typically have no effect on the network performance.

The choice of network parameters is mostly guided by computational constraints. For example, the output dimension of the first SFA step in

[3] The increase in dimensionality across layers leads to a small performance increase. A further increase of the dimensions is possible, but it has very little effect.

each layer must remain limited, since otherwise the dimensionality after the quadratic expansion explodes. On the other hand, the dimensionality reduction must preserve enough relevant information. In our experience, the network performance is robust to a wide range of parameter choices, so a reasonable ad hoc guess of a good parameter set should often be sufficient. Fine-tuning the parameters is mostly useful to optimize the trade-off between performance and computational cost; it is not necessary to achieve a reasonable level of performance.

2.3.2 Network Training. The layers are trained sequentially from bottom to top. For computational efficiency, we train only one node with stimuli from all node locations in its layer and replicate this node throughout the layer. For example, this means that the node in the lowest layer sees $30 \times 30 = 900$ times as many data as if it was trained at only a single location. This mechanism effectively implements a weight-sharing constraint. However, the system performance does not depend on this mechanism. The statistics of the training data are approximately identical for all receptive fields, so individually learned nodes would lead to the same results (but at higher computational cost). Individually learned nodes would even improve the overall performance in cases where the statistics vary across the visual field (e.g., due to a horizon in the images). While the weight sharing does ease the emergence of translation invariance, it is not at all sufficient.

For our simulations, we used 50,000 views for training of the two lower layers and 200,000 for the two upper layers. The random walk parameters of the training data were identical for all layers. The larger training set for the upper layers is motivated by the smaller multiplicative effect of the weight sharing. Since the training time of the entire model on a single PC is on the order of multiple days, the implementation is parallelized and training times thus reduced to hours. The simulated views are generated from their configuration (position, angles, and object identity) with floating-point precision and are not artificially discretized to a smaller configuration set. Therefore, it is highly unlikely that any two images in the training set were identical.

2.4 Feature Extraction with Linear Regression. In section 2.2 we presented the form of the expected SFA output and the problem of linear mixing of solutions for variables with identical timescales. The limited sampling of the training sequence and the function space used by the model generally leads to deviations from the theoretical predictions. Therefore a mixing in the SFA output can occur even if the underlying timescales are different. The higher-order harmonics are also affected by this. Overall this makes it practically impossible to prevent the mixing for complex applications with many different transformations as in section 3.2, which complicates the interpretation of the SFA output.

Our main objective here is to show that pose and identity information is indeed extracted by the learned slow features from the raw image data even if linearly mixed. The easiest way to do this is by calculating a multivariate linear regression of the SFA output against the known configuration values. This gives us a projection vector for each configuration variable (plus a constant offset). To estimate the value of a given configuration variable from a given output, one takes the scalar product of the projection vector with the SFA output vector and adds the offset value.

While the regression procedure is obviously supervised due to the use of the reference configuration values, it nevertheless shows that the relevant signals are easily accessible. Extracting this information from the raw image data linearly is not possible, as shown in section 3.3. One should also note that the dimensionality of the SFA output is smaller than the raw image data by two orders of magnitude.

For technical convenience, the configuration reference values were binned, which, for example, simplified the calculation of error bars. For each configuration variable (apart from object identity), 60 bins were used, which is small enough to not influence the results (tests with smaller bin sizes did not show any significant differences). Since the predicted SFA solutions are cosine functions of the position values (see section 2.2), one has to map the reference values correspondingly before calculating the regression. The results from the regression are then mapped with the inverse function. For example, one does not calculate the regression for position x, but for the predicted solution $y(x) = \cos(\pi x)$ instead. The result from the linear regression is then mapped with $\arccos(y)/\pi$ (after cutting off the values outside the interval $[-1; 1]$) to get the estimated position values.

For SFA solutions that code for rotational angles, the theory predicts both sine and cosine functions of the angle value (as described in section 2.2). Therefore, we calculated regressions for both mappings and then calculated the angles via the arctangent. This automatically matches the phase of the extracted angles to that of the reference values. If multiple objects are trained and separated with a blank, the solutions will in general be object dependent (see section 2.2). In theory, a global regression for all objects should work nevertheless, since the different solutions can be linearly combined into an object-independent solution (i.e., it lies in the space spanned by the solutions). However, for the complicated stimuli used here, this does not work. The limited function space available to the model leads to more complicated dependencies on object identity. The easiest solution to this is to perform an individual regression for every object. While this procedure sacrifices the object invariance of angle information, it does not affect the invariance under all other transformations.

As described in section 2.2, the object identity of N different objects is optimally encoded (under the SFA objective) by up to $N - 1$ uncorrelated step functions, which are invariant under all other transformations. In the SFA output, this should lead to separated clusters for the trained objects.

For untrained objects, those SFA outputs coding for object identity should take new values, which are again invariant under all other transformations. Thus, it should be possible to separate old and new objects. Linear mixing of the identity solutions with other features is to be expected.

To explore the potential classification ability of the model, we applied two very simple classifiers on the SFA output: a k-nearest-neighbor and a gaussian classifier. Since the linear mixing does not affect cluster separation, it should not affect the classifiers. The classification performance is affected by imperfect transformation invariance of the identity solutions (i.e., dependencies beyond the linear mixing). This can cause overlap between object clusters. Clusters also have to intersect if two distinct objects share an identical or highly similar view. Note that the dimensionality of the regression input also has to be matched with enough reference points to capture the shapes of the clusters.

3 Results

First, we consider the raw SFA output for an experiment with a reduced transformation set and clearly separated timescales (see section 2.2.1). Then the experiments with a larger transformation set and similar timescales are analyzed with the methods already described. The full analysis reports with detailed results are also available as online supplements (Franzius et al., 2009).

3.1 Reduced Transformation Set. To illustrate the SFA outputs of our model, we first present a simplified example, with fewer transformations and well-separated timescales. This should lead to SFA outputs that code only for one specific configuration variable per output and are invariant under all other transformations. Two sphere cluster objects with translation and in-plane rotation were used for this example. Both objects were already used for training the model. The two lowest layers were, as usual, trained with similar timescales for all transformations. In the training data of the two highest layers, very fast translations were used. Identity was the slowest feature and in-depth rotation lay in between. This manipulation of the timescales was done only in this educational example. By effectively discarding the position feature, we improve the clarity of the other features.

As predicted in section 2.2, the slowest output channel codes for object identity (see Figure 3). The output value is almost completely invariant under both rotation and translation. Outputs 2 to 5 are the model outputs coding for rotation angle. They nicely fit the theoretical prediction (i.e., the outputs are a basis for the predicted space of the slowest solutions). As expected, we get $4 = 2 \times 2$ uncorrelated solutions for the angle, since we trained with two different objects.

In summary, for cases of clearly separated timescales and few simultaneous decorrelated transformations as presented in this section, the model

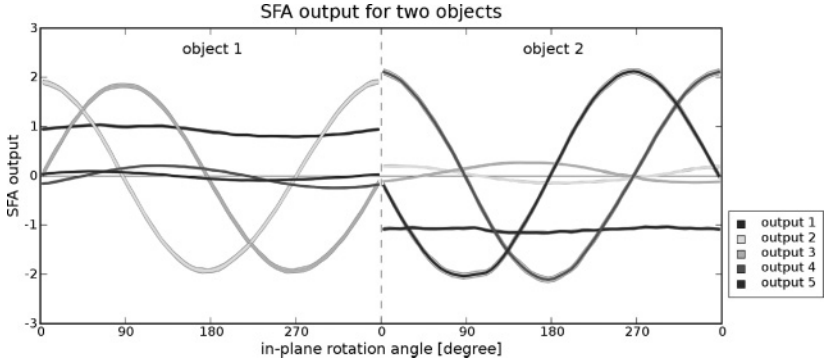

Figure 3: Five slowest SFA outputs for the simulation with a reduced transformation set. The in-plane angle dependence of the five slowest SFA outputs is shown for the two trained sphere cluster objects. The centers of the lines indicate the mean values, and the outer line borders represent ± 1 standard deviation or root mean square error (RMSE). Note that for each rotational angle, the objects were shown at many different positions, so the small RMSE illustrates the position invariance of the outputs. The slowest output (output 1) codes for object identity. The other four outputs code for the sine and cosine of the angle. Both the amplitudes and the phases of the solutions are object dependent, as described in section 2.2.

outputs are clearly predictable by variational calculus methods and directly encode the configuration parameters in an invariant manner.

3.2 Full Transformation Set. The results in this section are based on the full transformation set (as described in section 2.1), and the SFA outputs are subjected to supervised postprocessing (see section 2.4). The stimulus sets used for testing consisted of 100,000 views in total for the sphere cluster objects (covering old and new objects) and 100,000 views for the fish objects. Both were generated in the same way as the training data and were then shown to a hierarchical network that was trained with the same object type (as described in section 2.3). So each stimulus type (fish or sphere cluster objects) has its own hierarchical network.

We refer to objects that were shown during the training phase of the SFA layers in the hierarchical network as old objects, while the additional objects that were only shown in the testing phase of the hierarchical network are called new objects. There are 15 old and 10 new fish objects and 5 old and 5 new sphere cluster objects. For the postprocessing step, the 100,000 data points were divided into training and test data. From now on, the terms *training* and *test data* will therefore refer to the postprocessing, even though both are test data from the point of view of the fully trained hierarchical SFA network.

3.2.1 Position and Rotation Angles. To extract the x and y object coordinates from the model SFA output, we used multivariate linear regression as described in section 2.4. Half of the data (the training data) were used to calculate the regressions and the other half for testing. As one can see in Table 2 and Figure 4, this works reasonably well given the large size of the objects (which can reach 70% of the image; see Figure 1). For the sphere cluster objects, we get a root mean square error (RMSE) of 7% (relative to the position range) for old objects and around 9% for new objects. For the old and new fish objects, the error in the x-direction increases to 9% and 12%, respectively, due to the large object size in this direction.

To extract the in-plane and in-depth rotation angles for the sphere cluster objects, an individual regression for each object was calculated. This still requires invariance of the representations under the other transformations (including the other rotation type). As the results in Table 2 show, the in-plane angles were extracted with a mean RMSE of about 9 degrees the in-depth angles with about 11 degrees. The RMSEs for individual objects vary, ranging from 9 to 17 degrees for the in-depth rotation. We also verified that calculating the angles with global regressions for all objects did not work (results for new objects were at chance level).

For the fish, the mean RMSE for the in-depth angles is about 17 degrees for all 25 objects. A large part of this increase compared to the sphere cluster objects is due to systematic errors. The fish models naturally have a very similar front and back view, and therefore the model has difficulty differentiating between those two views, which can be clearly seen in Figure 5a. To verify this, we did the same analysis after removing all data points within 30 degrees of the front and back views. This brought the error down to the level of the sphere cluster objects (see Table 2).

As for the angles, individual regressions for all objects were used to calculate the size for the fish objects (size transformations were not used for the sphere cluster objects). This works with a RMSE of about 12% on average (see Figure 5b). The model cannot simply use the object area to infer the size, since the area covered by an object largely depends on the in-depth rotation angle and the object identity.

3.2.2 Classification. To quantify the classification ability of the model, three different classifiers were used on the SFA output. The classifiers were trained with about 5000 data points for each sphere object and 2000 data points for each fish object (half of the data, as for the regressions). The other half of the data points was then used to test the classifier performance. The gaussian classifier performed with about 95% hit rate for all 25 fish objects (see Table 2) and 98% for all 10 sphere objects. Training and testing with only the old or only the new sphere objects resulted in a performance of 99.9% (old) and 99.0% (new). The k-nearest-neighbor classifier ($k = 5$) worked at about the same performance level. We also included a simple nearest-cluster-center classifier, which performed much worse (66% on all spheres,

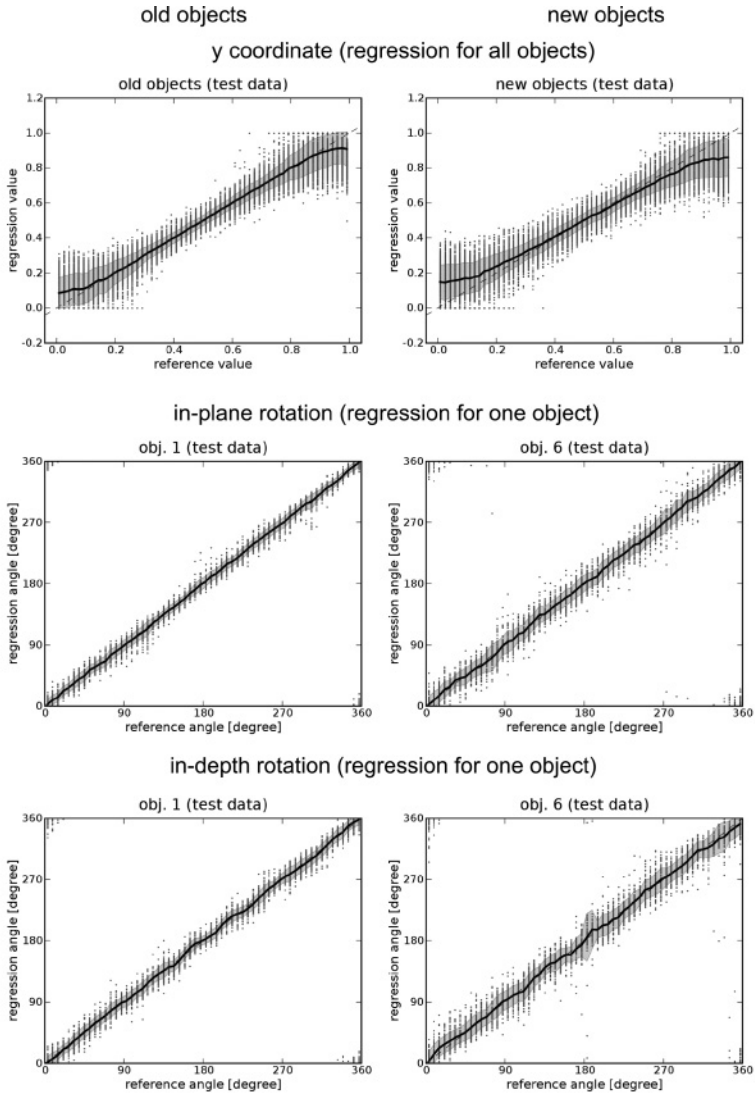

Figure 4: Reconstruction of sphere cluster object position and angle on test data. The variable values that were calculated with linear regression are plotted against the correct reference values. The gray dots are data points, the black line is the mean, and the gray area shows ± 1 RMSE. The regression of the y-coordinate was based on all five training objects. For the rotational angles, we show object-specific regressions. Object 1 was used during training of the SFA layers; object 6 was not.

Table 2: Results for Sphere Cluster and Fish Objects.

	Spheres			Fish		
	Old	New	All	Old	New	All
x	7%	8%	7%	9%	12%	10%
y	7%	9%	8%	7%	7%	7%
size				13%	12%	12%
ϕ_z (in-plane)	9°	9°	9°			
ϕ_y (in-depth)	10°	13°	11°	15°	19°	17°
ϕ_y no front/back				9°	11°	10°
Gaussian classifier	99.9%	99.0%	98.4%	96.4%	97.8%	95.2%
KNN classifier	99.2%	99.7%	99.0%	97.2%	97.5%	95.2%
NCC classifier	89.9%	70.2%	65.9%	83.4%	77.8%	74.0%

Notes: The first part shows the RMSE of x- and y-coordinate given in percent relative to the coordinate range (chance level is 28.9%). In the second part of the table we show the mean RMSE for the individual object regressions. The mean RMSE for the object size is given relative to the size value range (only fish). The next rows show the mean RMSE for the rotational angles (chance level is 104 degrees). The last row in the second part was calculated by omitting the ambiguous front and back views of the fish objects. In the third part of the table the classifier hit rates in percent are shown. The second row refers to the k-nearest-neighbor classifier ($k = 5$); the last row is the nearest cluster center classifier. Chance level is 10% for all sphere cluster objects and 4% for all fish objects.

74% on all fish). In Figure 6, some 2D projections of the data clusters are shown. They indicate that the data clusters for different objects are reasonably well separated (as shown by the classifier results). The classification errors for the fish objects mostly occur around the difficult front or back views. Removing all data points within 30 degrees of the front and back views raised the hit rate of the gaussian classifier to 99.8% on all 25 fish objects.

We also tested the ability of the model to classify based on limited training data for the postprocessing step (the hierarchical network was left unchanged). The fish objects were used for these tests, and training data were restricted to views where the fish were in the upper left image area. The in-depth rotation angles were restricted to the interval between 0 and 90 degrees. Testing of the classifiers was then done with the fish object in the opposite area and with angles from 120 to 330 degrees (i.e., a distance of at least 30 degrees to the known object views). Since the number of training data points was below the output dimensionality of 512, we used Fisher discriminant analysis (based on the full data set for the old objects only) to project the data down to 14 dimensions. This number matches the theoretically predicted uncorrelated SFA solutions for object identity (as described in section 2.2). The results are shown in Table 3.

For the first test only, a single randomly picked view per object was used to train the classifier (so the k of the k-nearest-neighbor classifier had to be set to one as well). This resulted in a hit rate on the 10 old objects of

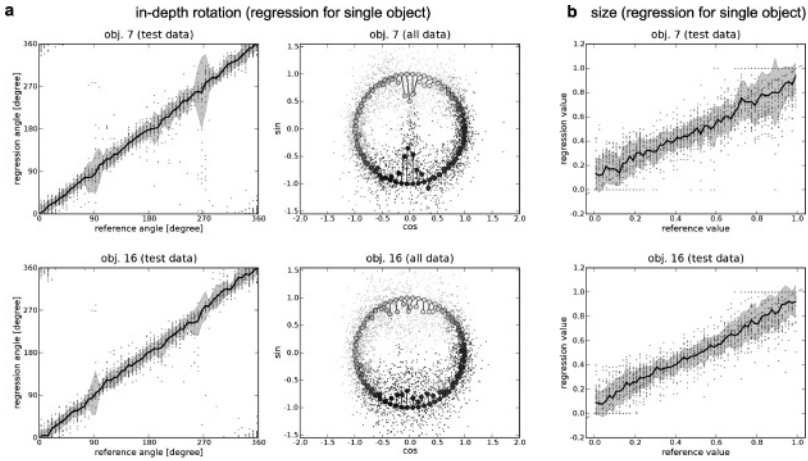

Figure 5: Reconstruction of in-depth rotation angle and size for fish objects on test data. (a) Regression results for the in-depth rotation angle for one fish used during training of the SFA layers (object 7, last one in second row in Figure 1c) and one new object (object 16, first one in third row of Figure 1c). The first plots in the top row show the regression angle versus the correct reference angle. The plots at the bottom show the regression values for the sine and cosine of the in-depth rotation angle. The shades of the points indicate the correct reference angle value. The mean regression values for the 60 different reference angle bins are shown as larger dots connected with a line to ideal values on the circle, which are also indicated by a large dot. The deviations at 90 and 270 degrees for object 7 are a result of the similar front and back views of this fish model. Object 7 was selected to illustrate this effect, which is less of a problem for many of the other objects (including object 16 shown on the right). (b) Regression for the size of a single object.

79.9%. For the new objects, this decreased to 57.9% and to 60.5% for all 25 fish objects (still well above the 4% chance level). Increasing the number of samples increased the performance, up to 79.0% on all 25 objects for the k-nearest-neighbor classifier ($k = 3$). Note that the training and test data were still drawn from the nonoverlapping parameter intervals described above.

An analogous test for extrapolating the regression results is not easily possible. Restricting the regression reference values to a limited interval (e.g., 0 to 90 degrees for the angle) leads to large systematic errors in the regression, since features like the sine and cosine of an angle are no longer uncorrelated. Therefore, one cannot use the standard procedure for linear regression. Using a more complicated algorithm, one could overcome this restriction but would also make the postprocessing part of our model more complex.

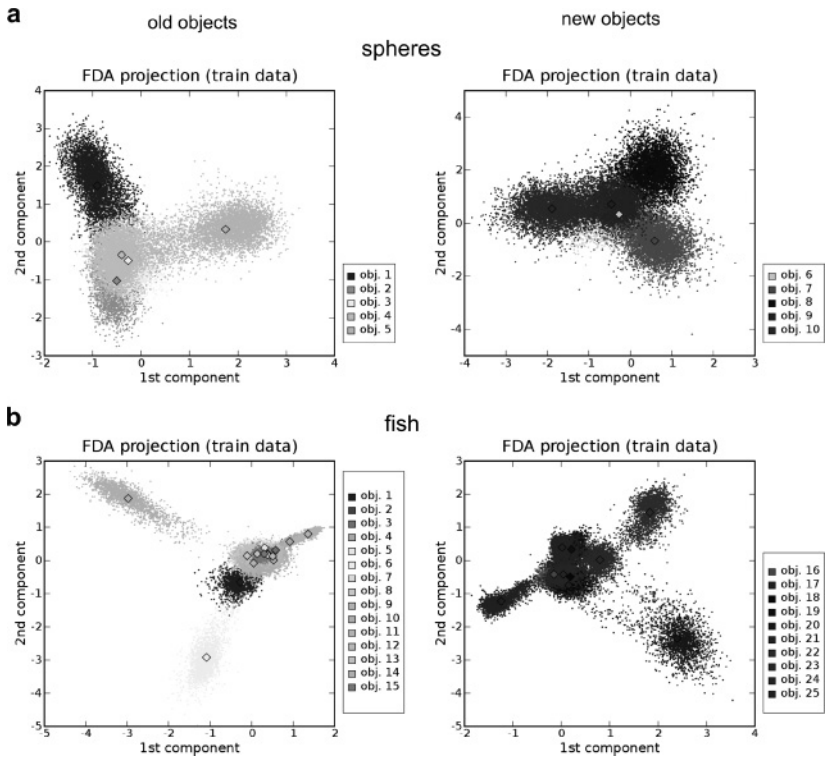

Figure 6: 2D projections of the data clusters. Two-dimensional projections of the data points are shown, which are colored according to object identity. The projection plane was chosen for each plot to maximize cluster separation (using Fisher discriminant analysis). (a) The data for the five old sphere cluster objects on the left and the five new ones on the right. (b) The projected data for the 15 old and 10 new fish objects.

Table 3: Classifier Hit Rates for Reduced Training Data of Fish Objects.

Number of Training Samples Per Object	Old Objects		New Objects		All Objects	
	G	KNN	G	KNN	G	KNN
1		79.9%		57.9%		60.5%
≈ 34	82.2%	90.6%	53.9%	75.6%	58.5%	76.6%
≈ 174	84.5%	90.7%	64.8%	82.3%	63.5%	79.0%

Notes: Training and test data were taken from different regions of the configuration space to test the generalization ability of the model; see text for details. KNN: the k-nearest-neighbor classifier ($k = 1$ in the first line, otherwise $k = 3$). G: Gaussian classifier.

3.3 Controls. To verify the robustness of our model and justify the model structure, we performed several control experiments.

3.3.1 Full Transformation Set. The following control experiments were done with the full transformation set for the sphere cluster or fish objects.

Choice of timescales. Our model strongly depends on the timescale of the transformations in the training images. We chose the random walk parameters such that the Δ-values of the transformations were in the same range (e.g., for the spheres, the smallest Δ is about 0.0045, and the largest 0.0081). Moving away from such balanced timescales generally leads to a decrease of the performance for the more quickly varying configuration variables. On the other hand, one can increase the performance for a particular variable by making it slower relative to the other variables. These dependencies were to be expected as a result of the limited function space available in the model. To verify this, we modified the fish random walk and trained a network with the new data. In one example, we increased the maximal velocity of the x-position from 0.06 to 0.12 and later 0.18, which increased the Δ-value to 0.017 and then 0.33. The RMSE for x on all 25 fish rose from 10% to 12% and then 22%. At the same time, the performance for all other configuration variables improved. For example, the gaussian classifier performance, on all 25 fish increased from 95.2% to 98.6% and 99.0%. If the goal was to maximize only the classifier performance, one would therefore make all other variables very fast. In this special case, SFA approximates nonlinear Fisher discriminant analysis (Berkes, 2005). Note, however, that the maximum transformation speed is limited by receptive field size: if speed is too high, no spatiotemporal structure can be extracted on lower layers with smaller receptive field sizes.

Blank frame. In the training sequences, we separated views of different objects with a blank frame. However, model performance is largely unaffected by omitting the blank.

Output dimensionality. The results presented so far are based on 512 SFA output dimensions (i.e., the 512 slowest outputs of the top layer), which may seem excessive. We found that a high number of outputs increased the quality of the results (see Table 4). The computational cost of working with 512 outputs is still relatively low (except for the k-nearest neighbor classifier). If the number of transformations is low, one can reduce the output number without sacrificing performance (the results in Figure 3 are an extreme case of this).

Nonlinearity. To verify that the nonlinear expansion is necessary, we trained and tested a purely linear network on the same fish object data as before. The hierarchical structure was still used, since otherwise, the covariance-matrix sizes would have been prohibitive. So in each layer, we omitted the quadratic expansion and the second SFA step. While the position of the objects was still extracted with an RMSE of 16%, in the x-direction

Table 4: Influence of the Number of SFA Output Outputs on the Performance (for Fish Objects).

Outputs	32	64	128	256	512
ϕ_y	40°	32°	25°	20°	17°
Classifier hit rate	80.7%	87.6%	91.2%	94.0%	95.2%

Notes: Results for different channel numbers. The first row shows the mean RMSE for the in-depth rotation angle of all 25 objects. The second row shows the hit rate of the gaussian classifier on all objects.

(as compared to 10% in the nonlinear network), the mean RMSE for the in-depth rotation angle increased to 60 degrees (as compared to 17 degrees). The hit rate of the gaussian classifier dropped to 39% (as compared to 95.2%) for all 25 fish objects. These results demonstrate the importance of the nonlinear expansion in the hierarchical network.

Baseline performance on pixel data. To verify that our stimulus set is nontrivial, we trained a supervised nearest cluster center classifier on the raw image data. It achieved a hit rate of only 14% on all 25 fish and 13% on the 10 sphere cluster objects. The dimensionality of the data ($155^2 = 24{,}025$ pixel) makes it difficult to test more sophisticated classifiers, and the computational cost would quickly exceed that of our model.

PCA instead of SFA. We replaced the SFA nodes in our model with PCA nodes. Due to the lack of whitening, the network then heavily relies on clipping. While better-than-chance level, the performance is poor compared to the SFA network (e.g., 41% hit rate of the of the gaussian classifier on all 10 sphere cluster objects). Introducing a whitening step reduces the clipping, but the performance remains low.

3.3.2 Reduced Transformation Set. In the following control experiments, we explore the impact of a reduced transformation set during training. This will be helpful to understand the ability of the network to learn the transformations used in the training stimuli. This is also important for understanding the contribution of the postprocessing steps to the overall performance:

Generalization to new stimulus class. As a test of the generalization capabilities of our model, we took a network that had been trained with sphere cluster objects and tested it with fish objects (the original random walk parameters from Table 1 were used). Surprisingly, the performance of this network was practically identical to that of the dedicated fish network. However, the performance of a fish network when applied to sphere cluster objects was clearly inferior to that of the dedicated sphere cluster network. In this case, the performance of the gaussian classifier dropped to 91% (as compared to 98%), and the mean RMSE for the angles grew by a factor

of two. Note that the sphere cluster objects incorporated the additional in-plane rotation, which was not present in the fish training data.

Changed Random Walk. We then used the random walk parameters of the fish for the sphere cluster objects and tested the performance of the fish network on this new stimulus set. The performance was now excellent (mean RMSE for the in-depth angle was 7 degrees and 99.97% hit rate of the gaussian classifier on all 10 objects). Together with the previous results, this suggests that the additional in-plane rotation caused the problems for the fish network, since it was not trained with two simultaneous rotations.

Test with Untrained Transformation. As another variant of the previous control experiments, we trained networks with reduced transformation sets and then tested their performance on stimuli with a larger transformation set. We trained a network with translations and different identities, but without any rotations, (this experiment was performed for both fish and sphere cluster objects). When the resulting network was tested with a stimulus set that included in-depth rotation, the results were surprisingly good. In general, the mean RMSE for the in-depth angle was a few degrees larger than in the normal case, and the classifier performance was just below 90%. However, when the in-plane rotation was added as well (for the sphere cluster objects) the performance took a major hit (the mean RMSE for the angles rose above 40 degrees for both in-plane and in-depth rotation, and the gaussian classifier performance dropped to 77% on all 10 sphere cluster objects). We also did a complementary experiment by training a network with in-depth rotation only and then testing it with a stimulus set that also included translation. In this case, the performance was very poor (e.g., only 50% hit rate of the gaussian classifier).

Analysis of in-depth estimation. One interpretation of the previous results would be that the extraction of the in-depth angle is made possible primarily by the supervised linear regression in combination with the 512 dimensions. To verify this, we tested a network in which the two SFA nodes in each layer are replaced with nodes that pick random orthogonal directions (which are fixed throughout the experiment) and added a whitening stage in each layer. On a stimulus set containing only in-depth rotation, this produced reasonable results (mean RMSE for the angle was 23 degrees, and gaussian classifier hit rate was 93.3% on all 10 sphere cluster objects). However, as soon as translations were introduced, the performance went down to chance level. In both cases, the whitening stage was trained with the same transformations that were used during testing.

Reduced output dimensionality. To check if the postprocessing results for untrained transformations are purely an artifact of the large number of dimensions, we checked the performance for a reduced number of outputs (by taking only the slowest 128 network outputs, as in one of the earlier control experiments). It turned out that the decrease in performance is comparable to that of a normal network, so the performance gap between a

network trained on a reduced transformation set and a fully trained network does not depend on the output dimensionality.

Generalization to untrained rotation type. The sphere cluster stimuli where also used to determine whether there is a qualitative difference in network learning between stimuli with in-depth and in-plane rotation. One network was trained without in-plane rotation, and a second one was trained without in-depth rotation (all other transformations were left unchanged and conformed to Table 1). Then both networks were tested with the standard sphere cluster testing stimuli, which include both in-depth and in-plane rotation. As expected, the performance in both cases was worse than that of a fully trained network. However, the performance of the network that was trained with in-plane rotation was slightly better than that with in-depth rotation (e.g., 96.7% versus 93.4% gaussian classifier hit rate and 16.6 degrees versus 20.8 degrees mean RMSE for the in-depth angle). The fact that the network was not able to create an efficient representation of the in-depth rotational angle supports the view that in-depth rotation is harder to learn than in-plane rotation.

The results suggest that for a single continuous transformation type (e.g., in-depth rotation), the postprocessing alone is able to produce reasonable results. But as soon as multiple transformations are present, this is no longer true, and the network is critical for the performance. A network that has been trained with translations only can deal reasonably well with a single kind of rotation (even though it is still outperformed by a properly trained network). When two rotations are included, then at least one of them must have been trained.

3.3.3 Background and Distractors. So far, we used only a white background without distractors. In principle, the model should be able to deal with changing background to some extent, since those changes typically happen on a different timescale those than of object identity. While the background properties would enter the model output, the relevant object variables should still be present as well. One could also artificially render the model invariant to the background or distractors by rapidly changing them (Einhäuser et al., 2005), thus sacrificing some realism. Unfortunately our tests in that direction were not very promising. The reason might be that the computational power of the network was not sufficient. Furthermore, in our simulations, contrary to those in Einhäuser et al. (2005), objects were not centered in the input image, such that a suppression of peripheral areas is no solution to cope with clutter influence. In general we believe that recognizing objects in cluttered scenes cannot be solved with the slowness principle in a feedforward network. It seems necessary to employ attentional top-down mechanisms for that.

In natural scenes, the assumption that the slowest features are the most interesting can also break down to some extent. The circadian rhythm, for example, might be extracted by SFA, although it is not necessarily of interest.

However, the timescales of the interesting features are typically not known. While SFA is based on the shortest (i.e., the input derivatives) and longest available timescale (i.e., the input signal length), these could be adapted as in Stone and Bray (1995), which is beyond the scope of this letter.

3.4 HRI50 Data Set. In addition to the rendered fish and sphere objects, we applied our model to the HRI50 database. Object identity could be robustly extracted with our model on this real-world database with high accuracy. We achieved a recognition rate of 95.6% with a nearest-neighbor classifier on the top-level output (baseline classification on raw data: 89.2%), but to what degree pose information is also extracted cannot be answered in this case because ground-truth pose data are missing. Published results on this data set are based on a combination of sparse shape features (Wersing & Körner, 2003) and color features. The highest published recognition rates are 94.7% (Kirstein et al., 2005) (but note the different classifier). These numbers show that our model also performs competitively for images of real objects and freely generated pose trajectories.

4 Discussion

In the previous section, we showed how the hierarchical model learns representations of object position, angle, and identity from quasi-natural stimuli. Here, we discuss advantages and limitations of our models, as well as the relation of our model to existing work.

We demonstrated how the unsupervised learning of slowly varying signals extracts relevant latent variables coding for object identity and pose. In section 3.1 we showed that these representations in principle can already encode object identity and pose independent of each other without any supervised postprocessing. Such a property is desirable for a visual system where unlabeled data are abundant and supervised learning with labeled data is rare or costly. The time structure of object presentation determines the learned representations, which is determined by the agent's behavior (e.g., a child or a robot). In our system, rotation invariance increases for an object if it is rotated rather quickly and switched rarely, which corresponds to a child inspecting objects in his or her hands. However, the results in section 3.2 show that the theoretically determined optimal solutions in our model can mix for higher numbers of objects and that a postprocessing step can then lead to good pose estimation and classification rates. We have demonstrated supervised classification and regression but unsupervised steps (e.g., clustering) or reinforcement learning, as demonstrated in Legenstein, Wilbert, & Wiskott (2010), are an alternative for constructing a completely unsupervised system. Nevertheless, as a small number of slow features on the top of the hierarchy code for object identity and pose, there are few parameters to fit in a supervised postprocessing step and thus few

training data are necessary, which minimizes the number of labeled training examples given by a teacher.

The primate visual system is organized in two hierarchical pathways. Classically, the ventral pathway is assumed to perform invariant object recognition ("what pathway"). This pathway involves the cortical areas V1 (primary visual cortex), V2, V4, and IT (inferotemporal cortex). On the way from V1 to IT, neurons show increasing receptive field size, stimulus specificity, and invariance. At the top of this hierarchy, in IT, many neurons code for objects with high invariance to position, angle, scale and so on. The dorsal pathway is assumed to perform complementary spatial computation ("where pathway") (Ungerleider & Mishkin, 1982). Most models of invariant object recognition are inspired by this interpretation of the ventral path and implement explicit mechanisms for increasing position invariance (Fukushima, 1980; Wersing & Körner, 2003; Rolls & Stringer, 2006), which successfully reduces view dependence of their representations. There is, however, some more recent evidence against a complete segregation of "what" and "where" information in the macaque brain (Hung, Kreiman, Poggio, & DiCarlo, 2005; Lehky, Peng, McAdams, & Sereno, 2008), showing that information about position and size of objects is also represented in the inferotemporal cortex of macaques, that is, in a top layer of the ventral pathway. This latter view is more consistent with our model, which performs invariant object recognition and simultaneous pose estimation. We conjecture that object identity and pose are tightly coupled: pose estimation requires information on object identity (or at least object category), and object identity needs to be inferred from an object's view in a specific pose. Therefore, an early segregation of identity and spatial configuration into separate processing streams would likely require redundant computing hardware.

We provide no layer-by-layer comparison of our model with the ventral stream, because such a comparison would not only require that the computational principle is similar but also that the specific implementation is comparable. Specifically, the computational power of a layer in our model is determined by the employed nonlinearity, which is hard to estimate for neural substrate, and thus we did not try to match against that of the visual system. Nevertheless, some model properties can be matched to the visual system or can serve as predictions for testing the model. The input data for our model consist of high-dimensional pixel images, which could be associated with a retinal representation. The first layer of our model computes quadratic functions on patches of these images. Earlier work has established that these functions closely approximate V1 complex cell properties (Berkes & Wiskott, 2005). On the top of the ventral stream, in IT, we predict neurons coding for invariant object representations as well as neurons representing object pose. Although information on object pose might be mixed with object identity information, a linear regression should be able to extract the 3D rotational angles from a population of IT neurons

for well-known objects. Furthermore, in a system optimizing a slowness function, the presentation statistics of newly seen objects should influence object representation. For example, frequent switching of presented views of two distinct objects should make the representations of the two objects in IT more similar. A comparison of intermediate layers of our model with the ventral stream is much harder. Nevertheless, we predict that the Δ-values of representations decrease toward the top of the ventral hierarchy. For example, the presentation of a rotating object should elicit more quickly varying responses in V1 than in V4 or in IT.

Our model contains no feedback connections, which is in clear contrast to the massive feedback connections present in the primate visual system. A pure feedforward model has a simpler architecture and can compute responses more quickly than a system involving cross-layer feedback. The early onset of object-specific firing in IT reported by Hung et al. (2005) suggests that the macaque visual system can compute object-specific results without extensive feedback across hierarchical layers. With this letter, we show how a model without feedback can compute view-invariant object-specific representations as well as object poses. Our model operates, however, in a restricted regime without attention, scene clutter, or top-down object bias, which might all require a feedback system.

Invariant object recognition by definition implies that pose information is discarded. Yet we claim that our model performs both invariant object recognition and pose estimation. The object presentation statistics used in our experiments (and likely also during object inspection by humans) implies that an object is typically viewed for an extended period of time, while it is rotated and shifted around. In this setting, object identity is a slower feature (i.e., has lower Δ-value) than position and orientation, and thus the slowest SFA solutions code for object identity and are invariant to pose (e.g., output 1 in Figure 3). For N objects with approximately similar viewing time, our model will find $N-1$ such solutions with similar Δ-values that jointly form a pose-invariant object population code and could be identified in an unsupervised way. The following SFA solutions have higher Δ-values and code for object pose. In Figure 3, outputs 2 to 5 code for the rotational angles of the objects. Both pose and identity information are obviously also present in the input data, but our system represents both in an explicit and low-dimensional way. Could our model also extract a pose code as slowest features? If so, it would be easier to identify those outputs coding for pose in an unsupervised way. One approach to get such results would be to alter the object presentation such that object pose changes more slowly than object identity, which seems implausible. However, it might be possible to extend our model with a learning rate adaptation mechanism (Franzius et al., 2007) and thus learn different invariance properties with the given movement statistics. A further possible extension involves a model hippocampus that provides memories of visual images in a different temporal order than originally perceived (Franzius & Wersing, 2010).

4.1 Related Work. The problem of invariant object recognition has been approached from two sides in the past (Wiskott, 2006). One approach is mainly motivated by the biological implementation in the (primate) brain with a focus on biological realism, generality, and unsupervised learning but was mostly limited to very simple stimuli. The other approach comes from computer vision, uses sophisticated machine learning approaches, works for complex stimuli, and is highly adapted to a specific problem but is often not biologically plausible. In this discussion, we focus on the biologically relevant models.

According to the classification by Rolls and Deco (2002), our network belongs to the category of feature hierarchy-based computational object recognition devices. Such systems extract increasingly complex features in a hierarchical system. In contrast to flat feature space systems that are typically insensitive to scrambled input images (Mel, 1997), object recognition in primates is highly sensitive to the relative location of features, for example, position of eyes and the nose in a face (Rolls, Tovee, Purcell, Stewart, & Azzopardi, 1994; Vogels, 1999; Grill-Spector et al., 1998). The increasing receptive field size in hierarchical feature systems naturally preserves susceptibility to scrambling, as each layer is sensitive to local arrangements of spatial features.

The slowness principle has been applied in many models of sensory systems in the past (Földiák, 1991; Stone & Bray, 1995; Bartlett & Sejnowski, 1998; Becker, 1999; Kayser, Einhäuser, Dümmer, König, & Körding, 2001; Rolls & Deco, 2002; Wiskott & Sejnowski, 2002; Berkes & Wiskott, 2005; Hipp, Einhäuser, Conradt, & König, 2005; Wyss, König, & Verschure, 2006; Franzius et al., 2007). Some of these unsupervised models learn on naturalistic input data and generate representations similar to neural codes, like V1 simple cells, V1 complex cells, or place cells, head direction cells, and spatial view cells in the hippocampal formation. A number of publications have used the slowness principle for invariant object or face recognition (Becker, 1999; Wiskott & Sejnowski, 2002; Stringer & Rolls, 2002; Rolls & Stringer, 2006), but to our knowledge, pose extraction has not been demonstrated before. Furthermore, most models use simplified stimuli and apply only one transformation (rotation or translation) and repeatedly present individual views.

VisNet (Rolls & Deco, 2002) is such a model based on the trace rule (Földiák, 1991; Rolls, 1992; Wallis & Rolls, 1997), which is closely related to slow feature analysis (Sprekeler et al., 2007). As in our model, features are learned in an unsupervised manner from the statistics of the network's stimuli. Only the lowest-layer weights are initialized explicitly to Gabor-like structures. Like our model, VisNet can learn a position-invariant (or view-invariant) but object-specific code. In contrast to our model, only a single invariant representation is extracted (i.e., object identity), and the other parameters (e.g., object position, rotation angles, lighting direction) are discarded. Furthermore, VisNet has a number of extra variables that

need to be tuned per layer: learning rate α, trace length η, inhibition radius σ, contrast δ, and activity percentile a. A further difference comes from the use of sparse coding in VisNet by means of the lateral inhibition and winner-takes-most architecture. In our model, redundancy reduction comes from the decorrelation constraint in SFA, which is performed in each node at each grid position, but no spatial decorrelation over adjacent nodes in the grid of a layer is enforced. While both SFA and the trace rule are linear, the quadratic expansions in our model allow the system to find highly nonlinear functions.

In Stringer and Rolls (2002), VisNet was applied to views of six spheres rotated indepth over an angle of 120 degrees. The spheres had different surface features consisting of different configurations of three line segments. After learning with the trace rule, a part of the neurons in the top layer of the model coded for sphere identity and were invariant to the rotational angle. These results obtained for a single transformation correspond to a simpler version of our results in section 3.1. However, there is no natural ordering of outputs in this model that would allow identifying neurons, which code for object identity, in an unsupervised way. Furthermore, the model performs no pose estimation.

Stringer, Perry, Rolls, and Proske, (2006) have used a variant of VisNet with a purely Hebbian learning rule. Interestingly, in one of their control experiments, the trace rule was applied to stimuli that rapidly changed object identity and slowly changed the angle. The authors found that the resulting representation codes poorly for object identity. We expect that this representation rather codes for object angle.

The model by Einhäuser et al. (2005) applies a very similar learning rule as our model that is basically a gradient descent on the optimization problem in section 2.2. Similar to VisNet, the first layer uses static Gabor wavelets to model complex cells in V1, whereas the second layer is optimized according to the slowness principle. As this model uses only two layers, the absolute sum of all outputs of the first layer with the same size and orientation is used as input for the second layer, which introduces a hard-coded translation invariance. After optimization, output units represent object identity independent of viewpoint. Between 10 and 50 training objects from the standard COIL database are used as stimuli in front of a homogeneous or cluttered background. The COIL database contains photos of real-world objects centered and rotating on a turntable. Thus, object transformations comprise in-depth rotation, changing object identity, some rescaling, but no translation. The model incorporates color information (as RGB channels) into the model. In our model, we intentionally reduced the input data to gray-scale representations because color histograms alone can often be highly informative about object identity.

There is a large body of work in the field of computer vision on 3D object pose estimation using local linear models from object views (Roweis & Saul, 2000; Jonsson & Felsberg, 2007). These unsupervised approaches

usually use Euclidean distance in pixel space as a similarity measure and ignore the temporal structure of the input data. Such a transformation parameterizes all transformations into a low-dimensional space, and it is not evident how such a system could, for example, implement position invariance. Furthermore, to our knowledge, there is no work on invariant object recognition and pose estimation available in such a system. In principle, such an approach could be used on the SFA output of our model instead of a supervised classification or regression step. Such a completely unsupervised system might benefit from certain invariances (e.g., position) and the lower-dimensional representation.

The model by Rahimi, Recht, and Darrell (2005) comes from the computer vision field and has no direct link to a biological system. However, the learning rule applied in this model is very similar to slowness learning. Slowness of the outputs is enforced by finding a smooth mapping of the inputs to a low-dimensional random field that is governed by a second-order Newtonion process. Few key frames are labeled by the user, and the system maps unseen stimuli to the low-dimensional manifold. This semisupervised system identifies the complex transformations in high-dimensional video data of people performing complex movements or of talking and grimacing people.

The model by Ullman and Bart (2004) is an extension of a simple feature-matching system using extended features. Image regions are learned (they are called fragments) that have maximal mutual information between a fragment and the object class it represents. As a fragment might occur at different positions in an image, this model has to explicitly try all possible fragment positions. A fragment might not be visible in all views, and so an "extended fragment" consists of a set of alternative fragments (e.g., different views of an eye). The model addresses only a single invariance (i.e., view), which is built in explicitly by the extended fragment approach.

The model by Wersing and Körner (2003) also uses biologically motivated hierarchical unsupervised learning for invariant object recognition and, in contrast to our model, is based on sparse coding. The model copes well with background clutter and is highly efficient for real-time applications as it only propagates sparse nonnegative activations in the hierarchy. In contrast to our approach, this model neglects the temporal structure of visual data and performs no pose extraction.

The work by Becker (1999) introduces a biologically motivated generative hierarchical model of cortical visual processing. The model performs unsupervised clustering with a gaussian mixture model gated by temporal context. In one simulation, the network performed viewpoint-independent face categorization on a set of 10 faces based on temporal continuity of face presentation during the training phase. In a second simulation, the model learned a pose-specific code after relaxing the temporal context signal. This architecture focuses on a more general context signal, which might be a

temporal one. In contrast to our model, it seems hard to predict the nature of the invariance properties of the representations learned by the model for a specific set of architectural design choices.

5 Conclusion

The model proposed here learns invariant object representations based on the statistics of the presented stimuli during the training phase. Specifically, representations of transformations that occur slowly (like orientation) or seldom (like object identity) are encoded first, while information on other transformations that occur most often or quickly is encoded later or discarded. An advantage of such a system is that no invariances have to be hard-wired. The unsupervised learning of the system is described by a comprehensive mathematical analysis that predicts specific output signal components. However, if many transformations occur simultaneously and on similar timescales, solutions tend to mix. For this case, a final step of supervised learning (i.e., classification or linear regression) can recover the relevant transformations. We show that the system generalizes well to previously unseen configurations (i.e., shows no overfitting for test positions and viewing angles of objects) and to previously unseen objects. However, the system behavior for completely different configurations, as for two simultaneously presented objects, cannot be predicted with our current theory.

With these results, we demonstrate a single mechanism for learning and representing both view-invariant object-specific representations and position and rotational information from complex high-dimensional data. We demonstrate that these capabilities are possible in a purely feedforward system. Our model can serve as a biologically plausible model of the ventral visual system with a single unsupervised learning rule from pixel data to a quasi-symbolic level. Furthermore, this efficient and mostly unsupervised classification and pose estimation might be interesting for technical applications.

References

Bartlett, M., & Sejnowski, T. (1998). Learning viewpoint invariant face representations from visual experience in an attractor network. *Network: Computation in Neural Systems, 9*(3), 399–417.

Becker, S. (1999). Implicit learning in 3d object recognition: The importance of temporal context. *Neural Computation, 11*(2), 347–374.

Becker, S., & Zemel, R. S. (2003). *Unsupervised learning with global objective functions*. Cambridge, MA: MIT Press.

Berkes, P. (2005). Handwritten digit recognition with nonlinear Fisher discriminant analysis. In *Proceedings of ICANN 2005* (Vol. 2, pp. 285–287). Berlin: Springer.

Berkes, P., & Wiskott, L. (2005). Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision, 5*(6), 579–602. http://journalofvision.org/5/6/9/, doi:10.1167/5.6.9.

Booth, M. C., & Rolls, E. T. (1998). View-invariant representations of familiar objects by neurons in the inferior temporal visual cortex. *Cerebral Cortex, 8*, 510–523.

DiCarlo, J., & Cox, D. (2007). Untangling invariant object recognition. *Trends in Cognitive Science, 11*, 333–341.

Einhäuser, W., Hipp, J., Eggert, J., Körner, E., & König, P. (2005). Learning viewpoint invariant object representations using a temporal coherence principle. *Biological Cybernetics, 93*, 79–90.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation, 3*, 194–200.

Franzius, M., Sprekeler, H., & Wiskott, L. (2007). Slowness and sparseness lead to place, head-direction and spatial-view cells. *PLoS Computational Biology, 3*(8), e166. doi:10.1371/journal.pcbi.0030166.

Franzius, M., & Wersing, H. (2010). Learning invariant visual shape representations from physics. In *Proceedings of Artificial Neural Networks* (pp. 298–302). Berlin: Springer.

Franzius, M., Wilbert, N., & Wiskott, L. (2009). *Online supplementals*. http://www.nikowilbert.de/supplements.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics, 36*(4), 193–202.

Grill-Spector, K., Kushnir, T., Hendler, T., Edelman, S., Itzchak, Y., & Malach, R. (1998). A sequence of object-processing stages revealed by fMRI in human occipital lobe. *Human Brain Mapping, 6*, 316–328.

Hashimoto, W. (2003). Quadratic forms in natural images. *Network: Computation in Neural Systems, 14*(4), 765–788.

Hipp, J., Einhäuser, W., Conradt, J., & König, P. (2005). Learning of somatosensory representations for texture discrimination using a temporal coherence principle. *Network: Computation in Neural Systems, 16*(2/3), 223–238.

Hung, C., Kreiman, G., Poggio, T., & DiCarlo, J. J. (2005). Fast readout of object identity from macaque inferior temporal cortex. *Science, 310*(5749), 863–866.

Jonsson, E., & Felsberg, M. (2007). Accurate interpolation in appearance-based pose estimation. *Lecture Notes in Computer Science, 4522*, 1.

Kayser, C., Einhäuser, W., Dümmer, O., König, P., & Körding, K. (2001). Extracting slow subspaces from natural videos leads to complex cells. In *Artificial Neural Networks—ICANN 2001 Proceedings* (pp. 1075–1080). Berlin: Springer.

Kirstein, S., Wersing, H., & Körner, E. (2005). Rapid online learning of objects in a biologically motivated recognition architecture. In *27th Pattern Recognition Symposium DAGM* (pp. 301–308). Berlin: Springer.

Kirstein, S., Wersing, H., & Körner, E. (2008). A biologically motivated visual memory architecture for online learning of objects. *Neural Networks, 1*, 65–77.

Legenstein, R., Wilbert, N., & Wiskott, L. (2010). Reinforcement learning on slow features of high-dimensional input streams. *PLoS Comput Biol., 6*(8), e1000894. doi:10.1371/journal.pcbi.1000894. http://dx.doi.org/10.1371%2Fjournal.pcbi.1000894.

Lehky, S., Peng, X., McAdams, C., & Sereno, A. (2008). Spatial modulation of primate inferotemporal responses by eye position. *PLoS ONE, 3*(10), e3492. doi:10.1371/journal.pone.0003492.

Li, N., & DiCarlo, J. (2008). Unsupervised natural experience rapidly alters invariant object representation in visual cortex. *Science, 321*, 1502–1507.

Li, N., & DiCarlo, J. (2010). Unsupervised natural experience rapidly reshapes size-invariant object representation in inferior temporal cortex. *Neuron, 67*, 1062–1075.

Mel, B. W. (1997). SEEMORE: Combining color, shape and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation, 9*(4), 777–804.

Nayar, S. K., Nene, S. A., & Murase, H. (1996). Real-time 100 object recognition system. In *Proc. of ARPA Image Understanding Workshop*. Piscataway, NJ: IEEE.

Olshausen, B. A., & Field, D. J. (1996). Natural image statistics and efficient coding. *Network: Computation in Neural Systems, 7*, 333–339.

Rahimi, A., Recht, B., & Darrell, T. (2005). Learning appearance manifolds from video. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vol. 1, pp. 868–875). Washington, DC: IEEE Computer Society.

Rolls, E. T. (1992). Neurophysiological mechanisms underlying face processing within and beyond the temporal cortical visual areas. *Philosophical Transactions of the Royal Society, 335*, 11–21.

Rolls, E. T., & Deco, G. (2002). *The computational neuroscience of vision*. New York: Oxford University Press.

Rolls, E. T., & Stringer, S. M. (2006). Invariant visual object recognition: A model, with lighting invariance. *Journal of Physiology–Paris, 100*, 43–62.

Rolls, E. T., Tovee, M. J., Purcell, D. G., Stewart, A. L., & Azzopardi, P. (1994). The responses of neurons in the temporal cortex of primates, and face identification and detection. *Experimental Brain Research, 101*(3), 473–484.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323–2326.

Sprekeler, H. (2009). *Slowness learning: Mathematical approaches and synaptic mechanisms*. Doctoral dissertation, Humboldt-Universität zu Berlin. http://edoc.hu-berlin.de/docviews/abstract.php?id.-296595.

Sprekeler, H., Michaelis, C., & Wiskott, L. (2007). Slowness: An objective for spike-timing-plasticity? *PLoS Computational Biology, 3*(6), e112.

Stone, J. V., & Bray, A. (1995). A learning rule for extracting spatio-temporal invariances. *Network: Computation in Neural Systems, 6*, 429–436.

Stringer, S. M., Perry, G., Rolls, E. T., & Proske, J. H. (2006). Learning invariant object recognition in the visual system with continuous transformations. *Biological Cybernetics, 94*, 128–142.

Stringer, S. M., & Rolls, E. T. (2002). Invariant object recognition in the visual system with novel views of 3D objects. *Neural Computation, 14*, 2585–2596.

Toucan Corporation. (2005). Toucan virtual museum. http://toucan.web.infoseek.co.jp/3DCG/3ds/FishModelsE.html.

Ullman, S., & Bart, E. (2004). Recognition invariance obtained by extended and invariant features. *Neural Networks, 17*, 833–848.

Ungerleider, L. G., & Mishkin, M. (1982). Two cortical visual systems. In D. J. Ingle, M. A. Goodale, & R.J.W. Mansfield (Eds.), *Analysis of visual behavior* (pp. 549–586). Cambridge, MA: MIT Press.

Vogels, R. (1999). Effect of image scrambling on inferior temporal cortical responses. *NeuroReport, 10*, 1811–1816.

Wallis, G., & Rolls, E. T. (1997). Invariant face and object recognition in the visual system. *Progress in Neurobiology, 51*(2), 167–194.

Wersing, H., & Körner, E. (2003). Learning optimized features for hierarchical models of object recognition. *Neural Computation, 15*(7), 1559–1588.

Wiskott, L. (1998). Learning invariance manifolds. In L. Niklasson, M. Bodén, & T. Ziemke (Eds.), *Proceedings of the 8th International Conference on Artificial Neural Networks* (pp. 555–560). Berlin: Springer.

Wiskott, L. (2003). Slow feature analysis: A theoretical analysis of optimal free responses. *Neural Computation, 15*(9), 2147–2177.

Wiskott, L. (2006). How does our visual system achieve shift and size invariance? In J. L. van Hemmen & T. J. Sejnowski (Eds.), *23 problems in systems neuroscience* (pp. 322–340). New York: Oxford University Press.

Wiskott, L., & Sejnowski, T. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation, 14*(4), 715–770.

Wyss, R., König, P., & Verschure, P. (2006). A model of the ventral visual system based on temporal stability and local memory. *PLoS Biology, 4*(5), e120.

Zito, T., Wilbert, N., Wiskott, L., & Berkes, P. (2008). Modular toolkit for Data Processing (MDP): A Python data processing framework. *Frontiers in Neuroinformatics, 2*.

Received November 6, 2008; accepted October 7, 2010.