2008 Proc 18th Intl Conf on Artificial Neural Networks
Lecture Notes in Computer Science, 5163:961-970

Invariant Object Recognition with Slow Feature Analysis

M. Franzius[1], N. Wilbert[1], L. Wiskott[1]

Institute for Theoretical Biology, Humboldt-Universität zu Berlin, Germany

Abstract. Primates are very good at recognizing objects independently of viewing angle or retinal position and 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, where each code is independent of all others. We demonstrate the model behavior on complex three-dimensional objects under translation and in-depth rotation on homogeneous backgrounds. A similar model has previously been shown to extract hippocampal spatial codes from quasi-natural videos. The rigorous mathematical analysis of this earlier application carries over to the scenario of invariant object recognition.

1 Introduction

Sensory signals convey information about the world surrounding an animal. However, a visual 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 lighting conditions change. A high-level representation of object identity in the brain should, however, remain constant or *invariant* under these different conditions. How could the brain extract this abstract information from the highly variable stimuli it perceives? How can this task be learned from the statistics of the visual stimuli in an unsupervised way?

The primate visual system is organized hierarchically with increasing receptive field sizes, increasing stimulus specificity and invariance towards to top of the (ventral) hierarchy, where many neurons are coding for objects with an extreme amount of invariance to position, angle, scale etc.

Many models of invariant object recognition have been proposed in the last decades [1]. However, many approaches fail or have not been shown to work for natural stimuli and complex transformations like in-depth rotations. But invariant object recognition is only one task the (primate) brain has to achieve in order to successfully interact with the environment. We do not only need to extract the identity of an object ("what is seen?") independently of its position and view direction, we also want to extract the position of an object ("where is it?") independently of its identity or viewing angle. The relative rotational

angle of a viewed object can be just as crucial ("does the tiger look at me?"). In principle, we might want a representation of any aspect (i.e., size, viewing angle, lighting direction etc.) independently of all the others and optimally, all these tasks should be solved with a single computational principle. In the following we will refer to the configuration of position and angles relative to the viewer as the *configuration* of an object (sometimes also called a pose). We will call a 2D image of an object in a specific configuration a *view*. 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-dimensional views of complex objects but restrict the problem to the cases of only one object present at any time point and a static homogeneous background. A good model should also *generalize* to previously unseen configurations, i.e., it should learn the relevant statistics of transformations rather than just memorizing specific views. It should also generalize to new objects, e.g., it should successfully estimate the position and orientation[1] of an object that was never shown before.

We apply here a hierarchical model based on the learning principle of *temporal slowness*. The structure of the resulting representation solely depends on the statistics of presentation of the training views. This information is encoded in an analytically predictable way and very simple to decode by linear regression. Except for minor changes, the model used here is identical to that used earlier for the modeling of place cells and head direction cells in the hippocampal formation from quasi-natural videos [2].

2 Methods

2.1 Stimulus Generation

The model was trained and tested with image sequences containing views of different objects. OpenGL was used to render the views of the objects as textured 3D-models in front of a white homogenous background. To prevent the model from relying on simple color cues (especially for object classification) we only used grayscale views for the results presented here. Two different object classes were used that are described below. For each object class the model was trained with five objects. In the testing phase we added five new objects, which the model had never seen during training.

In the first experiment the objects were clusters of textured spheres as shown in Figure 1 (a), which provide a difficult but generic task. The same six textured spheres (textures from VisTex database [3]) were used in different spatial arrangements for all the objects. For each object the spheres were randomly fixed on a hexagonal lattice of size $2 \times 2 \times 3$. As the examples in Figure 1 (a) illustrate, identifying the rotation angles and identities for such objects is quite difficult even for human observers. Using spheres has the advantage that the outline does

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

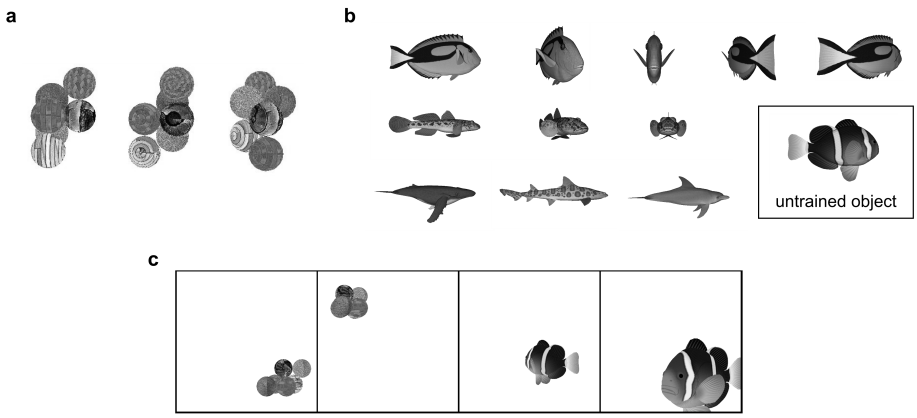

Fig. 1. Objects used for stimulus generation. Part (a) shows the sphere objects (each cluster of 6 spheres is one object). The first two views show the same object under different in-depth rotation angles, while the third view shows a different object. In part (b) the five fish objects used for training are shown with examples for the effect of in-depth rotation. The fish model on the bottom right is one of the five untrained fish used for testing. Part (c) shows some examples for the training and testing images.

not give a simple clue for the in-plane rotation angle. In the second experiment, models of different fish (see Figure 1 (b)) were used to provide more natural stimuli from a single object class (all models taken from [4]).

For sphere objects, the x-coordinate, the y-coordinate (vertical image 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 $360°$. Another configuration variable was the object identity ranging from one to ten, with objects one to five being the training objects. So the transformations the model 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. For the fish objects, the configuration variables were x, y, z, ϕ_y and object identity. So compared to the sphere objects we added translations in depth along the z-axis and removed the in-plane rotations. A pure z-translation changes both the object size and the position in the frame, due to the perspective projection.

The configurations for the training sequences were generated as a random walk procedure like in [2]. To generate a configuration in the sequence we add a random term to the current spatial and angular velocities of the currently shown object. By adjusting the magnitude of the velocity updates one can effectively choose the timescales of the changes which are relevant for SFA. The position and angles are then updated according to these velocities. In each step the object identity was changed with low probability ($p = 0.001$). A blank frame was inserted in between if a switch took place. This procedure adds some realism,

because in natural scenes a change in object identity without any changes in other configuration variables generally does not occur.

2.2 Slow Feature Analysis

Slow Feature Analysis 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:

Definition 1 (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 $y_j(t) := g_j(\mathbf{x}(t))$*

$$\text{minimize } \Delta(y_j) := \langle \dot{y}_j^2 \rangle_t \tag{1}$$

under the constraints

$$\langle y_j \rangle_t = 0 \quad \text{(zero mean)}, \tag{2}$$

$$\langle y_j^2 \rangle_t = 1 \quad \text{(unit variance)}, \tag{3}$$

$$\forall i < j : \langle y_i y_j \rangle_t = 0 \quad \text{(decorrelation and order)}, \tag{4}$$

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

Equation (1) introduces the Δ-value, which is a measure of the temporal slowness 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. The constraints (2) and (3) avoid the trivial constant solution and constraint (4) ensures that different functions g_j code for different aspects of the input.

It is important to note that although the objective is slowness, the functions g_j are instantaneous functions of the input, so that slowness cannot be enforced by low-pass filtering. Slow output signals can only be obtained 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 fast, 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 Slow Feature Analysis [5, 6]. This algorithm, which is based on an eigenvector approach, is guaranteed to find the global optimum. Biologically more plausible learning rules for the optimization problem, both for graded response and spiking units exist [7, 8]. If the function space is infinite-dimensional, the problem requires variational calculus and will in general be difficult to solve. In [2] it has been shown that the optimization problem for the high-dimensional visual input can be reformulated for the low-dimensional configuration input. This provides very useful predictions

for the SFA-output of our model (even though it is only based on the finite-dimensional SFA algorithm). We use these predictions for the postprocessing of our model output (see 2.4). For the detailed predictions and their derivations see [9] or [2].

2.3 Network Architecture

The computational model consists of a converging hierarchy of layers of SFA nodes (see Figure 2). Each SFA node finds the slowest output features from its input according to the SFA algorithm and performs the following sequence of operations: linear SFA for dimensionality reduction, quadratic expansion with subsequent additive Gaussian white noise (with a variance of 0.05), another linear SFA step for slow-feature extraction, and clipping of extreme values at ± 64 (Figure 2). Effectively, a node implements a subset of full quadratic SFA. The clipping removes extreme values that can occur on test data very different from training data.

In the following, 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 by 10 grayscale pixels that jointly cover the input image of 128 by 128 pixels. The nodes form a regular (i.e., non-foveated) 24 by 24 grid with partially overlapping receptive fields. The second layer contains 11 by 11 nodes, each receiving input from 4 by 4 layer 1 nodes with neighboring receptive fields, resembling a retinotopical layout. The third layer contains 4 by 4 nodes, each receiving input from 5 by 5 layer 2 nodes with neighboring receptive fields, again in a retinotopical layout. All 4 by 4 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 (called units), 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 layers are trained sequentially from bottom to top (with slightly faster varying views for the top layers). For computational efficiency, we train only one node with stimuli from all node locations in its layer and replicate this node throughout the layer. This mechanism effectively implements a weight-sharing constraint. However, the system performance does not critically depend on this mechanism. To the contrary, individually learned nodes improve the overall performance.

For our simulations, we use 100,000 time points for the training of each layer. Since 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 its configuration (position, angles, and object identity). The network is implemented in Python using the MDP toolbox [10], and the code is available upon request.

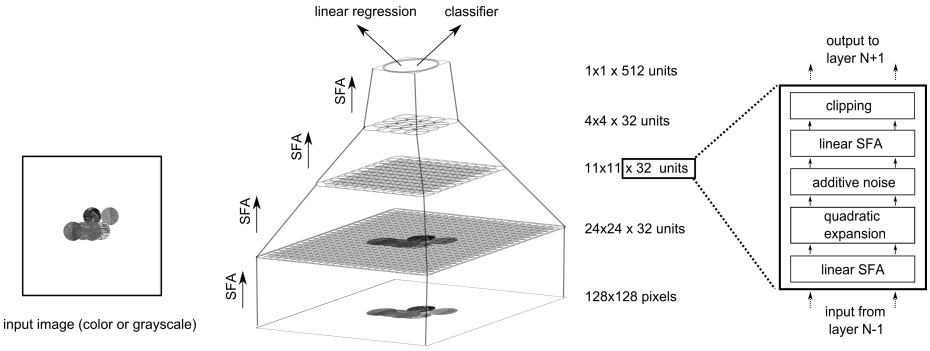

Fig. 2. Model architecture and stimuli. An input image is fed into the hierarchical network. The circles in each layer denote the overlapping receptive fields for the SFA-nodes and converge towards the top layer. The same set of steps is applied on each layer, which is visualized on the right hand side.

2.4 Feature Extraction with Linear Regression

The function space available to the model is large but limited (a subset of polynomials of degree $2^4 = 16$), which will generally lead to deviations from the theoretical predictions. This causes linear mixing of features in the SFA-output which is practically impossible to avoid in complex applications. Therefore we extracted the individual configuration features in a separate post-processing step.

Our main objective here was to show that the relevant features were indeed extracted from the raw image data. The easiest way to do this is by calculating a multivariate linear regression of the SFA-output against the known configuration values. While the regression procedure is obviously supervised, it nevertheless shows that the relevant signals are easily accessible. Extracting this information from the raw image data linearly is not possible (especially for the angles and object identity). One should note, that the dimension of the model output is smaller than the raw image data by two orders of magnitude. We also checked that the nonlinear expansions without SFA reduces the regression performance to almost chance level.

Since the predicted SFA solutions are cosine functions of the position values (see [9]), 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 those SFA solutions that code for rotational angles, the theory predicts both sine and cosine functions of the angle value. Therefore we did regressions for both mappings and then calculated the angles via the arctangent. If multiple objects are trained and separated with a blank (see 2.1), the solutions will in general be object dependent. This rules out global regressions for all objects. The easiest way around this is to perform individual regressions for all objects. While this procedure sacrifices the object invariance it does not affect the invariance under all other transformations.

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, thereby separating all objects. To explore the potential classification ability of the model we applied two very simple classifiers on the SFA-output: a k-nearest-neighbour and a Gaussian classifier.

3 Results

The stimulus set used for testing consisted of 100,000 views (about 10,000 per object), which were generated in the same way as the training data. Half of this data was used to calculate the regressions or to train the classifiers, the other half for testing[2].

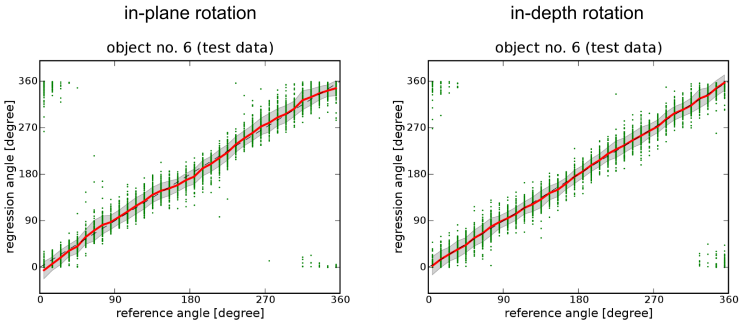

Fig. 3. Regression results for position and angle. For the first untrained sphere object the feature values that were calculated with linear regression are plotted against the correct reference values. The green dots are data points, the red line is the mean and the gray area shows ± one standard deviation around the mean value. The regression of the x coordinate was based on all five training objects.

Position and Rotational 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. As one can see in Figure 3 and Table 1, this works well for the sphere objects, with a standard deviation of 5% for trained objects and 7% for untrained objects. For the trained fish we achieve the same performance as for the trained sphere objects. For untrained fish the additional size variations from the z-transformations take their toll and pull the performance down to 14% for the y-coordinate.

[2] The detailed results are available at www.nikowilbert.de/supplements.

Table 1. Standard deviations for the coordinate regressions. The values are given in percent relative to the coordinate range. The chance level is 28.9%. For example, the x-coordinate of trained sphere objects on test data was reconstructed with an RMSE of 5%.

	Spheres		Fish	
	trained	new	trained	new
x	5%	7%	5%	12%
y	5%	7%	5%	14%

Table 2. Mean standard deviations for the angles and the z coordinate over all ten objects. For the fish we give in ϕ_y^* the mean absolute error for ϕ_y, since a large part of the error is systematic due to the 180° pseudo-symmetry. The chance level is 104°.

Spheres	ϕ_z	ϕ_y	Fish	ϕ_y	ϕ_y	z
	14.8°	15.8°		36.4°	23.4°	0.11%

To extract the in-plane and in-depth rotation angles for the spheres, 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, both rotational angles were extracted with about 15° standard deviation.

For the fish, the standard deviations are about twice as large as for the spheres, mostly due to systematic errors. The fish models have a very similar front and back view and therefore the model has difficulties to differentiate between those two views. When taking the mean absolute error instead (see Figure 2), the performance gap to the spheres is smaller because of the reduced influence of outliers. As for the angles, individual regressions for all objects were used to calculate the z coordinate for the fish objects (z-transformations were not used for the sphere objects).

Classification. To quantify the classification ability of the model, two classifiers were used on the SFA-output. The classifiers were trained with about 5000 data points per object (i.e., half of the data, as for the regressions). A random fraction of the remaining data points (about 150 per object) was then used to test the classifier performance. The k-nearest-neighbour classifier generally performed with about 96% hit rate (see Table 3). As expected, the Gaussian classifier performed not as well, with a hit rate between 88% and 96%.

Table 3. Classifier hit rates in percent. The columns labeled with "KNN" refer to the k-nearest-neighbour classifier ($k = 8$), while those with "G" refer to the Gaussian classifier. Chance level is 10% for all objects and 20% on the sets of training or test objects.

	trained objects		test objects		all objects	
	KNN	G	KNN	G	KNN	G
spheres	96.4	88.3	96.6	95.7	95.0	89.2
fish	97.7	94.6	99.2	88.1	96.7	88.8

4 Discussion

In the previous section we have shown how the hierarchical model learns independent representations of object position, angle, and identity from quasi-natural stimuli. The simultaneous extraction of all these different features is one novelty of our model. This capability is made possible by the invariance of one extracted feature with respect to the others (only limited with respect to object identity).

Related Work. The slowness principle has been applied in many models of the visual system in the past [11, 12, 13, 5, 6, 2]. VisNet is the most prominent example of hierarchical feed-forward neural network models of the primate ventral visual system [1]. Weights in this model are adapted according to the trace rule [11, 14, 15], which is closely related to Slow Feature Analysis [8]. 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. The ability of our model to code for more than object identity in a structured way is therefore the greatest functional difference.

Conclusion and Outlook. 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 or seldom are formed, while information on other transformations that occur most often or quickly are discarded. An advantage of such a system is that no invariances have to be "hard-wired". Instead we use only generic SFA units on all network layers. The unsupervised learning of the system can be 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. In this case a final step of linear regression or a simple classifier reconstructs 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, like for two simultaneously presented objects, cannot be predicted with our current theory. The performance of the network for cluttered background also remains to be tested.

References

[1] Rolls, E.T., Deco, G.: The Computational Neuroscience of Vision. Oxford University Press, New York (2002)

[2] Franzius, M., Sprekeler, H., Wiskott, L.: Slowness and sparseness lead to place, head-diretion and spatial-view cells. Public Library of Science (PLoS) Computational Biology **3**(8) (2007) e166

[3] Picard, R., Graczyk, C., Mann, S., Wachman, J., Picard, L., Campbell, L.: Vision texture. Downloaded from `http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html` (2002)

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

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

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

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

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

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

[10] Berkes, P., Zito, T.: Modular toolkit for data processing (version 2.0). `http://mdp-toolkit.sourceforge.net` (2005)

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

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

[13] Kayser, C., Einhäuser, W., Dümmer, O., König, P., Körding, K.: Extracting slow subspaces from natural videos leads to complex cells. Artificial Neural Networks - ICANN 2001 Proceedings (2001) 1075–1080

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

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