Quantitative comparison of stable representations of natural stereo-images and binocular complex cells

Bachelor's Thesis

Sebastian Bitzer (sbitzer@uos.de)
Cognitive Science Program
University of Osnabrück

Supervised by
Professor Peter König

November 27, 2004

Abstract

Complex cells in primary visual cortex exhibit particular spatial properties such as tuning for stimulus orientation and spatial frequency independent of precise stimulus position in the receptive field. Recently it has been shown that these neurons share important such properties with simulated cells, which were adapted to exhibit optimally stable activity to natural visual stimuli. Consequently, it has been suggested that complex cells can be described as forming optimally stable representations of their natural input.

The simulation from this study has now been extended into the 3D domain by optimising activities of simulated cells to binocular cat-cam video sequences of natural scenes and in the work at hand I compare properties of the resulting stable cells to those of complex cells in order to evaluate whether a similar conclusion as from the simulation with monocular stimuli can be drawn. Thereto I directly analyse subunit receptive fields, examine binocular interaction profiles and investigate responses to random-dot stereograms while mainly contrasting my findings to physiological results obtained by the groups of Ralph Freeman and Andrew Parker.

Similar to the monocular study I find that simulated and real binocular neurons have many properties in common. Stable cells again exhibit tuning for orientation and spatial frequency as well as position invariance. Furthermore, all of the simulated cells are disparity selective and their disparity tuning curves show characteristics comparable to those of striatal neurons. Moreover, a predominance of phase encoding can also be observed in both systems.

Nevertheless, I notice several differences between stable and complex cells. Many of them are due to the limitations of the cell model employed in the simulation, which for example does not allow consistent inhibitory input from one eye like it is found in some real neurons. One crucial discrepancy, which I cannot explain, is a greater number of stable cells preferring large phase disparities near $\pm\pi$. I propose that this may result from properties of the used natural stimuli.

In conclusion, I find striking similarities between stable and complex cells signifying that *binocular* complex cells can be described as forming optimally stable representations of natural visual input as well. However, the cell model should be adapted to incorporate recent advances in the modelling of striatal neurons and further investigations need to be done to explain open discrepancies concerning phase disparity.

Contents

1 Introduction

Cognition is one of the most amazing phenomena in biology. In the neurosciences we assume that cognitive functions such as seeing are implemented by the neural circuitry in the brain. This leads to the questions how this circuitry is formed and how single neurons contribute to it.

Due to the limited amount of information, which can be encoded in the genome of an organism, the details of neural organisation have to be learnt from the environment. Therefore in order to understand the circuitry in the brain it is important to know the relation between neurons and their natural stimuli. Accordingly, it has been shown that certain neuronal activities can be described as forming representations of natural stimuli which are optimally sparse (Olshausen and Field, 1996) or optimally stable (Körding et al., 2004).

Here I quantitatively analyse whether activities of complex cells in primary visual cortex (V1) to binocular stimuli can be described as forming optimally stable representations of these stimuli, too. Thereto, I compare properties of optimally stable cells obtained from a simulation to those of striatal cells on three different levels of analysis. I begin with a direct analysis of complex cell constituents, which are then reinvestigated indirectly via response properties of the cells and finally I examine activities of cells to binocular stimuli themselves.

The following two subsections introduce the topic of binocular cells in V1 and the simulation, which has been used to produce optimally stable representations of natural stimuli. In further sections I present my analysis and discuss these results.

1.1 Binocular complex cells in primary visual cortex

The studies mentioned in the following text have been done with cats or primates. I am interested in the more abstract properties of neuronal activation and assume that these are comparable in the early stage of cortical processing which I am looking at in the two species. This this goes along with my central hypothesis that neuronal activities are adapted to natural stimulus statistics, which should be roughly equal for these two species, too.

Already Hubel and Wiesel (1962) showed in their fundamental work that cells in V1 can be classified into two distinct categories according to their firing properties to bar or grating stimuli. Whereas so-called simple cells selectively respond to bars of a certain orientation, spatial frequency, velocity and contrast polarity at a certain position in their receptive field

(RF), so-called complex cells are invariant to local contrast and stimulus position while still maintaining specificity to spatial frequency, orientation and velocity.

Additionally, large subsets of both classes of cells are selective to binocular disparity, which is possible, because V1 is the first stage in the visual pathway where neurons receive significant input from left and right eye. Binocular disparity describes small differences between images of stimuli on the two retinae which result from the horizontal separation of the two eyes. Projections of stimuli outside of the fixation point will not fall on corresponding points of the retinae when the stimuli lie in different depth. These deviations from corresponding points are defined as disparity. Here I follow the common convention that far stimuli have positive disparity and near stimuli have negative disparity (see figure 1 for illustration).

Binocular simple cells can often be modelled as a linear binocular filter followed by a static nonlinearity (Anzai et al., 1999b; Chalupa and Werner, 2003; Freeman and Ohzawa, 1990). This means that it is possible to easily predict the response of a simple cell to binocular stimuli from the responses to monocular stimuli. However, this is not true for complex cells which exhibit substantial nonlinearities in their responses to binocular stimuli (DeAngelis and Anzai, 2003). Accordingly, binocular interaction profiles which describe the activity of cells to bars presented independently to the two eyes[1], show elongated regions along lines of constant disparity (Ohzawa et al., 1990, 1997; Anzai et al., 1999c), that is, these cells are activated for stimuli at all positions in their receptive field as long as the disparity is appropriate.

Hubel and Wiesel (1962) proposed a hierarchical model in which several simple cells with different optimal stimulus positions and else equal RF properties feed into one complex cell and thereby make it position invariant. But it has long been noted that it need not be actual simple cells, which constitute the input to complex cells in order to produce the reported nonlinearities.[2] Thus Ohzawa et al. (1990, 1997) suggest a more abstract hierarchical model in which the outputs of functional subunits are combined to produce the activity of a complex cell. This so-called binocular energy model consists of four binocular subunits with simple cell-like receptive fields. Each subunit is modelled as a linear binocular filter followed by a half-squaring nonlinearity. The first and second and the third and fourth

[1] for further explanations and examples see section 3.2

[2] see Mechler and Ringach (2002) for a criticism of classification into simple and complex cells in general

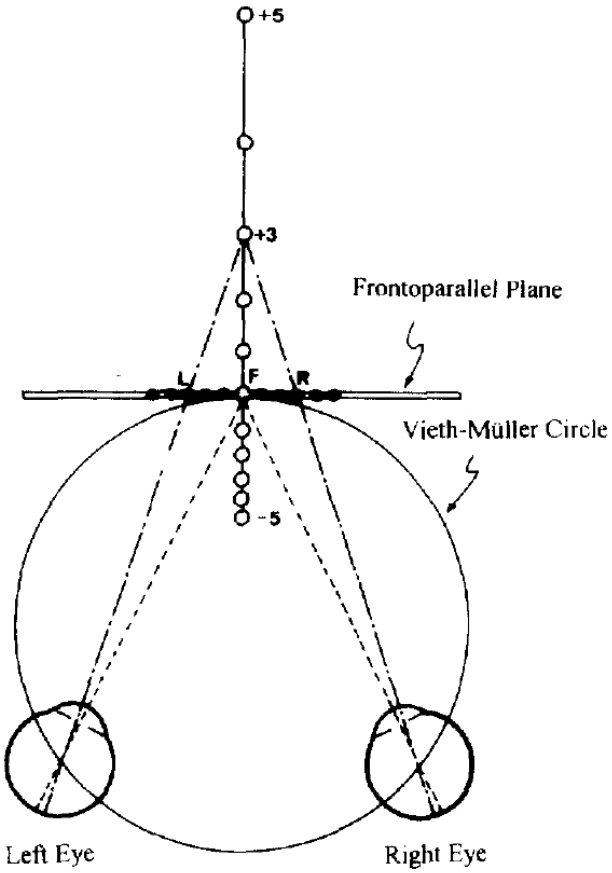

Figure 1: Schematic depiction of the geometry of horizontal binocular disparity; Images of fixation point F fall on corresponding points of the retinae. The difference between corresponding point and actual point of a stimulus projection onto one retina is defined as disparity. Thus in the fixation point there is zero disparity, stimuli behind the fixation point (far) are defined to have positive disparity and stimuli in front of the fixation point (near) have negative disparity. Near and far stimuli can be simulated by shifting left and right stimulus presentations on a frontoparallel plane (2D) instead of shifting in depth (3D). Far stimuli are sometimes said to produce uncrossed disparities and near stimuli to produce crossed disparities, because for near stimuli these left and right shifts have to cross each other. Vieth-Müller circle: theoretical circle consisting of all points whose images fall on corresponding points of both retinae; figure adapted from Gonzalez and Perez (1998)

subunits can be seen as units themselves. The subunit RFs within each of the two units are sign-inverted versions of each other while RFs across units stand in quadrature phase relationship which means that the phases of the underlying Gabor functions[3] are 90° apart. Thereby the sign-inversion leads to contrast invariance while the quadrature relationship results in position invariance. Left and right RFs of subunits have equal preferences for orientation and spatial frequency and differ only in properties responsible for disparity tuning of the cell. This is in good agreement with physiological findings (Hubel and Wiesel, 1962; Ohzawa et al., 1996).

For simulations the four-subunit model can be substituted with an equivalent two-subunit model in which every unit consists of only one subunit, which does full-squaring instead of half-squaring. An example of such a binocular energy model is depicted in figure 2A.

Two possible mechanisms for encoding disparity have been identified (Ohzawa et al., 1996, 1997; Anzai et al., 1999a) and are demonstrated in figure 2B. The position model assumes that the structure of left and right RFs is the same, but that the position of the two RFs is different and this difference then defines the preferred disparity of the cell. The phase model, on the other hand, assumes that the positions of the RFs are equal, but that RF structures differ in the two eyes by a given phase shift which in turn also defines optimal disparity. Evidence for both mechanisms has been found in physiological studies (Anzai et al., 1999a,c) and indeed the response of some cells can only be explained by a combination of both mechanisms (Prince et al., 2002a).

In my analysis I will present some of the studies mentioned here, especially Anzai et al. (1999c) and Prince et al. (2002a), in more detail in order to compare their findings with data obtained from our simulation.

1.2 Optimally stable representations of natural visual stimuli

Körding et al. (2004) report that optimally stable representations of natural visual stimuli resemble many properties of complex cells very well. These properties include spatial frequency and orientation tuning and invariance to stimulus translation and contrast polarity. Selim Onat extended their framework to incorporate binocular stimuli and I assess his results about whether optimally stable representations can describe binocular properties of cells in V1, too. Details of the following brief description of the simulation can be read in Onat et al..

[3] for specification of Gabor functions see section 2.1.1

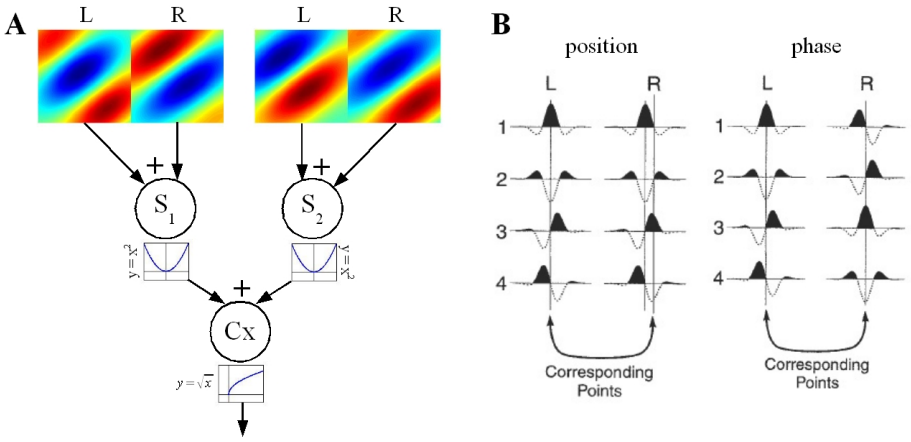

Figure 2: Energy model and disparity encoding. **A** Adapted binocular energy model used in our simulation. The model consists of two subunits which are binocular linear filters followed by a static nonlinearity (full-squaring). The output of the complex cell is a linear combination of subunit outputs followed by a square root. For the energy model to act as a complex cell Ohzawa et al. (1990, 1997) proposed that the subunit RFs stand in quadrature phase relationship as depicted here. Further left and right RFs should be equal except for properties defining selectivity for disparity. **B** Mechanisms of encoding disparity. Exemplary 1D profiles of RFs taken along a line orthogonal to orientation are shown. In position model left and right profiles are equal but at different positions. In phase model left and right profiles are at corresponding positions, but have different shapes (profiles with phase shift). figure B adapted from Ohzawa et al. (1997)

Natural visual stimuli have been obtained by recording image sequences from two cameras which were mounted in parallel on the head of a freely moving cat. This has yielded a pair of left and right natural images for each frame of the video whereby the two images differ a little bit due to the spatial separation of the two cameras (4.8cm). Because the theoretical fixation point of the two cameras lies in infinity, only near disparities should have been sampled, but a normalisation technique has been used to obtain all kinds of disparities. Always four 20x20 pixel (px) sized patches have been extracted from a randomly chosen position within the raw images (left and right patch of frame t and left and right patch of frame $t+1$, see figure 3). For computational efficiency reasons these patches have not been the input for the simulation, but their representations in the space of their first 100

principal components[4], which already explained $> 95\%$ of their variance.

Figure 3: Description of stimuli. **A** Setup of the cameras on a cats head. **B** Example of a raw image recorded with one camera. Image is converted to greyscale before further processing. **C** Schematic representation of choice of patches for one stimulus configuration within optimisation. From left and right images of frame t and $t+1$ one patch is chosen from a randomly drawn common position.

The simulation is based on the cell model as depicted in figure 2A which I already introduced above. Thus the activity of a simulated neuron is calculated in the first step for the first subunit by taking the inner product between left patch, I^l, and left weights, W_1^l, and right patch, I^r, and right weights, W_1^r. Further steps are defined by

$$A = \sqrt{(W_1^l I^l + W_1^r I^r)^2 + (W_2^l I^l + W_2^r I^r)^2}. \tag{1}$$

This definition of activation allows for many different response properties depending on the weights W, which I will denominate RFs of the simulated neurons in the following. As mentioned earlier Ohzawa et al. (1990) proposed that the simulated cells are complex-like, if the RFs of the subunits resemble two-dimensional Gabor functions which stand in quadrature phase relationship. I will examine this property in section 3.1.

A given set of 100 cells should act optimally stable which means that the activity of a cell should change as little as possible within one time step. This objective is motivated by the fact that relevant features within the visual field, like certain objects, do not change as rapidly as low-level features like luminance in a limited region of the visual field. To obtain stable representations of the stimuli the activity of the cells to the given patches has been optimised with respect to an objective function. A cell,

[4]actually components 2-101: mean has been taken out

which does not react to any stimulus, would be most stable. Obviously this is not wanted and to prevent this a term has been included in the stability objective, which reinforces some variability in the overall responses of the cell. Furthermore a decorrelation term has been utilised to penalise common responses of the cells, what should lead to different response patterns for each cell. Thus, in the simulation the following objective function has been maximised

$$\Psi = \underbrace{-\sum_{i=1}^{N} \frac{\langle (A_i(t) - A_i(t+\Delta t))^2 \rangle}{\sigma_{ii}}}_{\Psi_{stable}} \underbrace{- \frac{1}{(N-1)^2} \sum_{i \neq j} \sigma_{ij}^2}_{\Psi_{decorr}} \tag{2}$$

where N is the number of cells, $\langle\rangle$ denotes the average over stimuli and thus over time, $A_i(t)$ is activity of neuron i at time t and σ_{ij} is the covariance over stimuli between cells i and j.

Weights for each subunit of each simulated neuron have been obtained as a result of the converged optimisation process, which optimally stable represent (binocular) properties of the shown natural stimuli. In the following I have analysed these artificial RFs in several ways as explained in the methods section (2) to attain quantitative descriptions of RF properties comparable to physiological data.

2 Methods

Different physiological studies often employ different analyses while investigating the same or very similar subjects. Thus, for comparison of our simulation data with a wide range of physiological data I had to apply several different analysis techniques which are described in detail below. A basic RF analysis has been done by fitting 2D-Gabor functions to the RFs of subunits. Further I estimated the disparity tuning properties of the simulated neurons with random-dot stereograms (RDS) and fitting of the resulting tuning curves with 1D-Gabor functions. At last I obtained binocular interaction profiles of our cells, which I decomposed into components by a singular value decomposition, whose properties I in turn analysed with 1D-Gabor fits. All programming has been done in MATLAB® (The MathWorks, Inc.).

2.1 Fitting 1D and 2D-Gabor functions to disparity tuning curves and RFs, respectively

Curve fitting can be used to get insights in the given data by describing the data with parameters of a certain model. My data are one dimensional disparity tuning curves and two dimensional RFs and I want to express these in terms of Gabor functions, which are nonlinear in their parameters. Therefore, I use a nonlinear least squares method to fit the Gabors to my data. Details of the fitting procedure can be found in the corresponding sections below.

2.1.1 Gabor functions

Gabor functions are defined as the product of a Gaussian and a sinusoidal component (Gabor, 1946). For 1D-Gabor functions I use the definition

$$G(x) = A_o + A \cdot \exp\left(\frac{-(x-x_0)^2}{2W^2}\right) \cdot \cos(2\pi f(x-x_0) + \phi), \tag{3}$$

where A_o is the amplitude offset, A is the amplitude, x_0 is the centre of the Gaussian envelope, W is the width of the Gaussian, f is the frequency of the sinusoid and ϕ describes its phase. In two dimensions another Gaussian envelope is added. Thus, along with the definition of the centre in X- (x_0) and Y- (y_0) coordinates, two width parameters are needed - one for the minor axis (W_p) and one for the major axis (W_q). Further, parameters θ and γ are used to determine the orientation of the sinusoid and the Gabor

envelopes, respectively. This yields for 2D-Gabor functions

$$G(x,y) = A_o + A \cdot \exp\left(\frac{-p^2}{2W_p^2}\right) \cdot \exp\left(\frac{-q^2}{2W_q^2}\right) \cdot \cos(2\pi f u + \phi), \qquad (4)$$

with

$$p = (x - x_0)\cos\gamma + (y - y_0)\sin\gamma,$$

$$q = -(x - x_0)\sin\gamma + (y - y_0)\cos\gamma,$$

and

$$u = (x - x_0)\cos\theta + (y - y_0)\sin\theta.$$

These functions match the definitions as given in Anzai et al. (1999a) or Prince et al. (2002b). Preliminary experiments have shown that allowing γ to vary freely only marginally improves fits of RFs over fits in which γ was set to equal θ. Therefore I simplify the 2D-Gabors by postulating that $\gamma = \theta$, which additionally has computational advantages. Correspondingly, I will only write θ below.

For an example of 1D- and 2D-Gabor functions see figure 4. To gain a better understanding of the influences of each parameter in 1D-Gabors I have written a Matlab demo, which is a graphical interface allowing for interactive manipulations of all parameters of a 1D-Gabor function.

2.1.2 Nonlinear least squares and Levenberg-Marquardt

In the nonlinear least squares method the difference between fitted curve and data is minimised. This difference is defined by the χ^2 function, which depends on the values of the parameters $\boldsymbol{p} = (p_1, \ldots, p_k)$

$$\chi^2(\boldsymbol{p}) = \sum_{i=1}^{N} \left(d_i - y(x_i; \boldsymbol{p})\right)^2,$$

where N is the number of data points, d_i is the data value at point x_i and y is the value of the model with parameters $\boldsymbol{p}$ at this point. Usually the addends of the χ^2 function are weighted by the variance of the data points such that poorly approximated points do not influence the fitting procedure as much as confident points do. However I omit the weighting term, because the implementation of the fitting algorithm mentioned below does not support weights[5], or variances of data points are not given (RF and SVD fits).

[5] this is a problem I did not solve, but first comparisons with fits of another program, which supports weighting, revealed no crucial differences

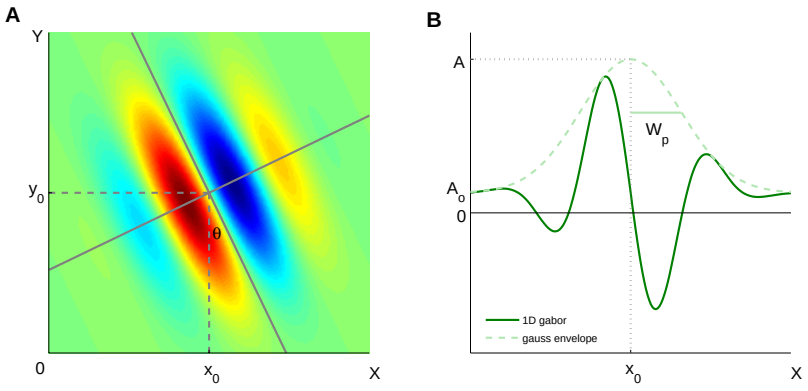

Figure 4: Gabor functions. **A** 2D-Gabor function. The function is the product of two Gaussian envelopes and a sinusoidal component. The envelopes determine the width of the function along and orthogonal to the orientation which is determined by θ (here $\theta = -\pi/7$). Centre lies at (x_0, y_0). **B** 1D-profile of the function shown in A along the line orthogonal to the orientation. The result is a 1D-Gabor function which is the product of the minor Gaussian envelope with the sinusoidal component. Width of the Gaussian is determined by W_p and amplitude by A. A small amplitude offset A_o has been introduced to illustrate that parameter. Phase in this example is $\phi = \pi/2$

I use the popular Levenberg-Marquardt method (Press et al., 1992) to find a minimum of the χ^2 function. This algorithm is implemented in the MATLAB® function `lsqcurvefit`, which also calculates values of the χ^2 function itself. Thus the crucial work in fitting Gabors lies in properly defining the function to fit and in finding good starting values for the optimisation.

According to whether I want to fit one dimensional or two dimensional data I apply the Gabor definitions as given above.

2.1.3 Finding good starting values

Good starting values are essential for the optimisation to converge to a local minimum. This becomes more important with more complex functions like the 2D-Gabor or the square root 1D-Gabor. Therefore I developed heuristics to estimate the values of critical parameters before optimisation. Phase, position and width of the Gabor functions turned out to be noncritical in most cases, that is, even with poor estimators of these parameters the

optimisation tends to converge as long as the other estimates are good.

In the 1D case I determine the frequency of the underlying Gabor-function with a discrete Fourier transform of the data by setting this frequency to the one with the most power in the resulting power spectrum. This estimate is so good that further optimisation of frequency is not needed. Consequently I fix this parameter to the estimated value during optimisation. In addition this procedure prevents that changes in the Gaussian component are cancelled by a change in frequency (see Prince et al., 2002b, for an example). The heuristic for the amplitude of the 1D-Gabor depends most importantly on the standard deviation of the data and the heuristic for the amplitude offset on the mean, or, in combination with tuning curves, on the values of the first and last data points, because these points estimate the activity to uncorrelated stimuli in the two eyes, which should be the baseline of the tuning curve.

When finding starting values for 2D-Gabor functions I rely on scripts which have been written before for estimating the centre of the 2D data (receptive fields) and the frequency. Again the frequency is fixed to a value determined with fast Fourier transformation. Additionally I fix the amplitude offset to 0 during optimisation, which is the mean value of the RFs. The amplitude is estimated by the maximum value of the data and orientation is found by correlating differently oriented, sinusoidal gratings with the RFs. The orientation, whose grating correlates best with the RF, is taken as an estimator.

In the case of binocular data I use the optimal parameters found for the left data as starting values for the right data. This ensures that both fits obtain comparable parameters.

2.1.4 Goodness of fit

I have three criteria, which together specify goodness of a fit. First, the optimisation must converge to a solution. Second, I calculate the R^2 value of the fit. And third, I determine confidence intervals for the fitted parameters.

Most of the time optimisation finds a minimum of the χ^2 function with starting values as described above. If this is not the case, I refit the data with starting values, which are carefully chosen by hand. Thus all fits reported here are the result of a converged optimisation process.

The R^2 value gives an estimate for how much of the data variance is explained by the fit. Here I use a degree of freedom adjusted definition of R^2 in which the sum of squared errors (SSE) of the fit is related to the sum of squares around the mean (SST) of the data. If the SSE is small compared

to the SST, then the fit is considered good and hence explains lots of the variance of the data. The following definition gives 1, if 100% of the variance is explained

$$R^2 = 1 - \frac{(n-1)\mathrm{SSE}}{(n-m)\mathrm{SST}} = 1 - \frac{(n-1)\sum_{i=1}^{n} w_i(d_i - y_i)^2}{(n-m)\sum_{i=1}^{n} w_i(d_i - \bar{d})^2},$$

where n is the number of data points, m is the number of fitted parameters, $w_i = 1/\sigma_i^2$ weights each addend with the variance of the data point, d_i is the data value, $\bar{d}$ is the mean over all data points and y_i is the value of the fitted curve.

Confidence intervals for the fitted parameters are estimated as given in Press et al. (1992). For this I evaluate how the first partial derivatives of the fitted function with respect to different parameters interact over all data points, what leads to the covariance matrix of the fitted parameters. I then extract standard errors for the parameters from there and determine 95% confidence intervals as the region of two standard errors around the parameter value. Unfortunately, these confidence intervals are quite large in comparison to the fitted values. For example, the maximum of the absolute value of fitted positions over all 100 (1D) tuning curves is 4.8 whereas the minimum of 2·standard error is 6.2. This makes clear that almost all fitted positions lie within every single confidence interval and renders these intervals seemingly useless. To overcome this flaw I included second partial derivatives in my calculations, which should have increased the accuracy of the estimated confidence intervals. Although this increases the range of observed interval sizes (minimum is now circa 0.8), many confidence intervals remain large (median 7.3). So, for all fits, except those where a 1D Gabor function with fixed frequency is fitted, I only consider first partial derivatives, because the gain of second partial derivatives does not match the increase in complexity. Nevertheless, the size of the confidence intervals still gives a good qualitative description of the accuracy of the estimated parameter values, which means that a fit with comparably large confidence intervals often also has questionable parameter values. This applies especially to tuning curves with very low frequency for which it is hard to determine the correct position (see figure 5C for an example).

2.2 Testing RFs with random dot stereograms

Prince et al. (2002a,b); Read and Cumming (2003) and Cumming (2002) use random-dot stereograms (RDS) to test neurons in V1 for disparity tuning. In order to compare my artificial cells with their findings I do the same.

Random-dot stereograms consist of bright and dark dots at random positions on a grey background. So they allow testing for disparity tuning independent of orientation, spatial frequency or exact position of the stimulus, because they contain complete spectra of these parameters and are correspondingly called broadband stimuli.

2.2.1 Generating RDS

To produce RDS I first generate one large random-dot stimulus, which contains an equal number of white (value = 1) and black (value = -1) dots placed at randomly drawn positions. All other parts of the stimulus get a value of 0. In my computations I use the smallest possible dot sizes of 1 pixel, but dot sizes may take larger values as long as these are expressed in multiples of one pixel. Thereby it is possible that one dot covers another dot partially or fully. The actual number of generated dots is controlled by a given dot density, which expresses in % how much of the RDS should be covered with dots. For different comparisons with reported data I use either a dot density of 25% or 50%, but results of both are very similar. The final step in producing a random-dot stimulus is then the subtraction of the mean from all values such that the mean of the stimulus is 0.[6]

From the large random-dot stimulus I extract the RDS according to a given disparity. Thereto I define one region of the large stimulus as a reference (right) and get the other region (left) by moving some value, which is determined by the given disparity, away from the reference. As long as these regions have some overlap they represent some disparity, otherwise they are uncorrelated.

I use two forms of disparity. In the horizontal disparity condition I move the left region only on the X-axis. This is depicted for one RDS in figure 5B. There a RDS with -3 pixel disparity is shown, which represents a near stimulus. Alternatively one can define disparity orthogonal to the RF orientation of the cell. In this case I shift the left region along a line orthogonal to this orientation instead only along the abscissa. I define orthogonal disparity such that if the orthogonal line is horizontal, then a negative disparity would result in a left shift of the left region and therefore would represent a near stimulus, exactly as it is the case in the horizontal condition.

[6] this is only needed in the case that, e.g. through covering, more bright or dark dots exist

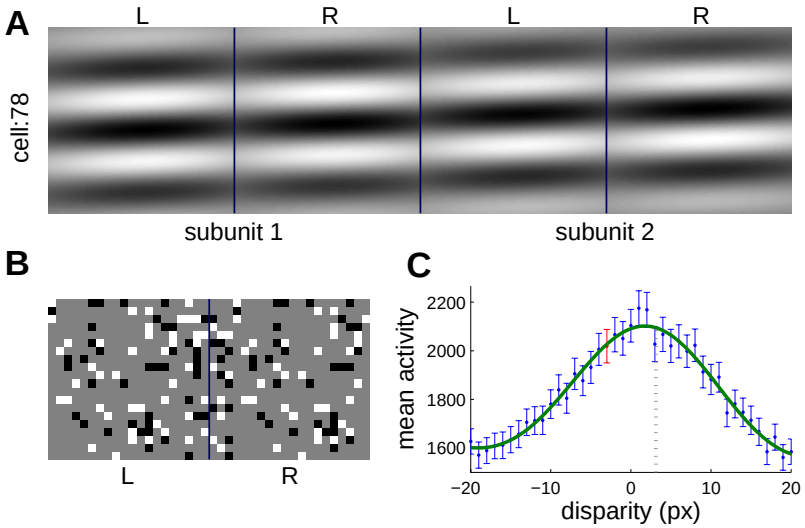

Figure 5: Activity of cell 78 to random-dot stereograms (RDS) with different disparities **A** RFs of cell 78 with orientation slightly less than $\pi/2$ (orientation defined as shown in fig. 4A). **B** Example of one RDS with 25% dot density and horizontal disparity of size -3 (pixels), which corresponds to a crossed disparity (near) and a shift of the left stimulus 3 pixels to the left with respect to the right stimulus. A disparity orthogonal to the RF orientation would mean that the left random-dot stimulus would be shifted 3 pixels up in this case. **C** Mean activities to RDS with different horizontal disparities fitted with a 1D-Gabor function. Error bars depict estimated 95% confidence intervals for the plotted means, red data point shows activity to RDS with disparity as shown in B. Although RF frequency is high, frequency of the tuning curve is very low. This makes it hard to determine the position (dotted line) of the fitted Gabor with high confidence (95% confidence interval for position $= 3.12 \pm 63.11$ px, $R^2 = 0.97$).

2.2.2 Activities to RDS

I "show" these generated RDS, which are of the same size as the RFs (20x20 pixel one eye), to the simulated cells in order to estimate their disparity tuning curves. These tuning curves describe the activity of a cell to certain disparities, while the activity of a cell to a stimulus is defined as in equation 1 and figure 2A. Thereby the probabilistic nature of RDS forces one to average the activities to sufficiently many RDS with the same disparity in order to obtain a meaningful estimate for disparity tuning. In my case 1000 different RDS suffice to get useful data points which can be fitted with a 1D-Gabor function[7]. Data points of one such tuning curve are mean activities to RDS in all disparities possible with resolution of one pixel, that is disparities $-20, -19, \ldots, 0, \ldots, 20$ where -20 and 20 already represent uncorrelated stimuli, because left and right stimulus have no overlap. Additionally, I estimate monocular activities to random-dot stimuli by showing a blank stimulus (all values equal) to one of the eyes and a random-dot stimulus to the other. Hereby the activity to a blank stimulus is always zero.

It is known that the variance of cell activities rises approximately proportional with its mean. Prince et al. (2002b) propose to take the square root of all computed activities before further processing to normalise variances, but this transform is not appropriate in my case (see 3.3.1). Thus I directly weight data points with their variance in the calculation of statistical measures, which assume equal variances.

To evaluate the tuning for disparity of the artificial cells I use the disparity discrimination index (DDI) (Prince et al., 2002b). This measure takes into account the amplitude of disparity tuning and additionally the variance of the tuning curve in order to describe how good a cell can discriminate between its preferred disparities. Therefore it can be defined in terms of maximum mean activity of the cell (a_{max}) and corresponding minimum (a_{min}) as follows

$$\mathrm{DDI} = \frac{a_{max} - a_{min}}{a_{max} - a_{min} + 2 \cdot \mathrm{RMS}} \tag{5}$$

The root mean squared error (RMS) is the square root of the summed variances around the data points across the tuning curve

$$\mathrm{RMS} = \sqrt{\frac{\mathrm{SSE}}{r - d}} = \sqrt{\frac{\sum_{i=-20}^{20} \sum_{j=1}^{r} (a_{ij} - \bar{a}_i)^2}{r - d}}$$

[7] Most means are 14-16 times larger than their 95% confidence intervals (figure 5C). The relation between confidence interval of the means, c, and number of presented RDS, r, is quadratic: $c/n \to r \cdot n^2$

where r is the number of different RDS used with each disparity (250), d is the number of different disparities (41), a_{ij} is activity to RDS j with disparity i and $\bar{a}_i$ is the mean activity to RDS with disparity i. Prince et al. (2002b) performed all of these calculations on the square root of activities, that is, on more or less normalised variances. A corresponding operation on my data would be to weight the addends of the SSE with the variance of $\bar{a}_i$, but this eliminates the influence of all variances on the DDI nearly completely, what contradicts the idea of the DDI. Thus the SSE remains unweighted.

Monocular activities (a_L, a_R) are used to compute the ocular dominance index

$$\text{ODI} = \frac{a_L}{a_L + a_R} \tag{6}$$

where in physiology monocular activities are taken relative to the recording site, that means instead of left and right one would take ipsilateral and contralateral eye. Cells with ODI near 0.5 are well balanced between left and right eye and hence are said to be binocular.

2.3 Obtaining disparities from SVD components of binocular interaction profiles of RFs

It is not possible to directly measure the activity of subunits underlying the response of complex cells. To get an approximation of the characteristics of the subunits anyway, Anzai et al. (1999c) calculated SVD (singular value decomposition) components of binocular interaction profiles and argued that these resemble well binocular interaction profiles of hypothetical subunits. For a direct comparison of their data with mine I retrace their calculations with my cells.

2.3.1 Binocular interaction profiles

Anzai et al. (1999c) measure binocular interaction RFs with spatiotemporal white noise generated according to binary m-sequences. The resulting profiles can be interpreted as the mean response of a cell to two bars with optimal spatial frequency and orientation, which are presented in different positions to each eye (in each eye one bar). They further define the interaction profile to be the difference of responses to bars with equal contrast in both eyes (match) and responses to bars with opposite contrast in the two eyes (mismatch) (see also Ohzawa et al., 1997).

I can calculate corresponding interaction profiles directly from the 2D-Gabor fits of the simulated RFs. Thereto I assume that a presented bar has either value 1 (bright) or -1 (dark). Then the activity to a bar with optimal orientation can be computed according to the cell model from the projection of the two dimensional fit to a one dimensional line orthogonal to the preferred orientation at the centre of the fit. Such a projection is already illustrated in figure 4. The value of this 1D-profile at the position of the hypothetical bar in the left eye and in the right eye corresponds here to $W^l I^l$ and $W^r I^r$, respectively, from equation 1 (definition of activity), which is then further processed according to the cell model to yield the activity of the cell to matching contrast bars at position X_L in the left eye and position X_R in the right eye. Thereby these positions are defined relative to the centre of the 2D-Gabor fit and $X_L = 0$ means that the bar in the left eye is presented in the centre of the RF. If the centre of a fit lies more than 2.2 pixel away from the centre of the patch[8], then the centre of the patch is chosen as reference instead, because otherwise no complete interaction profile could be produced. The value 2.2 is selected such that this reset of RF centres only happens to 2D-Gabor fits with very low frequencies for which it is hard to determine the correct centre, anyway. In this way I achieve a range of positions from -7.3 to 7.3 pixel. At the end of the process I again get 2D-profiles, this time in the dimensions of positions of bars in the RFs.

The only difference in the calculation of activity to non-matching contrast bars is that the 1D profile of one eye is inverted[9]. Thus instead of summing the weighted sum of inputs from both eyes in the activity equation, I subtract one weighted sum from the other in the mismatch condition.

The binocular interaction profile is then produced as defined by Anzai et al. (1999c) by subtracting non-matching contrast profile from matching profile, which is illustrated in figure 6.

2.3.2 Calculating disparities from SVD components

Subsequently, a singular value decomposition (SVD) is conducted by using the built-in MATLAB® function `svd`. The SVD finds components of the original binocular interaction profiles, which are mutually uncorrelated and weights them according to the influence each component has on the original profiles, or, put another way, weights them according to the amount of

[8] the patch containing the 2D-Gabor has a size of 20x20 pixel, its centre in terms of the Gabor function is (10.5,10.5)

[9] one bar has value -1, so it inverts the weights e.g. in $W^r I^r$

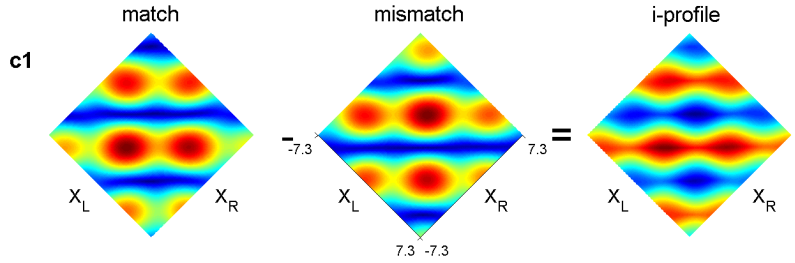

Figure 6: Calculation of binocular interaction profile from interaction profiles of bars with matching contrast and bars with non-matching contrast on the example of cell 1. X_L and X_R describe the position of left and right bar with respect to the RF centre and range between -7.3 and 7.3 pixel. Colours represent activity to a given configuration of bars. The colourmap is scaled according to the data. Thus in the match and mismatch profiles blue represents 0 and in the interaction profile blue corresponds to the negative value given to red and green is 0. Red has an equal interpretation in all three plots.

variance of the original profile they account for (see e.g. Press et al., 1992). As a result, the original profiles can be described as a linear sum of their SVD components.

In my case the SVD is mathematically equivalent to a principal component analysis (PCA), if one interprets the binocular interaction profiles as the covariance matrices needed for the PCA (Anzai et al., 1999c). Thus, the SVD on binocular interaction profiles could be limitedly circumscribed as a PCA on bar positions within the RFs.

The outputs of the SVD actually are left (l) and right (r) 1D-profiles for each SVD component along with its weight w. Thus the first component of a binocular interaction profile can be computed by

$$b_1 = l_1 \cdot w_1 \cdot r_1^t, \qquad \text{where } r_1^t \text{ is the transpose of } r_1.$$

The monocular profiles are fitted with 1D-Gabor functions, which are used to determine position and phase disparities of the first SVD component. While phase disparity is computed by simply subtracting left profile phase from right profile phase, I use a reference cell method for the calculation of position disparity in correspondence with Anzai et al. (1999c). Thereby the second component is taken as a reference and is assumed to have no position disparity. In order to comply to the conventions introduced by the definition of phase disparity I here subtract right position from left

position, which means that a positive disparity corresponds to far stimuli in both disparities. The resulting calculation for position disparity describes a relative position disparity and is as follows

$$\Delta x = \Delta x_1 - \Delta x_2, \qquad \Delta x_i = x_i^l - x_i^r,$$

where x_i^l and x_i^r are left and right Gabor positions of the i_{th} component. So, with Δx and Δx_i I obtain distributions for relative as well as absolute position disparities of first and second SVD components.

For completeness I here also state how I compute disparities directly from 2D-Gabor fits of RFs. Phase disparity is calculated exactly as described above. I define the horizontal position disparity of one subunit, Δx_i, to be the distance between two lines running through left and right RF centres, respectively, with a slope corresponding to the orientation of the RF. The sign of Δx_i is determined such that it corresponds to $x_i^l - x_i^r$. Δx then mathematically matches the relative position disparity perpendicular to RF orientation given in Anzai et al. (1999a).

3 Evaluation of simulation results

Many contemporary publications to the topic of binocular cells in striate cortex discuss the importance of phase versus position encoding of visual disparity. So this is also the prevalent ground for my comparisons of optimally stable cells and real neurons. Here I contrast the simulation with mainly two physiological data sets, which examine binocular properties of neurons in V1 on different levels of analysis. The data from Anzai et al. (1999c) describes binocular properties of functional, complex cell subunits by evaluating binocular interaction profiles and the data from Prince et al. (2002a,b) gives measures for disparity tuning of striatal cells obtained with RDS. Especially, it has to be noted that Prince et al. (2002a,b) examine solely responses to horizontal disparity, while Anzai et al. (1999c) investigate responses to disparity orthogonal to RF orientation. But before I go on to these data sets I will give a basic analysis of my subunit RF properties and will roughly compare the results of this analysis with reported properties of simple cells. All my comparisons suggest that a stability criterion can explain many properties of binocular complex cells, but I also find clear differences between simulation and physiology, which are partly due to the used cell model. These differences are discussed in section 4.

3.1 Subunit RFs and simple cells

Our cell model allows for infinitely many subunit RF shapes, but Ohzawa et al. (1990, 1997) point out that subunits should exhibit simple cell-like RFs, which stand in quadrature phase relationship, in order for the cell to act as a complex cell. Here I report that stable subunit RFs correspond well to the proposed binocular energy model, although they also show deviations in direct comparison with properties of simple cells.

3.1.1 Basic properties

Körding et al. (2004) already showed that optimally stable subunit RFs obtained with monocular natural images exhibit properties, which would be expected, if an energy model is assumed to underlie complex cell activities. These properties include selectivity for stimulus position, orientation and spatial frequency as well as a phase shift between subunits of about 90° (quadrature phase relationship of subunits). I can confirm these findings for the extension of the simulation to binocular natural images.

All of my subunit RFs are very well described by a 2D-Gabor function (R^2 of fit > 0.9 for all RFs, $R^2 > 0.97$ for 70% of all RFs) and thus they

are selective for local contrast, spatial frequency, orientation and position of a stimulus. The RFs are depicted in figure 7. There it can be seen that the Gabor wavelets mostly extent over the whole 20x20 pixel sized RF patches. This is in contrast to simulations with a sparseness objective in which RFs are more localised and do not extent over a whole patch (Goldbach, 2004). Nevertheless my RFs still resemble simple cell RFs, which have been reported to be well described by Gabor functions (Chalupa and Werner, 2003).

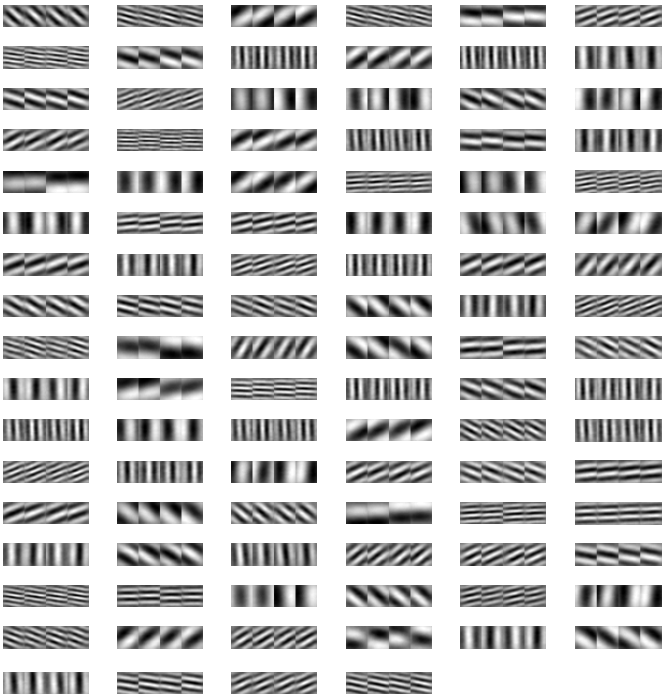

Figure 7: Stable subunit RFs obtained with binocular natural visual images. Depicted are all 100 simulated neurons, each with four RFs: subunit one left and right and subunit two left and right (in this order). Values of weights (RFs) are mapped to greyscale: bright means inputs are weighted positively, dark corresponds to negative weights. All RFs can be described by Gabor wavelets (R^2 of fit > 0.9 for all RFs).

Cells with our cell model become translation invariant, which is an important property of complex cells, if subunits stand in quadrature phase relationship. Figure 8 illustrates the distribution of phase differences between subunit RFs, which have been calculated from the corresponding parameter of the Gabor fits. It exhibits a clear peak at 90° and therefore indicates that most of the simulated cells hold this key component of the energy model. Apart from this apparent phase difference between subunits other

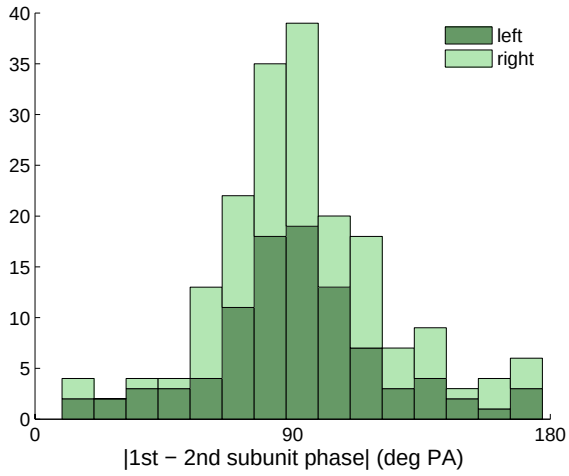

Figure 8: Distribution of absolute phase differences between first and second subunits in degree phase angle (PA). Dark green shows difference for left RFs of all cells and light green shows difference for right RFs. Most cells have subunits, which stand in quadrature phase relationship.

basic properties do not differ between subunits, exactly as it is predicted by the energy model. So I find strong correlations between orientations, spatial frequencies and RF centres of the two subunits, which is shown in figure 9.

But figure 9A additionally illustrates one clear difference between simulated RFs and those of real cells, which will also hold for the later analysis of complex cell response properties. While the population of striatal cells is in general tuned to all kinds of orientations, simulated cells mostly lack a tuning to orientations between 45° and 135° except for plain vertical orientations around 90°.

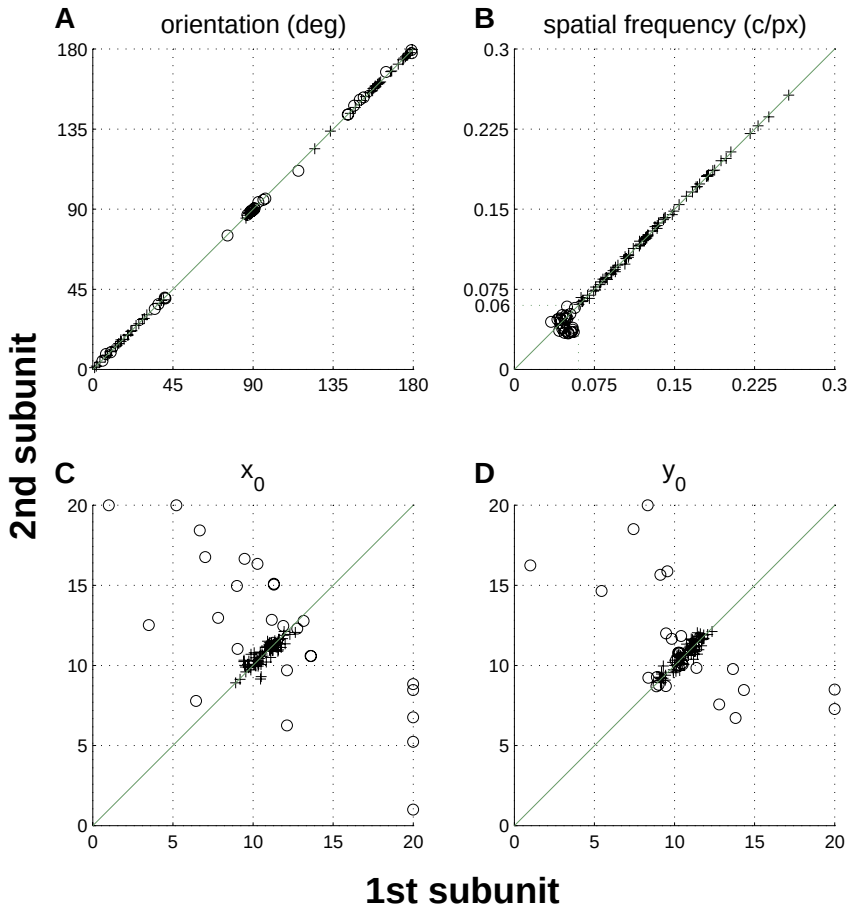

Figure 9: Comparison of basic RF properties between first and second subunits. For orientation and x- and y-value of the RF centre $[x_0, y_0]$, values of the left RFs of all cells are shown. Scatter plots for right RFs are similar and approximately equal for orientation, respectively. Spatial frequency is the mean of left and right RF spatial frequencies, but left and right RF frequencies are almost always equal, too. x_0 and y_0 are scattered more widely, because it is hard for the fitting procedure to determine these parameters correctly, which is especially true for low frequency RFs where RF centres are sometimes estimated to lie at the border of the RF patch with a value of 20 (centre of RF patch is [10.5,10.5]); RFs with low frequencies ($f < 0.06$ c/px, 28/100 RFs) depicted as $\circ$. When badly estimated centre values are taken out, I find a significant correlation between subunits for the RF centre as well ($x_0 : r = 0.86, P < 0.001$; $y_0 : r = 0.92, P < 0.001$, low frequencies excluded).

3.1.2 Binocular properties

As seen above subunit RFs have basic properties similar to simple cells. Although the binocular energy model does not assume that subunits represent actual simple cells it is an interesting further question whether binocular properties of subunits and those of simple cells are comparable. Therefore I calculated disparities within subunits from RF fits as described in section 2.3.2 and contrast them to data reported in Anzai et al. (1999a). I exclude two cells (52,74) from the simulation data, because their position disparity exceeds 10 pixel, what indicates that the estimate for position of the centre of left or right RF and so the whole fit is implausible.

The histograms plotted in figure 10 suggest that the distributions of disparities in subunits and simple cells have some critical features in common, but also reveal one main difference between simulation and physiology. Portrayed are counts of simulated and simple cells with certain phase and position disparities. Plots in figure 10A, C and E show distributions of first subunit phase disparity in degree phase angle (PA), phase disparity in stimulus space in pixel and position disparity in pixel, respectively, whereas figure 10B, D and F are their biological counterparts. I can not directly compare the ranges of disparities encoded, because the physiological data is given in degree visual angle (VA) and it is unclear how to convert this to pixel as long as biological RF sizes are unknown or differ widely in the reported data. Nevertheless statements about the form of disparity distributions can be made. So it can be seen that both disparities in stimulus space of the simulation are Gaussian-like with mean around zero as it is the case for simple cells. Furthermore Anzai et al. (1999a) report that the standard deviation of phase disparity in VA is about 1.6 times larger than the standard deviation of position disparity. A statistical analysis of my disparity distributions shows that phase has a larger variance than position (F test: F ratio $= 3.32$, df $= (97, 97)$, $P < 0.001$), too, whereby the ratio of phase versus position disparity standard deviations equals 1.82. So, this is in good correspondence with the physiological data. On the other hand, the histograms for phase disparity in PA uncover that here the distributions in simulation and biology differ. The simulation apparently produces more large phase differences than are found in simple cells. Accordingly, I count 29/98 cells with absolute phase differences larger than 120° PA in the simulation data whereas in physiology there are only circa 10/97.

Additional comparisons between subunits and simple cells underline the discovered relationships. Consequentially, like Anzai et al. (1999a), I do not find a correlation between position and phase disparities in pixel ($r =$

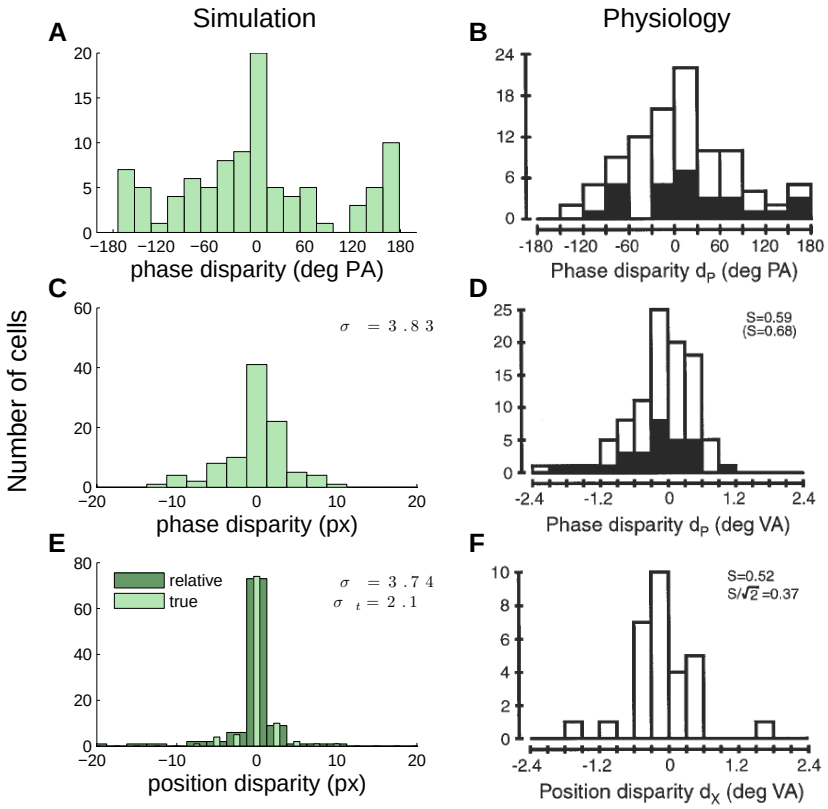

Figure 10: Phase and position disparity distributions for simulated and simple cell RFs. **A,C,E** Histograms for disparities of first subunit RFs. Phase disparity distribution in PA has a peak at 0°, but also peaks at the tails at ±180°. Phase disparity in pixel does not show this phenomenon and has a standard deviation of 3.83 px. Position disparities are centred at 0 px and have a smaller standard deviation of 3.74 px for relative and 2.1 px for true position disparities. **B,D,F** Histograms for disparity distributions of simple cells adapted from Anzai et al. (1999a). All distributions are centred around 0. More than 80% of all phase differences in PA lie within ±90°. Standard deviation of phase disparities in VA is 0.59°. Position disparities are relative position disparities determined with a reference cell method. Their standard deviation is 0.52°. The estimate for standard deviation of true position disparities is 0.37° (given that position disparities of cell and reference cell are independent).

$0.11, R^2 < 0.01\%$). Furthermore an assessment of disparities in relation with RF orientation reveals similar properties, too. In physiology it has been described that RFs with horizontal orientations ($0° \pm 20°$) encode a smaller range of phase disparities than RFs with vertical orientations ($90° \pm 20°$) (Anzai et al. (1999a): F ratio $= 2.94$, df $= (18, 25)$, $P < 0.01$). This is analogous to simulation data (F ratio $= 2.15$, df $= (30, 34)$, $P < 0.05$), although here the variance seems to increase faster from horizontal to vertical orientations, which is also expressed in the greater P-value.

The data presented here implies that most optimally stable cells building on the used cell model are a good realisation of the binocular energy model. In addition I find major similarities between subunit RFs and those of simple cells, which might be particularly interesting from the perspective that already Hubel and Wiesel (1962) proposed that complex cell responses could be constituted on simple cells as their input. Nevertheless not all properties of simple cells are reflected in the population of subunit RFs. Differences in the distributions of orientations and phase disparities in degree PA are my main corresponding findings here.

3.2 Comparison to complex cell data from Anzai et al.

Subunit RFs of real neurons can not be determined directly, but have to be estimated from the top-level output of the complex cells. Here I imitate the method from Anzai et al. (1999c) for the estimation of binocular subunit RF properties from responses of cat complex cells and compare mine with their results. I notice that this method does not recover interaction profiles of the original subunits, but rather computes functional subunits with resembling properties. Nonetheless, some of the findings from the direct RF analysis are represented in the binocular interaction profiles as well. Correspondingly I again find several similarities between optimally stable cells and real neurons and as major differences I report overrepresented tails in the distribution of phase encoding in simulation data and a dependence of disparity magnitudes on preferred orientation of simulated cells.

3.2.1 SVD components of binocular interaction profiles as subunits

Binocular interaction profiles represent the different responses of a cell to stimuli shown at different positions in the two eyes (see methods 2.3.1). If these profiles exhibit elongated regions of uniform activation along the front-

parallel axis where the position difference of left and right stimuli are equal (axis of equal disparity), then the corresponding cell is disparity selective. Except for two cells (25,76, very low spatial frequency) all of the simulated cells show clear elongated regions along the frontoparallel axis (example depicted in figure 6) similar to the description of disparity selective complex cells in Anzai et al. (1999c).

By calculating quadrature (mutually uncorrelated) components of the binocular interaction RFs with singular value decomposition, Anzai et al. (1999c) assume to obtain interaction profiles of underlying functional, complex cell subunits. They report that in more than 50% of the examined cells the SVD gives evidence for the existence of only two components, which significantly contribute to the binocular interaction profile of the cells. Thereby a key prediction of the energy model is confirmed. Nevertheless they also find neurons in which the interaction profile can be decomposed into three or four major components. In contrast, the SVD performed on simulated interaction profiles does not identify cells with more than two key components as expected for a two subunit model. Nonetheless, the population averages of the data variances to which the different components contribute (component weights) are quite similar in both data sets (see figure 11). Correspondingly the percentage of variance contributed by the first two components is circa 80% in physiology while it is about 87% in my simulation.

There are a few simulated cells for which the first SVD component already accounts for more than 70% of the data variance, what indicates that these components are no correct replication of the underlying subunits, because the subunit RFs have usually well balanced amplitudes as can be seen in figure 7. Rather the components should be seen as functional subunits, which are only linear combinations of subunits. Actually, if I compute binocular interaction profiles of the single subunits and compare those with the SVD components, it turns out that these seldom match. Instead the components represent the mean of subunit interaction profiles, or rotated, or shifted versions of them. An example of this behaviour can be seen in figure 12. Although consequently not the properties of true subunits are compared in the following, the comparison is still of interest, because functional properties of the cells are revealed.

3.2.2 Disparity encoding in component interaction profiles

The SVD on binocular interaction RFs conveniently gives monocular profiles of the component interaction profiles as output. I then use the monocular profiles to determine the component disparities by fitting 1D-Gabor func-

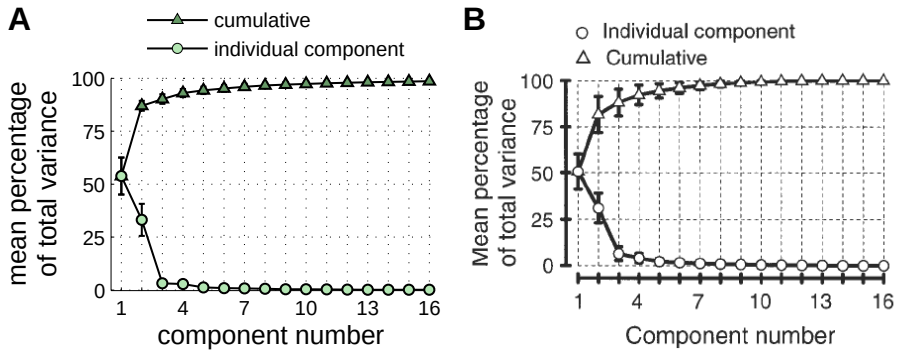

Figure 11: Contribution of components 1 to 16 to the data variance. Each data point represents the average of the population and error bars visualise ±SD. **A** Only the first two components contribute significantly to the variance in simulation data while taken together these two account for circa 87% of the variance. ($n = 100$) **B** In real neurons sometimes three or four components need to be considered to explain the data variance. Nevertheless the mean values are similar to those in A and the first two components already account for circa 80% of the variance. ($n = 48$)

tions to left and right profile and evaluating their parameters. All Gabor fits account for more than 96% of the variance of the curves, but I again exclude two cells (27,95) from the analysis, because their position disparities exceed 10 px implying bad fits. Thus, I here compare the simulation data set with size 98 to the one from Anzai et al. (1999c) of size 48.

The binocular energy model predicts that apart from a phase difference the subunits share their RF parameters such as preferred spatial frequency, orientation and disparity. Anzai et al. tested for this relation between first and second SVD components and found good correspondence with the prediction. However, while in simulation most of the parameters more or less agree in first and second components, there is no correlation between the Gabor position parameters as illustrated in figure 13. Since there is a correlation, when RFs are considered directly, this has to be a by-product of intermediate processing steps. One problem may be that inaccuracies accumulate along the path to monocular profiles and because position parameters are quite small, they are sensitive to such changes.

If it is true that estimates for Gabor position parameters of the monocular profiles are very noisy, then one would expect that the estimated distribution of position parameters is wider than the real one. Anzai et al.

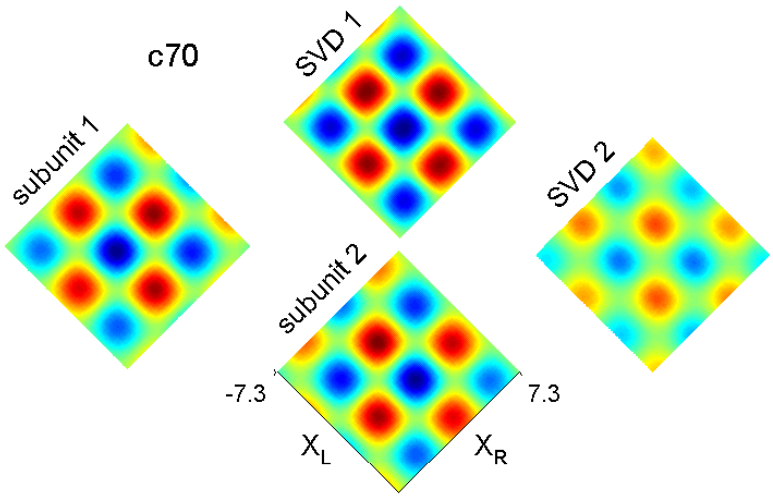

Figure 12: Binocular interaction profiles of subunits compared to those of first and second SVD components. If there is a dominant first SVD component, like here (cell 70), this resembles more the mean of the subunit interaction profiles rather than a single subunit. Otherwise the first component has been observed to represent the first subunit, the second subunit, or rotated, or shifted versions of them.

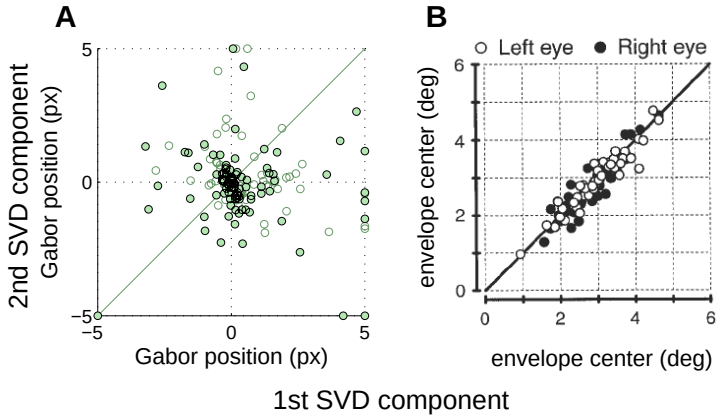

Figure 13: Comparison of Gabor position parameters (x_0) in first and second SVD components. **A** There is no correlation between x_0 in components of simulation. Black circles represent right profiles and light-coloured circles left. Data points lying outside the shown region are plotted on the nearest border. **B** Anzai et al. (1999c) find good agreement between the first two SVD components.

(1999c) report that position disparity of the first SVD component is limited to a relatively small extent compared to phase disparity. While they find that the standard deviation of phase disparities in pixel is about three times larger than the standard deviation of relative position disparities (SD ratio: 3.05), the distribution of phase disparities is only marginally wider than the distribution of relative position disparities in simulation data (SD ratio: 1.22) although I observe more large phases in PA when contrasted to physiology (histograms depicted in figure 14). Accordingly, the expected effect of noisy position estimates might have introduced this discrepancy. Thus, there would only be a marked divergence between the disparity distributions in simulated and real cells in the distribution of phase in PA suggesting that there are more cells in the simulation, which have monocular interaction profiles with different shapes.

For simple cells it has been reported that neurons with horizontal orientations predominantly encode small disparities which is thought to reflect that there are greater horizontal than vertical disparities in natural stimuli, because of the lateral displacement of the two eyes. However, this dependence of disparity on preferred stimulus orientation seems not to be existent for SVD components of complex cell interaction profiles. In figure 15 magnitudes of relative position and phase disparities are plotted against RF orientation. In both, simulation and data from Anzai et al. (1999c), cells with low orientations also encode a large range of phase disparities in PA. Nevertheless in simulated cells with horizontal orientations a slight tendency towards a smaller range of disparities can be observed, which is clearer with disparities in px (figure 15C) and there even significant on a F test between the variance of phase disparities in cells with orientations $\pm 30°$ from horizontal and the variance of phase disparities in cells with orientation $\pm 30°$ from vertical (F ratio: 1.93, df=(30,52), $P < 0.05$). Therefore, the SVD components in the simulation reflect the postulated property of natural stimuli that large vertical disparities are not as frequent as horizontal ones, whereas this is apparently not represented in components of complex cells.

Figure 15E,F illustrates the relationship between disparity of SVD components and RF spatial frequency. It has been proposed that a size-disparity correlation in which the range of disparities encoded is limited dependent on the periodicity of the RFs is advantageous for determining the correct depth of a stimulus Marr and Poggio (1979). Such a correlation is expected when phase is the dominant encoding principle, because the maximum phase disparity in pixel ($\pm 180°$ PA) directly depends on RF spatial frequency. As seen in the figure, phase disparities in pixel are widely scattered under the maximum phase limit in simulation and physiology though the scatter in

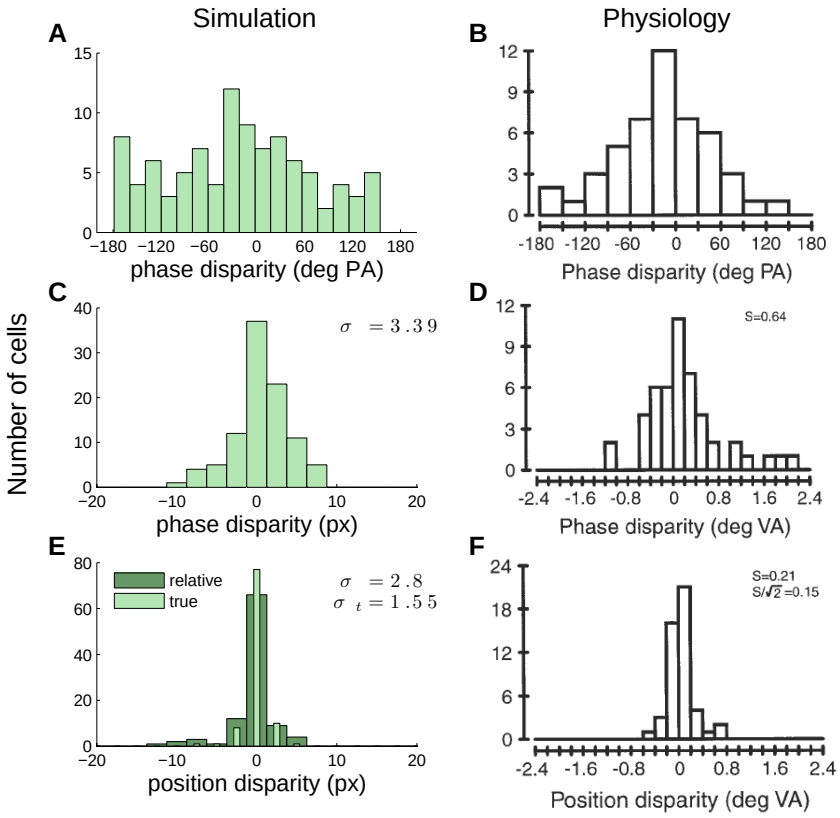

Figure 14: Phase and position disparity distributions of first SVD component for simulated and simple cells. (figure 10 for same comparison on subunit RF level) **A,B** The distribution of phase disparity in PA is much broader in the simulation. Especially phases near $\pm 180°$ are more frequent. **C,D,E,F** Position disparity is considerably denser compared to phase disparity in physiology (SD ratio: 3.05, relative position; 4.27, estimated true position). In simulated cells this is similar, but not as strong (SD ratio: 1.22, relative; 2.19 true position), which might be due to inaccuracies in the calculation.

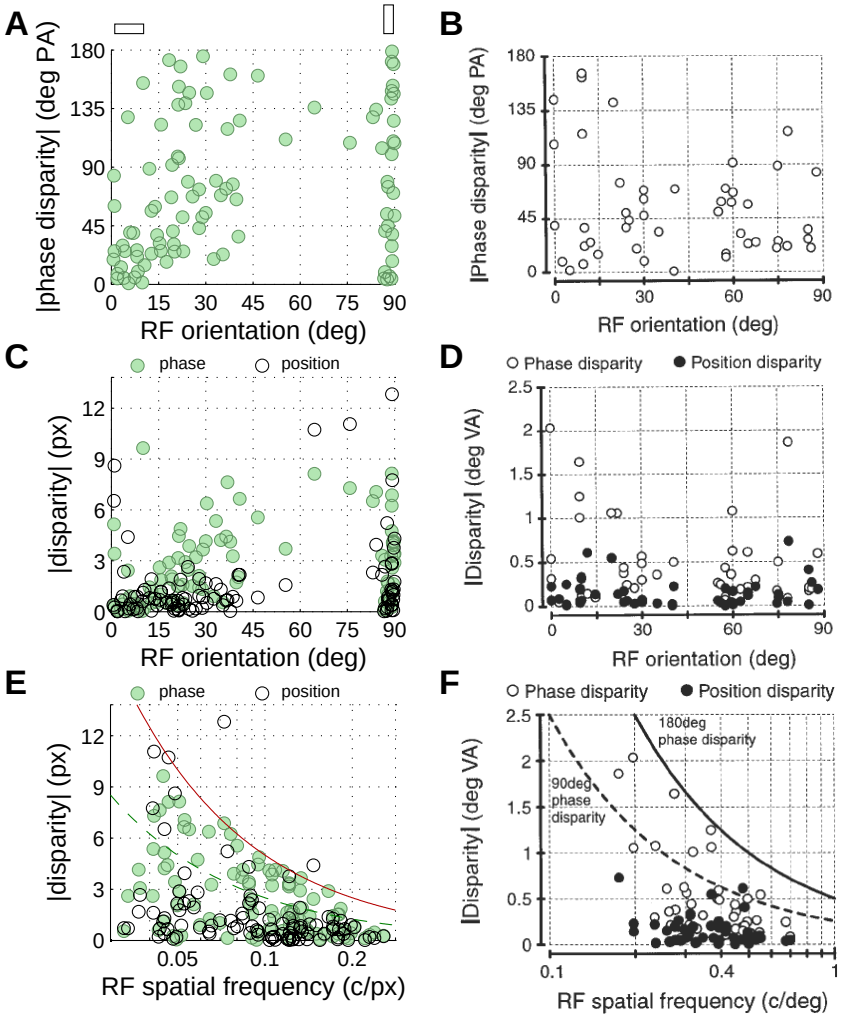

Figure 15: Relation of first SVD component disparities to RF orientation and spatial frequency. **A,B** In simulated and real cells the scatter of phase disparities in PA is large for all RF orientations. **C,D** While magnitudes of phase disparity in pixel are more or less evenly distributed in data from real neurons with horizontal or vertical orientation, the variance of phase disparities is smaller for simulated cells with horizontal orientation ($0 \pm 30°$) than for cells with vertical orientation ($90 \pm 30°$) (F ratio: 1.93, df=(30,52), $P < 0.05$) **E,F** A size-disparity correlation is found in both data sets. Solid line represents upper limit for phase disparities (180° PA), dashed line corresponds to 90° phase in PA. Biological data from Anzai et al. (1999c). Position disparities are relative position disparities.

physiology is not as evenly distributed as in simulation. Nonetheless, a clear dependence of phase disparity in pixel on RF spatial frequency can be observed in both data sets (regression analysis in data from Anzai et al.: slope= -1.99, $P < 0.01$) while there is no such dependence between phase disparity in PA and RF spatial frequency (not shown). Furthermore, Anzai et al. (1999c) find that position disparities are generally small and their range does not depend on RF spatial frequency (regression slope = -0.3, P = 0.07). In simulated cells, on the other hand, position disparities reach quite high values and it seems that they do depend on frequency as well, but it is unclear whether this reflects real properties of cells or is just a product of the above mentioned noise.

All in all the analysis of binocular interaction profiles reveals many similar properties of optimally stable and binocular complex cells. Both exhibit elongated regions along the frontoparallel axis within the interaction profiles suggesting sensitivity to disparity independent of stimulus position in the RF. Although a SVD on binocular interaction profiles does not recover the original subunit interaction profiles, the resulting components can be used to describe functional properties of the cells. Thereby the number of components found in physiology largely supports the use of a two-subunit energy model. Furthermore, the distributions of the different disparities in simulation and physiology show similar tendencies, which is also true for their relation to RF parameters spatial frequency and orientation. However, the analysis again yields that there are a lot more cells with high magnitude phase disparities (PA) in simulation than in biology. Additionally, simulated cells seem to be specialised for horizontal disparities while this is not the case for real complex cells.

3.3 Comparison to RDS data from Prince et al.

Prince et al. conducted an extensive physiological study of disparity tuning in monkey striate cortex in which they recorded responses to RDS from 787 cells. They quantified sensitivity for horizontal disparity of cells with the DDI (Prince et al., 2002b) and further examined 180 strongly disparity tuned cells by comparisons of fitted Gabor function parameters (Prince et al., 2002a). Unfortunately they did not discriminate between simple and complex cells in most of their analyses, but they classified 57 out of 226 cells for which they determined the F1:F0 ratio as simple (25%) suggesting that a large majority of reported results belong to complex cells. Furthermore they found no correlation between F1:F0 ratio and DDI, what indicates that

simple and complex cells have similar disparity sensitivities, which is supported by data from Anzai et al. (1999a,c) where distributions of phase and position parameters of simple and complex cells are similar, too.

I computed responses of my optimally stable cells to RDS with horizontal disparity as described in methods (2.2) and analysed them analogous to Prince et al.. All simulated cells exhibit statistically significant modulations of their disparity tuning curves. Although response types are largely comparable, my investigations uncover that simulated tuning curves are often wider and variances of data points differ from those of real neurons. Moreover, while in physiology phase and position encode similar ranges of disparity (phase slightly larger), disparity of simulated cells is predominantly determined by the phase component.

3.3.1 Activities to RDS and disparity sensitivity

Prince et al. (2002b) provide data in which they quantify activities of cells in V1 to RDS with horizontal disparity in order to make statements about the cells' strength of disparity tuning. They report that the variance of firing of their cells increases with mean firing rate. This relation is long known for cortical neurons (see Shadlen and Newsome, 1998, for discussion) and does not depend on the stimuli used (as long as they lead to the same mean firing). Variance of artificial activities, on the other hand, is the direct result of variability in the given stimuli, because in the calculation of activity to a certain stimulus no random process is involved. Thus the variance stabilising transform (square root of activities) employed by Prince et al. deals with a different kind of variance than I have given from my simulated cells. Then again, Prince et al. apparently define all RDS with the same disparity as one stimulus, which gives me the opportunity to compare their variances of mean firing rate to such a stimulus with my variances of activities to different RDS with equal disparity. Still this comparison should be judged with care, since the variances may result from different sources of variability.

Unfortunately Prince et al. do not give direct figures of unstabilised mean firing rates, but they report that before their square root normalisation Fisher transformed correlation values of mean firing rates and variances are 0.86 ± 0.72 (SD) on average over cells. The corresponding value for the Fisher transformed correlation of artificial activities to RDS with a certain disparity and their variances is 2.32 ± 0.19 indicating that this correlation is by far stronger than the one found in biology. Figure 16 illustrates this dependency of variability and strength of response very clearly. There, mean activities to equal disparity RDS are plotted against their vari-

ance and standard deviations. It can be seen that the average of activities has approximately a quadratic relation to variance and accordingly a linear one to standard deviation. Coherently I find that the ratio between standard deviation and mean activity is almost constant around a value of 0.53 ($\pm$0.018). Prince et al. diminish the reported correlation to a value of 0.21 ± 0.68 (Fisher transformed) by the square root transformation of their mean firing rates. The same operation on my activities reduces correlation only marginally (2.02 ± 0.27) and is therefore useless from that perspective. Another effect described by Prince et al. of this procedure was that it removed a positive skew from the response distributions for single stimuli. I made similar observations. However, since this is consistent over all cells and stimuli and the ratio of standard deviations and means is still approximately constant (0.28 ± 0.009), I do not apply the square root transform to my data.

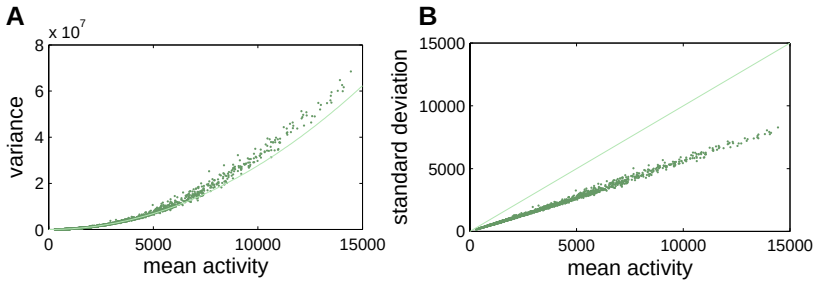

Figure 16: Variability of artificial activities. **A** Each data point represents mean activity of one cell to 1000 different RDS with a certain disparity plotted against its variance (41 different disparities $\cdot$ 100 cells = 4100 data points). The line describes the function $y = 0.53^2 x^2$ where 0.53 is the ratio of standard deviation divided by mean. **B** Same as A but with standard deviation. Mean and standard deviation have a linear relationship and their ratio is approximately constant at 0.53 ± 0.018. Plots with activities which have been scaled by a square root transform are very similar although ratio of standard deviation versus mean is reduced (0.28 ± 0.009).

I discussed these matters of variability in activities at the beginning, because the reported variances have strong influence on the DDI, which takes them into account in order to quantitatively express discriminability of maximum and minimum points on the disparity tuning profile. The idea is to rate tuning curves with lower variances better than tuning curves with the same strength of modulation but higher variance. As variability of simulated cell activities is considerably greater than the one found in physiology,

it is expected that the values of the DDI are lower. This can be seen in figure 17. Values for the DDI of simulated cells range from 0.03 to 0.16, but still offer a good description of disparity discriminability. So, for example, all very low frequency tuning curves have DDI<0.06 independent of their mean activity while tuning curves with higher frequency, that means with more considerable tuning, also get higher DDI values. Additionally Prince et al. tested their cells for disparity selectivity with a one-way ANOVA on their square root firing rate data and report that 55% of their V1 neurons show significant modulations at the 5% level. Although the assumptions of normally with equal variance distributed data are violated by the artificial activities[10] all 100 of my cells show highly significant ($P \ll 0.01$) modulations with disparity on a one-way ANOVA. This puts forward that even though DDI values are low through high variance, all of the simulated cells significantly change their response with disparity, which is backed by the fact that all of my cells are binocular according to the ODI ($\mu = 0.49 \pm 0.03$ SD). Consequently, the DDI values form a continuum from low to high disparity sensitivity, which is similar in biology.

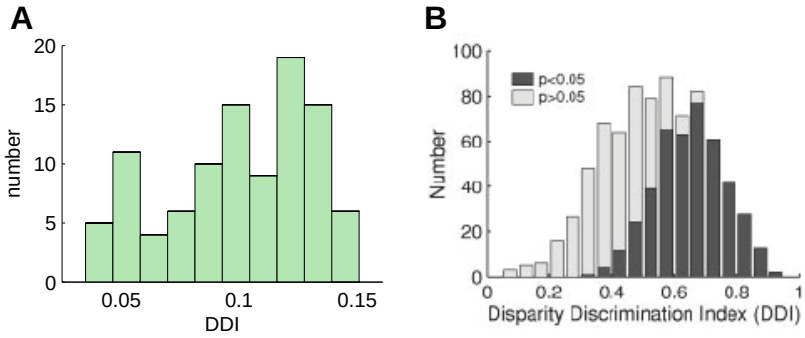

Figure 17: Histograms of DDI values of simulated and striatal cells. **A** DDI values for 100 simulated cells. Maximum is 0.151 and minimum is 0.035. A one-way ANOVA revealed that all cells exhibit significant modulations with disparity ($P \ll 0.01$). **B** DDI of 787 V1 neurons (data from Prince et al. (2002b)). Dark shaded are neurons which have significant disparity tuning on a one-way ANOVA ($P < 0.05$). Neither in A nor in B two distinct populations can be seen.

The DDI has been used to discover relationships between the strength of disparity tuning and other cell properties. First of all Prince et al. found

[10]The ANOVA test is known to be robust to modest violations of these assumptions.

that the DDI is independent of mean firing rate. In contrast, as variances of simulated activities heavily depend on their means and the DDI in turn depends on these variances it is no surprise that the DDI decreases with increasing mean activities for my data, but this dependency is more moderate than anticipated (Spearman's rank: $r_s = -0.33$, $P < 0.001$), which can be traced back to a simultaneous increase of $a_{max} - a_{min}$ from the definition of the DDI, eq. 5 ($r_s = 0.71$, $P \ll 0.001$). A scatter plot of mean activity versus DDI is depicted in figure 18A and B. Further plots in this figure show the DDI plotted against RF spatial frequency and RF orientation. Whereas frequency is not significantly correlated with the DDI ($r_s = -0.09$, NS) similar to physiology, I find a clear correlation between DDI and RF orientation ($r_s = 0.69$, $P \ll 0.001$) contrary to physiology. This relation is to some extent predicted by the binocular energy model, which states that the tuning curves of cells with preferred horizontal orientations to stimuli with horizontal disparity will have a low frequency and thus low DDI values will be assigned to them (see above). Correspondingly, the biological finding that orientation is not correlated with the DDI could be explained in two ways. Either Prince' DDI does not rate all low frequency tuning curves low, or the prediction of the energy model fails. As one can see in figure 22 this prediction is fulfilled to some degree by the measured V1 neurons, hence the former must be true, what indicates that ratings of my DDI and Prince' deviate, if certain tuning types are judged.

3.3.2 Tuning types described by position and phase encoding

Disparity tuning curves are often classified into six different categories according to their shape (Poggio et al., 1988). Here I examine tuning curves of simulated neurons for whether they exhibit the reported shapes. Thereto, I have fitted the tuning curves with Gabor functions and compare their parameters with those published by Prince et al. (2002a). Furthermore I contrast estimates for position and phase encoding of disparity from this study with disparity encodings obtained from my cells.

All of my tuning curves are well described by 1D-Gabor functions and have a R^2 value > 0.93, but I exclude four cells, because confidence intervals of their fits are complex, which indicates that these fits can not be trusted due to especially large confidence intervals[11]. So, a population of 96 cells is left for comparison with Prince et al.'s population of 180 real neurons.

[11] this criterion is more or less arbitrary, since there are other cells with large, but non-complex confidence intervals, but I did not want to exclude too many cells with this criterion since Prince et al. (2002a) have not done this either

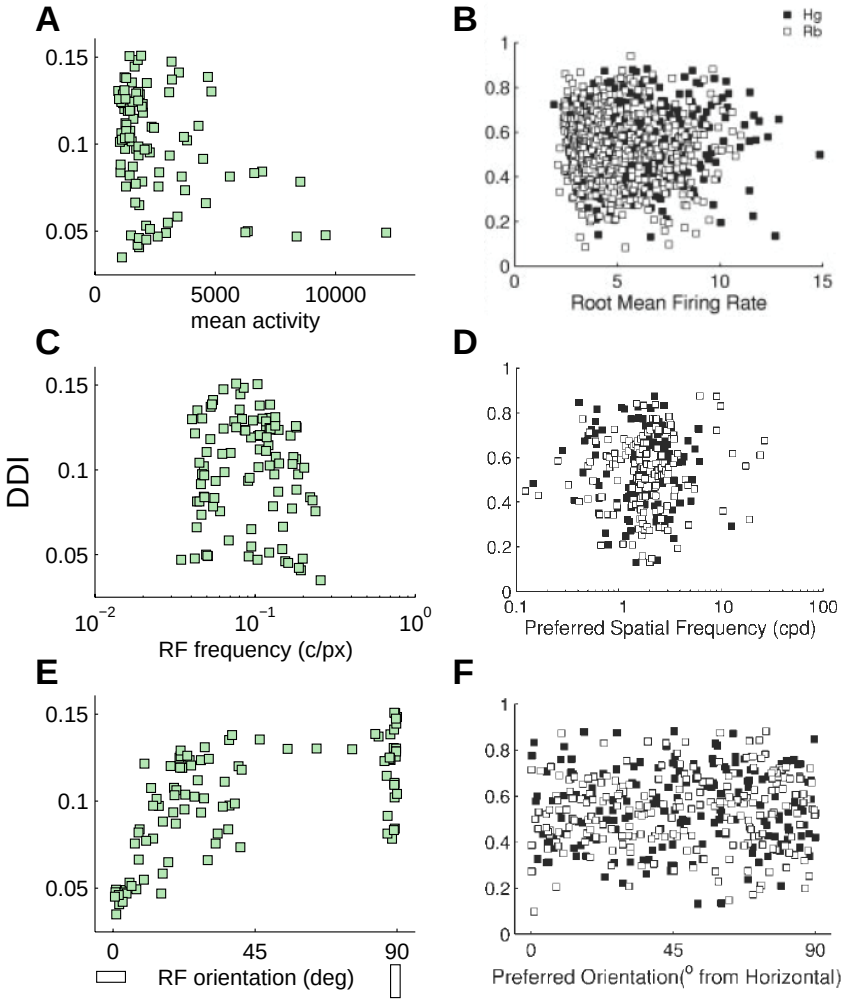

Figure 18: Relationship between DDI and other cell properties in simulation (left) and physiology (right, data from Prince et al. (2002b)). Correlation has been tested with Spearman's rank in all cases. **A,B** DDI versus mean activity. A slight correlation can be found in simulation data ($r_s = -0.33$, $P < 0.001$), but not in physiology. This correlation might be an overestimation, since all points over 5000 activity with DDI$<$ 0.05 represent low frequency tuning curves, which got consistently low DDI values. **C,D** DDI is neither in simulation ($r_s = -0.09$, NS) nor in real neurons ($r_s = 0.07$, NS) significantly correlated with frequency of the preferred grating stimulus. **E,F** Although preferred orientation of a cell and DDI show no significant correlation in biology ($r_s = 0.06$, NS), they do in simulation data ($r_s = 0.69$, $P \ll 0.001$) as predicted by the energy model.

The characteristics of the six tuning curve types are as follows: tuned zero (T0) cells have a tuning curve with a narrow peak at zero disparity, tuned excitatory (TE, near TN and far TF) cells resemble T0 cells, but their peak is shifted to near or far disparities, tuned inhibitory (TI) cells exhibit a trough around zero disparity and near (NE) and far (FA) cells show broad tuning for near and far disparities, respectively while being inhibited at opposite disparities. Therefore phase and position parameters of the fitted Gabor should suffice in order to classify the tuning curves according to this classification scheme. Then tuning curves with phase near zero correspond to T0 or TE types dependent on position while curves with phase around $\pm\pi$ are said to be TI and asymmetrical curves with phase near $\pm\pi/2$ can be classified as NE and FA, respectively. A scatter plot of these two parameters can be seen in figure 19 along with examples of tuning curves of all types from simulation and physiology. Prince et al. (2002a) found all tuning curve types in their data, but there were no clusters of certain types, rather they report a continuum of tuning curve forms from one type to another. This is comparable when disparity response profiles of simulated cells are considered. However it is hard to find good examples for TE tuning curves. Although there are cells, which are sharply tuned (similar to cell 22, T0) to near and far disparities their tuning curves are more asymmetric than symmetrical around their peak. Furthermore the peak disparity of TE cells is always near zero in my case[12]. This is comprehensible from the fact that disparity of TE cells is encoded by position disparity alone, because tuning curves of TE cells are symmetrical (phase = 0). As I have shown in section 3.1.2 that RFs of simulated cells exhibit only small position disparities, it follows that TE cells can only be tuned to disparities near zero.

Such a conclusion is valid, because the finding from RFs directly transfers to parameters of the tuning curves as can be seen in figure 20. There position is plotted against phase, which has been transformed into pixel space. It is clearly recognisable in this figure, that the distribution of phase shifts in pixel is much broader than that of position shifts. While position parameters have a mean of 0.22 px with standard deviation 1.25 px these values for phase are much larger: 0.89 ± 6.1 px. Compared to the RF analysis in section 3.1.2 this discrepancy has even increased with RDS since there the ratio of the standard deviations has been 1.82 while here it is 4.89, which is most probably the result of the use of horizontal disparities, because in

[12] Although points with phase near zero and larger position parameter can be seen in the scatter plot of figure 19A, I would not classify them as TE, because these are very broad Gaussian-like tuning curves

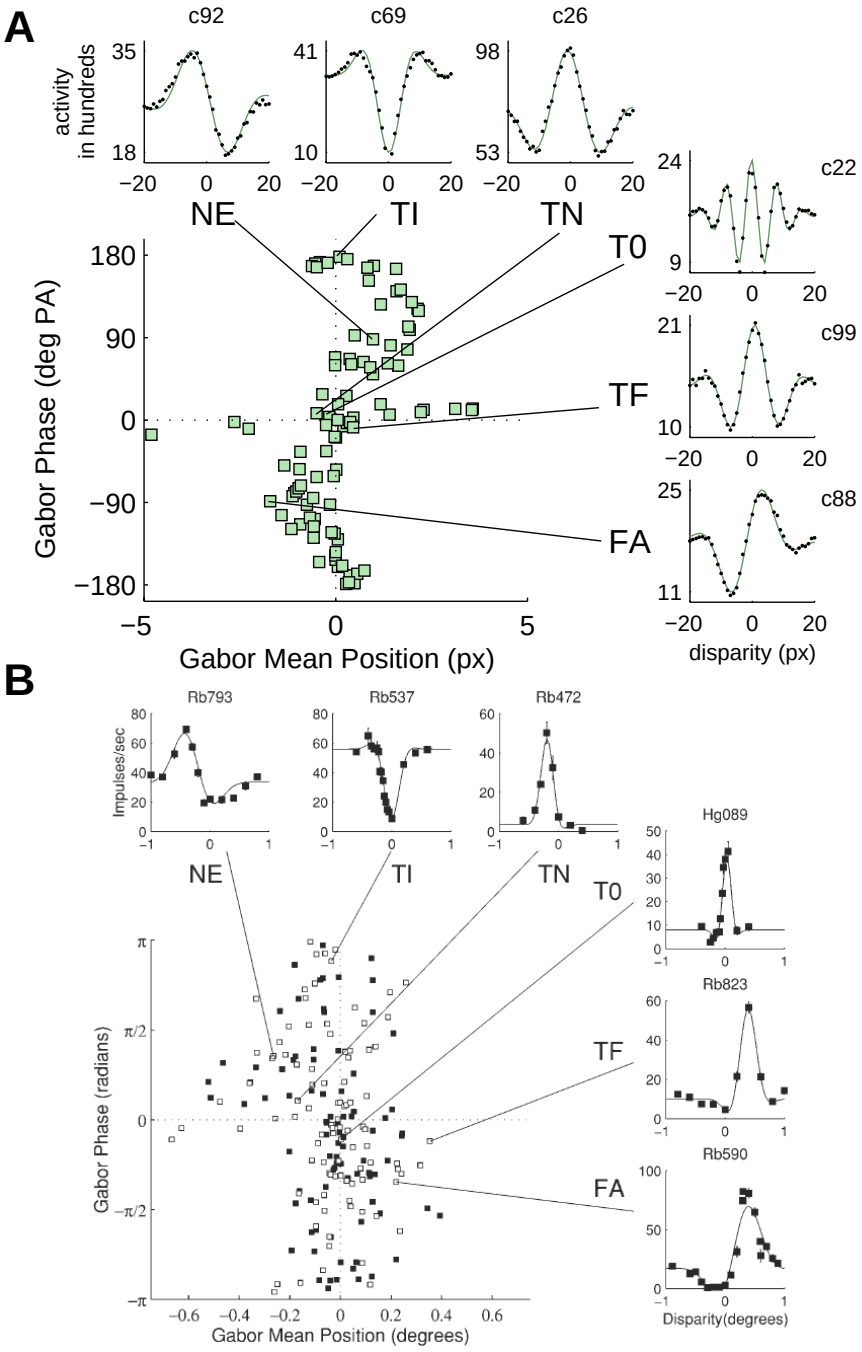

Figure 19: Tuning types in simulation (**A**) and physiology (**B**, adapted from Prince et al. (2002a)). See text for description.

comparison to disparities orthogonal to RF orientation the tuning curves show an increase in spatial scale and as position disparities seem not to have great influence in the encoding of disparity in my cells they even loose importance. However, the most important point made in figure 20 is that here an obvious difference between simulation and physiology is exposed. Although Prince et al. report that in their data the range of disparities encoded with phase is slightly larger than that encoded with position (SD ratio: 1.25), this finding is by far not as strong as it is in my data.

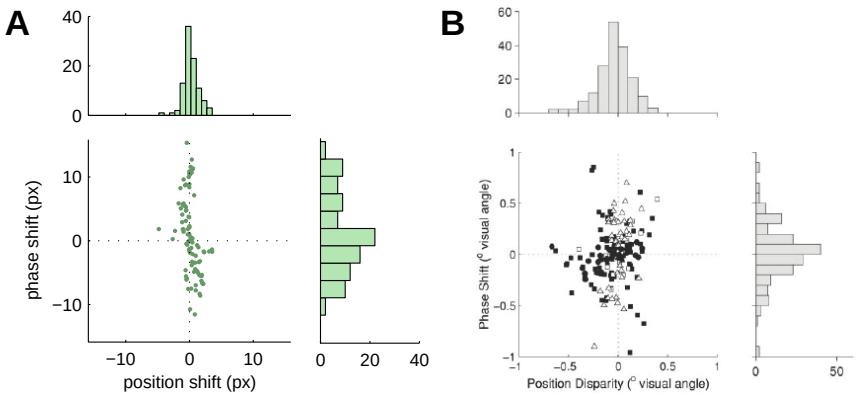

Figure 20: Comparison of phase and position parameters in simulation and physiology. Phase has been transformed into position space by division through $-1\cdot$ spatial frequency of the tuning curve. **A** Gabor mean position encodes a much smaller range of disparities than Gabor phase in simulated cells (SD ratio: 4.89). Position and phase are negatively correlated ($r_s = -0.52$, $P \ll 0.01$), that means contributions of both cancel each other, but as phase is generally considerably greater than position, a large range of disparities can still be encoded. **B** Position and phase parameters encode similar ranges of disparity in data from Prince et al. (2002a) (SD ratio: 1.25). Only a slight positive correlation is observed ($r_s = 0.24$, $P < 0.001$).

This discrepancy between the ratios of standard deviations in simulation and physiology is fostered through a further difference, which is depicted in figure 21. In the transformation of phase in PA to pixel a large phase in PA will produce a large value in pixel. However, for a large phase in PA there might be no straightforward interpretation of a large phase in pixel in terms of position shifts and preferred disparity. Obviously, this problem mainly concerns TI tuning curves, which have the largest phases (near $\pm\pi$), but are only inverted versions of T0 curves and their shift in phase can

not be compared to a shift in position with equivalent magnitude. Thus, more large phases will lead to a disproportionate representation of phase in the ratio of standard deviations when preferred disparities are concerned. This is the case when simulated cells and real neurons are compared, since there are more tuning curves with large phase in the simulation data than reported in biology (see figure 21). Correspondingly, this is expressed in the percentage of cells classified as TI by a criterion proposed in Prince et al. (2002a)[13] where only 16% of all cells were labelled TI. In contrast, I classify 26% of the simulated cells as TI according to the same criterion. Nevertheless, I here interpret the ratio of standard deviations as a measure to estimate different contributions of phase and position to the characteristics of the tuning curves and it still shows that the phase parameter has a much greater influence on tuning curves than position in my data.

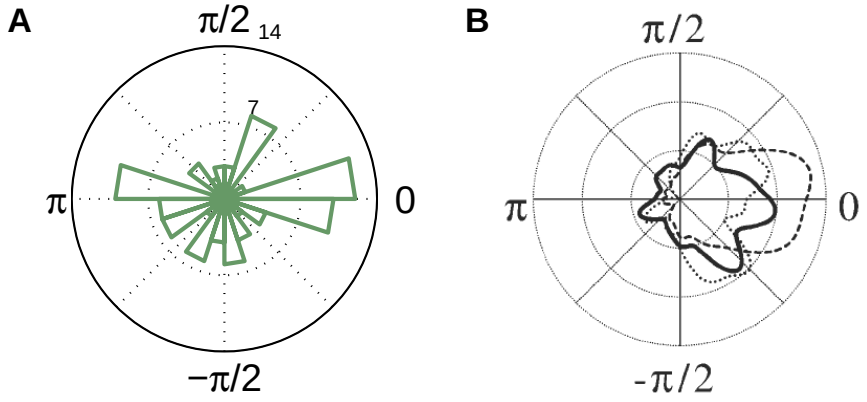

Figure 21: Distribution of fitted phases as polar plots. **A** There is a substantial number of simulated tuning curves with phases near $\pm\pi$. (distance from origin represents number of tuning curves with that phase) **B** In physiology, on the other hand, large values for phase are rare and phases near zero dominate. Data replotted from Prince et al. (2002a) (solid line) and includes data from Nieder and Wagner (2000) (dotted line) and Anzai et al. (1999a,c) and DeAngelis et al. (1991) (dashed line)

A last thing that I note from figure 19 regards the T0 cell there (c22). The tuning curve of this cell exhibits several peaks, which is seldom reported for real neurons. This phenomenon can be quantified as the ratio of the wavelength of the fitted tuning curve ($\lambda = 1/f$) to the standard deviation

[13] TI: $|phase| > \frac{3}{4}\pi$

of the fitted Gabor (W) and thus gives an estimate for the number of cycles in one standard deviation. Importantly, for this analysis I prohibited values of W greater than 10.25 px, that means I set $W = 10.25$ px for all tuning curves with $W > 10.25$ px. The rationale behind this is the following: Gabor functions with W much greater than 10.25 px allow modulation of the disparity tuning curve behind ± 20 px, what is implausible, because all disparities smaller than -20 px and greater than 20 px represent uncorrelated stimuli in left and right RFs. Therefore the tuning profiles should be flat in these regions. So, if $W > 10.25$ px, this estimate has to be wrong and $W = 10.25$ px is the next plausible estimate. This non-linear transformation of the standard deviation parameter does not change the subsequent results.

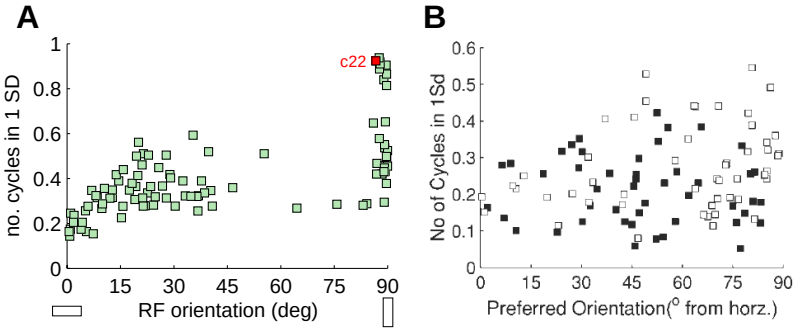

Figure 22: Number of cycles per standard deviation of the fitted Gabor (W/λ) as a function of RF orientation. **A** The minimum of W/λ in simulated tuning curves is 0.14 and the maximum is 0.94. In red c22 from figure 19 is shown. **B** In data from Prince et al. (2002b) tuning curves cover a smaller range of W/λ and most of them stay below 0.5. Yet, both data sets exhibit a correlation predicted by the energy model of W/λ with RF orientation as determined with a F test between variations of cells in the first halfs of the plots (0-45°) and second halfs of the plots (45-90°) (simulation: $P < 0.001$, physiology: $P < 0.005$).

Figure 22 depicts this ratio W/λ in relation to preferred orientation of the cells. There it can be seen that W/λ ranges from approximately 0.15 to 0.9 in simulation while the same quantity ranges only from about 0.05 to 0.5 in real tuning curves. Nonetheless I find in both data sets a dependence of the number of cycles per SD on orientation as expected when the energy model is assumed (see end of last section 3.3.1). Moreover, this relationship is significant when a F test is considered, which compares the variances of W/λ of cells with orientations smaller than 45° from horizontal with this of

cells with orientations smaller than 45° from vertical in simulation (F ratio: 4.66, df=(33,61), $P < 0.001$) and physiology (F ratio and df not given, $P < 0.005$).

In summary, investigating disparity tuning of simulated cells with random dot stereograms yields a different type of response variances if compared with response variances of neurons in V1. As a result, a measure which takes these variances into account, like the DDI, also differs in the two conditions. Moreover, the distribution of the DDI for optimally stable cells has to be altered with respect to physiology, because in contrast to physiology all optimally stable cells show significant disparity tuning. Although general types of this tuning are similar to findings in biology, the analysis of simulated disparity tuning reveals further discrepancies. So, I frequently observe tuning curves with several peaks indicating that even cells with high preferred spatial frequency have wide RFs, which is uncommon in real neurons. In correspondence with findings in previous sections phase is the dominant disparity encoding principle in my data, but this dominance exceeds by far the one reported in Prince et al. (2002a). Furthermore the tendency towards large phase disparities in PA is carried over from the RF level, but is still not found in data from real cells. I conclude that optimally stable cells exhibit response properties to RDS, which are roughly similar to those of binocular neurons, although clear deviations between both can be seen on a closer look.

4 Discussion

In this work I have compared properties of binocular complex cells to properties of simulated cells, which exhibit optimally stable activity to binocular natural stimuli in order to examine whether a stability criterion can be used to explain binocular properties of striatal cells.

From infinitely many possibilities the optimisation for stability has selected Gabor wavelets as subunit RF shapes. Consistent with physiological data these subunit RFs have all their properties in common apart from phase, which stands in quadrature relationship between the two subunits of the cell model. Thus, given a slightly modified version of the binocular energy model as skeleton, most optimally stable representations of natural stimuli implement the Gabor energy model. Consequentially, stable cells suffer from the same shortcomings as the Gabor energy model, which are discussed in the following subsection (4.1). On the other hand, the energy model has been successfully used to model certain properties of complex cells. Correspondingly I find several similarities between simulated and striatal cells such as their sensitivity to stimulus orientation and spatial frequency. Furthermore I report that stable cells exhibit elongated regions along the frontoparallel axis in binocular interaction profiles demonstrating that these cells respond to stimuli with a certain disparity independent of their position in the RF, which is a crucial feature of binocular complex cells (Anzai et al., 1999c). It has also been found in physiology that two subunits, as used here, are sufficient to represent the responses of most complex cells (Anzai et al., 1999c). Additionally the Gabor energy model successfully predicts that tuning curves to random-dot stereograms with different disparity can be described by a 1D-Gabor function. So, this is the case for optimally stable, but also for striatal cells. Prince et al. (2002a) report that tuning curve shapes form a continuum between particular prototypical shapes, what is comparable with the tuning curves of simulated cells although not all prototypes are as frequent as in real cells.

Quantitative comparisons of binocular properties also suggest many similarities between optimally stable and real cells. Accordingly, all my analyses yield evidence that the distributions of preferred disparity of stable and complex cells have core features in common. Especially, there is a similar dominance of phase encoding in simulation as reported in physiology, which is expressed in a greater range of disparities encoded by a phase mechanism than by a position difference between subunit RFs. Furthermore, in simple cells, which are thought to constitute functional subunits of complex cells, phase and position disparities have been found to be uncorrelated (Anzai

et al., 1999a) as observed in subunit RFs from the simulation. In general subunit RF properties and those of simple cells match surprisingly well. For example, in both, simulation and physiology, it has been noticed that RFs with horizontal orientation encode a smaller range of disparities than those with vertical orientation (Anzai et al., 1999a). A further example, which also extends to the complex cell response level, is a size-disparity correlation where the range of disparities represented depends on the periodicity of the tuning curve. This correlation is due to the dominance of phase encoding, because disparities encoded by this scheme are limited by spatial frequency of the RFs. Correspondingly a size-disparity correlation is evident in simulated and striatal cells (Anzai et al., 1999a,c; Prince et al., 2002a).

Although the similarities between properties of optimally stable representations of natural stimuli and those of complex cells are striking I also observe several clear discrepancies. Many of them are attributable to the employed cell model, but others may be due to the stability criterion, or the properties of natural stimuli, or may simply reflect inaccuracies in the applied methods. The former are discussed below and the latter in the final subsection (4.2).

4.1 Differences attributable to the binocular energy model

As mentioned above, most optimally stable cells implement a Gabor energy model. Therefore every discrepancy appearing between this model and real neurons also holds between stable cells and real neurons. A large part of the literature concerning binocular striatal cells evaluates properties of the energy model and I here discuss some shortcomings brought forward so far.

The energy model consists of two subunits, which stand in quadrature phase relationship, but Anzai et al. (1999c) found complex cells requiring more than two functional subunits to explain their binocular responses. Anzai et al. propose that the extra subunits may be needed to extent the RF of a complex cell in one dimension. But there are more reasons why more than two subunits may be needed. A substantial one has been published by Cumming (2002). He reports that independent of RF orientation a V1 neuron modulates its firing rate over a wider range of horizontal disparity than vertical disparity. In contrast, the energy model predicts that this depends on orientation, which can be observed in simulated cells. Hence, several subunits encoding different horizontal disparities may be needed to be combined in order to yield this preference for horizontal disparity.

Another discrepancy concerns the response to uncorrelated RDS in the two eyes. According to the energy model this has to be larger than the

response to a random-dot stimulus presented solely to the dominant eye, because monocular responses are combined linearly in the energy model. Correspondingly, the ratio of "dominant monocular response" divided by "uncorrelated response" is consistently smaller than one in simulation data, but Read and Cumming (2003) found many striatal cells where this ratio was significantly greater than one and Prince et al. (2002b) also report that the uncorrelated response is close to the mean of the monocular responses. Therefore, Read et al. (2002); Read and Cumming (2003) put forward a modification of the basic energy model in which threshold nonlinearities are included before monocular activities are combined. Such a model allows for consistent inhibitory input from one eye and thus a response to uncorrelated stimuli could be smaller than a response to a stimulus presented only to the dominant eye. The proposed modification also addresses another limitation of the basic energy model, which predicts that the response to anticorrelated RDS[14] is the exact inverted version of the response to correlated RDS (Read et al., 2002). On the contrary it has been found that even though the response is inverted in V1 neurons the modulation of the response is weaker.

Finally, RFs of neurons are known to extent not only in space, but also in time (DeAngelis and Anzai, 2003). The here presented framework has no time dimension and for that reason no time dependent properties of neurons such as tuning to direction of a moving stimulus can be modelled.

4.2 Other discrepancies and optimally stable representations of natural stimuli

Not all differences between optimally stable and striatal cells can be ascribed to the employed cell model. Although the cell model supports sensitivity to disparity, the optimisation process was free to build RFs, which do not result in disparity tuning of a cell. Nevertheless I find that all simulated cells significantly modulate their activity to stimuli with different disparities while there is a substantial subset of neurons in V1 for which this is not the case. However, it should be noticed that all cells in the simulation receive equally strong input from both eyes whereas in striatal cells the strengths of monocular inputs may be very different. Optimal stability has also been found to well describe the response of monocular complex cells (Körding et al., 2004). Thus the question of disparity tuned or not is likely to depend on the balance of monocular inputs.

Disparity tuning of simulated cells has been tested with RDS. The result-

[14] black is white, white is black in the different eyes

ing activities of stable cells display many different characteristics compared to the activities of striatal neurons. Especially variances of the responses to different RDS with equal disparity are much larger in simulation and increase quadratically with the mean while the ratio of variance to mean has been observed to be constant in physiology. It has been proposed that it is beneficial for neurons in a network to maintain such a constant ratio of variance and mean (Shadlen and Newsome, 1998) and that they may use certain mechanisms to achieve this, which are apparently missing in the model. Because variances in simulation and physiology are so different, the DDI taking them into account also exhibits some divergences between both, which are not further discussed here.

There is a marked absence of RFs with oblique orientations in the simulation data. This clearly reflects properties of natural stimuli, in which oblique orientations are not as frequent as horizontal or vertical ones, too (Einhäuser, 2004). The phenomenon is called the oblique effect and it has also been observed in physiological and psycho-physical experiments (Einhäuser, 2004), but as is demonstrated in figure 18E,F the effect is extremely strong in stable cells whereas it is not evident in a large data set of monkey striatal cells from Prince et al. (2002b). Why the effect is completely absent in this data set while it can be seen in others is not clear to me.

Another discrepancy related to RF orientation arises from Anzai et al. (1999c)'s finding that in components of binocular interaction profiles of horizontal, complex cells a large range of disparity[15] is encoded. Although I observe a similar trend in the simulation, it still exhibits a preference to encode a smaller range of vertical disparity as it is expected when monocular inputs are laterally displaced. Interestingly, the deficiency in vertical disparity is clearer if subunit RFs are analysed directly. Therefore, it could be that the method applied in this case reduces the dependence of encoded disparity range on RF orientation in simulation and physiology. Furthermore a proper evaluation of Anzai et al.'s finding is difficult, because the data set is rather sparse in the considered regions.

Above I stated as a similarity that there is a size-disparity correlation between RF spatial frequency and range of encoded disparities. On the other hand, I do not find a "size-size" correlation between RF spatial frequency and the width of the RFs, or formulated differently, RF sizes do not change with RF spatial frequency and rather extend over their whole possible space, which equals the size of their inputs (20x20 px). As a result the number of cycles within disparity tuning curves of stable cells is increased in comparison

[15] orthogonal to orientation, thus vertical

with striatal cells (Prince et al., 2002b). From a stability perspective it makes sense to expand RFs over the whole patch, since then it is possible that the response to a small stimulus remains equal, when the stimulus changes its position to another part of the input or RF, respectively, while a position shift to an empty part of the RF will extinguish the activity of a cell in any case. Thus, whether complex cells are adapted to their natural stimuli to be most stable also depends on whether cells with high spatial frequency RFs only receive input from a limited part of the visual field, or more generally, whether RFs extend over the same part of the visual field as their combined input. Unfortunately, this is extremely hard to investigate.

The most critical observation that I make through all my analyses of preferred disparity in stable cells is a considerably greater number of large phase disparities (near $\pm\pi$) than found in physiology (Anzai et al., 1999a,c; Prince et al., 2002a) expressed in more different RF shapes between the two eyes and more TI tuning curves. Goldbach (2004) also found this tendency in a similar simulation, which implemented a sparseness instead of a stability criterion. As a consequence the effect might be due to the properties of natural visual stimuli, because it is apparently independent of the applied objective. In return, it needs to be investigated whether opposite contrast in the monocular stimuli is similarly frequent as large phase disparity in optimised cells. However such a finding would not help explaining the minority of phase disparities near $\pm\pi$ in striatal neurons, which would even be counterintuitive, if an adaptation of neurons to their natural stimuli is assumed. The issue remains unresolved, but it also contributes to a further deviation of simulation and physiology. So I notice that phase encodes a larger range of disparities compared to position in stable cells, which has been reported from striatal cells, too, except that in V1 the ratio of phase divided by position differs when contrasted to simulation data. This difference is highly dependent on the employed method, but it tends to be larger in optimally stable cells, which could be explained by the higher number of phases near $\pm\pi$. Yet, this relationship still needs to be proven quantitatively.

In conclusion, I find striking similarities between stable and complex cells signifying that also binocular complex cells can be described as forming optimally stable representations of natural visual input. However, the cell model should be adapted to incorporate recent advances in the modelling of striatal neurons and further investigations need to be done to explain an open discrepancy concerning phase disparity.

Acknowledgements

I am grateful to Peter König and Selim Onat under whose supervision I learnt an awful lot about data analysis in the neurosciences. I thank Markus Goldbach for being a great fellow and Daniel Weiller for fruitful discussions. I also thank numerous friends such as Robert Freund, Stephan Weller and Egon Stemle for providing the needed antipole to do this work. Last but not least, I am grateful to my grandparents Maria and Rolf Voigt for their loving support of my studies.

References

A. Anzai, I. Ohzawa, and R. D. Freeman. Neural mechanisms for encoding binocular disparity: receptive field position versus phase. *J Neurophysiol*, 82(2):874–90, Aug 1999a.

A. Anzai, I. Ohzawa, and R. D. Freeman. Neural mechanisms for processing binocular information I. Simple cells. *J Neurophysiol*, 82(2):891–908, Aug 1999b.

A. Anzai, I. Ohzawa, and R. D. Freeman. Neural mechanisms for processing binocular information II. Complex cells. *J Neurophysiol*, 82(2):909–24, Aug 1999c.

L. M. Chalupa and J. S. Werner, editors. *The Visual Neurosciences*. MIT Press, 2003.

B. G. Cumming. An unexpected specialization for horizontal disparity in primate primary visual cortex. *Nature*, 418(6898):633–6, Aug 2002. doi: 10.1038/nature00909. URL `http://dx.doi.org/10.1038/nature00909`.

G. C. DeAngelis and A. Anzai. *A modern view of the classical receptive field: linear and nonlinear spatiotemporal processing by V1 neurons*, pages 704–719. Volume 1 of , Chalupa and Werner (2003), 2003.

G. C. DeAngelis, I. Ohzawa, and R. D. Freeman. Depth is encoded in the visual cortex by a specialized receptive field structure. *Nature*, 352(6331):156–9, Jul 1991. doi: 10.1038/352156a0. URL `http://dx.doi.org/10.1038/352156a0`.

W. Einhäuser. *Bottom-up and Top-down Processing of Natural Scenes*. PhD thesis, Institute of Neuroinformatics, ETH Zürich, 2004.

R. D. Freeman and I. Ohzawa. On the neurophysiological organization of binocular vision. *Vision Res*, 30(11):1661–76, 1990.

D. Gabor. Theory of communication. *J. Inst. Elec. Eng.*, 93:429–457, 1946.

M. Goldbach. Quantitative properties of sparse binocular representations of natural visual images. Bachelor's thesis, University of Osnabrueck, Institute for Cognitive Science, Neurobiopsychology department, October 2004.

F. Gonzalez and R. Perez. Neural mechanisms underlying stereoscopic vision. *Prog Neurobiol*, 55(3):191–224, Jun 1998.

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J Physiol London*, 160: 106–54, Jan 1962.

K. Körding, C. Kayser, W. Einhaeuser, and P. König. How are complex cell properties adapted to the statistics of natural stimuli. *Journal of Neurophysiology*, 91:206–212, 2004.

D. Marr and T. Poggio. A computational theory of human stereo vision. *Proc R Soc Lond B Biol Sci*, 204(1156):301–28, May 1979.

F. Mechler and D. L. Ringach. On the classification of simple and complex cells. *Vision Res*, 42(8):1017–33, Apr 2002.

A. Nieder and H. Wagner. Horizontal-disparity tuning of neurons in the visual forebrain of the behaving barn owl. *J Neurophysiol*, 83(5):2967–79, May 2000.

I. Ohzawa, G. C. DeAngelis, and R. D. Freeman. Stereoscopic depth discrimination in the visual cortex: neurons ideally suited as disparity detectors. *Science*, 249(4972):1037–41, Aug 1990.

I. Ohzawa, G. C. DeAngelis, and R. D. Freeman. Encoding of binocular disparity by simple cells in the cat's visual cortex. *J Neurophysiol*, 75(5): 1779–805, May 1996.

I. Ohzawa, G. C. DeAngelis, and R. D. Freeman. Encoding of binocular disparity by complex cells in the cat's visual cortex. *J Neurophysiol*, 77 (6):2879–909, Jun 1997.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583): 607–9, Jun 1996.

S. Onat, C. Kayser, and P. König. On the time course of disparity in natural visual stimuli. to be published.

G. F. Poggio, F. Gonzalez, and F. Krause. Stereoscopic mechanisms in monkey visual cortex: binocular correlation and disparity selectivity. *J Neurosci*, 8(12):4531–50, Dec 1988.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing*. Cambridge University Press, second edition, 1992. URL `www.nr.com`.

S. J. D. Prince, B. G. Cumming, and A. J. Parker. Range and mechanism of encoding of horizontal disparity in macaque V1. *J Neurophysiol*, 87(1): 209–21, Jan 2002a.

S. J. D. Prince, A. D. Pointon, B. G. Cumming, and A. J. Parker. Quantitative analysis of the responses of V1 neurons to horizontal disparity in dynamic random-dot stereograms. *J Neurophysiol*, 87(1):191–208, Jan 2002b.

J. C. A. Read and B. G. Cumming. Testing quantitative models of binocular disparity selectivity in primary visual cortex. *J Neurophysiol*, 90(5):2795–817, Nov 2003. doi: 10.1152/jn.01110.2002. URL `http://dx.doi.org/10.1152/jn.01110.2002`.

J. C. A. Read, A. J. Parker, and B. G. Cumming. A simple model accounts for the response of disparity-tuned V1 neurons to anticorrelated images. *Vis Neurosci*, 19(6):735–53, 2002.

M. N. Shadlen and W. T. Newsome. The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. *J Neurosci*, 18(10):3870–96, May 1998.

A Source code

Principal Matlab functions and scripts, which I have written during the work on this thesis, are listed below. I have organised them into the categories Gabor fitting, RF analysis, interaction profiles, RDS and displaying.

Gabor fitting

fitGabor1f.m

```
% [pars,residual,R2,Vres,conf,exitflag] = ...
%                fitGabor1f(dataX, dataY, start, showFit, varY)
%
% tries to fit a one dimensional Gabor to the datapoints
% given as X and Y coordinate vectors in dataX and dataY with frequency of
% the Gabor set static to the value given in start values (start)
% start must be a 6-element vector in the form: [A,x0,W,f,phi,Ao]
% if showFit is on (1), data is plotted together with the fit
% if varY is given the values are used as weights (1./varY) for the
% calculation of confidence intervals and R2
%
% output:
%       pars     - the fitted parameters of the gabor
%       residual - value of the residual fitY-dataY
%       R2       - R-square (explained variation)
%       Vres     - residual variance
%       conf     - confidence intervals for parameters - quadratic approximation
%       exitflag - exitflag of optimisation (<1, if no convergence)

% version 1.0 {Sebastian Bitzer}
function [parsFull,residual,R2adj,Vres,conf,exitflag] = ...
                            fitGabor1f(dataX, dataY, start, showFit, varY)

if (size(dataX,1) > 1)
    dataX = dataX';
end
if (size(dataY,1) > 1)
    dataY = dataY';
end
if (nargin>4)
    if( size(varY,1) > 1)
```

```
        varY = varY';
    end
else
    varY = ones(1,length(dataY));
end


%%  Gabor function to fit
%   defined here in order to have easy access to calculated frequency

    function [y,J] = gabor(params, x)

        A     = params(1);
        d0    = params(2);
        sig   = params(3);
        f     = start(4);
        phi   = params(4);
        RMean = params(5);

        y = RMean + A*exp(-(x-d0).^2/(2*sig^2)) .* cos(2*pi*f*(x-d0) + phi);
        % y = max(0,y); % this should be on, but makes derivatives difficult

        if (size(x,2) > 1)
            x = x';
        end

        % first partial derivatives (Jacobian)
        J(:,1) = exp(-1/2*(x-d0).^2/sig^2).*cos(2*pi*f*(x-d0)+phi);
        J(:,2) = A*(x-d0)/sig^2.*exp(-1/2*(x-d0).^2/sig^2) .* ...
                 cos(2*pi*f*(x-d0)+phi)+2*A*exp(-1/2*(x-d0).^2/sig^2) .* ...
                 sin(2*pi*f*(x-d0)+phi)*pi*f;
        J(:,3) = A*(x-d0).^2/sig^3.*exp(-1/2*(x-d0).^2/sig^2) .* ...
                 cos(2*pi*f*(x-d0)+phi);
        J(:,4) = -A*exp(-1/2*(x-d0).^2/sig^2).*sin(2*pi*f*(x-d0)+phi);
        J(:,5) = ones(length(x),1);

    end

    % second partial derivatives of Gabor function
    function rho2G = rho2G(params,x)
```

```
A     = params(1);
d0    = params(2);
sig   = params(3);
f     = start(4);
phi   = params(4);
RMean = params(5);

rho2G = zeros(5,5,length(x));

rho2G(1,2,:) = (x-d0)/sig^2.*exp(-1/2*(x-d0).^2/sig^2) .* ...
               cos(2*pi*f*(x-d0)+phi)+2*exp(-1/2*(x-d0).^2/sig^2).* ...
               sin(2*pi*f*(x-d0)+phi)*pi*f;
rho2G(1,3,:) = (x-d0).^2/sig^3.*exp(-1/2*(x-d0).^2/sig^2) .* ...
               cos(2*pi*f*(x-d0)+phi);
rho2G(1,4,:) = -exp(-1/2*(x-d0).^2/sig^2).*sin(2*pi*f*(x-d0)+phi);
rho2G(2,1,:) = rho2G(1,2,:);
rho2G(2,2,:) = -A/sig^2*exp(-1/2*(x-d0).^2/sig^2) .* ...
               cos(2*pi*f*(x-d0)+phi)+A*(x-d0).^2/sig^4 .* ...
               exp(-1/2*(x-d0).^2/sig^2).*cos(2*pi*f*(x-d0)+phi)+...
               4*A*(x-d0)/sig^2.*exp(-1/2*(x-d0).^2/sig^2) .* ...
               sin(2*pi*f*(x-d0)+phi)*pi*f - ...
               4*A*exp(-1/2*(x-d0).^2/sig^2) .* ...
               cos(2*pi*f*(x-d0)+phi)*pi^2*f^2;
rho2G(2,3,:) = -2*A*(x-d0)/sig^3.*exp(-1/2*(x-d0).^2/sig^2) .* ...
               cos(2*pi*f*(x-d0)+phi)+A*(x-d0).^3/sig^5 .* ...
               exp(-1/2*(x-d0).^2/sig^2).*cos(2*pi*f*(x-d0)+phi)+...
               2*A*(x-d0).^2/sig^3.*exp(-1/2*(x-d0).^2/sig^2) .* ...
               sin(2*pi*f*(x-d0)+phi)*pi*f;
rho2G(2,4,:) = -A*(x-d0)/sig^2.*exp(-1/2*(x-d0).^2/sig^2) .* ...
               sin(2*pi*f*(x-d0)+phi) + ...
               2*A*exp(-1/2*(x-d0).^2/sig^2) .* ...
               cos(2*pi*f*(x-d0)+phi)*pi*f;
rho2G(3,1,:) = rho2G(1,3,:);
rho2G(3,2,:) = rho2G(2,3,:);
rho2G(3,3,:) = -3*A*(x-d0).^2/sig^4.*exp(-1/2*(x-d0).^2/sig^2).*...
               cos(2*pi*f*(x-d0)+phi)+A*(x-d0).^4/sig^6 .* ...
               exp(-1/2*(x-d0).^2/sig^2).*cos(2*pi*f*(x-d0)+phi);
rho2G(3,4,:) = -A*(x-d0).^2/sig^3.*exp(-1/2*(x-d0).^2/sig^2) .* ...
               sin(2*pi*f*(x-d0)+phi);
rho2G(4,1,:) = rho2G(1,5,:);
```

```
        rho2G(4,2,:) = rho2G(2,5,:);
        rho2G(4,3,:) = rho2G(3,5,:);
        rho2G(4,4,:) = -A*exp(-1/2*(x-d0).^2/sig^2).*cos(2*pi*f*(x-d0)+phi);
    end


%%  Start values
if length(start)~=6
    error('start values are needed! (form: [A,x0,W,f,phi,Ao])');
else
    oStart = [start(1:3),start(5:6)];
end


%%  Optimisation (fitting)

lb = [0,-inf,-inf,-inf,-inf];
ub = [inf,inf,inf,inf,inf];

options = jklsqcurvefit('defaults');
options = optimset(options,'MaxFunEvals',10000,'MaxIter',10000);
options = optimset(options,'Jacobian','on');

[pars, resnorm, residual, exitflag, output, lambda, J] = ...
    jklsqcurvefit(@gabor, oStart, dataX, dataY,lb,ub,options);


%%  Analysis
%   (according to Numerical Recipies chapters 15.5 and 15.6)

% covariance matrix and confidence intervals
    J = full(J);

    % weighted with variance
    lin = repmat(1./varY,size(J,2),1) .* J' * J;
    sq  = sum( repmat(reshape((dataY-gabor(pars,dataX))./varY, ...
                      [1,1,length(dataY)]), [5,5,1]) ...
               .* rho2G(pars, dataX), 3 );

    a   = lin-sq;
```

```
    C = inv(a);                                % covariance matrix

    dChi = 4;                                  % 95.4 confidence interval
    conf = sqrt(dChi)*sqrt(diag(C))';          % confidence intervals

% residual variance
    Vres = residual*residual' / (length(dataY) - length(pars));
    % VresNorm = Vres/(max(dataY)-min(dataY));
    VresNorm = Vres/mean(varY);

% explained variation (R-square)
    sst = sum((dataY-mean(dataY)).^2 ./ varY);
    sse = sum((dataY-gabor(pars,dataX)).^2 ./ varY);
    R2adj = 1 - sse*(length(dataY)-1) / (sst*(length(dataY)-length(pars)));

    parsFull = [pars(1:3), start(4), pars(4:5)];


%%  Plot data together with fitted Gabor

if (showFit)
    h = figure;
    clf
    hold on;
    plot(dataX,dataY,'o','MarkerEdgeColor','b')
    x = min(dataX):.1:max(dataX);
    plot(x,gabor(pars, x),':k')
    a = annotation('textbox',[0.15 0.9 0.15 0.09]);
    set(a,'String', {num2str(Vres), num2str(VresNorm)});
    hold off;
end

end
```

gabor1.m

```
% [y,J] = gabor1(params, x)
% computes the values of the one dimensional gabor function given by
% params at positions x and also returns its Jacobian matrix wrt x
```

```
% version 1.0 {Sebastian Bitzer}
function [y,J] = gabor1(params, x)

A     = params(1);
d0    = params(2);
sig   = params(3);
f     = params(4);
phi   = params(5);
RMean = params(6);

y = RMean + A*exp(-(x-d0).^2/(2*sig^2)) .* cos(2*pi*f*(x-d0) + phi);
% y = max(0,y); % this should be on, but makes derivatives difficult

if (size(x,2) > 1)
x = x';
end

J(:,1) = exp(-1/2*(x-d0).^2/sig^2).*cos(2*pi*f*(x-d0)+phi);
J(:,2) = A*(x-d0)/sig^2.*exp(-1/2*(x-d0).^2/sig^2) .* ...
         cos(2*pi*f*(x-d0)+phi)+2*A*exp(-1/2*(x-d0).^2/sig^2) .* ...
         sin(2*pi*f*(x-d0)+phi)*pi*f;
J(:,3) = A*(x-d0).^2/sig^3.*exp(-1/2*(x-d0).^2/sig^2).*cos(2*pi*f*(x-d0)+phi);
J(:,4) = -2*A*exp(-1/2*(x-d0).^2/sig^2).*sin(2*pi*f*(x-d0)+phi).*(pi*(x-d0));
J(:,5) = -A*exp(-1/2*(x-d0).^2/sig^2).*sin(2*pi*f*(x-d0)+phi);
J(:,6) = ones(length(x),1);


end
```

fitGabor2.m

```
% [pars,residual,R2,Vres,conf,exitflag] = ...
%                fitGabor2(dataX, dataY, start, showFit, varY)
%
% tries to fit a two dimensional Gabor to the datapoints given as
% X and Y coordinate vectors in dataX and dataY with frequency and
% amplitude offset of the Gabor set static to the values given in
% start values (start)
% start must be a 9-element vector in the form:
```

```
%     [x0,y0,Wp,Wq,phi,theta,A,f,Ao]
% if showFit is on (1), data is plotted together with the fit
% if varY is given the values are used as weights (1./varY) for the
% calculation of confidence intervals and R2
%
% output:
%    pars     - the fitted parameters of the gabor
%    residual - value of the residual fit-RF
%    R2       - R-square statistics (variation explained by fit in %)
%               as no variances for the individual pixels of
%               the RF are given this R-square is not weighted
%               (see curve fitting toolbox online help:
%                evaluating the goodness of fit)
%    Vres     - residual variance
%    conf     - confidence intervals for parameters - linear approximation
%    exitflag - info whether optimization successful (<1, if no convergence)

% version 1.0 {Sebastian Bitzer}
function [pars_full,residual,R2adj,Vres,conf,exitflag] = ...
                    fitGabor2(dataX, dataY, start, showFit,varY)

dataX = dataX(:)';
dataY = dataY(:)';
if (nargin>4)
    varY = varY(:)';
else
    varY = ones(size(dataY));
end


%%  Gabor function to fit
%   defined here in order to have easy access to calculated frequency

    % see stand-alone gabor2.m for more info about this function
    function fxy = gabor(params, xlong)

        % x-y value transformation:
        % it is assumed that y<=20, in terms of pixel patches this means that
        % the patch has 20 rows, the top left pixel in the patch is defined
        % as x=1, y=1, consequentially if the patch has 20 columns, the
```

```
        % bottom right pixel will be x=20, y=20
        % thus xlong=1:400 will produce 400 values for a pixel patch of
        % width=20 and length=20 with the corresponding line and column
        % numbers as x and y values
        x = floor((xlong-1)./20)+1;
        y = mod((xlong-1),20)+1;

        x0     = params(1);
        y0     = params(2);
        Wp     = params(3);
        Wq     = params(4);
        phi    = params(5);
        theta  = params(6);
        gam    = params(6);
        A      = params(7);
        f      = start(8); %params(9);      % static
        A0     = start(9); %params(10);     % static

        p = (x-x0)*cos(gam) + (y-y0)*sin(gam);
        q = -(x-x0)*sin(gam) + (y-y0)*cos(gam);
        u = (x-x0)*cos(theta) + (y-y0)*sin(theta);
        fxy = A0 + A*exp(-(p.^2)/(2*Wp^2)).*exp(-(q.^2)/(2*Wq^2)) .* ...
              cos(2*pi*f*u + phi);
    end


%%  Start values
if length(start)~=9
    error('start values are needed! (form: [x0,y0,Wp,Wq,phi,theta,A,f,Ao])');
else
    oStart = start(1:7);
end


%%  Optimisation (fitting)
% lb = [];
% ub = [];
lb = [1,1,0,0,-inf,-inf,-inf];
ub = [20,20,25,25,inf,inf,inf];
```

```
options = jklsqcurvefit('defaults');
options = optimset(options,'MaxFunEvals',2000,'MaxIter',1000);
% options = optimset(options,'Jacobian','on');

[pars, resnorm, residual, exitflag, output, lambda, J] = ...
    jklsqcurvefit(@gabor, oStart, dataX, dataY,lb,ub,options);
if (exitflag == 0)                              % if not converged
    oStart(5) = -oStart(5);                     % try with inverted phase
    [pars, resnorm, residual, exitflag, output, lambda, J] = ...
        jklsqcurvefit(@gabor, oStart, dataX, dataY,lb,ub,options);
end

pars_full = [pars, start(8), start(9)];


%%  Analysis
%   (according to Numerical Recipes chapters 15.5 and 15.6)

% covariance matrix and confidence intervals
    J = full(J);

    a   = repmat(1./varY,size(J,2),1) .* J' * J;

    C = inv(a);                                 % covariance matrix

    dChi = 4;                                   % 95.4 confidence interval
    conf = sqrt(dChi)*sqrt(diag(C))';           % confidence intervals

% residual variance
    Vres = residual*residual' / (length(dataY) - length(pars));
    % VresNorm = Vres/(max(dataY)-min(dataY));
    VresNorm = Vres/mean(varY);

% explained variation (R-square)
    sst = sum((dataY-mean(dataY)).^2 ./ varY);
    sse = sum((dataY-gabor(pars,dataX)).^2 ./ varY);
    R2adj = 1 - sse*(length(dataY)-1) / (sst*(length(dataY) - length(pars)));


%%  Plot data together with fitted Gabor
```

```
if (showFit)
    figure;
    both = [RF, reshape(gabor(pars,dataX),20,20)];
    imagesc(both), colormap(gray);
end

end
```

gabor2.m

```
% fxy = gabor2(params, xy)
%
% calculates values of a Gabor wavelet (2D-Gabor function) with parameters
% given in params at particular x-y coordinates
%
% xy determines the position, size and resolution of the desired section
% of the wavelet for x and y dimensions while coordinate system is defined
% with respect to row and column numbering of RF patches, thus point (1,1)
% is the upper left pixel of a RF when displayed with imagesc
% examples: xy = 1:20,     yields a section of the wavelet corresponding to
%                          a RF in position, size and resolution
%           xy = 1:0.1:20, reduced step size increases resolution in pixel
%           xy = -20:40,   displays a greater section of the wavelet with
%                          area 1:20 in the middle of the section
%
% output fxy is a long vector and has to be reshaped to be plotted properly
% e.g.: imagesc(reshape(gabor2(params,1:20),20,20))
% number of rows is always equal to number of columns
%
% see Anzai et al. 1999c (Neural Mechanisms for ... Receptive Field
% Position ...), appendix for further info about Gabor functions

% version 1.0 {Sebastian Bitzer}
function fxy = gabor2(params, xy)

    xy = xy(:)';
    l = length(xy);
    x = reshape(repmat(xy,l,1),1,l*l);
    y = repmat(xy,1,l);
```

```
    x0     = params(1);            % center x
    y0     = params(2);            % center y
    Wp     = params(3);            % width of minor gaussian
    Wq     = params(4);            % width of majof gaussian
    phi    = params(5);            % phase of sinusoid
    theta  = params(6);            % orientation of sinusoid
    gam    = params(6);            % orientation of gaussian
    A      = params(7);            % amplitude
    f      = params(8);            % frequency of sinusoid
    Ao     = params(9);            % amplitude offset

    p = (x-x0)*cos(gam) + (y-y0)*sin(gam);
    q = -(x-x0)*sin(gam) + (y-y0)*cos(gam);
    u = (x-x0)*cos(theta) + (y-y0)*sin(theta);

    fxy = Ao + A*exp(-(p.^2)/(2*Wp^2)).*exp(-(q.^2)/(2*Wq^2)) .* ...
          cos(2*pi*f*u + phi);

end
```

RF analysis

grating.m

```
% fxy = grating(params, xlong)
%
% takes x-y values within one vector, transforms these to seperate x and y
% values (see below) and calculates values of a grating with parameters
% given in params at these x-y values
%
% x-y value transformation:
%       it is assumed that y<=20, in terms of pixel patches this means that
%       the patch has 20 rows, the top left pixel in the patch is defined
%       as x=1, y=1, consequentially if the patch has 20 columns, the
%       bottom right pixel will be x=20, y=20
%       thus xlong=1:400 will produce 400 values for a pixel patch of
%       width=20 and length=20 with the corresponding line and column
%       numbers as x and y values
%
```

```
% output fxy is a long vector and has to be reshaped to be plotted properly
% e.g.: imagesc(reshape(grating(params,1:400),20,20))

% version 1.0 {Sebastian Bitzer}
function fxy = grating(params, xlong)

    xlong = xlong(:);

    % transform 1D x to 2D x and y coordinates
    x = floor((xlong-1)./20)+1;
    y = mod((xlong-1),20)+1;

    theta  = params(1);             % orientation
    phi    = params(2);             % phase
    A      = params(3);             % amplitude
    f      = params(4);             % frequency
    A0     = params(5);             % amplitude offset
    x0     = params(6);             % center-x
    y0     = params(7);             % center-y

    u = (x-x0)*cos(theta) + (y-y0)*sin(theta);
    fxy = A0 + A*cos(2*pi*f*u + phi);

end
```

findOri.m

```
% oriBest = findOri(f, A, RFlong)
% determines orientation oriBest of RF given in one vector,
% [400,1]=size(RFlong), by correlating the RF with gratings with
% frequency f and amplitude A

% version 1.0 {Sebastian Bitzer}
function oriBest = findOri(f, A, RFlong)

    corrBest = 0;
    for o = -pi/2:0.05:pi/2             % for all orientations in 0.1 steps
        pars = [o,0,A,f,0,10,10];
        t = grating(pars, [1:400]');    % produce grating with that ori
        corr = abs(t'*RFlong);          % correlate RF and grating,
```

```
                                        % if phase of grating is wrong,
                                        % scalar product is negative
        if (corr > corrBest)            % if correlation is better
            corrBest = corr;            % store it
            oriBest  = o;               % and remember orientation
        end
    end

end
```

fitGaborToRF.m

```
% [fits, residuals, R2, Vres, conf, exitflag] = ...
%                   fitGaborToRF(areaFile,cells)
%
% fits Gabor wavelets (2D-Gabor functions) to subunit RFs of simulated
% cells
% areaFile is the file name of a file containing the variable 'areas'
% from the simulation which will be loaded
% cells is a vector of cell numbers, cells with these numbers (IDs) have to
% exist in 'areas'
% heuristic start values for fitting are chosen
%
% output:
%    fits     - the fitted parameters of the gabor
%               [ncells,9] = size(fits)
%    residual - value of the residual fit-RF
%    R2       - R-square statistics (variation explained by fit in %)
%               as no variances for the individual pixels of
%               the RF are given this R-square is not weighted
%               (see curve fitting toolbox online help:
%                evaluating the goodness of fit)
%    Vres     - residual variance
%    conf     - confidence intervals for parameters
%               [ncells,7] = size(conf)
%    exitflag - info whether optimization successful (<1, if no convergence)

% version 1.0 {Sebastian Bitzer}
function [fits, residuals, R2, Vres, conf, exitflag] =...
                           fitGaborToRF(areaFile,cells)
```

```
load(areaFile,'areas');

ncells = length(cells);

residuals = zeros(ncells*4,20*20);
fits      = zeros(ncells*4,9);
R2        = zeros(ncells*4,1);
Vres      = zeros(ncells*4,1);
conf      = zeros(ncells*4,7);
exitflag  = zeros(ncells*4,1);


for i = 1:ncells                % for cells
    cell = cells(i)
    [rf(1,:,:), rf(2,:,:), rf(3,:,:), rf(4,:,:)] = read_cell(cell,areas);
    for sub = 1:4                   % for each subunit
        sub

        dataX = [1:400]';
        RF = squeeze(rf(sub,:,:));
        dataY = RF(:);


%       START VALUES:
        [y0,x0] = gravity_center(RF);               % determine center of RF
        t1 = round(y0); t2 = round(x0);             % temporary variables
        As = mean(mean(RF(t1-1:t1+1,t2-1:t2+1)));   % 3x3 surrounding of center

        scalar = pi/normpdf(0,0,30);                % peak of the normal pdf
                                                    % should be pi later
        % if surround of center is near 0, then center is presumably at
        % edge of sinusoid, i.e. high phase shift; if As<0 gabor should be
        % inverted by phase shift
        phis = normpdf(As,0,30)*scalar+(As<0)*pi;
        A = max(dataY);                             % amplitude = max of RF
        [t1,f,t2] = get_ori_freq_fft(RF);           % frequency
        ori = findOri(f, As, dataY);                % orientation of sinusoid
        Wp = 8;                                     % width of minor gaussian
        Wq = 10;                                    % width of major gaussian
```

```
        Ao = 0;                                         % amplitude offset

        start = [x0,y0,Wp,Wq,phis,ori,A,f,Ao];


%       FITTING
        offset = (i-1)*4+sub;
        if (mod(offset,2) == 0)     % if right RF start with left RF values
            [fitslast, residuals(offset,:), R2(offset), Vres(offset), ...
             conf(offset,:), exitflag(offset)] = ...
                      fitGabor2(dataX,dataY,fitslast,0);
        else                        % if left RF start with heuristic
            [fitslast, residuals(offset,:), R2(offset), Vres(offset), ...
             conf(offset,:), exitflag(offset)] = ...
                      fitGabor2(dataX,dataY,start,0);
        end
        fits(offset,:) = fitslast;
    end
end
```

RFAnalysis.m

```
% [ori,ori90,f,phaseDifLR,phaseDif,phaseDifPx, posDif, posDifRel] = ...
%                RFAnalysis(fits)
%
% calculates RF properties from parameters of RF fits, only difference to
% RFAnalysisSVD.m is computation of position disparity
%
% output:
%   ori         - RF orientation from 0 (horizontal) over pi/2 (vertical) to pi
%                 [ncells,nsubunits] = size(ori);
%   ori90       - orientation wrt obliqueness from 0 (horizontal) to pi/2
%                 (vertical)
%                 [ncells,nsubunits] = size(ori90);
%   f           - RF spatial frequency
%                 [ncells,nsubunits] = size(f);
%   phaseDifLR  - phase difference between 1st and 2nd subunit RFs of one
%                 cell from 0 to 2pi
%                 [ncells,2] = size(phaseDifLR); (:,1)-left, (:,2)-right
%   phaseDif    - phase difference between left and right RFs of one
```

```
%                 subunit from 0 to 2pi
%                 [ncells,nsubunits] = size(phaseDif);
%   phaseDifPx  - phase difference between left and right RFs of one
%                 subunit in pixel
%                 [ncells,nsubunits] = size(phaseDifPx);
%   posDif      - position disparity between left and right RFs of one
%                 subunit in pixel as defined in Anzai et al. 1999 (pos-phase)
%                 [ncells,nsubunits] = size(posDif);
%   posDifRel   - relative position difference by subtracting 2nd from 1st
%                 subunit in pixel
%                 [ncells,1] = size(posDifRel);
%
% further info about disparity calculation in Sebastian Bitzer's bachelor's
% thesis

% version 1.0 {Sebastian Bitzer}
function [oriSin,ori90,fSub,phaseDifLR,phaseDifSub,phasePx, ...
          posDifSub, posDif] = ...
                 RFAnalysis(fits)

%% Prolog
    nfits = size(fits,1);
    even  = [1:floor(nfits/2)]*2;
    odd   = [1:ceil(nfits/2)]*2-1;
    sub2  = [1:floor(nfits/4)]*2;
    sub1  = [1:ceil(nfits/4)]*2-1;
    first = find(mod(4:403,4) < 2);
    second = setdiff(1:400,first);


%% Orientation
%  note that these transformations are appropriate for orientation only,
%  if you take the transformed orientations as parameters for gabor2, the
%  RF may invert (because the whole gabor is rotated)
    oriSin = normOri(fits(:,6));

    function oriSub = normOri(oriFit)
        ori = oriFit + pi/2;        % make -pi/2 to 0 (horizontal)
        ori = mod(ori,pi);          % discard multiples of pi and transfrom
                                    % negative to positive orientations
```

```
        oriM = mean(reshape(ori,2,nfits/2))';      % calculate mean
                                                   % between right and left
        t = find(oriM > min(ori(odd),ori(even))+pi/4); % if one ori at 0
                                                       % the other at pi
        oriM(t) = ori(odd(t));      % take ori of left RF as mean

        oriSub(:,1) = oriM(sub1);
        oriSub(:,2) = oriM(sub2);
    end

    ori90 = oriSin;                 % only horizontal-vertical discrimination
    big   = find(oriSin>pi/2);      % find orientation >90 degrees
    ori90(big) = pi - ori90(big);   % transform to corresp. orientation <90


%% Frequency
    f = fits(:,8);
    f = mean(reshape(f,2,nfits/2))';
    fSub(:,1) = f(sub1);
    fSub(:,2) = f(sub2);


%% Phase
%  note: a positive phase means peak of tuning curve is shifted to left
    phase = fits(:,5);
    phase = rem(phase,2*pi);
    big   = find(abs(phase)>pi);
    phase(big) = phase(big) - sign(phase(big))*2*pi;

%  first subunit phase - second subunit phase
    phaseDif = phase(first)-phase(second);
big   = find(abs(phaseDif)>pi);
phaseDif(big) = phaseDif(big) - sign(phaseDif(big))*2*pi;
    phaseDifLR(:,1) = phaseDif(sub1);
    phaseDifLR(:,2) = phaseDif(sub2);

%  note: positive phase difference means peak of right RF is shifted to
%        the left compared to left RF (far)
    phaseDif = phase(even)-phase(odd);
big   = find(abs(phaseDif)>pi);
```

```
phaseDif(big) = phaseDif(big) - sign(phaseDif(big))*2*pi;
    phaseDifSub(:,1) = phaseDif(sub1);
    phaseDifSub(:,2) = phaseDif(sub2);
    phasePx = phaseDifSub/(2*pi) ./ fSub;


%% Position (with second subunit as reference cell)
    posX = fits(:,1);                       % RF centres x-value
    posY = fits(:,2);                       % RF centres y-value
    fl = first(sub1);                       % first subunit left RF
    fr = first(sub2);                       % first subunit right RF
    sl = second(sub1);                      % second subunit left RF
    sr = second(sub2);                      % second subunit right RF

    posDifSub = zeros(nfits/4,2);

    % following calculations based on consideration of lines running
    % through RF centres with slope corresponding to RF orientation
    % definition of a line: y = m*x + n
    % assumption: left and right RFs have equal orientations
    % subunit 1
    m  = tan(oriSin(:,1)+1e-10);            % slope corresponding to ori
    mo = tan(oriSin(:,1)+1e-10+pi/2);       % slope perpendicular to ori
    n1 = posY(fl) - m .*posX(fl);           % line 1: left RF
    n2 = posY(fl) - mo.*posX(fl);           % line 2: perpendicular to RFs,
                                            % intersecting with line 1 at
                                            % left RF centre
    n3 = posY(fr) - m .*posX(fr);           % line 3: right RF
    x = (n2-n3)./(m-mo);                    % intersection line 2 and 3
    y = m.*x+n3;                            % y-value of intersection

    % pos disp is length of vector from left RF centre to point of intersection
    for i=1:100
        posDifSub(i,1) = norm([x(i),y(i)]-[posX(fl(i)),posY(fl(i))]);
    end
    % negative, if line 1 is left from line 3
    posDifSub(:,1) = sign(posX(fl)-x).*posDifSub(:,1);

    % subunit 2
    m  = tan(oriSin(:,2)+1e-10);
```

```
    mo = tan(oriSin(:,2)+1e-10+pi/2);
    n1 = posY(sl) - m .*posX(sl);
    n2 = posY(sl) - mo.*posX(sl);
    n3 = posY(sr) - m .*posX(sr);
    x = (n2-n3)./(m-mo);
    y = m.*x+n3;
    for i=1:100
        posDifSub(i,2) = norm([x(i),y(i)]-[posX(sl(i)),posY(sl(i))]);
    end
    posDifSub(:,2) = sign(posX(sl)-x).*posDifSub(:,2);

    % this is equal to the computation of relative position disparity as
    % given in Anzai et al. 1999 (position vs. phase), which would have
    % been simpler to do, but in this way I have absolute position
    % disparities, too
    % second subunit assumed to have posDif=0
    posDif = posDifSub(:,1)-posDifSub(:,2);
end
```

Interaction profiles

iProfile.m

```
% [iPrf,marg] = iProfile(fits,cells,res)
%
% calculates binocular interaction profiles with given resolution res from
% RF fits for given cells
%
% profile for one monocular subunit RF is defined as values along the line
% perpendicular to RF orientation through RF centre, x-axis of the
% resulting 1D-profile is defined wrt RF centre, for the binocular
% interaction profiles values at different x-coordinates in left and right
% monocular profiles are combined and further processed according to the
% cell model
%
% output:
%   iPrf - binocular interaction profiles, columns correspond to differnt
%          positions in left RFs, rows correspond to different position in
%          right RFs, third dimension for different cells
%   marg - range of positions (x-coordinates) used in iPrf
```

```
%          => iPrf(1,:)     represents position -marg in RF space
%             iPrf(end,:)   represents position marg in RF space
%
% further info in Sebastian Bitzer's bachelor's thesis and in Anzai et al.'s
% 1999 series of papers to disparity
%
% for how to display interaction profiles analogous to Anzai et al. see
% iPrfFig.m, alternatively imagesc(imrotate(iPrf,45)) will also do, but is
% not as nice

% version 1.0 {Sebastian Bitzer}
function [iPrf,marg] = iProfile(fits,cells,res)

    dev = 2.2;          % allowed deviation from center [10.5,10.5]
                        % all higher frequency RFs lie within <= 2.2
    marg = 10.5-dev-1;  % margin (largest distance from center in order
                        % to stay in 20x20 patch given possible deviation)
    step = 2*marg/res;  % step size in order to have res+1 steps
    nsteps = res+1;

    iPrf = zeros(nsteps,nsteps,length(cells));
    iPrfMatch = zeros(nsteps,nsteps);
    iPrfMisMatch = zeros(nsteps,nsteps);

    for i=1:length(cells)

%         c = [10.5;10.5];

        % subunit 1 left
        c = fits(cells(i)*4-3,1:2)';
        if (max(abs(c-[10.5;10.5])) > dev)
            c = [10.5;10.5];
        end
        vDist = calcLine(-marg:step:marg, fits(cells(i)*4-3,6));
        params = fits(cells(i)*4-3,:);
        subL(1,:) = gabor1Prf(params, repmat(c,1,size(vDist,2))-vDist);

        % subunit 2 left
        c = fits(cells(i)*4-1,1:2)';
        if (max(abs(c-[10.5;10.5])) > dev)
```

```
        c = [10.5;10.5];
    end
    vDist = calcLine(-marg:step:marg, fits(cells(i)*4-1,6));
    params = fits(cells(i)*4-1,:);
    subL(2,:) = gabor1Prf(params, repmat(c,1,size(vDist,2))-vDist);

    % subunit 1 right
    c = fits(cells(i)*4-2,1:2)';
    if (max(abs(c-[10.5;10.5])) > dev)
        c = [10.5;10.5];
    end
    vDist = calcLine(-marg:step:marg, fits(cells(i)*4-2,6));
    params = fits(cells(i)*4-2,:);
    subR(1,:) = gabor1Prf(params, repmat(c,1,size(vDist,2))-vDist);

    % subunit 2 right
    c = fits(cells(i)*4,1:2)';
    if (max(abs(c-[10.5;10.5])) > dev)
        c = [10.5;10.5];
    end
    vDist = calcLine(-marg:step:marg, fits(cells(i)*4,6));
    params = fits(cells(i)*4,:);
    subR(2,:) = gabor1Prf(params, repmat(c,1,size(vDist,2))-vDist);

    for j = 1:size(vDist,2)
        iPrfMatch(j,:) = sqrt((subL(1,j)+subR(1,:)).^2 + ...
                              (subL(2,j)+subR(2,:)).^2);
        iPrfMisMatch(j,:) = sqrt((subL(1,j)-subR(1,:)).^2 + ...
                                 (subL(2,j)-subR(2,:)).^2);
    end
    iPrf(:,:,i) = iPrfMatch - iPrfMisMatch;
end

function fxy = gabor1Prf(params,xy)
    x0     = params(1);             % center x
    y0     = params(2);             % center y
    Wp     = params(3);             % width of minor gaussian
    Wq     = params(4);             % width of majof gaussian
    phi    = params(5);             % phase of sinusoid
    theta  = params(6);             % orientation of sinusoid
```

```
        gam    = params(6);             % orientation of gaussian
        A      = params(7);             % amplitude
        f      = params(8);             % frequency of sinusoid
        A0     = params(9);             % amplitude offset

        x = xy(1,:);
        y = xy(2,:);

        p = (x-x0)*cos(gam) + (y-y0)*sin(gam);
        u = (x-x0)*cos(theta) + (y-y0)*sin(theta);

        fxy = A0 + A*exp(-(p.^2)/(2*Wp^2)) .* ...
              cos(2*pi*f*u + phi);
    end

    % calculate vector pointing "dist" units in the direction of "slope"
    % (used to compute datapoints on the line orthogonal to orientation)
    % it is important to understand that slope assumes an increase for
    % higher values of y, but in this context of images and rows a
    % bigger y value means to go down
    function vDist = calcLine(dist, ori)
        o = mod(ori+pi/2,pi);               % normalise and orthogonalise
        slope = -tan(o+1e-10+pi/2);         % slope of line orthogonal to
                                            % orientation ori in rad
        v = [1,slope];                      % make a vector from slope
        v = v/norm(v);                      % normalise the slope vector
        vDist = v'*dist(:)';                % scale by the given distance
        vDist(1,:) = -vDist(1,:);           % reflect about x-axis to correct
                                            % for interpretation of y and make
                                            % negative dist left of center
    end
end
```

SVDAnalysis.m

```
% [w,fitsSVD,R2SVD,exitSVD] = SVDanalysis(iPrf,marg,RFfits,cells,sav)
% decomposes interaction profile "iPrf" of cells given in "cells" with
% singular value decomposition (SVD) and fit resulting left and right
% parts of the first two components with a 1D Gabor in order to obtain
% fitted parameters, "marg" is the margin used during creation of "iPrf"
```

```
% and defines the x values for fitting, "RFfits" are used as heuristic for
% fitting and to fill corresponding slots in output "fits", right profiles
% get fitted parameters of left profiles as starting values, in any case
% Gabor is first fitted with frequency held at frequency of RFfits, if this
% fit accounts for less than 97% of the data variance, Gabor is refitted
% with free frequency, fit with greater R2 is taken
% if sav is on (1) results will also be stored in a file SVDAnalysis<date>.mat
%
% outputs:
%   w    - matrix containing weights for each component of each cell
%          [ncomponents,ncells] = size(w)
%   fitsSVD - matrix containing fitted parameters for first and second
%          components in the format of RFfits, orientation is inserted from
%          RFfits, fields of second gaussian are left zero, can be used
%          with RFAnalysisSVD.m
%          [ncells*4,9] = size(fitsSVD)
%   R2SVD   - R-square of Gabor fits
%          [ncells*4,1] = size(R2SVD)
%   exitSVD - exitflags of optimisation procedure (<1 if not converged)
%          [ncells*4,1] = size(exitSVD)
%
% version 1.0 {Sebastian Bitzer}
function [w,fitsSVD,R2SVD,exitSVD] = SVDanalysis(iPrf,marg,RFfits,cells,sav)

    [nsteps,nsteps,ncellsIn] = size(iPrf);
    ncells = length(cells);
    L = zeros(nsteps);
    R = zeros(nsteps);
    W = zeros(nsteps);
    w = zeros(nsteps,ncells);
    R2SVD = zeros(ncells*4,1);
    pars = zeros(ncells*4,6);
%     conf = zeros(ncells*4,6);     % different in fitGabor1 and fitGabor1f
    exitSVD = zeros(ncells*4,1);
    fitsSVD = zeros(ncells*4,9);

    step = 2*marg/(nsteps-1);
    x = -marg:step:marg;

    for i = 1:ncells
```

```
        cells(i),

%% SVD
        [L,W,R] = svd(squeeze(iPrf(:,:,cells(i))));
        w(:,i) = diag(W);       % singular values (weights)
        L12 = L(:,[1,2]);       % left first and second components only
        R12 = R(:,[1,2]);       % right first and second components only

        % % to show first component:
        %t = zeros(nsteps), t(1,1) = W(1,1);
        %figure, imagesc(imrotate(L*t*R',45))
        % % alternatively:
        %figure, imagesc(imrotate(L12(:,1)*w(1)*R12(:,1)',45))

%% fit first component left
        y = L12(:,1);
        start = [max(y)*2, 0, marg, RFfits(cells(i)*4-3,8), 0, 0];
        off = i*4-3;
        [R2SVD(off),exitSVD(off),pars(off,:)] = fitSVD(start);

%% fit first component right
        y = R12(:,1);
%         start = [max(y)*2, 0, marg, RFfits(cells(i)*4-3,8), 0, 0];
        start = pars(i*4-3,:);
        start(1) = max(y)*2;
        off = i*4-2;
        [R2SVD(off),exitSVD(off),pars(off,:)] = fitSVD(start);

%% fit second component left
        y = L12(:,2);
        start = [max(y)*2, 0, marg, RFfits(cells(i)*4-3,8), 0, 0];
        off = i*4-1;
        [R2SVD(off),exitSVD(off),pars(off,:)] = fitSVD(start);

%% fit second component right
        y = R12(:,2);
%         start = [max(y)*2, 0, marg, RFfits(cells(i)*4-3,8), 0, 0];
        start = pars(i*4-1,:);
        start(1) = max(y)*2;
        off = i*4;
```

```
        [R2SVD(off),exitSVD(off),pars(off,:)] = fitSVD(start);
    end

%% fitting function
    function [R2,exit,pars] = fitSVD(strt)
        R2a = 0; exita = 0;
        [parsf,res,R2f,Vres,conf,exitf] = fitGabor1f(x, y, strt, 0);
        if (exitf==0 | R2f < 0.97)
            [parsa,res,R2a,Vres,conf,exita] = fitGabor1(x, y, strt, 0);
        end
        if (R2f > R2a)
            R2 = R2f;
            exit = exitf;
            pars = parsf;
        else
            R2 = R2a;
            exit = exita;
            pars = parsa;
        end
    end

%% collecting fitted parameters
%  this format is only used for compatibility to RFAnalysis.m
%  notice the empty values of the second gaussian
    RFindex = repmat(cells,4,1) * 4 - repmat([3:-1:0]',1,ncells);
    fitsSVD = [pars(:,2),...                % envelope position
            zeros(ncells*4,1),...           % env pos y (not applicable)
            pars(:,3),...                % envelope width
            zeros(ncells*4,1),...           % env width y (not applicable)
            pars(:,5),...                % phase
            RFfits(RFindex(:),6),...        % orientation
            pars(:,1),...                % amplitude
            pars(:,4),...                % frequency
            pars(:,6)];                  % amplitude offset

    if (nargin > 4 & sav == 1)
        dateStr = strcat(datestr(now,'dd'),datestr(now,'mm'));
        save(strcat('SVDAnalysis',dateStr,'.mat'),'*SVD','w','marg','iPrf','cells')
    end
end
```

RFAnalysisSVD.m

```
% [ori,ori90,f,phaseDifLR,phaseDif,phaseDifPx, posDif, posDifRel] = ...
%                RFAnalysisSVD(fits)
%
% calculates RF properties from parameters of RF fits, only difference to
% RFAnalysis.m is computation of position disparity, therefore only this is
% printed here

% version 1.0 {Sebastian Bitzer}
function [oriSin,ori90,fSub,phaseDifLR,phaseDifSub,phasePx, ...
          posDifSub, posDif] = ...
                 RFAnalysisSVD(fits)

       ...
       calculation of oriSin, ori90, fSub,
       phaseDifLR, phaseDifSub, phasePx omitted
       see RFAnalysis.m
       ...

%% Position (with second subunit as reference cell)
    % left-right: left shift of right is positive (far)
    posDif = fits(odd,1) - fits(even,1);
    posDifSub(:,1) = posDif(sub1);
    posDifSub(:,2) = posDif(sub2);
    % second subunit assumed to have posDif=0
    posDif = posDifSub(:,1)-posDifSub(:,2);
end
```

RDS

genRDS.m

```
% RDS = genRDS(RDSsize, dots, dens)
%
% generate Random Dot Stimuli of size "RDSsize" [row,col]
% with density "dens" of black and white dots in %/100 of image
% and size of dots in "dots" [row,col], dots are allowed
% to overlap
% e.g. genRDS([20,20], [1,1], 0.5)
%
```

```
% output:
%   RDS - RDS with requested properties

% version 1.0 {Sebastian Bitzer}
function RDS = genRDS(RDSsize, dots, dens)

RDS_big = zeros(RDSsize(1)+2*dots(1),RDSsize(2)+2*dots(2));

nPix = prod(RDSsize);

nDots = floor(nPix*dens/prod(dots));

if (nDots < 2)
    error(strcat('density or dot size too high for size of RDS ', ...
                 '(dots do not fit into figure with given density)');
end

dotPos = [floor(rand(nDots, 1)*(RDSsize(1)-0.01))+1, ...
          floor(rand(nDots, 1)*(RDSsize(2)-0.01))+1];

for i=1:nDots
    offset = [dotPos(i,1)+1, dotPos(i,2)+1];
    RDS_big(offset(1):offset(1)+dots(1)-1, offset(2):offset(2)+dots(2)-1) = ...
           mod(i,2)*-1 + mod(i+1,2);
end

RDS = RDS_big(1+dots(1):end-dots(1), 1+dots(2):end-dots(2));

% when dots have values -1 and 1, then mean is already 0
% (except there are unequal numbers of black and white dots: covered dots)
% if genRDS used with RDStoActSingle mean normalisation should be done there
% RDS = RDS - mean(mean(RDS));
```

RDStoActSingle.m

```
% [cellAct,m,V] = RDStoActSingle(cell, areas, iter, dotDens, mode, varargin)
%
% calculates activity of a cell to random-dot stereograms
% "cell" is a cell number (id) which is used to get corresponding cell
% properties from "areas", "iter" determines the number of used RDS with
```

```
% "dotDens" dot densitiy and further properties dependent on "mode":
%     'disp'  : RDS with disparity
%               varargin: first - disparity in pixels of the RDS patch
%                         second - flag whether disparity should be
%                                  horizontal (0) or orthogonal to preferred
%                                  orientation (1)
%                          third - "ori" from RFAnalysis, is used to produce
%                                  a disparity which is always orthogonal to
%                                  the preferred orientation of the cell
%     'mono'  : RDS in one eye, in the other grey background
%               varargin: first - 'left' if RDS in left eye, else right eye
%     'uncorr': uncorrelated RDS in both eyes
%     'blank' : blank patch to both eyes (patch filled with 1)
%
% output:
%   cellAct - calculated activities to different RDS with given properties
%             [1,iter] = size(cellAct)
%   m       - mean of cellAct
%   V       - variance of cellAct
%
% example:
% [cellAct,m,V] = RDStoActSingle(1, areas, 1000, 0.25, 'disp', -1, 1, ori)
%
% for further info esp. about definition of disparity see
% Sebastian Bitzer's bachelor's thesis

% version 1.0 {Sebastian Bitzer}
function [cellAct,m,V] = RDStoActSingle(cell, areas, iter, dotDens, ...
                                        mode, varargin)

[wLs1, wRs1, wLs2, wRs2] = read_cell(cell, areas, []);
[m,n] = size(wLs1);
norm  = areas{2}.theNorm;
nSub  = areas{2}.nrSubunits;

cellAct = zeros(1,iter);

w(:,1) = [wLs1(:); wRs1(:)];        % subunit 1
w(:,2) = [wLs2(:); wRs2(:)];        % subunit 2
```

```
if (strcmp(mode,'disp'))
    disp = varargin{1};
    if (abs(disp) > n || abs(disp) > m)
        warning(strcat('disparity is not allowed to exceed the ', ...
                       'dimensions of the image: switching to mode "uncorr"'))
        mode = 'uncorr';
    end

    if (varargin{2} == 1)
        ori = varargin{3}(cell,1);        % orientation of 1st subunit
                                          % (should be equal to 2nd)
        slope = -tan(ori+1e-10+pi/2);     % slope of line orthogonal to
                                          % orientation ori in rad
        oDisp = calcOriDisp(disp, slope);
    else
        oDisp = [0,disp];
    end
end

for i = 1:iter                            % for iter different RDS

    switch mode
        case 'disp'
            RDS = genRDS([m,n].*3, [1,1], dotDens);    % generate big RDS

            % RDS for left and right RFs (made from big RDS)
            rRDS = reshape(RDS(m+1:2*m, n+1:2*n), 1, m*n);
            rRDS = rRDS - mean(rRDS);
            lRDS = reshape(RDS(m+1+oDisp(1):2*m+oDisp(1), ...
                               n+1-oDisp(2):2*n-oDisp(2)), 1, m*n);
            lRDS = lRDS - mean(lRDS);
        case 'mono'
            if (strcmp(varargin{1}, 'left'))
                lRDS = reshape(genRDS([m,n], [1,1], dotDens),1,m*n);
                lRDS = lRDS - mean(lRDS);
                rRDS = ones(1,m*n);
            else
                lRDS = ones(1,m*n);
                rRDS = reshape(genRDS([m,n], [1,1], dotDens),1,m*n);
                rRDS = rRDS - mean(rRDS);
```

```
            end
        case 'uncorr'
            lRDS = reshape(genRDS([m,n], [1,1], dotDens),1,m*n);
            lRDS = lRDS - mean(lRDS);
            rRDS = reshape(genRDS([m,n], [1,1], dotDens),1,m*n);
            rRDS = rRDS - mean(rRDS);
        case 'blank'
            lRDS = ones(1,m*n);
            rRDS = ones(1,m*n);
        otherwise
            error(strcat('unknown mode - choose between ', ...
                         '"disp", "mono", "uncorr" and "blank"')
    end

    I = [lRDS, rRDS];         % input: combined left + right RDS

    subAct = I*w;                                % activities of subunits

    cellAct(i) = sum(subAct.^norm).^(1/norm);   % activity of cell
end

m   = mean(cellAct);         % mean of cells response (over RDS)
V   = var(cellAct);          % variance of cells respsonse


% calculate oriented disparity offsets
% (disparity orthogonal to preferred orientation of cell)
function oDisp = calcOriDisp(disp, slope)

v = [slope,1];                    % make a vector from slope
v = v/norm(v);                    % normalise the slope vector
oDisp = round(v*disp);            % scale by the given disparity
```

RDStoAct.m

```
% [act,m,V] = RDStoAct(areas, iter, dotDens, mode, varargin)
%
% calculates activity of all cells in "areas" to random-dot stereograms,
% thereto calls RDStoActSingle.m
% "iter" determines the number of used RDS with "dotDens" dot densitiy and
```

```
% further properties dependent on "mode":
%     'disp'  : RDS with disparity
%               varargin: first - disparity in pixels of the RDS patch
%                         second - flag whether disparity should be
%                                  horizontal (0) or orthogonal to preferred
%                                  orientation (1)
%                          third - "ori" from RFAnalysis, is used to produce
%                                  a disparity which is always orthogonal to
%                                  the preferred orientation of the cell
%     'mono'  : RDS in one eye, in the other grey background
%               varargin: first - 'left' if RDS in left eye, else right eye
%     'uncorr': uncorrelated RDS in both eyes
%     'blank' : blank patch to both eyes (patch filled with 1)
%
% output:
%   act - calculated activities to different RDS with given properties
%         [iter,ncells] = size(cellAct)
%   m   - mean of cellAct
%         [1,ncells] = size(m)
%   V   - variance of cellAct
%         [1,ncells] = size(V)
%
% example:
% [act,m,V] = RDStoAct(areas, 1000, 0.25, 'disp', -1, 1, ori)
%
% for further info esp. about definition of disparity see
% Sebastian Bitzer's bachelor's thesis

% version 1.0 {Sebastian Bitzer}
function [act,m,V] = RDStoAct(areas, iter, dotDens, mode, varargin)

cellAct = zeros(1,iter);
act     = zeros(iter,100);
m       = zeros(1,100);
V       = zeros(1,100);

for j = 1:areas{2}.nrNeurons                    % for every neuron
                                                % get activity
    [cellAct, m(j), V(j)] = ...
                RDStoActSingle(j, areas, iter, dotDens, mode, varargin{:});
```

```
    act(:,j) = cellAct';                          % and store it

end
```

calcActs.m

```
% script for computing activities of cells to RDS as well as
% MUratio, DDI and ODI
%
% needed variables in workspace:
%        areas    - containing the RFs
%        ori      - orientation of RFs [ncells,nsubunits]=size(ori)
%
% saves all activities in file with name RDSAct<date><options>.mat

% version 1.0 {Sebastian Bitzer}


iter  = 1000;   % number of iterations
orthFlag = 0;   % use disparity orthogonal to preferred orientation (1)
                % or horizontal disparity (0)
dispRange = 20; % range of disparities over which should be sampled
                % (-dispRange:dispRange)
dotDens = 0.25; % average dot density in RDS in %
                % 0.25 for Prince et al, 0.5 for Read and Cumming

ncells = areas{2}.nrNeurons;
ndisps = dispRange*2+1;


%% producing activities to RDS stimuli

    'mono left'
[monoLAct,  monoLV]  = RDStoAct(areas, iter, dotDens, 'mono','left');
    'mono right'
[monoRAct,  monoRV]  = RDStoAct(areas, iter, dotDens, 'mono','right');
    'uncorrelated'
[uncorrAct, uncorrV] = RDStoAct(areas, iter, dotDens, 'uncorr');
    'blank'
[blankAct,  blankV]  = RDStoAct(areas, 1   , dotDens, 'blank');
```

```
monoLActM   = mean(monoLAct);           % monocular left activities (mean)
monoRActM   = mean(monoRAct);           % monocular right activities (mean)
uncorrActM  = mean(uncorrAct);          % activities to uncorrelated RDS
blankActM   = mean(blankAct);           % activities to blank stimuli
domMonoAct  = max([monoLActM; monoRActM]);


%% producing activities to RDS stimuli with disparity

    'disparity:'
dispAct   = zeros(ndisps,iter,ncells);  % raw activities
dispActM  = zeros(ndisps,ncells);       % mean activities
dispV     = zeros(ndisps,ncells);       % variance of activities
for i = -dispRange:dispRange
    i
    dOffset = i+dispRange+1;
    if orthFlag
        [dispAct(dOffset,:,:), dispActM(dOffset,:), dispV(dOffset,:)] = ...
                RDStoAct(areas, iter, dotDens, 'disp', i, orthFlag, ori);
    else
        [dispAct(dOffset,:,:), dispActM(dOffset,:), dispV(dOffset,:)] = ...
                RDStoAct(areas, iter, dotDens, 'disp', i, orthFlag);
    end
end



%% ratio of dominant monocular to uncorrelated response

MUratio = domMonoAct./uncorrActM;


%% disparity discrimination index (DDI)

[m,n] = size(dispActM);
Rmax = max(dispActM);
Rmin = min(dispActM);
SSE  = sum(squeeze(sum((dispAct - ...
                        repmat(reshape(dispActM,m,1,n),[1,iter,1])).^2)));
```

```
RMS  = sqrt(SSE/(iter-m));

DDI = (Rmax - Rmin) ./ ((Rmax - Rmin) + 2*RMS);


%% ocular dominance index (ODI)

ODI = monoLActM./(monoLActM + monoRActM);


%% save variables

dateStr = strcat(datestr(now,'dd'),datestr(now,'mm'));
save(strcat('RDSAct',dateStr,'i',num2str(iter),'o',num2str(orthFlag),'d',...
            num2str(dispRange),'dD',num2str(dotDens),'.mat'), ...
     'iter', 'orthFlag', '*Act', '*ActM', 'dispV', 'MUratio', 'RMS', ...
     'Rmax', 'Rmin', 'DDI', 'ODI','dotDens','dispRange');
```

fitRDS.m

```
% [parsRDS, R2RDS, confRDS, goodCellsRDS, exitflagRDS] = fitRDS(actFile)
%
% fits 1D-Gabor functions to disparity tuning data contained in actFile
% from which it loads dispActM and dispV
% heuristic start values for fitting are chosen
%
% output:
%       parsRDS  - the fitted parameters of the gabor
%       R2RDS    - R-square statistics (variation explained by fit in %)
%                  as no variances for the individual pixels of
%                  the RF are given this R-square is not weighted
%                  (see curve fitting toolbox online help: evaluating the
%                  goodness of fit)
%       confRDS  - confidence intervals for parameters
%   goodCellsRDS - cells with R2>0.75 and no complex confidence intervals
%    exitflagRDS - info whether optimization successful (<1, if no convergence)
%
% additionally saves all fits in file with name from actFile in which
% "RDSAct" is replaced with "RDSFits"
```

```
% version 1.0 {Sebastian Bitzer}
function [pars, R2, conf, goodCells, exitflag] = fitRDS(actFile)

load(actFile,'dispActM','dispV');

[nData, cells] = size(dispActM);
nPar  = 6;

pars   = zeros(cells,nPar);         % parameters for Gabor
res    = zeros(cells,nData);        % residuals
R2     = zeros(1,cells);            % R-square (explained variation)
C      = zeros(nPar-1,nPar-1,cells);% covariance matrix
varres = zeros(1,cells);            % residual variance
conf   = zeros(cells,nPar-1);       % confidence intervals
                                    % (one less because frequency via FFT)
exitflag = zeros(1,cells);


%% fitting
dataX = -20:20;

for cell=1:cells
    cell

    dataY = dispActM(:,cell)';

%   START VALUES:
    % amplitude
    A = std(dataY)*2;

    % frequency
    resol = 1000;   % resolution
    y = [1:resol]./resol;
    Y = fft(hann(size(dataX,2))'.*(dataY-mean(dataY)),resol);
    Pyy = Y.*conj(Y);
    Df = y(find(Pyy(1:resol/2) == max(Pyy(1:resol/2))));

    % amplitude offset
    if (dataX==[-20:20])
        Ao = mean([dataY(1),dataY(end)]);
```

```
    else
        Ao = mean(dataY);
    end

    start = [A,0,5,Df,0,Ao];

%   FITTING
    [pars(cell,:), res(cell,:), R2(cell), varres(cell), ...
     conf(cell,:), exitflag(cell)] = ...
        fitGabor1f(dataX, dataY, start, 0, dispV(:,cell));
end


%% saving

% find cells with 75% variance explained and sampled disparity
% confidence interval for gabor phase non-complex
goodCells = find(R2'>=0.75 & abs(imag(conf(:,2)))==0);

parsRDS = pars;
R2RDS = R2;
confRDS = conf;
goodCellsRDS = goodCells;
exitflagRDS = exitflag;

if (length(strfind(actFile,'RDSAct')) == 0)
    dateStr = strcat(datestr(now,'dd'),datestr(now,'mm'));
    fname = strcat('RDSFits',dateStr,'.mat');
    warning('\nno standard actFile-string, saving in: %s',fname);
    save(fname, '*RDS');
else
    save(strrep(actFile,'RDSAct','RDSFits'), '*RDS');
end
```

Displaying

showRFAnalysis.m

```
% script containing collection of diagrams to visualise RFAnalysis data
% analogous to Anzai et al. (1999)
```

```
% it is not recommended to run this script in the command line since
% several figure windows will open simultaneously, use cell mode instead to
% run selected cells containing one diagram each
% cell "prolog" needs to be run once before diagrams can be constructed
% for final diagrams which also display data from Anzai see directory figs

% version 1.0 {Sebastian Bitzer}


%% prolog for SVDAnalysis data
%  has different calculation of position disparity
[ori,ori90,f,phaseDifLR,phaseDif,phaseDifPx, posDif, posDifRel] = ...
    RFAnalysisSVD(fitsSVD);
good = find(abs(posDif(:,1)) < 10);
p = 20; % patch size

%% prolog for comparison with simple cells
[ori,ori90,f,phaseDifLR,phaseDif,phaseDifPx, posDif, posDifRel] = ...
    RFAnalysis(fits);
good = find(abs(posDif(:,1)) < 10);
p = 20; % patch size

%% weights of SVD
%  works only when w of SVDAnalysis is in workspace
figure,
hold on

tmp = cumsum(w./repmat(sum(w),101,1));
m = mean(tmp');
s = std(tmp');
h = errorbar(m(1:16)*100,s(1:16)*100,'-^k');
set(h, 'MarkerFaceColor',[0,0,0.8],'MarkerSize',8);

tmp = w./repmat(sum(w),101,1);
m = mean(tmp');
s = std(tmp');
h = errorbar(m(1:16)*100,s(1:16)*100,'-ok');
set(h, 'MarkerFaceColor',[0,0.8,0],'MarkerSize',8);

axis([0.5,16,0,100]);
```

```
set(gca,'XTick',[1:16],'XTickLabelMode','manual', ...
    'XTickLabel',{'1','','','4','','','7','','','10','','','13','','','16'});
set(gca,'YTick',[0,25,50,75,100]);
set(gca,'PlotBoxAspectRatio',[1.5,1,1]);
grid on
legend('cumulative','individual component','Location','East')
xlabel('component number','FontSize',14)
ylabel('mean percentage of total variance','FontSize',14)
hold off
clear m s tmp


%% histograms
n = 8;          % 2*n+1 is number of bins
figure, set(gcf,'Position',[200 10 385 685]);
subplot(3,1,1),
    hist(phaseDif(good,1),2*n+1);
    set(get(gca,'Children'),'FaceColor',[.7,.9,.7])
    set(gca,'XLim',[-3.5,3.5])
    set(gca,'XTick',[-pi,-2/3*pi,-pi/3,0,pi/3,2/3*pi,pi], ...
            'XTickLabelMode','manual', ...
            'XTickLabel',[-180,-120,-60,0,60,120,180]);
    xlabel('phase disparity (deg PA)','FontSize',14);
    text(0.3,1.2,'Simulation','FontSize',16,'Units','normalized');
    box off
subplot(3,1,2),
hist(phaseDifPx(good,1),[-n:n]*20/n);
    set(gca,'XLim',[-20,20])
    set(get(gca,'Children'),'FaceColor',[.7,.9,.7])
    xlabel('phase disparity (px)','FontSize',14);
    ylabel('Number of cells','FontSize',16)
text(0.76,0.9,strcat('$\sigma=\;$',num2str(std(phaseDifPx(good,1)),3)),...
                'Units','normalized','Interpreter','latex','FontSize',12);
    box off
subplot(3,1,3),
hist(posDifRel(good),[-n:n]*20/n);
    set(gca,'XLim',[-20,20])
    set(get(gca,'Children'),'FaceColor',[.7,.9,.7])
xlabel('position disparity (px)','FontSize',14);
text(0.76,0.9,strcat('$\sigma=\;$',num2str(std(posDifRel(good)),3)),...
```

```
                'Units','normalized','Interpreter','latex','FontSize',12);
    hold on
    [n2,xout] = hist(posDif(good,1),[-n:n]*20/n);
    bar(xout,n2,0.3,'w');
    text(0.74,0.78,strcat('$\sigma_1 =\; $',num2str(std(posDif(good,1)),3)),...
                'Units','normalized','Interpreter','latex','FontSize',12);
    box off

clear n n2 xout


%% phase vs. position
figure,
plot(phaseDifPx(good,1),posDif(good,1),'ok');
x = get(gca,'XLim');
set(gca,'YLim',x);
line(x,[0,0],'Color','k')
line([0,0],x,'Color','k')

ylabel('position Disparity (px)','FontSize',14);
xlabel('phase Disparity (px)','FontSize',14);
title('1st subunit','FontSize',16);
grid on;
[r,P] = corrcoef(phaseDifPx(good,1),posDif(good,1))
fr = @(x1) r(1,2)*x1;
r = fr(phaseDifPx(good,1));
sst = sum((posDif(good,1)-mean(posDif(good,1))).^2);
sse = sum((posDif(good,1)-r).^2);
R2r = 1 - sse*(length(good)-1) / (sst*(length(good) - 1))

clear r P x fr sst sse R2r


%% orientation vs. phase
figure,
plot(ori90(good,1)/pi,abs(phaseDif(good,1))/pi,'ok');
set(gca,'XTick',[0,1/12,2/12,1/4,4/12,5/12,1/2], ...
        'XTickLabelMode','manual', ...
        'XTickLabel',[0,15,30,45,60,75,90]);
set(gca,'YTick',[0,1/4,1/2,3/4,1], ...
```

```
        'YTickLabelMode','manual', ...
        'YTickLabel',[0,45,90,135,180])
xlabel('RF orientation (deg)','FontSize',14);
ylabel('|phase disparity| (deg PA)','FontSize',14);
title('1st subunit','FontSize',16);
grid on;

range = 20;     % ranges for analysis of variance in degree

% standard deviation of phases for cells with horizontal orientations
% in parenthesis: std+2*standard error obtained from bootstrap estimate
small = find(ori90(:,1)<=range/90*pi/2);
small = intersect(small,good);
F = var(abs(phaseDif(small,1)));
smStd = std(abs(phaseDif(small,1)))/pi*180;
smPh = abs(phaseDif(small,1))/pi*180;
smBootSE = std(bootstrp(1000,'std',smPh));
text(0.02,1.05,strcat('\sigma =',num2str(smStd,4),...
                      '(',num2str(smBootSE*2+smStd,4),')'),...
               'Units','normalized')

% standard deviation of phases for cells with vertical orientations
% in parenthesis: std-2*standard error obtained from bootstrap estimate
large = find(ori90(:,1)>=(90-range)/90*pi/2);
large = intersect(large,good);
F = F/var(abs(phaseDif(large,1)));
laStd = std(abs(phaseDif(large,1)))/pi*180;
laPh = abs(phaseDif(large,1))/pi*180;
laBootSE = std(bootstrp(1000,'std',laPh));
text(0.85,1.05,strcat('\sigma =',num2str(laStd,4),...
                      '(',num2str(laStd-laBootSE*2,4),')'),...
               'Units','normalized')

% F-test
F = 1/F             % because F should be >=1 for two-sided test
strcat('df=(',num2str(length(large)-1),',',num2str(length(small)-1),')')
P = 1-fcdf(F,length(large)-1,length(small)-1)
clear small large smStd laStd smPh laPh smBootSE laBootSE F P range

%% orientation vs phase and position
```

```
figure,
hold on
plot(ori90(good,1)/pi,abs(phaseDifPx(good,1)),'ok');
plot(ori90(good,1)/pi,abs(posDifRel(good)),'ok','MarkerFaceColor','k');
set(gca,'XTick',[0,1/12,2/12,1/4,4/12,5/12,1/2], ...
        'XTickLabelMode','manual', ...
        'XTickLabel',[0,15,30,45,60,75,90]);
set(gca,'YTick',[0,2,4,6,8,10,12], ...
        'YTickLabelMode','manual', ...
        'YTickLabel',[0,2/p*100,4/p*100,6/p*100,8/p*100,10/p*100,12/p*100])
xlabel('RF orientation (deg)','FontSize',14);
ylabel('|disparity| (% of patch)','FontSize',14);
title('1st subunit','FontSize',16);
grid on;
hold off;

range = 30;     % ranges for analysis of variance in degree

% standard deviation of phases for cells with horizontal orientations
% in parenthesis: std+2*standard error obtained from bootstrap estimate
small = find(ori90(:,1)<=range/90*pi/2);
small = intersect(small,good);
F = var(abs(phaseDifPx(small,1)));
smStd = std(abs(phaseDifPx(small,1)))/pi*180;
smPh = abs(phaseDifPx(small,1))/pi*180;
smBootSE = std(bootstrp(1000,'std',smPh));
text(0.02,1.05,strcat('\sigma =',num2str(smStd,4),...
                      '(',num2str(smBootSE*2+smStd,4),')'),...
              'Units','normalized')

% standard deviation of phases for cells with vertical orientations
% in parenthesis: std-2*standard error obtained from bootstrap estimate
large = find(ori90(:,1)>=(90-range)/90*pi/2);
large = intersect(large,good);
F = F/var(abs(phaseDifPx(large,1)));
laStd = std(abs(phaseDifPx(large,1)))/pi*180;
laPh = abs(phaseDifPx(large,1))/pi*180;
laBootSE = std(bootstrp(1000,'std',laPh));
text(0.85,1.05,strcat('\sigma =',num2str(laStd,4),...
                      '(',num2str(laStd-laBootSE*2,4),')'),...
```

```
              'Units','normalized')

% F-test
F = 1/F             % because F should be >=1 for two-sided test
strcat('df=(',num2str(length(large)-1),',',num2str(length(small)-1),')')
P = 1-fcdf(F,length(large)-1,length(small)-1)
clear small large smStd laStd smPh laPh smBootSE laBootSE F P range


%% frequency vs. phase
figure,
plot(f(good,1),abs(phaseDif(good,1))/pi,'ok');
set(gca, 'XScale','log')
set(gca,'YTick',[0,1/4,1/2,3/4,1], 'YTickLabelMode','manual', ...
        'YTickLabel',[0,45,90,135,180])
xlabel('RF spatial frequency (c/px)','FontSize',14);
ylabel('|phase disparity| (deg PA)','FontSize',14);
title('1st subunit','FontSize',16);
grid on;


%% frequency vs. phase and position
figure,
plot(f(good,1),abs(phaseDifPx(good,1)),'ok');
hold on
plot(f(good,1),abs(posDif(good,1)),'ok','MarkerFaceColor','k');
x = get(gca,'XLim');
y = get(gca,'YLim');
x = max(x(1),1/4/y(2)):0.001:x(2);
plot(x,1/4./x,'--','Color',[0,0.5,0]);
x = get(gca,'XLim');
x = max(x(1),1/2/y(2)):0.001:x(2);
plot(x,1/2./x,'-','Color',[0.7,0,0]);
xlabel('RF spatial frequency (c/px)','FontSize',14);
ylabel('|disparity| (px)','FontSize',14);
title('1st subunit','FontSize',16);
legend('phase','position','90° phase','180° phase');
grid on;
hold off;
```

```
clear x y


%% phase vs. phase disparity (subunits)
figure,
hold on;
plot(phaseDif(good,1),phaseDif(good,2),'+k');
x = get(gca,'XLim');
y = get(gca,'YLim');
% y = [x(:),y(:)];
set(gca,'XLim',[min(y(1),x(1)),max(y(2),x(2))], ...
        'YLim',[min(y(1),x(1)),max(y(2),x(2))]);
y = get(gca,'YLim');
plot(y,y,'-k');
xlabel('phase disparity 1st subunit (rad)','FontSize',14);
ylabel('phase disparity 2nd subunit (rad)','FontSize',14)
grid on;
hold off;

clear x y


%% frequeny vs. frequency (subunits)
figure,
hold on;
plot(f(:,1),f(:,2),'+k');
x = get(gca,'XLim');
y = get(gca,'YLim');
% y = [x(:),y(:)];
set(gca,'XLim',[min(y(1),x(1)),max(y(2),x(2))], ...
        'YLim',[min(y(1),x(1)),max(y(2),x(2))]);
y = get(gca,'YLim');
plot(y,y,'-k');
xlabel('spatial frequency 1st subunit (c/px)','FontSize',14);
ylabel('spatial frequency 2nd subunit (c/px)','FontSize',14)
grid on;
hold off;

clear x y
```

```
%% position vs. position disparity (subunits)
figure,
hold on;
plot(posDif(good,1),posDif(good,2),'+k');
x = get(gca,'XLim');
y = get(gca,'YLim');
% y = [x(:),y(:)];
set(gca,'XLim',[min(y(1),x(1)),max(y(2),x(2))], ...
        'YLim',[min(y(1),x(1)),max(y(2),x(2))]);
y = get(gca,'YLim');
plot(y,y,'-k');
xlabel('position disparity 1st subunit (px)','FontSize',14);
ylabel('position disparity 2nd subunit (px)','FontSize',14)
grid on;
hold off;

clear x y


%% orientation vs. orientation (subunits)
figure,
hold on;
plot(ori(:,1),ori(:,2),'+k');
x = get(gca,'XLim');
y = get(gca,'YLim');
% y = [x(:),y(:)];
set(gca,'XLim',[min(y(1),x(1)),max(y(2),x(2))], ...
        'YLim',[min(y(1),x(1)),max(y(2),x(2))]);
y = get(gca,'YLim');
plot(y,y,'-k');
xlabel('orientation 1st subunit (rad)','FontSize',14);
ylabel('orientation 2nd subunit (rad)','FontSize',14);
grid on;
hold off;

clear x y


%% envelope width vs. envelope width (SVD components)
```

```
figure,
hold on;
sub1l = fitsSVD(1:4:397,3);
sub2l = fitsSVD(3:4:399,3);
plot(sub1l,sub2l,'+k');
x = get(gca,'XLim');
y = get(gca,'YLim');
% y = [x(:),y(:)];
set(gca,'XLim',[min(y(1),x(1)),max(y(2),x(2))], ...
        'YLim',[min(y(1),x(1)),max(y(2),x(2))]);
y = get(gca,'YLim');
plot(y,y,'-k');
xlabel('envelope width 1st subunit (px)','FontSize',14);
ylabel('envelope width 2nd subunit (px)','FontSize',14);
grid on;
hold off;

[r,P] = corrcoef(abs(sub1l(find(sub1l<10 & sub2l<10))), ...
                 abs(sub2l(find(sub1l<10 & sub2l<10))))

clear x y sub1l sub2l r P


%% envelope centre vs. envelope centre (SVD components)
figure,
hold on;
sub1l = 1:4:397;
sub2l = 3:4:399;
sub1r = 2:4:398;
sub2r = 4:4:400;
plot(fitsSVD(sub1l,1),fitsSVD(sub2l,1),'+k');
plot(fitsSVD(sub1r,1),fitsSVD(sub2r,1),'ob');
x = get(gca,'XLim');
y = get(gca,'YLim');
% y = [x(:),y(:)];
set(gca,'XLim',[min(y(1),x(1)),max(y(2),x(2))], ...
        'YLim',[min(y(1),x(1)),max(y(2),x(2))]);
y = get(gca,'YLim');
plot(y,y,'-k');
xlabel('envelope centre 1st subunit (px)','FontSize',14);
```

```
ylabel('envelope centre 2nd subunit (px)','FontSize',14);
grid on;
hold off;

clear x y sub1l sub2l


%% phase difference of RFs (subunits)
figure,
hold on;
pdlrall = phaseDifLR(good,:);
[n(:,1),xout] = hist(abs(pdlrall(:,1))/pi,15);
[n(:,2),xout] = hist(abs(pdlrall(:,2))/pi,xout);
bar(xout,n,1,'stacked')
c = get(gca,'Children');
set(c(1),'FaceColor',[.7,.9,.7])
set(c(2),'FaceColor',[.4,.6,.4])
set(gca,'XTick',[0,0.5,1], 'XTickLabelMode','manual', ...
        'XTickLabel',[0,90,180],'FontSize',12);
xlabel('|1st - 2nd subunit phase| (deg PA)','FontSize',14)
legend('left','right'), legend('boxoff')
hold off;

clear pdlrall n xout c
```

showPrince.m

```
% script containing collection of diagrams to visualise RDS data analogous
% to Prince et al. (2002)
% it is not recommended to run this script in the command line since
% several figure windows will open simultaneously, use cell mode instead to
% run selected cells containing one diagram each
% cell "prolog" needs to be run once before diagrams can be constructed
% for final diagrams which also display data from Prince see directory figs

% version 1.0 {Sebastian Bitzer}


%% prolog
load('RDSFits1310i1000o0d20dD0.25.mat','parsRDS','R2RDS','confRDS',...
```

```
     'goodCellsRDS');
load('RDSAct1310i1000o0d20dD0.25.mat','DDI','dispActM','dispAct',...
     'Rmax','Rmin')


%% hist dominant monocular response / uncorrelated response

figure,
hold on;
[n,x] = hist(MUratio, 0.1:0.2:8);
bar(x,n,1,'FaceColor','none');
plot(ones(1,max(n)+6),(0:max(n)+5),'--k');
axis([0,8,0,max(n)+5]);
xlabel('dominant monocular response / uncorrelated response', ...
       'FontSize', 16);
ylabel('number of cells','FontSize', 16);
hold off;


%% DDI histogram
figure,
for i = 1:size(dispActM,2)
    [anovaP(i), tab] = anova1(squeeze(dispAct(:,:,i))',[]','off');
    Findex(i) = tab{2,4}/(tab{3,4}+tab{2,4});
end
hist(Findex), xlabel('F_{index}')
figure
sig = find(anovaP<0.01);
[n(:,1),xout] = hist(DDI(sig),15);
[n(:,2),xout] = hist(DDI(setdiff(1:100,sig)),xout);
bar(xout,n,1,'stacked')
c = get(gca,'Children');
set(c(1),'FaceColor',[.4,.6,.4])
set(c(2),'FaceColor',[.7,.9,.7])
legend('P<0.01','P>0.01'), legend('boxoff')

clear anovaP tab Findex i n sig xout c


%% DDI vs. mean firing
```

```
figure
hold on
plot(mean(squeeze(mean(dispAct))),DDI,'sk','MarkerFaceColor','k')
figure
plot(mean(squeeze(mean(dispAct))),(Rmax-Rmin)./(Rmax+Rmin),...
     'sk','MarkerFaceColor','k')
hold off

%% position vs. phase
figure('Position',[300,300,400,400]);

phase = parsRDS(goodCellsRDS,5);
phase = rem(phase,2*pi);
big   = find(abs(phase)>pi);
phase(big) = phase(big) - sign(phase(big))*2*pi;
pos = parsRDS(goodCellsRDS,2);

plot(pos, phase./pi, 's', ...
                'MarkerEdgeColor','k', ...
                'MarkerFaceColor','k', ...
                'MarkerSize',5);

xlabel('Gabor Mean Position (pixel)','FontSize',14);
ylabel('Gabor Phase (multiples of \pi)','FontSize',14);

big = ceil(max(abs(pos)));
axis([-big,big,-1.1,1.1]);
set(gca,'YTick',[-1,-0.5,0,0.5,1], 'YTickLabelMode','manual',...
        'YTickLabel',[-1,-0.5,0,0.5,1])
x = get(gca,'XLim');
y = get(gca,'YLim');
line(x,[0,0],'LineStyle',':','Color','k')
line([0,0],y,'LineStyle',':','Color','k')

all = length(phase);
{
strcat('TI:',num2str(length(find(phase>3/4*pi | phase<-3/4*pi))/all,2))
strcat('TE:',num2str(length(find(phase>-1/4*pi & phase<1/4*pi))/all,2))
strcat('NE:',num2str(length(find(phase>-3/4*pi & phase<-1/4*pi))/all,2))
strcat('FA:',num2str(length(find(phase>1/4*pi & phase<3/4*pi))/all,2))
```

```
}

clear big x y all pos phase

%% position vs. phase (px)
figure,

phase = parsRDS(goodCellsRDS,5);
phase = rem(phase,2*pi);
big   = find(abs(phase)>pi);
phase(big) = phase(big) - sign(phase(big))*2*pi;
phase = -phase/(2*pi)./parsRDS(goodCellsRDS,4);
pos = parsRDS(goodCellsRDS,2);
big = max(abs(phase))+0.5;

subplot(2,2,1),
    hist(pos);
    axis([-big,big,get(gca,'YLim')])
subplot(2,2,3),
    hold on
    plot(pos, phase,'.k');
    axis([-big,big,-big,big])
    plot([-big,big],[0,0],':k')
    plot([0,0],[-big,big],':k')
    set(gca,'DataAspectRatio',[1,1,1],'PlotBoxAspectRatio',[1,1,1])
    xlabel('position shift (px)','FontSize',14);
    ylabel('phase shift (px)','FontSize',14);
    hold off;
subplot(2,2,4),
    hist(phase);
    axis([-big,big,get(gca,'YLim')])
    set(gca,'XDir','reverse')
    view(90,90);

'corrcoef(pos,phase):'
[r,P] = corrcoef(pos,phase)

clear big r P phase pos

%% parameter comparison
```

```
figure,
subplot(4,4,1),
    hist(parsRDS(goodCellsRDS,2));
subplot(4,4,5),
    plot(parsRDS(goodCellsRDS,2), parsRDS(goodCellsRDS,5)./pi, 'o', ...
                'MarkerEdgeColor','k', ...
                'MarkerFaceColor','w', ...
                'MarkerSize',5);
subplot(4,4,6),
    hist(parsRDS(goodCellsRDS,5));
subplot(4,4,9),
    plot(parsRDS(goodCellsRDS,2), abs(parsRDS(goodCellsRDS,3)), 'o', ...
                'MarkerEdgeColor','k', ...
                'MarkerFaceColor','w', ...
                'MarkerSize',5);
subplot(4,4,10),
    plot(parsRDS(goodCellsRDS,5)./pi, abs(parsRDS(goodCellsRDS,3)), 'o', ...
                'MarkerEdgeColor','k', ...
                'MarkerFaceColor','w', ...
                'MarkerSize',5);
subplot(4,4,11),
    hist(abs(parsRDS(goodCellsRDS,3)));
subplot(4,4,13),
    plot(parsRDS(goodCellsRDS,2), parsRDS(goodCellsRDS,4), 'o', ...
                'MarkerEdgeColor','k', ...
                'MarkerFaceColor','w', ...
                'MarkerSize',5);
    set(gca,'YScale','log');
subplot(4,4,14),
    plot(parsRDS(goodCellsRDS,5)./pi, parsRDS(goodCellsRDS,4), 'o', ...
                'MarkerEdgeColor','k', ...
                'MarkerFaceColor','w', ...
                'MarkerSize',5);
    set(gca,'YScale','log');
subplot(4,4,15),
    plot(abs(parsRDS(goodCellsRDS,3)), parsRDS(goodCellsRDS,4), 'o', ...
                'MarkerEdgeColor','k', ...
                'MarkerFaceColor','w', ...
                'MarkerSize',5);
    set(gca,'YScale','log');
```

```
subplot(4,4,16),
    hist(parsRDS(goodCellsRDS,4));
    set(gca,'XScale','log');


%% phase histogram in polar plot
figure,
[trose,r] = rose(parsRDS(goodCellsRDS,5),20);
myPolar(trose,r);
set(get(gca,'Children'),'LineWidth',3,'Color',[0.08,0.17,0.55])

clear trose r


%% frequency vs. maximum interaction position
figure,

x = -20:.1:20;
for i=1:100
    y = gabor1(parsRDS(i,:),x);
    maxi(i) = x(find(max(y)==y));
end

semilogx(parsRDS(goodCellsRDS,4),abs(maxi(goodCellsRDS)),...
         'sk','MarkerFaceColor',[.7,.9,.7])
hold on
x = get(gca,'XLim');
y = get(gca,'YLim');
x = max(x(1),1/4/y(2)):0.001:x(2);
plot(x,1/4./x,'--','Color',[0,0.5,0]);
x = get(gca,'XLim');
x = max(x(1),1/2/y(2)):0.001:x(2);
plot(x,1/2./x,'-','Color',[0.7,0,0]);

clear x y maxi


%% orientation vs. phase in PA
<see showRFAnalysis.m>
```

```
%% orientation vs. phase and pos
<see showRFAnalysis.m>
```

showAllRDS.m

```
% showAllRDS(cells,RDSActFile,FitFile)
%
% displays fitted tuning curves of cells given in "cells" together with
% data points, R-square, confidence intervals, DDI and activities to
% monocular and uncorrelated stimuli
% loads *ActM, DDI, parsRDS, confRDS and R2RDS from RDSActFile and FitFile,
% respectively; if these are not given, standard ones are chosen

% version 1.0 {Sebastian Bitzer}
function showAllRDS(cells,RDSActFile,FitFile)

if (nargin < 3)
    FitFile = 'RDSFits1310i1000o0d20dD0.25.mat';
    if (nargin < 2)
        RDSActFile = 'RDSAct1310i1000o0d20dD0.25.mat';
    end
end

load(RDSActFile,'*ActM','DDI');
load(FitFile,'parsRDS','confRDS','R2RDS');

dataX = -20:20;

figure('Position',[300,300,700,400]);

for i=1:length(cells)

    clf
    hold on;

    plot(dataX, dispActM(:,cells(i))', '+b',...
         dataX, repmat(monoLActM(cells(i)),1,41),'r',...
         dataX, repmat(monoRActM(cells(i)),1,41),'g',...
         dataX, repmat(monoLActM(cells(i))+monoRActM(cells(i)),1,41),'m',...
         dataX, repmat(uncorrActM(cells(i)),1,41),'k');
```

```
x = min(dataX):.1:max(dataX);
plot(x,gabor1(parsRDS(cells(i),:), x),'Color',[0,0.5,0])

maxY = monoLActM(cells(i))+monoRActM(cells(i));
minY = min([monoLActM(cells(i)),monoRActM(cells(i)), ...
            dispActM(:,cells(i))']);
axis([dataX(1)-3,dataX(end)+40,minY-30,maxY+30]);
len = length(dataX);
set(gca,'XTick', [dataX(1),dataX(ceil(len/4)),dataX(ceil(len/2)), ...
                  dataX(ceil(3*len/4)),dataX(end)])

d0 = parsRDS(cells(i),2);
plot([d0,d0],[minY,maxY-30],'--k');

text(0.55,0.92,{['cell: ',num2str(cells(i))]},'FontSize',16, ...
              'VerticalAlignment','top','Units','normalized')

info = cell(1,11);

info(1) = {['R-square = ',num2str(R2RDS(cells(i)),2)]};
info(2) = {' '};
info(3) = {'params (\pm confidence):'};
info(4) = {['       A: ',num2str(parsRDS(cells(i),1),4), ...
            ' \pm',num2str(confRDS(cells(i),1),4)]};
info(5) = {['       d_0: ',num2str(parsRDS(cells(i),2),3), ...
            ' \pm',num2str(confRDS(cells(i),2),3)]};
info(6) = {['       \sigma: ',num2str(parsRDS(cells(i),3),3), ...
            ' \pm',num2str(confRDS(cells(i),3),3)]};
info(7) = {['       f: ',num2str(parsRDS(cells(i),4),3)]};
info(8) = {['       \phi: ',num2str(parsRDS(cells(i),5),3), ...
            ' \pm',num2str(confRDS(cells(i),4),3)]};
info(9) = {['       R_m: ',num2str(parsRDS(cells(i),6),4), ...
            ' \pm',num2str(confRDS(cells(i),5),4)]};
info(10) = {' '};
info(11) = {['DDI = ',num2str(DDI(cells(i)),4)]};


text(0.55,0.82,info,'FontSize',14, 'VerticalAlignment','top', ...
                   'Units','normalized')
```

```
    legend('disp','left','right','l+r','uncorr','fit','d0', ...
           'Location','BestOutside')
    legend('boxoff')

    hold off;

    waitforbuttonpress
end
```