LETTER Communicated by Richard Zemel

Population Computation of Vectorial Transformations

Pierre Baraduc
Pierre.Baraduc@snv.jussieu.fr
Emmanuel Guigon
guigon@ccr.jussieu.fr
INSERM U483, Université Pierre et Marie Curie
75005 Paris, France

Many neurons of the central nervous system are broadly tuned to some sensory or motor variables. This property allows one to assign to each neuron a preferred attribute (PA). The width of tuning curves and the distribution of PAs in a population of neurons tuned to a given variable define the collective behavior of the population. In this article, we study the relationship of the nature of the tuning curves, the distribution of PAs, and computational properties of linear neuronal populations. We show that noise-resistant distributed linear algebraic processing and learning can be implemented by a population of cosine tuned neurons assuming a nonuniform but regular distribution of PAs. We extend these results analytically to the noncosine tuning and uniform distribution case and show with a numerical simulation that the results remain valid for a nonuniform regular distribution of PAs for broad noncosine tuning curves. These observations provide a theoretical basis for modeling general nonlinear sensorimotor transformations as sets of local linearized representations.

1 Introduction

Many problems of the nervous system can be cast in terms of linear algebraic calculus. For instance, changing the frame of reference of a vector is an elementary linear operation in the process of coordinate transformations for posture and movement (Soechting & Flanders, 1992; Redding & Wallace, 1997). More generally, coordinate transformations are nonlinear operations that can be linearized locally (Jacobian) and become a simpler linear problem (see the discussion in Bullock, Grossberg, & Guenther, 1993). Vectorial calculus is also explicitly or implicitly used in models of sensorimotor transformations for reaching and navigation (Grossberg & Kuperstein, 1989; Burnod, Grandguillaume, Otto, Ferraina, Johnson, & Caminiti, 1992; Touretzky, Redish, & Wan, 1993; Redish & Touretzky, 1994; Georgopoulos, 1996).

Although linear processing is only a rough approximation of generally nonlinear computations in the nervous system, it is worth studying for at least two reasons (Baldi & Hornik, 1995): (1) it displays an unexpected wealth of behaviors, and (2) a thorough understanding of the linear regime is necessary to tackle nonlinear cases for which general properties are difficult to derive analytically. The problem of the neural representation of vectorial calculus can be expressed in terms of two spaces: a low-dimensional space corresponding to the *physical* space of the task and a high-dimensional space defined by activities in a population of neurons (termed *neuronal* space) (Hinton, 1992; Zemel & Hinton, 1995). In this framework, a desired operation in the physical space (*vectorial* transformation) is translated into a corresponding operation in the neuronal space, the result of which can be taken back into the original space for interpretation. The goal of this article is to describe a set of mathematical properties of neural information processing that guarantee appropriate calculation of vectorial transformations by populations of neurons (i.e., that computations in the physical and neuronal spaces are equivalent).

An appropriate solution relies on three mechanisms: a decoding-encoding method that translates information between the spaces, a mechanism that favors the stability of operations in the neuronal space, and an unsupervised learning algorithm that builds neuronal representations of physical objects. We will show that these mechanisms are closely related to common properties of neural computation and learning: the distribution of tuning selectivities in the population of neurons and the width of the tuning curves, the pattern of lateral connections between the neurons, and the distribution of input-output patterns used to build synaptic interactions between neuronal populations.

In this article, we present a theory that unifies these three mechanisms and properties (generally considered separately; Mussa-Ivaldi, 1988; Sanger, 1994; but see Zhang, 1996; Pouget, Zhang, Deneve, & Latham, 1998) into a unique mathematical framework based on the neuronal population vector (PV; Georgopoulos, Kettner, & Schwartz, 1988) in order to explain how neuronal populations can perform vectorial calculus. In contrast to our extensive knowledge of the representation of information by populations of tuned neurons, little attention has been devoted to the learning processes in these populations. Here we show how Hebbian and unsupervised error-correcting rules can be used in association with lateral connections to allow the learning of linear maps on the basis of input-output correlations provided by the environment. In this context, we reveal a trade-off between the width of the tuning curves and the uniformity of the distribution of preferred directions. Finally, a statistical approach validates our hypotheses in realistic networks of a few thousand noisy neurons.

A particular application of this theoretical framework is the computing of distributed representations of transpose or inverse Jacobian matrices which play a central role in kinematic and dynamic transformations (Hinton, 1984;

Mussa-Ivaldi, Morasso, & Zaccaria, 1988; Crowe, Porrill, & Prescott, 1998). Recent results highlight the relevance of this theory to the understanding of the elaboration of directional motor commands for reaching movements (Baraduc, Guigon, & Burnod, 1999).

2 Notations and Definition

In the following text, we consider a population of N neurons. We note $\mathcal{E} = \mathbb{R}^N$ the neuronal space and $\mathbb{E} = \mathbb{R}^D$ (typically $D = 2, 3$) the physical space. Lowercase letters (e.g., x) are vectors of the neuronal space. Uppercase letters (e.g., X) are vectors of the physical space. Matrices are indicated by uppercase bold letters roman for $\mathbb{E}$ (e.g., $\mathbf{M}$), calligraphic for $\mathcal{E}$ (e.g., $\mathcal{M}$), and italic for $D \times N$ matrices (e.g., $\boldsymbol{E}$). A dot $(\cdot)$ stands for the dot product in $\mathcal{E}$ or $\mathbb{E}$.

Each neuron j is tuned to a D-dimensional vectorial parameter, that is, it has a preferred attribute in $\mathbb{E}$, which is noted E_j, and its firing rate is given by

$$x_j = f_j(X \cdot E_j, \beta_j), \tag{2.1}$$

where f_j is the tuning function of the neuron, X a unit vector of the physical space (Georgopoulos, Schwartz, & Kettner, 1986), and β_j a vector of parameters. The assumption is made that the distributions of these parameters and the distribution of PAs are independent (Georgopoulos et al., 1988). In the particular case of cosine tuning, the firing rate of neuron j is

$$x_j = X \cdot E_j + b_j, \tag{2.2}$$

where b_j is the mean firing rate of the neuron (Georgopoulos et al., 1986). We note $\boldsymbol{E}$ the $D \times N$ matrix of vectors E_j.

The PAs are considered either as a set of fixed vectors or as realizations of a random variable with a given distribution P_E (in this latter case, index i is removed). The mean is denoted by $\langle\rangle$ and the variance by V.

3 Cosine Tuning and Vectorial Processing in Neural Networks

As a simple case of distributed computation, in this section, we derive conditions that are sufficient to represent and learn vectorial transformations between populations of cosine-tuned neurons. The case of other tuning functions will be treated later (section 4) in the light of this approach.

3.1 Encoding-Decoding Method: Distributed Representation of Vectors. Here we address the representation of vectors by distributed activity patterns in populations of cosine-tuned neurons. We show that a condition on the distribution of preferred attributes is sufficient to faithfully recover information from the activity of the population. This condition is mathematically exact for populations of infinite size, but still leads to accurate representations for populations of biologically reasonable size (e.g., $> 10^3$).

The firing frequency of the population in response to the presentation of a vector X of the physical space is $x = \boldsymbol{E}^T X + b$, where x and b are vectors in $\mathcal{E}$ (equation 2.2 written in matrix notation). Based on some hypotheses on $\boldsymbol{E}$ and b, the vector X can be decoded by computing a population vector (Georgopoulos et al., 1988; Mussa-Ivaldi, 1988; Sanger, 1994). The population vector can be defined by

$$X^* = \frac{1}{N}\sum_i (x_i - b_i)E_i = \frac{1}{N}\boldsymbol{E}(x - b).$$

A perfect reconstruction ($X^* \propto X$) is obtained if the PAs are such that (Mussa-Ivaldi, 1988; Sanger, 1994)

$$\boldsymbol{E}\boldsymbol{E}^T \propto \mathbf{I}_D, \tag{3.1}$$

where $\mathbf{I}_D$ is the $D \times D$ identity matrix.

In a population of neurons, the offset b could be deduced from the activity of the network over a sufficiently long period of time and subtracted via an inhibition mechanism (e.g., global inhibition if all neurons have the same mean firing rate). However, we will consider here the general case:

$$X^* = \mathbf{Q}X + \frac{1}{N}\boldsymbol{E}b,$$

where $\mathbf{Q} = \frac{1}{N}\boldsymbol{E}\boldsymbol{E}^T$. We make the assumption that the components of the PAs have zero mean, are uncorrelated, and have equal variance σ_E^2 (regularity condition). From our hypothesis, mean firing rates b are independent of the distribution of PAs. Then $\mathbf{Q}$ converges in probability toward $\sigma_E^2\mathbf{I}_D$ (see section A.1). Using similar arguments, we can demonstrate that $\frac{1}{N}\boldsymbol{E}b$ converges in probability toward 0.

In the following, we call a family of tuning properties $\{E_i, b_i\}$ that satisfies the regularity condition *regular basis*. We use the term *basis* to indicate that a regular family can be used as a basis. However, it is not a basis in a mathematical sense. If $X \in \mathbb{E}$, $x = \boldsymbol{E}^T X + b$ is called the *distributed representation* of X, or simply a *population code*.

3.1.1 Finite N. The preceding equalities hold only at the limit $N \to +\infty$. To ascertain if the proposed computational scheme has any relevance to biology, we need to quantify the distortions introduced when populations of finite size are used.

Without loss of generality, we can suppose the input to be $X = (1, 0, \ldots, 0)$. The variance of the decoded output, normalized by $1/\sigma_E^2$, is in this case

$$\mathrm{V}\left(\frac{\boldsymbol{E}x}{\sigma_E^2}\right) = \mathrm{V}\left(\frac{1}{N^2\sigma_E^4}\boldsymbol{E}\boldsymbol{E}^T X\right) = (\delta^2/\sigma_E^4, 1, \ldots, 1)/N,$$

where $\delta^2 = \mathrm{V}(E_{i1}^2)$.

Is this variance small enough in practice? For a uniform distribution of PAs on a three-dimensional sphere, $\delta^2/\sigma_E^4 = 13/5$, which results in an angular variance of less than 0.55 degree for $N = 1000$. For a distribution of PAs (of the same norm) clustered along the axes, $\delta^2/\sigma_E^4 = 2$—hence, an angular variance of less than 0.48 degree if $N = 1000$. This suggests that this encoding scheme is reasonably accurate with small populations of neurons.

The regularity condition thus guarantees that encoded information can be recovered from actual firing rates with an arbitrary precision in a sufficiently large population of neurons. The regularity condition includes a zero-mean assumption for PA components, which is not used in Sanger (1994). Any departure from this requirement translates the output vectors by a constant amount, which needs to be small in practice. The zero-mean assumption is not a major constraint, since most experimentally measured distributions of selectivity are roughly symmetrical (see section 6). In this sense, our definition of regularity is more general than the previous ones (Mussa-Ivaldi, 1988; Sanger, 1994; Zhang, Ginzburg, McNaughton, & Sejnowski, 1998), as it allows a proper probabilistic treatment when mean firing rate is nonzero.

3.2 Distributed Representation of Linear Transformations. The preceding section has shown how a correspondence can be established between vectors in external space and population activity. In this section, we extend this correspondence to linear mappings and define the notion of input and output preferred attributes.

Consider a linear map from $\mathbb{E}$ to $\mathbb{F}$, which are real physical vectorial spaces. Let $\mathbf{M}$ be its matrix on the canonical bases and $\boldsymbol{E}$, $\boldsymbol{F}$ be regular bases in $\mathbb{E}$ (N_E neurons) and $\mathbb{F}$ (N_F neurons), respectively. We define

$$\mathcal{M} = \frac{1}{N_E N_F} \boldsymbol{F}^T \mathbf{M} \boldsymbol{E} \tag{3.2}$$

as the matrix of the distributed linear map.

In the limit $N_E, N_F \to +\infty$, and assuming that $\sigma_E = \sigma_F = 1$, we have $\mathbf{Q}_E = \mathbf{Q}_F = \mathbf{I}_D$. Then $\mathcal{M}$ operates on the distributed representations as $\mathbf{M}$ does in the original space. Let x be the distributed representation of a vector $X \in \mathbb{E}$ (i.e., $x = \boldsymbol{E}^T X$). Taking $Y = \mathbf{M}X$, we have $\mathcal{M}x = \boldsymbol{F}^T \mathbf{M} \boldsymbol{E} \boldsymbol{E}^T X = \boldsymbol{F}^T (\mathbf{M}X) = \boldsymbol{F}^T Y$. Thus, $y = \mathcal{M}x$ is the distributed representation of Y.

If we assume that the vectorial input (resp. output) is represented by the collective activity of a population of neurons x_j (resp. y_i), and that a weight matrix $\mathcal{M}$ links the input and output layers, then the network realizes the transformation $\mathbf{M}$ on the distributed vectors. It is immediate that $\boldsymbol{F} \mathcal{M} \boldsymbol{E}^T = \mathbf{M}$. Thus, the distributed map can be read using the classical population vector analysis.

3.2.1 Finite N_E and N_F. As in the preceding section, it must be checked whether this distributed computation is still precise enough in the case of finite populations. To answer this question while keeping the derivations simple, we assume $N_E = N_F = N$, $\sigma_E = \sigma_F = \sigma$, and take the identity mapping for the transformation $\mathbf{M}$.

The variance of the (normalized) decoded output Y/σ^2 writes in this case:

$$\begin{aligned}
\mathrm{V}\left(\frac{\boldsymbol{F}\mathcal{I}x}{\sigma^2}\right) &= \mathrm{V}\left(\frac{1}{N^2\sigma^4}\boldsymbol{F}\boldsymbol{F}^T\boldsymbol{E}\boldsymbol{E}^T X\right) \\
&= \frac{1}{\sigma^8}\left[\mathrm{V}(\mathbf{Q}_F)\langle\mathbf{Q}_E\rangle_2 + \langle\mathbf{Q}_F\rangle_2\mathrm{V}(\mathbf{Q}_E) + \mathrm{V}(\mathbf{Q}_F)\mathrm{V}(\mathbf{Q}_E)\right] X_2 \\
&= \left[2\left\{\frac{1}{N}(\mathbb{I}_{D,D} - \mathbf{I}_D) + \frac{\delta^2}{N\sigma^4}\mathbf{I}_D\right\} + \frac{1}{N}\mathbb{I}_{D,D} + \cdots\right] X_2,
\end{aligned}$$

where $\mathbb{I}_{m,n}$ is an $m \times n$ matrix of ones and the ellipsis stands for terms dominated by $1/N^2$. Here the notation $\mathbf{Q}_2$ means the matrix of ij-component $\mathbf{Q}_{ij}^2$.

For $D = 3$ and $N = 1000$, in the case of a uniform distribution, the preceding equation translates into an angular variance of 0.84 degree; in the clustered case, the variance is 0.74 degree. Our scheme of distributed computation is thus viable with small populations of neurons.

Consequently, in the following sections, derivations will be made for infinite populations with $\boldsymbol{E}\boldsymbol{E}^T = \mathbf{I}_D$, which allows us to write equalities instead of proportionalities. We will thereby ignore the $\sigma^2 N$ term, except in the study of the effect of noise (see section 3.3.2). We will also assume that $b = 0$, which makes proofs more straightforward. The general case is considered in section A.3.

3.2.2 Selectivities of Output Units. In a network that computes $y = \mathcal{M}\,x$, how can one characterize the behavior of an output unit that fires with y_i?

This output unit i can be described by its intrinsic PA F_i in the output space $\mathbb{F}$. However, this vector is independent of the mapping that occurs between the input and output spaces, and thus does not fully define the role of the unit. In fact, two vectors can be associated with the output unit i. The first is the vector of $\mathbb{E}$ for which the unit is most active (input PA). Since the unit i fires with input X as $F_i^T\mathbf{M}X$, it is cosine tuned to the input, and its input PA is the column vector $\mathbf{M}^T F_i$.

The second vector (output PA) is $\mathbf{M}^\dagger F_i$, where $\mathbf{M}^\dagger$ is the Moore-Penrose inverse of $\mathbf{M}$. In the case where the output layer is considered as a motor layer whose effects can be measured in the input space through $\mathbf{M}^\dagger$, the output PA can be interpreted in an intuitive way. Indeed, the effect in sensory space of the isolated stimulation of the unit i is precisely the vector $\mathbf{M}^\dagger F_i$ of $\mathbb{F}$. Thus, the output PA corresponds to projective properties of the cell while input PA is related to receptive properties. Note that in general, the input and output PAs of a unit do not coincide (Zhang et al., 1998).

3.2.3 Weight and Activity Profiles. The distributed representation $\mathcal{M}$ has interesting structural properties. The transpose of the ith row of $\mathcal{M}$ is $(F_i^T \mathbf{M} \boldsymbol{E})^T = \boldsymbol{E}^T(\mathbf{M}^T F_i) \in \text{Im } \boldsymbol{E}^T$. In the same way, the jth column of $\mathcal{M}$ is $\boldsymbol{F}^T(\mathbf{M} E_j^T) \in \text{Im } \boldsymbol{F}^T$. Thus, the profile of the weight rows (resp. columns) is identical to the profile of the input (resp. output) activities.

Later we will consider the case where the entries of the matrix $\mathcal{M}$ are activities rather than static weights. We show below that "cosine" lateral connections between rows ($\boldsymbol{E}^T\boldsymbol{E}$) and columns ($\boldsymbol{F}^T\boldsymbol{F}$) stabilize population codes in $\mathcal{E}$ and $\mathcal{F}$, respectively. Thus, lateral connections can help to build an exact matrix of activities from an underspecified initial state.

3.3 Neuronal Noise and Stabilization of Representations. Noise has a strong impact on population coding (Salinas & Abbott, 1994; Abbott & Dayan, 1999). Therefore, it is important to understand how noise affects the reliability of our computational scheme. We will consider here two forms of noise: additive gaussian and Poisson noise.

3.3.1 Additive Gaussian Noise. Assume that a gaussian noise η is added to the population code x. How does this noise affect the encoding-decoding scheme—that is, how large is the variance of the decoded quantity? If η is independently distributed, we can show that the variance of the extra term due to the noise ($\boldsymbol{E}\eta/N$) is proportional to $1/N$ (see section A.2). Conversely, if the additive noise is correlated among neurons, as seems to be the case in experimental preparations (Gawne & Richmond, 1993; Zohary, Shadlen, & Newsome, 1994), it is easy to demonstrate that

$$\text{V}(\boldsymbol{E}\eta/N) = \frac{(1-c)\sigma_\eta^2\sigma_E^2}{N}, \tag{3.3}$$

where σ_η^2 is the variance and c the correlation coefficient of the noise. Thus, for this correlated noise as for the uncorrelated one, the variance of the encoded quantity decreases with a $1/N$ factor. Besides, the decoding error decreases as a function of c, as does the minimum unbiased decoding error (Abbott & Dayan, 1999). In fact the correlations act to decrease the total entropy of the system. The $1/N$ reduction of variance demonstrated for additive gaussian noise no longer holds with multiplicative noise; in such a case, an active (nonlinear) mechanism of noise control may be needed.

3.3.2 Poisson Noise. In the case of an uncorrelated Poisson noise, $V(\eta_i) = x_i$. It is straightforward to show that the variance of the noise term is inferior to $x_{\max}/N$, where $x_{\max}$ is the highest firing rate in the population (see section A.2). Thus, as for the gaussian noise, the variance decreases linearly with the number of neurons. Correlations in the noise alter this behavior, and the variance becomes dominated by a term independent of N. This term

can be computed for a few special cases of PA distribution; for example, we have

$$V(\boldsymbol{E}\eta_i/N) \leq 0.035\, c\, x_{\max} \tag{3.4}$$

for a uniform 3D distribution and

$$V(\boldsymbol{E}\eta_i/N) \leq 0.22\, c\, x_{\max} \tag{3.5}$$

for PAs clustered along the 3D axes (see section A.2). A reduction of the variance in the correlated case is thus obtained through the distributed coding, even if scaling N does not result in any additional benefit. It can also be noted that uniform distributions of PAs seem more advantageous as far as noise issues are concerned.

To sum up, for the two types of noise treated here, the variability in the decoded quantity is inferior to the variability affecting individual neurons. For gaussian or uncorrelated Poisson noise, using large populations of cells limits even more the noise in the decoded vectors, as noise amplitude depends on $1/\sqrt{N}$. This is not the case with correlated Poisson noise, and more powerful nonlinear methods could be employed (see, e.g., Zhang, 1996; Pouget et al., 1998).

3.3.3 Stabilizing Distributed Representations. The reduction of the noise in the decoded vector shown in the preceding sections can inspire ways to limit the noise inside a population. We show here that filtering the population activity through the matrix $\mathcal{W}_{\mathcal{E}} = \boldsymbol{E}^T\boldsymbol{E}/N$ has this desirable effect.

Before proving this fact, we first note that $\mathcal{W}_{\mathcal{E}}$ is the distributed representation of $\mathbf{I}_D$ in $\mathbb{E}$ (see equation 3.2). Matrix $\mathcal{W}_{\mathcal{E}}$ is a projection of $\mathcal{E}$ (Strang, 1988). If we note $\mathcal{E}_p$ the image of $\mathcal{E}$ by $\mathcal{W}_{\mathcal{E}}$, then $\mathcal{E}_p$ is a D-dimensional subspace of $\mathcal{E}$. Elements of $\mathcal{E}_p$ are population codes since they can be written $\boldsymbol{E}^T(\boldsymbol{E}x')$, $x' \in \mathcal{E}$. In fact, the operation of $\mathcal{W}_{\mathcal{E}}$ is a decoding-reencoding process. As the variance of the decoded vector coordinates is inferior to the neuronal variance (preceding sections), we can expect from $\mathcal{W}_{\mathcal{E}}$ good properties regarding noise control.

To demonstrate them, we write $\mathcal{W}_{\mathcal{E}}(x+\eta) = \mathcal{W}_{\mathcal{E}}x + \boldsymbol{E}^T\boldsymbol{E}\eta/N$. The term $\mathcal{W}_{\mathcal{E}}x$ is in general different from x (except if $x \in \mathcal{E}_p$), but it preserves part of the information on x, since the population vectors of x and $\mathcal{W}_{\mathcal{E}}x$ are the same. The variance of $\mathcal{W}_{\mathcal{E}}\eta$ is the first diagonal of $(\boldsymbol{E}^T\boldsymbol{E}\mathcal{Q}\boldsymbol{E}^T\boldsymbol{E})/N^2$, where $\mathcal{Q}$ is the correlation matrix of the noise.

Building on the results of the previous sections, it is easy to demonstrate that equations similar to equations 3.3, 3.4, and 3.5 apply. For additive gaussian noise, we find that

$$\mathrm{V}(\mathcal{W}_{\mathcal{E}}\eta) = \frac{(1-c)\sigma_\eta^2\sigma_E^2}{N}\,\mathbb{I}_N.$$

Thus, the effect of $\mathcal{W}_{\mathcal{E}}$ is to limit gaussian noise in the population. For Poisson noise, the formulas of section 3.3.2 generalize in the same way, leading to a decrease in the variance of the neuronal activity that is proportional to $1/N$ for uncorrelated noise and independent of N in the correlated case. Moreover, even if $\langle \eta \rangle \neq 0$, for independent noise, we get $\langle \mathcal{W}_{\mathcal{E}} \eta \rangle = 0$ in the limit $N \to +\infty$. This property, due to the fact that $\mathcal{W}_{\mathcal{E}}$ has balanced weights, can be used to sort out the relevant information from a superposition of uncorrelated codes.

The matrix $\mathcal{W}_{\mathcal{E}}$ can be viewed as a weight matrix, either of feedforward connections between two populations of N_E neurons or of lateral interactions inside a population, and extracts the population code of any input pattern in a single step. However, if $\mathcal{W}_{\mathcal{E}}$ slightly deviates from the definition, it is no longer a projection, and iterations of $\mathcal{W}_{\mathcal{E}}$ are likely to diverge or fade. A simple way to prevent divergence is to use a saturating nonlinearity (e.g., sigmoid). A more realistic solution is to adjust the shape of a nonsaturating nonlinearity to guarantee a stable behavior (Yang & Dillon, 1994; Zhang, 1996). In particular, an appropriate scaling of the gain of the neurons (maximum of the derivative of the nonlinearity) to the largest eigenvalue of $\mathcal{W}_{\mathcal{E}}$ leads to the existence of a Lyapunov function for continuous network dynamics.

If the distribution of PAs is uniform, $\mathcal{W}_{\mathcal{E}}$ is a circulant matrix (Davis, 1979). Iterations of a circulant matrix can extract the first Fourier component of the input, provided the first Fourier coefficient of the matrix is greater than 1 and all other coefficients strictly less than 1 (Pouget et al., 1998). Here, $\mathcal{W}_{\mathcal{E}}$ corresponds to the special case where the first Fourier coefficient is 1 and all others are zero.

We could as well consider $\mathcal{W}_{\mathcal{F}} = \boldsymbol{F}^T \boldsymbol{F}$ as a matrix of recurrent connections on the output layer to suppress noise on this layer.

3.4 Learning Distributed Representations of Linear Transformations. Up to now, we have demonstrated that a correspondence between external and neural spaces could be established and maintained. This correspondence permits a faithful neural representation of external vectors and mappings. It remains now to be shown whether a distributed representation of a linear mapping can be built from examples using a local synaptic learning rule. We prove below that it is indeed possible, provided the training examples satisfy a part of the regularity condition.

3.4.1 Hebbian Learning of Linear Mappings. Let $\mathbf{M}$ be a linear transformation and $(X^\nu, Y^\nu = \mathbf{M}X^\nu)$ be training pairs in $\mathbb{E} \times \mathbb{F}, \nu = 1, \ldots, N_{\text{ex}}$. Hebbian learning writes

$$\mathcal{M}^*_{ij} \propto \sum_{\nu=1}^{N_{\text{ex}}} y_i^\nu x_j^\nu,$$

where (x^ν, y^ν) are the distributed representations of the training samples. Then,

$$\mathcal{M}^* \propto \sum_\nu \boldsymbol{F}^T Y^\nu (X^\nu)^T \boldsymbol{E} \propto \sum_\nu \boldsymbol{F}^T \mathbf{M} X^\nu (X^\nu)^T \boldsymbol{E} \propto \boldsymbol{F}^T \mathbf{M} \boldsymbol{E}$$

if the training examples satisfy

$$\sum_\nu X^\nu (X^\nu)^T \propto \mathbf{I}_{\dim \mathbb{E}}. \tag{3.6}$$

In this case, the matrix $\mathcal{M}^*$ is proportional to the required matrix. Thus, any distributed linear transformation can be learned modulo a scaling factor by Hebbian associations between input and output activities if the components of the training inputs are uncorrelated and have equal variances (zero mean is not required). In practice, to control for the weight divergence implied by standard Hebbian procedures, the following stochastic rule is used:

$$\Delta \mathcal{M}^*_{ij} \propto (y^\nu_i x^\nu_j - \mathcal{M}^*_{ij}). \tag{3.7}$$

3.4.2 Nonregular Distribution of Examples and Tuning Properties. Regularity may be a restricting condition in some situations. Distributions of PAs are not necessarily regular, or it may not be possible to guarantee that training examples are regularly distributed. This latter case can occur when learning a (generally ill-defined) linear mapping from samples of its inverse (Kuperstein, 1988; Burnod et al., 1992; Bullock et al., 1993). We denote $\mathbf{M}_1$ the inverse mapping. Training consists of choosing an output pattern y^ν, calculating the corresponding input pattern $x^\nu = \boldsymbol{E}^T \mathbf{M}_1 \boldsymbol{F} y^\nu$, and then using (x^ν, y^ν) as examples. If the y^ν are regular, Hebbian learning leads to the representation of $\mathbf{M}_1^T$ but not $\mathbf{M}_1^{-1}$ (or a generalized inverse if $\mathbf{M}_1$ is singular or noninjective).

An appropriate solution to this problem is obtained if the learning takes place only for the $\mathcal{M}_{ij}$ receiving maximal x input, that is, $\mathcal{M}_{ij_{\max}(\nu)}$, where $j_{\max}(\nu) = \arg\max_j x^\nu_j$. If the vectors x^ν have the same mean norm, we can assimilate x^ν whose largest coordinate is the jth to e_j (distributed representation of E_j). Then the jth column of $\mathcal{M}$ writes

$$\mathcal{M}_{\cdot j} = \sum_{\boldsymbol{E}^T \mathbf{M}_1 \boldsymbol{F} y^\nu = e_j} y^\nu = \boldsymbol{F}^T \left(\sum_{Y^\nu \in \mathbf{M}_1^{-1}(E_j)} Y^\nu \right). \tag{3.8}$$

It is clear that the latter sum is an element of $\mathbf{M}_1^{-1}(E_j)$. Section A.4 shows that when the F_i are regular, the sum converges toward $\mathbf{M}_1^\dagger E_j$. The matrix $\mathcal{M}$ is then exactly the distributed representation of the Moore-Penrose inverse of $\mathbf{M}_1$. Informally, this winner-take-all learning rule works by equalizing

learning over input vectors, whatever their original distribution. In pratice, a soft competitive approach can be used (e.g., to speed up the learning), but the proportion of winners must be kept low in the presence of strong anisotropies. It must be noted that this applies only if the vectors x^ν have the same norm on average. If this condition is not fulfilled, a correction by $1/\|x^\nu\|^2$ must be applied.

This rule developed in a Hebbian context naturally extends to the parameter-dependent case.

3.4.3 Learning Parameter-Dependent Linear Mappings. We now treat the more general case where a linear mapping depends on a parameter. Typically, such a mapping arises as a local linear approximation (Jacobian) of a nonlinear transformation (see Bullock et al., 1993). Consider a nonlinear mapping $\psi = \phi(\chi)$ (e.g., ϕ is the inverse kinematic transformation for an arm; χ are the cartesian coordinates of the arm end point and ψ the joint angles). Linearization around χ_0 gives $\dot{\psi} = \mathbf{M}(\chi_0)\dot{\chi}$, $\mathbf{M}$ being the Jacobian of ϕ. If the value $\psi_0 = \phi(\chi_0)$ is given, the nonlinear mapping can be computed by incrementally updating ψ with $\dot{\psi} = \mathbf{M}\dot{\chi}$ along any path starting at χ_0. Thus, the problem reduces to computing a parameter-dependent linear mapping, which can be written, using previous notations, as $Y = \mathbf{M}(P)X$, where P is a parameter. We denote by $\mathbb{P}$ the physical space of parameters and $\mathcal{P}$ the space of the neuronal representation of parameters (e.g., $\mathbb{P}$ is the two-dimensional space of joint angles and $\mathcal{P}$ can be a set of postural signals).

A solution to this problem is to consider the coefficients $\mathcal{M}_{ij}$ corresponding to the distributed representation of $\mathbf{M}$ not as weights, but as activities of neurons modulated by the parameter $P \in \mathbb{P}$, and to assume a multiplicative interaction between $\mathcal{M}_{ij}$ and x_j. In the simplest case where P modulates linearly the coefficients, this can be written

$$y = \mathcal{M}\, x \qquad \text{and} \qquad \mathcal{M} = \mathcal{V} p \quad \left(\text{i.e., } \mathcal{M}_{ij} = \sum_k \mathcal{V}_{ijk} p_k\right), \tag{3.9}$$

where $\mathcal{V}$ is a set of weights defined over $\mathcal{E} \times \mathcal{F} \times \mathcal{P}$ and $p \in \mathcal{P}$.

Then the mapping is learned by retaining for each neuron of layer $\mathcal{M}$ the relationship between the input p and the desired output $\mathcal{M}^*_{ij} = \sum_\nu y_i^\nu x_j^\nu$. Thus, the weights $\mathcal{V}$ can be obtained by

$$\Delta \mathcal{V}_{ijk} \propto (y_i^\nu x_j^\nu - \mathcal{M}_{ij}) p_k^\nu, \tag{3.10}$$

which is a stochastic error-correcting learning rule. Contrary to the standard delta rule, equation 3.10 does not require an external teacher, as the reference signal is computed internally. Moreover, connectivity $\mathcal{V}_{ijk}$ can be far from complete, as lateral connections between $\mathcal{M}_{ij}$ units can help to form the desired activity profile (see section 3.3.3; Baraduc et al., 1999). Note that if

the parameter P is coded by a population of cosine-tuned neurons (i.e., p is a distributed representation of P), then equation 3.10 simplifies to a Hebbian rule:

$$\Delta \mathcal{V}_{ijk} \propto y_i^\nu x_j^\nu p_k^\nu.$$

In a more general case, the activities $\mathcal{M}_{ij}$ can depend on p via a perceptron or a multilayer perceptron. The learning rule, equation 3.10, can then be transformed to include a transfer function and possibly be the first step of an error backpropagation.

4 Generalization to Other Tuning Functions

It can be asked whether the mechanisms and properties of distributed computation proposed here depend on the specific cosine tuning that has been assumed (see equation 2.2). We now show that these results can be extended to a broad class of tuning functions (see equation 2.1), if we assume that the E_i have a uniform distribution. Following Georgopoulos et al. (1988), we use a continuous formalism (see also Mussa-Ivaldi, 1988). Given the previous assumptions, the uniformity guarantees that the population vector points in the same direction as the encoded vector (Georgopoulos et al., 1988):

$$\int\int f(X \cdot E, \beta) E \, dP_E \, dP_\beta = X. \tag{4.1}$$

The independence of β and E allows writing (Georgopoulos et al., 1988)

$$\int\int f(X \cdot E, \beta) E \, dP_E \, dP_\beta = \int \left(\int f(X \cdot E, \beta) E \, dP_{E|\beta} \right) dP_\beta.$$

Thus, any demonstration made with constant β can be easily generalized to varying β. Accordingly, we remove β in the following calculations.

4.1 Encoding-Decoding Method.

4.1.1 Distributed Representation of Vectors. The distributed representation of a vector X in $\mathbb{E}$ is no more a vector but a function $x = x(E) = f(X \cdot E)$. According to our hypothesis, the vector X can be recovered from its distributed representation x (see equation 4.1).

The dot product of the distributed representations of two vectors X and Z in $\mathbb{E}$ is defined by

$$h(X, Z) = \int f(X \cdot E) f(Z \cdot E) \, dP_E. \tag{4.2}$$

We first observe that h can be manipulated as a tuning function. A vector can be reconstructed from tuning curve functions (see equation 4.1), as well

as from h:

$$\int h(X,E)E\,dP_E = \int\int f(X\cdot E')f(E\cdot E')E\,dP_E\,dP_{E'}$$
$$= \int f(X\cdot E')\underbrace{\int f(E\cdot E')E\,dP_E}_{E'}\,dP_{E'}$$
$$= X. \tag{4.3}$$

This property is immediate in the cosine case since $h = f =$ dot product. In the general case, it can be shown that $h(X, Z)$ is a function of $X \cdot Z$ and that if f is nondecreasing, so is h (see section A.5).

4.1.2 Distributed Representation of Linear Transformations. There is a theoretical form (no longer a matrix, but a function) for the distributed representation of a linear mapping $\mathbf{M}$. It is defined by

$$\mathcal{M}\,(E,F) = g(F^T\mathbf{M}E)$$

and

$$y(F) = \int \mathcal{M}\,(E,F)\,x(E)\,dP_E, \tag{4.4}$$

where $y(F)$ is the distributed output corresponding to the distributed input $x(E) = f(X \cdot E)$ of a physical vector X. This exact counterpart of the cosine case (see equation 2.2) is easily demonstrated by showing that $\int_F y(F)F\,dP_F = Y$, with $Y = \mathbf{M}X$.

4.2 Stabilizing Distributed Representations. In the same way, there is a straightforward generalization of matrix $\mathcal{W}_{\mathcal{E}}$ (see section 3.3.3) defined by

$$\mathcal{W}_{\mathcal{E}}(E,E') = f(E\cdot E').$$

However, unlike the cosine case, these theoretical forms are not particularly useful since they are not in general similar to versions obtained by learning. Thus, in the following section, we derive and use Hebbian versions $\mathcal{M}^*$ and $\mathcal{W}^*_{\mathcal{E}}$ of $\mathcal{M}$ and $\mathcal{W}_{\mathcal{E}}$.

4.3 Learning Distributed Representation of Linear Transformations. The learning rules for the fixed or the parameter-dependent mapping still apply. We use a continuous formalism for both tuning functions and training examples. A straightforward derivation proves that the distributed transformation can be learned as before through input-output correlations. It can be shown that the distributed map corresponding to a linear transformation $\mathbf{M}$ between vectorial spaces $\mathbb{E}$ and $\mathbb{F}$ is represented by the function

$$\mathcal{M}^*(E,F) = \int f(X^\nu\cdot E)\,g(\mathbf{M}X^\nu\cdot F)\,dP_\nu, \tag{4.5}$$

where P_ν is the distribution of training examples (see appendix A.6). It can be seen that $\mathcal{M}^*(E, F)$ is a function of $E \cdot F$ using the method developed for equation 4.2.

Next we define $\mathcal{W}^*_\mathcal{E}$ as the distributed representation of the identity mapping on $\mathbb{E}$ obtained by learning (see equation 4.5)

$$\mathcal{W}^*_\mathcal{E}(E, E') = \int f(X^\nu \cdot E)\, f(X^\nu \cdot E')\, dP_\nu.$$

From equation 4.2, we see that $\mathcal{W}^*_\mathcal{E} = h(E, E')$.

The function $\mathcal{W}^*_\mathcal{E}$ can be used as a feedforward or lateral interaction function. Any input distribution $x(E)$ is transformed as

$$\mathcal{W}^*_\mathcal{E} x(E) = \int h(E, E')\, x(E')\, dP_{E'}. \tag{4.6}$$

If x is the distributed representation of a vector X of the physical space, it is immediately clear that

$$\mathcal{W}^*_\mathcal{E} x(E) = \int h(E, E')\, f(X \cdot E')\, dP_{E'},$$

which is a function of nondecreasing function $X \cdot E$ (see the method in section A.5). $\mathcal{W}^*_\mathcal{E}$ modifies the profile of activity, but changes neither the preferred attribute nor the center of mass of a population code.

If x is any distribution, the result of equation 4.6 depends on the shape of the dot product function (and thus the tuning function since the two are tightly related; see section 4.4). $\mathcal{W}^*_\mathcal{E}$ is a Fredholm integral operator with kernel h. If the kernel is degenerate—that is, it can be written as a finite sum of basis functions (e.g., Fourier series)—Im $\mathcal{W}^*_\mathcal{E}$ is a finite dimensional space generated by these functions. Thus, $\mathcal{W}^*_\mathcal{E}$ suppresses all other harmonics. The case of a cosine distribution of lateral interactions (see section 3) corresponds to a two-dimensional space generated by cos and sin functions (and $\mathcal{W}^*_\mathcal{E}$ is a projection). Gaussian distribution of weights, which contains a few significant harmonics, is known empirically to suppress noise efficiently (Douglas, Koch, Mahowald, Martin, & Suarez, 1995; Salinas & Abbott, 1996).

However, $\mathcal{W}^*_\mathcal{E}$ is not in general a projection, which is problematic if $\mathcal{W}^*_\mathcal{E}$ represents a transform through recurrent connections. Solutions in the discrete spatial case have been discussed (see 3.3.3) and extend to this case (Zhang, 1996). In particular, the scaling of the largest eigenvalue is possible since $\mathcal{W}^*_\mathcal{E}$ has the largest eigenvalue, which is equal to $\|\mathcal{W}^*_\mathcal{E}\|$.

After learning, the output neurons are not tuned to input vectors as they are during the learning phase; that is, $g(\mathbf{M}X \cdot F)$ are not their tuning functions. Indeed, activity of an output neuron is

$$y(F) = \int_\nu h(X, X^\nu)\, g(\mathbf{M}X^\nu \cdot F)\, dP_\nu, \tag{4.7}$$

which is generally not equal to $g(\mathbf{M}X \cdot F)$. However, using the same reasoning as for equation 4.2 (see section A.5), we can show that

$$y(F) = \tilde{g}(\mathbf{M}X \cdot F).$$

It follows that the y are still broadly tuned to $\mathbf{M}X$; moreover, the PAs in input space keep the same expression $F^T\mathbf{M}$ as in the cosine case.

4.4 Numerical Results for 2D Circular Normal Tuning Functions. Contrary to the cosine case, learning with tuning function g leads to a different output tuning $\tilde{g}$. Is this change important? How similar are these two tuning curves?

We illustrate here the differences among the intrinsic tuning functions (f and g), the dot product (h), and the output tuning function ($\tilde{g}$), using circular normal (CN) tuning functions (Mardia, 1972) in $\mathbb{R}^2$. These functions have a profile similar to a gaussian while being periodic. Their general expression is $f(\cos\theta) = Ae^{K\cos\theta} + B$. We used the following version for both input and output tuning,

$$f(u) = g(u) = \frac{e^{Ku} - e^{-K}}{e^K - e^{-K}},$$

where K controls the width at half-height. Thus, f and g take values between 0 and 1 if the coded vectors and the PAs are unit vectors.

With these assumptions, $h = f * f$, where $*$ is the convolution, and thus their respective Fourier coefficients verify $\hat{h}_n = \hat{f}_n^2$. Interestingly, the distribution of the Fourier coefficients of CN functions is such that h, once normalized between 0 and 1, is very close to a broader CN function h_{CN}. In our numerical simulations, the relative error was

$$\frac{\|h_{\mathrm{normalized}} - h_{\mathrm{CN}}\|}{\|h_{\mathrm{normalized}}\|} < 2\%,$$

where $\|h\|$ denotes the $\mathcal{L}_2$-norm of h. However, the convolution leads to a widening of h compared to f (see Figure 1A), since it favors the largest Fourier coefficients, which happen to be the first for CN functions. This broadening effect is maximal for f of width ≈ 110 deg (see Figure 1B). Since $\tilde{g} = h * g$ (see equation 4.7), $\tilde{g}$ is still broader than h (see Figure 1B).

These results show that feedforward or recurrent neural processing preserves the general shape of intrinsic tuning functions but increases their width. After about two to five feedforward steps, the tuning of output neurons ($\tilde{g}$) is close to a cosine.

5 Deviations for Nonuniform Distributions of PAs

The preceding results on noncosine tuning curves were obtained for a uniform distribution of PAs, whereas a weaker constraint (regularity condition)

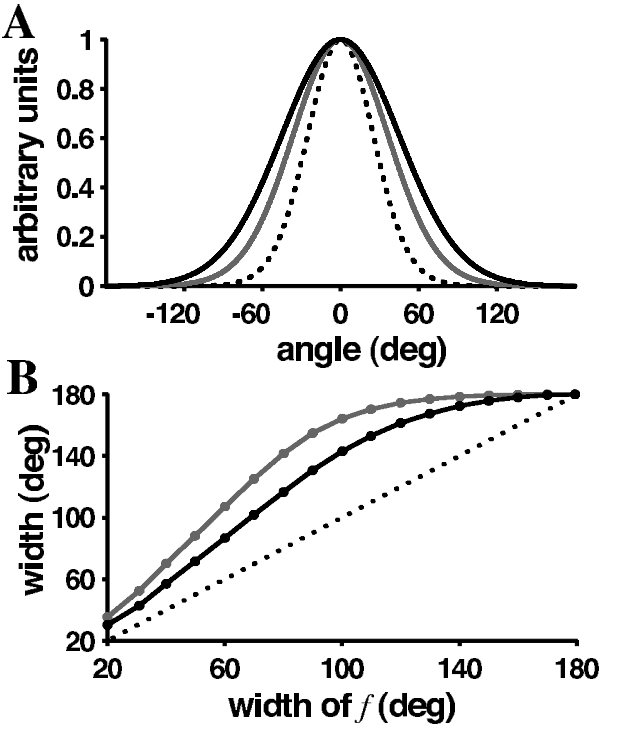

Figure 1: (A) Shape of the intrinsic tuning curve of input and output neurons (f, dotted line), the distributed dot product (h, gray line), and input tuning of output neurons ($\tilde{g}$, solid line). The width (at half-height) of f was 60° ($K = 5.2$). The curves for h and $\tilde{g}$ were constructed from the first 20 Fourier coefficients of f. (B) Width (at half-height) of h (gray line) and $\tilde{g}$ (solid line) as a function of the width of f ($K = .01$–45).

was sufficient in the cosine case. Here we explore numerically to what extent the population computation can be accurate for a regular nonuniform distribution of PAs. In relation to electrophysiological data (Oyster & Barlow, 1967; Lacquaniti, Guigon, Bianchi, Ferraina, & Caminiti, 1995; Wylie, Bischof, & Frost, 1998), such a distribution was assumed clustered along preferred axes (here in 2D).

To express the clustering along the axis $\theta = 0$, the probability density of a vector $E = (\cos\theta, \sin\theta)$ was assumed to follow $dP_E/d\theta \propto \exp(-\theta^2/V)$ for $\theta \in]-\pi/4; \pi/4]$. The same density was used modulo $\pi/2$ for the directions $\theta = \pi/2, \pi$, and $3\pi/2$. The resulting densities for four different values of V, from $V = 3$ (moderately clustered distribution) to $V = 10^{-12}$ (PAs aligned on the axes), are plotted in the inset of Figure 2.

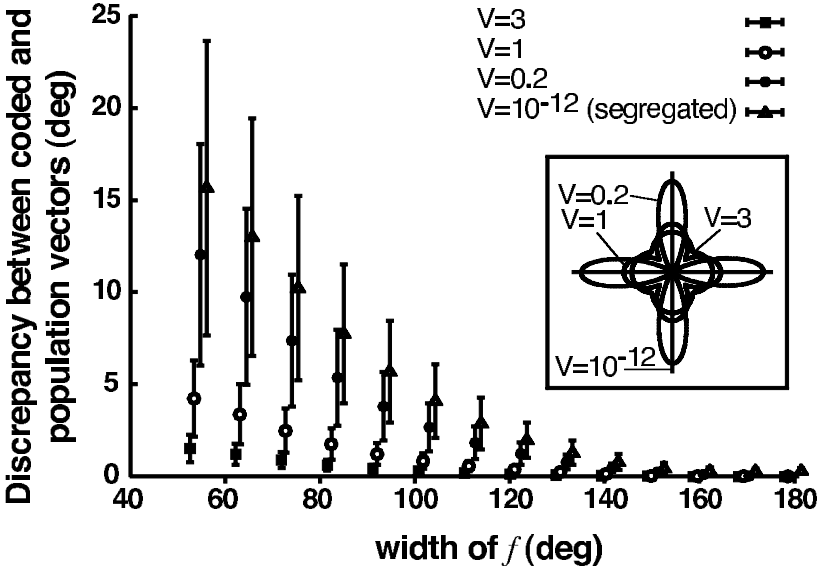

Figure 2: Precision of the distributed computation as measured by the discrepancy between the decoded input and output in the case of the distributed identity mapping. The error in the transformation is plotted as a function of the tuning width of f and the clustering of the 2D basis vectors E around the axes. The inset shows the four distributions of E that have been tested (see the text). For each condition, the error was calculated as the mean absolute difference between encoded and decoded vectors over 1000 trials (i.e. 1000 randomly chosen encoded vectors).

To illustrate how the scheme of distributed computation proposed here behaves in these conditions, we measured the errors induced by the distributed computation $\mathcal{W}\mathcal{E}$ of the identity function. The population was sampled exactly regular, so that heavy computations involving large numbers of neurons could be avoided. Assuming tuning functions to be circular normal, we computed the angular difference between the decoded input and output vectors for different distributions of E and different tuning widths. The results shown in Figure 2 were obtained by computing the identity on 1000 random vectors on a regular population of 1000 neurons.

As expected, the most uniform distribution behaves best, generating very small errors. The deviation of the population vector increases with the clustering of the basis vectors. However, the more the tuning curves broaden, the less this effect is pronounced. In particular, if the tuning width is greater than 100 degrees, the directional error in the population vector is always inferior to 5 degrees. We conclude that the distributed computation of linear

mappings is still possible with minimal error in the case of clustered PAs when tuning curves are sufficiently broad.

6 Discussion

This article has addressed the calculation of vectorial transformations by populations of cosine-tuned neurons in a linear framework. We have shown that appropriate distributed representations of these transformations were made possible by simple and common properties of neural computation and learning: decoding with the population vector, regular distributions of tuning selectivities and input-output training examples, Hebbian learning, and cosine-tuned lateral interactions between neurons. We have analytically extended this result to the noncosine broadly tuned case for uniform distributions and numerically to regular nonuniform distributions.

The use of the population vector may appear problematic because it is in general not an optimal decoding method (Salinas & Abbott, 1994). Statistical optimality is clearly an important theoretical issue (Snippe, 1996; Pouget et al., 1998), but it is unclear whether it is also a relevant concept for computation in the nervous system. As emphasized by several authors (Paradiso, 1988; Salinas & Abbott, 1994), the use of a large number of cells to estimate a parameter is likely to overcome variability in single-cell behavior. In fact, accuracy (small bias and low variance compared to the coding range) may be more important than optimality. Furthermore, the main difficulty with the PV method is its poor behavior when used for biased distributions of preferred directions (Glasius, Komoda, & Gielen, 1997) or populations of sharply tuned neurons (Seung & Sompolinsky, 1993). We have restricted our theory to regular or uniform distributions and broadly tuned neurons. For regular distributions, the PV method is an optimal linear estimator (Salinas & Abbott, 1994). Broadly tuned neurons allow the PV method to approach the maximum likelihood method for Poisson noise (Seung & Sompolinsky, 1993).

The question arises whether electrophysiological data actually satisfy the regularity condition. This is clearly the case for uniform distributions (Georgopoulos et al., 1986; Schwartz, Kettner, & Georgopoulos, 1988; Caminiti, Johnson, Galli, Ferraina, & Burnod, 1991). However, not all distributions are uniform (Hubel & Wiesel, 1962; Oyster & Barlow, 1967; van Gisbergen, van Opstal, & Tax, 1987; Cohen, Prud'homme, & Kalaska, 1994; Prud'homme & Kalaska, 1994; Lacquaniti et al., 1995; Rosa & Schmid, 1995; Wylie et al., 1998), and it remained to be checked if these distributions are regular. A particular distribution is a clustering of PAs along preferred axes (Oyster & Barlow, 1967; Cohen et al., 1994; Prud'homme & Kalaska, 1994; Lacquaniti et al., 1995; Wylie et al., 1998; see also Soechting & Flanders, 1992). Populations of neurons in posterior parietal cortex of monkeys have such a distribution of PAs and satisfy the regularity condition ($p < 0.01$, unpublished observations from the data of Battaglia-Mayer et al., 2000). The same was seen

in anterior parietal cortex (E. Guigon, unpublished observations from the data of Lacquaniti et al., 1995). This latter observation indicates that vectorial computation can occur not only in uniformly distributed neuronal populations, but also at the different levels of a sensorimotor transformation where neurons are closely related to receptors or actuators (Soechting & Flanders, 1992).

The regularity condition allows basic operations of linear algebra to be implemented in a distributed fashion. A similar principle was first proposed by Touretzky et al. (1993). They introduced an architecture called a sinusoidal array, which encodes a vector as distributed activity across a neuronal population (see equation 2.2), and they used this architecture to solve reaching and navigation tasks (Touretzky et al., 1993; Redish, Touretzky, & Wan, 1994; Redish & Touretzky, 1994). However, in their formulation, vector rotation (which is a linear transformation) was implemented in a specific way, using either shifting circuits (Touretzky et al., 1993) or repeated vector addition (Redish et al., 1994). In our framework, vector rotation can be represented by a distributed linear transformation as any morphism (see section 3.2).

We derived closely related results for a broad class of tuning functions (see equation 2.1), although under more restricting hypotheses (uniform distribution of PAs). A theoretically unbiased population vector can be constructed from a nonuniformly distributed population of neurons by adjusting the distribution of tuning strength (Germain & Burnod, 1996) or tuning widths (Glasius et al., 1997). However, these methods cannot be used to release the uniformity constraint since the hypothesis of independence of PAs and parameters distribution is violated. A particular example of nonuniform distribution of PAs is their clustering along axes (Oyster & Barlow, 1967; Cohen et al., 1994; Prud'homme & Kalaska, 1994; Lacquaniti et al., 1995; Wylie et al., 1998). In this case, although the operation of the network is only exact for pure cosine tuning curves, we have shown numerically that a good approximate computation is still possible if the tuning is sufficiently broad.

Salinas and Abbott (1995) derived a formal rule to learn the identity mapping in dimension 1 (i.e., $x \longrightarrow x$ through uniformly distributed examples). Their demonstration relies on the fact that the tuning curves and synaptic connections depend on only the magnitude of the difference between preferred attributes. Our results generalize this idea to arbitrary linear mappings in any dimension. The generalized constraint is that the tuning curves and connections depend on the scalar product of preferred attributes, which includes the one-dimensional case. Salinas and Abbott (1995) also provided a solution to $(x, y) \longrightarrow x + y$ in dimension 1. However, their method may not be generalizable to higher dimensions. In fact, this transformation is not a (bi)linear transformation and is not easily accounted by our theory (except in the cosine case; see also Touretzky et al., 1993). Interestingly, when one asks how information is read out from distributed maps of directional

signals, vector averaging and winner-take-all are more likely decision processes than vector summation (Salzman & Newsome, 1994; Zohary, Scase, & Braddick, 1996; Groh, Born, & Newsome, 1997; Lisberger & Ferrera, 1997; Recanzone, Wurtz, & Schwarz, 1997).

An important application of our theory is learning a coordinate transformation from its Jacobian. This problem can be solved formally as an ensemble of position-dependent linear mappings (Baraduc et al., 1999). However, unlike previous models (Burnod et al., 1992; Bullock et al., 1993), it is not required that position information be coded in a topographic manner. Arbitrary codes for position can be used provided that the mapping between the position and the distributed representation of the Jacobian (see equation 3.9) is correctly learned. The most interesting point is that neurons of the network display realistic firing properties, which resemble those of parietal and motor cortical neurons. These results render the theory presented here attractive for modeling sensorimotor transformations.

Appendix

A.1 Convergence of Q in Probability for Regular PAs. Here we show that $\mathbf{Q} = \frac{1}{N}\boldsymbol{E}\boldsymbol{E}^T$ converges in probability toward the identity matrix (up to a multiplicative constant) if the distribution of the PAs E_i^T is regular.

The kth ($1 \leq k \leq D$) diagonal term of $\mathbf{Q}$ is

$$Q_{kk} = \frac{1}{N}\sum_{i=1}^{N} E_{ik}^2,$$

which tends in probability toward σ_E^2 when N tends to infinity. Indeed,

$$\mathrm{V}\,(Q_{kk}) = \frac{1}{N^2}\sum_{i=1}^{N} \mathrm{V}(E_{ik}^2) = \frac{\mathrm{V}\left(E_{1k}^2\right)}{N}.$$

The off-diagonal element Q_{kl} ($k \neq l$) of $\mathbf{Q}$ is $Q_{kl} = \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} E_{ik}E_{il}$; hence,

$$\lim_{N\to\infty} \langle Q_{kl} \rangle = 0 \qquad \text{and} \qquad \lim_{N\to\infty} \mathrm{V}\,(Q_{kl}) = \lim_{N\to\infty} \mathrm{V}\,(E_{ik}E_{il})\,/N = 0.$$

Thus, $\mathbf{Q}$ converges in probability toward $\sigma_E^2\mathbf{I}_D$.

A.2 Correlated Noise. Writing $\mathcal{Q}$ the correlation matrix of the noise, the variance $\mathrm{V}\,(\boldsymbol{E}\eta/N)$ of the read-out vector can be expressed as the first diagonal of matrix

$$\mathbf{V} = \frac{1}{N^2}\,\boldsymbol{E}\mathcal{Q}\,\boldsymbol{E}^T.$$

For an independently distributed gaussian noise, $\mathcal{Q}$ is proportional to the identity matrix and $\mathrm{V}(\boldsymbol{E}\eta/N) \propto 1/N$. In the case of correlated gaussian noise,

$$\mathcal{Q} = \sigma_\eta^2 \left[\mathbf{I}_N + c(\mathbb{I}_{N,N} - \mathbf{I}_N)\right],$$

and we get

$$\mathbf{V} = \frac{1-c}{N^2}\sigma_\eta^2 \boldsymbol{E}\boldsymbol{E}^T = \frac{(1-c)\sigma_\eta^2\sigma_E^2}{N}\mathbf{I}_D.$$

For Poisson noise, the noise correlation matrix is $\mathcal{Q}_{ij} = \left[(1-c)\delta_{ij}x_i + c\sqrt{x_i x_j}\right]$. If there is no correlation ($c = 0$), matrix $\mathbf{V}$ is $\boldsymbol{E}\mathrm{diag}(x_i)\boldsymbol{E}^T/N^2$, and its ith diagonal term writes

$$\frac{1}{N^2}\sum_k x_k E_{ki}^2 \le \frac{1}{N}x_{\max}\mathbf{Q}_{E_{ii}} = \frac{x_{\max}}{N}.$$

For nonzero c, the term

$$V_c = \frac{c}{N^2}\,\mathrm{diag}\left(\boldsymbol{E}\left[\sqrt{x_i x_j}\right]_{ij}\boldsymbol{E}^T\right) = \frac{c}{N^2}\left(\boldsymbol{E}\sqrt{x}\right)^2$$

must be added. This term is independent of N and can be evaluated numerically for a few types of distribution of $\boldsymbol{E}^T$. For instance, if the PAs are uniformly distributed in 3D space and the minimum firing rate equals zero, and assuming that the norms of the E_i and their directions are independently distributed,

$$\begin{aligned} V_c &= c\|E\|\left[\frac{1}{4\pi}\int_{-1}^{1}\int_{2\pi}\sqrt{1+s}\left(\sqrt{1-s^2}\cos\theta, \sqrt{1-s^2}\sin\theta, s\right)\, ds\, d\theta\right]^2 \\ &\le c\,x_{\max}\left[\frac{1}{2}\int_{-1}^{1} s\sqrt{1+s}\, ds\right]^2 \\ &\le \frac{8}{225}c\,x_{\max}, \end{aligned}$$

hence, the upper bound of equation 3.4 (here $\|E\|$ denotes the mean norm of the E_i vectors). The derivation of equation 3.5 is left to the reader.

The demonstration of the properties of $\mathcal{W}_{\mathcal{E}}$ is analogous.

A.3 General Cosine Tuning. In most of the sections on coding and decoding, the baseline term b was 0. We now show how the results change for a nonzero baseline.

We can use an approach similar to that in section 3.1 to show that

$$\frac{1}{N}\sum x_i y_i \longrightarrow X \cdot Y + \hat{b} \qquad \text{in probability as} \quad N \to \infty,$$

where x, y are distributed representations of physical vectors X, Y, and

$$\hat{b} = \lim_{N\to\infty} \frac{b^T b}{N}$$

depends only on b. Thus, the scalar product of two vectors deduces easily from the scalar product of their distributed representations.

The expression for matrix $\mathcal{W}_{\mathcal{E}}$ (see section 3.3.3), which we write $\mathcal{W}$ for simplicity, transforms to

$$\mathcal{W}' = \mathcal{W} + \frac{b}{N\bar{b}}\mathbb{I}_N^T,$$

where $\bar{b}$ the mean over i of b_i. It can be checked that $\mathcal{W}'$ is a projection on the affine subspace $\mathcal{E}_p + b$ and possesses the same properties as $\mathcal{W}$.

Learning a linear transformation amounts to calculating the matrix

$$\mathcal{M}^* = \sum_\nu (\boldsymbol{F}^T Y^\nu + b_F)(\boldsymbol{E}^T X^\nu + b_E)^T,$$

where b_E and b_F denote the mean activity of input and output neurons, respectively. If the training inputs satisfy the regularity condition, we have

$$\mathcal{M}^* = \underbrace{\boldsymbol{F}^T \mathbf{M} \left(\sum_\nu X^\nu (X^\nu)^T\right) \boldsymbol{E}}_{\kappa\mathcal{M}} + \underbrace{\boldsymbol{F}^T \mathbf{M} \left(\sum_\nu X^\nu\right) b_E^T}_{0}$$
$$+ \underbrace{b_F \left(\sum_\nu (X^\nu)^T\right) \boldsymbol{E}}_{0} + N_{\text{ex}} b_F b_E^T,$$

where κ is a proportionality constant defined by equation 3.6. The regularity condition leads to $\kappa = \rho^2 N_{\text{ex}}/D_E$, where ρ is the mean norm of input examples and $D_E = \dim \mathbb{E}$. Hence,

$$\mathcal{M}^* \propto \mathcal{M} + \frac{\rho^2}{D_E} b_F b_E^T.$$

Thus, appropriate mapping occurs, although there is no guarantee that the output baseline activity will equate the baseline activity of the training patterns.

A.4 Nonuniform X^ν: Convergence Toward the Moore-Penrose Inverse. When F_i are uniformly distributed in $\mathbb{F}$, learning from the examples of a noninvertible mapping $\mathbf{M}_1$ between output and input converges toward the distributed representation of its Moore-Penrose inverse.

Start from equation 3.8, and take $Y^\nu \in \mathbf{M}_1^{-1}(E_j)$. As the Moore-Penrose inverse of $\mathbf{M}_1$ is zero on the kernel of $\mathbf{M}_1$, we can write $Y^\nu = \mathbf{M}_1^\dagger E_j + K^\nu$, where $K^\nu \in \ker \mathbf{M}_1$.

If we assume that y^ν are uniformly distributed in $\mathcal{F}_p$ (index p is defined in section 3.3.3), then the distribution of Y^ν is uniform. It follows that the distribution of the K^ν is symmetric with respect to zero. Hence,

$$\sum_{\mathbf{M}_1 Y^\nu = E_j} Y^\nu \propto \mathbf{M}_1^\dagger E_j.$$

The proportionality factor is identical for all j if equation 3.7 is used, which completes the proof.

A.5 Distributed Dot Product. We show now that $h(X, Z)$ is a function of $X \cdot Z$, assuming that the encoded vectors are unit vectors. We note S the unit sphere of $\mathbb{E}$ and define

$$S_\alpha(X) = \{U \in S \mid U \cdot X = \alpha\}.$$

Then

$$\begin{aligned} h(X, Z) &= \int_\alpha \int_{S_\alpha(X)} f(\alpha) f(Z \cdot (\alpha X + E^\perp))\, dP_E\, d\alpha \\ &= \int_\alpha f(\alpha) \underbrace{\int_{S_\alpha(X)} f(\alpha Z \cdot X + Z \cdot E^\perp)\, dP_E}_{h_\alpha(X,Z)}\, d\alpha, \end{aligned}$$

where $E^\perp$ is the projection of E on the subspace orthogonal to X. Let us define

$$S_{\alpha u} = \{E \in S \mid Z \cdot E^\perp = u \text{ and } X \cdot E = \alpha\},$$

and write

$$\begin{aligned} h_\alpha(X, Z) &= \int_u f(\alpha Z \cdot X + u) \int_{S_{\alpha u}} dP_E \\ &= \int_u f(\alpha Z \cdot X + u)\, dP_u, \end{aligned}$$

which depends only on $X \cdot Z$. This is the required result. Moreover, if f is nondecreasing (which is generally the case for a tuning function), it is immediate that h is nondecreasing.

A.6 Hebbian Learning of Distributed Maps: General Case. The following derivation shows that Hebbian learning of linear mappings can still be achieved.

Using equation 4.4, we obtain

$$\int_F y(F)F\, dP_F = \int_E \int_F \int_\nu f(X^\nu \cdot E) g(\mathbf{M}X^\nu \cdot F) x(E) F\, dP_E\, dP_F\, dP_\nu$$

$$= \int_\nu \int_F \underbrace{\left[\int_E f(X^\nu \cdot E) f(X \cdot E)\, dP_E \right]}_{h(X, X^\nu)} g(\mathbf{M}X^\nu \cdot F) F\, dP_F\, dP_\nu$$

$$= \int_\nu h(X, X^\nu) \underbrace{\left[\int_F g(\mathbf{M}X^\nu \cdot F) F\, dP_F \right]}_{\mathbf{M}X^\nu} dP_\nu$$

$$= \int_\nu h(X, X^\nu) \mathbf{M}X^\nu\, dP_\nu.$$

If we assume that the distribution of training examples has the same properties as the distribution of PAs, then $Y = \mathbf{M}X$ (using equation 4.3). This proves that the vector represented in the output activities is correct.

Acknowledgments

We thank Yves Burnod for fruitful discussions, Alexandre Pouget and an anonymous reviewer for helpful comments, and Marc Maier and Pierre Fortier for revising our English.

References

Abbott, L., & Dayan, P. (1999). The effect of correlated variability on the accuracy of a population code. *Neural Comp., 11*, 91–101.

Baldi, P., & Hornik, K. (1995). Learning in linear networks: A survey. *IEEE Trans. Neural Netw., 6*(4), 837–858.

Baraduc, P., Guigon, E., & Burnod, Y. (1999). Where does the population vector of motor cortical cells point during arm reaching movements? In M. Kearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems, 11* (pp. 83–89). Cambridge, MA: MIT Press. Available on-line at: http://www.snv.jussieu.fr/guigon/nips99.pdf.

Battaglia-Mayer, A., Ferraina, S., Mitsuda, T., Marconi, B., Genovesio, A., Onorati, P., Lacquaniti, F., & Caminiti, R. (2000). Early coding of reaching in the parietooccipital cortex. *J. Neurophysiol., 83*(4), 2374–2391.

Bullock, D., Grossberg, S., & Guenther, F. (1993). A self-organizing neural model of motor equivalence reaching and tool use by a multijoint arm. *J. Cogn. Neurosci., 5*, 408–435.

Burnod, Y., Grandguillaume, P., Otto, I., Ferraina, S., Johnson, P., & Caminiti, R. (1992). Visuo-motor transformations underlying arm movements toward visual targets: A neural network model of cerebral cortical operations. *J. Neurosci., 12*, 1435–1453.

Caminiti, R., Johnson, P., Galli, C., Ferraina, S., & Burnod, Y. (1991). Making arm movements within different parts of space: The premotor and motor cortical representation of a coordinate system for reaching to visual targets. *J. Neurosci., 11*, 1182–1197.

Cohen, D., Prud'homme, M., & Kalaska, J. (1994). Tactile activity in primate primary somatosensory cortex during active arm movements: Correlation with receptive field properties. *J. Neurophysiol., 71*, 161–172.

Crowe, A., Porrill, J., & Prescott, T. (1998). Kinematic coordination of reach and balance. *J. Mot. Behav., 30*(3), 217–233.

Davis, P. (1979). *Circulant matrices*. New York: Wiley.

Douglas, R., Koch, C., Mahowald, M., Martin, K., & Suarez, H. (1995). Recurrent excitation in neocortical circuits. *Science, 269*, 981–985.

Gawne, T., & Richmond, B. (1993). How independent are the messages carried by adjacent inferior temporal cortical neurons? *J. Neurosci., 13*, 2758–2771.

Georgopoulos, A. (1996). On the translation of directional motor cortical commands to activation of muscles via spinal interneuronal systems. *Cogn. Brain Res., 3*(2), 151–155.

Georgopoulos, A., Kettner, R., & Schwartz, A. (1988). Primate motor cortex and free arm movements to visual targets in 3-dimensional space. II. Coding of the direction of movement by a neuronal population. *J. Neurosci., 8*, 2928–2937.

Georgopoulos, A., Schwartz, A., & Kettner, R. (1986). Neuronal population coding of movement direction. *Science, 233*, 1416–1419.

Germain, P., & Burnod, Y. (1996). Computational properties and auto-organization of a population of cortical neurons. In *Proc. International Conference on Neural Networks, ICNN'96* (pp. 712–717). Piscataway, NJ: IEEE.

Glasius, R., Komoda, A., & Gielen, C. (1997). The population vector, an unbiased estimator for non-uniformly distributed neural maps. *Neural Netw., 10*, 1571–1582.

Groh, J., Born, R., & Newsome, W. (1997). How is a sensory map read out? Effects of microstimulation in visual area MT on saccades and smooth pursuit eye movements. *J. Neurosci., 17*(11), 4312–4330.

Grossberg, S., & Kuperstein, M. (1989). *Neural dynamics of adaptive sensory-motor control* (Exp. ed.). Elmsford, NY: Pergamon Press.

Hinton, G. (1984). Parallel computations for controlling an arm. *J. Mot. Behav., 16*(2), 171–194.

Hinton, G. (1992). How neural networks learn from experience. *Sci. Am., 267*(3),145–151.

Hubel, D., & Wiesel, T. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol. (Lond.), 160*, 106–154.

Kuperstein, M. (1988). Neural model of adaptive hand-eye coordination for single postures. *Science, 239*, 1308–1311.

Lacquaniti, F., Guigon, E., Bianchi, L., Ferraina, S., & Caminiti, R. (1995). Representing spatial information for limb movement: Role of area 5 in the monkey. *Cereb. Cortex, 5*(5), 391–409.

Lisberger, S., & Ferrera, V. (1997). Vector averaging for smooth pursuit eye movements initiated by two moving targets in monkeys. *J. Neurosci., 17*(19), 7490–7502.

Mardia, K. (1972). *Statistics of directional data*. London: Academic Press.

Mussa-Ivaldi, F. (1988). Do neurons in the motor cortex encode movement direction? An alternative hypothesis. *Neurosci. Lett., 91*, 106–111.

Mussa-Ivaldi, F., Morasso, P., & Zaccaria, R. (1988). Kinematic networks. A distributed model for representing and regularizing motor redundancy. *Biol. Cybern., 60*, 1–16.

Oyster, C., & Barlow, H. (1967). Direction-selective units in rabbit retina: Distribution of preferred directions. *Science, 155*, 841–842.

Paradiso, M. (1988). A theory for use of visual orientation information which exploits the columnar structure of striate cortex. *Biol. Cybern., 58*, 35–49.

Pouget, A., Zhang, K., Deneve, S., & Latham, P. (1998). Statistically efficient estimation using population coding. *Neural Comput., 10*(2), 373–401.

Prud'homme, M., & Kalaska, J. (1994). Proprioceptive activity in primate primary somatosensory cortex during active arm reaching movements. *J. Neurophysiol., 72*(5), 2280–2301.

Recanzone, G., Wurtz, R., & Schwarz, U. (1997). Responses of MT and MST neurons to one and two moving objects in the receptive field. *J. Neurophysiol., 78*(6), 2904–2915.

Redding, G., & Wallace, B. (1997). *Adaptive spatial alignment*. Hillsdale, NJ: Erlbaum.

Redish, A., & Touretzky, D. (1994). The reaching task: Evidence for vector arithmetic in the motor system? *Biol. Cybern., 71*(4), 307–317.

Redish, A., Touretzky, D., & Wan, H. (1994). The sinusoidal array: A theory of representation for spatial vectors. In F. Eeckman (Ed.), *Computation and neural systems* (pp. 269–274). Boston: Kluwer.

Rosa, M., & Schmid, L. (1995). Magnification factors, receptive field image and point-image size in the superior colliculus of flying foxes: Comparison with primary visual cortex. *Exp. Brain Res., 102*, 551–556.

Salinas, E., & Abbott, L. (1994). Vector reconstruction from firing rates. *J. Comput. Neurosci., 1*, 89–107.

Salinas, E., & Abbott, L. (1995). Transfer of coded information from sensory to motor networks. *J. Neurosci., 15*(10), 6461–6474.

Salinas, E., & Abbott, L. (1996). A model of multiplicative neural responses in parietal cortex. *Proc. Natl. Acad. Sci. U.S.A., 93*(21), 11956–11961.

Salzman, C., & Newsome, W. (1994). Neural mechanisms for forming a perceptual decision. *Science, 264*, 231–237.

Sanger, T. (1994). Theoretical considerations for the analysis of population coding in motor cortex. *Neural Comput., 6*, 29–37.

Schwartz, A., Kettner, R., & Georgopoulos, A. (1988). Primate motor cortex and free arm movements to visual targets in three-dimensional space. I. Relations

between single cell discharge and direction of movement. *J. Neurosci., 8*, 2913–2927.
Seung, H., & Sompolinsky, H. (1993). Simple models for reading neuronal population codes. *Proc. Natl. Acad. Sci. U.S.A., 90*, 10749–10753.
Snippe, H. (1996). Parameter extraction from population codes: A critical assessment. *Neural Comput., 8*(3), 511–529.
Soechting, J., & Flanders, M. (1992). Moving in three-dimensional space: Frames of reference, vector, and coordinate systems. *Annu. Rev. Neurosci., 15*, 167–191.
Strang, G. (1988). *Linear algebra and its applications* (3rd ed.). San Diego: Harcourt Brace Jovanovich.
Touretzky, D., Redish, A., & Wan, H. (1993). Neural representation of space using sinusoidal arrays. *Neural Comput., 5*, 869–884.
van Gisbergen, J., van Opstal, A., & Tax, A. (1987). Collicular ensemble coding of saccades based on vector summation. *Neuroscience, 21*, 541–555.
Wylie, D., Bischof, W., & Frost, B. (1998). Common reference frame for neural coding of translational and rotational optic flow. *Nature, 392*, 278–282.
Yang, H., & Dillon, T. (1994). Exponential stability and oscillation of Hopfield graded response neural network. *IEEE Trans. Neural Netw., 5*(5), 719–729.
Zemel, R., & Hinton, G. (1995). Learning population codes by minimizing description length. *Neural Comput., 7*(3), 549–564.
Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory. *J. Neurosci., 16*(6), 2112–2126.
Zhang, K., Ginzburg, I., McNaughton, B., & Sejnowski, T. (1998). Interpreting neuronal population activity by reconstruction: Unified framework with application to hippocampal place cells. *J. Neurophysiol., 79*(2), 1017–1044.
Zohary, E., Scase, M., & Braddick, O. (1996). Integration across directions in dynamic random dot displays: Vector summation or winner take all? *Vision Res., 36*(15), 2321–2331.
Zohary, E., Shadlen, M., & Newsome, W. (1994). Correlated neuronal discharge rate and its implications for psychophysical performance. *Nature, 370*, 140–143.

Received February 17, 2000; accepted July 5, 2001.