Biol. Cybern. 77, 433–446 (1997)

Biological Cybernetics

Biologically inspired calibration-free adaptive saccade control of a binocular camera-head

Jörg Bruske, Michael Hansen, Lars Riehn, Gerald Sommer

Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität Kiel, Preusserstrasse 1–9, D-24105 Kiel, Germany

Received: 27 August 1996 / Accepted in revised form: 22 July 1997

Abstract. This paper describes fast and accurate calibration-free adaptive saccade control of a four-degrees-of-freedom binocular camera-head by means of Dynamic Cell Structures (DCS). The approach has been inspired by biology because primates face a similar problem and there is strong evidence that they have solved it in a similar way, i.e., by error feedback learning of an inverse model. Yet the emphasis of this article is not on detailed biological modeling but on how incremental growth of our artificial neural network model up to a prespecified precision results in very small networks suitable for real-time saccade control. Error-feedback-based training of this network proceeds in two phases. In the first phase we use a crude model of the cameras and the kinematics of the head to learn the topology of the input manifold together with a rough approximation of the control function off-line. In contrast to, for example, Kohonen-type adaptation rules, the distribution of neural units minimizes the control error and does not merely mimic the input probability density. In the second phase, the operating phase, the linear output units of the network continue to adapt on-line. Besides our TRC binocular camera-head we use a Datacube image processing system and a Stäubli R90 robot arm for automated training in the second phase. It will be demonstrated that the controller successfully corrects errors in the model and rapidly adapts to changing parameters.

1 Introduction

Saccades are fast eye movements used by animals to change fixation from one point in the visual field to another. In the context of our binocular camera-head we refer to saccades as fast eye *and* head movements that serve the same purpose, i.e., rapid change of fixation. Since the emphasis is on rapidity and accuracy, pure feedback control is not the optimal choice for either animals or robots because of long delays and transient effects associated with the feedback loops. For example, in the case of human eye movements of up to 1000 deg/s and a delay for visual feedback of about 200 ms, there is no time for visual feedback to guide the eye to its final position (Carpenter 1988). The human saccade control system must therefore calculate the pattern of muscle activation in advance for each target position on the retina (Carpenter 1988) – in other words it must have solved the inverse kinematics[1] problem (Dean et al. 1991).

According to an idea first put forward by Kawato et al. (1987) and Miyamoto et al. (1988), the inverse kinematics can be learned by error feedback, i.e., by using an error signal proportional to an error in target coordinates as training feedback for biological or artificial neural networks. In addition to there being strong neurobiological evidence that error feedback is also the way monkeys and humans learn and maintain saccadic accuracy (Dean et al. 1994), Kawato has successfully applied this principle to a variety of robotic tasks, including trajectory control of industrial robots (Kawato 1990).

Error feedback learning for saccade control of a simulated (in contrast to an electromechanically or, synonymously, physically realized) camera-head with conventional artificial neural net (ANN) architectures has been systematically studied by Dean et al. (1991). Mayhew et al. (1992) report implementation of a layered control system for a four-degrees-of-freedom stereo camera-head utilizing their Parametrized Interpolating Look-Up Table (PILUT) architecture.

Inspired by Dean et al. (1991) we have chosen a similar approach to adaptive saccade control but selected a locally linear approximation scheme based on Dynamical Cell Structures (DCS) (Bruske et al. 1995) for control of our (physical) binocular camera-head since this type of ANN ideally meets the demands of the control task: First, the calculation of the controller output has to be as fast as possible to allow control at video rate. Incrementally growing DCS meet this requirement by (i) growing the network only as large as necessary to meet a prespecified precision and, furthermore, by (ii) utilizing only a small subset of its neural units for output calculation. Second, in both the human and

Correspondence to: J. Bruske (Tel: +49 431 560480, Fax: +49 431 560481, e-mail: jbr@informatik.uni-kiel.de)

[1] In robotics, the term ‘inverse kinematics’ of, for example, a robot arm refers to the problem of determining the joint angles required to move the end effector to a certain (cartesian) position.

our artificial vision system the angular rotations around the horizontal viewing, pan and tilt axes (cf. Fig. 1) required to fixate are a linear function of the retinal x and y coordinates of the target only in the case of a restricted field of view (so that the difference between a planar and a spherical retina is not important and the small angle approximation holds) and a small initial tilt angle. With increasing tilt the optical axes of the cameras and the pan axis are no longer perpendicular and the nonlinearity of the control law increases. Hence when using an approximation scheme based on locally linear approximation, the density of neural units should be high in regions of the input space with a high tilt component. Contrary to Kohonen-type networks or PILUT, growing DCS are able to achieve this by allocating new neural units in regions of the input space where the approximation error is high. This is exactly the case for regions of the input space deviating from the linear control law.

Finally, due to physical constraints phase space trajectories of multidimensional systems such as the camera-head system usually lie on submanifolds that locally may be of very much lower intrinsic dimensionality than the input space of the system. DCS just attempt a similar reduction in dimensionality in that they place units in the input submanifold and adapt their lateral connection structure towards an optimally topology-preserving map that is utilized for improved adaptation and approximation.

The remainder of this paper is organized as follows. In Sect. 2 we will first relate our adaptive saccade controller to saccade control in humans and other primates, i.e., explain the biological motivation. We will then become more technical and in Sect. 3 have a closer look at the kinematics and inverse kinematics for calibration-dependent model-based control of the camera-head. Insights from this analysis are used in the design of the DCS-based adaptive saccade controller in Sect. 5 as well as for the discussion of the experimental results in Sect. 6. DCS are briefly introduced in Sect. 4. In Sect. 7 we relate the paper to previous work and in Sect. 8 we provide a final summary.

2 Relation to primate saccade control

Our work bears relation to biology in three respects. First, we try to solve a similar problem under similar constraints that the human visual system has already solved, i.e., the problem of fast and accurate saccade control. Second, we use error feedback learning for solving the problem – a learning mechanism well grounded in neurobiology (Kawato 1995) and probably used for maintaining accurate saccade control in the human visual system (Dean et al. 1994). Finally, we make use of artificial neural networks, which are biologically inspired as well. We now discuss these issues in more detail.

2.1 Requirements for saccades in artificial and biological systems

In biology, saccades denote the fast movements of the eyes that are used to bring a new part of the visual field to the foveal region (Carpenter 1988). They have to be as accurate as possible straight away since correctional saccades take additional time. Because they are so fast, external feedback is inappropriate for control because of the long latency of such feedback loops. Since this is exactly the problem we are trying to solve for our artificial camera-head, we have termed it 'saccade control problem', although we additionally control the pan χ and tilt ϕ of the head, (cf. Fig. 1). Interestingly, similar restrictions apply to both the biological and our artificial system: Human eyes as well as our cameras can move with a velocity of up to 1000 deg/s. In our artificial vision system the delay for visual feedback is about 40 ms, due to image processing and data transfer among various buses. The human visual system has a latency of at least 120 ms (Carpenter 1988). A major challenge for both human saccades and control of our cameras is that the required change in eye position depends not only on the visual (retinal) coordinates of the target but also on the initial head and eye position, if the optical axes of the cameras are not initially perpendicular to both the pan and tilt axes.

2.2 Adaptive saccade control by error feedback learning

Although human saccades are generally thought to be *preprogrammed* or *ballistic* (Carpenter 1988), this only means that once initiated they cannot be modified in flight. It does not answer the question whether the necessary activation patterns are genetically preprogrammed or learned. Indeed, there are a number of lines of evidence pointing to the importance of learning (Dean et al. 1994). Human infants, for example, produce hypometric saccades that become accurate during the first year (Aslin 1987); adults adapt to the effects of eye muscle weakening (Zee and Optican 1985); and subjects under laboratory conditions come to anticipate the surreptitious movement of a visual target during the saccade to it (Deubel et al. 1986). Based on these and similar findings Dean et al. put forward their model of brainstem-cerebellar interactions for learning and maintaining saccadic accuracy based on Kawato's principle of feedback error learning. The model is similar to Kawato's models for adaptive modification of the vestibulo-ocular reflex and ocular-following response (Kawato 1995). We will now briefly review this model of human adaptive saccade control and then relate it to our artificial saccade control system.

As pointed out by Dean et al., typical models of saccade control contain a variant of the internal feedback pulse-generator proposed by Robinson (1975). Such a simplified model (Dean et al. 1994) comprises a simple *feedback controller* converting target coordinates in visual coordinates to a desired change in eye position. This desired change in eye position is then passed to a *pulse or burst generator* that outputs a velocity command to the oculomotor neurons. An *internal feedback loop with a resettable integrator* integrates the pulses (velocity command) of the burst generator and subtracts this estimate of the current eye displacement from the desired change in eye position (hence stopping the burst generator when input is below a threshold). Finally, a *further integrator* also integrates the velocity signal to provide a steady-state position command to the motor neurons that is necessary to maintain the new eye position. While such a model can account for a large amount of data about the behavior and physiology of the saccadic control system in monkeys and humans (Dean et al. 1994), it cannot learn to

make accurate saccades nor does it include information concerning the eye position at the start of the saccade. Dean et al. therefore extended this model by an *aptive controller* that receives input from both the feedback controller (about the visual coordinates of the target) and the oculomotor neurons (about the initial eye position). The output of this additional controller alters the gain in the internal feedback loop, thereby controlling saccade generation. Utilizing error feedback learning, the controller is adapted by an error signal also emanating from the feedback controller.

While it has been suggested that the burst generator is comprised of groups of neurons in midbrain and pons, and it is assumed that the functionality of the simple feedback controller is located in the superior colliculus, Dean et al. have hypothesized that the adaptive controller is located in the posterior vermis. In their model, grounded on anatomical and physiological evidence, the information about the visual coordinates of the target is first projected to the contralateral nucleus recticularis tegmenti pontis which in turn is connected with the posterior vermis through a heavy mossy fiber projection. The source of the signal about the initial eye position is not entirely clear, but the authors quote evidence that it is the nucleus prepositus hypoglossi that projects to the posterior vermis via mossy fibers. Finally, the error signal is thought to be provided by climbing fibers that arise from a specific subregion of the inferior olive, which itself is the target of a projection from the superior colliculus.

In contrast to Dean et al. (1994), our adaptive saccade controller is not intended to imitate or model biology in the first place but to control a particular electromechanical device – the TRC camera head. Nevertheless our controller has a number of structural parallels with its biological counterpart, mainly the assumed overall structure composed of a simple feedback controller and an adaptive inverse model, the latter being adapted by an error signal provided by the feedback controller (cf. Fig. 4). Also there is a strict separation between the visual coordinates of the target and the initial eye and head positions. As in the biological model, the simple feedback controller is provided with the visual target information only, whereas the adaptive inverse model receives information about the initial positions as well. The main functional difference compared with the biological model is that our adaptive controller does not control saccades indirectly by modifying an internal feedback loop but rather directly by generating the input for the internal controllers of the camera-head.

2.3 Biological plausibility of DCS

In spite of being initially inspired by biology, the standard ANN models [Multi-Layer Perceptron (MLP) and Radial Basis Function (RBF) networks] certainly have little to do with biological neural networks, at least on the neuroanatomical level. They represent nonlinear function approximators, which turn out to work very well, both in theory and in practice. ANN are capable of learning and hence can be considered models of higher-level brain functions associated with cognitive capabilities such as learning and generalization that are closely related to function approximation. The DCS we have used in this paper are RBF networks with an additional lateral connection structure formed by Hebbian learning. Poggio (1990) has outlined his 'theory of how the brain might work' by means of HyperBF[2] networks. On a functional level he suggests that *the brain uses modules for multivariate function approximation as basic components of several of its information processing subsystems and that these modules are released by HyperBF networks*. He also indicates how these networks can be implemented in terms of biologically plausible mechanisms and circuitry: the cerebellum is proposed to consist of a set of approximation modules, the granule cells to correspond to the basis functions that receive input via mossy fibers, the Purkinje cells to correspond to the output units that sum the weighted activities of the basis functions and the climbing fibers to provide a teacher signal. Within this theory the adaptive saccade controller would be implemented by a HyperBF network, similar to our DCS-based adaptive controller. Martinetz et al. (1994) comment on a possible analogy between their Topology Representing Networks (TRNs) and the architecture of biological neural networks, which would enable the network to represent the neighborhood and topological relations between features just as in DCS[3]. Yet to our knowledge neurobiological evidence for this analogy is lacking.

3 The camera-head

Let us now introduce our binocular camera system, the TRC[4] BiSight. We first describe the imaging geometry of the system and then report results concerning the inverse kinematics for model-based control. Finally, we discuss the implications of the inverse kinematics for calibration-free adaptive control. Besides the binocular case we also treat the simpler monocular case.

3.1 The TRC BiSight camera system

The TRC BiSight (Fig. 1) has four mechanical degrees of freedom, given by the pan angle χ, the tilt angle ϕ and the horizontal viewing directions (h.v.d.s) θ_1 and θ_r for the left and right camera, respectively. The maximum control values are $\pm 160^\circ$ for pan, $\pm 90^\circ$ for tilt and $\pm 45^\circ$ for the h.v.d.s. The precision is $\pm 0.0225^\circ$ for pan and tilt and $\pm 0.006^\circ$ for the h.v.d.s. The CCD cameras are mounted on the tilt axes and the length of the stereo baseline is 25 cm. Each camera has three optical degrees of freedom: the zoom (11.5–69 mm), the focus (50cm–∞) and the aperture. The image size of each camera is 512^2 pixels.

3.2 Geometry of the camera-head

For the configuration of the TRC BiSight, the mapping from a point $\bar{p}_w = (x_w, y_w, z_w, 1)$ in homogeneous world coordinates to a point $\bar{p}_c = (x_c, y_c, z_c, 1)$ in camera coordinates is

[2] Poggio's HyperBFs are just generalized RBFs using elliptical instead of radially symmetrical weighting functions.

[3] According to the definition of TRN (Martinetz et al. 1994), DCS can be classified as TRNs with a RBF associated with each node in the graph of the TRN.

[4] Transitions Research Corporation, Danbury, Connecticut, USA.

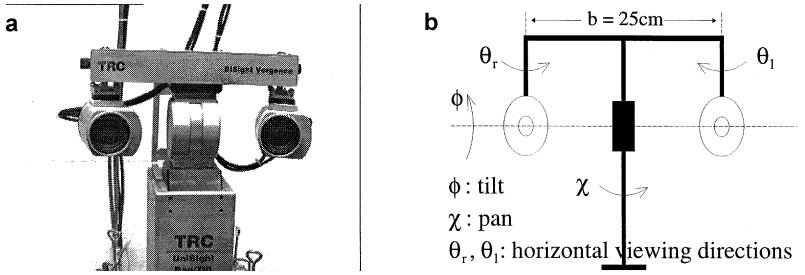

Fig. 1. a The TRC BiSight binocular head. **b** Diagram of the TRC head

given by

$$\bar{p}_{c_{l,r}} = T_{\theta_{l,r}}^{-1} T_{\phi}^{-1} T_{\chi}^{-1} \bar{p}_w \tag{1}$$

l and r indicating the left and the right camera, respectively, with transformation matrices

$$T_{\chi} = \begin{bmatrix} \cos\chi & 0 & -\sin\chi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\chi & 0 & \cos\chi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$T_{\phi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

$$T_{\theta_l} = \begin{bmatrix} \cos\theta_l & 0 & \sin\theta_l & -b/2 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_l & 0 & \cos\theta_l & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and T_{θ_r} analogous to T_{θ_l} (changed index and reversed sign of translation and rotation). Here, we assume the origin of the world coordinate system to be at the point of intersection of the pan and tilt axes, and the origins of the camera coordinate systems to be at the point of intersection of the tilt axis with the corresponding view axis.

The mapping from a point $\bar{p}_c = (x_c, y_c, z_c, 1)$ in camera coordinates to image coordinates $\bar{p}_i = (x_i, y_i, 1)$ is given by

$$\bar{p}_i = P_c \left(\frac{1}{z_c} \bar{p}_c \right), \text{with } P_c = \begin{bmatrix} \alpha_{xc} & 0 & x_{0c} & 0 \\ 0 & \alpha_{yc} & y_{0c} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

The parameters $\alpha_{xc}, \alpha_{yc}, x_{0c}, y_{0c}$ are the *intrinsic* parameters of the cameras and have to be carefully determined by a calibration procedure as described in, for example, Tsai (1986) or Faugeras (1993).

3.3 Inverse kinematics

Knowing the geometry one can solve the inverse kinematics for both the monocular and the binocular system. In order to make the solution of the inverse problem unambiguous we apply the following constraints:

- for the monocular system we require the pan axis to be fixed,
- for the binocular system we require the two cameras to verge symmetrically, i.e., $\theta_l = \theta_r = \theta_{l,r}$

Solving (1)–(3) for the required angles (necessary to fixate the target given the current angles and retinal coordinates) is now straightforward[5] and yields the model-based inverse kinematics for monocular and binocular fixation, respectively:

$$\bar{\omega}^* = (\phi^*, \theta^*) = f_{\text{direct}}^{\text{monocular}}(x, y, \theta, \phi) \quad \text{(monocular case)} \tag{4}$$

$$\bar{\omega}^* = (\chi^*, \phi^*, \theta^*_{l,r}) = f_{\text{direct}}^{\text{binocular}}(x_l, y_l, x_r, y_r, \theta_{l,r}, \phi, \chi) \quad \text{(binocular case)} \tag{5}$$

It is, however, advantageous not to calculate the new angles $\bar{\omega}^*$ directly because (after some manipulations) it turns out that the change in angles $\Delta\bar{\omega}$ is independent of the tilt in the monocular case and of the pan in the binocular case. The inverse kinematics now take the form

$$\Delta\bar{\omega} = (\Delta\phi, \Delta\theta) = f_{\text{direct}}^{\text{monocular}}(x, y, \theta) \quad \text{(monocular case)} \tag{6}$$

$$\Delta\bar{\omega} = (\Delta\chi, \Delta\phi, \Delta\theta_{l,r}) = f_{\text{direct}}^{\text{binocular}}(x_l, y_l, x_r, y_r, \theta_{l,r}, \phi) \quad \text{(binocular case)} \tag{7}$$

3.4 Discussion

Having derived a model of the inverse kinematics of the camera-head, one can use it for model-based saccade control. However, one must bear in mind the central assumptions behind this:

- tilt and view axes intersect at the optical center,
- the CCD chip of the camera has been accurately placed by the manufacturer,
- the base length is known exactly, and
- the intrinsic parameters have been determined with high accuracy.

The first assumption is valid only if the system is well manufactured. If tilt and view axes did not intersect at the optical center the complexity of the model would increase significantly. The second assumption is only approximately valid, since the CCD chips are usually slightly rotated around the optical axis of the cameras. The model could, however, be extended to cope with these rotations at the cost of introducing yet another set of intrinsic parameters. Finally, to determine the intrinsic parameters (and perhaps the base length as well) one must employ a carefully designed calibration method. Yet any change in the intrinsic parameters (e.g., by zooming), change of the base length or wear and

[5] Since the rather lengthy formulae for the following functions and their derivation with help of the MAPLE tool set do not contribute to the understanding of the article and our emphasis is on adaptive rather than model-based control, they have been omitted here.

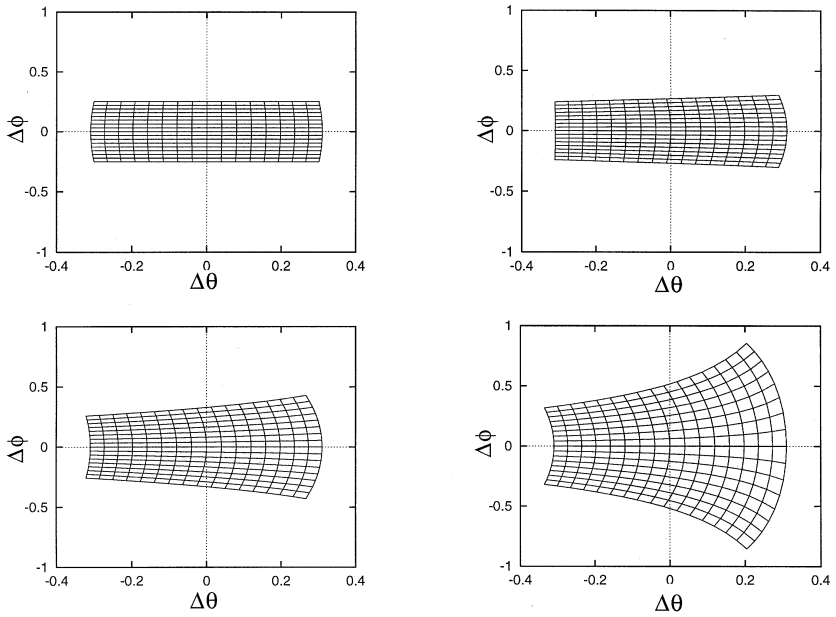

Fig. 2. Increasing nonlinearity of the computed mapping from image coordinates to changes in tilt and horizontal viewing direction (h.v.d.), $(x_i, y_i)|_{\phi,\theta} \rightarrow (\Delta\phi, \Delta\theta)$, for varying initial h.v.d. θ and constant tilt $\phi = 0°$. The figures show the mapping of the retinal (x_i, y_i) grid to the control values $(\Delta\phi, \Delta\theta)$ for $\theta = 0°$ (*top left*), $\theta = 20°$ (*top right*), $\theta = 40°$ (*bottom left*) and $\theta = 60°$ (*bottom right*)

tear of the internal controllers of the camera-head requires a new calibration.

Hence even in the presence of a perfect model, calibration-free adaptive control can be advantageous, provided the adaptive controller is flexible enough to reach the required precision. The analysis of the system then sheds some light on the kind of controller one should employ. In the monocular case the controller need not know the previous tilt angle, ϕ (6). Furthermore, the inverse model has a strong linear component for small initial h.v.d., i.e., when the optical axis of the camera is nearly perpendicular to the tilt axis. This is illustrated in Fig. 2, showing that for small initial h.v.d. θ the mapping of (x_i, y_i, θ) to $(\Delta\phi, \Delta\theta)$ is nearly linear (top left) while becoming increasingly nonlinear with increasing initial h.v.d. Analysis of the binocular case reveals independence of pan (7), and the mapping from $(x_l, y_l, x_r, y_r, \theta_{l,r}, \phi)$ to $(\Delta\chi, \Delta\phi, \Delta\theta_{l,r})$ here becomes increasingly nonlinear with respect to both initial h.v.d. and tilt. The nonlinear dependency on the tilt is due to the fact that the optical axes of the cameras are no longer perpendicular to the pan axes for tilt angles different from zero. An adaptive controller can exploit this, and our DCS-based controller does so: it learns a locally linear approximation of the actual inverse kinematics and will place more nodes in regions of increasing nonlinearity. The latter is an intrinsic property of DCS and, as we demonstrate in Sect. 6, leads to a distribution of nodes aimed at minimizing the approximation error (and not just reflecting the input probability density, as, e.g., in the case of the Kohonen feature map).

4 Dynamic Cell Structures

DCS, as introduced in Bruske et al. (1995), denote a class of approximation schemes that are based on RBFs and attempt to learn and utilize optimally topology preserving feature maps (OTPMs).[6]

The architectural characteristics of DCS (Fig. 3) are

- one hidden layer of RBF units,
- a dynamic lateral connection structure between these units, and
- a layer of (usually linear) output units.

Training algorithms for DCS rest on adapting the lateral connection structure towards an OTPM by employing a competitive Hebbian learning rule and activating and adapting RBF units only in the neighborhood of the current stimulus, where 'neighborhood' relates to the simultaneously learned topology.

We use a normalized RBF approximation scheme and hence the output of a DCS network calculates as[7]

$$\bar{y}(\bar{x}) = \frac{\sum_{i \in \mathrm{Nh}^+(\mathrm{bmu}_{\bar{x}})} \bar{o}^i \mathrm{rbf}((\bar{x} - \bar{c}^i)^2)}{\sum_{i \in \mathrm{Nh}^+(\mathrm{bmu}_{\bar{x}})} \mathrm{rbf}((\bar{x} - \bar{c}^i)^2)} \tag{8}$$

where $\mathrm{rbf}((\bar{x} - \bar{c}^i)^2)$ denotes an RBF with center $\bar{c}^i$. The vectors $\bar{o}^i$ can be thought of as output weight vectors attached to each rbf unit. The activation function rbf: $\mathscr{R}^+ \rightarrow \mathscr{R}^+$ is strictly monotonically decreasing with $\mathrm{rbf}(0) = 1.0$ and $\mathrm{rbf}(\infty) = 0$. In the experiments reported in this article the activation function has been realized by a rational function, $\mathrm{rbf}(x) = 1/(1 + \sigma x^2)$, with fixed σ. With the lateral connection structure between the RBF units being represented by an adjacency matrix C, the neighborhood $\mathrm{Nh}^+(j)$ of neural unit j is defined as the unit itself together with its direct topological neighbors $\mathrm{Nh}(j)$:

[6] In contrast to the introduction in Bruske et al. (1995), we here no longer require DCS to learn a perfectly topology-preserving map, since perfect topology preservation as defined in Martinetz et al. (1994) can only be checked for if prior knowledge of the data distribution, i.e., knowledge of the 'density' condition in theorem 3 in Martinetz et al. (1994), is given, which is not the case here.

[7] In the following we denote vectors by $\bar{x}$, components of a vector by $\bar{x}_i$ and enumerations of vectors by $\bar{x}^i$.

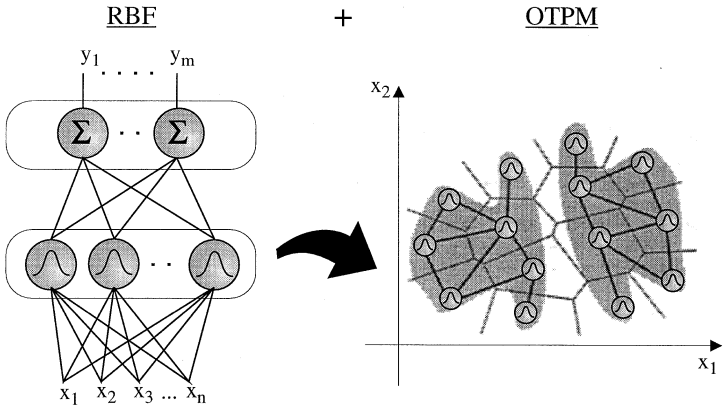

Fig. 3. Dynamic cell structures (DCS) are Radial Basis Function (RBF) networks (*left*) plus lateral connection structure between the RBF units attempting to build an optimally topology preserving map (OTPM) by competitive Hebbian learning (*right*; closely modified from Fig. 5d of Martinetz et al. 1994)

$$\mathrm{Nh}^{+}(j) = \mathrm{Nh}(j) \cup j = \{i | C_{ij} \neq 0, 1 \leq i \leq N\} \cup j \tag{9}$$

and the best matching unit $\mathrm{bmu}_{\vec{x}}$ is given by

$$(\vec{c}^{\,\mathrm{bmu}} - \vec{x})^2 \leq (\vec{c}^{\,i} - \vec{x})^2, (1 \leq i \leq N) \tag{10}$$

The adjacency matrix C is adapted by a competitive Hebbian learning rule

$$\Delta C_{ij} = \begin{cases} 1 : y_i \cdot y_j \geq y_k \cdot y_r \forall (1 \leq k, r \leq N) \\ 0 : \text{otherwise} \end{cases} \tag{11}$$

with $y_i = \mathrm{rbf}((\vec{x} - \vec{c}^{\,i})^2)$. Given the training set T (either explicitly in the form of a number of samples or implicitly via a density function on the input space) the lateral connection structure converges to the OTPM of the set of centers $S = \{\vec{c}^{\,1}, \ldots, \vec{c}^{\,N}\}$ with probability 1, with the OTPM of S given T defined as the graph (adjacency matrix C) with

$$C_{ij} \neq 0 \Leftrightarrow \exists x \in T \; \forall 1 \leq k \leq N, k \notin \{i, j\} : \\ \max\{(\vec{x} - \vec{c}^{\,i})^2, (\vec{x} - \vec{c}^{\,j})^2\} \leq (\vec{x} - \vec{c}^{\,k})^2 \tag{12}$$

The importance (and naming) of OTPMs stems from the fact that they are optimally topology preserving in the sense of the topographic function introduced in Villmann et al. (1994) for measuring the degree of topology preservation. Compared with the usual RBF approximation, learning and exploiting the topology of the input space not only speeds up the computation of the output value (because only a small number of neural units need to be evaluated in each step) but can also improve approximation quality (because in the usual RBF scheme units whose centers have small Euclidean distance to the stimulus but large intra-manifold distance to the point of projection of the stimulus onto the manifold can impair the approximation fidelity).

In order to adapt the output vectors $\vec{o}^{\,i}$, we employ gradient descent on an error function $E(\vec{y})$, (see Sect. 5.2.1 for gradient-based output layer adaptation in conjunction with error feedback learning). The centers $\vec{c}^{\,i}$ can be adapted by gradient descent as well but may alternatively be trained by a Kohonen-type learning rule utilizing the lateral connection structure. Both these center adaptation rules are investigated in the context of our application in Sect. 5.2.2.

Growing DCS are realized by inserting additional RBF units in regions of the input space where the approximation performance is unsatisfactory (hence increasing the resolution in this region). This is achieved by attaching an additional error variable (*resource value*) to each neural unit that monitors the performance of the network when this unit is involved in output calculation. In Sect. 5.2.3 we will show how the fixation error is used to update the error variables and to control growth of the network for adaptive saccade control. As already mentioned, and central to our application, the *error-driven insertion* of new units has the advantage that the distribution of neural units attempts to minimize the error (and does not merely reflect the input probability density, as would be the case with a Kohonen-type rule). See also Fritzke (1995a) for more on incremental growth of RBF networks utilizing resource values.

While error-driven refinement of approximation schemes works very well in practice, it is an open question whether biological systems use similar mechanisms. All that is known is that in the adult brain no new neurons are available.

Finally, we want to point out the close connection of RBF networks and DCS with Sugeno-type fuzzy logic controllers (for details see Bruske et al. 1996a). Here, the main idea is that each RBF node with its attached output vector may be viewed as a Sugeno-type fuzzy rule, and (8) just implements the Sugeno fuzzy inference rule. This allows incorporation of prior knowledge as well as analysis of DCS in terms of fuzzy logic.

5 DCS for adaptive saccade control

Having investigated the kinematics of our camera-head and outlined the ideas behind DCS, we now describe our feedback error learning scheme for adaptive saccade control. The actual training of the system will proceed in two stages. First, we use a crude simulation of the cameras and the kinematics of the head to learn saccade control up to a predefined precision off-line. In the second phase, the operating phase, we continue to adapt the output layer of the DCS network on-line to cope with deviations of the physical camera-head from the simulated one and deviations due to changing parameters. The first phase could be on-line as well but simulation has the advantage that training takes less time (no real head movement) and that, in the second phase, the physical system is under reasonable control right from the start. Stopping center adaptation in the operating phase also avoids potential instabilities that may arise due to the sensitivity of the center adaptation rules to a possibly changing, nonstationary input probability density in the operating phase [cf.

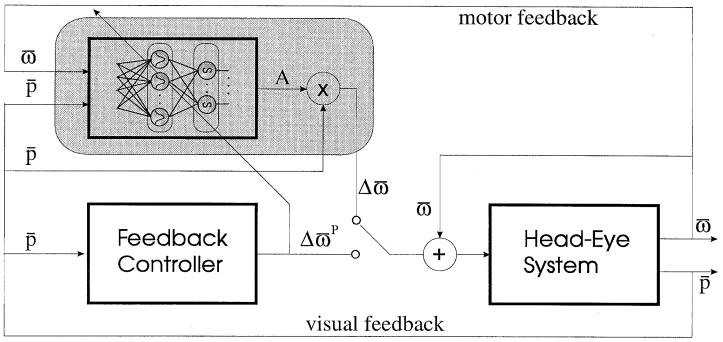

Fig. 4. DCS-based adaptive saccade control. The DCS network associates an input vector of angles and retinal coordinates $(\bar{\omega}, \bar{p})$ with a (Jacobian) matrix A, which is then multiplied by the retinal coordinate vector $\bar{p}$ to yield a change in angles $\Delta\bar{\omega}$. The control value is the sum of this change plus the old angles, $\Delta\bar{\omega} + \bar{\omega}$. After fixation, the retinal coordinates of the target point are transformed into a further change in angles $\Delta\bar{\omega}^p$ by the proportional feedback controller. This serves as an error signal for the DCS network and can also be used for a correctional saccade

Ritter and Schulten (1986) for an analysis of the center distribution as a function of the input probability density in the context of the Kohonen rule].

5.1 Controller output calculation

Exploiting the strong linear component of the saccade control problem (see Sect. 3.4) we do not directly associate the input vector of retinal coordinates and angles $\bar{u} = (\bar{p}, \bar{\omega})$ with an output vector but with a (Jacobian) matrix A. This matrix is then multiplied by the vector of retinal coordinates $\bar{p}$ to yield the change in angles. Hence we approximate the inverse kinematics with a locally linear mapping, interpolating between the linear transforms A^i attached to the nodes of our DCS network. See Fig. 4 for illustration.

More specifically, for the binocular case we use a five-dimensional input vector, $\bar{u} = (x_l, y_l, x_r, \theta_{l,r}, \phi)$, which we associate with a 3×3 (Jacobian) matrix A, the output of the DCS network. Here, x_l, y_l, x_r denote the retinal (image) coordinates of the target on the left and right camera, $\theta_{l,r}$ are the h.v.d.s of the left and right camera and ϕ is the tilt angle of the camera-head. The output of the controller $\Delta\bar{\omega}$ is calculated as

$$\Delta\bar{\omega} = (\Delta\chi, \Delta\phi, \Delta\theta_{l,r}) = A\bar{p} \tag{13}$$

here $\Delta\chi, \Delta\phi, \Delta\theta_{l,r}$ denote the changes (rotations) in pan, tilt and h.v.d.s necessary to fixate the target, and $\bar{p} = (x_l, y_l, x_r)$ is the retinal coordinate vector. Recall that because of symmetrical vergence $\theta_l = \theta_r$ and that $\Delta\bar{\omega}$ does not depend on χ. Further, we exploit the fact that the four retinal coordinates x_l, y_l, x_r, y_r are not independent[8] to exclude y_r from both the input $\bar{u}$ and the retinal coordinate vector $\bar{p}$.

In the monocular case, we use a three-dimensional input vector, $\bar{u} = (x, y, \theta)$, which we associate with a 2×2 matrix A. With retinal coordinates $\bar{p} = (x, y)$ the output of the controller then calculates as

$$\Delta\bar{\omega} = (\Delta\phi, \Delta\theta_1) = A\bar{p} \tag{14}$$

Utilizing a DCS network, the matrix A is computed as a normalized weighted sum of the matrices A^i attached to the rbf units of the DCS network,

$$A = \sum_{i \in \mathrm{Nh}^+(\mathrm{bmu})} A^i h_i \text{ with } h_i = \frac{\mathrm{rbf}((\bar{u} - \bar{c}^i)^2)}{\sum_{j \in \mathrm{Nh}^+(\mathrm{bmu})} \mathrm{rbf}((\bar{u} - \bar{c}^j)^2)} \tag{15}$$

[8] The target point is uniquely determined as the intersection of the two viewing rays defined by (x_l, y_l) and (x_r, y_r), respectively. It is, however, uniquely determined also as the intersection of the viewing plane defined by x_r and the viewing ray defined by (x_l, y_l).

5.2 Feedback error learning with DCS

After fixation we use the output of a simple proportional feedback controller to adapt the parameters of the DCS network. This is the principle of feedback error learning, which is illustrated in Fig. 5 and served as the design principle of our adaptive saccade controller (Fig. 4). Advantageous properties of feedback error learning of inverse kinematics are (i) that in contrast to direct inverse modeling, feedback error learning does not suffer from the problem that the global optimal solution (with respect to the error function used for training) is not necessarily a correct inverse model [see Jordan and Rumelhart (1992) for the 'convexity problem' as an example] and (ii) simplicity, i.e., in contrast to, for example, forward-and-inverse modeling no additional backpropagation through a forward model is necessary. A detailed discussion of this and related learning paradigms is beyond the scope of this article, and the reader is referred to Kawato (1990) and Jordan and Rumelhart (1992).

The drawback of error feedback learning is that difficult problems such as the inverse kinematics of a robot arm with redundant degrees of freedom may require quite sophisticated feedback controllers, yet in our case of symmetric fixation with a binocular camera-head, a proportional feedback controller suffices. Its output $\Delta\bar{\omega}^p = (\Delta\chi^p, \Delta\phi^p, \Delta\theta^p_{l,r})$ is proportional to the mean target's retinal x-coordinates, the left camera's y-coordinate and the difference in the x-coordinates after fixation:

$$(\Delta\chi^p, \Delta\phi^p, \Delta\theta^p_{l,r}) = (k_1(x_l + x_r), k_2 y_1, k_3(x_l - x_r)) \tag{16}$$

with $k_1, \ldots, k_3$ the gain factors of the proportional controller. If fixation is perfect, $\Delta\bar{\omega}^p$ will be the null vector.

For the monocular case, the output of the feedback controller is $\Delta\bar{\omega}^p = (\Delta\phi^p, \Delta\theta^p)$ and the components are simply proportional to the y-coordinate and the x-coordinates of the camera after fixation:

$$(\Delta\phi^p, \Delta\theta^p) = (k_1 y, k_2 x) \tag{17}$$

Note that for both the binocular and the monocular case the control laws (16) and (17) suffice to fixate. The reason for training an additional inverse model is just to accelerate fixation, i.e., to fixate in one step without relying on feedback.

5.2.1 Output layer adaptation. In order to adapt the output layer of our DCS network, i.e., the matrices A^i, we have to translate the feedback error $\Delta\bar{\omega}^p$ into a difference between matrices. The feedback error $\Delta\bar{\omega}^p$ indicates that

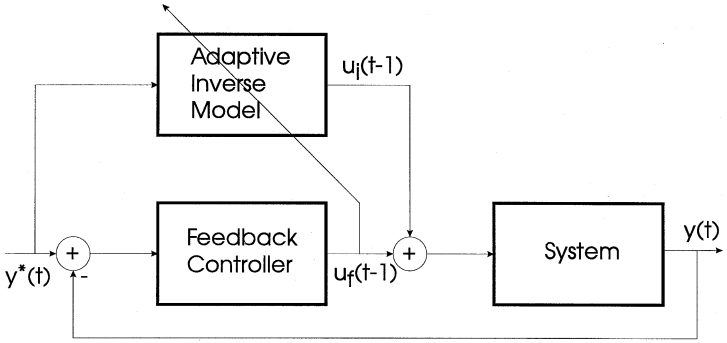

Fig. 5. Feedback error learning

$$\Delta\bar{\omega}^* = \Delta\bar{\omega} + \Delta\bar{\omega}^p = A\bar{p} + \Delta\bar{\omega}^p \tag{18}$$

is a better output of the controller. This output would have been obtained by a matrix A^* with

$$A^* = A + \Delta\bar{\omega}^p \frac{\vec{p}^T}{\|\bar{p}\|^2} \tag{19}$$

as is easily checked by multiplying (19) by $\bar{p}$. Using A^* as the desired output of our DCS network we can define a difference matrix

$$\Delta A = A^* - A = \Delta\bar{\omega}^p \frac{\vec{p}^T}{\|\bar{p}\|^2} \tag{20}$$

and can use the following α-LMS (least mean square) rule for adapting the output layer, with $0 < \alpha < 1$:

$$\Delta A^i = \alpha \left(\Delta\bar{\omega}^p \frac{\vec{p}^T}{\|\bar{p}\|^2} \right) h_i, i \in \mathrm{Nh}^+(\mathrm{bmu}) \tag{21}$$

implementing a gradient descent on $(\Delta A)^2$. We set $\alpha = 0.7$ in all the following experiments.[9]

5.2.2 Center adaptation. For center adaptation we report experiments with two different learning rules introduced in the following. An evaluation of these learning rules in the context of our learning tasks is provided in Sect. 6.1.

– The well-known Kohonen-type learning rule (Kohonen 1987) used, for example, in Fritzke (1995a) and Bruske et al. (1995):

$$\Delta\vec{c}^i = \varepsilon_i(\bar{u} - \vec{c}^i), \quad i \in \mathrm{Nh}^+(\mathrm{bmu}) \tag{22}$$

– A gradient-modulated learning rule with centers restricted to moving in the direction of the negative gradient:[10]

$$\Delta\vec{c}^i = \begin{cases} \varepsilon_i(\bar{u} - \vec{c}^i): & \frac{\partial}{\partial \mathrm{rbf}_i}(\Delta A)^2 > 0 \\ 0 \quad : & \text{otherwise} \end{cases}, i \in \mathrm{Nh}^+(\mathrm{bmu}) \tag{23}$$

with

$$\frac{\partial}{\partial \mathrm{rbf}_i}(\Delta A)^2 = \nabla_A(\Delta A)^2 \cdot \frac{\partial}{\partial \mathrm{rbf}_i} A = \sum_{\mathrm{kl}} \left(\Delta\bar{\omega}^p \frac{\vec{p}^T}{\|\bar{p}\|^2} \right)_{\mathrm{kl}} \left(\frac{A_i - A}{\sum_{j \in \mathrm{Nh}^+(\mathrm{bmu})} \mathrm{rbf}_j} \right)_{\mathrm{kl}} \tag{24}$$

In the above learning rules, the ε_i denote learning constants. We have $\varepsilon_i = \varepsilon_{\mathrm{bmu}} = 0.05$ for the bmu and $\varepsilon_i = \varepsilon_{\mathrm{Nh}} = 0.006$ for the direct neighbors of the bmu.

The Kohonen-type learning rule (22) is simpler but has the disadvantage that the distribution of neural units does not depend on the output error of the network but on the input probabilitiy density only. Although frequently used in regression, classification and control tasks it is a very poor learning rule for those tasks because of this missing error sensitivity. Its use can only be justified if the approximation error is correlated with input probability density (e.g., uniform approximation error and uniform input probability density) or if there is an additional mechanism taking care of an error-dependent distribution of units (like our error-driven insertion strategy: see below).

A gradient-based center adaptation rule promises to be the more appropriate choice because it aims directly at minimizing the expected output error. The additional error-driven insertion strategy of neural units helps to alleviate the well-known problem of local minima for this learning rule (by 'globally' inserting units where the local approximation error is largest). Furthermore, gradient calculation is fast in DCS networks even in implementations on serial hardware because it is restricted to the bmu and its neighbors. Yet in conjunction with RBF networks, gradient descent is frequently observed to move neural units out of the input manifold (since a local optimum may be reached in this way), and the same is true for DCS. This is the motivation for considering the gradient-modulated learning rule (23), where the stimulus attracts the unit just as in the case of the Kohonen rule but only if the movement is along the negative gradient. Hence units stay in the input manifold and only move in directions of decreasing error.

5.2.3 Node insertion. A new unit is inserted whenever the resource value of the bmu, r_{bmu}, *and* the current fixation error, e_{fix}, exceed the prespecified precision δ:[11]

$$r_{\mathrm{bmu}} > \delta \quad \text{and} \quad e_{\mathrm{fix}} > \delta \tag{25}$$

[9] The parameter setting in this and the following adaptation rules is not critical, i.e., small variations lead to qualitatively similar behavior. Yet a good trade-off between speed of adaption ($\alpha \to 1$) and stability ($\alpha \to 0$) has to be found by experimentation.

[10] Since the gradient of a RBF with respect to the center is always in the direction of the difference between stimulus and center vector, i.e., $\nabla_{\vec{c}}\mathrm{rbf}((\bar{u} - \vec{c})^2) \sim -(\bar{u} - \vec{c})$, (23) does indeed adapt the center vector in the direction of the negative gradient of $(\Delta A)^2$.

[11] In our case, we usually require $\delta = 2.5$ pixels, which is 0.5% of the image size of the cameras (512×512 pixels).

In case of insertion, the resource of the bmu is set to zero, or else it is updated using a floating average rule:

$$r_{\text{bmu}} = \begin{cases} 0 & : \text{ insertion} \\ \beta e_{\text{fix}} + (1-\beta) r_{\text{bmu}} & : \text{ otherwise} \end{cases} \quad (26)$$

Note that this insertion rule is (i) purely local and (ii) purely error driven. Making insertion depend on the best-matching unit's resource, which is a local average of the fixation error, the procedure becomes less sensitive to noise than by relying only on e_{fix}. Setting the resource to zero after insertion and choosing a small averaging constant β we avoid over-frequent insertions in the neighborhood of a single unit.

The output matrix A^{new} of a newly inserted unit is initialized with A^{bmu}, and its center is initialized with the current stimulus, $\bar{c}^{\text{new}} = \bar{u}$. Finally, it is (mutually) connected with the bmu. Between successive insertions we require at least $n \log n$ training steps without insertion,[12] n being the current number of neural units, to allow the lateral connection structure of the DCS network to build an OTPM. Due to this strategy we need a relatively large number of trials for meeting high precision demands. However, in the off-line phase emphasis is on as small a number of neural units and as good an OTPM as possible, since these will no longer be adapted in the on-line phase. Training in the off-line phase stops when the averaged fixation error falls below the prespecified precision.

5.2.4 Topology learning. Since units are allowed to move, the Hebbian learning rule (11) must be modified to allow 'forgetting' of neighborhood relations (lateral connections) that no longer exist. This can easily be achieved by extending (11) with a decay term. However, care has to be taken to avoid 'dead neurons', i.e., units that are disconnected from the remaining ones and are utilized no further. Our solution is a nonsymmetric Hebbian learning rule similar to that proposed in Ahrns et al. (1995) and detailed in Bruske et al. (1996b).

5.3 Off-line training phase

In off-line training we use a (crude) model[13] of the TRC BiSight (and not the system itself) to calculate the new retinal coordinates of the target after applying the controller output. Input vectors are generated randomly. The only constraints are that the h.v.d. θ_l and tilt ϕ are restricted to an interval of interest and that the retinal coordinates x_l, y_l, x_r may not exceed the field of view of the cameras.

Note that we do error feedback learning (i.e. no supervised training) in the off-line phase just as in the following on-line phase. In particular, we do not need the inverse kinematics. Our off-line training phase could actually be on-line as well (with the TRC BiSight instead of the model); the only reasons to keep it off-line are to accelerate training[14] and to have reasonable control in the working phase right from the start.

5.4 On-line training in the operating phase

In the operating phase we continue to adapt the matrices A^i in the output layer by error feedback, just as in the off-line phase. However, we no longer grow the network nor do we adapt the centers or the lateral connection structure any longer.

Targets for fixation are generated by randomly moving our Stäubli R90 robot arm in its workspace with a light source attached to its gripper (Fig. 6). Relying on our Datacube image processing system, we can calculate the retinal coordinates of the target with respect to the two cameras of our TRC binocular camera-head (Fig. 6a) at video rate. The controller output as calculated according to (13) and (15) is then applied to fixate the target, and the output of the proportional controller is used for error feedback learning. The latter is also used for a correctional saccade.

6 Experimental results

In this section we will first investigate the two learning rules introduced in Sect. 5.2.2 for center adaptation. We do this in the context of analytically more simple monocular fixation (see Sect. 3). We then proceed with the better learning rule (the gradient-modulated rule) and apply it for learning binocular fixation. We further demonstrate the effect of on-line training and give an example of how the system adapts to changing camera parameters.

Throughout all the following experiments we use the same set of learning parameters for the DCS network.[15] Inputs have been normalized to the interval $[-1, 1]$. The fixation error is calculated as the mean of the retinal coordinates, i.e.,

$$e_{\text{fix}} = \frac{|x| + |y|}{2} \quad \text{(monocular case)}$$
$$\text{and} \quad (27)$$
$$e_{\text{fix}} = \frac{|x_l| + |x_r| + |y_l|}{3} \quad \text{(binocular case)}$$

Only one retinal y-coordinate is evaluated for error calculation in the binocular case since the y-coordinates cannot be controlled independently. There is only one common tilt joint and in the case of fixation with symmetric vergence $y_l = y_r$.

6.1 Monocular adaptive fixation

Taking the monocular case as a test bed we compare the two learning rules of Sect. 5.2.2, Kohonen-type and gradient-modulated center adaptation. Targets were generated with a uniform h.v.d. distribution in $[-40°, 40°]$ and covered the

[12] This heuristic is motivated by theoretical results concerning the time complexity of building OTPMs as given in Martinetz et al. (1994).

[13] The model is given by (1) and (3) with a coarse estimate of the camera parameters.

[14] Training on 20 000 target points needs less than 5 min off-line compared with 5.5 h on-line (with 1 s for each movement of the robot arm for generating the next target point).

[15] Parameters are α = 0.7 (output layer learning rate), σ = 1.0 (width of basis functions), β = 0.03 (resource averaging constant), ε_{bmu} = 0.05, ε_{Nh} = 0.006 (center adaptation rates).

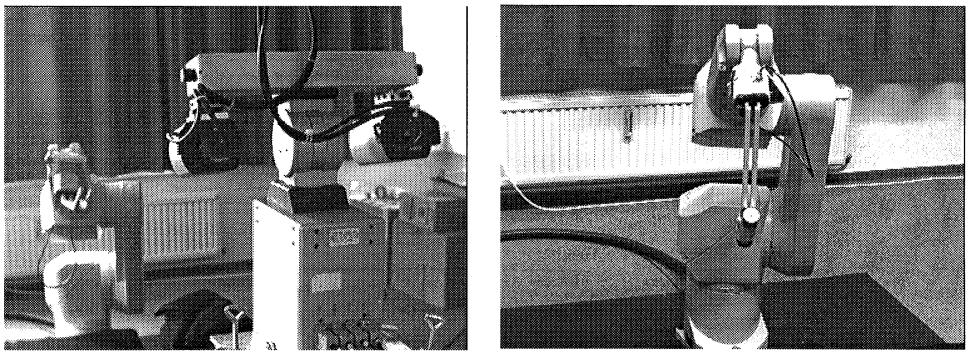

Fig. 6a,b. On-line training with the robot arm. **a** Setup for on-line training. **b** The Stäubli R90 robot arm

Table 1. Averaged number of training steps n and number of allocated neural units N with standard deviations for the 441 two learning rules for different precisions δ

δ	Kohonen-type learning rule	Gradient-based learning rule
3.5 pixels	n : 10542 ± 460	n : 8184 ± 1017
	N : 65 ± 1.3	N : 57 ± 3.7
2.9 pixels	n : 30879 ± 1855	n : 17988 ± 2890
	N : 105 ± 2.9	N : 82 ± 5.9
2.5 pixels	n : 65788 ± 6252	n : 39216 ± 2827
	N : 147 ± 6.0	N : 117 ± 4.0

whole 512×512 pixel retina of the camera. Table 1 shows the number of training steps and the number of allocated neural units for the two learning rules as a function of the required precision (fixation error). Both the number of training steps and the number of neural units are averaged over ten runs. The gradient-modulated rule consistently outperforms the Kohonen-type rule, i.e., it allocates significantly less units and takes nearly half the training time of the Kohonen-type rule. Not surprisingly, the table also implies that the number of units needed to reach a required precision grows superlinearly. Figure 7 shows the distribution of neural units as a function of initial h.v.d. The first row shows the frequency of center insertion as a function of the h.v.d. while the second row shows the number of neural units as a function of the h.v.d. after training. The histograms are averaged over ten runs.

The learning rules show the same qualitative behavior: although targets have been generated with a uniform distribution with respect to the h.v.d., far more neural units are inserted in regions of large h.v.d. than in regions of small h.v.d. (first row). This is in accordance with our expectation based on the analysis of the inverse kinematics for monocular fixation in Sect. 3.3. There we saw that the control law becomes increasingly nonlinear the larger the h.v.d. Hence approximation with a local linear model becomes more difficult for large h.v.d.s, and in order to reach the same precision for all h.v.d.s, the error-driven insertion strategy allocates more units for large angles.

The error-driven distribution of centers is maintained by the learning rules (second row). However, since the Kohonen-type rule follows the probability density, this rule would eventually generate a uniform distribution of units (if center adaptation continued but no new units were inserted). It is the missing error-sensitivity of this rule that is responsible for its non-optimal performance (Table 1). A closer look at the distribution for the Kohonen rule after training (second row in Fig. 7) actually indicates that this flattening is already in effect for small and medium angles. Only the higher insertion rate for large h.v.d.s and the border effect (moving centers from the border towards the center) have prevented the peaks at large angles from flattening.

6.2 Binocular adaptive fixation

The gradient-modulated learning rule turned out to be the more appropriate for binocular asymmetric vergence too, and the results below all refer to this learning rule. The workspace for the off-line phase was $\phi \in [-30°, 30°]$ for tilt and $\theta_{l,r} \in [0°, 10°]$ for the h.v.d.s. Targets were generated with a uniform distribution with respect to the tilt and covered both camera images. The workspace used for training in the off-line phase includes the workspace for on-line training, which was $\phi \in [-14°, 2°]$, $\theta_{l,r} \in [2°, 4°]$ and $\chi \in [-4°, 9°]$. The latter reflects the work space of our robot arm.

For the very high precision of 0.5% (2.5 pixels) we need only 19155 ± 1340 training saccades and 84 ± 2.6 neural units. Again, the numbers denote averages and standard deviations over ten runs. For one of these runs, Fig. 8 shows the course of the pixel error (averaged) and the number of neural units in the DCS network versus the number of training steps. On reaching this precision after 18000 simulated saccades only 82 neural units have been inserted into the network. We could reach this precision with fewer training steps but since we want as few neural units and as good an OTPM as possible, we invest in the off-line phase. After all, the whole off-line phase takes only 3 min for 20 000 saccades on a Sparc 4 workstation. And the very reasonable accuracy of 1% (5 pixels) is reached after 3500 steps with fewer than 40 neural units.

Figure 9 shows the averaged distribution of neural units depending on the tilt angle, at time of insertion (left) and at the end of training (right). Again this distribution is in accordance with expectation: insertion frequency and density of neural units increase with increasing tilt and hence with increasing nonlinearity, in spite of the uniform training target distribution with respect to the tilt.

Now we use the pretrained controller for saccade control of the physical TRC binocular camera-head. As demonstrated in Fig. 10 the error at the start of the operating phase

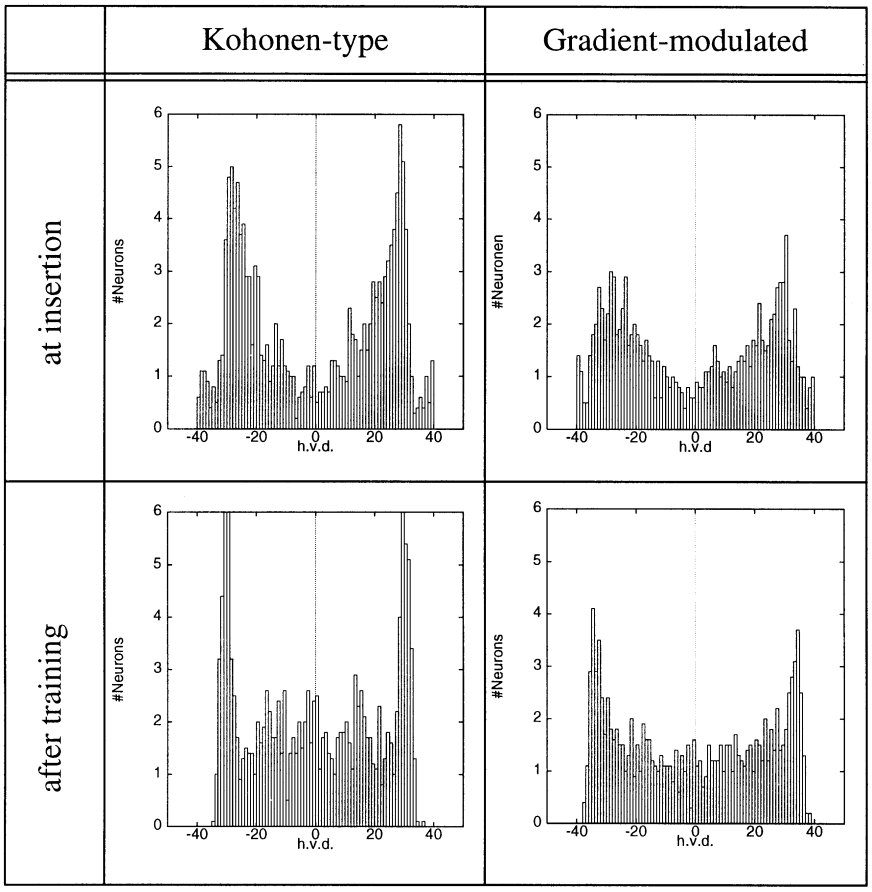

Fig. 7. Distribution of centers projected on the h.v.d. $\theta[^\circ]$ for the two learning rules. *Upper row*: at the time of insertion. *Lower row*: at the end of the off-line learning phase

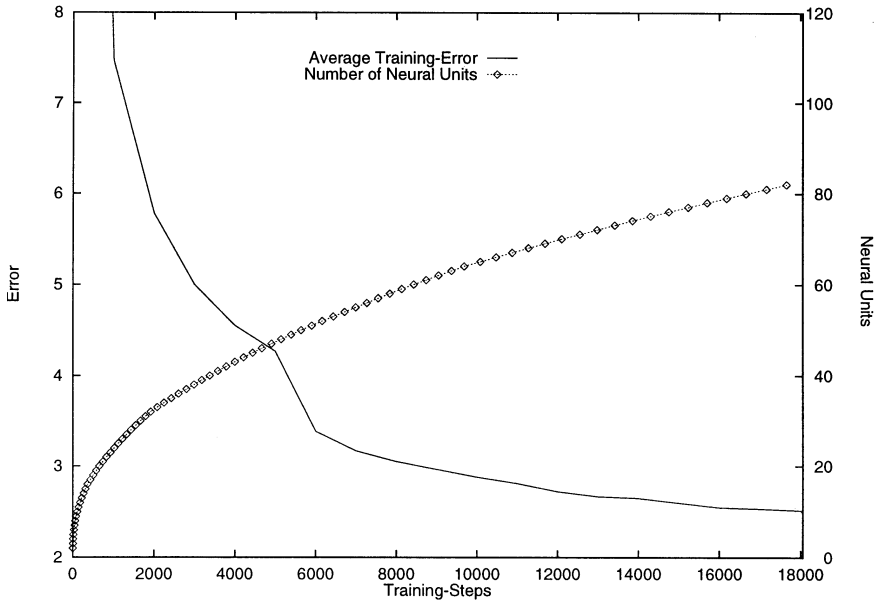

Fig. 8. Average binocular fixation error in pixels and the number of neutral units in the DCS network versus the number of training steps in the off-line training phase

is about 7 pixels (1.5%), which is due to deviations of the model from the physical system. However, due to on-line learning by error feedback the error drops to 2.0 pixels within 1000 saccades.

Finally we performed an experiment similar to that described in Henson (1978). In Henson's (monocular) experiment subjects were fitted with a contact lens that made them artificially myopic. This myopia was then neutralized by a conventional spectacle lens. Yet because the contactlens moves with the eye, the subject experiences a prismatic effect, based toward the center of the lens for the eye, at all positions of gaze other than that through the optical center of the spectacle. Henson has reported that after 14 min (about 250 saccades) the oculomotor system had managed to adapt to the new conditions and the saccades showed almost the same distribution of undershoots, overshoots and normomet-

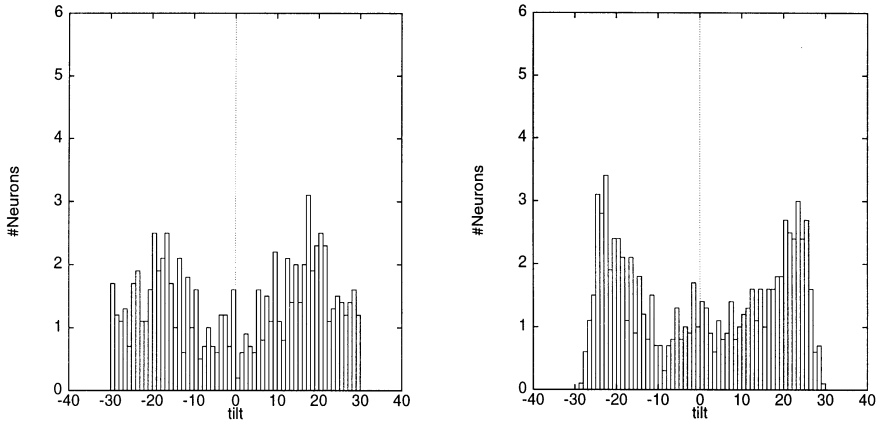

Fig. 9. Distribution of centers projected on the tilt angle θ (deg) for learning binocular fixation with the gradient-modulated learning rules: At the time of insertion (*left*) and at the end of training (*right*)

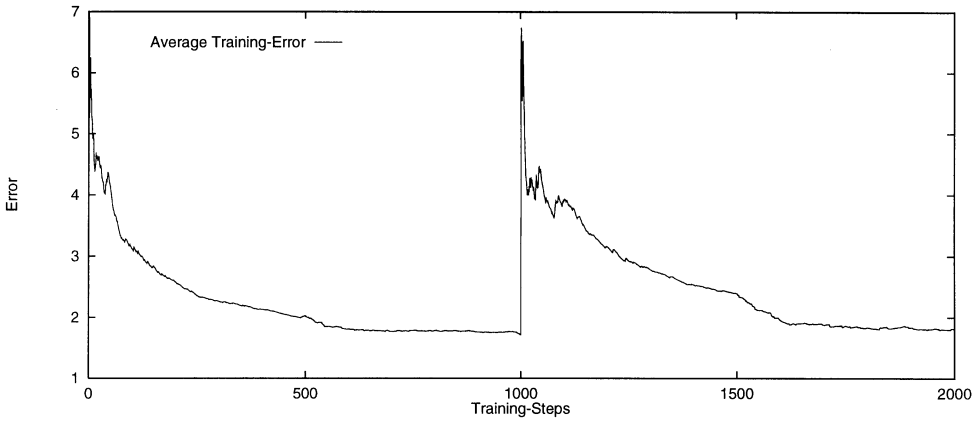

Fig. 10. Average fixation error in pixels versus the number of training steps in the operating phase

ric movements as before wearing the contactlens spectacle combination.

To demonstrate on-line adaptivity to altered visual input in our binocular system, we increased the zoom of one of the cameras as reflected by the peak in the error plot in Fig. 10. Increasing the zoom increases the focal length and hence the intrinsic parameters α_{xc} and α_{yc} in (3) as well as x_{0c} and y_{0c} if the latter differ from zero. Due to magnification, smaller changes in angles are now required to compensate the same retinal coordinates. The controller is able to adapt to the new parameter setting of the cameras within the next 600 saccades (Fig. 10).

7 Related work

The adaptive saccade controller based on DCS is related to the work of Dean et al. (1991) and Mayhew et al. (1992), and this is the first topic to be discussed here. We continue with the relation of our DCS architecture to similar architectures and finally provide a brief overview of other active vision projects involving binocular camera-heads.

7.1 Adaptive saccade control of a binocular camera-head with ANN

The most closely related approach to adaptive saccade control of a binocular camera-head with artificial neural networks is that of Dean et al. (1991) and Mayhew et al. (1992). Dean et al. (1991) tested several types of networks for saccade control of a *simulated* four-degrees-of-freedom camera-head. These included a linear associator, a Multi-Layer Perceptron (MLP), Albus' Cerebellar Model Arithmetic Computer (CMAC) and two-layered hierarchical networks, the last consisting of a linear net plus a CMAC or MLP trained to compensate the error of the linear net. They used supervised as well as feedback error learning and directly associated a seven-dimensional input vector (including pan) with changes in pan, tilt and h.v.d. Dean et al. noted a strong linear component, which was their motivation to use the two-layered networks.[16] One of their conclusions was that *it would be useful to have a net that could adjust its own granularity* and networks that could be *trained incrementally* – a major issue in this paper.

Interestingly, the weight analysis of their MLP networks revealed very small weights from pan input-units but large from tilt input-units. In light of the inverse kinematics of binocular fixation this can be well explained – the controller output indeed does not depend on the pan.

Mayhew et al. (1992) discussed adaptive saccade control as an application of their PILUT architecture. No performance results were given, yet the PILUT bears similarity to our approach: a first layer acts as a relatively coarsely coded indexing network (comparable to our hidden layer of RBF

[16] For their best networks (the two-layered ones), Dean et al. (1991) reported fixation errors of 5 pixels with simulated 256^2 pixel camera images. In a similar workspace and in similar training time our DCS controller reaches 2.5 pixel accuracy on 512^2 pixel camera images.

units) that blends local piecewise linear approximations carried out in the second layer. However, they did not report attempts to incrementally grow the indexing layer nor did the PILUT learn the topology of the input manifold.

In a more recent article, Zahn et al. (1996) reported on supervised training of a monocular camera system to generate time-optimal saccades. The interesting point about this work is that an MLP was trained to learn directly the appropriate acceleration and deceleration times for fixating a target point with maximum torque values (bang-bang control). This therefore made unnecessary the additional controller that translates changes in gaze direction (the output of our as well as Dean's saccade controller) into a corresponding torque profile.

7.2 Network architecture

Concerning the characteristics of our DCS networks, DCS as introduced in Sect. 4 are closely related to Fritzke's Growing Cell Structures (GCS; Fritzke 1995a) and Martinetz' Topology Representing Networks (TRN; Martinetz et al. 1994). In fact, DCS can be regarded as merging GCS with TRN (for a detailed discussion see Bruske et al. 1995). Independently of our work, Fritzke has further developed his GCS and created his Growing Neural Gas (GNG), which is similar to our DCS (Fritzke 1995b).

7.3 Relation to other active vision systems

There are a growing number of different research groups within the field of active vision that use binocular vision systems. However, emphasis here is usually not on adaptivity and learning but on better and more robust estimation of depth, motion, shape, etc., and hence they have employed standard (model-based) calibration.

As one of the first, Abbot et al. (1988) elaborated on improved surface reconstruction by combining the mechanical and optical degrees of freedom of an active vision system. Later, Krotkov (1989) combined stereo with vergence and depth from focus for more robust depth estimation. Pahlavan et al. (1994) and Uhlin et al. (1994) investigated the potential of active stereo systems for scene exploration by dynamic fixation and attentive vision on a broader scale, and these groups share an interest in real-time applications for motion and depth estimation with our own research group. The work of Daniilidis et al. (1996) and Hansen et al. (1996) represents first steps towards a repertoire of oculomotor behaviors for vision-based reactive real-time navigation.

8 Conclusion

Utilizing a DCS network for learning calibration-free saccade control from error feedback we started where previous work (Dean et al. 1991) ended. By incrementally growing the network dup to the size where it meets a prespecified accuracy level we tried to utilize as few neural units as possible to achieve the accuracy level. Since only a minor fraction of these units is involved in calculating the output of the DCS network, the generation of a controller output is very fast (even on conventional hardware) and suitable for real-time saccade control. By virtue of building optimally topology-preserving maps, the centers of the neural (RBF) units and the lateral connection structure are restricted to the relevant regions of the input space. Different learning rules have been tested for center adaptation, most successfully the gradient-modulated one. The density of neural units was demonstrated to be high in regions of the input space where they are actually needed, i.e., in regions where the required sensory-motor mapping is nonlinear and thus difficult to approximate by the locally linear model. Insight into the inverse kinematics helped to reduce the dimension of the input space and to design the DCS-based controller architecture.

The saccade controller has been successfully demonstrated to learn saccade control with high precision and to adapt to changing parameters on a physical four-degrees-of-freedom binocular camera-head.

Acknowledgements. We would like to thank both our anonymous reviewers for their detailed reviews. Their constructive criticism and many helpful hints have been much appreciated and helped to increase both the clarity and readability of our manuscript. This paper is based on Technical Report No. 9608, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel, 1996.

References

Abbott AL, Ahuja N (1988) Surface reconstruction by dynamic integration of focus, vergence and stereo. Proc Int Conf Computer Vision, pp 532–543

Ahrns I, Bruske J, Sommer G (1995) On-line learning with dynamic cell structures. Proc Int Conf Artificial Neural Networks, ICANN'95, vol 2, pp 141–146

Aslin RN (1987) Motor aspects of visual development in infancy. In: Salapatek P, Cohen L (eds) Handbook of infant perception. Academic Press, New York, pp 43–113

Bruske J, Sommer G (1995) Dynamic cell structure learns perfectly topology preserving map. Neural Comput 7: 845–865

Bruske J, Ahrns I, Sommer G (1996a) Neural fuzzy control based on reinforcement learning and dynamic cell structures. Proc Fourth Eur Congr Intelligent Techniques and Soft Computing, Aachen, pp 710–714

Bruske J, Riehn L, Hansen M, Sommer G (1996b) Dynamic cell structures for calibration-free adaptive saccade control of a four-degrees-of-freedom binocular head. Technical report 9608. Inst. f. Inf. u. Prakt. Math., CAU zu Kiel

Carpenter R (1988) Movement of the eyes, 2nd edn. Pion, London

Daniilidis K, Krauss C, Hansen M, Sommer G (1996) Real-time tracking of moving objects with an active camera, J Real Time Imaging (in press)

Dean P, Mayhew JE, Thacker N, Langdon PM (1991) Saccade control in a simulated robot camera-head system: neural net architectures for efficient learning of inverse kinematics. Biol Cybern 66: 27–36

Dean P, Mayhew JE, Langdon PM (1994) Learning and maintaining saccadic accuracy: a model of brainstem-cerebellar interactions. J Cogn Neurosci 6: 117–138

Deubel H, Wolf W, Hauske G (1986) Adaptive gain control of saccadic eye movements. Hum Neurobiol 5: 245–253

Faugeras O (1993) Three-dimensional computer vision. MIT Press, Cambridge, Mass

Frisone F, Firenze F, Morasso P (1995) Application of topology-representing networks to the estimation of the intrinsic dimensionality of data. Proc Int Conf Artificial Neural Networks, ICANN'95, vol 1, pp 323–327

Fritzke B (1995a) Growing cell structures: a self-organizing network for unsupervised and supervised learning. Neural Networks 7: 1441–1460

Fritzke B (1995b) A growing neural gas network learns topologies. Adv Neural Inform Processing Syst 7: 497–504

Hansen M, Sommer G (1996) Active depth estimation with gaze and vergence control using gabor filters. Proc 13th Int Conf Pattern Recognition: pp 334–348

Henson DB (1978) Corrective saccades: effects of altering visual feedback. Vision Res 18: 63–67

Jordan M, Rumelhart D (1992) Forward models: supervised learning with a distal teacher. Cogn Sci 16: 307–354

Kawato M (1990) Computational schemes and neural network models for formation and control of multijoint arm trajectory. In: Miller WT, Sutton RS, Werbos PJ (eds) Neural networks for control. MIT Press, Cambridge, Mass, pp 197–228

Kawato M (1995) Cerebellum and motor control. In: Arbib M (ed) The handbook of brain theory and neural networks. MIT Press, Cambridge, Mass, pp 172–178

Kawato M, Furukawa K, Suzuki R (1987) A hierarchical neural-network model for control and learning of voluntary movement. Biol Cybern 57: 169–185

Kohonen T (1987) Adaptive, associative, and self-organizing functions in neural computing. Appl Optics 26: 4910–4918

Krotkov E (1989) Active computer vision by cooperative focus and stereo. Springer, Berlin Heidelberg New York

Martinetz T, Schulten K (1994) Topology representing networks. Neural Networks 7: 507–522

Mayhew JE, Zheng Y, Cornell S (1992) The adaptive control of a four-degrees-of-freedom stereo camera head. Phil Trans R Soc Lond 337: 315–326

Miyamoto H, Kawato M, Setoyama T, Suzuki R (1988) Feedback-error-learning neural network for trajectory control of a robotic manipulator. Neural Networks 1: 251–265

Pahlavan K, Uhlin T, Eklundh JO (1994) Integrating primary ocular processes. In: Proc 3rd Eur Conf Computer Vision. (Lecture notes in computer science 588) Springer, Berlin Heidelberg New York, pp 546–554

Poggio T (1990) A theory of how the brain might work. Cold Spring Harbor Symp Quant Biol 55: 899–910

Ritter H, Schulten K (1986) On the stationary state of Kohonen's self-organizing sensory mapping. Biol Cybern 54: 99–106

Robinson DA (1975) Oculomotor control signals. In: Lennerstrand G, Bach-y-Rita P (eds) Basic mechanisms of ocular motility and their clinical implications. Pergamon Press, Oxford, pp 337–374

Tsai R(1986) An efficient and accurate camera calibration technique for 3D machine vision. Proc Int Conf Computer Vision and Pattern Recognition, pp 364–374

Uhlin T, Nordlund P, Maki A, Eklundh JO (1995) Towards an active visual observer. Proc 5th Int Conf Computer Vision, pp 679–686

Villmann T, Der R, Martinetz T (1994) A novel approach to measure the topology preservation of feature maps. Proc Int Conf Artificial Neural Networks, ICANN'94, pp 298–301

Zahn V, Eckmiller R (1996) How to train an unknown 2D camera movement system to generate time-optimal saccades. Proc Int Conf Neural Information Processing, Hong Kong, pp 401–404

Zee DS, Optican LM (1985) Studies of adaptation in human oculomotor disorders. In: Berthoz A, Melvill Jones G (eds) Adaptive mechanisms in gaze control. Elsevier, Amsterdam, pp 165–176