Available online at www.sciencedirect.com

ScienceDirect

Knowledge-Based Systems 19 (2006) 348–355

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

A camera-direction dependent visual-motor coordinate transformation for a visually guided neural robot

Cornelius Weber *, David Muse, Mark Elshaw, Stefan Wermter

Hybrid Intelligent Systems, School of Computing and Technology, University of Sunderland, UK[1]

Received 28 October 2005; accepted 28 November 2005
Available online 17 February 2006

Abstract

Objects of interest are represented in the brain simultaneously in different frames of reference. Knowing the positions of one's head and eyes, for example, one can compute the body-centred position of an object from its perceived coordinates on the retinae. We propose a simple and fully trained attractor network which computes head-centred coordinates given eye position and a perceived retinal object position. We demonstrate this system on artificial data and then apply it within a fully neurally implemented control system which visually guides a simulated robot to a table for grasping an object. The integrated system has as input a primitive visual system with a what–where pathway which localises the target object in the visual field. The coordinate transform network considers the visually perceived object position and the camera pan-tilt angle and computes the target position in a body-centred frame of reference. This position is used by a reinforcement-trained network to dock a simulated PeopleBot robot at a table for reaching the object. Hence, neurally computing coordinate transformations by an attractor network has biological relevance and technical use for this important class of computations.

Keywords: Frame of reference transformations; Neural networks; Boltzmann machine; Reinforcement learning; Robotics

1. Introduction

There is a view of current developments leading to a major advance in a future personal robot industry, as illustrated by citations such as: "In thirty years I think it [the personal robot industry] will be bigger than the personal computer industry" [16]. Yet a key capability currently limiting robotic expansion is image processing, one function of which is to identify the position of an object and making it available to mechanical actuators, for example for grasping.

The control of the human body is a complex task due to the complexity of the body geometry and the difficulty to extract information from the world by sensors like vision and to transform it into a motor-relevant representation. So to simply grasp an object, we need to (i) visually localise an object, (ii) infer its position in body-centred coordinates which are relevant for control of the arm and hand and (iii) activate the relevant muscles to perform a grasp.

Here, we present a neural model which consists of three systems, (i) a visual, (ii) a coordinate transform and (iii) a motor system, which performs such a task on a simulated robot. The complexity of the human body is schematically addressed by the robot camera which can pan-tilt its gaze direction during the docking. This accounts for the fact that the eyes and/or the head can move with respect to the body, which makes it necessary to transform a visually identified location into a body-centred location as is relevant for the motor control.

In [18], we have implemented a vision controlled robotic docking action that was trained by reinforcement learning. The assumption that the robot camera was fixed to the body allowed a direct match from pixel coordinates to body-centred coordinates. Because of the fixed camera, objects had to be in a confined space so that they were visible. The grasping manoeuvre is

* Corresponding author.
E-mail address: cornelius.weber@sunderland.ac.uk (C. Weber).
[1] www.his.sunderland.ac.uk

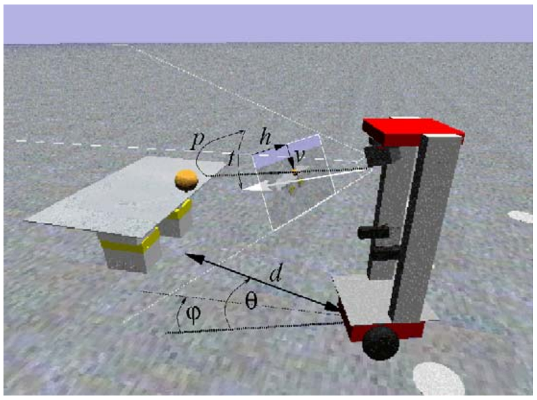


Fig. 1. The simulated PeopleBot robot with its short black grippers in its environment. Coordinates are as follows: θ is the angle of the target w.r.t. the forward direction of the robot and d is the distance between the robot and the target object. φ is the robot orientation angle w.r.t. an axis that is perpendicular to the table and will be used later, directly in motor control. θ and d comprise the body-centred representation of the target object which is relevant for motor control but not given directly by sensors. p is the pan angle and t is the tilt angle of the camera. h and v are the horizontal and vertical image pixel coordinates of the perceived target object (the white frame shows the image seen by the robot camera). The body-centred coordinates (θ, d) are to be computed as a function of easily accessible values (h, v) and (p, t).

shown at the following URL: www.his.sunderland.ac.uk/robotimages/Cap0001.mpg.

In [11], the camera is allowed to move, but it is assumed that it is fixating the target when computing its direction. Hence a reaching map can be defined using only the camera posture. When a robot or the object is moving, however, a camera can hardly fixate the object, in particular when using commercially available, slow pan-tilt camera mounts.

Let us briefly discuss the underlying geometry of the coordinate transformation problem. For simplicity we will in the following not distinguish eye- and head-position, accounting for a pivoting camera which is mounted on a robot body. Fig. 1 visualises the geometry of our setup with a PeopleBot robot.[2] If the robot is to grasp the fruit object on the table, then the following coordinates are important for controlling the motors: the distance d of the robot to the target object and the angle θ at which the object is to the left or the right of the robot body. In order to avoid bumping into the table with the robot's "shoulders" when near the table, the angle φ of the robot rotation w.r.t. the table edge will later also be used for motor coordination.

While d and θ are required for motor control, the robot sensory system represents the object only visually, delivering the perceived horizontal and vertical positions h and v of the object in the visual field. Knowing also the camera pan- and tilt-angles p and t, it is possible to compute d and θ. We assume a constant elevation of the camera over the target object which allows the distance of the object to be estimated from how low it is perceived, thus from v and t. This compensates for not using a stereo camera. In summary, (d, θ) are a function of (h, v, p, t), and the purpose of the coordinate transform network is to learn and compute this function.

It would be possible, even with complicated human geometries, to compute this function using deterministic vectorial transformations. Humans, however, *learn* sensory-motor coordination, which allows for adaptations during evolution and ontogenesis. In the mammalian cortex transitions between visual and motor representations are made in the posterior parietal cortex (PPC) which lies at a strategic position between the visual cortex and the motor cortex. PPC neurons are modulated by the direction of hand movement, as well as by visual, eye position and limb position signals [2]. These multi-modal responses allow the PPC to carry out computations which transform the location of targets from one frame of reference to another [3,4].

Models of neural coordinate transformations originally dealt with the "static" case, in which, for example, Cartesian coordinates (c_1, c_2) of an object (e.g., as seen on the retina) are neurally transformed into joint angles (θ_1, θ_2) of an arm required to reach the target [8]. Such a model is static by not accounting for the influence of another variable, such as the rotation of the head. To account for such a modulating influence, we need dynamic, adjustable mappings.

A standard way to achieve a dynamic mapping is to feed the two inputs such as Cartesian coordinates c and head rotation r into a hidden layer. These inputs are coded as population vectors $\boldsymbol{x}^c$ and $\boldsymbol{x}^r$ for neurons arranged along a one-dimensional line where the location of an approximately Gaussian-shaped activation hill encodes the value. Both inputs are used in a symmetric way. The working principle of the use of the hidden layer is described as [17]: "One first creates a two-dimensional [hidden] layer with an activity equal to the [outer] product of the population activity in $\boldsymbol{x}^c$ and $\boldsymbol{x}^r$. Next, a projection from this layer to an output layer implements the output function $\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}^c, \boldsymbol{x}^r)$".

Such a network with two one-dimensional input layers, a one-dimensional output layer and a two-dimensional hidden layer has been termed a basis function network [5]. Because of its structure, the output layer is symmetric with the input layers and the network can be used in any direction. Lateral weights within each layer allow for a template fitting procedure during which attractor network activations generate approximately Gaussian-shaped hills of activations. In a "cue integration" mode the network receives input with additive noise at all three visible layers and produces the correct, consistent hills with maximum likelihood [5].

The gain field architecture [14] adds a second hidden layer which subtracts certain inputs to remove unwanted terms from the solution on the first hidden layer. This allows it to encode not only the position of a hill of activation, but also its amplitude. Since this breaks the symmetry between the input layers and the output layer, this network is used only in one direction.

[2] Konstantinos Karantzis implemented the PeopleBot in the gazebo robot simulator.

The use of the hidden layer as the outer product of input layers has the advantage that the hidden code or the weights can easily be constructed using algebraic transformations. A learning algorithm for all the weights is not given with these models. A specific disadvantage is the large dimensionality of the hidden layer: if both input layers are two-dimensional, as in our problem, then the hidden layer would have to be represented as a 4-dimensional hyper-cube.

Here, we propose a network which learns the coordinate transformation. Every unit is connected with all other units by connection weights which are trained according to the Boltzmann machine learning rule [1] (see [9] for an introduction). This rule is biologically plausible using only local, Hebbian and anti-Hebbian learning. After training, the network generates the distribution of the training data using stochastic units. The learning rule furthermore allows to include any number of additional hidden units in order to make the network more powerful in generating a complex data distribution. The hidden code would self-organise during learning without requiring the network designer to construct it. For our data, however, no hidden layer was required. During learning, all three areas receive their respective coordinate as training data in a symmetric fashion, while after learning, missing information in any area can be recovered based on the principle of pattern completion.

Using two-dimensional input areas and artificial test data described in Section 2, we will describe training of the network in Section 3 and show how it performs coordinate transformations in Section 4, as required for the robotic scenario. In Section 5, we show how the coordinate transformation network can be applied as part of a neural control system that docks a robot at a table to grasp a visually identified object. In Section 6, we will discuss the results and underlying assumptions and Section 7 gives a summary.

2. Test scenario

Using the symbols from Fig. 1, let us introduce the following notation for two-dimensional coordinates: $\boldsymbol{\alpha}^{vis} = (h, v)$ is the position of the object of interest in the camera image. $\boldsymbol{\alpha}^{head} = (p, t)$ is the camera rotation angle. $\boldsymbol{\alpha}^{body} = (\theta, d)$ is the body-centred position of the object of interest. As a first test example, we choose an abstract coordinate transformation defined by

$$\boldsymbol{\alpha}^{body} = \boldsymbol{\alpha}^{vis} + \boldsymbol{\alpha}^{head}, \tag{1}$$

where the individual terms of the vectors are:

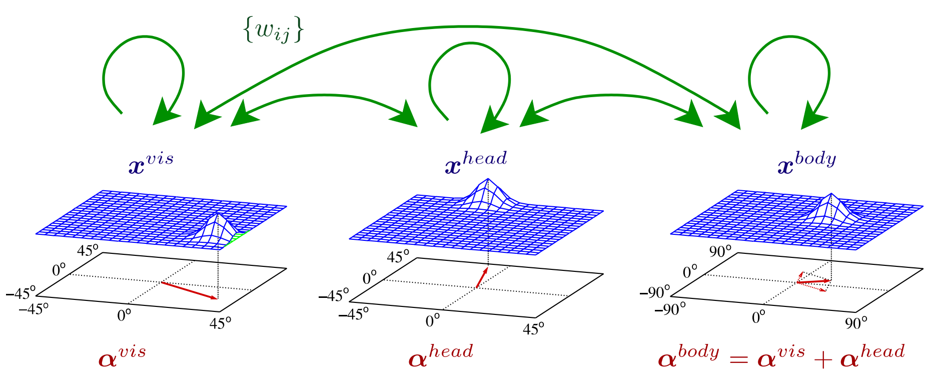
$$\theta = h + p, \tag{2}$$

$$d = v + t. \tag{3}$$

Eq. (2) describes well the true relation between body-centred, visually perceived and camera position in the horizontal plane. Eq. (3), however, would not describe the true, non-linear, relation between the coordinates in the vertical dimension. Therefore, these equations are just a simplified account of more general coordinate transformations.

The distance d of the object in the real scenario can somehow be computed from v and t given a realistic constraint that the object is on floor-level and assuming an elevated camera position. Hence the coordinates $\boldsymbol{\alpha}^{vis}$ and $\boldsymbol{\alpha}^{head}$ contain sufficient information to compute $\boldsymbol{\alpha}^{body}$. Other body-centred coordinates like distances in x- and y-direction may alternatively be computed.

We represent our coordinate vectors $\boldsymbol{\alpha}^{vis}$, $\boldsymbol{\alpha}^{head}$ and $\boldsymbol{\alpha}^{body}$ on neural sheets as neural activation vectors $\boldsymbol{x}^{vis}$, $\boldsymbol{x}^{head}$ and $\boldsymbol{x}^{body}$, respectively. Such a coding of low-dimensional vectors by high-dimensional neuron activations is called population coding. Neural activations within the $\boldsymbol{x}$ vectors are defined by envelopes of Gaussians centred on the corresponding positions of the $\boldsymbol{\alpha}$ vectors. Fig. 2 depicts the algebraic transformation and the mapping to neural representations. Neural population coding allows arbitrary coordinate representations by replacing the coordinate systems in the lower part of Fig. 2.



$\{w_{ij}\}$
$\boldsymbol{x}^{vis}$
$\boldsymbol{x}^{head}$
$\boldsymbol{x}^{body}$
45°
0°
–45°
–45°
0°
45°
45°
0°
–45°
–45°
0°
45°
90°
0°
–90°
–90°
0°
90°
$\boldsymbol{\alpha}^{vis}$
$\boldsymbol{\alpha}^{head}$
$\boldsymbol{\alpha}^{body} = \boldsymbol{\alpha}^{vis} + \boldsymbol{\alpha}^{head}$

Fig. 2. A coordinate transformation on two-dimensional manifolds. (Below) Three two-dimensional vectors, $\boldsymbol{\alpha}^{vis}$, $\boldsymbol{\alpha}^{head}$ and $\boldsymbol{\alpha}^{body}$ and their relation to each other which is also graphically explained by the dotted lines in the right graph. Note the extended range of the body-centred coordinate system, which makes the other vectors projected into that system appear half as long. (Middle) The neural population code $\boldsymbol{x}^{vis}$, $\boldsymbol{x}^{head}$ and $\boldsymbol{x}^{body}$ representing each vector on two-dimensional sheets of neurons. A hill of neural activation carries the information about the corresponding position of each vector. (Top) The network architecture: the set of weights $\{w_{ij}\}$ connects every unit with every other in all three layers.

3. Architecture and training

The coordinate transform network architecture is depicted in Fig. 2, top. It consists of three fully connected areas which represent $\boldsymbol{x}^{vis}$, $\boldsymbol{x}^{head}$ and $\boldsymbol{x}^{body}$. The Boltzmann machine learning rule uses randomised unit update rather than structured information flow and lends itself to highly interconnected networks with the possibility to introduce any number of additional hidden units to boost performance. With its binary stochastic units it is powerful in learning a given data distribution. For the specific purpose of function approximation other biologically plausible schemes (e.g., [12]) would also be possible.

The Boltzmann machine has two running modes: in the *clamped phase* the data distribution is forced upon the visible units of the network. The network activation states $\boldsymbol{x}$ are then subject to the distribution $P^{+}_{\boldsymbol{x}}$ where the upper index "+" denotes the *clamped phase*. Since in our case there are no hidden units, the network state consists only of the three visible input areas, i.e.: $\boldsymbol{x} := \{\boldsymbol{x}^{vis}, \boldsymbol{x}^{head}, \boldsymbol{x}^{body}\}$. The other running mode is the *free running phase* in which the distribution $P^{-}_{\boldsymbol{x}}$ over the network states arises from the stochasticity of the units and is determined by the network parameters, such as weights and thresholds. Here, the upper index "−" denotes the *free running phase*.

The goal of learning is that the distribution $P^{-}_{\boldsymbol{x}}$ generated by the network approximates the data driven distribution $P^{+}_{\boldsymbol{x}}$ which is given. $P^{+}_{\boldsymbol{x}} \approx e^{-E(x)}$ is a Boltzmann distribution which depends on the network energy $E(\boldsymbol{x}) = \sum_{i,j} w_{ij}x_i x_j$ where w_{ij} denotes the connection weight from neuron j to neuron i. Therefore, $P^{-}_{\boldsymbol{x}}$ can be molded by training the network parameters. Derivation of the "distance"[3] between $P^{-}_{\boldsymbol{x}}$ and $P^{+}_{\boldsymbol{x}}$ w.r.t. the network parameters leads to the learning rule

$$\Delta w_{ij} = \epsilon\left(\sum_{\{x\}} P^{+}_{\boldsymbol{x}} x_i x_j - \sum_{\{x\}} P^{-}_{\boldsymbol{x}} x_i x_j\right) \tag{4}$$

with learning step size ϵ which we set between 0.0025 and 0.001. Computing the left term corresponds to the *clamped phase* of learning, the right term to the *free running phase*. Without hidden units, the left term in Eq. (4) can be rewritten as $\sum_{\mu}^{data} x_i^{\mu} x_j^{\mu}$ where μ is the index of a data point. Without hidden units thus the *clamped phase* does not involve relaxation of activations.

The right term of Eq. (4) can be approximated by sampling from the Boltzmann distribution. This is done by recurrent relaxation of the network in the *free running phase*. The stochastic transfer function

$$P(x_i(t+1) = 1) = \frac{1}{1 + e^{-\sum_j w_{ij} x_j(t)}} \tag{5}$$

computes the binary output $x_i \in \{0, 1\}$ of neuron i at time step $t+1$. Repeated relaxation approximates a Boltzmann distribution of the activation states.

During training, the two phases are computed alternating. One randomly generated data point is presented to account for the *clamped phase*. Then a relatively short relaxation of the network, consisting of updating all units for 15 iterations using Eq. (5) is performed to account for the *free running phase*. Units are initialised in this phase by activating every unit with a probability of 0.1, regardless of its position. During testing, we also used the deterministic continuous transfer function

$$x_i(t+1) = \frac{1}{1 + e^{-\sum_j w_{ij} x_j(t)}}. \tag{6}$$

Self-connections were omitted in the expectation that they would grow very large, but later assessment showed that this would not have been the case. Instead, a threshold θ_i was added to each unit. It was treated during training as a weight that was connected to an external unit with a constant activation of -1. There are no further weight constraints.

4. Coordinate transformation results

We have explored two methods of sampling the training data, described below. Both satisfy Eq. (1). From the two-dimensional $\boldsymbol{\alpha}$ vectors, high-dimensional neural population vectors $\boldsymbol{x}$ are produced, as visualised in Fig. 2. The Gaussian envelopes over the neural activations have a maximum value of 1. The Boltzmann machine learning rule accepts continuous values between 0 and 1 which are treated as probabilities of a neuron being in the active state.

The first method was to uniformly sample $\boldsymbol{\alpha}^{vis}$ and $\boldsymbol{\alpha}^{head}$, and then to produce the position $\boldsymbol{\alpha}^{body}$ dependent on these. This leads to a non-uniform distribution of $\boldsymbol{\alpha}^{body}$ which is biased toward the centre (background shading in Fig. 5(a)). The reason for this is that there are more combinations possible from the visual- and head-input to produce a position in the middle of the body centred coordinate system.

The second method was to uniformly sample $\boldsymbol{\alpha}^{body}$ first, and then randomly generate one of $\boldsymbol{\alpha}^{vis}$ or $\boldsymbol{\alpha}^{head}$, and construct the other so that geometrical relations are met, if possible. This was not always the case as some combinations of, e.g., $\boldsymbol{\alpha}^{head}$ and $\boldsymbol{\alpha}^{body}$ would require $\boldsymbol{\alpha}^{vis}$ to be outside of its range. For example if $\boldsymbol{\alpha}^{body}$ denotes an object position 50° (to the right) and $\boldsymbol{\alpha}^{head}$ denotes a head angle of −30° (to the left), then the object would have to be in the visual field at a position of 80° in order to satisfy Eq. (1) and $50° = 80° + (-30°)$, which is outside of the range of the visual field. In these cases, a different random input activation was generated until a mapping could be made to produce the required output. Note that both methods produce

[3] Correctly, the distance has to be termed the Kullback–Leibler divergence. See [9] for a derivation of the learning rule.

the same training data, but in a different probability distribution.

The network was trained to produce the correct vector $\boldsymbol{x}^{body}$ with 50,000 data points which took around 5 h to complete running on a Linux based desktop. The size of each layer was 15×15 units. Even though the network can work in any direction or perform "cue integration" [5] if input is applied to all areas, we will consider only the task where it is initialised with $\boldsymbol{x}^{vis}$ and $\boldsymbol{x}^{head}$ as input vectors. Then it will produce a vector $\boldsymbol{x}^{body}$ that is consistent with these inputs ("function approximation" [5]). $\boldsymbol{x}^{vis}$ and $\boldsymbol{x}^{head}$ will also fluctuate slightly over time, if allowed to change. This is shown in Fig. 3 where we see that the activations representing $\boldsymbol{x}^{vis}$ move slightly downward between time steps 2 and 14. These fluctuations can be avoided by clamping the input units so that the data representation remains fixed on them.

In Fig. 4, we see how constant input $\boldsymbol{x}^{vis}$ and $\boldsymbol{x}^{head}$ effects the output area representing the body centred coordinates.

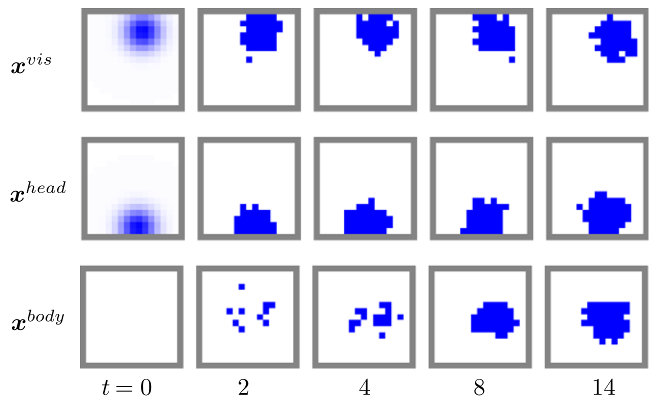


Fig. 3. Neural activations produced by the trained network for one example case. At time step $t = 0$ (left column) the network is initialised with continuous Gaussian shaped inputs on two of the three areas, $\boldsymbol{x}^{vis}$ and $\boldsymbol{x}^{head}$, and zeros within $\boldsymbol{x}^{body}$. Values are between zero (white) and 1 (dark). In the following iterations, at time steps indicated at the bottom of this figure, the network maintains binary stochastic activations. These account for the positive inputs but also fill in the missing input by generating an appropriate vector $\boldsymbol{x}^{body}$.

For example $\boldsymbol{x}^{head}$ in Fig. 4(a) represents a head position to the very far right and it contributes a strong positive net input to the right of the area representing the body centred position, as seen in Fig. 4(b). To the left of this area there is an even larger, inhibitory net input. The reason for this is that when the head is turned to the very right, the visual field does not cover the left side of the body, so the object that is acted upon cannot be at the very left of the body. Note that every data point involves a hill of neural activation on every area which implies that the object is always assumed to be seen. Thus, based on just one observation, such as $\boldsymbol{x}^{head}$, a rough estimate of $\boldsymbol{x}^{body}$ can be given even if $\boldsymbol{x}^{vis}$ is undetermined.

Similarly, a visually perceived position incurs a constraint in the body centred target position. If both constraints are combined, as in Fig. 4(b), right, then a relatively small region in the body representation area receives maximal net input. The remaining external influence onto the area is mainly inhibitory. Finally, Figs. 4(c) and (d) demonstrate that via recurrent relaxation using the lateral weights, after only two steps a focused pattern of activation emerges around the correct position $\boldsymbol{x}^{body}$, that is shown in Fig. 4(a). More examples of data and network output pairs are concatenated in an animation which can be seen at: http://www.his.sunderland.ac.uk/supplements/AI05/.

The errors produced by the network on the body centred area are illustrated in Figs. 5(a) and (b), here after 10,0000 learning steps with Gaussian centres avoiding the outermost units, in order to reduce boundary effects. Positions encoded by the network and those of the data were obtained by sliding a Gaussian over the area and selecting the position with maximum overlap to the neural activations. Data are averaged over altogether 20,000 data points while using Eq. (5) as stochastic neuronal activation function. On the left, we can see a tendency of the network to generate peripheral positions of $\boldsymbol{x}^{body}$ and thus $\boldsymbol{\alpha}^{body}$ toward the centre. The reason for this can be seen in the background shading of Fig. 5(a), which shows that a greater density of $\boldsymbol{\alpha}^{body}$ has been produced to the centre of its area

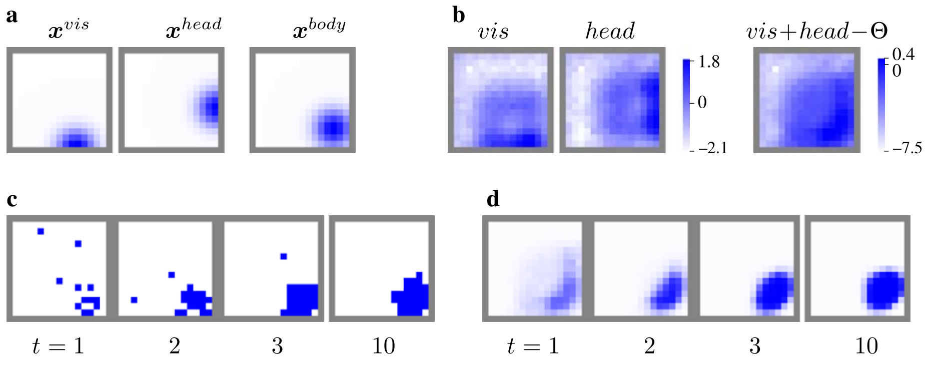


Fig. 4. Neural activations for an example case. (a) shows input vectors $\boldsymbol{x}^{vis}$ and $\boldsymbol{x}^{head}$ and the corresponding correct target vector $\boldsymbol{x}^{body}$. (b) Individual net input into the body representation area from the $\boldsymbol{x}^{vis}$ (left) and $\boldsymbol{x}^{head}$ (middle) vectors as well as the sum of both inputs minus thresholds Θ (right). Scale bars to the right indicate the values. (c) shows the binary vector $\boldsymbol{x}^{body}$ that emerges from the constant input in (b) and from further relaxation at time steps given below. It reproduces the target vector as a binary, stochastic code. (d) shows $\boldsymbol{x}^{body}$ emerging when using the continuous transfer function Eq. (6).

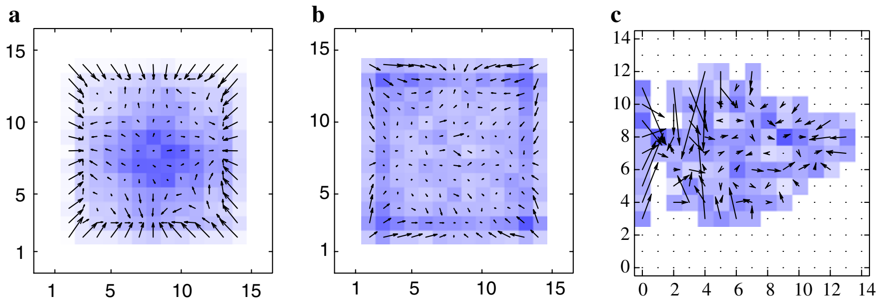


Fig. 5. Deviations of the predicted positions on the area coding the body-centred positions. The shading of the background indicates the data density of the generated positions. In (a), the distribution is biased toward the centre, as the visual- and head-positions were sampled homogeneously. In (b), body-centred positions were sampled uniformly, before generating visual and head-positions. Arrows show the systematic network errors, pointing from the correct target position toward the predicted position. (c) is a similar plot for the robotic data, where the ordinate denotes θ ranging from $-90°$ to $90°$ and the abscissa denotes $\sqrt{d}$ with d ranging from 0 to 2 m. See Section 5 for details.

when homogeneously sampling $\boldsymbol{\alpha}^{vis}$ and $\boldsymbol{\alpha}^{head}$. The learnt network represents this trend in its weights and biases.

In order to verify this influence of the inhomogeneity, we used the second method of sampling the training data, which ensured that activation hills were uniformly distributed on the body representation area as shown in Fig. 5(b). Comparing it with Fig. 5(a), we see that the strong tendency to predict positions away from the low density data regions around the border has been greatly reduced. We have furthermore averaged the errors over single trials, thus also capturing the noise induced by the stochasticity of the neurons resulting from Eq. (5). The average deviation between the correct body-centred object position and its estimation by the network was 0.79 and 0.67 units for the first and second method of sampling, respectively. With 15 units covering 180° (see Fig. 2) this corresponds to 9.48° and 8.04° deviation of the network estimation of $\boldsymbol{\alpha}^{body}$. This data shows that the network can produce usable and robust solutions of the dynamic coordinate transformation problem.

5. Application within a robot control system

The integrated robot control system is shown in Fig. 6. The visual system feeds into the coordinate transform system which then feeds into the motor system. The "eye-move" motor area is only concerned with focusing the camera on the target object. Its connections have been learnt in an error-driven fashion, strengthening (weakening) the weights if the camera movement was too little (too large) to focus an object. Independent of the range of the camera movement, when the robot is moving, the camera moves too slow for centring the object permanently, but fast enough to keep it somewhere in the field of view.

Apart from the coordinate transform network, the model including the visual and motor systems has been presented before [18]. There it docks the robot to the target object which must be relatively close, since the camera is fixed facing downward. While there the visual "where" representation was directly fed into the "state" area (cf. Fig. 6), now

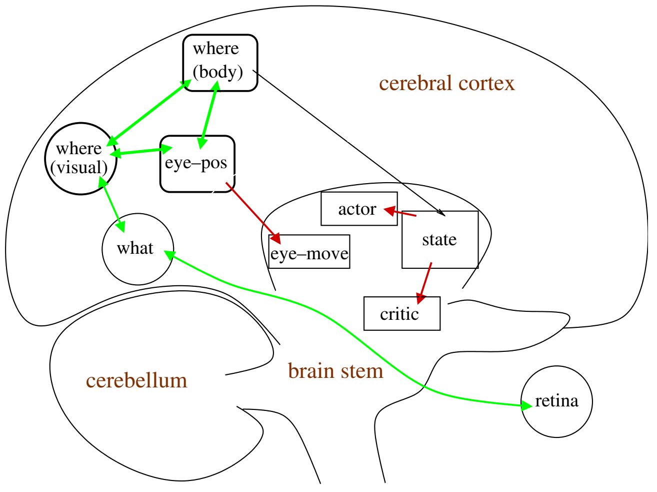


Fig. 6. The three areas involved in the coordinate transformation embedded in the full control model and overlaid on a sketch of the brain. The boxes denote the implemented areas of neurons; round in the visual system, oval-boxes in next steps of the coordinate transform system and rectangular in the motor system. Thick arrows represent trained connections, partially in both directions as the arrow heads suggest, the thin arrow denotes mere copying. Not displayed are recurrent within-area connections in each of the coordinate transform areas (bold outlines). Light green arrow colour denotes associative learning while dark red denotes error-driven or actor-critic reinforcement learning of connections which is associated with the basal ganglia. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

the body-centred "where" representation feeds into the "state" area instead. This body-centred "where" area codes along one axis for the body-centred target object angle θ (cf. Fig. 1), and along the other axis for the square root $\sqrt{d}$ of the distance between the robot and the object (the square root function expands close ranges where higher precision is needed). Fig. 5(c) shows the errors produced by the coordinate transform network. Not all of the area received any data when the robot was moved to randomised positions, as the background shading shows. Large errors along the axis of the angle θ at small distances d can be explained by the training data: similar visual inputs h, v and pan-tilt values p, t were sometimes paired with largely differing, contradicting values of d, θ. This is a consequence of a "singularity" at the origin, where changing θ does not result in a change of other values, imprecise object localisation and robot movement and sensor latencies.

The state space area then contains an additional third dimension coding for the robot rotation angle φ. The robot needs this information to avoid crashing into the table with its wide shoulders when approaching from the side. Demonstration videos of the system controlling the simulated PeopleBot robot can be seen at: http://www.his.sunderland.ac.uk/supplements/AI05/.

Next, we will implement this demonstration on the real PeopleBot robot.

6. Discussion

We have presented a network that can perform dynamic coordinate transformations on test data and in a practical scenario based on the principle of pattern completion of an auto-associator attractor network. The network transforms a hill of neural activation, but not a general pattern, because during training such a hill of activation was always given to the output area. Thereby the weights have been trained to produce the competition that allows just one hill to emerge on each area. This would be a limitation in e.g., the lower visual system where this transformation would remove all except the positional information. However, we conceive the network to reside in a higher level of the dorsal "where" stream in the PPC, which is known to abstract from information that is processed in the lateral "what" stream.

As the network will produce a refined output, even if the input is more scattered, we observe an effect similar to attention, which is also attributed to the PPC [7]. This allows the system to handle noisy input data as the emerging activation hill removes the noise, leading to robustness of the network. It should be noted that while such a network architecture allows just one hill of activation for *rate* coded neurons, it would allow more than one hill if these were separated in another dimension such as the *phase* of spiking neurons (e.g., [13]).

There is a restriction on the range over which our network can make a transformation. As we have seen in Fig. 4, location estimates of $\boldsymbol{x}^{vis}$ and $\boldsymbol{x}^{head}$ are added to obtain an estimate of $\boldsymbol{x}^{body}$. It is important that each of the two inputs must convey information about and provide a bias for the location of the target. If for example the head position was irrelevant to the target position, then the visual estimate which per se permits many body-centred positions could not be narrowed down any further. This does not imply that the target object is altered by the head position, but it means that only such a target may be chosen which is within the visual field. This is a realistic assumption accounting for all cases in which an agent's hand is visually guided to an object.

Extensions of existing networks have been made to achieve object-centred representations with a basis function network [6] or to transform another's frame of references into a self-centred frame of reference with a gain field architecture [15] which is necessary to perform imitation. We believe that our simplified architecture without any hidden layer also lends itself to such tasks.

We have so far assumed that each area's input arrives in a topographically arranged way and as a Gaussian-shaped activation hill. A specific question is how to learn these input mappings. The three areas convey different information which implies different, possibly independent mechanisms. (i) The topographic mapping of the visual object position is particularly evident in the lower visual system. (ii) The eye position is represented in several cortical areas, e.g., V6A, 7a, V3a, LIP, MT, MST, PO and the ventral premotor cortex (PMv) [10]. Many of these areas also respond to e.g., visual stimuli and hand position. Therefore, the development of an eye position map might be guided by other, e.g., visual maps, and might take into account factors such as the activations of the eye muscles. Since the eyes and the head can be moved independently (for most robots and their camera this is not the case), another mapping for the head position, possibly dependent on the muscles and vestibular signals, would be required. Finally, (iii) the development of the map of a body centred coordinate system might be controlled by factors such as the joint angles of the arm with which to reach the object or factors such as the duration to reach its position by self-movement. Thus, such a map is motor related and might reside in posterior parietal cortex where neurons in the monkey are also modulated by the direction of hand movement, as well as by visual, eye position and limb position signals [2].

7. Summary

We have developed an artificial neural network capable of performing a dynamic coordinate transformation to generate body centred coordinates based on the visual information and head orientation (pan-tilt of a robot's camera). It differs from static coordinate transformations in that the transformed variable (here, a visual object coordinate) is modulated by another coordinate (the head position) in order to obtain the target variable (the object coordinate relative to the agent's body). The network learns in a biologically plausible way and activations converge rapidly to a focused pattern on the output layer. This removes the need to manually solve and implement any intermediate computational steps of the transformation using a large number of additional units as is the case with other systems. The model advances our understanding of this important class of processes in the brain and helps extending the range of robotic applications.

Acknowledgements

This is part of the MirrorBot project supported by a EU, FET-IST programme, Grant IST-2001-35282, coordinated by Prof. Wermter.

References

[1] D. Ackley, G. Hinton, T. Sejnowski, A learning algorithm for Boltzmann machines, Cogn. Sci. 9 (1985) 147–169.

[2] C.A. Buneo, M.R. Jarvis, A.P. Batista, R.A. Andersen, Direct visuomotor transformations for reaching, Nature 416 (2002) 632–636.

[3] Y.E. Cohen, R.A. Andersen, A common reference frame for movement plans in the posterior parietal cortex, Nature Rev. Neurosci. 3 (2002) 553–562.

[4] J.D. Crawford, W.P. Medendorp, J.J. Marotta, Spatial transformations for eye-hand coordination, J. Neurophysiol. 92 (2004) 10–19.

[5] S. Deneve, P.E. Latham, A. Pouget, Efficient computation and cue integration with noisy population codes, Nature Neurosci. 4 (8) (2001) 826–831.

[6] S. Deneve, A. Pouget, Basis functions for object-centered representations, Neuron 37 (2003) 347–359.

[7] S.R. Friedman-Hill, L.C. Robertson, L.G. Ungerleider, R. Desimone, Posterior parietal cortex and the filtering of distractors, PNAS 100 (7) (2003) 4263–4268.

[8] Z. Ghahramani, D.M. Wolpert, M.I. Jordan, Generalization to local remappings of the visuomotor coordinate transformation, J. Neurosci. 16 (21) (1996) 7085–7096.

[9] S. Haykin, Neural Networks. A Comprehensive Foundation, MacMillan College Publishing Company, 1994.

[10] K. Nakamura, H.H. Chung, M.S.A. Graziano, C.G. Gross, Dynamic representation of eye position in the parieto-occipital sulcus, J. Neurophysiol. 81 (1999) 2374–2785.

[11] L. Natale, G. Metta, G. Sandini, A developmental approach to grasping, In Developmental Robotics AAAI Spring Symposium, 2005.

[12] R.C. O'Reilly, Biologically plausible error-driven learning using local activation differences: the generalized recirculation algorithm, Neurocomputing 8 (1996) 895–938.

[13] A. Raffone, C. van Leeuwen, Dynamic synchronization and chaos in an associative neural network with multiple active memories, Chaos 13 (2003) 1090–1104.

[14] E. Sauser, A. Billard, Three dimensional frames of references transformations using recurrent populations of neurons, Neurocomputing 64 (2005) 5–24.

[15] E. Sauser, A. Billard, View sensitive cells as a neural basis for the representation of others in a self-centered frame of reference, In: Proceedings of the Third International Symposium on Imitation in Animals and Artifacts, Hatfield, UK, 2005.

[16] T. Doi, Sony vice president. <www.sony.net/sonyinfo/qrio/interview/>, 2004.

[17] A. van Rossum, A. Renart, Computation with populations codes in layered networks of integrate-and-fire neurons, Neurocomputing 58–60 (2004) 265–270.

[18] C. Weber, S. Wermter, A. Zochios, Robot docking with neural vision and reinforcement, Knowledge-Based Systems 17 (2–4) (2004) 165–172.