Available online at www.sciencedirect.com
SCIENCE DIRECT®
Knowledge-Based Systems 17 (2004) 165–172
Knowledge-Based SYSTEMS
www.elsevier.com/locate/knosys

Robot docking with neural vision and reinforcement

Cornelius Weber*,[1], Stefan Wermter, Alexandros Zochios

Centre for Hybrid Intelligent Systems, University of Sunderland, DGIC, St Peters Way, Sunderland, SR6 0DD, UK

Available online 10 April 2004

Abstract

We present a solution for robotic docking, i.e. approach of a robot toward a table so that it can grasp an object. One constraint is that our PeopleBot robot has a short non-extendable gripper and wide 'shoulders'. Therefore, it must approach the table at a perpendicular angle so that the gripper can reach over it. Another constraint is the use of vision to locate the object. Only the angle is supplied as additional input. We present a solution based solely on neural networks: object recognition and localisation is trained, motivated by insights from the lower visual system. Based on the hereby obtained perceived location, we train a value function unit and four motor units via reinforcement learning. After training the robot can approach the table at the correct position and in a perpendicular angle. This is to be used as part of a bigger system where the robot acts according to verbal instructions based on multi-modal neuronal representations as found in language and motor cortex (mirror neurons).
© 2004 Elsevier B.V. All rights reserved.

Keywords: Reinforcement learning; Unsupervised learning; Multi-modal representation; Hybrid model

1. Introduction

There have been a lot of insights into neural networks and the way the brain works. Also, there have been simulations of how such networks can perform interesting tasks. But when it comes to robotic implementation, traditional artificial intelligence methods are still dominant. One reason for this is that both sophisticated perception and motor skills need to be combined in order to display non-trivial behaviour. However, these cannot be achieved with a single algorithm/network type. Therefore, we propose a hybrid network trained with reinforcement, unsupervised and supervised training to perform a vision-based docking action.

For docking, usually non-trainable models of vision are used such as optic flow from log-polar vision [1], correlation operators on search templates [11] or other devices such as laser range finders [10,14] are used as sensory input. Models of grasping need more sophisticated geometrical information about the object, and use vision algorithms based on graph matching [2], Gabor jets [9] or 3D geometrical object models [8]. The control scheme employed is usually based on geometrical calculations. Also a reinforcement solution for docking has been presented [6] in which to pre-process the input a neural gas was used for clustering and a neural field for topological action coding. Visual goal recognition was done with colour-based threshold operations.

Many of the docking or grasping scenarios are part of larger projects or goals such as implementing a perceptually guided robot [2] and using hand gestures for robot teaching [9]. Emphasis is put on high-speed performance [8], recharging batteries [10], using embodied representations [1] or addressing top-level control issues [11]. In our case, higher-level mirror neuron behaviour shall be modelled (see Section 4). Therefore, we seek a neural network representation of a complex behaviour to obtain realistic input to the envisaged higher level.

Our robot (Fig. 1) has a single behaviour: in any state it selects the action which leads to the largest expected reward. This makes the action selection network the core part. It consists of four neurons, one of them is 'on' at any time, which denote forward, backward, left and right movement of the robot. During training, they are guided by the firing rate of one 'value function' unit which assigns a fitness value to any state. Together, these five neurons are trained by reinforcement learning, in which a scalar reinforcement signal is given only at the end of each training action sequence. The value of the signal is positive,

* Corresponding author. Tel.: +44-191-515-3274; fax: +44-191-515-3461.

E-mail addresses: cornelius.weber@sunderland.ac.uk (C. Weber); stefan.wermter@sunderland.ac.uk (S. Wermter); a.zochios@talk21.com (A. Zochios).

[1] Web: www.his.sunderland.ac.uk.

0950-7051/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2004.03.012

Fig. 1. The PeopleBot robot during the docking manoeuvre. On the left are visible the camera pointing downward (mounted underneath the top plate) and the black grippers. Right, the scenario from above. The black grippers are hardly to be seen over the dark robot's base; a fraction of the camera can be seen bright.

if the robot docks at the object in parallel to the table, or negative, if the robot's shoulders bump into the table at an angle or if the object is lost out of sight (Fig. 2).

The input to the action selection network is the robot's visual perceptual state, defined by its relative position to the target, an orange fruit at the border of a table. The vision module is thus the peripheral part. Vision skills are trained unsupervised as well as supervised. Unsupervised training leads to a sparsely coded hidden representation of an input image. The perceptually important representation of the target object within the image is then trained in a supervised manner: a recurrent associator neural network learns to associate the internal representation of the entire image with the (given) position of the target object. Additional input to the action selection network is the robot rotation angle φ, supplied by the robot's internal odometry.

2. Methods

The peripheral vision module is trained before the action selection network so that it can supply it the necessary visually obtained perception as input. Overall, we have three training phases: first, training the weights W^{td} and W^{bu} between the visual input and the 'what' area (Fig. 3), second, training the lateral weights W^{lat} within and between the 'what' and the 'where' area, and finally, training the weights W^c and W^m from the conceptual space to the critic of the motor outputs, respectively.

2.1. Training the vision module—feature detectors

In the first phase, we will obtain feature detector neurons on the 'what' area to have a more abstract, higher level representation $\vec{u}$ of the input image $\vec{I}$ (Fig. 3). This is done based on the idea that the model should generate the data $\vec{I}$ from a sparse representation $\vec{u}$. This is done by the wake-sleep algorithm [7], in which two learning steps are alternatingly repeated until training has completed: (i) in the 'wake phase', train the top-down, generative weights W^{td} based on the difference between a randomly chosen natural image $\vec{I}^{orig}$ and its reconstruction $\vec{I}^{rec}$ obtained from the internal representation $\vec{u}$ of the picture and (ii) in the 'sleep phase', train the bottom-up, recognition weights W^{bu} based on the difference between a random hidden code $\vec{u}^{orig}$ and its reconstruction $\vec{u}^{rec}$ obtained from the visual representation $\vec{I}$ of the original hidden code. Training involves only local learning rules and results in localised edge detectors akin to the simple cells of visual area V1. With additional modifications to the learning algorithm the mapping is also topographic.

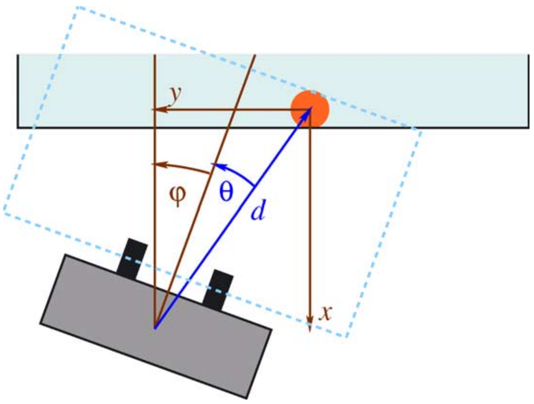

Fig. 2. Geometry of the scenario in top view. The figure depicts the table, above, with the orange fruit target on it (filled circle). Below is the robot with its short black grippers. The rectangular field that is visible from the robotic camera facing downward is outlined by a dotted line. Real world coordinates (x, y, φ) specify the position and rotation angle of the robot. The perceived position of the target within the robot's visual field is then defined by the perceived angle θ and distance d.

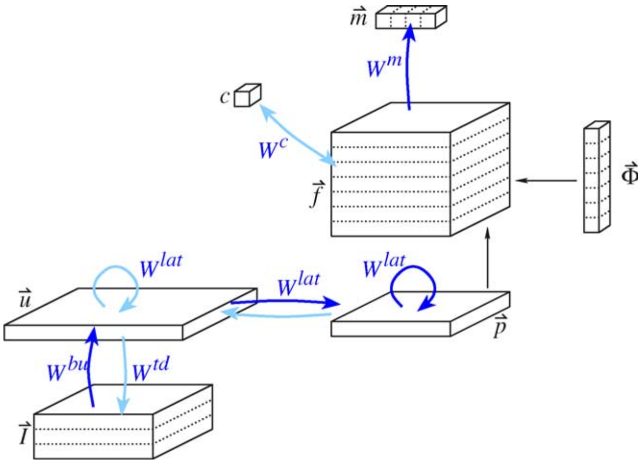

Fig. 3. The neural network. Thick arrows denote trained weights W. Only the ones depicted dark are used during performance while those depicted bright are involved in training. Letters other than W denote activations (vectors) on the neural sheets: $\vec{I}$ is the camera image which contains three layers for its red, green and blue colour components. $\vec{u}$ is the hidden representation ('what') of the image. $\vec{p}$ contains the perceived location ('where') of the target within the image. $\vec{\Phi}$ has a Gaussian profile centred on the rotation angle φ of the robot. $\vec{f}$ is the perceptual space, made up by multiplying $\vec{p}$ and $\vec{\Phi}$. c, the critic, holds the value function which is assigned to each perceptual state $\vec{f}$ unit. $\vec{m}$ are the four motor unit activations.

The following pseudo-code describes the training algorithm in which the wake phase and the sleep phase alternate each other repeatedly. Wake phase:

1. Take a picture $\vec{I}^{\text{orig}}$.
2. Get a sparse hidden representation on the 'what' area $\vec{u}$.
3. Reconstruct the picture $\vec{I}^{\text{rec}}$.
4. Top–down weight update: $W^{\text{td}} \approx (\vec{I}^{\text{orig}} - \vec{I}^{\text{rec}})\cdot\vec{u}$.

Sleep phase:

1. Generate a sparse, topographic random hidden code $\vec{u}^{\text{orig}}$.
2. Get the imagined picture $\vec{I}$.
3. Reconstruct the hidden code $\vec{u}^{\text{rec}}$.
4. Bottom-up weight update: $W^{\text{bu}} \approx (\vec{u}^{\text{orig}} - \vec{u}^{\text{rec}})\cdot\vec{I}$.

This algorithm, described in detail in Ref. [12], approximates the Helmholtz machine [3]. Fig. 4, left, shows examples of trained weights, most of which have become localised edge detectors, while some neurons are colour selective.

2.2. Training the vision module—object localisation

The second phase, training the lateral weights W^{lat} between and within the 'what' and 'where' areas (Fig. 3), requires the first phase to be completed. Intra-area lateral connections within the 'where' area (visual area V1) were originally implemented to endow the simple cells with biologically realistic orientation tuning curves: their orientation tuning curves were sharpened via competition, mediated by the lateral weights. In addition, shift invariances were trained and thus V1 complex cells generated [12]. The function of the lateral weights is to memorise the underlying representation over time. As an attractor of a real-valued recurrent network, the representation is thereby simplified. We exploit this for pattern completion where the representation $\vec{u}$ of an image with an object of interest is given on the 'what' area while its location $\vec{p}$ on the where area is not given—while it has always been given during training.

Training is done by the following procedure: every image $\vec{I}$ now contains a simulated orange fruit at a particular location and this location is reflected—in a supervised manner—as a Gaussian on the 'where' area. So the lateral weights are trained to memorise the internal representation $(\vec{u},\vec{p})$ of the image and the location of the orange. After training, when we have the representation $\vec{u}$ of an image with an orange but do not know the location of the orange. Then pattern completion will give us its location, coded in $\vec{p}$.

The following pseudo-code describes a step of the training algorithm, which is repeatedly applied after the 'what' network has been trained.

1. Take a natural image with an orange placed at location $\vec{L}$.

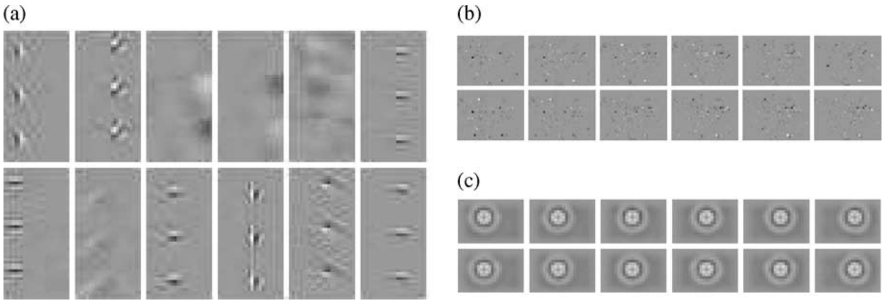

Fig. 4. A selection of trained weights of the vision module. Dark shades of grey denote negative; bright, positive weights. (a) The receptive fields of 12 'what' units, taken from the centre of W^{bu}. Each unit has three sets of weights to the red, green and blue sub-layers of the input. Three of the upper units are colour selective, as the weights are different in different sub-layers. (b) and (c) The receptive fields of 12 'where' units, taken from those parts of W^{lat} which are depicted dark in Fig. 3. The weights from the 'what' to the 'where' area, (b), are sparse. The recurrent weights within the 'where' area, (c), are centre-excitatory and surround inhibitory, because they were trained to maintain a Gaussian activity profile. Self-connections were set to zero.

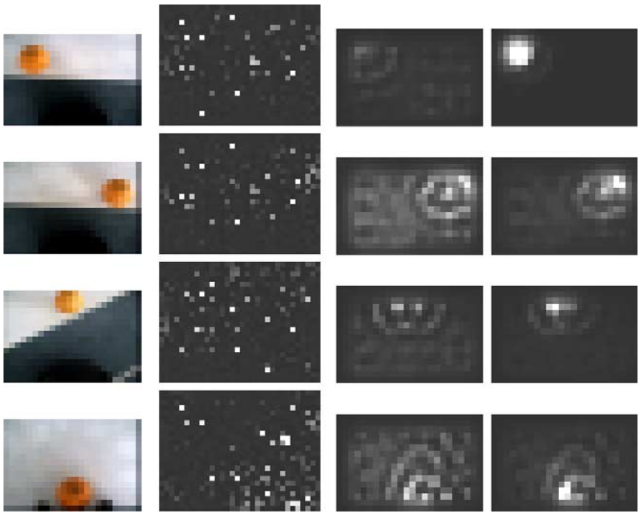

Fig. 5. Each row shows, left, the 24 × 16 pixel camera image, which is originally in color. Then its representation $\vec{u}$ on the 'what' area. Active units are bright. The third and fourth picture in each row are the representation $\vec{p}$ on the 'where' area at the first and the last time step of a 10 iteration relaxation. The last row corresponds to the goal position where the orange is between the tips of the gripper.

2. Get the hidden representation on the 'what' area $\vec{u}$.
3. Initialise $\vec{p}^L$ on the 'where' area to contain a Gaussian at location $\vec{L}$.
4. Initialise activities on 'what' and 'where' areas as: $\vec{A}^{\text{orig}} = \{\vec{u}, \vec{p}^L\}$.
5. Relaxate activations using W^{lat} for a couple of steps; memorise as $\vec{A}^{\text{attrac}}$.
6. Weight update: $W^{\text{lat}} \approx (\underbrace{\vec{A}^{\text{orig}} - \vec{A}^{\text{attrac}}}) \cdot \vec{A}^{\text{attrac}}$.

The under-braced term is the association error between the desired state and the one memorised as an attractor. Since $\vec{p}^L$ is not naturally contained in the data but produced artificially, training is supervised. Details and parameters are given in Ref. [13].

Trained weights are shown in Fig. 4, (b) and (c), while Fig. 5 demonstrates their performance at object localisation. The representation $\vec{p}$ on the 'where' area is at the first time step (third column) purely a result of the feed-forward input from $\vec{u}$ from the 'what' area. After relaxation (right column), recurrent connections W^{lat} within the 'where' area have cleaned up the representation (while $\vec{u}$ was fixed).

2.3. Reinforcement training of the action module

In the last phase, we apply reinforcement learning to the weights W^m of the motor units and the weights W^c of the value function unit. Their common input is the robot's own perceived state $\vec{f}$ which is different for every different visually perceived target location $\vec{p}$ and every different robot rotation angle $\vec{\varphi}$. The representation of $\vec{p}$ is multiplexed over, here seven, layers to obtain $\vec{f}$ (Fig. 3). Each layer corresponds to a rotation angle of the robot. Only the layer(s) nearby the actual angle have non-zero activity (Eq. (2)). The weights W^c assign each state $\vec{f}$ a critic value c which is initially positive only at the goal: in our case when the target is perceived in the middle of the lower edge of the visual field and when the robot rotation angle φ is zero. During performance, states that lead quickly to the goal will also be assigned a higher value c by strengthening their connections to the critic unit. The weights W^m to the motor units which have been activated simultaneously are also increased, if the corresponding action leads to a better state, i.e. one which is assigned a larger c. The algorithm has been described for a rat navigation task in Ref. [4]; in the following, we will give details of our implementation.

2.3.1. World and perception model

Reinforcement training of the weights W^c and W^m involves as input the perceptual state $\vec{f}$ and as outputs the value c and motor action $\vec{m}$. All these values can be simulated to avoid costly real robotic actions. The simulation runs with 'real' world coordinates from which the perceived state $\vec{f}$ can easily be computed. The real world coordinates (x, y, φ) (Fig. 2) are updated based on the movement commands contained in the output vector $\vec{m}$ for the robot speed v and the rotation speed $\dot{\varphi}$:

$$x(t+1) = x(t) - v \cdot \Delta t \cos(\varphi)$$

$$y(t+1) = y(t) - v \cdot \Delta t \sin(\varphi) \tag{1}$$

$$\varphi(t+1) = \varphi(t) + \dot{\varphi} \cdot \Delta t$$

Motor units $i = 1$ and 2 set the velocity v to 0.9 and -0.9, respectively. Units 3 and 4 set the angular velocity $\dot{\varphi}$ to 0.1 and -0.1, respectively.

Using the relation $(y/x) = \tan(\varphi + \theta)$, we get the robot's perceived angle θ and distance d to the target (Fig. 2)

$$\theta = \arctan\left(\frac{y}{x}\right) - \varphi, \quad d = \sqrt{x^2 + y^2}$$

which is used to draw the perceived target onto the simulated vision input. As a shortcut, instead of drawing a simulated orange fruit to the vision input area, we directly placed a Gaussian onto the 'where' area as activation pattern $\vec{p}$.

We expand the representation $(\vec{p}, \varphi)$ so that every different combination of values leads to a different state $\vec{f}$ in the expanded perceptual space. First, φ is transformed into a 7-dimensional vector $\vec{\Phi}$ which represents values of φ between -45 and 45° as the centre of its Gaussian activity profile. $\vec{\Phi}$ is thus an ensemble of heading direction (robot rotation angle) cells, which contain the information about φ as a population code. The perceptual state vector $\vec{f}$ is a product of the perceived target location $\vec{p}$ and the heading direction vector $\vec{\Phi}$:

$$f_{ijk} = p_{ij} \cdot \Phi_k, \quad i = 1\ldots24,\ j = 1\ldots16,\ k = 1\ldots7 \tag{2}$$

which is a Gaussian in a 24 × 16 × 7-dimensional cube. Two examples of $\vec{f}$ are depicted in Fig. 6, right.

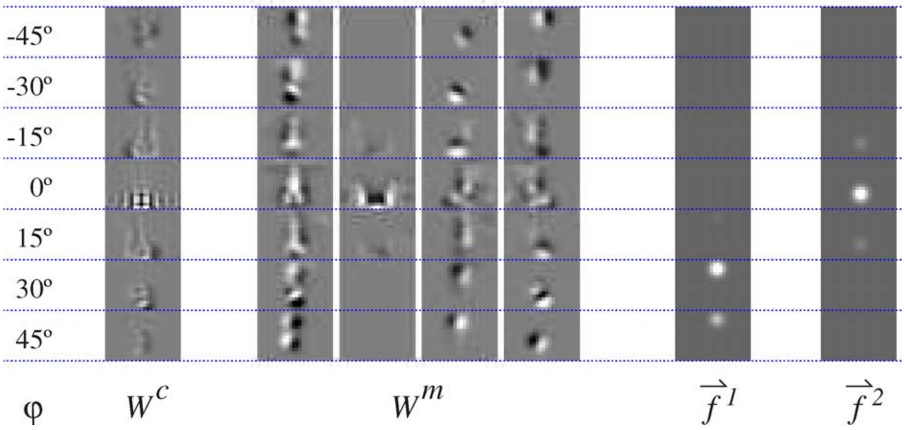

Fig. 6. The weights W^c and W^m after reinforcement training, and example activity patterns $\vec{f}$. Positive connections $W^{c,m}$ are white, negative connections dark. Each column shows the connections to one recipient unit from the perceptual space in which $\vec{f}$ resides: it consists of seven 24×16 sized fields, each of which is devoted a target object representation when the robot is at the angle φ, given left. The four units which receive W^m encode motor movements for-from left to right-forward, backward, left turn and right turn. The activation $\vec{f}^1$ corresponds to a situation where the robot is at an angle slightly larger than 30° and sees the target at the upper (distant) edge of the visual field (this corresponds roughly to the turning point in Fig. 7, the situation in Fig. 2 or the percept in the third row in Fig. 5). $\vec{f}^2$ is near the goal position, as the robot sees the target right in front and has an angle $\varphi = 0$ (fourth row in Fig. 5).

The condition whether the robot 'shoulders' hit the table is tested in (x, y, φ)-space. If the distance x (of the front middle of the robot) from the table is smaller than the absolute value of $\sin(\varphi)\cdot wh$ with wh the half-width of the robot, then one of the robot edges would intrude the table. This constraint in x and φ translates implicitly to the robot's perceptual space $\vec{f}$ and the robot will learn through the negative 'reward' to avoid this region.

2.3.2. Reinforcement algorithm

A trial begins by setting the robot to a random initial position (x, y, φ). We have to make sure that the target is visible and, for simplicity, we set the robot in parallel to the table, i.e. $\varphi = 0$. Within one trial, the following steps are performed until a non-zero reward signal R is given. Each step involves one motor action and the reading of perceptions before and after.

1. Compute the perceived target $\vec{p}$ and from this, the perceived state $\vec{f}$.
2. Compute the critic activation: $c = \sum_j w_j^c \cdot f_j$.
3. Compute the probability $P(m_i = 1)$ for motor unit i to be active:

$$P(m_i = 1) = \frac{e^{2a_i}}{\sum_{i'} e^{2a_{i'}}}, \quad \text{with } a_i = \sum_j w_j^m \cdot f_j \qquad (3)$$

   The probabilities sum up to one over the motor units. One unit is set active.
4. Move the simulated robot according to its motor output, using Eq. (1).
5. Compute the perceived target location $\vec{p}'$ and state $\vec{f}'$.
6. Compute the critic activation: $c' = \sum_j w_j^c \cdot f_j'$.
7. Set the reward signal:

$$R = \begin{cases} 1 & \text{if goal reached } (\varphi = 0 \textbf{ and } \text{target centred at lower edge of visual field}), \\ -0.3 & \text{if target at visual field border } \textbf{or} \text{ robot hits table}, \\ 0 & \text{else.} \end{cases}$$

8. Compute the prediction error: $\delta = R - (c - \gamma \cdot c')$ between the actual reward R and the critic evaluation $c - \gamma \cdot c'$. The critic evaluation is based on the assumption that the value function increases in time with future values decaying by the discount factor $\gamma = 0.9$:

$$c(t) = R(t) + \gamma \cdot R(t+1) + \gamma^2 \cdot R(t+2) + \cdots.$$

9. Update the critic's weights: $\Delta w_j^c \propto \delta \cdot f_j$.
10. Update the weights of the only active motor unit i : $\Delta w_{ij}^m \propto \delta \cdot m_i \cdot f_j$.

Each trial thus constitutes a robot's experience about either reaching the goal or loosing the target or hitting the table. It consists of a sequence of actions and several learning steps. The more trials have been done, the better the robot performs, and the shorter each will be, and the more likely they will end with a positive reward.

Note that in order to learn every weight, it is not necessary that every perceptual state has occurred. The state description $\vec{f}$ is made up of a Gaussian covering several state space units simultaneously (see Fig. 6, right, for a visualisation). A critic weight w_j^c is thus updated similar to weight $w_{j'}^c$ if j and j' are neighbours in the perceptual space; analogously motor weights w_{ij}^m and $w_{ij'}^m$. This topological relation exploits the fact that similar states imply similar optimal actions.

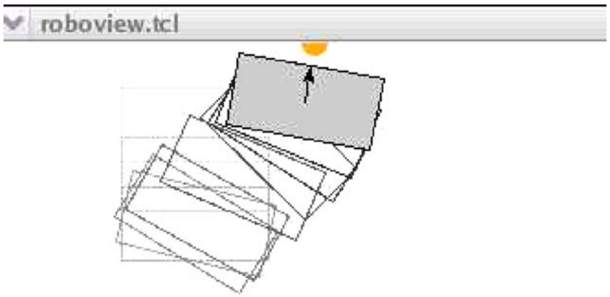

Fig. 7. The simulated trained robot during the docking manoeuvre. The upper bar corresponds to the table location and the half-circle represents the target. The robot's grippers (not depicted) are near the front of the arrow shown on the robot. Outlines of previous poses are in brighter grey. After start, the robot first moved backward in order to turn and then approach the target. Shown is a screen capture of a graphical user interface.

3. Results

The weights obtained by reinforcement learning are shown in Fig. 6. The weights W^c to the value function unit are over-trained as can be seen from their wiggly structure around the goal at angle zero. At earlier stages they are smooth and positive around the goal position. Nevertheless, training was continued so that the simulated robot has reached its goal thousands of times. The motor weights W^m are seemingly unaffected by over-training—their shape does not change noticeably. However, their absolute values continue to grow. As an effect, the motor unit outputs become more deterministic (cf. Eq. (3)).

The structure of the motor weights W^m is complex and only partly obvious. It is obvious in the simple situation where the robot perceives the target in front of him while having a rotation angle of $\varphi = 0$. This corresponds to the perceptual input $\vec{f}^2$ in Fig. 6. The weights to motor unit one (left column of W^m in Fig. 6) in this area are positive (white) so to excite the 'forward' unit. The 'backward' motor unit (second column of W^m) has inhibitory connections (black) originating from this area, thus suppressing its response. The 'backward' motor unit, however, has positive (white) connections to the sides of this position, so that the robot moves back if the target is perceived nearby to the left or to the right (only at a rotation angle $\varphi = 0$).

3.1. Simulated robot performance

A successful example of simulated docking performance is depicted in Fig. 7 which displays a surprisingly complex and successful movement. We have observed the performance limits to be reached, if the offset (y, Fig. 2) to the target is large. This extreme case leads to two possible actions: first, a movement at which the perceived target reaches the border of the visual field or second, a seemingly successful drive toward the target, but in a narrow angle (φ large) so that the robot's 'shoulders' eventually touch the table. Success in these cases might be possible with a complex strategy involving small movements forth and back including turning, but is not discovered by the algorithm, possibly because of the rough discretisation of

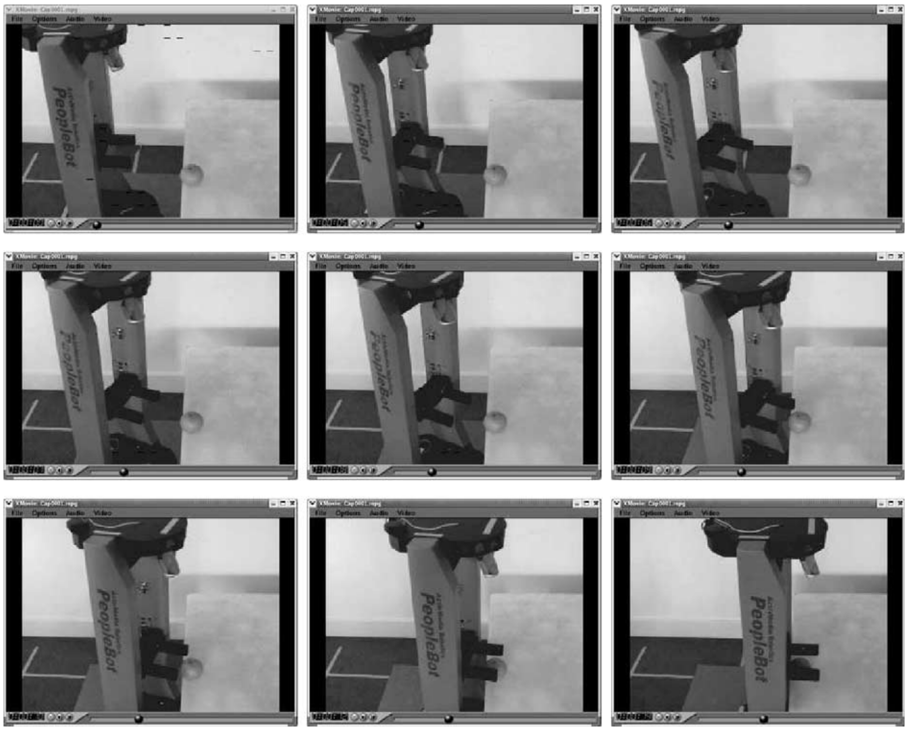

Fig. 8. Snapshots from a docking sequence. The video can be seen at: `http://www.his.sunderland.ac.uk/robotimages/Cap0001.mpg`.

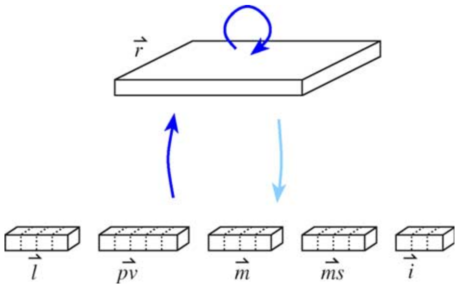

Fig. 9. The envisaged mirror neuron network. Mirror neuron properties are expected to evolve among some of the neurons in the top layer. They carry an internal representation $\vec{r}$ of all of the inputs, below. The inputs are from multiple modalities including higher-level representations. The vector $\vec{l}$ contains representations from language areas. $\vec{pv}$ contains the visual perception which includes the identity and perceived location of a target to be grasped. $\vec{m}$ are the motor unit activations including wheels, pan-tilt camera and gripper. $\vec{ms}$ denotes motor sensory unit activations and may also include available idiothetic information such as the rotation angle φ of the robot. $\vec{i}$ are other internal states such as the value function of the critic.

the angle space, in which φ is represented by only seven increments.

3.2. Application to the PeopleBot robot

Without having done any training on a real robot, we used the trained network successfully for robotic control. Only a subset of all weights (displayed dark in Fig. 3) is used during performance. The image $\vec{I}$ is now taken from the robotic camera that looks down at a fixed angle to the space in front (Figs. 1 and 8). The robot orientation angle φ is taken from a robot-internal proprioceptive update mechanism (the starting angle is always zero). Finally, the outputs are directed to the wheels, the control of which accepts values for speed and rotational speed.

The bottleneck of our application is vision: the recognition of the orange fruit which constitutes the target is brittle and disturbed, for example, by large contrasts in the surrounding. In addition, we have a distortion of vision, because the camera does not point exactly vertically. Thus the physical model, Eq. (1), is only a rough approximation. This, however, does not matter, first, because in any case the speed values must be adjusted to reasonable values. Secondly, because the concept of assigning one motor output to every state works even, if the speed is too slow and the state at the next time step remains the same: then the motor directive will simply remain, until eventually, another state is reached.

Another restriction is the small size of the visual field, limited by the narrow position of the camera and its maximal zoom. A narrow table must be used to increase the visual field—a too large visual field again would lead to problems in target recognition. Finally, we have not yet implemented the command to close the gripper at arrival at the target.

4. Discussion

In this paper, we demonstrated that purely neural network based vision and control algorithms can successfully be applied to a real robotic docking problem.

Currently, we are developing a higher-level associative network in which mirror neurons shall emerge (Fig. 9). Mirror neurons have been found in motor and language cortex and fire either when an action is performed or when it is observed, or both [5]. The network receives information from multiple modalities and represents them as a hidden code $\vec{r}$. The vertical connections are trained with a sparse coding unsupervised learning scheme similar to the Helmholtz machine described earlier in this paper. The inputs are collected from robotic actions, which are performed interactively in the environment. The data contain only instantaneous information, i.e. the whole sequence of actions is not known. Therefore, neurons do not necessarily fire over a sustained period in time as do mirror neurons. However, since $\vec{r}$ is a distributed code, some units may specialise to code for longer sequences. The horizontal recurrent connections (depicted as open circle) are trained as an associator neural network. They are used in a neural activation relaxation procedure which is expected to (i) clear noise of the representation $\vec{r}$, (ii) predict the hidden code of the next time step and (iii) display prolonged firing. As a possible extension, associator recurrent connections may also feed back to the input, acting as a forward model. This would be particularly interesting for the cortical feedback to the motor units, because of implications for motor control: after repetitive exercising this network might be able to perform the action sequence. This would render superfluous the reinforcement trained part with its large perceptual state $\vec{f}$.

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] N. Barnes, G. Sandini, Direction control for an active docking behaviour based on the rotational component of log-polar optic flow, in: ECCV2000—Proceedings of the European Conference on Computer Vision, vol. 2, 2000, pp. 167–181.

[2] M. Becker, E. Kefalea, E. Mal, C. von der Malsburg, M. Pagel, J. Triesch, J.C. Vorbrggen, R.P. Wrtz, S. Zadel, Gripsee: a gesture-controlled robot for object perception and manipulation, Autonom. Rob. 6 (1999) 203–221.

[3] P. Dayan, G.E. Hinton, R. Neal, R.S. Zemel, The Helmholtz machine, Neur. Compd. 7 (1995) 1022–1037.

[4] D.J. Foster, R.G.M. Morris, P. Dayan, A model of hippocampally dependent navigation, using the temporal difference learning rule, Hippocampus 10 (2000) 1–16.

[5] V. Gallese, L. Fadiga, L. Fogassi, G. Rizzolatti, Action recognition in the premotor cortex, Brain 119 (1996) 593–609.

[6] H.M. Gross, V. Stephan, M. Krabbes, A. neural, field approach to topological reinforcement learning in continuous action spaces, in: Proceedings of the WCCI-IJCNN, 1998.

[7] G.E. Hinton, P. Dayan, B.J. Frey, R. Neal, The wake-sleep algorithm for unsupervised neural networks, Science 268 (1995) 1158–1161.

[8] A. Namiki, Y. Nakabo, I. Ishii, M. Ishikawa, High speed grasping using visual and force feedback, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1999.

[9] H. Ritter, J. Steil, C. Noelker, F. Roethling, P. McGuire, Neural architectures for robotic intelligence, Rev. Neurosci. (2003).

[10] M.C. Silverman, D. Nies, B. Jung, G.S. Sukhatme, Staying alive: a docking station for autonomous robot recharging, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2002.

[11] J. Spofford, J. Blitch, W. Klarquist, R. Murphy, Vision-guided heterogeneous mobile robot docking, in: Sensor Fusion and Decentralized Control in Robotic Systems II, 1999.

[12] C. Weber, Self-organization of orientation maps, lateral connections, and dynamic receptive fields in the primary visual cortex, in: G. Dorffner, H. Bischof, K. Hornik (Eds.), Proceedings of the ICANN, Springer, Berlin, 2001, pp. 1147–1152.

[13] C. Weber, S. Wermter, Object localization using laterally connected what and where associator networks, in: Proceedings of the ICANN, 2003, pp. 813–820.

[14] M. Williamson, R. Murray-Smith, V. Hansen, Robot docking using mixtures of gaussians, Adv Neural Inform Process Syst 11 (1999) 945–951.