ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Learning visuo-motor coordination for pointing without depth calculation

Ananda Freire[1,2], Andre Lemme[2], Jochen Steil[2], Guilherme Barreto[1]

1- Federal University of Ceará - Dept.of Engineering of Teleinformatics
Av. Mister Hull, S/N - Bloco 725, Fortaleza - Brazil

2- Institute for Cognition and Robotics (CoR-Lab) & Faculty of Technology
Bielefeld University, 33615 Bielefeld - Germany

Abstract.
Pointing refers to orienting a hand, arm, head or body towards an object and is possible without calculating the object's depth and 3D position. We show that pointing can be learned as holistic direct mapping from an object's pixel coordinates in the visual field to joint angles, which define pose and orientation of a human or robot. To this aim, we record real world and noisy training images together with corresponding robot pointing postures for the humanoid robot iCub. We then learn and comparatively evaluate pointing with an multi-layer perceptron, an extreme learning machine and a reservoir network, but also demonstrate that learning fails at reconstructing the depth of trained objects.

1 Introduction

Learning of visually guided behavior and sensory-motor hand-eye coordination have been studied in cognitive robotics mainly in the context of reaching to a point in 3D coordinates. Any cognitive agent, however, also performs pointing either by moving arms or hand in active gesturing or by orienting the body or the head towards a relevant direction in the environment, for instance for shaking hands or talking to communication partner. In addition, pointing relates to a variety of behavioral responses, for example: a protective reflex (to intercept a dangerous object) or blocking something from the sight (a bright light) [1, 2].

Typically reaching and pointing are considered together and there is evidence for a sensory-motor mapping that allows us to reach, to point and to manipulate objects [3]. To learn such mappings, data driven neural network approaches have been proven to be useful on many platforms and scenarios with different learning rules and architectures. Examples are visual servo control [4], a direct mapping (feedfoward) approach [5], or a self-organizing map [6]. But the common point is the focus on reaching tasks and 3D positioning and to use more than just image features as input (e.g. joint angles, 3D coordinates in space, or incrementally acquired information from corrective movements). Other model-based systems make use of methods that are complex [7] and often involve separate stereo-matching algorithms for depth calculation that require precise calibration of the camera system and computationally expensive search for the best stereo match.

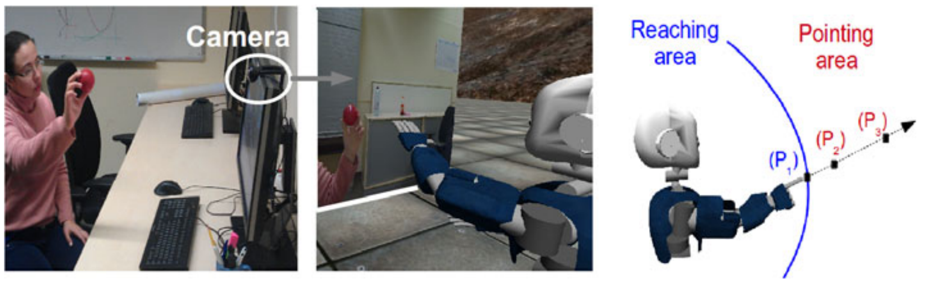

Fig. 1: On the two firsts images, the iCub's simulator tracks and points to a moving ball. On the third image is shown the injectivity characteristic of the mapping from pixel coordinates of both cameras ($\mathbf{u} = (i_L, j_L, i_R, j_R)^T$) to the joint angles ($\mathbf{q} = (\theta_1, ..., \theta_7)$) of the robot's left arm.

A direct mapping from pixel coordinates to 3D position can typically not be learned, due to the well known fact that the stereo-matching is ill-posed.

Pointing, however, does not imply a need for depth calculation, as only the orientation along some direction needs to be adjusted (Fig. 1). This distinguishes pointing from reaching and motivates that a direct neural implementation of this very basic human skill could nevertheless be possible. Previous work of Marjanović et al (1996) and Shademan et al (2009) explored pointing, but both use visual information as an error signal in a closed loop controller. Marjanović makes use of two mappings (from images coordinates to the eye motors, and then to the coordinates of arm motors) and Shademan does not implement neural networks and uses a camera mounted on the end effector (eye-in-hand configuration). In this work, we directly approach learning of pointing by means of connectionist networks, such as the Multilayer Perceptron (MLP), the Extreme Learning Machine (ELM) and in the framework of Static Reservoir Computing (SRC), using only pixels coordinates as input and joint angles as output.

2 Neural Networks

This section gives an overview of the main characteristics of the neural architectures used in this work.

Multi-Layer Perzeptron (MLP) - Since the MLP is a well known network, it won't be described in detail. Here the joint angles are transformed to radians and the input is normalized to $[-0.9, 0.9]$. For weight adaptation, we used batch backpropagation with hyperbolic tangent as activation function. The least number of hidden neurons that provided the best performance is 3, with learning rate of 0.005 and needed 222 epochs for stabilizing the training error.

Extreme Learning Machine (ELM) - The ELM is a three layer network with randomly and fixed input weight matrix connecting the input layer with the hidden layer, $\mathbf{W}_{inp}$ [8]. The weight matrix to the output layer, $\mathbf{W}_{out}$, can be determined using pseudo-inverse with Tikhonov regularization. Thus, the mapping is given by $\mathbf{y} = f\left(\mathbf{W}_{inp}^T\mathbf{u}\right), \mathbf{v} = \mathbf{W}_{out}\mathbf{y}$, where $\mathbf{u}(t) \in \mathbb{R}^p$ is the current input vector, T indicates transposition and f is a logistic activation function. For estimating $\mathbf{W}_{out}$, we assume that a training sequence $((\mathbf{u}(k), \mathbf{v}(k)), k = 1...N$ is available. All inputs are presented to the network and the corresponding network states $(\mathbf{y}(k), \mathbf{v}(k))$ are collected (harvested). Mapping to the desired output is enforced by mapping states $\mathbf{y}$ to outputs $\mathbf{v}$. This can be accomplished by the linear regression

$$\mathbf{W}_{out} = (\mathbf{Y}^T\mathbf{Y} + \alpha\mathbb{1})^{-1}\mathbf{Y}^T\mathbf{V} \tag{1}$$

collect all hidden states and desired outputs in respective matrices. The parameter α in (1) weights the contribution of a regularization constraint.

For improving ELM's performance, the Intrinsic Plasticity (IP) approach is used. It is an unsupervised online learning rule that adapts bias (b_i) and slope (a_i) of the neuron's activation function, tuning them into more suitable regimes, maximizing information transmission and acting as a feature regularizer [9]. In this paper, a variation of this approach for batch learning is used. This adaptation is done in a way that the desired exponential distribution f_{des} for the neurons activation $y_i(k) = (1 + \exp(-a_i x_i(k) - b_i))^{-1}$ is realized. For each hidden neurons, all the arriving synaptic sum $\mathbf{x}_i = \mathbf{w}_i^T\mathbf{U}$ is collected, where $\mathbf{U} = (\mathbf{u}(1), ..., \mathbf{u}(N))$. Then random targets $\mathbf{t} = (t_1, ..., t_N)^T$ from the desired output distribution, and the collected stimuli are drawn in ascending order. The model $\Phi(\mathbf{x}_i) = (\mathbf{x}_i^T, (1, ...1)^T)$ is built so we can calculate $(a_i, b_i)^T = (\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x}_i) + \alpha\mathbb{1})^{-1}\Phi(\mathbf{x}_i)^T f^{-1}(\mathbf{t})$, where f^{-1} is the inverse logistic function [10].

In the experiments explained below the following parameters are used. The input weight matrix $(\mathbf{W}_{inp})$ is initialized within a normal distribution between $[-0.5, 0.5]$ with density of 50%. The hidden layer consists of 30 neurons with activation functions adjusted by batch IP with $\mu = 0.2$. The linear regression parameter is set to $\alpha = 0.01$.

Static Reservoir Computing (SRC) - Also known as attractor-based computation with reservoirs, comprises a recurrent network with nonlinear hidden neuron, that are randomly and sparsely chosen and remain fixed. The network's dynamics can be described as:

$$\mathbf{y}(k) = f(\mathbf{W}_{inp}\mathbf{u}(k) + \mathbf{W}_{res}\mathbf{y}(k-1)) \tag{2}$$

The output is also described by $\mathbf{v} = \mathbf{W}_{out}\mathbf{y}$. The transients are removed by iterating the dynamics, with a clamped input pattern $\mathbf{u}(k)$ until the network state is converged [11]. The SRC's reservoir size has 50 neurons 20% of weights is different from zero, with values between $[-1, 1]$ and spectral radius of 0.99. The inputs were normalized to [0.1,0.9], $\mathbf{W}_{inp}$ is initialized like the ELM input matrix and the regression parameter is $\alpha = 10^{-6}$.

All the networks implemented have only one single hidden layer with parameters defined empirically. They are trained and tested 50 times, the network with the best performance is used in the later evaluation.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

3 Learning to point

Recording of training data. Our goal is to map pixel coordinates obtained from binocular vision directly to joint angles, which maps all positions along a particular pointing direction to one pointing posture (Fig. 1, right). That means, the target hand position is projected to the outer hull of the 3D egoshpere which surrounds the robot and defines its reachable area.

Training data is created by presenting a red ball within the visual field of the robot in a cube with ca. 70 cm edge length, which is outside of the iCub's workspace. Corresponding arm postures of the iCub trying to reach the ball with the left arm are recorded, which effectively implements a pointing because the arm cannot reach positions outside the egoshpere. We use state of the art methods available on iCub's software repository, which reconstruct from the images first the 3D position of the ball through stereo matching and then perform a respective arm movement. The following data is collected: pixel coordinates of the ball from both cameras (i_L, j_L) and (i_R, j_R), ball coordinates (x_b, y_b, z_b), joint angles $(\theta_1, ..., \theta_7)$ and end effector position (x_e, y_e, z_e). The dataset for 491 points was divided in 391 samples for training and 100 for testing.

Performance evaluation. Two measures are computed: First, the positioning error of the hand when driven by the network's output. Second, and more interesting, the distance between two points generated by the projection of the ball's position on the pointing ray of the iCub's training pose and the network's response. In Tab. 1 the mean and variance (var) for training and testing errors are displayed.

From Tab. 1 can be noticed that the errors in hand positioning are on average below two centimeters for the best performance. This is quite reasonable qualitatively, given the high amount of noise in the training data, which originates mostly in a large variance of the computation of the 3D position of the ball performed by the iCub. Nonetheless even a small positioning error can create a large gap between its respective pointing direction and the desired one. The pointing accuracy error shows an average around 6cm for the SRC approach. The error increases with increasing distance of the ball and the results have therefore a large variance. The SRC network performs best in both measures, which is plausible as it provides the most complex features through its non-linear

	MLP		**ELM**		**SRC**	
Training	*mean*	*var*	*mean*	*var*	*mean*	*var*
hand position.	2.5367	2.5752	1.7476	1.6445	1.5409	1.4692
pointing accuracy	7.3904	20.5445	6.3959	18.0396	5.9961	16.4145
Testing	*mean*	*var*	*mean*	*var*	*mean*	*var*
hand position.	2.7921	2.6407	2.1096	2.3962	1.9800	2.4007
pointing accuracy	6.9372	14.0923	5.9311	8.0128	5.8982	8.9411

Tab. 1: Training and Test results for the positioning of the end-effector as well as the pointing direction (in centimeters).

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

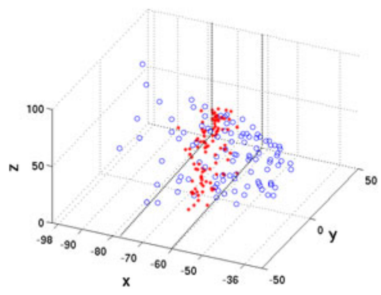

Fig. 2: SRC performance in centimeters in stereo matching task, shows that the depth value is averaged (on the x-axis) over the training data, which effectively implements the desired projection of training data along the pointing ray.

mixing of inputs that is performed by the recurrent dynamics. Overall, we conclude that a direct mapping is feasible and hypothesis that the networks achieve that by avoiding the intermediate stereo computation to reconstruct the actual 3D coordinates of the ball.

The following investigation underlines this claim by using the best performing architecture from the previous section and training it for three different tasks: for output of joints angles as before, for direct output of the hand position and for reconstruction of the 3D ball position, which we had recorded as an intermediate result from the iCub during training. Additionally, a second training adds the ball coordinates directly as inputs ($\mathbf{u}(k) = (x_b(k), y_b(k), z_b(k))^T$). In Tab. 2, the rows represent the input and columns the output.

If the hypothesis holds that the networks do not internally perform depth computation, then we expect two results: First, the additional input should significantly improve the performance as it adds information which is certainly useful, but not yet computed. Second, the network should not be able to reconstruct the 3D position, even if explicitly trained for it. Both effects are clearly visible in Tab. 2 and Fig. 2, confirming the the holistic mapping implements the pointing without depth calculation.

Training	**Joints**		**Hand Pos.**		**Ball Coord.**	
	mean	*var*	*mean*	*var*	*mean*	*var*
Pixels	1.5409	1.4692	1.592	1.3945	11.1818	50.4705
Coordinates	0.73254	0.28297	0.55604	0.12806	-	-
Testing	*mean*	*var*	*mean*	*var*	*mean*	*var*
Pixels.	1.98	2.4007	2.0171	2.1365	14.0733	68.1853
Coordinates	0.89349	0.46479	0.73071	0.21302	-	-

Tab. 2: Training and Test results of different mappings.

4 Conclusion

We have shown that a direct learning of pointing without depth calculation is possible if respective visuo-motor training data are available. As shown on Tab. 2 and Fig. 2 a stereo matching with this direct approach is not successful, however, as pointing in various forms is pervasive in human and robot interaction, it is a very relevant result. Further work beyond this more proof-of-concept now shall address to collect more and possibly higher quality training data from more sophisticated experimental setups to learn a more precise pointing. This could be done for instance by used compliance and kinesthetic teaching or automatised procedures to present the object in the visual field. Once a good mapping is learned, this can become a very important component for cognitive robots connecting for instance attention, which selects where to look (or orient the head) next, and cognitive layers, which may iteratively point, track and approach an object.

Acknowledgement
This work was supported by CNPq, an entity of the Brazilian government dedicated to the scientific and technological development, by DAAD, the German Academic Exchange Service and by the European Communitys FP7 under grant agreement No 248311 - AMARSi.

References

[1] M. Marjanović, B. Scassellati, and M. Williamson. Self-taught visually-guided pointing for a humanoid robot. In *From Animals to Animats: Proceedings of 1996 Society of Adaptive Behavior*, pages 35–44. MIT Press, 1996.

[2] A. Shademan, A.-M. Farahmand, and M. Jagersand. Towards learning robotic reaching and pointing: An uncalibrated visual servoing approach. In *Computer and Robot Vision, 2009. CRV '09. Canadian Conference on*, pages 229–236, May 2009.

[3] J. Feng. *Computational Neuroscience: A Comprehensive Approach.* Chapman & HallCRC, 2004.

[4] D. Ramachandram and M. Rajeswari. A short review of neural network techniques in visual servoing of robotic manipulators. In *Malaysia - Japan Seminar On Artificial Intelligence Applications In Industry*, pages 24–25, Malaysia, June 2003.

[5] G. Sun and B. Scassellati. A fast and efficient model for learning to reach. *International Journal of Humanoid Robotics*, 2(4):391–413, 2005.

[6] H. Ritter, T. Martinetz, and K. Schulten. Topology conserving maps for learning visuomotor-coordination. *Neural Networks*, 2:159–168, 1989.

[7] M. Jones and D. Vernon. Using neural networks to learn hand-eye co-ordination. *Neural Computing & Applications*, 2(1):2–12, 1994.

[8] G. Huang, Q. Zhu, and C. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 170(1-3):489–501, December 2006.

[9] J. Triesch. A gradient rule for the plasticity of a neurons intrinsic excitability. In *Int. Conf. on Artificial Neural Networks*, page 6579, 2005.

[10] K. Neumann and J. Steil. Batch intrinsic plasticity for extreme learning machines. In *Artificial Neural Networks and Machine Learning - ICANN*, pages 339–346. 2011.

[11] C. Emmerich, F. Reinhart, and J. Steil. Recurrence enhances the spatial encoding of static inputs in reservoir networks. In *Proc. ICANN*, pages 148–153, 2010.