AGI Architecture Measures Human Parameters and Optimizes Human Performance

András Lőrincz and Dániel Takács

Eötvös Loránd University, Budapest H-1117, Hungary,
andras.lorincz@elte.hu dtakacs@gmail.com,
WWW home page: http://nipg.inf.elte.hu

Abstract. AGI could manifest itself in human-computer interactions. However, the computer should know what is on the mind of the user, since reinforcement learning, the main building block of AGI is severely troubled for partially observed states. Technological advances offer tools to uncover some of these hidden components of the '*state*'. Here, for the first time, we apply an AGI architecture for the optimization of human performance. In particular, we measure facial parameters and optimize users' writing speed working with head motion controlled writing tool. We elaborate on how to extend this optimization scheme to mind reading and HCI.

Keywords: AGI architecture, computer-human interface, reinforcement learning

1 Introduction

AGI developments face the problem of how to measure and compare achievements. The Turing test [7] and Turing games [10, 17] could be good candidates. However, humans are 'equipped' with excellent evolution-tailored sensors to read the mind of the partner, develop models (make theory) about the others' mind. Such sensors and such theory are also needed for AGI, especially if AGI aims human-computer interaction and collaboration, since reinforcement learning, the main building block of AGI, is severely troubled for partially observed states and so AGI is 'handicapped' without 'knowing' what is on the mind of the user. Here, for the first time, we use an AGI architecture for learning human parameters and for the optimization of human performance. AGI components comprise of (i) a sensory information processing system, (ii) a system that estimates the user's autoregressive exogeneous (ARX) process in the context of the actual goal, (iii) the inverted form of the ARX process for influencing the situation, as well as (iv) event learning [23] and an optimistic initial model [21] to optimize long-term human performance. In the experiments we measure parameters of the face in real time explore the relevant part of the parameter space and exploit that knowledge in head controlled writing using writing tool Dasher [24].

The paper is built as follows. We review our architecture in Section 2. Then we detail the experimental setup (Section 3). Results are presented in Section 4. In the Discussion (Section 5), we elaborate on the missing / neglected components of the architecture and also on to what extent mind reading occurred and how to extend it further in interactions. Conclusions are drawn in Section 6.

2 Architecture

At a very high level, the architecture is made of the following components: sensory processing unit, control unit, inverse dynamics unit, and decision making unit. Although it looks simple, one has to worry about a number of things, such as the continuity of space and time, the curse of dimensionality, if and how space and time should be discretized, and planning in case of uncertainties, e.g., in partially observed situations, including information about purposes, cognitive, and emotional capabilities of the user. A detailed description of the proposed architecture has been provided in [9] and some architectural components are under development. A simplified version of the architecture has been used for illustration; the algorithm learned to optimize the motion of a pendulum from raw visual information of 40,000 dimensions [11]. Below, we review the components of the architecture together with the present state of our developments:

Sample selection: This stage selects samples under random control. Selection tries to 'keep samples apart'.
- For high dimensional inputs, sparse (combinatorial) representations are being developed to overcome the curse of dimensionality [6].

Low-dimensional embedding: In the illustration, the dimension of the low dimensional manifold was known. Selected samples were embedded into the low dimensional space and were used for interpolation.
- Online group structured dictionary learning methods [19] are under development for similar purposes in high dimensions and for large input sets.

Identification of the ARX process: Out-of-sample estimations can be used for the identification of the ARX process [11]. Bayesian interrogation can speed-up the learning process [15].

Control: The inverted ARX process can be used for control both in the under-controlled and over-controlled cases.

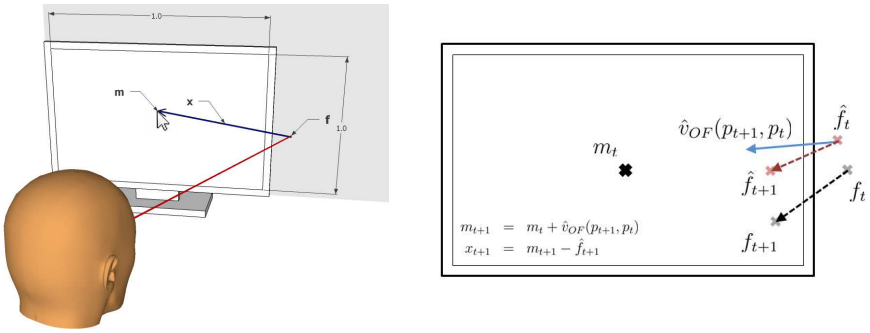
LQR solution to the inverse dynamics: A demonstration was designed for the under-controlled situation. We used a linear-quadratic regulator (LQR) [11].

Event learning: RL has been rewritten into a novel event learning formalism [23] in order to connect RL and continuous control. Event learning has been extended to switching combinatorial modules on and off [20].

Exploring space and learning states: Optimistic initial model (OIM) was used for exploring the space and for learning the values of *events*.
- OIM suits large RL problems since extended to the factored case [22].
- RL states has to be learned. We have argued that states and state-state transitions correspond to close-to-deterministic processes and switches between these processes, respectively [5]. Supplementary material on RL agent searching for Markov models in near-deterministic world can be found here [12].

Here, we use sophisticated preprocessing of sensory information (i.e., the head pose) before invoking components of the AGI architecture. Preprocessing should be rewritten to match the architecture. Such rewriting can make use of structured representations and dictionary learning and should be possible (see, e.g. [13]), but at this stage of development we included domain knowledge. Below, we detail the architectural components used in the experiments.

Typical pose estimations use PCA methods for shape, texture, and details, see, e.g., [2, 16] and references therein. We needed larger pose angle tolerance than offered by the presently available open source solutions and used a commercial software[1] for this purpose. We used our a Viola-Jones face detector and flow field estimation for detecting the face and relative changes of the head pose, respectively[2]. Input to the learning algorithm was hand made: Let us denote the screen size normalized position of the cursor by $(m_x, m_y) \in [0, 1]^2$ and the head pose by $(f_x, f_y) \in \mathbb{R}^2$. The two-dimensional vector $x = [m_x - f_x, m_y - f_y]^T \in \mathbb{R}^2$ was taken as the state (Fig. 1).



(a) (b)

Fig. 1. Illustration and parameters of the experiments. (a): Experimental arrangement. m: cursor position, f: position where the roll axis of head pose crosses the screen. x: 'observation'. (b): True and estimated quantities at time t. $v_{OF}(p_{t+1}, p_t)$: optic flow based estimation of the motion vector f. p_t: positions of feature points on the 2D projected face at time t (see Fig. 2(a)).

ARX estimation. The AR model assumes the following form

$$x_{t+1} = m_{t+1} - f_{t+1} \tag{1}$$

$$m_{t+1} = m_t + v_t \tag{2}$$

where $m_t \in \mathbb{R}^2$ is the position of the cursor at time t, $f_t \in \mathbb{R}^2$ is the point where the roll axis of the pose hits the screen as shown in Fig. 1(b), $v_t \in \mathbb{R}^2$ is the speed vector of the projected f_t on the screen over unit time and no additional noise was explicitly assumed. We have direct access to the cursor position and need to estimate the other parameters. Since the formulation above means that $v_t = f_{t+1} - f_t$ it follows that $x_{t+1} = x_t$ in the absence of estimation errors and control. The goal is to control and optimize x for writing speed.

We do not have direct access to f_t or x_t, but use an estimation $\hat{f}_t \in \mathbb{R}^2$ (Fig. 1(b)). This estimation was noisy so we estimated v_t independently using optic flow and subsequent image patches covering the face $(\hat{v}_{OF}(p_{t+1}, p_t))$, see Fig. 2(a). Here, $p_t \in \mathbb{R}^{2k}$

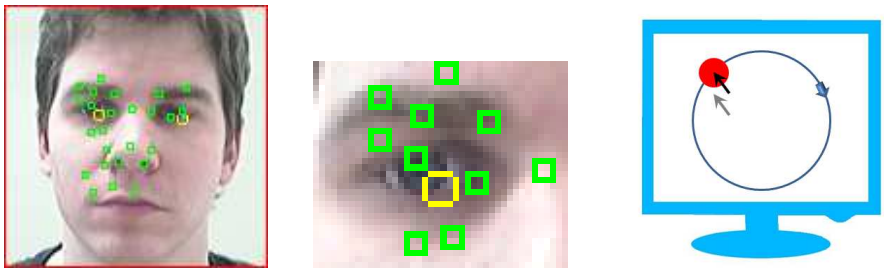
[1] FaceAPI `http://www.seeingmachines.com/`

[2] `http://chacal.web.elte.hu/.MouSense/MouSenseSetup-1.1.exe`

denotes the 2D coordinates of k characteristic points within the facial region of the image. In turn, Eqs. (1)-(2) can be written into the following ARX form

$$\hat{x}_{t+1} = \hat{x}_t + [\hat{v}_{OF}(p_{t+1}, p_t) - (f_{t+1} - f_t)] + Bu_t + n_t \tag{3}$$

where n_t denotes zero mean and Σ variance normal distribution ($\mathcal{N}(0, \Sigma)$) representing the noise of the ARX process. Since $\hat{v}_{OF}(p_{t+1}, p_t) \propto f_{t+1} - f_t$ and the more precise the optic flow estimation is, the smaller the contribution of the 2^{nd} term of the r.h.s, we neglected this term.



(a) Face with eye (yellow) and optic flow (green) markers

(b) Close-up

(c) ARX identification method

Fig. 2. Tools for human-computer interaction: (a)-(b): features for optic flow estimation (green markers), eye tracker and head pose estimation (yellow marker), (c): computer 'game' measures the ARX parameters of head motion.

Collecting a number of data $\hat{x}_1, \ldots, \hat{x}_T$, we can estimate the unknown parameters, i.e., the elements of matrix B in the ARX process. We tried two methods; direct control and gain control. In the case of direct control, we changed the position of the mouse directly; control values corresponded to distances on the screen

$$\hat{x}_{t+1} = \hat{x}_t + Bu_t + n_t. \tag{4}$$

In another case, we controlled the gain of the mouse as follows: let g_{+x_1} and g_{-x_1} denote the sensitivity of the mouse in positive and negative horizontal directions, respectively, and similarly for the vertical axis (x_2). Then for control vector $u = (u_{x_1}, u_{x_2}) \in [-2, 2]^2$ we modified the gain as follows

$$g_{+x_i} + g_{-x_i} = 2 \tag{5}$$

$$g_{+x_i} - g_{-x_i} = u_{x_i} \tag{6}$$

($i = 1, 2$). The multiplicative nature of this second design, however, gives rise to no effect if the position of the mouse is not changed by the user, which happens to be typical for the Dasher application. Results for this control mode are poor for Dasher and are not presented here.

Inverse dynamics. One can invert equation (4) to ask: what is the control value that could yield desired state x_d in the next time instant? This inverted equations gives rise to the following answer

$$u_t = \hat{B}^{-1}(\hat{x}_d - \hat{x}_t). \tag{7}$$

Inserting this control value into Eq. (3) we get $\hat{x}_{t+1} \approx \hat{x}_d$, where equality is spoiled by the approximations we made and the additional noise contributions. We note that Eq. (7) suffices our present purposes, but for more demanding problems it is too simple. There are many ways to upgrade the control scheme to sophisticated non-linear 'plants', e.g., by applying feedback linearization [4].

Event learning We define the control problem within the *event learning* framework that provides the actual and desired states to a backing controller and the controller tries to satisfy the 'desires'. For a given experienced state and desired state pair the controller provides a control value or a control series. Then, event learning learns the limitations of the backing controller and optimizes the RL policy in the event space accordingly [23].

We discretize the space and have N distinct states. The estimated value of an event E^{π}_{i,i^+} in event learning is the estimated long-term cumulated discounted reward of event $e = (i, i^+)$ in a Markov decision process under fixed policy $\pi = \pi(i, i^+)$, with actual state i and desired state i^+, $(i, i^+ = 1, 2, \ldots, N)$. The '+' sign indicates that the second argument of event e is the successor of the state represented by the first argument.

Optimistic Initial Model (OIM). OIM is an extension of the optimistic initial value method (OIV) [3]; it resolves the exploration exploitation dilemma by building a model. OIM, alike to OIV, gives exploration boost to RL by assuming that every discovered and not yet fully explored state–action pair – in our case state–desired state pair – may lead to the a highly promising Garden of Eden state. Dynamic programming updates can not, only experiences can lower the exploration boost in OIM giving rise to considerable advantages. OIM policy [21] is not fixed, but it converges to the optimal policy.

3 Experiments

Beyond the optic flow based head motion detector, we used the FaceAPI SDK for head pose estimation. A calibration procedure was used at the beginning of the experiments. The principle is shown in Fig. 2(c): a red dot was moving on a circular path on the screen. The user could move the cursor by moving its head. The task was to keep the cursor within the red dot. If the cursor was within the dot then it speeded up, otherwise it slowed down. During the experiments, random control values were added to the motion in order to estimate matrix B. As expected, a close to diagonal matrix was learned with relatively small off-diagonal elements, being about one fifth of the diagonal values. Diagonal elements corresponded to the scaling (normalization) in the horizontal and vertical directions. This calibration was sufficient to learn the ARX process and to estimate the inverse dynamics.

The Optimistic Initial Model was configured as follows.

State space: Discretized differences between 2d cursor position and 2d crossing point of roll axis of head pose on the screen.

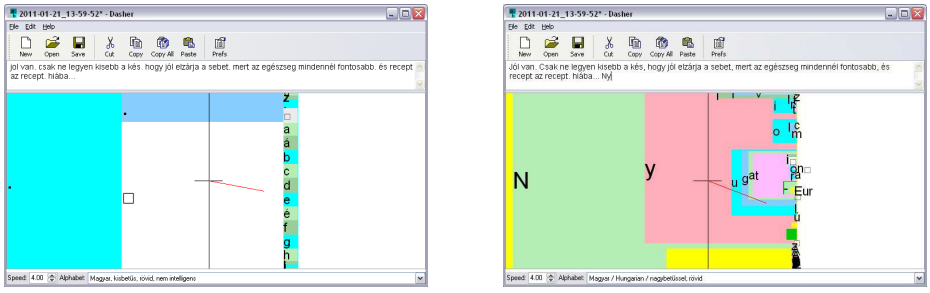
- vertical direction: five regions with centers at (-0,8, -0,4, 0, 0,4, 0,8)
- horizontal direction: three regions with centers at (-0,4, 0, 0,4)

Duration of time steps: 5 s.

Reward: Number of typed minus number of deleted letters during time steps.

Actions: Either the actual state itself, or one of the neighbor states (maximum number is 4) was chosen as the desired state. Inverse ARX was applied to move to the desired state (i.e., to modify the angle between the direction of the cursor from the head and the direction of the roll axis of the head pose).

Performance optimization was conducted with the Dasher writing tool (Fig. 5) [24]. We used two versions: the version with uniform probabilities for each letter (Fig. 3(a))



(a) Dasher without database (b) 'Intelligent' Dasher

Fig. 3.

and the 'intelligent' version that utilizes the method of prediction by partial matching and scales letter areas according to their estimated probabilities (Fig. 3(b)).

Two subjects conducted the experiments for both cases independently. Experiments lasted for five sessions. Each session had five writing periods and four breaks between, with and about 600 characters to be typed in each period. Periods took about 20 minutes.

4 Results

Convergence of optimization is shown in Fig. 4. 5(a)-5(f). Details of the values of different events are shown in Fig. 5. Figures represent the event values (without the exploration boost); the expected long term cumulated value of choosing a next desired state (which could be the actual state) and then continuing decision making according to the OIM policy. Intelligent Dasher is about 2 times faster. Strategies of the two subjects were different. For intelligent Dasher and for subject 1, optimal performance was reached with cursor position to the left and to the top relative to the crossing point of the roll axis of the head pose on the screen (subfigure (b), second column, top subplot).

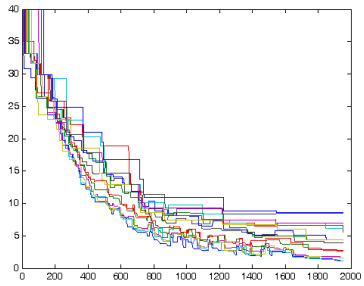


Fig. 4. Convergence of OIM. States are discovered before step 200. Convergence as achieved for most states by step 2000, but exploration continues for the rarely achievable states.

Events that changed the vertical difference were sometimes advantageous. For subject 2, and for the intelligent case, large values were reached for diverse angle differences, so the computer was not supposed to do anything. Situation was somewhat different for the uniform distribution, behavior resembled to that of subject 1 in the intelligent case.

Optimal difference between cursor position and the crossing point of the roll axis of the head pose on the screen was not zero: it was better to look to the right and observe the coming letters sooner, but to keep the cursor closer at a safer region, i.e., closer to the origin of the Dasher screen, where squares of the letters are larger.

5 Discussion

Results show that personalization is advantageous for head pose driven writing with Dasher. How far does this experiment go in terms of mind reading and how to extend it further? What do we have and what do we need?

The computer has a head pose estimation and a performance gauge in the context of Dasher. Furthermore, the method of prediction by partial matching (PPM) is also providing estimations about what are the probabilities of next letters, the next words, etc., so – to some extent – some details about what is on the mind of the user. Since individuals use different words, further personalization is possible by adapting PPM's dictionary. Additional information can be gained by categorizing the context of writing, e.g., if it is an essay, or a letter to a friend, and so on. Another source of information is performance, the computer may infer that the user is sleepy/tired, or if acoustic information is available, then if the user is distracted from writing. These factors may influence the mood of the user. Facial expressions can be recognized by the computer, but we rarely use facial expressions when working with the computer. On the other hand, if the computer can use facial expression for expressing uncertainty, doubts, or being pleased about our performance, then humans will quickly react and provide invaluable information and feedback to the computer for collaboration: games show that people like to express emotions even if they know that the partner is an artificial agent [10].

Can we generalize from the slightly different optimal settings for the two subjects of this experiment? How different and how regular are we? Recent studies show that

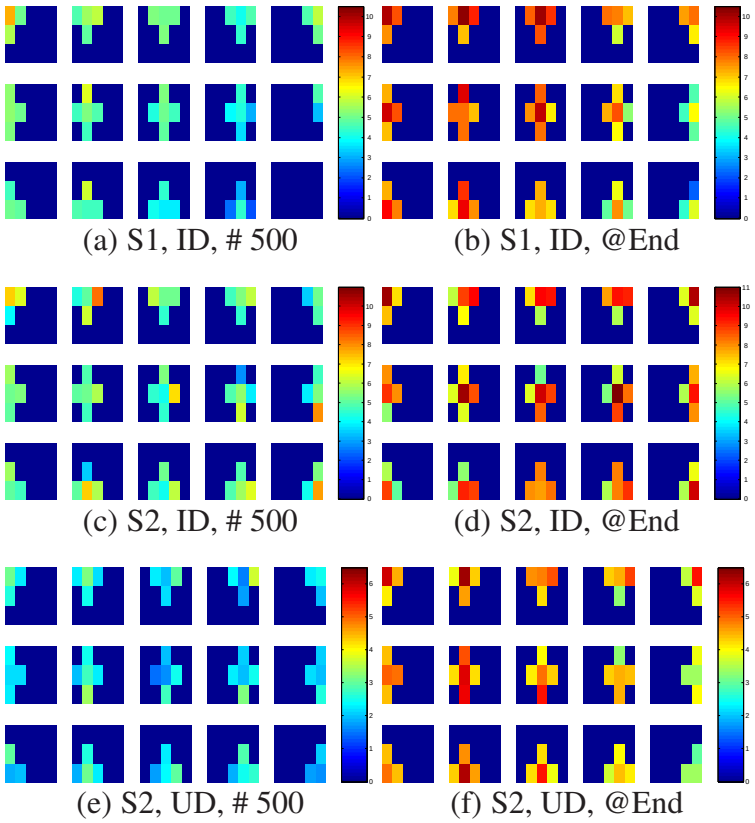


Fig. 5. Event values of $e(i, i^+)$ state – desired state pairs *without* the 'Garden of Eden' exploration boost. Coding: 3×5 subplots, each representing 3×5 state-state pairs (although only the state itself or neighbors were allowed as desired states in the experiments). Notations: ID/UD: intelligent/uniform Dasher. S1/S2: Subject 1st/2nd, 500/End: After 500 time steps / at the end. Left column: after 500 time steps. Right column: at the end of optimization. (c)-(d): more optimization is needed. Example: upper left subfigure of (a): optimal policy: stay. Worst policy: decrease horizontal angle difference between direction of cursor and estimated direction of head pose (note the sign of the terms in x). Middle: decrease vertical angle. Typical optimal behavior: look to the right of the cursor. For large vertical angle differences; keep it. For smaller vertical angle differences: changes to the vertical angle differences have about the same value as keeping it. Intelligent Dasher is about a factor of 2 faster.

our everyday customs are surprisingly regular [14, 18]. Although we clearly like free decision making, that is the option that we can make our own choices, we also unlike to be inefficient in our daily lives and we optimize our routines. Optimal policies are, however, (typically) deterministic, and thus the computer can help us to uncover unobserved but relevant information, but these can be very different from person to person, since the optimal policies are different.

From the point of view of AGI, interaction and collaboration is a sensitive gauge about AGI advances. However, for efficient interactions, the computer should be able to measure cognitive capabilities and emotional drives of the user, should 'translate' information to the user, help in understanding, and to develop common customs. Technology is on its way to solve these issues in the coming years, provided that a scalable and multi-modal AGI architecture is available. In the present work, we used natural language processing (the PPM of Dasher) and machine vision that monitored the face of the user. Any additional information, e.g., noises in the environment, emotion detection in speech, among other things can help the computer in interaction and collaboration. Games, or game-like activities can be used for cognitive and emotional profiling as well as for the prediction of performance in different tasks. Note that both tools we used, i.e., the calibration tool that estimated the ARX parameters and the writing tool have a game-like characters, they are similar to balancing games and FPS games, respectively. Both of them are challenging and both develop certain skills that could be useful for normal subjects and for people with special needs [8].

6 Conclusions

We have used an AGI architecture [11] for the optimization of human performance during head pose driven writing. We combined visual and textual information, the computer predicted the coming letters/words using the method of prediction by partial matching. It also identified the dynamics of head motion, used the inverse dynamics to control the direction of head and optimized performance by means of novel reinforcement methods, such as event learning [23] and the optimistic initial model [21]. We have argued that the architecture is scalable in most aspects, including sensory information processing, robust control, and reinforcement learning. We referred to recent findings in the literature that it might be easy to learn and predict our everyday routines [14, 18], so the computer could help in our activities by providing information in due course. Beyond the customs of the user, the computer should also be able to estimate the user's actual cognitive and emotional capabilities and to modulate the interaction accordingly in order to optimize the content for the user. Recent algorithmic developments (see, e.g., [1] and references therein) and technological advances promise fast progress in this respect.

The relevance of the present study is that we used tools to measure human parameters, coupled those to our AGI architecture and used the architecture to improve human performance.

Acknowledgments. Research was supported by the European Union and co-financed by the European Social Fund (grant agreement no. TÁMOP 4.2.1./B-09/1/KMR-2010-0003).

References

1. Chen, Y., Xu, H., Caramanis, C., Sanghavi, S.: Robust matrix completion with corrupted columns. http://arxiv.org/abs/1102.2254 (2011)

2. Cristinacce, D., Cootes, T.: Automatic feature localisationwith constrained localmodel. Pattern Rec. 41, 3054–3067 (2008)
3. Even-Dar, E., Mansour, Y.: Convergence of optimistic and incremental Q-learning. In: Advances in Neural Information Processing Systems 14. pp. 1499–1506 (2001)
4. Khalil, H.K.: Nonlinear Systems. Prentice Hall, NJ (2002)
5. Lőrincz, A.: Learning the states of Markov decision processes: A brain inspired neural model (2011), submitted
6. Lőrincz, A., Palotai, Z., Szirtes, G.: Sparse and silent coding in neural circuits. http://arxiv.org/abs/1010.4138 (2010), submitted
7. Loebner, H.: Parsing the Turing Test, chap. How to Hold a Turing Test Contest, pp. 173–179. Springer Netherlands (2009)
8. Lőrincz, A.: Machine situation assessment and assistance: Prototype for severely handicapped children. In: Proc. Reg. Conf. Embedded and Ambient Syst., Selected Papers. pp. 61–68. John von Neumann Computer Society (2008)
9. Lőrincz, A.: Learning and representation: From compressive sampling to the 'symbol learning problem'. In: Handbook of Large-Scale Random Networks, pp. 445–488. Springer, Heidelberg (2009)
10. Lőrincz, A.: Turing game approach to measure and advance machine intelligence. European Future Technologies Conference, Prague (2009), exhibition
11. Lőrincz, A., Bárdosi, Z., Takács, D.: Sketch of an AGI architecture with illustration. In: 3rd Conf. on Artif. Gen. Intel. (2010), http://dx.doi.org/10.2991/agi.2010.40
12. Matuz, G., Lőrincz, A.: Decision making agent searching for Markov models in near-deterministic world. http://arxiv.org/abs/1102.5561 (2011)
13. Murphy-Chutorian, E., Trivedi, M.M.: Head pose estimation in computer vision: A survey. IEEE Tr. Pattern Anal. Mach. Intell. 31, 607–626 (2009)
14. Pentland, A.: On the collective nature of human intelligence. Adaptive Behavior 15, 189–198 (2007)
15. Póczos, B., Lőrincz, A.: Identification of recurrent neural networks by Bayesian interrogation techniques. J. of Mach. Learn. Res. 10, 515–554 (2009)
16. Saragih, J., Lucey, S., Cohn, J.: Deformable model fitting by regularized landmark mean-shifts. Int. J. Comp. Vision (in press)
17. Schaul, T., Togelius, J., Schmidhuber, J.: Measuring intelligence through game. J. Artif. Gen. Intell. 2, 1–19 (2010)
18. Song, C., Qu, Z., Blumm, N., Barabási, A.: Limits of predictability in human mobility. Science 327, 1018–1021 (2010)
19. Szabó, Z., Póczos, B., Lőrincz, A.: Online group-structured dictionary learning. In: Comp. Vision and Pattern Rec. (2011), accepted
20. Szita, I., Lőrincz, A.: Learning to play using low-complexity rule-based policies: Illustrations through Ms. Pac-Man. Journal of Artificial Intelligence Research 30, 659–684 (2007)
21. Szita, I., Lőrincz, A.: The many faces of optimism: a unifying approach. In: 25th Int. Conf. on Mach. Learn. pp. 1048–1055. Omnipress, Helsinki (2008)
22. Szita, I., Lőrincz, A.: Optimistic initialization and greediness lead to polynomial time learning in factored MDPs. In: 26th Int. Conf. on Mach. Learn. p. 126. Omnipress, Montreal (2009)
23. Szita, I., Takács, B., Lőrincz, A.: Epsilon-MDPs: Learning in varying environments. J. Mach. Learn. Res. 3, 145–174 (2003)
24. Ward, D.J., MacKay, D.J.C.: Fast hands-free writing by gaze direction. Nature 418, 838 (2002)