Control Policy with Autocorrelated Noise in Reinforcement Learning for Robotics

Paweł Wawrzyński

***Abstract*—Direct application of reinforcement learning in robotics rises the issue of discontinuity of control signal. Consecutive actions are selected independently on random, which often makes them excessively far from one another. Such control is hardly ever appropriate in robots, it may even lead to their destruction. This paper considers a control policy in which consecutive actions are modified by autocorrelated noise. That policy generally solves the aforementioned problems and it is readily applicable in robots. In the experimental study it is applied to three robotic learning control tasks: Cart-Pole SwingUp, Half-Cheetah, and a walking humanoid.**

***Index Terms*—Machine learning, reinforcement learning, actorcritics, robotics.**

I. Introduction

Reinforcement learning (RL) addresses the problem of an agent that optimizes its reactive policy in a poorly structured and initially unknown environment [1]. The primary application of RL is robotics where the agent becomes a robot's controller and the robot itself with its surrounding becomes the agent's environment. Reinforcement learning offers the prospect of efficient robot behaviour being learned rather than programmed by a human designer.

A typical setting in which RL is applied in robotics is as follows. There are two levels of control. The lower level is based on servomotors in the robot's joints. The servomotors are fed with desired joint positions and try to make the joints follow them. At the higher level, the controller determines desired servomotors' positions based on the robot state. A learning (through reinforcement) component resides at the higher control level. Within typical scheme of the learning component operation, the desired servomotors' positions are periodically selected on random, and consecutive selections are only stochastically dependent through the robot state. But that means that the consecutive desired servos' positions are far from one another. This results in a characteristic jerking of the robot which is an unhealthy robot behaviour and may lead to its destruction.

Applications of RL in robotics are surveyed in [2]. More general discussion on policy search in robotics is presented in [3]. The work [4] presents an RL algorithm that optimizes robotic primitives. This algorithm overcomes the problem of robot jerking during learning at the cost of giving up the framework of Markov Decision Process (MDP) [1]. In [5] a method is presented that enables optimization of robotic primitives by means of RL algorithms based on MDP framework, but it does not alleviate the problem of robot jerking. The current paper is intended to fill this gap.

In this paper, control policy is introduced that may undergo reinforcement learning and have the following properties:

- It is applicable to robot control optimization.
- It does not lead to robot jerking.
- It can be optimized by any RL algorithm that is designed to optimize a classical stochastic control policy.

The policy introduced here is based on a deterministic transformation of state combined with a random element in the form of a specific stochastic process, namely the moving average.

The paper is organized as follows. In Section II the problem of our interest is defined. Section III presents the main contribution of this paper i.e., a stochastic control policy that prevents robot jerking while learning. In Section IV an analysis of this policy is presented. Section V contains an experimental study in which the policy is applied in two simulated and one real robotic learning control tasks. The last section concludes the paper.

II. Problem Formulation

We consider the standard RL setup [1]. A Markov Decision Process (MDP) defines a problem of an agent that observes its state, s_t, in discrete time $t = 1, 2, 3,...$, performs actions, a_t, receives rewards, r_t, and moves to other states, s_{t+1}. A particular MDP is a tuple $\langle S, A, P_s, r \rangle$ where S and A are the state and action spaces, respectively; $\{P_s(\cdot|s,a) : s \in S, a \in A\}$ is a set of state transition distributions; we write $s_{t+1} \sim P_s(\cdot|s_t,a_t)$ and assume that each P_s is a density. Each state transition generates a reward, $r_t \in R$. Here we assume that each reward is depends deterministically on the current action and the next state, $r_t = r(a_t,s_{t+1})$. The agent learns to assign actions to states so as to may expect in each state highest rewards in the future.

Here we consider robotic applications of the above general framework. Therefore, both spaces of interest are multidimensional continuous: $S = R^{n_s}$ and $A = R^{n_a}$. Also, it is assumed that s_t reflects the state of a certain continuous-time system at discrete time instants. Let $\tau \in R$ denote continuous time. The dynamics of that system could be described by an equation of the form

$$\frac{ds(\tau)}{d\tau} = f\left(s(\tau), \alpha(\tau)\right), \tag{1}$$

where f is unknown, $\delta > 0$ denotes time discretization, $s_t =$

Manuscript received August 5, 2014; revised December 1, 2014.

Paweł Wawrzyński is with Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland (e-mail: p.wawrzynski@elka.pw.edu.pl).

DOI: 10.7763/IJMLC.2015.V5.489

$s(\tau_0 + t\delta)$, and $a(\tau) = a_t$ for $\tau \in [\tau_0 + t\delta, \tau_0 + (t+1)\delta)$.

The subject of our interest is a stochastic control policy that produces actions. The following properties of this policy are required:

1) It is possible to optimize this policy by any RL algorithm designed for stochastic control policy optimization.
2) Fine time discretization does not prevent the learning algorithm from efficient operation nor it results in robot jerking. However, fine time discretization may require adjustment of some parameters of the learning algorithm and the policy.

III. Policy Definition

A. Generic Definition

Let the actions by produced by the following function

$$a_t = h(s_t, \xi_t, \theta) \tag{2}$$

where s_t is the state, $\xi_t \in \mathrm{R}^{n_\xi}$ is a random element, and $\theta \in \mathrm{R}^n$ is a parameter (e.g., neural weights). Typically a policy that produces actions is defined as a probability distribution parameterized by the state and a vector, θ, whose optimization is an objective of learning. But technically, actions are always computed on the basis of a certain function, h, and a finite-dimensional random element, ξ_t. With the additional assumptions that ξ_t have the same distribution for various t, and ξ_t is stochastically independent from ξ_{t+i} for $i \neq 0$, eq. (2) is a typical representation of a stochastic control policy in reinforcement learning.

In this paper the following set or requirements is imposed on (2):

1) ξ_t has the *same* (stationary) distribution for each t.
2) ξ_t is stochastically independent from ξ_{t+i} for $|i| \geq M$, where $M > 0$ is a certain constant.
3) $E\|\xi_t - \xi_{t-i}\|^2 < E\|\xi_t - \xi_{t-i-1}\|^2$ for $0 \leq i < M$.
4) h is continuous with respect to all its arguments.

The first two conditions are required for the policy to be applicable in known reinforcement learning algorithms, e.g., in actor-critics [6], [7]. The latter two make consecutive control actions close to one another. Strict "continuity" of the control signal is not possible in continuous time. However, if h is continuous and ξ_t is on average close to ξ_{t+1}, consecutive actions are close to each other as well.

B. Specific Definition

A specific design of a policy based on the above requirements may be the following. Let

$$a_t = h(s_t, \xi_t, \theta) = g(s_t; \theta) + \xi_t, \tag{3}$$

where g is a certain approximator parametrized by θ with input s_t, and ξ_t is defined as follows. Let

$$\xi_t \sim N(0, I\sigma^2/M) \tag{4}$$

be random vectors stochastically independent for different t, $M > 0$ be constant, and

$$\xi_t = \sum_{j=0}^{M-1} \zeta_{t-j}. \tag{5}$$

A stochastic process defined in (5) is known as the moving average. Let us now verify the conditions from the previous section.

1) Each ξ_t is a sum of normal random vectors, therefore each ξ_t has the same (stationary) distribution $N(0, \sigma^2 I)$.
2) ξ_t and ξ_{t+i} are, for $|i| > M$, *computed* from different ζ-s, thus they are stochastically independent.
3) For $0 \leq i < M$ we have

$$E\|\xi_t - \xi_{t-i}\|^2 = E\left\|\sum_{j=0}^{M-1} \zeta_{t-j} - \zeta_{t-i-j}\right\|^2 \tag{6}$$

$$= E\left\|\sum_{j=0}^{i-1} \zeta_{t-j} - \sum_{j=0}^{i-1} \zeta_{t-M-j}\right\|^2 \tag{7}$$

$$= 2i\sigma^2/M, \tag{8}$$

and therefore

$$E\|\xi_t - \xi_{t-i}\|^2 < E\|\xi_t - \xi_{t-i-1}\|^2 = 2(i+1)\sigma^2/M. \tag{9}$$

4) If g is continuous with respect to its arguments, then h is continuous too.

The distribution of action defined according to (3) is normal with mean $g(s_t; \theta)$ and covariance matrix $I\sigma^2$.

IV. Analysis

In this section we investigate the following question: how to parametrize control policy to assure constant level of randomness in state trajectory for any given time discretization, δ. In this order, *power of noise* is defined below, it is derived for ξ_t defined in the previous section, and it is proven that the randomness in state trajectory is proportional to that power.

A. Power of ξ

Let

$$\xi(\tau) = \xi_t \text{ for } t : \tau_0 + t\delta \leq \tau < \tau_0 + (t+1)\delta. \tag{10}$$

We define the *power of* ξ as

$$P_\xi = \lim_{T\to\infty} \frac{1}{T} E\left(\int_{\tau_0}^{\tau_0+T} \xi(\tau)\mathrm{d}\tau \int_{\tau_0}^{\tau_0+T} \xi(\tau)^T \mathrm{d}\tau \right). \tag{11}$$

In the case of the moving-average (5), its power has the value

$$P_\xi = \lim_{t\to\infty} \frac{1}{t\delta} E\left(\sum_{i=0}^{t-1} \delta\xi_i \sum_{i=0}^{t-1} \delta\xi_i^T \right) \tag{12}$$

$$= \lim_{t\to\infty} \frac{1}{t\delta} E\left(\sum_{i=0}^{t-1} \delta M\zeta_i \sum_{i=0}^{t-1} \delta M\zeta_i^T \right) \tag{13}$$

$$= \frac{1}{t\delta} \delta^2 M^2 t I\sigma^2/M \tag{14}$$

$$= I\sigma^2 \delta M. \tag{15}$$

B. Constant Randomness in State Trajectory

A learning controller requires randomness in its actions. Its states trajectories need to be diversified, in order to find out which actions are good and which are inferior. But the level of

randomness should be selected reasonably. We stabilize randomness in state trajectory, such that for any given time discretization, δ, we are able adjust parameters of ξ_t (5), to keep the conditional variance

$$V(s(\tau+\Delta)|s(\tau)) \tag{16}$$

constant for constant Δ. From (1) we have

$$s(\tau+\Delta) = s(\tau) + \int_{\tau}^{\tau+\Delta} f(s(\tau'), a(\tau'))\mathrm{d}\tau'. \tag{17}$$

Let us denote

$$g_0 = g(s(\tau);\theta) \tag{18}$$

and consider small Δ such that for $\tau' \in [\tau, \tau+\Delta)$ we have

$$f(s(\tau'), a(\tau')) \cong f(s(\tau), a(\tau')) \tag{19}$$

$$g(s(\tau');\theta) \cong g_0. \tag{20}$$

Then we have

$$f(s(\tau'), a(\tau')) \cong f(s(\tau), g_0 + \xi(\tau')) \tag{21}$$

$$\cong f(s(\tau), g_0) + \frac{\partial f(s(\tau), g_0)}{\partial g_0}\xi(\tau'). \tag{22}$$

Therefore, from (17) we have

$$s(\tau+\Delta) \cong s(\tau) + \Delta f(s(\tau), g_0) + \frac{\partial f(s(\tau), g_0)}{\partial g_0}\int_{\tau}^{\tau+\Delta}\xi(\tau')\mathrm{d}\tau'. \tag{23}$$

For given τ and $s(\tau)$, the integral is the only random element above. Therefore

$$V(s(\tau+\Delta)|s(\tau)) \cong \frac{\partial f(s(\tau), g_0)}{\partial g_0}\int_{\tau}^{\tau+\Delta}\xi(\tau')\mathrm{d}\tau' \times \tag{24}$$

$$\times \int_{\tau}^{\tau+\Delta}\xi(\tau')^T\mathrm{d}\tau'\frac{\partial f(s(\tau), g_0)}{\partial g_0^T} \tag{25}$$

$$\cong \frac{\partial f(s(\tau), g_0)}{\partial g_0}\Delta P_\xi \frac{\partial f(s(\tau), g_0)}{\partial g_0^T} \tag{26}$$

$$\cong P_\xi \Delta \frac{\partial f(s(\tau), g_0)}{\partial g_0}\frac{\partial f(s(\tau), g_0)}{\partial g_0^T}. \tag{27}$$

It is seen that conditional variance of state is proportional to the power of ξ_t.

C. Conclusions and Illustration

The results above lead to the following conclusions.

1) Conditional variance of state is proportional to the power of ξ_t. Therefore, in order to keep this variance constant when manipulating time discretization, we need to adjust other coefficients in (15) accordingly.

2) The policy (3) with ξ_t (5) and $M = 1$ is a traditional control policy applied in RL: actions are only stochastically dependent through the state. The drawback of such a policy is clearly visible in (15). Variance of noise, σ^2, required to stabilize the power of ξ is inversely proportional to time discretization, δ. Therefore, if δ is very small, then σ^2 needs to be very large. But that makes such a policy not feasible as it requires large changes in control signal to be performed at large frequency.

Fig. 1 presents ξ_t (5) with different parameters but the same power. Top part of Fig. 1 presents noise typically applied in a control policy in reinforcement learning; it leads to "discontinuity" of control signal (its literal "continuity" is not possible in discrete time). The middle part presents autocorrelated ξ_t which has the same power but leads to more continuous control signal.

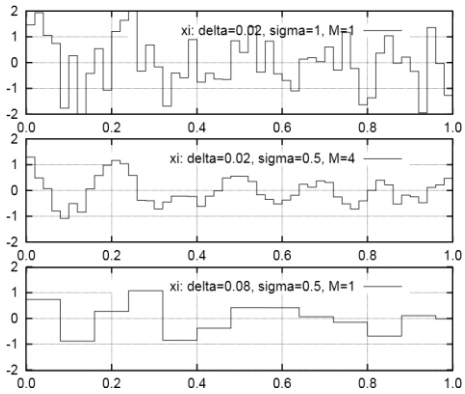

Fig. 1. The figures above demonstrates runs of ξ_t with various parameters, but the same power, P_ξ=0.02.

V. Experimental Study

In this study reinforcement learning is applied to optimize control policy in two simulated robotic learning control tasks and one real robotic learning control task. The simulated tasks are Cart-Pole Swing-Up [8] and Half-Cheetah [9], and the real one is humanoid walking [5]. The learning algorithms applied are actor-critics. The control policy applied is one introduced in Sec. III. The main question investigated concerns feasibility of the proposed policy: Is it possible to obtain similar (or better) learning performance with finer time discretization and smaller time-to-time differences in control signal?

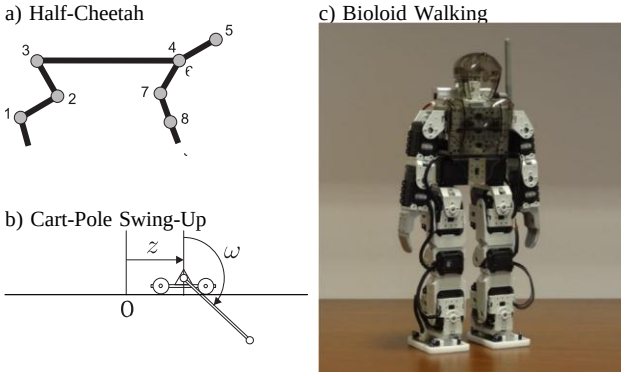

Fig. 2. Learning control problems.

A. Adjusting Parameters to Changing Time Discretization

If an actor-critic algorithm with the policy introduced here is successfully applied to a certain task, it can also be applied to this task with different time discretization. However, that requires adjustment of several parameters. Suppose time discretization δ is changed to $\delta' < \delta$. New parameters will be denoted below with primes:

- A weight of a reward received in a specific time in the

future should remain the same. But that reward will be received after more discrete time steps. The discount factor should be modified accordingly:

$$\gamma' = 1 - (1 - \gamma)\delta'/\delta. \quad (28)$$

By the same token, the λ parameter should be adjusted as

$$\lambda' = 1 - (1 - \lambda)\delta'/\delta. \quad (29)$$

- The sum of discounted rewards expected at each state should remain the same. Therefore all rewards except those received at the end of an episode (succeeded of failed) should be multiplied by the factor

$$\delta'/\delta. \quad (30)$$

- The actor and the critic updates are performed more often but they use the same stream of information. The step sizes for the actor, β^θ, and the critic, β^v should thus be smaller, they should be multiplied by the factor

$$\delta'/\delta. \quad (31)$$

- In order to keep the power of noise (15) at the same level, we need to apply

$$M' = M\delta/\delta'. \quad (32)$$

(Otherwise we could increase σ^2 accordingly, but that could destroy the robot.)

B. Half-Cheetah

Half-Cheetah [8] is a planar model of cat (Fig. 2a). The goal of its control is to make it run as fast as possible. With time discretization $\delta = 0.02$sec., 31 state variables, 6 control variables, and complex dynamics, Half-Cheetah represents a difficult system for efficient control.

We perform two experiments with Half-Cheetah. In both cases we use the same learning algorithm: Classic Actor-Critic with Experience Replay as in [8]. In the first case, we use the typical control policy based on a neural network and independently sampled normal noise. This policy rises the issue of control signal discontinuity. In the second case, the policy is applied discussed in Sec. III which overcomes the problem of control signal discontinuity.

The parameters of both experiments are listed as following:

param.	δ	σ	M	γ	λ	$\beta\theta$	βv
in [8]	0.02	5	1	0.99	0.9	10^{-5}	10^{-5}
here	0.02	3	3	0.99	0.9	10^{-5}	10^{-5}

The resulting learning curves are presented in Fig. 3.a. It is seen, that in the novel policy does not deteriorate learning efficiency nor ultimate performance. In fact, bot the learning efficiency and the ultimate performance are better when the new policy is applied.

C. Cart-Pole Swing-Up

Cart-Pole Swing-Up task [10] is a pendulum attached freely to a car that moves on a straight track (Fig. 2b). The objective of control is to swing the pendulum and stabilize it upwards, by pushing the car.

The learning control algorithm applied here is the classic actor-critic [6]. Details of its implementation are taken from [8]. An action is independently selected every $\delta = 0.1$sec. from the normal distribution with standard deviation equal to 2.

We also perform an experiment in which time discretization is $\delta = 0.01$sec., which is more typical for robotic applications. We apply the control policy from Sec. III with coefficients selected according to Section V-A. The parameters of both experiments are listed as following:

param.	δ	σ	M	γ	λ	$\beta\theta$	βv
in [8]	0.1	2	1	0.95	0.5	0.003	0.003
here	0.01	2	10	0.995	0.95	0.0003	0.0003

The resulting learning curves are presented in Fig. 3.b. It is seen, that in the original setting the learning is about twice faster. However, the ultimate performance obtained is the same.

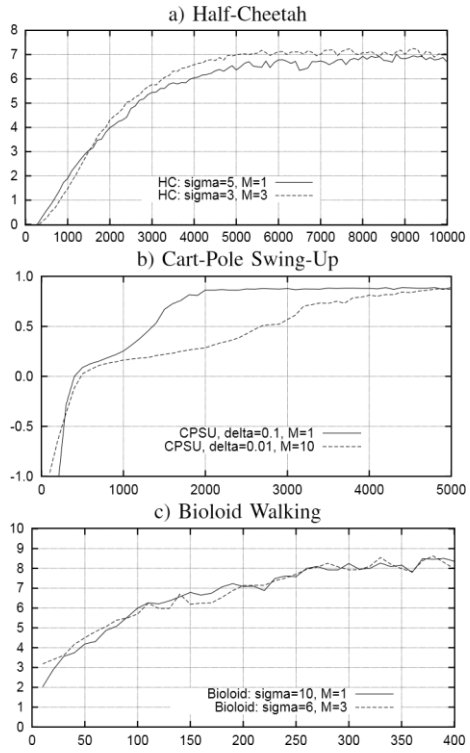

Fig. 3. Learning curves i.e., average rewards vs. episode number. Each curve averages 10 runs.

D. Bioloid Walking

Biolid[1] is a small humanoid robot (Fig. 2c). The control objective is to make this robot walk as fast as possible without falling and spinning about the vertical axis. A robot controller may learn to do so with the use of an actor-critic algorithm, as presented in [5]. Within the setting applied in [5] control signal discontinuity creates significant difficulties. The robot is jerking, its servomotors often get overloaded.

We perform an experiment in which we apply the policy introduced in Section III in Bioloid that is learning to walk fast. The parameters of both experiments are listed as the following.

param.	δ	σ	M	γ	λ
in [5]	0.03	10	1	0.99	0.9
here	0.03	7	3	0.99	0.9

[1]Bioloid Premium manufactured by Robotis: http://www.robotis.com

Step-sizes are absent from the table above because the algorithm applied calculates them autonomously.

The resulting learning curves are presented in Fig. 3c. It is seen, that with the new control policy the learning is equally efficient as in the original setting. Moreover, the course of experiment is much healthier for the robot. It is not jerking and its servomotors remain cool all the time.[2]

E. Discussion

The field of reinforcement learning offers the prospect of robots that learn instead of being programmed. Most algorithms developed in this field learn to make the best selection of the control action for each state the controlled system may enter. In order to learn, at each state the algorithm selects a control action on random thereby collecting experience that allows to tell which actions are good and which are not. However, these actions need to make difference, that is they need to assure appropriate level of randomness in state trajectory. The finer time discretization, the less significant each particular action is, and the more noise it should contain. That leads to excessively discontinuous control signal, inappropriate for robotic applications.

A remedy to the aforementioned problem is a control policy with autocorrelated noise. It assures both continuity of control signal and appropriate randomness in state trajectory. In this experimental study it was verified whether such a policy can be optimized with the use of known RL algorithms. Experiments with Half-Cheetah and Bioloid Walking revealed that such a policy may be applied without deterioration of learning efficiency or ultimate performance. Experiments with Cart-Pole Swing-Up and tenfold decrease of time discretization yielded ambiguous result. The ultimate control performance was as good as for traditional policy, but the speed of learning was smaller. It is easy to interpret that result. With coarse time distretization and independently selected control actions, it is straightforward to verify quality of each action and adjust policy to make this action more (or less) probable. With finer time discretization and the policy presented in Sec. III each action starts an experiment that is continued in further actions in which random elements are present autocorrelated with the first one. But this autocorrelation decreases with time passing. Some information is lost and thus slower learning. But this price may be worth paying in those robotic applications where fine time discretization has no alternative.

VI. Conclusions

In this paper the problem of continuity of control signal is analyzed, which arises when reinforcement learning is applied in robotics. It is demonstrated why this problem is so difficult to handle within the traditional reinforcement learning paradigm in which consecutive control actions are stochastically dependent only thorough the state. A control policy is introduced that is based on autocorrelated noise. It alleviates the problem of control signal discontinuity, and it may be optimized by known RL algorithms. In the experimental study that policy was applied to two simulated and one real robotic learning control problems. The policy was successfully optimized through reinforcement learning.

[2] The resulting control policy may be seen at: https://www.youtube.com/watch?v=O2rx4Bdwn24

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[2] J. Kober, D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, no. 11, pp. 1238-1274, 2013.

[3] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1-2, pp. 1-142, 2013.

[4] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, no. 1-2, pp. 171-203, 2011.

[5] P. Wawrzynski, "Reinforcement learning with experience replay for model-free humanoid walking optimization," *International Journal of Humanoid Robotics*, in press, 2014.

[6] H. Kimura and S. Kobayashi, "An analysis of actor/critic algorithm using eligibility traces: Reinforcement learning with imperfect value functions," in *Proc. the 15th ICML*, 1998, pp. 278-286.

[7] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Incremental natural actor-critic algorithms," *Automatica*, vol. 45, no. 11, 2009.

[8] P. Wawrzynski, "Real-time reinforcement learning by sequential actorcritics and experience repla," *Neural Networks*, vol. 22, pp. 1484-1497, 2009.

[9] P. Wawrzynski and A. K. Tanwani, "Autonomous reinforcement learning with experience replay," *Neural Networks*, vol. 41, pp. 156-167, 2013.

[10] K. Doya, "Reinforcement learning in continuous time and space," *Neural Computation*, vol. 12, no. 1, pp. 219-245, 2000.

Paweł Wawrzyński received his M.Sc. degree in computer science from Warsaw University of Technology in 2001 and, M.Sc degree in economics from Warsaw University in 2004, and Ph.D. degree in computer science from Warsaw University of Technology in 2005. Since 2006 he has been working as an assistant professor at Institute of Control and Computation Engineering in Warsaw, Poland. His research interests include robotics, machine learning, neural networks, and cognitive science.