ICIC Express Letters
Part B: Applications
Volume **7**, Number **10**, October **2016**

ICIC International ©2016 ISSN 2185-2766
pp. **2123–2128**

VISION-BASED REINFORCEMENT LEARNING IN ENVIRONMENTS WITH ACTION NOISE

Atsushi Ueno, Natsuki Kajihara and Tomohito Takubo

Graduate School of Engineering
Osaka City University
3-3-138, Sugimoto, Sumiyoshi-ku, Osaka 558-8585, Japan
{ ueno; takubo }@info.eng.osaka-cu.ac.jp; kajihara@kdel.info.eng.osaka-cu.ac.jp

Received March 2016; accepted June 2016

Abstract. *We have developed a vision-based reinforcement learning system that can learn appropriate suboptimal goal-directed behavior quickly. In this paper, we propose a new version of the system for tackling action noises. We make two improvements. First, we introduce a new definition of distance in the input space defined by the normalized Bag-of-Visual-Words representation. By using this definition, two inputs at two close points in position and orientation space tend to be close also in the input space. Second, we introduce fine control of actions for following the promising path obtained in learning. We verify the effectiveness of these two improvements by simulation experiments in a simple 3D room environment with some action noise.*
Keywords: Reinforcement learning, Vision, Goal-directed behavior, Action noise

1. Introduction.

Reinforcement learning (RL) is suitable for learning appropriate behavior in unknown environments. When acting in a real environment, visual information is so rich that it is very useful for agents in it to have vision. However, typical RL methods cannot handle raw visual input data since it is very high dimensional. So far, there have been two different approaches for tackling this problem. One utilizes some image feature descriptor in order to map images to inputs of an RL algorithm. The other utilizes deep neural networks in order to learn features and behavior.

In the former approach, the definition of features is man-made. Jodogne and Piater [1] proposed a method based on this approach. In the method, a feature-based image classifier is introduced in front of an RL algorithm. This method has been tested on real camera images and succeeded in learning goal-directed behavior in several tasks. It is, however, not applicable to learning of robots with vision because its learning is based on images that are collected in a manner unnatural for such robots: they are cropped out from a 2D image or collected from a fixed set of about a thousand images in a small state space. Pérez *et al.* [2] proposed a method that uses a *Bag-of-Visual-Words* (BoVW) [3] representation for each image as an input to RL. A BoVW for each image is a histogram of visual words appearing in the image. A visual word is a characteristic local feature. Many local features are collected from many images in advance and clustered into groups based on local brightness gradients. Each group corresponds to each visual word. The paper [2] tackled a task in which a robot with an on-board camera learns goal-directed behavior, but did not show effective results.

In the latter approach, the features themselves are learned based on the images faced by the agent in learning. Therefore, the same agent can learn flexibly in various environments. Mnih *et al.* [4] proposed excellent methods based on this approach. Their agent can learn appropriate behavior in various tasks, including a learning task of finding rewards in random 3D mazes using vision. A common weak point of such methods with deep neural networks is that they need a large amount of computational power in order to tune millions or tens of millions of parameters. For example, training took approximately 3 days on a standard multi-core CPU in the above random 3D maze task [4].

We have been developing a vision-based RL system that can learn appropriate suboptimal goal-directed behavior quickly [5]. This type of fast suboptimal learning is important for the following two reasons:

- It is suitable for tasks that do not require proficiency. It costs too much to acquire proficiency in tasks that may be performed only a few dozens of times in the learner's lifetime.
- It can provide a reasonable initial solution to a more exhaustive learning method. In a learning problem with a huge solution space, we think that two-stage learning is feasible. First, a fast learning method is used to find a suboptimal solution. Second, a more exhaustive learning method is used to refine it.

Our RL system has two features. First, the input space for RL is defined by the normalized BoVW representation as with the method of Pérez *et al.* [2]. Since visual words in this representation can be defined autonomously based on the images faced by the learning agent,[1] we think that BoVW is suitable as an input to vison-based learning without previous knowledge of the environment. Second, it uses a strongly exploitation-oriented RL method. In this method, the continuous input space is discretized into a lot of hyperspheres and the blank space. The inside region of each hypersphere is regarded as a discretized state. Each state has an assigned action which is executed every time when the current input gets in the region of the state. An agent with this method obtains chains of states from start points to goal points by trial-and-error learning from scratch. Once it gets to a goal point, the method constructs a chain from the success route removing all the closed paths on the route. Thereafter, it adheres to the path on the chain and does not search for better solutions as long as the agent continues to succeed. Though it can obtain only a suboptimal solution, and the knowledge about the environment it has obtained is limited around the obtained path, this limitation means the lightness of computation. Therefore, it is suitable for learning complicated tasks quickly. An agent with this RL system has the ability to certainly learn a good path from a start to a goal in a simple 3D room environment within a few dozens of trials. We, however, found that the ability deteriorates in cases with some action noise. In such cases, the agent tends to fail to follow the obtained path, which makes it vulnerable or lost.

In this paper, we propose improvements for this problem. These improvements are important because almost all actions in a real environment are noisy. First, we introduce a new definition of distance in the BoVW input space. By using this definition, two inputs that are made of two images photographed at two close points in position and orientation space tend to be close also in the input space. Second, we introduce fine control of actions for following the obtained path. We verify the effectiveness of these two improvements by simulation experiments in a simple 3D room environment with some action noise.

2. Previously Proposed System.

In this section, we explain our previously proposed RL system [5]. It specializes in RL in goal-directed tasks. An agent has a fixed start point and a fixed goal region. It starts from the start point and moves while searching the goal region. Time is discretized into time steps. At each time step, the agent takes an image with an on-board camera, generates an input to the RL system from the image, and decides and executes an action. When it gets to a point in the goal region, it receives a large *reward* and is brought back to the start point. In this paper, a sequence of actions from the start point is called an *episode*. The RL system continues learning through episodes. It can learn quickly an appropriate sequence of actions from the start point to the goal region. Neither any previous knowledge about the environment, such as a map,

[1]As we have seen above, the definition of underlying features is man-made.

nor any instruction is given to the agent. It has to execute the learning autonomously based only on images from the camera and rewards from the environment. The reward on arrival in the goal region is called the *goal reward*. A negative small reward (a *penalty*) is given when something wrong happens, such as bumping into a wall.

2.1. **State definition.** The definition of a state of this system is based on that of the method of Miyazaki *et al.* [6]. A state s is characterized by the center coordinate $\boldsymbol{\mu}_s$, an assigned action a_s, and the value V_s. Each state s is created based on an experience in which the agent got an input $\boldsymbol{x}$ and executed an action a. When creating state s, an isotropic Gaussian function with a standard deviation σ is constructed centering on $\boldsymbol{x}$ in the input space. $\boldsymbol{x}$ is saved as the center coordinate $\boldsymbol{\mu}_s$, and a is saved as the assigned action a_s. The region in which the product of the Gaussian function and V_s is greater than a threshold forms an n-dimensional hypersphere, which is defined as the region of state s. The assigned action of each state is executed every time when a new input gets in the state region. The value of each state is used for specifying the size and life-span of the state.

2.2. **Random-Walk Phase.** The learning is divided into two phases: Random-Walk Phase and Refining Phase. First, the agent is in Random-Walk Phase. In this phase, the agent takes an image and executes a random action at each time step. Each episode finishes when the agent gets to the goal region (a success episode) or exceeds the upper limit of time steps (a failure episode). The limit is denoted by Th_{step}. All the images taken in the episode are stored if it has been a success one, or deleted if it has been a failure one. Then, for either case, the agent is brought back to the start point. The agent keeps on executing episodes of random-walk behavior until the number of stored images becomes greater than a threshold Th_{image}. Then, the system uses a BoVW technique to create the input space and an initial set of states. All features in the first Th_{image} images in Random-Walk Phase are detected and described by the SURF descriptor [7], visual words are defined by clustering all the features, the space of the normalized histograms of them is defined as the input space for RL, and BoVW representations for all images in one of the shortest episodes are generated and normalized to create a set of states. In creating a state, if there is already another state at a very close position (the distance $< l_{near}$), only the one closer to the goal in the episode is kept. This is the same measure as used in [6] for removing closed paths in the episode. The values of all the created states are initialized to the initial value V_{init}, and updated by the goal reward in the manner described in below Section 2.3. They are registered into the main memory, and the learning transits to Refining Phase.

2.3. **Refining Phase.** At each time step, the agent takes an image and generates the current input. A state corresponding to the input is searched for among the registered states in the main memory. In this process, a score of the current input $\boldsymbol{x}$ is calculated for each state s by the following equation:

$$p_s(\boldsymbol{x}) = \exp\left[-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_s\|^2}{2\sigma^2}\right] \cdot V_s. \tag{1}$$

The state corresponding to $\boldsymbol{x}$ is the one with the highest score which is higher than a constant threshold p_{border}. The agent executes the assigned action of the state. If $\boldsymbol{x}$ has no corresponding state, a random action is executed, and a new state corresponding to it is created. Its value is set to the initial value V_{init}, and it is stored in the temporary memory.

When the agent obtains the goal reward r_{goal}, the values of all states in that episode are updated by the following update rule:

$$V_{s_t} \leftarrow V_{s_t} + r_{goal} \cdot \gamma^{L-t},$$

where s_t is the state at time step t ($t = 0, 1, 2, \dots, L$) in the episode, and $\gamma \in [0, 1]$ is a discount rate parameter. When the agent obtains a penalty, the value of the penalized state is updated by the following update rule:

$$V_s \leftarrow a_p \cdot V_s - b_p,$$

where a_p and b_p are penalty parameters. If the number of visits to any state exceeds a constant threshold Th_{visit}, it is judged that there is some closed path passing through the state. Then, all the states which are visited more than $Th_{visit} - 1$ times in the episode are judged to be included in the closed path and penalized together.

The termination condition of each episode is the same as in Random-Walk Phase. After an episode finishes, all the states in the temporary memory are registered into the main memory only if the episode has been a success one. They are deleted in all cases before the next episode starts.

3. **Proposed Improvements.**

We have developed a new version of our vision-based RL system by making two improvements for tackling action noises. Its processes are the same as in the previous version described in Section 2 except for these improvements.

3.1. **A new definition of distance in the input space.**

As shown in Equation (1), the previous version uses the Euclidean distance to measure between two points in the input space. We have found that two inputs that are made of two images photographed at two close points in position and orientation space are not always close to each other in the input space. Especially, when the numbers of features in the images are small, the distance tends to be estimated too large. The cause of this seems to be that the Euclidean distance is not suitable for measuring in the normalized BoVW input space, whose element is a kind of histogram.

For handling this problem, we introduce a new definition of distance in the input space based on the histogram intersection. It is used to measure the similarity between two histograms, $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$, and defined as follows:

$$H(\boldsymbol{h}_1, \boldsymbol{h}_2) = \frac{\sum_{i=1}^{n} \min(h_1(i), h_2(i))}{\sum_{i=1}^{n} h_1(i)},$$

where $h_a(i)$ ($a = 1, 2$) denotes the i-th component of histogram $\boldsymbol{h}_a$. The new definition of distance between two points, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, in the input space is as follows:

$$Dist(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1 - H(\boldsymbol{x}_1, \boldsymbol{x}_2)}{H(\boldsymbol{x}_1, \boldsymbol{x}_2)}.$$

This definition of distance is used in Equation (1) to decide the corresponding state to each input.

3.2. **Fine control of actions.**

We have assumed that the agent is assigned fixed length actions in experiments with the previous version. In environments with action noise, the agent tends to get out of the obtained path to the goal accumulating small errors of actions. Getting out of the path is unfavorable preventing the agent from getting to the goal and making the path vulnerable or lost. Therefore, we introduce fine control of actions for following the sequence of the inputs in the last success episode.

In Refining Phase, the agent checks if it is still on the following path by the following inequality: $Dist(\boldsymbol{x}_k, \boldsymbol{x}'_k) < l_{near}$ ($k = 1, 2, \dots, t$), where t is the current time step, $\boldsymbol{x}_k$ is the input at time step k in the current episode, and $\boldsymbol{x}'_k$ is the input at time step k in the last success episode. If it is still on the following path, at the next time step $t + 1$, it controls the length of each action to stop at a point as near as possible to the point in the input space in the last success episode. To put it in detail, it observes 21 inputs $\boldsymbol{x}_{t+1,j}$ ($j = 0, 1, \dots, 20$) at 21 points ranging from 90 percent to 110 percent of the fixed length of the action with 1 percent increments. The terminal point of the action and the input at time step $t+1$ are decided as follows:

$$\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x}_{t+1,j}} Dist(\boldsymbol{x}_{t+1,j}, \boldsymbol{x}'_{t+1}).$$

Once it gets out of the following path, it stops this action control from then on.

4. **Experiments.** We prepare a simple environment in order to verify the effectiveness of the improvements. Figure 1 shows the square room (1.4m×1.4m) environment with a robot and a square object (0.20m×0.20m) at the center of the room. The start point is located at 0.30 meters right of and 0.30 meters below the top left corner of the room, and the starting direction is down. The goal is a square region (0.35m×0.35m) located at the bottom right corner. All walls of the room and the object are filled with 32 photos from Caltech-256 and have adjacent no-entry zones 0.10 meters wide. The robot observes the environment with the mounted camera which takes images of 480×320 pixels and whose horizontal and vertical angles of view are 90 degrees and 67.5 degrees, respectively. It has three discrete actions: a 0.10-meter forward movement and 45-degree left and right rotations. All lengths of actions have errors, each of which is distributed uniformly between −10 and 10 percent of the specified length. In random selection, it selects a forward movement with a probability of 50 percent, and both left and right rotations are selected with a probability of 25 percent.

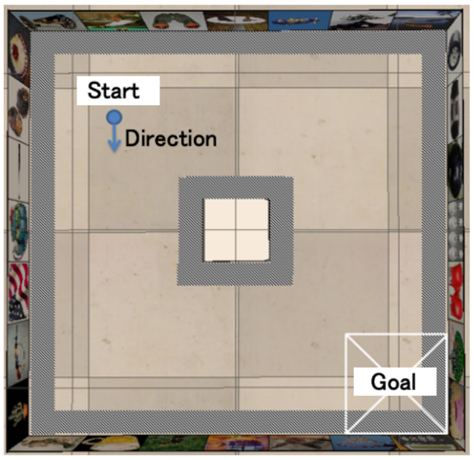

Figure 1. A square room environment

The parameters for learning are set as follows: $\sigma = 0.7$, $Th_{step} = 3000$, $Th_{image} = 1000$, $l_{near} = 2.0$, $V_{init} = 0.1$, $p_{border} = 0.001$, $Th_{visit} = 3$, $r_{goal} = 1.0$, $\gamma = 0.99$, $a_p = 0.1$, b_p for a closed path (b_p^{closed}) is set to 0.2, and b_p for a bump (b_p^{bump}) is set to 0.1. The number of dimensions of the input space is set to 1000.

Figure 2 shows the learning curves of the new and the previous version of the RL system averaged over 10 runs. The learning of the new version converges after approximately 50 episodes. We think that it is very quick in comparison with the training times of exhaustive learning methods, such as the method in [4].[2] The convergence value is 35.9, while the average by the previous version during the same period is about 64.5. This result shows that the learning performance has been substantially improved by the improvements for tackling action noises. We have checked and found that the numbers of removed states after convergence in the new version are substantially low: 4.3 per episode, while 24.8 per episode in the previous version. It shows that the new version is successful in maintaining

[2]Note that the target of the method in [4] is to learn a complex task exhaustively over a few or several dozens of hours, which is different from that of ours.

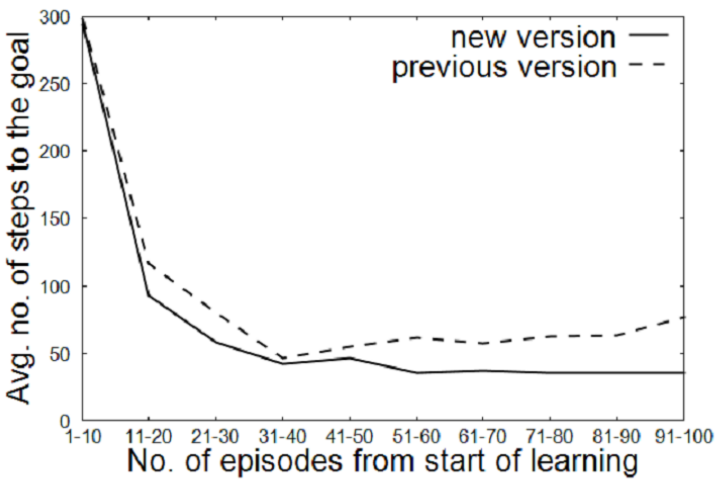

Figure 2. Learning curves. They are averaged in every 10 episodes.

the obtained path. The minimum number of time steps to the goal is 18 in this task. The new version takes nearly twice the time. This difference is, however, not important. It has quickly obtained and maintained paths to the goal without closed subpaths, which is precisely the purpose of our developing RL system.

5. **Conclusions.** In this paper, we have proposed a new version of our vision-based RL system for goal-directed tasks with action noise. We have made two improvements. First, we have introduced a new definition of distance in the input space defined by the normalized BoVW representation. By using this definition, two inputs at two close points in position and orientation space tend to be close also in the input space. Second, we have introduced fine control of actions for following the promising path obtained in learning. With these improvements, the agent has the ability to quickly obtain and maintain appropriate goal-directed behavior in the simple 3D room environment with some action noise. We are now trying to develop a vision-based RL robot in a real environment.

REFERENCES

[1] S. Jodogne and J. H. Piater, Closed-loop learning of visual control policies, *J. Artificial Intelligence Research*, vol.28, pp.349-391, 2007.

[2] X. Pérez, C. Angulo, S. Escalera and D. Pardo, Vision-based navigation and reinforcement learning path finding for social robots, *Achievements and New Opportunities in Computer Vision: Proc. of the 5th CVC Workshop*, pp.121-125, 2010.

[3] J. Yang, Y.-G. Jiang, A. G. Hauptmann and C.-W. Ngo, Evaluating bag-of-visual-words representations in scene classification, *Proc. of Int. Workshop on Multimedia Information Retrieval*, pp.197-206, 2007.

[4] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, *Asynchronous Methods for Deep Reinforcement Learning*, arXiv:1602.01783 [cs.LG], 2016.

[5] A. Ueno, N. Kajihara, N. Fujii and T. Takubo, Vision-based path learning for home robots, *Proc. of the 10th Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, pp.411-414, 2014.

[6] K. Miyazaki, H. Kimura and S. Kobayashi, An extension of the rational policy making algorithm to continuous state spaces, *Trans. Japanese Society for Artificial Intelligence*, vol.22, no.3, pp.332-341, 2007 (in Japanese).

[7] H. Bay, T. Tuytelaars and L. V. Gool, SURF: Speeded up robust features, *Proc. of European Conf. on Computer Vision*, pp.404-417, 2006.