Reinforcement Learning in Computer Vision

A.V. Bernstein[1], E.V. Burnaev[2]
Skolkovo Institute of Science and Technology, Moscow, Russia
[1]A.Bernstein@skoltech.ru, [2]E.Burnaev@skoltech.ru

ABSTRACT

Nowadays, machine learning has become one of the basic technologies used in solving various computer vision tasks such as feature detection, image segmentation, object recognition and tracking. In many applications, various complex systems such as robots are equipped with visual sensors from which they learn state of surrounding environment by solving corresponding computer vision tasks. Solutions of these tasks are used for making decisions about possible future actions. It is not surprising that when solving computer vision tasks we should take into account special aspects of their subsequent application in model-based predictive control. Reinforcement learning is one of modern machine learning technologies in which learning is carried out through interaction with the environment. In recent years, Reinforcement learning has been used both for solving such applied tasks as processing and analysis of visual information, and for solving specific computer vision problems such as filtering, extracting image features, localizing objects in scenes, and many others. The paper describes shortly the Reinforcement learning technology and its use for solving computer vision problems.

Keywords: Computer vision, reinforcement learning

1. INTRODUCTION

The general goal of Computer vision (CV) is high-level understanding of content of images (recognizing objects, scenes, or events; classification of detected objects, discovering geometric configuration and relations between objects). In other words, machine vision models the world from images and videos received from visual sensing system by recognizing an environment and estimating its required features.

In real applications, this high-level understanding of content of images, received from certain visual sensing system, is only the first key step in solving various specific tasks such as mobile robot navigation in uncertain environments, road detection in autonomous driving systems, etc. In other words, specific decision making problems are solved on the basis of information about surrounding environment extracted from captured images. Therefore, efficiency of used image understanding technologies should be estimated from efficiency of decisions made on basis of the resulted "surrounding environment understanding"; such decisions, in turn, are determined by results of previous actions of a system. Thus, it is not surprising that when solving CV tasks we should take into account special aspects of their subsequent application in model-based predictive control. For example, navigation decisions for a mobile robot [1–3] are made based on estimated robot localization, which in turn is a result of specific CV task, namely, passive vision-based robot position estimation [4–7]. Thus, various solutions of this CV task should be estimated on basis of results of performed actions (further navigation and path planning).

Nowadays, one of the main drivers of high-level understanding of surrounding environment from images is the application of machine learning methods to CV tasks (image registration, segmentation, classification; 3D reconstruction; object detection and tracking). Vision tasks are formulated as learning problems, and machine learning is an essential and ubiquitous tool for automatic extraction of patterns or regularities from images. Thus, machine learning is now a core part of machine vision. Reinforcement learning (RL) is one of modern machine learning technologies in which learning is carried out through interaction with the environment and allows taking into account results of decisions and further actions based on solutions of corresponding CV tasks.

In the paper, we consider the use of RL technologies in solving various CV tasks concerning processing and analysis of visual information such as filtering, extracting image features, localizing objects in scenes, and many others. The paper is organized as follows. Section 2 provides a short description of the RL technology. Section 3 contains examples of applying this technology to typical CV tasks.

2. REINFORCEMENT LEARNING TECHNOLOGY

RL has gradually become one of the most active research areas in machine learning, artificial intelligence, and neural network research, and has developed strong mathematical foundations and impressive applications. One of the primary goals of this field of artificial intelligence is to produce fully autonomous systems that interact with their environments to learn optimal behaviors, improving over time through trial and error process. The RL technology is based on a simple idea to create a learning system that wants something, that adapts its behavior in order to maximize a special signal from its environment [8–11]. RL is close to such important direction in statistics as adaptive design of experiments [12, 13].

The main element is a learning of an active decision-making autonomous agent (or, simply, the agent), which learns its behavior through trial-and-error interactions with a dynamic environment (described as a dynamic process) to achieve a goal despite uncertainty about the environment. At successive time steps (t), the agent makes an observation of the environment state (s_t), selects an action (a_t) and applies it back to the environment, modifying the state (s_{t+1}) at next moment $(t+1)$. The goal of the agent is to find adequate actions for controlling this process. In order to do that in an autonomous way, it uses a technique known as the RL.

Formally, the RL can be modeled as a Markov decision process, which consists of:

- a set of environment states $S = \{s\}$, plus a distribution of starting states $p(s_0)$ at starting moment $t = 0$;
- a set of actions $A = \{a\}$;
- a transition dynamics (probability function) $T(s_{t+1}|s_t, a_t)$ that map a pair, consisting of a state s_t and an action a_t at time t onto a distribution of states s_{t+1} at time $(t+1)$.

Beyond this model, one can identify three main subelements of the RL system: a policy, a reward function, and a value function. A policy is defined as a map $\pi : S \times A \to [0,1]$ that determines the probability $\pi(a|s) = \mathbb{P}(a_t = a|s_t = s)$ of taking action $a_t = a$ at moment (t) in state $s_t = s$.

A reward function is a map $r : S \times A \to \mathbb{R}$ that determines an immediate/instantaneous reward (return) $r_t = r(s_t, a_t)$ that is received (earned) at moment t, in going from state s_t to state s_{t+1} under action a_t. Denote by $p(s_t, a_t, s_{t+1})$ the probability of this transition with taking into account a randomness of the action a_t (with probability $\pi(a_t|s_t)$) and the transition $s_t \to s_{t+1}$ (with probability $T(s_{t+1}|s_t, a_t)$). For simplicity, we consider only deterministic (nonrandomized) policies when the action a is a nonrandom function $a = \pi(s)$ of the state s; thus, $r_t = r(s_t, \pi(s_t))$.

Introduce a discount factor $\gamma \in (0,1)$, which forces recent rewards to be more important than remote ones (i.e., lower values place more emphasis on immediate rewards). Under chosen policy π and starting state s, a value function

$$V_\pi(s) = \lim_{M\to\infty} \mathbb{E}\left[\sum_{t=0}^{M} \gamma^t \times r(s_t, \pi(s_t))|s_0 = s\right] \tag{1}$$

is defined as expected discounted reward and intuitively denotes the total discounted reward earned along an infinitely long trajectory starting at the state s, if policy π is pursued throughout the trajectory. Thus, $V_\pi(s) = \mathbb{E}[R|s,\pi]$, where $R = \sum_{t=0}^{\infty} \gamma^t \times r_t$ is a discounted reward.

The problem is to find a stationary policy π^* of actions $\{a_t = \pi^*(s_t)\}$ called optimal policy which maximizes the function $V_\pi(s)$ for all starting states: $V_{\pi^*}(s) = \max_\pi V_\pi(s) \equiv V^*(s)$. Under appropriate assumptions, it was proven [14] that optimal value function $V^*(s)$ is unique, although there can be more than a single optimal policy π^*. Dynamic Programming [14] is the theory behind evaluation of an optimal stationary policy π^* for the problem above and encompasses a large collection of techniques for the evaluation.

This optimal value function can be defined as the solution to the simultaneous equations

$$V^*(s) = \max_{a\in A}\left\{ r(s,a) + \gamma \times \sum_{s'\in S} T(s'|s,a) \times V^*(s') \right\}, \tag{2}$$

and, given the optimal value function (2), we can specify the optimal policy as

$$\pi^*(s) = \arg\max_{a\in A}\left\{ r(s,a) + \gamma \times \sum_{s'\in S} T(s'|s,a) \times V^*(s') \right\}. \tag{3}$$

Therefore, one way to find the optimal policy is to find the optimal value function (2) that can be determined by a simple iterative algorithm called value iteration, which can be shown to converge to the correct V^* values [15, 16]. Under known model consisting of transition probability function $T(s'|s,a)$ and reward function $r(s,a)$, these techniques use various iteration schemes and Monte-Carlo techniques for obtaining an optimal policy.

The RL technology is primarily concerned with how to obtain the optimal policy when such model is not known in advance. The agent must interact with the environment directly to obtain information which by means of an appropriate algorithm can be processed to produce an optimal policy.

The most popular technique in RL is Q-learning [17, 18] which is a model-free method used to find an optimal action-selection policy for any given (finite) Markov decision process. It works by learning an action-valued function

$$Q^*(s,a) = r(s,a) + \gamma \times \sum_{s' \in S} T(s'|s,a) \times \max_{a'} Q(s',a'), \tag{4}$$

written recursively, that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. Then $V^*(s) = \max_{a \in A} Q^*(s,a)$ and $\pi^*(s) = \arg\max_{a \in A} Q^*(s,a)$.

Q-learning consists in approximating of the action-valued function $Q^*(s,a)$ (4) and uses an experience tuple (s,a,r,s') summarizing a single transition in the environment, in which s is the agent's state before the transition, a is its choice of action, r is the instantaneous reward it receives, and s' is its resulting state. There are various current policies how to choose the action. Initially, since the agent knows nothing about what it should do, it acts randomly. When it has some current version $Q(s,a)$ of the action-valued function, it can use a greedy policy, choosing the action which maximizes this function. Another policy called ε-greedy is to choose the greedy policy with probability $(1-\varepsilon)$ and random action with probability ε.

Let $Q_{old}(s,a)$ be an old value of the action-valued function which is updated from the received experience tuple (s,a,r,s') resulting in a new value

$$Q_{new}(s,a) = Q_{old}(s,a) + \alpha \times \left[r(s,a) + \gamma \times \max_{a'} Q_{old}(s',a') - Q_{old}(s,a) \right], \tag{5}$$

here α is a learning rate, decreasing slowly. After some trial-and-error interactions, the agent begins to learn its task and performs better and better. If each action is executed in each state an infinite number of times on an infinite run and α is decaying with appropriate speed, then Q values converge with probability 1 to $Q^*(s,a)$ (4) [16, 19, 20].

Although RL had some successes in the past, previous approaches lacked scalability and were inherently limited to fairly low-dimensional problems. These limitations exist because RL algorithms share the same complexity issues as other algorithms: memory complexity, computational complexity, and as in case of machine learning algorithms, sample complexity [21]. Deep learning [22] is a modern tool which, relying on the powerful function approximation and representation learning properties of deep neural networks, has allowed to overcome these problems. Deep learning has accelerated progress in RL with the use of deep learning algorithms within RL defining the field of Deep Reinforcement Learning (DRL) [23, 24]. Deep learning enables RL to scale to decision-making problems that were previously intractable, i.e., settings with high-dimensional state and action spaces. DRL algorithms have already been applied to a wide range of problems, such as robotics, where control policies for robots can now be learned directly from camera inputs in the real world, succeeding controllers that used to be hand engineered or learned from low-dimensional features of robot's states.

3. REINFORCEMENT LEARNING IN COMPUTER VISION TASKS

Computer vision tasks such as image processing and image understanding involve solving decision making problems, and recent research has shown that RL can be applied to such problems. A significant barrier to solving image-based problems with RL is massive amount of data involved. This complicates the task of deriving state information for learning. It becomes impossible to use tabular methods to store past experience. Not only do tabular approaches present unrealistic memory requirements, but for large state spaces, an agent is not able to visit all state-action pairs, and thus the time needed to fill these tables becomes increasingly problematic. In case of a large and continuous state-space the problem becomes intractable. This is known as the curse of dimensionality and requires some form of generalization in addition to careful feature selection [25].

We give a short overview of successful applications of RL to solving parameter selection problem in various computer vision tasks. Many multi-step computer vision algorithms involve various parameters which should be chosen in an "optimal" way in each processing step of the computer vision procedures such as constructing membership functions in fuzzy image processing filters to represent "brightness" of a particular image [25], image segmentation [26–30], edge detection [31, 32], extracting image features [33–37]. Here we consider application of RL to the parameter selection problem in two examples: in both cases an RL agent helps to find an optimal threshold for edge detection and feature selection algorithms.

3.1 Parameter Selection in Edge Detection Algorithm

An edge detector is a simple algorithm to extract edges from an image. Applying an edge detector to an image generates a set of connected curves which follow boundaries of objects (real objects, surfaces, texture changes). Most edge detection methods use either first-order or second-order derivatives, and known Sobel, Canny and Prewit edge detectors are just some examples based on first-order derivative information. Most edge detection methods are based on a threshold that says how large an intensity change between two neighboring pixels must be in order to say that there should be an edge between these pixels. Most of the times, the threshold depends on the image, its noise level or its details. As a result, the threshold is chosen manually, which makes these methods difficult to apply on large image datasets [32].

RL-procedure is based on an environment (consisting of thresholds s in Canny and Sobel edge detector algorithms), on a general RL agent with only two possible actions (to increment or decrement state's value), and on a reward r equal to a well-known Pratt Figure of Merit measure of quality [38] of edge detector's outputs for a given input image under chosen state (threshold). Learning algorithm, in which greedy policy is used, calculates the Q-function (4). The Sobel edge detector extracts edges from the input images using a threshold value provided by the RL agent, the extracted edges are assessed by evaluating a Pratt Figure of Merit module (whose result is used as a reward for the RL agent), which takes the corresponding action and provides another edge detector threshold. The learning process continues until the optimal threshold is found.

3.2 Image Classification using Visual Features

The goal of image classification is to map an image to a class of objects. Recent successes in visual object recognition are due to the use of local-appearance approaches [39–41]. Such approaches first locate highly informative patterns in an image and in a picture of the object to be recognized, using interest point detectors (for example, edge detectors), then match these interest points using a local description of their neighborhood, called visual features. If there are enough matches, the image is taken as belonging to the object class. As visual features are vectors of real numbers, there exists an unbounded number of features.

A large number of non-trivial, powerful techniques devoted to the visual recognition of objects have been developed during the last decades, and RL is one of tools that selects the best descriptor for every image from a given set.

The key idea in RL of Visual Classes, where the agent is faced with visual inputs, is to focus the attention of the agent on a small number of very distinctive visual features that allow the agent to reason upon visual classes rather than raw pixels, and that enhance its generalization capabilities [33, 36, 40]. The challenging problem is to refine visual classes dynamically when the agent identifies inconsistencies in earned discounted returns when faced with that class. Therefore, the solution to this problem (the learning algorithm) will have to solve simultaneously a computer vision problem (image classification) and an RL problem (construction of the optimal control law). In other words, the solution requires joining RL technique with a Visual Object Recognition framework.

The RL-based visual feature extracting system includes some embedded supervised learning algorithm (particular Image classifier) which have to distinguish between visual inputs and able to refine a class on request by learning a new distinctive visual features that are powerful enough to distinguish any functionally-distinguishable percept. The classifier translates a low-level information (raw values of pixels) into a high-level information (image class) that will itself feed the RL module (algorithm). Based on this information, the RL agent has to inform the image classifier when learning of a new visual class is required. Since there is no external supervisor telling the agent when a refinement is needed, the RL algorithm can only rely on some robust criterion (for example, the aliasing criterion in [33]) able to decide when the classification is not accurate enough.

Formally, the states are represented by considered images and the agent has two possible action — to require refinement of current classes or not. Used robust criterion provides a return (reward or punishment) that evaluates decision when classification is sufficiently accurate or not.

The agent consequently learns visual classes only through interactions (tuples) $(s_t, a_t, r_{t+1}, s_{t+1})$. Intuitively speaking, the role of the agent is to identify functionally distinguishable percepts: it should distinguish between percepts that involve different returns when it chooses the same actions.

3.3 Other examples of RL-based algorithms in Computer Vision

Many decisions made by a robot are based on visual information (images) captured by robot's visual systems (cameras) [1, 2, 36, 42] with the use of extracted visual features describing environment for a current task. After that, in response to observations from a camera, robot's control system chooses actions that move the robot in order to reach a target configuration. Visual servoing, or a visumotor learning (learning vision-based manipulation skills) is a classical problem in robotics [43–46]. RL techniques allow constructing learned policy (choice of an action from current knowledge about the environment), which maps raw captured images directly to robot's actuating mechanism (for example, torques at robot's motors) [47–55].

Other successful applications of RL technique concern constructing a fast classifier that predicts salient objects given large-scale visual features [56], predicting collision-free motor commands from monocular images [57], choosing the best descriptor from a given bag of features corresponding to a given image, and selecting the best classifier from this bag

[35, 36], visual tracking in videos [58], searching of multiple objects under interactions in a given image [59], active detection for localizing objects in scenes [60], etc.

4. CONCLUSIONS

We briefly reviewed the RL technology. We provided a short overview of modern applications of the RL technology to Computer Vision tasks such as filtering, extracting image features, localizing objects in scenes.

Acknowledgements: The first author was supported by the Russian Science Foundation grant (project 14-50-00150). The second author was supported by the RFBR grants 16-01-00576 A and 16-29-09649 ofi_m.

REFERENCES

[1] J. Pauli, “Learning-Based Robot Vision,” Lecture Notes in Computer Science, vol. 2048 ‘Principles and Applications,’ Springer, Heidelberg, 292 pp., (2001).
[2] G.N. DeSouza and A.C. Kak, “Vision for mobile robot navigation: A survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(2), 237-267, (2002).
[3] F. Bonin-Font, A. Ortiz, and G. Oliver, “Visual navigation for mobile robots: A survey,” Journal of Intelligent and Robotic Systems, 53(3), 263-296, (2008).
[4] J. Ham, Y. Lin, and D.D. Lee, “Learning nonlinear appearance manifolds for robot localization,” In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), 1239-1244, (2005).
[5] J.L. Crowley and F. Pourraz, “Continuity Properties of the Appearance Manifold for Mobile Robot Position Estimation,” Image and Vision Computing, 19(11), 741-752, (2001).
[6] A.P. Kuleshov, A.V. Bernstein, and E.V. Burnaev, “Mobile Robot Localization via Machine Learning”, Lecture Notes in Computer Science, vol. 10358 ’Machine Learning and Data Mining in Pattern Recognition’, Springer International Publishing AG, Verlag, 276-290, (2017).
[7] A. Kuleshov, A. Bernstein, E. Burnaev, Yu. Yanovich, “Machine Learning in Appearance-based Robot Self-localization”, 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE Conference Publications, (2017).
[8] L. P. Kaelbling, M. L. Littman, and A. P. Moore, “Reinforcement learning: A survey,” Journal of Artificial Intelligence Research, 4, 237-285, (1996).
[9] C. Ribeiro, “Reinforcement Learning Agents,” Artificial Intelligence Review, 17(3), 223-250, (2002).
[10] R. Sutton and A. Barto, “Reinforcement learning: an introduction (Adaptive computation and machine learning),” MIT Press, Cambridge, Mass., (1998).
[11] A. Gosavi, “Reinforcement Learning: A Tutorial Survey and Recent Advances,” INFORMS Journal on Computing, 21(2), 178-192, (2008).
[12] E. Burnaev, M. Panov. “Adaptive design of experiments based on gaussian processes,” Lecture Notes in Artificial Intelligence. Proceedings of SLDS-2015. A. Gammerman et al. (Eds.), vol. 9047, 116–126, (2015).
[13] E. Burnaev, I. Panin, B. Sudret. “Effecient Design of Experiments for Sensitivity Analysis based on Polynomial Chaos Expansions,” Ann. Math. Artif. Intell., (2017), doi:10.1007/s10472-017-9542-1
[14] M. L. Puterman, “Markovian Decision Processes - Discrete Stochastic Dynamic Programming,” New York, John Wiley& Sons Inc., (1994).
[15] R. Bellman, “Dynamic Programming,” Princeton University Press, Princeton, NJ, (1957).
[16] D.P. Bertsekas, “Dynamic Programming: Deterministic and Stochastic Models,” Prentice-Hall, Englewood Cliffs, NJ, (1987).
[17] C.J.C.H. Watkins, “Learning from Delayed Reward,” PhD thesis, Kin’s College, University of Cambridge, (1989).
[18] C.J.C.H. Watkins and P. Dayan, “Q-learning,” Machine Learning, 8(3), 279-292, (1992).
[19] J. Tsitsiklis, “Asynchronous stochastic approximation and Q-learning,” Machine Learning, 16(3),185-202, (1994).
[20] T. Jaakola, M. Jordan, and S. Singh, zzOn the convergence of stochastic iterative dynamic programming algorithms,” Neural Computation, 6(6), 1185-1201, (1994).
[21] A.L. Strehl, L. Li, E. Wiewiora, J. Langford, and M.L. Littman, “PAC Model-Free Reinforcement Learning,” in Proceedings of the Twenty-Third International Conference on Machine Learning (ICML-06), 881-888, (2006).
[22] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” Nature, 521(7553), 436-444, (2015).
[23] K. Arulkumaran, M.P. Deisenroth, M. Brundage, and A.A. Bharath, “A Brief Survey of Deep Reinforcement Learning,” in arXiv:1708.05866v2 [cs.LG], 1-16, (2017).
[24] Y. Li, “Deep Reinforcement Learning: An Overview,” in arXiv:1701.07274v5 [cs.LG], 1-70, (2017).
[25] G.W. Taylor, “A Reinforcement Learning Framework for Parameter Control in Computer Vision Applications,” in Proceedings of the First Canadian Conference on Computer and Robot Vision (CRV’04), 17-19 May 2004, Publisher: IEEE, 1-8, (2004).
[26] J. Peng, and B. Bhanu, “Closed-Loop object recognition using reinforcement learning”, IEEE Transaction on Pattern Analysis and Machine Intelligence, 20(2), 139-154, (1998).

[27] F. Sahba, H.R. Tizhoosh, and M. Salama, “Application of opposition-based reinforcement learning in image segmentation,” in Proceedings of the IEEE Symposium on Computational Intelligence in Image and Signal Processing, Honolulu, HI, 246-251, (2007).
[28] M. Shokri, and H.R. Tizhoosh, “A reinforcement agent for threshold fusion,” Appl. Soft Comput., 8, 174-181, (2008).
[29] F. Sahba, H.R. Tizhoosh, and M. Salama, “Application of reinforcement learning for segmentation of transrectal ultrasound images,” Biomed Central Medical Imaging, 8(8), 1-10, (2008).
[30] S. Ghajari, M. Bagher, and N. Sistani, “Improving the quality of image segmentation in Ultrasound images using Reinforcement Learning,” Communications on Advanced Computational Science with Applications, 2017(1), 33-40, (2017).
[31] N. Siebel, S. Grunewald, and G. Sommer, “Created edge detectors by evolutionary reinforcement learning,” Evolutionary Computation, IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong, 2008, 3553-3560, (2008).
[32] A. Gherega, R.M. Udrea, and M. Radulescu, “A Q-Learning Approach to Decision Problems in Image Processing,” in Proceedings of the Fourth International Conferences on Advances in Multimedia (MMEDIA 2012), April 29 - May 4 2012, Wilmington, DE, 60-66, (2012).
[33] S. Jodogne, and J.H. Piater, “Interactive Selection of Visual Features through Reinforcement Learning,” in Proceedings of the 24th SGAI international conference on innovative techniques and applications of artificial intelligence, 285-298, (2004).
[34] M. Pinol, A.D. Sappa, and R. Toledo, “Multi-Table Reinforcement Learning for Visual Object Recognition,” in Proceedings of the Fourth International Conference on Signal and Image Processing (ICSIP 2012), 469-479, (2012).
[35] M. Pinol, A.D. Sappa, A. Lopez, and R. Toledo, “Feature Selection Based on Reinforcement Learning for Object Recognition,” Adaptive Learning Agent Workshop, 4-8 June, Valencia, Spain, 33-39, (2012).
[36] M. Pinol, “Reinforcement learning of visual descriptors for object recognition,” PhD thesis, Universitat Autonoma de Barcelona, Spain, 1-109, (2014)
[37] De-Rong Liu, Hong-Liang Li, and D. Wang, “Feature Selection and Feature Learning for High-dimensional Batch Reinforcement Learning: A Survey,” International Journal of Automation and Computing, 12(3), 229-242, (2015).
[38] W.K. Pratt, “Digital image processing,” John Wiley and Sons, New York, (1991).
[39] A.V. Bernstein, “Machine Vision and Appearance based Learning,” Proceedings of SPIE, vol. 10341 Ninth International Conference on Machine Vision (ICMV 2016), 103411O (17 March 2017), Publisher: SPIE, Washington, USA, 1-6, (2017).
[40] J.H. Piater, “Visual Feature Learning,” PhD thesis, Computer Science Department, University of Massachusetts, Amherst, MA, (2001).
[41] F. Zhu, “Visual Feature Learning,” PhD thesis, Department of Electronic and Electrical Engineering, University of Sheffield, (2015).
[42] A.V. Bernstein, “Manifold Learning in Machine Vision and Robotics,” Proceedings of SPIE, vol. 10253 ’2016 International Conference on Robotics and Machine Vision (ICRMV 2016), 102530G (8 February 2017)’, Publisher: SPIE, Washington, USA, 1-6, (2017).
[43] P.I. Corke, “Visual control of robot manipulators – A review,” Visual servoing, 7, 1–31, (1993).
[44] S. Hutchinson, G.D. Hager, and P.I. Corke, “A tutorial on visual servo control,” IEEE Transactions on Robotics and Automation, 12(5), 651-670, (1996).
[45] B. Espiau, F. Chaumette, and P. Rives, “A new approach to visual servoing in robotics,” IEEE Transactions on Robotics and Automation, 8(3), 313-326, (2002).
[46] F. Chaumette, and S, Hutchinson, “Visual servo control. I. Basic approaches,” IEEE Robotics & Automation Magazine, 13(4), 82-90, (2006).
[47] J. Kober, E. Oztop, and J. Peters, “Reinforcement learning to adjust robot movements to new situations,” Robotics: Science and Systems, MIT Press Journal, 6, 33-40, (2011).
[48] M. Deisenroth, C. Rasmussen, and D. Fox, “Learning to control a low-cost manipulator using data-efficient reinforcement learning,” Robotics: Science and Systems, MIT Press Journal, 7, 57-64, (2012).
[49] S. Lange, M. Riedmiller, and A. Voigtlander, “Autonomous reinforcement learning on raw visual input data in a real world application”, in Proceedings of the International Joint Conference on Neural Networks, IEEE, 1-8, (2012).
[50] T. Lampe, and M. Riedmiller, “Acquiring visual servoing reaching and grasping skills using neural reinforcement learning,” in Proceedings of the International Joint Conference on Neural Networks, IEEE, 1-8, (2013).
[51] M. Sadeghzadeh, D. Calvert, and H.A Abdullah, “Self-learning visual servoing of robot manipulator using explanation-based fuzzy neural networks and Q-learning,” Journal of Intelligent and Robotic Systems, 78(1), 83-104, (2015).
[52] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, “Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control,” In arXiv:1511.03791v2 [cs.LG], 1-8, (2015).
[53] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” Journal of Machine Learning Research, 17(39), 1-40, (2016).
[54] C. Finn, X.Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, “Deep Spatial Autoencoders for Visuomotor

Learning," in arXiv:1509.06113v3 [cs.LG], 1-9, (2016).
[55] A.X. Lee, S. Levine, and P. Abbeel, "Learning Visual Servoing with Deep Features and Fitted Q-iteration," in arXiv:1703.11000v2 [cs.LG], 1-20, (2017).
[56] C. Craye, D. Filliat, and J.-F. Goudou, "Environment Exploration for Object-Based Visual Saliency Learning," in Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA 2016), Stockholm, Sweden, 2303-2309, (2016).
[57] F. Sadegh, S. Levine, "CAD2RL: Real Single-Image Flight Without a Single Real Image," in arXiv:1611.04201v4 [cs.LG], 1-12, (2017).
[58] D. Zhang, H. Maei, X. Wang, and Y.-F. Wang, "Deep Reinforcement Learning for Visual Object Tracking in Videos," in arXiv:1701.08936v2 [cs.CV], 1-10, (2017).
[59] X. Kong, B. Xin, Y. Wang, and G. Hua, "Collaborative Deep Reinforcement Learning for Joint Object Search," in arXiv:1702.05573v1 [cs.CV], 1-11, (2017).
[60] J.C. Caicedo, and S. Lazebnik, "Active Object Localization with Deep Reinforcement Learning," in arXiv:1511.06015v1 [cs.CV], 1-9, (2015).