Discounted Sampling Policy Gradient for Robot Multi-objective Visual Control

Meng Xu[1(✉)], Qingfu Zhang[1,2], and Jianping Wang[1,2]

[1] Department of Computer Science, City University of Hong Kong, Hong Kong, China
mxu247-c@my.cityu.edu.hk, {qingfu.zhang,jianwang}@cityu.edu.hk
[2] Shenzhen Research Institute, City University of Hong Kong, Shenzhen, China

Abstract. Robot visual control often involves multiple objectives such as achieving high efficiency, maintaining stability, and avoiding failure. This paper proposes a novel Vision-Based Control method (VBC) with the Discounted Sampling Policy Gradient (DSPG) and Cosine Annealing (CA) to achieve excellent multi-objective control performance. In our proposed visual control framework, a DSPG learning agent is employed to learn a policy estimating continuous kinematics for VBC. The deep policy maps the visual observation to a specific action in an end-to-end manner. The DSPG agent finally can update the policy to obtain the optimal or near-optimal solution using shaped rewards from the environment. The proposed VBC-DSPG model is optimized using a heuristic method. Experimental results demonstrate that the proposed method performs very well compared with some classical competitors in the multi-objective visual control scenario.

Keywords: Multi-objective visual control · Kinematics · Discounted sampling policy gradient · Cosine annealing

1 Introduction

Vision-based control is a mainstream approach to drive the mobile robot to achieve multi-objective control by visual feedback, such as navigation and path planning [1,2]. The error of the feature points on the image plane is used as the feedback to drive the mobile robot towards the targets. Feature error reflects the difference between the current image features and the desired image features. One practical advantage of the VBC method is that there is no need for a

This work was supported by National Key Research and Development Project, Ministry of Science and Technology, China (Grant No. 2018AAA0101301), National Natural Science Foundation of China (Grant No. 61876163), in part by Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. JCYJ20200109143223052) and Hong Kong Research Grant Council (GRF 11200220).

H. Ishibuchi et al. (Eds.): EMO 2021, LNCS 12654, pp. 441–452, 2021.
https://doi.org/10.1007/978-3-030-72062-9_35

geometric model of the target. A key issue is how to achieve an efficient mapping from the 2D feature errors to the velocities in the 3D Cartesian coordinate system. The kinematics bridges two [3]. The multi-objective visual control requires the motion process to consider different objectives such as stability, rapidity, and keeping the target. To achieve the trade-off between different objectives, a feasible way is to use modular controllers [4]. Another way is to regard performance efficiency as the primary objective and others as constraints [5]. The latter can simplify the multi-objective control task into a single objective task with constraints. Previous methods heavily rely on dynamic modeling, and it is difficult to make these dynamic models scalable and generalizable.

The Jacobian matrix transforms the multi-objective visual control problem into a matrix estimation and optimization problem [6]. Three conventional methods are used to approximate the Jacobin matrix for multi-objective visual control. The first one is to employ the current matrix; one is the desired matrix and the other is the average matrix [7]. The universal Jacobian matrix involves a linear combination of the current matrix and the desired matrix. A control parameter ranging from 0 to 1 is used to balance the current matrix and the desired one. It is appropriate to assign a smaller value for the parameter to ensure stable convergence when a mobile robot is close to the target. Instead, a larger value achieves faster convergence [8]. Previous work used the proportional-integral-differential (PID) method to estimate the Jacobian matrix but the performance is limited due to the unsuitable parameters [9]. Until now, the estimation of the matrix still needs human experts' experience. Reinforcement Learning (RL) does not require any prior knowledge about the environment and it drives agents to explore the environment to obtain the best solution via trial and error.

RL methods have been demonstrated to have excellent performance in developing more advanced robots, including multi-objective visual control systems [10]. [11] developed a novel RL method to stabilize a biped robot on a rotating platform. The proposed method addresses the overfitting problem with guaranteed model complexity. However, these conventional methods with tabular RL have a bottleneck due to its discrete action space. It is also helpless for a high-dimensional or continuous control task. Advances in deep learning models have made it possible to extract high-level features from sensory data, and it provides an opportunity of scaling to problems with infinite state spaces. Deep Reinforcement Learning (DRL) applies the computational power of deep learning to relaxing the curse of dimensionality for complex tasks [12]. It can learn a direct mapping from the state space to the policy space in an end-to-end way. Carlos Sampedro et al. developed a deep reinforcement learning controller to achieve the mapping from state space to linear velocity commands for multirotor aerial robots [13]. The learning performance of the DRL based controller is significantly impacted by hyperparameters. A heuristic strategy is an approach to Hyperparameters [14].

This paper proposes a reinforcement learning method with a heuristic strategy to obtain the transient matrix for multi-objective VBC systems. DSPG algorithm employs discounted return sampling which provides an opportunity for

the control system with continuous spaces. The proposed method reduces the computation of the performance gradient to an expectation that is estimated by the discounted sampling return. It requires only a fixed computing interval instead of a predetermined environmental model. The deep policy network produces control actions via an on-policy learning method. An appropriate action hence is chosen to improve the performance of the conventional multi-objective visual control, in terms of the convergence rate and stability. The hyperparameter for the proposed RL based controller is optimized by a heuristic strategy, which can give better performance to the learning model of mobile robots.

Major contributions of this work are stated as follows:

1) A VBC method with the DSPG algorithm (VBC-DSPG) is proposed for the multi-objective control. The DSPG algorithm is used to select the time-varying Jacobian matrix online.
2) A parameter tuning method with the Cosine Annealing for the VBC-DSPG method is proposed. The learning rate of the RL model is tuned to improve the learning performance of the proposed method.
3) Related experiments are conducted to investigate the potential of the suggested scheme in a renowned robot platform.

This paper is organized as follows. Section 2 gives a brief introduction to the Multi-objective visual control model. Section 3 gives details of the proposed VBC-DSPG method with a Cosine Annealing. Section 4 demonstrates the experiment setup and experimental results. In the last section, the conclusions are presented.

2 Vision-Based Multi-objective Control

2.1 Multi-objective Visual Control Model

In a visual control process, a robot uses an RGB-D vision to extract features from the current image, which is $\mathbf{S}_c = (\mathbf{s}_1^c, \mathbf{s}_2^c, \ldots, \mathbf{s}_\mathrm{N}^c)^\mathrm{T}$. To use visual control methods with the closed-loop feedback mechanism, we denote the desired features as $\mathbf{S}_* = (\mathbf{s}_1^*, \mathbf{s}_2^*, \ldots, \mathbf{s}_\mathrm{N}^*)^\mathrm{T}$. Then, the errors $\mathbf{e}(t) = [\mathbf{s_1^c} - \mathbf{s_1^*}, \mathbf{s_2^c} - \mathbf{s_2^*}, \ldots, \mathbf{s_N^c} - \mathbf{s_N^*}]^\mathrm{T} \in \mathfrak{R}^{2N\times 1}$ between the current features and the desired features can be calculated. The feature errors $\mathbf{e}(t)$ is a time-varying vector and its change depends on the motion of the robot. The velocities for the robot are derived from the time derivative for the vector $\mathbf{e}(t)$ and it is as follows

$$\dot{\mathrm{e}}(t) = d\mathbf{e}(t)/dt = \left(d\left[\mathbf{s}_1^c - \mathbf{s}_1^*\right]/dt, d\left[\mathbf{s}_2^c - \mathbf{s}_2^*\right]/dt, \ldots, d\left[\mathbf{s}_N^c - \mathbf{s}_N^*\right]/dt\right). \quad (1)$$

Multi-objective visual control requires the movement of a robot to simultaneously satisfy three different objectives, which are less fluctuation, short time, and no loss of target. The Jacobian matrix J transforms the change rate for feature errors into velocities of the robot. The conversion process is as follows

$$\dot{\mathbf{e}}(t) = \mathbf{J}\left[\omega_t, v_t^x, v_t^y\right]^\mathrm{T} = \mathbf{J}[\boldsymbol{\omega}, \mathbf{v}]^\mathrm{T}, \mathbf{J} \in \mathfrak{R}^{2\,N*3}, \quad (2)$$

where N is the number of the features. The previous study [4] used $\dot{\mathbf{e}}(t) = -\varphi\mathbf{e}(t)$ to ensure an exponential decoupled decrease for the error vector $\mathbf{e}(t)$. Equation (3) gives the visual control law using the universal Jacobian matrix.

$$\begin{array}{l} [\omega_t^r, v_t^x, v_t^y]^{\mathrm{T}} = -\varphi\hat{\mathbf{J}}^{+}\mathbf{e}(t), \hat{\mathbf{J}}^{+} \in \Re^{3*2\,N} \\ \mathbf{J} = x\mathbf{J}_1 + (1-x)\mathbf{J}_2 \end{array}, \tag{3}$$

where $\hat{\mathbf{J}}^{+}$ is the pseudo-inverse for the Jacobian matrix.$\varphi$is a constant, which is the servo gain [4]. Based on the Jacobian matrix, a multi-objective visual control model is given by,

$$\begin{array}{l} \text{minimize } F(x) = [f_1(x), f_2(x), \ldots, f_m(x)], x \in [0,1] \\ \text{subject to } \begin{cases} \varphi(\mathbf{e}(t), \mathbf{J}(x)) = [\boldsymbol{\omega}, \mathbf{v}]^{\mathrm{T}} \\ \mathbf{J}(x) = x\mathbf{J}_1 + (1-x)\mathbf{J}_2 \end{cases} \end{array}, \tag{4}$$

where x is the control parameter and its value ranges from 0 to 1. $f_1(x)$, $f_2(x), \ldots,$ $f_m(x)$ represents the objectives for a perfect control performance, which are defined in a specific task. $\varphi(\bullet)$ is a conversion function mapping the feature errors to velocities of the robot. In this work, these objectives are less fluctuation, short time, and the low probability of target loss. An appropriate value for the control parameter x can satisfy the high-efficiency multi-objective control.

2.2 Computation of the Jacobian Matrix

Figure 1 describes a coordinate transformation model for visual features. In the transformation model, the coordinate for the origin $O_1\,(u_o, v_o)$ on the image plane is mapped to the coordinate for the origin O_2 on the camera plane.

If the coordinate in pixel for the point $\mathbf{a}$ on the camera plane is $\mathbf{a}(u, v)$, the 3D point concerning the point $\mathbf{a}$ on the Cartesian coordinate system is $\mathbf{A}$. Take the point $\mathbf{a}$ and the point $\mathbf{A}$, for example, Eq. (5) gives the transformation from a point on the camera plane to a 3D point in the Cartesian coordinate system. The value of the pixel for the point $\mathbf{a}$ is $\mathbf{d}\,(u', v')$ on the image plane.

$$\begin{cases} X_a = Z_a d_x\,(u - u_o)\,/f^2 \\ Y_a = Z_a d_y\,(v - v_o)\,/f^2 \end{cases}, \tag{5}$$

where the focal length of the RGB-D camera is f. The scaling constants from the image plane to the camera plane in the x-axis and y-axis directions are represented by ϑ_x and ϑ_y. For the $j-th$ feature point (u_j, v_j), the expression for the Jacobian matrix is given by Eq. (6).

$$\mathbf{J}_j = \begin{bmatrix} -\left(f/Z_a\vartheta_x\right) & 0 & v_j\left(\vartheta_y/\vartheta_x\right) \\ 0 & -\left(f/Z_a\vartheta_x\right) & u_j\left(\vartheta_x/\vartheta_y\right) \end{bmatrix}, \tag{6}$$

For the N features, the whole expression for the Jacobian matrix is $\mathbf{J} = [\mathbf{J}_1, \mathbf{J}_2, \ldots, \mathbf{J}_N]^{\mathrm{T}} \in \Re^{2N*3}$.

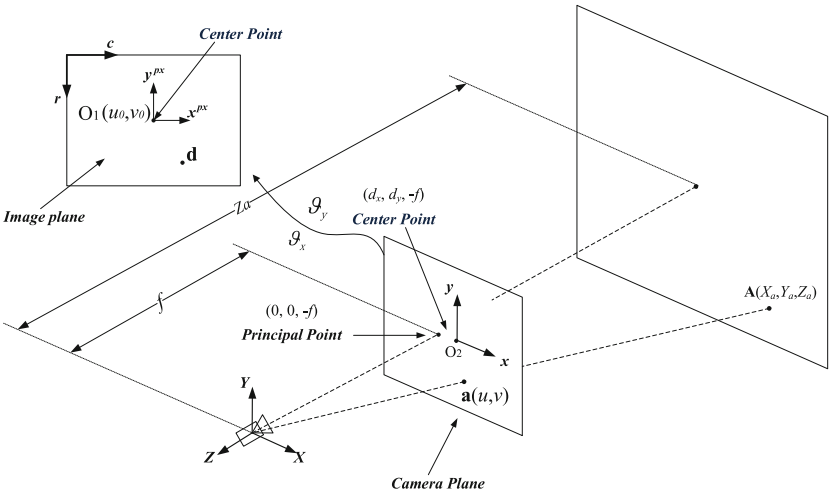


Fig. 1. Coordinate transformation model for visual features.

3 A Discounted Sampling Policy Gradient with a Heuristic Strategy for Multi-objective Visual Control

3.1 Reinforcement Learning

An RL agent selects an action a_t using a policy $\pi(s, a)$ when receives state s_t. Then, the agent gets a reward r_t from the environment and moves to the next state s_{t+1}. This process is repeated until the RL agent reaches the target position. The action policy π is constantly updated with the exploration of the agent. The RL method maximizes the expectation of the cumulative rewards $R(s_t) = r_t + \gamma R(s_{t+1}) = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ for each state. γ is the discount factor. The value function is used to represent the expectation for the cumulative reward. The value function for the state s_t is given by,

$$\begin{aligned} V(s_t) &= \mathrm{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s = s_t\right] \\ &= \sum_a \pi(s,a) \sum_{s_{t+1}} P^a_{s_{t+1}} \left(r_t + \gamma V(s_{t+1})\right) \end{aligned}. \tag{7}$$

The Q-learning method uses the action-value function to represent the expectation for the cumulative reward. The action-value function for Q-learning in one-step prediction to the RL problems can be represented by,

$$Q(s,a) = \mathrm{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right]. \tag{8}$$

For the RL problems, the policy gradient optimizes the policy with the gradient in the policy space, instead of the value function. This method is practical when encountering a task with a stochastic action space [15]. DSPG algorithms

are often used to address the problem of action selection in a continuous or high-dimensional space for RL tasks with unstructured environments. The agent is driven to move to the next state and receive a reward from the environment. The discounted reward sampling in a fixed interval allows us to estimate the value function, resulting in a policy that is considered to be optimal or closer to optimal.

3.2 Discounted Sampling Policy Gradient with a Cosine Annealing

Similar to the Deep Q-network (DQN) method, a nonlinear neural network function approximator is used for the DSPG method. The DSPG method uses low-dimensional observations, such as the joint angles or pixels, to learn competitive policies for many cases.

The goal of the DSPG method is to maximize the long-term discounted return with the weights θ, which is $J(\theta) = \mathrm{E}_{\tau \sim p_\theta(\tau)} \left[\sum_t r\left(s_t, a_t\right)\right]$. The optimal parameter space setting θ^* makes the robot get an optimal behavior trajectory τ, which can obtain the maximum long-term return $r(\tau) = \max \sum_t r\left(s_t, a_t\right)$. The better the trajectory means the robot can make a wise decision to select a time-varying matrix $\mathbf{J}$. s_0 is the initial state. The probability of a trajectory with a differentiable distribution function $p_\theta(\tau)$ is shown in Eq. (9).

$$p_\theta(\tau) = p\left(s_0\right) \prod_{t=0}^{T-1} p\left(s_{t+1} \mid s_t, a_t\right) \pi_\theta\left(a_t \mid s_t\right), \tag{9}$$

where $\pi_\theta\left(a_t \mid s_t\right)$ is the parameterized policy and $p\left(s_{t+1} \mid s_t, a_t\right)$ is the state transition probability. The gradient for the objective function is given by,

$$\begin{aligned} &\nabla_\theta J(\theta) = \int p_\theta(\tau) r(\tau) d\tau \\ &= \mathrm{E}_{\tau \sim p_\theta(\tau)} \left[\left(\sum_t \nabla_\theta \log\left(\pi_\theta\left(a_t \mid s_t\right)\right)\right)\left(\sum_t r\left(s_t, a_t\right)\right)\right] \\ &\approx \frac{1}{N} \sum_{j=1}^{N} \sum_t \left[\nabla_\theta \log\left(\pi_\theta\left(a_{t,j} \mid s_{t,j}\right)\right)\left(\sum_{t'=t} r\left(s_{t',j}, a_{t',j}\right)\right)\right], \end{aligned} \tag{10}$$

where the objective function samples N trajectories. The sampled returns are used to approximate the value function. In an episode, the effect of current reward on current policy decreases with the increase of time step. So, a discounted return sampling method for the expectation is proposed. The objective function with a discounted return sampling is given by,

$$\begin{aligned} &\nabla_\theta J(\theta) = \int p_\theta(\tau) r(\tau) d\tau \\ &\approx \frac{1}{N} \sum_{j=1}^{N} \sum_t \left[\nabla_\theta \log\left(\pi_\theta\left(a_{t,j} \mid s_{t,j}\right)\right)\left(\sum_{t'=t} \gamma^{t-t'} r\left(s_{t',j}, a_{t',j}\right)\right)\right], \end{aligned} \tag{11}$$

where γ is the discount factor. η_t is the learning rate. The updating law for the policy network is given by,

$$\theta_{t+1} \leftarrow \theta_t + \eta_t \nabla_{\theta_t} J\left(\theta_t\right). \tag{12}$$

Hyperparameter strategies evaluate the performance of each configuration on the learning tasks. This method can complete the training process of automatically parameter testing for a machine learning model. However, the time

cost of the evaluation for a configuration is expensive. In the process of Hyperparameter tuning, re-evaluation is necessary, which will be very inefficient for complex models. Some heuristic methods, such as evolutionary algorithms, are difficult to apply in the tuning of Hyperparameter. Previous works often leverage the Simulated Annealing (SA) to optimize the Hyperparameters [16]. As a heuristic method, SA simulates the annealing process in the thermodynamics energetics. A rule of the agent learning process should be satisfied: more current experience should be reserved in the early stage of learning. Conversely, the more prior experience should be preserved. A Cosine Annealing scheme is employed to tune the learning rate for the RL controller. The updating law with the Cosine Annealing is given by,

$$\eta_t \leftarrow \eta_{\min} + \frac{1}{2}\left(1 + \cos\left(\frac{T_t}{T_{\max}}\pi\right)\right)(\eta_{\max} - \eta_{\min}), \tag{13}$$

where $\eta_{\max}$ and $\eta_{\min}$ are the maximum and minimum of learning rate respectively, which are constant. T_t represents the number of current epochs. $T_{\max}$ is the maximum epochs. An exploration noise N sampled from a noise process is added to the policy network to develop an exploration action policy. The action policy with an exploration noise is given by,

$$a_t = \pi_\theta(s_t) + \mathrm{N}. \tag{14}$$

3.3 An Adaptive Multi-objective Visual Control Model with DSPG (VBC-DSPG)

DSPG method employs the policy network to select a time-varying matrix and uses a sample discounted return to estimate the value of the action function. An RGB-D camera receives image features by the contour recognition methods. Then, the current observation s_t is calculated. The observation information consists of a 4-dimensional vector defined by Eq. (15).

$$\mathbf{S}_t = (\mathbf{e}_x, d\mathbf{e}_x/dt, \mathbf{e}_y, d\mathbf{e}_y/dt), \tag{15}$$

where $\mathbf{e}_x, \mathbf{e}_y$ represents the vector for the feature errors of the current positions concerning the desired positions defined in the image plane respectively. The current policy network receives the vector of the current observation as the input and outputs a distribution $\pi_\theta(a_t \mid s_t)$ for the action space. This work defines a mapping function to ensure that the output action is limited between 0 and 1 because the control parameter x ranges from 0 to 1. Equation (16) gives the expression for the mapping function.

$$x = f(a_t) = \frac{1}{\pi}\arctan(a_t) + 0.5. \tag{16}$$

As illustrated in Fig. 2, the DSPG agent consists of a feed-forward neural network with an input layer, an output layer, and hidden layers of 20 and 40

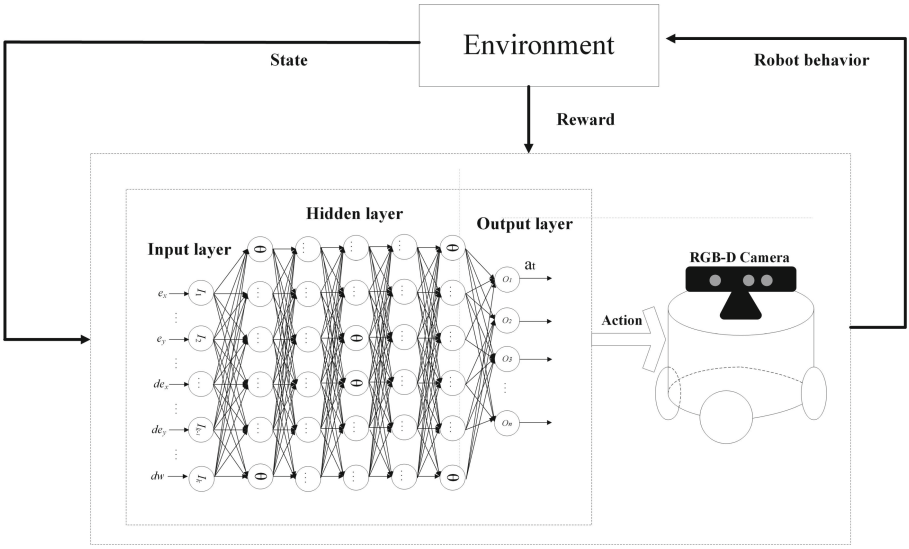


Fig. 2. Structure of the proposed DSPG model.

units. In hidden layers, the Sigmoid function is utilized as the activation function mapping an input to an output.

In the VBC-DSPG method, the shaping reinforcement signal r_t in Eq. (17) is used to update the policy.

$$r_t = -\frac{R}{2}\left(\sum_{i=1}^{N}\sqrt{\left(e_x^i\right)^2+\left(e_y^i\right)^2}+\sqrt{\left(de_x^i/dt\right)^2+\left(de_y^i/dt\right)^2}\right)/\left(N\sqrt{row^2+col^2}\right), \quad (17)$$

where the weight and length of the image plane are represented by row and col respectively. The environment gives the reward R to the RL agent depending on the actual situation. The number of feature points is N. The robot receives a higher reward if it is closer to the target.

The main motivation for employing the reinforcement signal is to drive the robot to achieve the target as soon as possible. In Eq. (17), the reward term penalizes the learning agent when the feature errors are big. The weights of the policy network are updated in the way that the robot adaptively selects a time-varying matrix for Multi-objective visual control.

4 Experiments

4.1 Experimental Environment

As shown in Fig. 3, we construct a simulation model with the same dynamics as the real device using commercial robotic software, Webots7.0.3. In the simulation, the robot learns policy using the DSPG algorithm and when the

policy converges, this policy is optimized using the Cosine Annealing. Noise and disturbance are added in the simulation environment to create a complex task environment close to the real-world. Then, the experiments on a mobile robot are performed to appraise the practicality of the proposed method. Finally, the proposed method and the conventional pseudo inverse kinematics methods are used to select the matrix. Experimental results are recorded to test whether the proposed method has better performance.

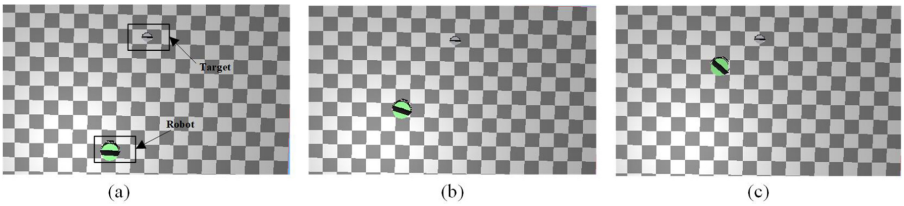


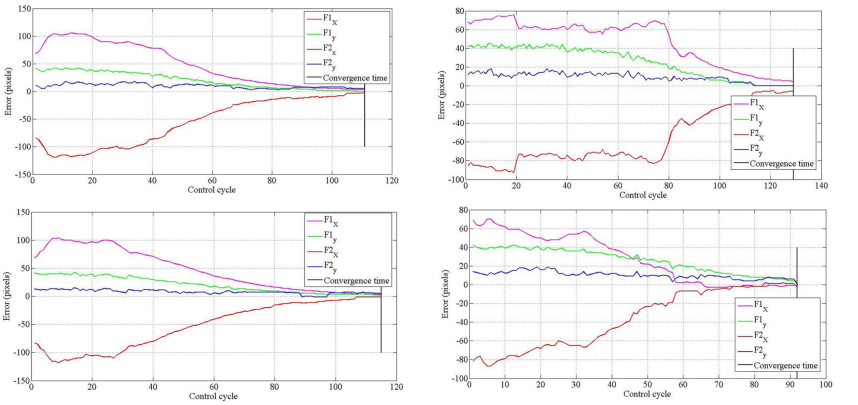
Fig. 3. A visual control task in the simulation environment.

4.2 Training for RL

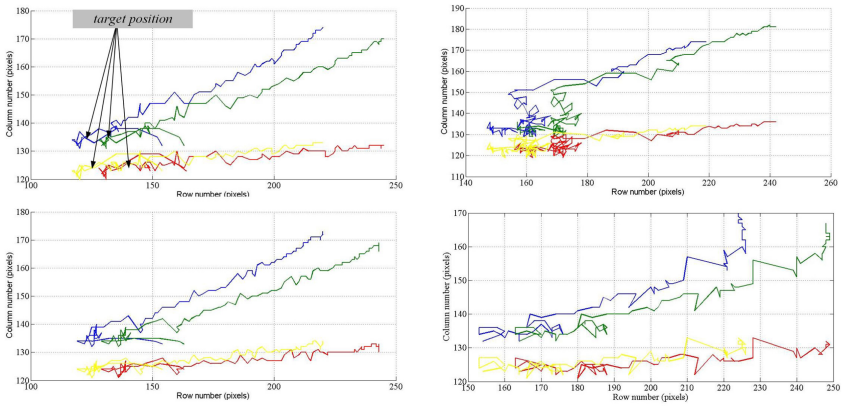
An episodic RL setting is used to train the RL agent. This control task requires the mobile robot to move from an initial position to a target position, as shown in Fig. 3. The visual control process of a robot should satisfy multiple objectives: less fluctuation, short time, and no loss of target. In each episode, the initial position for the RL agent is randomly selected, and the desired position is (0.463 m, 0.593 m, 37.8°). In each time step, the DSPG agent acts with an added exploration noise. The gain φ° is set to 0.35. The hyperparameters for the DSPG agent are shown below. The initial value for the learning rate η is 0.5. The discount γ is 0.95. The reward R is 10. The maximum epochs $T_{\max}$ is 1000.

The rules for the RL agent are as follows. 1). This episode will be terminated if the robot loses some features. 2). This episode will be terminated if the feature errors remain a certain pixel for a certain time. 3). This episode will be terminated if the robot does not arrive at the desired position after the maximum number of steps. 4). After a new action is selected by the RL agent, the matrix is updated using Eq.(3). 5). The robot returns to the starting position if a new episode starts.

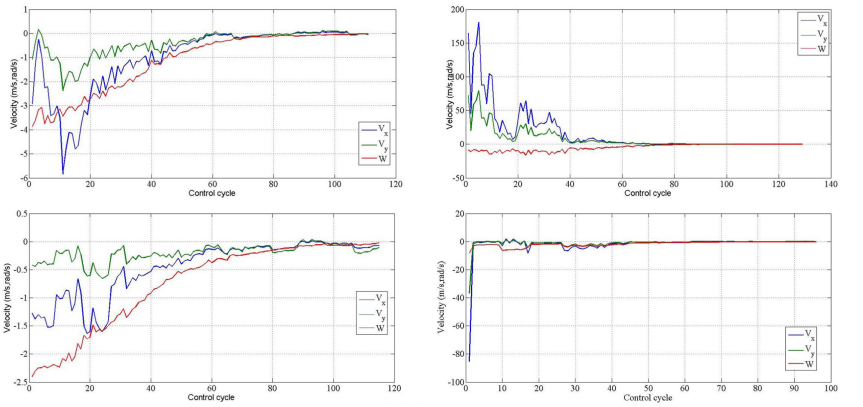
When this episode is terminated negatively, the reward -10 is given to the RL agent. In other episodes, the RL agent is rewarded with the reward function. With this configuration, the agent learns a stable motion behavior after considerable training episodes. Since the RL agent is given a negative reward before reaching the desired position, the reward is a negative value per episode. After training, agents will learn a policy from state space to behavior space.



(a) Feature error. A-JM,C-JM,D-JM, Proposed method



(b) Feature trajectories. A-JM,C-JM,D-JM, Proposed method



(c) Velocities. A-JM,C-JM,D-JM, Proposed method

Fig. 4. Experimental results, in terms of the Feature error, Feature trajectories, and Velocities. For each item, from left to right, from top to bottom are A-JM, C-JM, D-JM, and Proposed method.

4.3 Simulation

In the simulation, to verify the effectiveness of the proposed method, the proposed VBC-DSPG method and three competitors were run on the platform. The initial position is (0.0289 m, 0.062 m, 0.07°) and the desired position is (0.463 m, 0.593 m, 37.8°). Three competitors include the Jacobian matrix method using the current one (C-JM) [1], the matrix using the desired one (D-JM) [1], and the matrix using the average one (A-JM) [4]. The four methods were tested 50 times and the average values for these 50 times are shown in Fig. 4.

Experimental results show that the C-JM method converges in around 129 control cycles. The D-JM method converges in around 115 control cycles. The A-JM method converges in around 111 control cycles. The convergence time for the proposed method is around 92 control cycles. Although the three conventional competitors reach the desired position, the convergence rates for the three competitors are slower than the proposed method. Different methods select different Jacobian matrix, which results in different behavior. In Fig. 4, at the beginning of the control process, the proposed method chooses a larger velocity, which drives the robot to reach the target position faster.

However, in the beginning, excessive-velocity may lead to instability, difficulty in control, and even loss of target, resulting in danger, such as C-JM methods. Figure 4 shows the feature trajectories for the four methods. Conventional methods cause one dimension to reach the desired position, while others do not. So, there are many fluctuations in the feature trajectory. The proposed method enables the robot to learn a policy to achieve better multi-objective control performance.

5 Conclusion

This paper developed a new multi-objective visual control method that integrates discounted sampling policy gradient and a heuristic strategy. A DSPG based scheme is proposed to handle the multi-objective visual control challenges of mobile robots by selecting an appropriate Jacobian matrix adaptively. The policy network is obtained by a training process. Then, the network can automatically calculate a time-varying matrix to achieve a robust control performance. The proposed method has been extensively compared with three conventional methods.

References

1. Santamarianavarro, A., Andradecetto, J. : Uncalibrated image-based visual servoing. In: 2013 IEEE International Conference on Robotics and Automation, pp. 5247–5252 (2016)
2. Fraichard, Thierry, Levesy, Valentin: From crowd simulation to robot navigation in crowds. IEEE Robot. Autom. Lett. **5**(2), 729–735 (2020)
3. Chaumette, F., Hutchinson, S.: Visual servo control. I. Basic approaches. IEEE Robot. Autom. Mag. **13**(4), 82–90 (2006)

4. Wang, B., et al.: Parallel structure of six wheel-legged robot model predictive tracking control based on dynamic model. In: 2019 Chinese Automation Congress, pp. 5143–5148 (2019)
5. Sun, W., et al.: Multi-objective control for uncertain nonlinear active suspension systems. Mechatronics **24**(4), 318–327 (2014)
6. Malis, E: Improving vision-based control using efficient second-order minimization techniques. In: 2004 International Conference on Robotics and Automation, pp. 1843–1848 (2004)
7. Marey, M., Chaumette, F.: Analysis of classical and new visual servoing control laws. In: 2008 International Conference on Robotics and Automation, pp. 3244–3249 (2008)
8. Watanabe, K., et al.: Image-based visual PID control of a micro helicopter using a stationary camera. Adv. Robot. **22**(2–3), 381–393 (2008)
9. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: a survey. Int. J. Robot. Res. **32**(11), 1238–1274 (2013)
10. Zhang, J., et al.: VR-goggles for robots: real-to-sim domain adaptation for visual control. IEEE Robot. Autom. Lett. **4**, 1148–1155 (2019)
11. Xi, A., et al.: Balance control of a biped robot on a rotating platform based on efficient reinforcement learning. IEEE/CAA J. Automatica Sinica **6**(4), 938–951 (2019)
12. Agostinelli, F., et al.: Solving the Rubik's cube with deep reinforcement learning and search. Nat. Mach. Intell. **1**(8), 356–363 (2019)
13. Sampedro, C., et al.: Image-based visual servoing controller for multirotor aerial robots using deep reinforcement learning. In: 2018 International Conference on Intelligent Robots and Systems, pp. 979–986 (2018)
14. Sehgal, A., et al.: Deep reinforcement learning using genetic algorithm for parameter optimization. In: 2019 Third IEEE International Conference on Robotic Computing, pp. 596–601 (2019)
15. Wang, Lixin., Wang, Maolin, Yue, Ting: A fuzzy deterministic policy gradient algorithm for pursuit-evasion differential games. Neurocomputing **362**, 106–117 (2019)
16. Samma, H., et al.: Q-learning-based simulated annealing algorithm for constrained engineering design problems. Neural Comput. Appl. **32**(9), 1–15 (2019). https://doi.org/10.1007/s00521-019-04008-z