RESEARCH ON MODELING AND OBJECT TRACKING FOR ROBOT ARM BASED ON DEEP REINFORCEMENT LEARNING

ANH-DUNG NGUYEN, TIEN-DAT VU, QUANG-ANH VU, THAI-VIET DANG*

Hanoi University of Science and Technology, Hanoi, Vietnam

DOI: 10.17973/MMSJ.2025_06_2025059

e-mail to corresponding autho: viet.dangthai@hust.edu.vn

Robotic arms (RAs) are progressively advancing and extensively utilized across several industrial and research domains. In conjunction with that advancement, researchers have incorporated numerous intelligent algorithms to facilitate the RA's obstacle avoidance strategy. The paper proposes the Deep Reinforcement Learning (DRL) method to control the RA to maneuver the ball on the workspace to a specified location. Based on Resnet18-Unet model, 2D images will be analyzed to get the coordinate of the object's center. The input's data is utilized to train the Proximal Policy Optimization (PPO). Throughout this training procedure, the RA will discern suitable moves to maneuver the ball to the designated location. Based on the action's reward, RA's operation aims for maximum total reward results. The simulation results are completely conducted by PyBullet environment.

KEYWORDS

Deep reinforcement learning, robot arm, obstacle avoidance, object tracking, trajectory planning

1 INTRODUCTION

Robotic systems have been significantly integrated into industrial processes throughout the past decade. The inclination towards automatic systems and reconfigurable production lines has prompted control modules to implement methodologies. The vision-based robotic perception system has recently been utilized in various domains such as military, industrial and domestic sectors [Dang 2024]. Therefore, object detection (OD) enables robots to autonomously analyze object information from live images [Nguyen[b] 2024]. In self-driving applications and navigation, object tracking (OT) face complex challenges, as the entities involved in OT are inherently more intricate to manage a series of images [Maaß 2020].

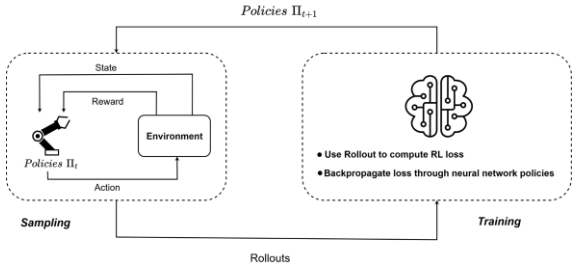
Trajectory planning is a vital issue in robotic systems, especially in the motion control of robotic manipulators. In addition, the most important issue involves determining a continuous, unobstructed path from the initial point to the destination. To do so, robotic systems consider potential obstacles, kinematic constraints, and the robot's dynamic modelling to efficiently track the required trajectory. Furthermore, it is crucial to prevent collisions or unintended movements. Currently, various investigations are underway regarding the control of RA's movement. Diverse methodologies, including PID, linear quadratic regulator (LQR), and fuzzy, may be employed for this objective [Zabi´nski 2006]. Classical PID-based feedback control is the primary choice. However, PID controller is improved to overcome derivative filtering [Deniz 2018] or lead-lag compensation [Pebrianti 2017] in process control. Because of nonlinear factors' influence, the multi-regional PID method is combined with fuzzy method [Nguyen 2025]. The evolution of DRL and neural network methods (NNMs) has been increasingly developed. Hence, reinforcement learning seamlessly integrates with deep learning techniques [Nguyen 2020]. The Q-learning algorithm [Peng 2022] marked a significant milestone to effectively solve control problems. In addressing uncertainties within environments, Chen et al. [Chen 2022] introduced the improved collision avoidance trajectory based on DRL. Moreover, [Zheng 2023] utilized control policies, parameterized by NNMs using Proximal Policy Optimization (PPO) algorithms. Additionally, [Cai 2023] introduced a novel DRL-based approach for RA's path planning. Nonetheless, despite the pro-gress, these methods continue to face significant challenges. The DRL reward functions solve the problem thoroughly instead of just suggesting it in a multidimensional coordinate system.

Increasing RA's trajectory planning efficiency in dynamic environments requires optimizing exploration within complex, high-dimensional spaces [Nguyena 2024]. RA is frequently employed with LiDAR sensors or IMU sensors to gather environmental data. But LiDAR may struggle to detect lowlying obstacles or experience signal interference in intricate environments, while IMU sensors may become distorted over time, adversely impacting the RA's computational performance. Consequently, the camera is an appropriate option for the task of gathering environmental data for vision-based RAs [Nguyen[b] 2024, Nguyen[a] 2024, Dang 2025]. The paper utilizes the DRL method to enable the RA to maneuver the ball on the workspace to a specified location. Images captured by the 2D camera will be analyzed to get the object's center. Based on the proposed PPO, every RA's action is executed by a reward, with the objective of maximizing the total reward. The simulation results are completely conducted in Pybullet environment.

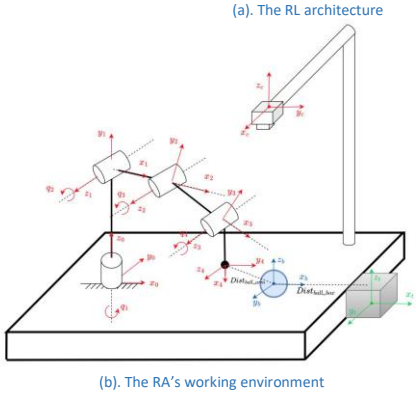
The subsequent sections of the paper are structured as follows. Section 2 focuses on how to process information from the environment, design components for the RL, and DRL models. Section 3 presents the experimental results. Finally, the paper concludes with a summary of the findings and suggestions for future research.

2 PROPOSED METHOD

2.1 Model architecture



(a). The RL architecture



(b). The RA's working environment

Figure 1. RA's controller

RL is a subset of machine learning wherein an agent acquires knowledge through interaction with the environment to optimize cumulative reward. RL operates on the principles of experimentation and environmental feedback, wherein the agent executes behaviours contingent upon the present state of the environment and receives feedback in the form of incentives. The research examines the RA to manipulate joint rotation angles to accurately strike a ball towards a target, in Fig. 1a. Next, Fig. 1b illustrates the RA's working environment including: a ball with a radius of R, a bucket as the destination, a top-down camera, in which the robot's base is stationary, the camera and the bucket are stationary. Finally, Proximal Policy Optimization (PPO) is a policy optimization technique derived from the Policy Gradient methodology among reinforcement learning methods. PPO enhances prior methodologies by constraining the policy update rate to a permissible range, hence preventing excessive alterations between updates.

2.2 RL State

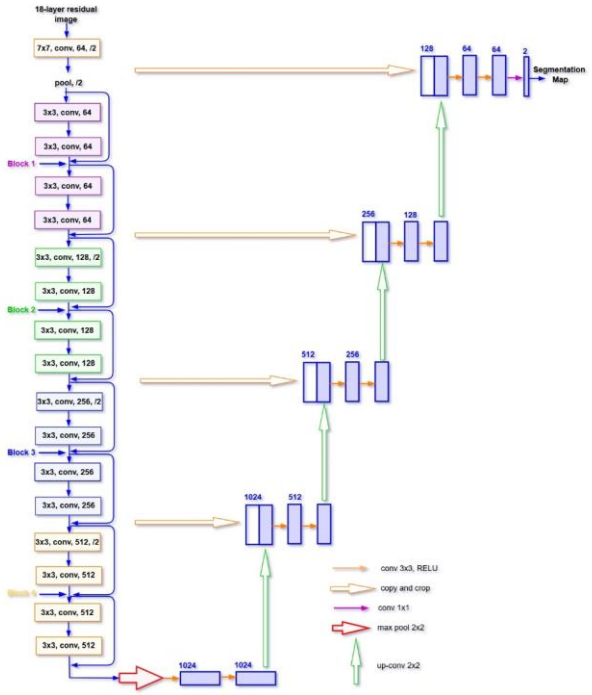


Figure 2. Resnet18-Unet

The state (1) includes information about the center position of the ball $s_{ball}=\{x_b,y_b,z_b\}$, the end position of the RA $s_{arm\_end}=\{x_4,y_4,z_4\}$, the center position of the box $s_{box}=\{x_t,y_t,z_t\}$, to calculate the distance difference $Dist_{ball\_arm}$, and $Dist_{ball\_box}$ to RL control the ball.

$$state=\{s_{ball},s_{arm\_end},s_{box}\} \quad (1)$$

The system uses a 2D camera arranged in Fig. 1b to observe the entire working area and all objects in it. The location determination process is performed through converting images from RGB to segmentation images by Resnet18-Unet (Fig. 2), then the shape recognition method through the geometric characteristics of each object to ensure high accuracy and stable operation in different environmental conditions. Moreover, the input image is processed to remove noise and highlight the geometric features of the object. The Hough Circle Transform method is applied to detect circles in the image after preprocessing. Therefore, the center of the ball is determined by (2):

$$(x-x_b)^2+(y-y_b)^2=r^2 \quad (2)$$

where: (x_b,y_b) is the coordinate of the center of the circle, *r* is the radius of the circle.

The Contour Detection method is used to detect areas that could be boxed. Similarly, the center of the box is determined by the midpoint of the bounding box, in (3):

$$x_t=x+\frac{w}{2},y_t=y+\frac{h}{2} \quad (3)$$

where: *x,* and *y* are the coordinate of the origin of the camera; (x_t,y_t) is the coordinate of the center of the box; *w* and *h* are the width and height of the bounding box, respectively.

Because the ball only moves in a plane and the box is fixed, so $z_b=const,z_t=const$. From there, get the coordinates of ball $s_{ball}=\{x_b,y_b,z_b\}$ and box $s_{box}=\{x_t,y_t,z_t\}$ according to the original RA. $s_{arm\_end}=\{x_4,y_4,z_4\}$ will be continuously updated by the forward and inverse kinematics as the RA moves.

2.3 RL Action

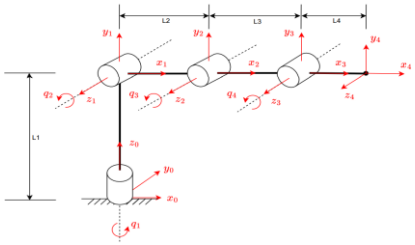


Figure 3. The kinematics of the robot arm

$$Move=\{\Delta x=random(-n,n);\Delta y=random(-n,n);\Delta z=random(-n,n)\} \quad (4)$$

The action here is set according to the moving point of the last link, with each step, the last link will move so that each *x*, *y*, *z* axis is not more than *(-n, n)* units, then use inverse kinematics to calculate the displacement of each joint, continuously update so that the RA tracks the trajectory. The RA consists of four links and four rotating joints, the coordinate system (Fig. 3), the D-H table is calculated in Table 1.

i	*a*	*α (°)*	*d*	*θ*
1	0	90	L_1	q_1
2	L_2	0	0	q_2
3	L_3	0	0	q_3
4	L_4	0	0	q_4

Table 1. The D-H table

To calculate the RA's inverse kinematics, it is necessary to determine a transformation matrix from link *i-1* to *i*, in (5).

$$^{i-1}_{i}T=\begin{bmatrix} c\theta_i & -s\theta_i c\alpha & s\theta_i s\alpha & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha & c\alpha & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

where $c\theta_i=\cos(\theta_i),s\theta_i=\sin(\theta_i),s\theta_{ij}=\sin(\theta_i+\theta_j),c\theta_{ij}=\cos(\theta_i+\theta_j)$ $,s\theta_{ijk}=\sin(\theta_i+\theta_j+\theta_k),$ and $c\theta_{ijk}=\cos(\theta_i+\theta_j+\theta_k)$

From the transformation matrices (5), the general transformation matrix is calculated from link 0 to link 4 in (6):

$$^{0}_{4}T=^{0}_{1}T\times^{1}_{2}T\times^{2}_{3}T\times^{3}_{4}T=\begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Hence, corresponding coordinates P_x, P_y, P_z, let φ = 0 so that the last link is according to top-down in (7):

$$\begin{aligned} P_x &= c_1\left(L_1 + L_3c_{23} + L_2c_2 + L_4c_{234}\right) \\ P_y &= s_1\left(L_1 + L_3c_{23} + L_2c_2 + L_4c_{234}\right) \\ P_z &= d_1 + L_3s_{23} + L_2s_2 + L_4s_{234} \\ \theta_2 &+ \theta_3 + \theta_4 = \varphi \end{aligned} \tag{7}$$

By solving the RA's inverse kinematics problem, the four angles $\theta_1, \theta_2, \theta_3,$ and θ_4 are determined as (8):

$$\begin{aligned} \theta_1 &= a\tan 2\left(P_y, P_x\right) \\ S_1 &= P_xc_1 + P_ys_1 - L_1 - L_4c_{234} \\ S_2 &= P_z - d_1 - L_4s_{234} - 2S_1L_2c_2 - 2S_2L_2s_2 \\ &= L_3^2 - L_2^2 - S_1^2 - S_2^2 \rightarrow a\sin\left(\theta_2\right) + b\cos\left(\theta_2\right) = c \rightarrow \theta_2 \\ \theta_2 &+ \theta_3 = a\tan 2\left(s_{23}, c_{23}\right) \rightarrow \theta_3 \\ \theta_4 &= \varphi - \theta_2 - \theta_3 \end{aligned} \tag{8}$$

2.4 RL Reward

The Reward section consists of two parts (Fig. 4):

- First phase: When the RA initiates movement without contacting the ball, the reward function will assess the reward according to the RA's motion; the greater the distance of the last step, the more points will be subtracted.

- Second phase: Upon contact with the ball and to the initial center position of the ball, the RA will cease movement, and the reward will be computed depending on the ball's deviation; greater deviation will result in a higher deduction of points.

```
def com_reward:
reward=0
if collision_1:
    reward += 3
    reward += dist_ball_box*13
else:
    reward += - dist_ball_arm*18
    reward += -8
if distance_ball_goal <= epsilon:
    reward += 130
terminal=True
```

Figure 4. RL reward in programming

The goal is for the RA to understand how the final step affects the direction of the ball, hence adjusts its trajectory to match the destination. In Fig. 5, the Proximal Policy Optimization Algorithms (PPO) model [11] is utilized to train the RA action including two neural networks: one neural network of policy, and one value neural network. Policy has the input as the state; the output as the action_dim vector; representing the average of Gaussian distribution.

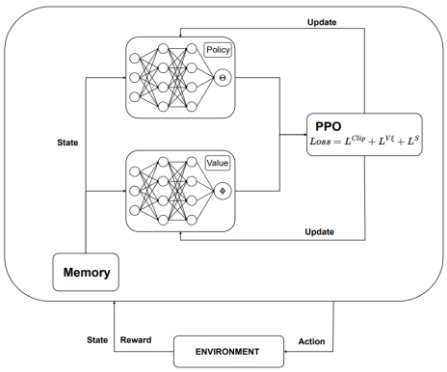


Figure 5. Training Process with PPO and NN Methods

The value has the input as the state, the output as the advantage value of state. Therefore, those parameters are passed into PPO to calculate the loss. Next, to adjust the two neural networks (9) and (10):

$$L_t^{CLIP+VF+S}(\theta) = \mathrm{E}_t\left[L_t^{CLIP}(\theta) - c_1L_t^{VF}(\theta) + L_t^S\right] \tag{9}$$

where L^{CLIP} plays the role of adjusting the update rate of the old and new policies within the limitation. Hence, the Policy is not much fluctuated to stabilize the training process.

$$L_t^{CLIP}(\theta) = E[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t] \tag{10}$$

Then, the main coefficients in (10) are defined as follows:

- $r_t(\theta) = \dfrac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}$ is the ratio between the action probability of the new and old policies.
- ε is the PPO's clipping parameter, the value of adjusting the policy limit.

Finally, the Critic loss function reduces the error between the estimated value and the actual value to estimate the value of the environment, in (11):

$$L^{VF} = \left(V(s_t) - R_t\right)^2 \tag{11}$$

where $V(s_t)$) is the output value of the neural network value; R_t is the value of the common part calculated from the environment.

Because L^{CLIP} support PPO to increasingly converge the optimal trajectory. Hence, the RA will lose the ability to explore more trajectories. In addition, the entropy function increases the exploration ability in (12):

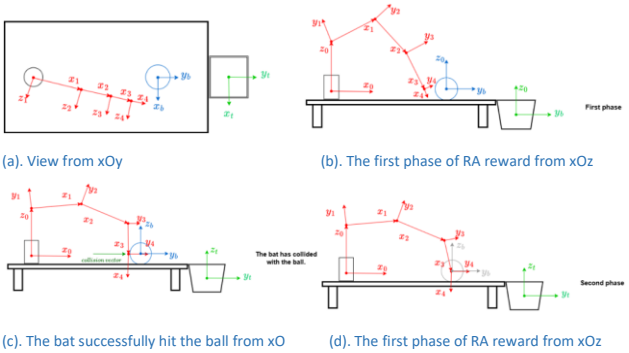
$$L_t^S = c_2S[\pi_\theta](s_t) \tag{12}$$

- Distribution entropy $S[\pi_\theta]$: the entropy of the policy.
- *c:* a coefficient adjusts the influence of entropy.

3 EXPERIMENTAL RESULTS AND DISCUSSION

3.1 Simulation environment

The simulation process is performed on a computer device with the following pa-rameters: CPU: 13th Gen Intel i9-13900K 5.5Ghz, RAM: 64GB, GPU: GeForce RTX 4080 16GB GDDR6X [14]. The head is designed in the shape of a baseball bat so that when it contacts the ball, it will be a point contact, creating an additional element of complexity for the RA to find the correct contact position between the end link and the ball. In first phase, the RA will need to find the direction and point of contact with the ball so that the ball can move towards the box. That direction will be towards the center of the ball, without causing the ball to bounce or veer, in second phase (Fig. 6).



(a). View from xOy (b). The first phase of RA reward from xOz

(c). The bat successfully hit the ball from xO (d). The first phase of RA reward from xOz

Figure 6. Simulated results of RA operation

Based on Pybullet environment, the model trains 20000 episodes with each episode being 125 steps (Fig. 7). The original RA's position is [0.0,0.0,0.0], the end joint position is [0.42, -0.08, 0.22, the ball position is [0.8, 0.0, 0.0], the box position is [0.97, 0.0, -0.1].

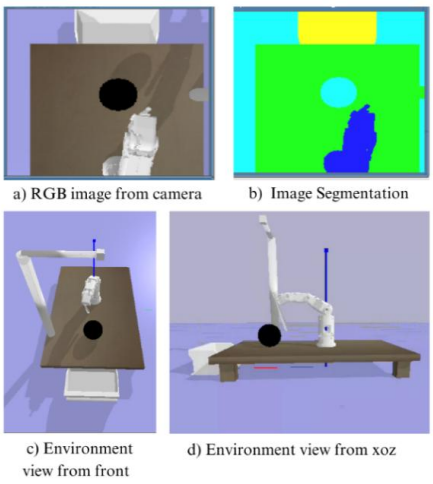


Figure 7. RA's simulation using Pybullet environment

3.2 Experimental Results

At the beginning of training, the results are quite unstable, although there is success, it is due to randomness. As time goes on, the obtained results become more stable because the robot has learned how to hit the ball optimally and which trajectory to choose to get the largest reward (Fig. 8a).

(a). Training reward

(b). Success rate over each 100 epoch

Figure 8. Reward through time

Then, Fig 8b shows the success rate of each 100 epochs corresponding to the reward. The training rate is higher because the policy has converged. Despite the increasing search rate, the next 100 epochs will not be fully optimized but can continue to explore other trajectories. Therefore, this proves that the proposed model operates successfully and explores completely the environment. Fig. 9 demonstrates the feasibility of the results, we tested in random initial values instead of just having one fixed point for training, each point is marked as the success rate over 300 epochs, randomly choosing the position in the range $\left[0.8, random(-y, y), 0.0\right]$. At y = 0.005 m, the training results are still feasible and stable above 90%, however when starting to increase the distance, the success rate has decreased, this proves that the model can still work with small displacement, with large displacement the model will become inaccurate. After the training process, the RA has a moving trajectory as shown in Figs. 10 and 11 at about position y = 0.005, the RA's trajectory is adjusted to match the direction of the ball to the barrel, not moving too strong or too light or flying off the table. Based on obtained RL results of proposed state, reward and action, the PPO's application is feasible in controlling the robot according to the trajectory.

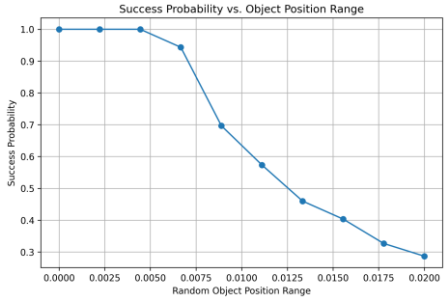
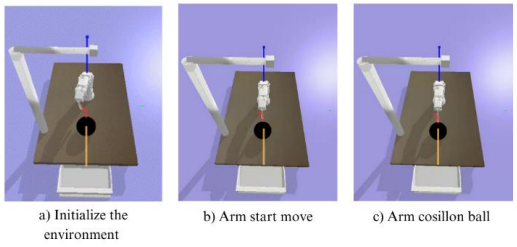


Figure 9. Success rate on random object position range



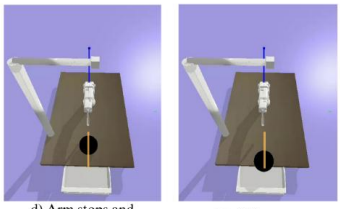


Figure 10. Robot arm is viewed from the front plane (Red is the robot endpoint trajectory, orange is the ball trajectory)

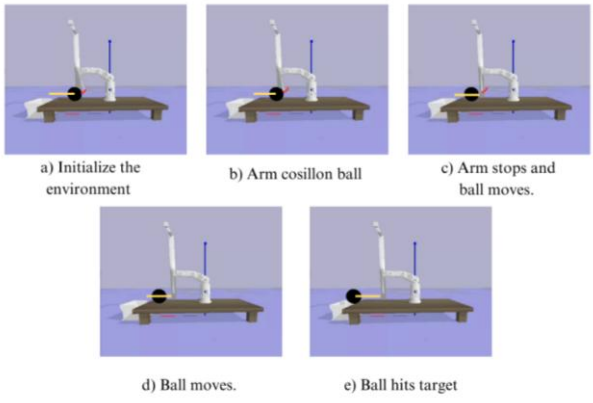


Figure 11. RA is viewed from the xOz plane (Red line is the robot endpoint trajectory, orange line is the ball trajectory)

4 CONCLUSIONS

The paper proposed DRL method to facilitate the RA's movement of the ball to a designated position within the workspace. Based on Resnet18-Unet, 2D images are applied Contour Detection and Hough Circle Transform to accurately determine the coordinates of the object's centroid. Furthermore, OT is used to monitor the ball's trajectory. Then, the data is employed to train the PPO algorithm to stabilize policy updates and enhance training efficiency compared to traditional RL methods. Based on proposed DRL method, training results being viable and stable at above 90% over 300 epochs with randomly choosing the position of the ball on the PyBullet environment. Finally, the proposed method consolidates the integration of DRL with vision-based perception systems and underscores its potential scalability for robot navigation in the real-world.

ACKNOWLEDGMENTS

This research is funded by Hanoi University of Science and Technology (HUST) under project number T2024-PC-033. Computer Vision and Autonomous Mobile Robot Laboratory (CVMR), 821M-C7, SME, HUST is gratefully acknowledged for providing a research location, guidance, and expertise.

REFERENCES

[Cai 2023] Cai, J., et al. Prediction-based path planning for safe and efficient human-robot collaboration in construction via deep reinforcement learning. Journal of computing in civil engineering. 2023, Vol.37, No.1, 04022046, ISSN: 0887-3801. https://doi.org/10.1061/(ASCE)CP.1943-5487.00010

[Chen 2022] Chen, P. et al. A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. Neurocomputing. 2022, Vol.497, pp 64-75, ISSN: 0925-2312. https://doi.org/10.1016/j.neucom.2022.05.006

[Dang 2024] Dang, T.V., et al. Hybrid Mobile Robot Path Planning Using Safe JBS-A*B Algorithm and Improved DWA Based on Monocular Camera. Journal of Intelligent & Robotic Systems. 2024, Vol.110, No.151, pp 1-21, ISSN: 0921-0296. https://doi.org/10.1007/s10846-024-02179-z

[Dang 2025] Dang, T.V., et al. KD-SegNet: Efficient Semantic Segmentation Network with Knowledge Distillation Based on Monocular Camera. Computers, Materials & Continua, 2025, Vol.82, No.2, pp 2001-2026, ISSN: 1546-2218. https://doi.org/10.32604/cmc.2025.060605

[Deniz 2018] Deniz, M., et al. Experimental Verification of Lead-Lag Compensators on a Twin Rotor System. Electrical, Control and Communication Engineering. 2018, Vol.14, No.2, pp 164-171, ISSN: 2255-9159. https://doi.org/10.2478/ecce-2018-0020

[Maaß 2020] Maaß, F.L.: An analysis on object detection and tracking with a tilt-pan camera on an embedded device. Hamburg University of Applied Sciences, Tech. https://doi.org/10.13140/RG.2.2.28430.23365

[Nguyen[a] 2024] Nguyen, C.H., et al. Optimal obstacle avoidance strategy using deep reinforcement learning based on stereo camera. MM Science Journal. 2024, Vol.10, pp 7556-7561, ISSN: 803-1269. https://doi.org/10.17973/MMSJ.2024_10_2024078

[Nguyen[b] 2024] Nguyen V.T., et al. A Comprehensive RGB-D Dataset for 6D Pose Estimation for Industrial Robots Pick and Place: Creation and Real-World Validation. Results in Engineering. 2024, Vol.24, pp 103459, ISSN: 2590-1230. https://doi.org/10.1016/j.rineng.2024.103459

[Nguyen 2025] Nguyen, V.T, et al. Optimal Two-Wheeled Self-Balancing Mobile Robot Strategy of Navigation using Adaptive Fuzzy controller-based KD-SegNet. Intelligent Service Robotics, ISSN: 1861-2776

[Nguyen 2020] Nguyen, T.T., at al. Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications. IEEE Transactions on Cybernetics. 2020, Vol.50, No.9, pp 3826-3839, ISSN: 2168-2267. https://doi.org/10.1109/TCYB.2020.2977374.

[Pebrianti 2017] Pebrianti, D. et al. Intelligent control for visual servoing system. Indonesian Journal of Electrical Engineering and Computer Science. 2017, Vol.6, No.1, pp 72-79, ISSN: 2502-4752. https://doi.org/10.11591/ijeecs.v6.i1.pp72-79

[Peng 2022] Peng, G., et al. Deep reinforcement learning with a stage incentive mechanism of dense reward for robotic trajectory planning. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2022, Vol.53, No.6, pp 3566-3573, ISSN: 0018-9472. https://doi.org/10.1109/TSMC.2022.3228901

[Zabi´nski 2006] Zabi´nski, T., et al. Design and implementation of visual feedback for active tracking. Machine Graphics and Vision. 2006, Vol.15, No.3-4, pp 681-690, ISSN: 1230-0535.

[Zheng 2023] Zheng, Q., et al. Robotic arm trajectory tracking method based on improved proximal policy optimization. In: Proceedings of the Romanian Academy, Series A: Mathematics, Physics, Technical Sciences, Information Science. 2023, Vol.24, No.3, pp 237-246, ISSN: 1454-9069. https://doi.org/10.59277/PRA-SER.A.24.3.05

CONTACTS:

Assoc. Prof. Dr. Thai-Viet Dang
Department of Mechatronics, School of Mechanical Engineering, Hanoi University off Science and Technology
01 Dai Co Viet Road, Hai Ba Trung, Hanoi, 10000, Viet nam
+84 989458581, viet.dangthai@hust.edu.vn