A Robust Autonomous UAV Control System Based on Kalman Filter, LSTM, MLP and DQN

Zhenxiao Jiang, *Tongji University*

August, 2024

Abstract

In the modern world, autonomous control systems have been applied in a wide range of fields, profoundly impacting everyone's life. They not only free humans from these tasks, but also provide a more efficient and reliable alternative beyond the limitations of humans. In recent years, the rapid development of artificial intelligence (AI) and machine learning (ML) has added wings to these systems. In this paper, we will focus on the robustness of autonomous control systems, which is important in real life but is often overlooked. We propose a framework combining pre-determined algorithms and several AI based neural networks including LSTM, MLP and DQN. Then we pick one of the circumstance, a post-disaster UAV rescue mission featuring measurement interference and potential breakdowns, and simulate it with a computer program. A specific system is built and trained based on our framework and evaluated in the environment. The results show that the system exhibits significantly superior robustness and control performance compared with other established models. Looking ahead, the potential scalability and applicability of this framework can be further explored and can serve as a reference in various fields.

Our code can be downloaded at
https://github.com/ZhenXiao-Jiang/robust_control_system.git/.

Index Terms: Robust Control System, Machine Learning, Kalman Filter, LSTM, MLP, DQN, DDQN.

Contents

Section 1. Introduction

1.1 Topic Overview

In the modern world, autonomous control systems have been applied in a wide range of fields, including but not limited to manufacturing [1], [2], [3], logistics and transport [4], energy management [5] and operations conducted by unmanned vehicles like UAVs (Unmanned Aerial Vehicles) [6]. They are also applied in other fields like agriculture [7] and disaster response [8].

These systems can perceive relevant environmental variables in real time, analyse them and make real time decisions without human help. They not only free humans from these tasks, but also provide a more efficient and reliable alternative beyond the limitations of humans. Therefore, they are regarded as one of the cornerstones of modern technological advancement.

Traditional automated control systems are essentially pre-determined algorithms designed on mathematics models, physics principles and engineering methods [9], [10]. They can achieve precise control in real time in the specified condition.

In recent years, the rapid development of artificial intelligence (AI) and machine learning (ML) has added wings to these systems [11]. With the framework of AI, ML provides [12], [13] powerful data processing and decision support capabilities to help the system perform more effectively in complex environments [11], [14], [15], [16], [17], [18]. In other words, they become less reliant on the precise model constructed by engineers and thus be able to handle more complex conditions. In fact, AI is gradually becoming dominant in driving control [19], [20], [21], path tracking control [22], [23], path planning [20], [24], and multi-intelligence collaboration [25].

However, problems arise with such achievements, and our paper mainly focus on one of them, the robustness of the system. Most of the established systems do quite well in optimizing the outcomes in the ideal environment, but in real life, there always exists measurement errors and potential breakdowns. Under such assumptions, we should no longer expect the system to always make exactly the optimal decision. Instead, we hope them to avoid unacceptable outcomes, which a human operator would do. In other words, we hope the system to be robust.

For example, in a modern post-disaster rescue operation, a UAV with a autonomous control system is sent to the core area to take the trapped personnel out safely. It can get information from GPS in real time and its task is simply to avoid obstacles such as tall buildings and intricate power lines. Unfortunately, it can't get its precise position because of real life conditions like the damage to basic communication infrastructure caused by the disaster or extreme weather. To make things worse, secondary disasters are potential and may lead to complete breakdown of its sensors. In fact, the real world contains too many circumstances to be fully enumerated. However, in this mission, any mistake of the system costs lives, which is totally unacceptable.

To solve this problem, our research aims to improve the robustness of the autonomous control system by proposing a new framework. The proposed framework is a combination of several AI based neural networks and pre-determined algorithms, which are interconnected with each other in an effective way. Although the specific code in this paper is a specific system for the specific mission mentioned above, the framework can hopefully be used in other conditions with little adjustment.

1.2 Literature Review

In recent years, many works have been done focusing on the robustness of different autonomous control systems.

For traditional automatic control systems, the robustness is mainly enhanced by increasing the complexity of the system with new specific mechanisms.

In [26], a perturbation observer is proposed to achieve perturbation decay in a specified time. An auxiliary dynamic system with time-varying gain is constructed to handle the input saturation phenomenon to form a robust tracking control scheme for AUVs. In order to improve the robustness of the path tracking control system of an AUV, [27] proposed a robust model predictive control system designed by solving a cluster point model using LMI. [28] established a networked path-tracking control method including feedback linearisation, event-triggered controller, and safe sliding controller to the path-tracking control problem of self-driving vehicles under sensor and actuator attacks. In order to achieve path tracking without lateral velocity information, [29] used a robust H1 static output feedback controller based on a hybrid genetic algorithm and linear matrix inequalities with good results.

In addition, for those systems incorporated with AI and ML, the model of the neural network can be effective in enhancing the robustness of the model. [30], [31] studied event-triggered deep reinforcement learning based on event triggering and proposed an Event-Triggered Deep Q-network (ETDQN) suitable for autonomous driving decision making. In order to solve directional planning problems such as turning at rational intersections, [32] established a learning autonomous driving motion planning method based on conditional deep Q networks and fuzzy logic. In order to solve the problem of illegal eavesdropping and interference in autonomous vehicle networks, [33] proposed a solution based on distributed Kalman filtering and deep reinforcement learning techniques. For the problem of autonomous flight of a tilt-rotor aircraft in a complex and dense environment, [34] proposed a motion planning algorithm using the principle of minimum energy, the A* algorithm and an optimised cost function.

However, pre-determined algorithms like the Kalman filter are designed based on assumptions on noise patterns, which is not always the case in real life. In addition, almost all of the strategies mentioned above ignore the breakdown that is actually potential in the real world. To further enhance the robustness of the automatic control systems, especially in extreme conditions, a new framework is in urgent need.

Besides, many works are done in the UAV control problem mentioned above.

Deep Q-Network (DQN) was applied to train UAVs to perform autonomous control tasks, enabling UAVs to plan safe flight paths, avoid obstacles and reach their destinations in complex environments, and achieved good results in tests [37], [38],

[39], [40]. In order to solve the problem that traditional reinforcement learning and DQN cannot handle the continuous action space, [35] used the DDPG algorithm for optimisation to output continuous and smooth control values through the policy gradient method.

In this paper, we won't do further optimization on the DQN itself. Instead, we will take it as part of our framework and adjust it only for its interconnection with other parts. We plan to use a dual Deep Q-Network (DDQN), a reinforcement learning algorithm that combines deep learning and Q-learning [41].

1.3 Structure of the Paper

In Section 2, we will first formulate the specific problem we study on mathematically. An original virtual environment will be introduced in 2.1 as the cornerstone of the following research. The robust control problem will then be carefully defined in 2.2.

In Section 3, we will introduce the framework proposed in detail. The internal structure and concrete settings will be introduced in 3.1. As our system includes several neural networks to be trained beforehand, the training method and process will be introduced in 3.2 in detail.

In Section 4, we will evaluate the capacity of our system in this specific mission by comparing it with a control group. In 4.1, we will first evaluate the anti-interference capability of the system, especially the filter module, by comparing its accuracy under the given environment. In 4.2, we apply them to the virtual environment and compare their outcomes to evaluate the overall performance.

In Section 5, we will conclude our results and indicate the possible future works.

Section 2. Problem Formulation

In this section, we are going to formulate the specific problem we study on. An original virtual environment will be introduced first as the base of our subsequent research. The detailed configuration of the environment will be given in 2.1.2. Then, we will define the problem of robust control in our environment.

2.1 Virtural Environment

2.1.1 Overview

Our virtual environment is designed to simulate post-disaster UAV rescue operations under extreme conditions.

A brief visualization of the environment is shown in Fig. 1.

At the center of this virtual environment lies a single-story building, which consists of two perimeter walls (represented by black lines in Fig. 1), dividing the entire area into three distinct zones labeled Zone 0, Zone 1, and Zone 2. These zones are interconnected by gates of different sizes on the walls. An exit is located in Zone 2 (marked by the green area in the Fig. 1).

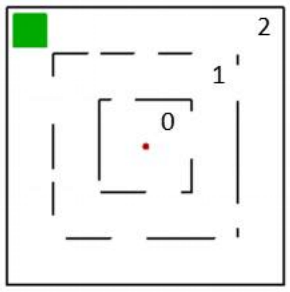

Fig. 1. Virtual Environment Visualization

An agent (represented by the red point in the figure) initially positioned within the inner area (a random location within Zone 0) must, at each time step, choose to move in one of the four cardinal directions (north, south, east, or west) or to permanently halt its movement and waiting for further rescue operations. The velocity of the agent suffers a tiny random interference at each move to simulate the mechanical error in real life condition, making it impossible for any system to precisely predict the position after any move.

At each time step, the agent receives positional information from GPS about its current position. The original observation is subject to continuous random interference to simulate the unstable post-disaster condition. In addtion, to simulate possible severe secondary disasters, a breakdown is set to happen at a random time step.

2.1.2 Detailed Configurations

The precise coordinate setting of the background is shown in Fig. 2.

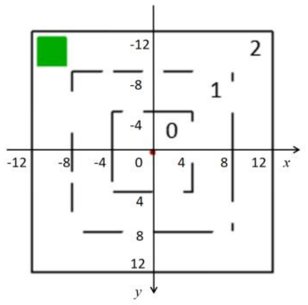

Fig. 2. Coordinate Visualization

The velocity at each time step is defined as follows:

$$v = v_0 + \alpha\varepsilon$$

where

$$\varepsilon \sim N(0, \sigma^2)$$

$$\varepsilon < \mathrm{b}$$

In this specific environment, we have

$$v_0 = 0.3$$
$$\alpha = 0.05$$
$$\sigma^2 = 1$$
$$\mathrm{b} = 2$$

The persistent random interference can be mathematically described as follows:

$$pos_{observation} = pos_{real} + \varepsilon$$

$$\varepsilon = \sum \alpha_i \varepsilon_i$$

where ε_i obey different types of probability distributions, including Gaussian distribution, uniform distribution. α_i are coefficients.

In this specific environment, we have two groups of setting:

Group 1:

$$\alpha_1 = 0.8, \alpha_2 = 0.2$$
$$\varepsilon_1 \sim N(0,2), \varepsilon_1 < 3$$
$$\varepsilon_2 \sim U(-1,1)$$

Group 2:

$$\alpha_1 = 0.8, \alpha_2 = 0.2$$
$$\varepsilon_1 \sim N(0,3), \varepsilon_1 < 5$$
$$\varepsilon_2 \sim U(-1,1)$$

Given that the noise can reach 10 (Group 1) or 14 (Group 2) times of the velocity per time step, they can be regarded as significant enough to simulate the hardest condition in real life. The two groups will be used in Section 4 separately for examination.

A potential breakdown that can occur at any time t_0 and persist thereafter, manifesting in any of the following three scenarios:

Scenario 1:

$$x_{observation} = x_{real} + \varepsilon + \mathrm{b} \ , \ |\mathrm{b}| \gg \varepsilon$$

Scenario 2:

$$x_{observation} = random$$

Scenario 3:

$$x_{observation(t=T)} = x_{observation(t=t_0)}$$

In this specific environment, we have

$$\mathrm{b} = U(6,10)$$
$$t_0 = U(10, 300)$$

which is determined at the beginning of each episode, ensuring that its probability of occurrence gradually increases over time and it won’t happen at the very beginning.

2.2 Robust Control Problem

To be regarded as robust, an agent must first avoid making the worst decision as far as possible while gradually getting closer to its final goal.

In our settings, we assume that any collision with the wall is fatal and getting to the exit area means immediate escape. In addition, we take intentional and permenant stop as an alternative option.

Therefore, the agent should first try its best to avoid ended up in a crash, and then try to get to the exit as soon as possible.

Noted that there is a possible breakdown in our environment, and once it happens, there won't be enough information for the agent to get its current position. In such cases, we regard intentional stop as the most robust choice.

Therefore, the robustness should be evaluated by the collision rate, the escape rate and the accuracy of intentional stop.

Section 3. Robust Autonomous UAV Control System

In this section, we are going to propose a new robust autonomous UAV control system which consists of three main sections. We will begin with an overview and then describe the internal structure of each section. There are several neural networks used in the system and they are trained separately. The training settings and process will be introduced in 3.2.

3.1 System Structure

The internal structure of our system is shown in Fig. 3.

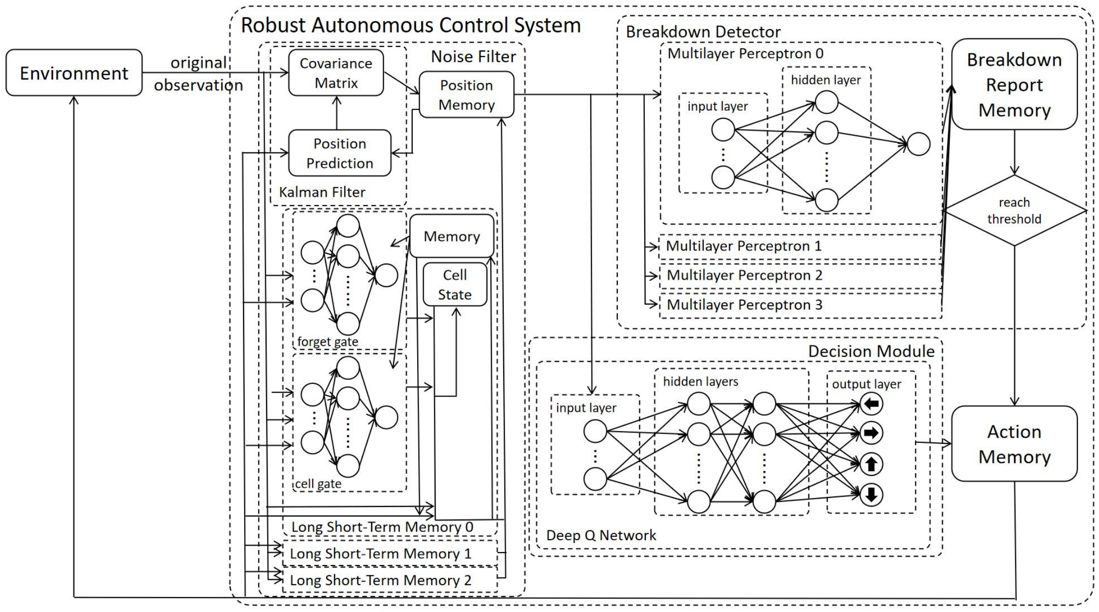

Fig. 3. Internal Structure of the Robust Autonomous Control System

3.1.1 Overall Structure

As a whole, the system is an end-to-end one. It takes the original observation of its position as input and gives out an action as output at each time step.

The system mainly have three memory modules containing its past predicted positions, breakdown reports and past actions.

When receiving the original observation at a time step, the Noise Filter Module will process it first. The module will call data from the position memory and the action memory to get a much more reliable position and put it into the position memory.

After that, the Breakdown Detector Module call a set of data in the position memory. The module will examine the set and store the report into the breakdown report memory. Once the number in the memory reaches the threhold, a breakdown is assumed to happen and the system take the action to stop.

Otherwise, the Decision Module call the latest data in the position memory and find out the optimal action under such position. The action will be stored in the action memory and then be called as final output.

3.1.2 Noise Filter

The Noise Filter Module contains a Kalman filter model and three separate Long Short-Term Memory (LSTM) networks.

The Kalman filter makes prediction of its current position by adding up the memory of the last position and the expected move determined by the action. Then it uses a covariance matrix to combine its prediction with the current observation.

The LSTM network used in this system is deliberately designed. More specifically, it has classical cell state, memory, forget gate, but its input gate is simplified and its output gate is replaced by a pre-designed algorithm.

The forget gate takes the observation, the expected move determined by the action, the data in its own memory and the current value of cell state as input (7 dimensional). It contains one hidden layer and has 10 nodes in the layer. Its output is processed by a sigmoid function and thus ranges from 0 to 1. The output *f* determines how much of the cell state is forgotten.

The cell gate takes the observation, the expected move determined by the action and the data in its own memory of cell state as input (6 dimensional). It also contains one hidden layer and has 10 nodes in the layer. Its output is processed by a sigmoid function and thus ranges from 0 to 1. The output *i* determines how the cell state is renewed.

Then the cell state *c* is renewed as follows:

$$c_{new} = c \times f + i \times (1 - f)$$

The network gives its processed position bythe following pre-designed algorithm:

$$pos = pos_{observation} * (1 - c) + (pos_{memory} + v_{memory}) * c$$

This alteration is introduced for easier training process and higher efficiency.

At the beginning of each episode, the cell state will be reseted to 0 and the memory will be reseted by the first observation.

To further decrease the randomness, the module takes the average of the outcome of both the Kalman filter and the three LSTMs as its final outcome.

3.1.3 Breakdown Detector

This section mainly consists of Multilayer Perceptron (MLP). Four separate MLPs are built to achieve better performance. Each of them contains one hidden layer and has 20 nodes in the layer. Its output is processed by a sigmoid function and thus ranges from 0 to 1. Each of them report when the final value is greater than 0.5.

If at least one of these MLPs report, the module add 1 to the breakdown report memory. If none of them report, the memory will be reset to 0. When the memory reaches 3, a breakdown is believed to exist and the module will directly determine the final action.

3.1.4 Decision Module

The Decision Module contains only a Deep Q Network (DQN). It has two hidden layers with 10 nodes in each layer. It calls the current processed postion in the position memory as input and outputs four values based on the current position, representing the quality of moving towards each direction. Then the module finds out the direction with the maxium Q value and store it in the action memory.

3.2 Training Process

There are totally 8 separate neural networks in our system. They are all trained separately beforehand in slightly different virtual environments.

3.2.1 LSTM Training

The three LSTMs are trained under the same environment with different initial parameters. The environment contains an agent whose moves are similar to the final environment. The observation pattern is exactly the same as the final pattern, but the walls, the exit and the potential breakdown are removed. The LSTM received exactly the same input as mentioned in 3.1.2. In the training process, it get the actual postion to optimize its parameters. The suggested c^* is calculated as follow:

$$c^* = \frac{x_{real} - x_{observation}}{x_{memory} + v_{memory} - x_{observation}}$$

Then the loss function of c is define as follow:

$$loss_c = 0.5 * (c - c^*)^2$$

The parameters in the two gates are then updated according to the gradient descent algorithm. The learning rate gradually decreases from 0.01 to 0.001. The update momentum is set to 0.2. The batch size is set to 4. In the experiment, the models are fully trained after around 10000 episodes, when their separate accuracy stay unchanged.

3.2.2 MLP Training

The four MLPs are trained in the same virtual environment with different initial parameters. The environment is almost the same as mentioned in 3.2.1 with the potential breakdown is added. Every MLP receives the input as mentioned in 3.1.3, and generate a output ω range from 0 to 1 (due to the sigmoid function placed in the end) to represent the possibility to report. Then the label τ of each set of input is 1 if a breakdown exists and otherwise 0.

The loss function is defined as follow:

$$loss_\omega = 0.5 * (\omega - \tau)^2$$

The parameters are then updated according to the gradient descent algorithm. The learning rate gradually decreases from 0.01 to 0.001. The update momentum is set to

0.2. The batch size is set to 4. In the experiment, the models are fully trained after 20000 episodes, when their separate accuracy stay unchanged.

3.2.3 DQN Training

In this paper, we adopt double deep Q-learning method to train the DQN section. The virtual environment is the non-interference version the final environment with deliberately set rewards.

The rewards in the environment are set to encourage the agent to gradually shift its zone and get closer to the exit. The reward to get to the exit is huge, and the cost to collide with any wall is also huge.

The model is expected to estimate the quality of each action at any given state, which can be defined by Bellman Equation:

$$Q^*(s_t, a_i) = r_{(s_t, a_i)} + \gamma * \max\{Q^*(s_{t+1}, a_i) | a_i \in A_{t+1}\}$$

When training, the model can get the exact reward of its latest move and updates its parameters θ by gradient descent algorithm, minimizing the loss function:

$$L_\theta = \left(Q_\theta(s_t, a_i) - y_{(t,i)}\right)^2$$

where

$$y_{(t,i)} = r_{(s_t, a_i)} + \gamma * \max\{Q_{\theta'}(s_{t+1}, a_i) | a_i \in A_{t+1}\}$$

where θ' represents the parameters of the target network introduced by DDQN to optimize the training process.

The learning rate gradually decreases from 0.01 to 0.001. The update momentum is set to 0.2. The exploration rate is set to 0.2. The target DQN is updated every 10 episodes.

In our experiment, it is fully trained after about 100000 episodes when it can always succeed in getting to the exit under precise observation.

Section 4. System Capability Evaluation

In this section, we will first evaluate the anti-interference capability of the system, especially the filter module, by comparing its accuracy under the given environment with other models. Then, we apply the system and other established models to the virtual environment and compare their outcomes to evaluate the overall performance.

The control group includes a DQN and an established hidden markov model, which can make predictions and combine it with the observation with an increasing confidence rate.

4.1 Anti-Interference Capability Evaluation

To evaluate the anti-interference capability, we have applied several models to our environment and record the error of the filter at each step.

The average varriance of the processed position of different models are shown in Fig. 4.

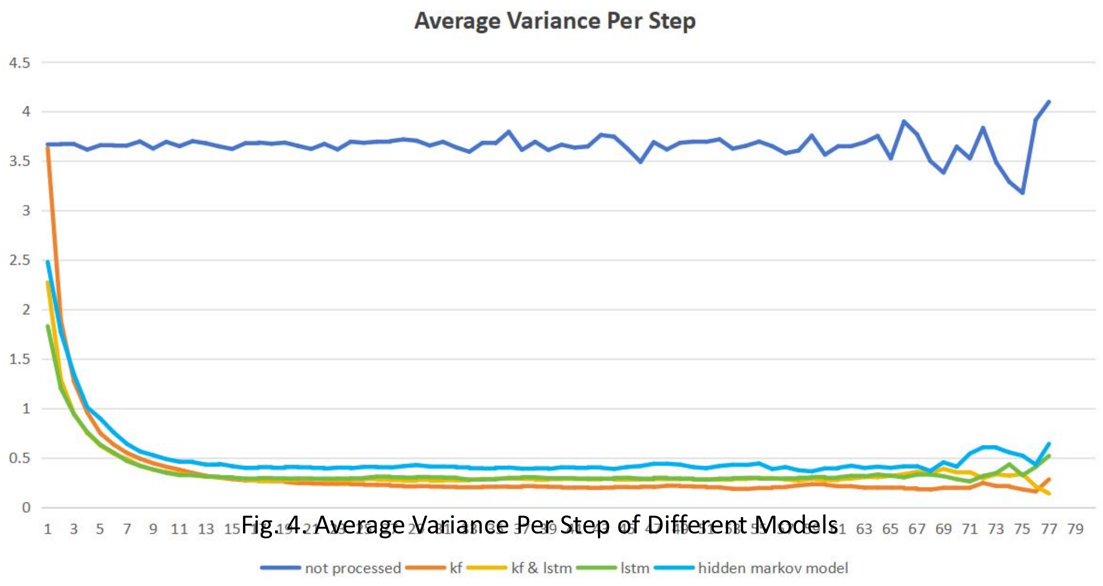

Fig. 4. Average Variance Per Step of Different Models

The average varriance of every given model gradually decreases as they receive more data.

The established hidden markov model has the highest varriance in most steps, which is regarded as the worst filter among them.

The Kalman filter alone takes too many steps to lower its varriance, leading to possible early collisions.

The LSTM alone doesn't perform well enough at last, leading to eventual failure.

Therefore, the model which combines the Kalman filter and the LSTM is supposed to be the best one to work well early.

As another aspect of anti-Interference capability, the maxium error at each step is also recorded in Fig. 5.

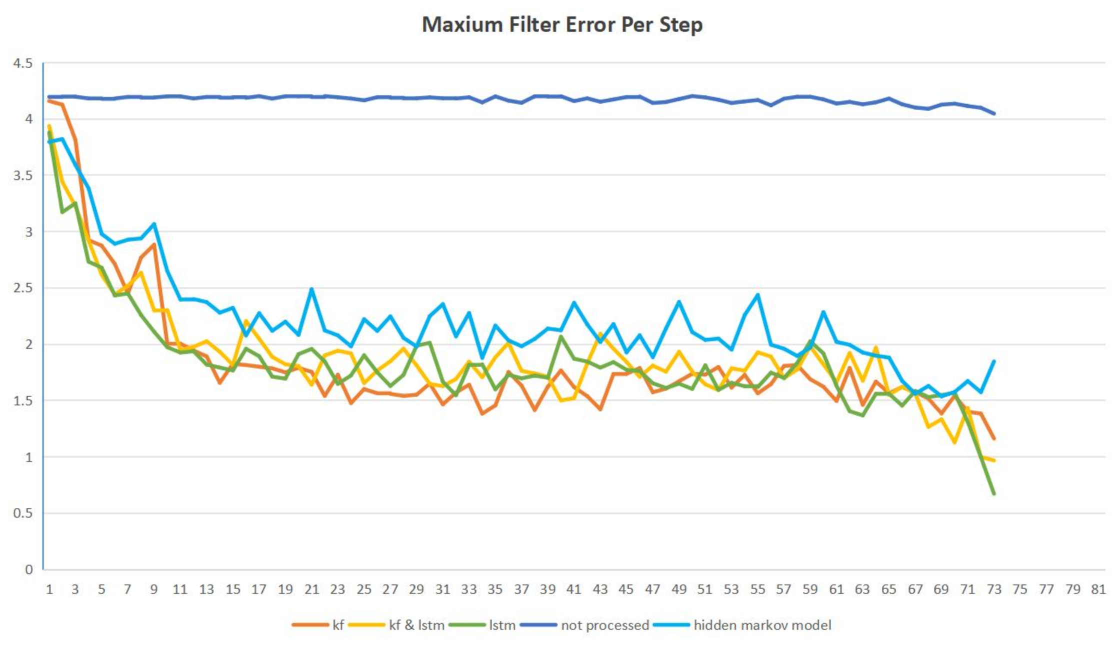

Fig. 5.Maxium Filter Error Per Step of Different Models

The established hidden markov model is still the worst among the filters while the other three are quite similar to each other.

Therefore, it's reasonable to adopt both Kalman filter and LSTM in our system to filter the interference.

4.2 Overall Performance Evaluation

To evaluate the overall performance, we apply our system along with the control group mentioned before to the environment with two groups of settings mentioned in 2.1.2. Each experiment is repeated 10000 times to get valid data.

The outcome of the methods in Group 1 is shown in Table 1.

Table 1.Performance of the Robust System and Other Methods under Significant Interference and Potential Breakdowns

Method	Collision	Escape	Stop	Detection success
DQN	75.9%±0.5%	24.1%±0.5%	0.0%	0.0%
Hidden Markov Model	39.9%±0.5%	60.1%±0.5%	0.0%	0.0%
Robust System	20.7%±0.5%	61.8%±0.5%	17.5%±0.5%	94.5%±0.5%

The first column in the table indicates the methods, followed by their proportion among the three possible outcomes. The last column of data represents the proportion of successful detection of potential breakdowns.

In order to more intuitively demonstrate how our system and other methods react to the environment, we visualize one of the episodes for two of them.

Fig. 6 and Fig. 7 respectively shows the results of the performance of the DQN alone and the robust system we proposed, where the red line represents the route the agent actually takes and the blue dots are the record of observations.

The level of interference can also be intuitively seen in the figures.

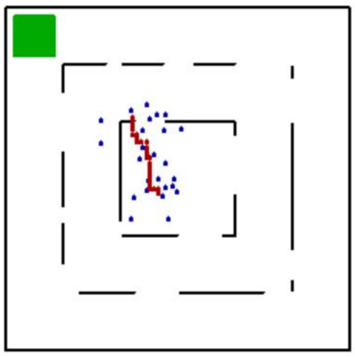
Fig. 6. Performance 1 of DQN Alone

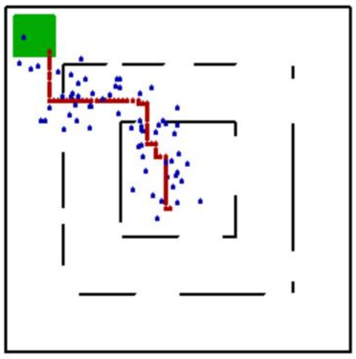
Fig. 7. Performance 1 of the Robust System

According to the data generated in the experiment, a DQN alone is the worst in this environment as it ends up crashing in most cases. A hidden markov model does better as it succeeds in fingding the exit in around 60% of the cases. However, it doesn't have the ability to detect the potential breakdown and ends up crashing in the rest of the episodes.

In comparison, our robust model can detect the breakdowns before a collision happens in most cases. The escape rate also shows that it seldom report breakdowns incorrectly. Therefore, it manages to maintain a slightly higher escape rate with a far lower collision rate. As mentioned in 2.2, our system better meets the requirements in the robust control problem in this environment.

The outcome of the methods in Group 2 is shown in Table 2.

Table 2.Performance of the Robust System and Other Methods under Extreme Interference and Potential Breakdowns

Method	Collision	Escape	Stop	Detection success
DQN	97.7%±0.5%	2.3%±0.5%	0.0%	0.0%
Hidden Markov Model	52.4%±0.5%	47.6%±0.5%	0.0%	0.0%
Robust System	34.3%±0.5%	51.0%±0.5%	14.7%±0.5%	92.0%±0.5%

In this group, we also visualize one of the episodes for two of the methods.

Fig. 8 and Fig. 9 respectively shows the results of the performance of the DQN alone and the robust system we proposed.

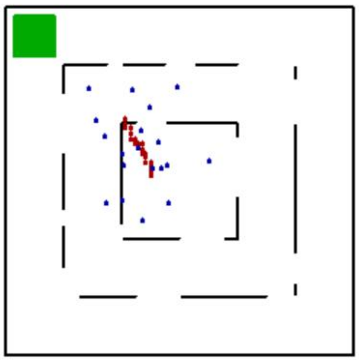
Fig. 8. Performance 2 of DQN Alone

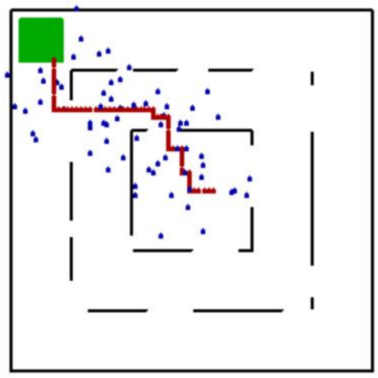
Fig. 9. Performance 2 of the Robust System

Obviously, the interference is greatly increased to an extreme level in Group 2.

All of the methods are negatively affected by the increased interference. The DQN alone can hardly handle the environment any more. The hidden markov model is also seriously affected with a lower escape rate.

The accuracy of breakdown detection of our robust system is also affected, but compared with the control group, it still has a great advantage on the collision rate and the escape rate. It’s safe to say that our system is still the best method under such extreme environment according to 2.2.

Therefore, it’s safe to conclude that the system we proposed is more effective and robust in dealing with extreme environments than the established methods.

Section 5. Conclusion and Future Work

5.1 Conclusion

In this paper, we provide an in-depth study on the robust control problem and come to a novel framework. To prove its robustness, we construct a specific system for robust autonomous UAV control under extreme conditions based on Kalman filter, LSTM, MLP and DQN. The inner modules are carefully interconnected with each other within the system and finally make the system an end-to-end one. According to the experiment, the new system is more robust than other existing methods (DQN and traditional hidden markov models). The results show that delicate built-in filters and detectors based on the combination of the pre-determined algorithms and pre-trained neural networks are effective in improving the robustness of a control system.

5.2 Future Works

Although the system we proposed has achieved some success in dealing with extreme conditions, there is still room for improvement. First, the LSTM network integrated in the system is actually a simplified version for easier training, but a complete or even reinforced LSTM network may hopefully have a stronger ability to filter the noise, which requires further research. Second, the DQN in the decision module in our system is a quite naive one which is still prone to noise. A more advanced decision module like the Robust-DDPG in [36] may hopefully improve the final results, which also requires further research.

On the other hand, although the specific system in this paper only focus on the UAV control problem, the framework of filter-detector-decision can hopefully be used in other fields to enhance the robustness of autonomous control systems. Possibly, some specific adjustments (especially on the decision module) is needed which requires specific research. Hopefully, the framework can serve as a reference for future design for robust autonomous control systems in other fields.

Reference

[1] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, “Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination,” Computer Networks, vol. 101, pp. 158 – 168, 2016.

[2] M. Bertolini, D. Mezzogori, M. Neroni, and F. Zammori, “Machine learning for industrial applications: A comprehensive literature review,” Expert Systems with Applications, vol. 175, 2021.

[3] G. Fragapane, D. Ivanov, M. Peron, F. Sgarbossa, and J. O. Strandhagen, “Increasing flexibility and productivity in industry 4.0 production networks with autonomous mobile robots and smart intralogistics,” Annals of Operations Research, vol. 308, no. 1-2, pp. 125 – 143, 2022.

[4] G. Fragapane, R. de Koster, F. Sgarbossa, and J. O. Strandhagen, “Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda,” European Journal of Operational Research, vol. 294, no. 2, pp. 405 – 426, 2021.

[5] C.-S. Karavas, G. Kyriakarakos, K. G. Arvanitis, and G. Papadakis, “A multi-agent decentralized energy management system based on distributed intelligence for the design and control of autonomous polygeneration microgrids,” Energy Conversion and Management, vol. 103, pp. 166 – 179, 2015.

[6] S. A. H. Mohsan, M. A. Khan, F. Noor, I. Ullah, and M. H. Alsharif, “Towards the unmanned aerial vehicles (uavs): A comprehensive review,” Drones, vol. 6, no. 6, 2022.

[7] P. K. Reddy Maddikunta, S. Hakak, M. Alazab, S. Bhattacharya, T. R. Gadekallu, W. Z. Khan, and Q.-V. Pham, “Unmanned aerial vehicles in smart agriculture: Applications, requirements, and challenges,” Ieee Sensors Journal, vol. 21, no. 16, pp. 17 608 – 17 619, 2021.

[8] N. Zhao, W. Lu, M. Sheng, Y. Chen, J. Tang, F. R. Yu, and K.-K. Wong, “Uav-assisted emergency networks in disasters,” Ieee Wireless Communications, vol. 26, no. 1, pp. 45 – 51, 2019.

[9] S. Xu and H. Peng, “Design, analysis, and experiments of preview path tracking control for autonomous vehicles,” Ieee Transactions on Intelligent Transportation Systems, vol. 21, no. 1, pp. 48 – 58, 2020.

[10] Z. Peng, J. Wang, and Q.-L. Han, “Path-following control of autonomous underwater vehicles subject to velocity and input constraints via neurodynamic optimization,” Ieee Transactions on Industrial Electronics, vol. 66, no. 11, pp. 8724 – 8732, 2019.

[11] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, “Optimal and autonomous control using reinforcement learning: A survey,” Ieee Transactions on Neural Networks and Learning Systems, vol. 29, no. 6, pp. 2042 – 2062, 2018.

[12] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, “A survey of deep learning applications to autonomous vehicle control,” Ieee Transactions on Intelligent Transportation Systems, vol. 22, no. 2, pp. 712 – 733, 2021.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning, a brief survey,” Ieee Signal Processing Magazine, vol. 34, no. 6, pp. 26 – 38, 2017.

[14] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Perez, “Deep reinforcement learning for autonomous driving: A survey,” Ieee Transactions on Intelligent Transportation Systems, vol. 23, no. 6, pp. 4909 – 4926, 2022.

[15] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction second edition introduction,” Editorial Material; Book Chapter, 2018.

[16] V. Talpaert, I. Sobh, B. R. Kiran, P. Mannion, S. Yogamani, A. El-Sallab, and P. Perez, “Exploring applications of deep reinforcement learning for real-world autonomous driving systems,” in 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP), 2019, pp. 564 – 572.
[17] T. Zhang, J. Lei, Y. Liu, C. Feng, and A. Nallanathan, “Trajectory optimization for uav emergency communication with limited user equipment energy: A safe-dqn approach,” Ieee Transactions on Green Communications and Networking, vol. 5, no. 3, pp. 1236 – 1247, 2021.
[18] X. Wu, H. Chen, C. Chen, M. Zhong, S. Xie, Y. Guo, and H. Fujita, “The autonomous navigation and obstacle avoidance for usvs with anoa deep reinforcement learning method,” Knowledge-Based Systems, vol. 196, 2020.
[19] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, “Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving,” Transportation Research Part C-Emerging Technologies, vol. 117, 2020.
[20] J. Chen, S. E. Li, and M. Tomizuka, “Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning,” Ieee Transactions on Intelligent Transportation Systems, vol. 23, no. 6, pp. 5068 – 5078, 2022.
[21] A. Singla, S. Padakandla, and S. Bhatnagar, “Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge,” Ieee Transactions on Intelligent Transportation Systems, vol. 22, no. 1, pp. 107 – 118, 2021.
[22] N. H. Amer, H. Zamzuri, K. Hudha, and Z. A. Kadir, “Modelling and control strategies in path tracking control for autonomous ground vehicles: A review of state of the art and challenges,” Journal of Intelligent Robotic Systems, vol. 86, no. 2, pp. 225 – 254, 2017.
[23] Q. Zhang, J. Lin, Q. Sha, B. He, and G. Li, “Deep interactive reinforcement learning for path following of autonomous underwater vehicle,” Ieee Access, vol. 8, pp. 24 258 – 24 268, 2020.
[24] S. Guo, X. Zhang, Y. Du, Y. Zheng, and Z. Cao, “Path planning of coastal ships based on optimized dqn reward function,” Journal of Marine Science and Engineering, vol. 9, no. 2, 2021.
[25] W. Wei, J. Wang, Z. Fang, J. Chen, Y. Ren, and Y. Dong, “3u: Joint design of uav-usv-uuv networks for cooperative target hunting,” Ieee Transactions on Vehicular Technology, vol. 72, no. 3, pp. 4085 – 4090, 2023.
[26] J. Li, X. Xiang, D. Dong, and S. Yang, “Prescribed time observer based trajectory tracking control of autonomous underwater vehicle with tracking error constraints,” Ocean Engineering, vol. 274, 2023.
[27] F. Tian, Z. Li, F.-Y. Wang, and L. Li, “Parallel learning-based steering control for autonomous driving,” Ieee Transactions on Intelligent Vehicles, vol. 8, no. 1, pp. 379 – 389, 2023.
[28] H.-T. Sun, C. Peng, X. Ge, and Z. Chen, “Secure event-triggered sliding control for path following of autonomous vehicles under sensor and actuator attacks,” Ieee Transactions on Intelligent Vehicles, vol. 9, no. 1, pp. 981 – 992, 2024.
[29] C. Hu, H. Jing, R. Wang, F. Yan, and M. Chadli, “Robust output-feedback control for path following of autonomous ground vehicles,” Mechanical Systems and Signal Processing, vol. 70-71, pp. 414 – 427, 2016.
[30] J. Lu, L. Han, Q. Wei, X. Wang, X. Dai, and F.-Y. Wang, “Eventtriggered deep reinforcement learning using parallel control: A case study in autonomous driving,” Ieee Transactions on Intelligent Vehicles, vol. 8, no. 4, pp. 2821 – 2831, 2023.
[31] F. Dang, D. Chen, J. Chen, and Z. Li, “Event-triggered model predictive control with deep

reinforcement learning for autonomous driving," Ieee Transactions on Intelligent Vehicles, vol. 9, no. 1, pp. 459 – 468, 2024.
[32] L. Chen, X. Hu, B. Tang, and Y. Cheng, "Conditional dqn-based motion planning with fuzzy logic for autonomous driving," Ieee Transactions on Intelligent Transportation Systems, vol. 23, no. 4, pp. 2966 – 2977, 2022.
[33] Y. Yao, J. Zhao, Z. Li, X. Cheng, and L. Wu, "Jamming and eavesdropping defense scheme based on deep reinforcement learning in autonomous vehicle networks," Ieee Transactions on Information Forensics and Security, vol. 18, pp. 1211 – 1224, 2023.
[34] L. Yang, Y. Xu, J. Tian, and D. Wang, "Autonomous path planning and anti-wind disturbance control of tilt-rotor aircraft in complex and dense environment," Journal of Applied Science and Engineering, vol. 28, no. 1, pp. 61 – 73, 2025.
[35] Q. Yang, Y. Zhu, J. Zhang, S. Qiao, J. Liu, and Ieee, "Uav air combat autonomous maneuver decision based on ddpg algorithm," in IEEE 15th International Conference on Control and Automation (ICCA), ser. IEEE International Conference on Control and Automation ICCA, 2019, pp. 37 – 42.
[36] K. Wan, X. Gao, Z. Hu, and G. Wu, "Robust motion control for uav in dynamic uncertain environments using deep reinforcement learning," Remote Sensing, vol. 12, no. 4, 2020.
[37] G. Munoz, C. Barrado, E. Cetin, and E. Salami, "Deep reinforcement learning for drone delivery," Drones, vol. 3, no. 3, 2019.
[38] O. Perez-Gil, R. Barea, E. Lopez-Guillen, L. M. Bergasa, C. GomezHuelamo, R. Gutierrez, and A. Diaz-Diaz, "Deep reinforcement learning based control for autonomous vehicles in carla," Multimedia Tools and Applications, vol. 81, no. 3, pp. 3553 – 3576, 2022.
[39] X. Ruan, D. Ren, X. Zhu, J. Huang, and Ieee, "Mobile robot navigation based on deep reinforcement learning," in 31st Chinese Control And Decision Conference (CCDC), ser. Chinese Control and Decision Conference, 2019, pp. 6174 – 6178.
[40] L. Li, D. Wu, Y. Huang, and Z.-M. Yuan, "A path planning strategy unified with a colregs collision avoidance function based on deep reinforcement learning and artificial potential field," Applied Ocean Research, vol. 113, 2021.
[41] N. Gao, Z. Qin, X. Jing, Q. Ni, and S. Jin, "Anti-intelligent uav jamming strategy via deep q-networks," Ieee Transactions on Communications, vol. 68, no. 1, pp. 569 – 581, 2020.
[42] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. G. van Sloun, and Y. C. Eldar, "Kalmannet: Neural network aided kalman filtering for partially known dynamics," Ieee Transactions on Signal Processing, vol. 70, pp. 1532 – 1547, 2022.