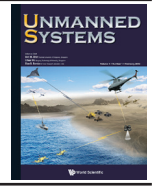
Unmanned Systems, Vol. 14, No. 1 (2026) 257–266

DOI: 10.1142/S2301385025500931

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us



Modeling of Small Unmanned Helicopter Using a Self-Constructed Kernel Function APSO-LSSVM

Jian Zhou*, Junyi Shi, Weixin Wang, Jian Lu

School of Electronic Information, Xi 'an Polytechnic University, Lintong, Xi'an, Shaanxi, China

The complex nonlinear and strongly coupled dynamics of small unmanned helicopters make mathematical modeling challenging. Traditional approaches often rely on least squares support vector machine (LSSVM) algorithms using standard kernel functions, which are limited in their learning and generalization capabilities, resulting in insufficient accuracy for flight control systems. This paper proposes an adaptive particle swarm optimization-based LSSVM (APSO-LSSVM) method, incorporating a self-constructed kernel function. Using Mercer's theorem, the custom kernel addresses the limitations of conventional kernels and is integrated into the LSSVM framework. An adaptive particle swarm optimization algorithm, capable of dynamically adjusting inertia weights and learning factors, optimizes the model parameters, overcoming the standard particle swarm optimization's tendency to get trapped in local optima. The model is trained using flight test data from a self-developed small unmanned helicopter, and its identification performance is cross-validated in the time domain against traditional models. Experimental results show that the proposed method significantly improves the modeling accuracy of small unmanned helicopters.

Keywords: Self-constructed kernel function; adaptive particle swarm optimization algorithm; least squares support vector machine; small unmanned helicopter; model identification.

US

1. Introduction

Small unmanned helicopters (SUHs) are frequently utilized in military and civil applications given their little stature, light weight, ability to hover at a fixed point, low-speed cruise, and ease of maneuvering. A high-precision model is the basis for realizing high-performance autonomous flight control of SUHs, and since SUHs are a kind of controlled system with multiple inputs and multiple outputs, nonlinearity, strong coupling, and open-loop instability, it is exceedingly challenging to create a dynamics model that is accurate [1]. The main methods for modeling SUHs are first-principle techniques and techniques for system identification [2]. The first principle method constructs a mathematical model based on the aerodynamic theory of each component of a SUH. This method can reflect the theoretical model of the physical mechanism of the system, but for the parameters that cannot be obtained by calculation are processed by the assumption of valuation or elimination, which makes the model accuracy greatly reduced, affecting the accuracy of the controller design. Through the functional link between the measured input and output data, the system identification approach builds the mathematical model of SUH using flight test data [3, 4]. This method has higher model accuracy, but needs a significant quantity of measured data to ensure the accuracy of identification. The commonly used methods are frequency domain identification method assisted by optimization algorithm and technique for identifying the temporal domain [5]. The system's frequency domain response characteristics can be obtained using the frequency domain identification approach, which is convenient to analyze the system performance in the frequency domain and has certain robustness to noise. However, it requires Fourier transform of the data and usually only obtains a linear model of the system, which is only applicable to stable or quasi-stationary systems [6, 7].

Received 15 May 2024; Accepted 1 December 2024; Published 17 January 2025. This paper was recommended for publication in its revised form by editorial board member, Biao Wang.
Email Address: *zhoujian0627@163.com*

Time-domain system identification does not require system stability and can estimate the system model directly from input and output data in the time domain. It has strong real-time performance and can produce both linear and nonlinear models [8, 9].

While deep learning methods are capable of handling complex time-series data, they typically require large datasets and substantial computational resources, making them impractical for real-world applications [10]. In comparison, extreme learning machine (ELM) methods offer fast training speeds, but they are highly sensitive to data noise, potentially compromising model stability [11]. Similarly, Gaussian process regression (GPR) performs well on small datasets but suffers from high computational complexity, restricting its use in large-scale applications [12]. Given the highly nonlinear nature of SUHs, linear regression models are inadequate. Therefore, this paper adopts an adaptive particle swarm optimization least squares support vector machine (APSO-LSSVM) approach, which offers several advantages, including fast training, robust generalization, and strong adaptability to small datasets. APSO-LSSVM builds on the traditional support vector machine (SVM) framework by incorporating least squares for rapid parameter estimation and adaptive particle swarm optimization (APSO) to automatically adjust model parameters, thereby enhancing both accuracy and computational efficiency. The LSSVM, introduced by Suykens [13], extends the original SVM algorithm proposed by Vapnik [14]. LSSVM uses a nonlinear mapping to transform input samples from a low-dimensional space to a higher-dimensional feature space, where it identifies linear relationships between input and output variables. By replacing inequality constraints with equality constraints and optimizing the squared error, LSSVM accelerates the solution process while maintaining strong generalization performance. Although LSSVM has been successfully applied in model predictive control [15, 16], conventional LSSVM typically relies on a single kernel function, which may not capture the full characteristics of the dataset, leading to suboptimal prediction accuracy. To address this limitation, researchers such as Tian *et al.* [17, 18] proposed composite kernel functions tailored to specific dataset characteristics, compensating for the shortcomings of single kernel approaches.

This paper proposes a SUH modeling method based on a self-constructed kernel model using APSO-LSSVM. The main contributions of this work are as follows:

(1) A complex kernel function was constructed by combining radial basis function (RBF) and polynomial kernel functions based on kernel construction theory.
(2) A LSSVM prediction model for SUHs was developed.
(3) The APSO algorithm was applied to optimize model parameters, significantly speeding up the parameter search process compared to the standard particle swarm optimization algorithm.
(4) The method was validated using flight test data from a self-developed SUH. Its accuracy and effectiveness were demonstrated through comparisons with LSSVM models using both single and hybrid kernel functions.

This paper is arranged as follows. Section 2 introduces the construction of the custom kernel function and its integration into the LSSVM model. Section 3 describes the process of optimizing model parameters using the APSO algorithm. In Sec. 4, a comparative analysis of the proposed model is presented. Finally, Sec. 5 concludes the paper with a summary of the key findings.

2. Least Squares Support Vector Machines Based on Self-constructed Kernel Functions

This study employs the LSSVM for the modeling and identification of SUHs. The selection of an appropriate kernel function is crucial for determining model accuracy [19]. However, traditional algorithms are often constrained by standard kernel functions, which can limit predictive performance. To address this issue, a custom kernel function tailored to the unique characteristics of the SUH dataset is designed to enhance prediction accuracy.

2.1. *Self-constructed kernel function*

The authors of [20] explain that kernel methods embed data into a suitable vector space, called the feature space, where linear algebra, geometry, and statistical algorithms can be applied to identify relationships within the embedded data. According to integral operator theory, continuous symmetric functions that satisfy the conditions of Mercer's theorem in compact spaces can approximate the eigenfunctions of the original space. Thus, kernel functions must satisfy Mercer's condition to ensure valid function approximation. Kernel functions are typically constructed in two primary ways:

(1) Direct construction
The nonlinear mapping function is determined based on the characteristics of the data under study, and the kernel function corresponding to it is constructed with $K(u, v) = \phi(u) \cdot \phi(v)$. This construction method has a clear understanding of the nature of the application background data. However, in practical problems, finding the appropriate nonlinear mapping ϕ is a difficult task.
(2) Closed operations
The construction of operations on one or more kernels using analytic expressions in closed form follows from

Mercer's theorem, and we say that this class of kernel functions is closed under these operations if the property of finite semipositive definiteness is preserved.

Given legal kernel functions $k_1(x, x')$ and $k_2(x, x')$, the kernel functions constructed in the following forms are also legal [21]:

$$k(x, x') = ck_1(x, x'), \tag{1}$$

$$k(x, x') = f(x)k_1(x, x')f(x'), \tag{2}$$

$$k(x, x') = q(k_1(x, x')), \tag{3}$$

$$k(x, x') = \exp(k_1(x, x')), \tag{4}$$

$$k(x, x') = k_1(x, x') + k_2(x, x'), \tag{5}$$

$$k(x, x') = k_1(x, x')k_2(x, x'), \tag{6}$$

$$k(x, x') = k_3(\phi(x), \phi(x')), \tag{7}$$

$$k(x, x') = x^T A x'. \tag{8}$$

Given the complexity of SUHs as control systems, explicitly constructing a suitable nonlinear mapping through traditional methods proves challenging. Therefore, this paper adopts a closed operation approach to construct the required kernel function. Prior to constructing the kernel, it is essential to select several base kernel functions for further operations.

In traditional LSSVM algorithms, the radial basis function (RBF) is commonly used as the kernel function to map sample points into a high-dimensional space [22, 23]. The RBF kernel is expressed as follows:

$$k_R(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right). \tag{9}$$

The expression of the Poly kernel function is

$$k_P(x, x') = [(x^T \cdot x') + u]^v. \tag{10}$$

The RBF kernel is a local kernel with limited generalization ability, often leading to local optima and affecting model accuracy. In contrast, the polynomial kernel has strong global properties, making it suitable for capturing the overall characteristics of the data. However, it tends to perform poorly on high-dimensional data, increasing the risk of overfitting and sensitivity to parameter selection.

Based on kernel construction theory and closed operations, the custom kernel function is derived and expressed as

$$k(x, x_k) = \left[\exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) + p\right]^q. \tag{11}$$

By combining the advantages of both the RBF and polynomial kernels, this custom kernel not only enhances the model's generalization ability but also more effectively captures the complex nonlinear characteristics of the data. Therefore, the custom kernel theoretically possesses stronger representational power, significantly contributing to the improved modeling accuracy of SUHs.

2.2. *Introducing self-constructed kernel functions into LSSVM*

For the regression model in the original weight space:

$$y = f(x) = \omega^T \varphi(x) + b, \tag{12}$$

where $x \in R^n$ is a vector, $y \in R$, is a scalar, b represent the bias term. Let ω be the weight vector, and $\varphi(x) : R^n \to R^h$ be the mapping from the original space to a high-dimensional Hilbert space.

Given a training dataset $\{x_i, y_i\}_{i=1}^N$, based on the principle of structural risk minimization, the estimation problem can be formulated as the following optimization problem:

$$\begin{aligned} &\min_{\omega,b,\zeta} J(\omega, \zeta) = \frac{1}{2}\omega^T\omega + \frac{\lambda}{2}\sum_{i=1}^{N}\zeta_i^2 \\ &\text{s.t.} \quad y_i = \omega^T\varphi(x_i) + b + \zeta_i \quad (i = 1, 2, \ldots, N), \end{aligned} \tag{13}$$

where $\lambda \geq 0$ is the regularization parameter used to balance model complexity and accuracy, and it is a scalar. ζ_i^2 denotes the regression error between the actual output and the predicted value, which is also a scalar.

In order to resolve the optimization issue described above, the Lagrange multiplier method is presented in order to convert the initial issue into an unrestricted optimization issue.

$$L(\omega, b, \zeta, \alpha) = J(\omega, \zeta) - \sum_{i=1}^{N}\alpha_i[\omega^T\varphi(x_i) + b + \zeta_i - y_i], \tag{14}$$

where $\alpha_i (i = 1, 2, \ldots, N)$ represent the multiplier of Lagrange, and it is a scalar. Given the KKT circumstances, we can obtain

$$\begin{aligned} &\frac{\partial L}{\partial \omega} = 0 \to \omega = \sum_{i=1}^{N}\alpha_i\varphi(x_i), \\ &\frac{\partial L}{\partial b} = 0 \to \sum_{i=1}^{N}\alpha_i = 0, \\ &\frac{\partial L}{\partial \zeta_i} = 0 \to \alpha_i = \lambda\zeta_i, \\ &\frac{\partial L}{\partial \alpha_i} = 0 \to \omega^T\varphi(x_i) + b + \zeta_i - y_i = 0. \end{aligned} \tag{15}$$

By eliminating the original variables, the vector ω and the scalar ζ, the problem is simplified to the following dual

problem:

$$\begin{bmatrix} 0 & I_N^T \\ I_N & \Omega + I/\lambda \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix}, \tag{16}$$

where $y = [y_1, y_2, \ldots, y_N]^T$, $I_N = [1, 1, \ldots, 1]^T$, Ω is a square matrix of $L \times L$.

$$\Omega_{ij} = (\varphi(x_i))^T \varphi(x_j) = K(x_i, x_j) \quad (i, j = 1, 2, \ldots, N), \tag{17}$$

where $K(x_i, x_j)$ is the kernel function, the solutions of α and b can be obtained by solving formula (16), then, the LSSVM model can be expressed as

$$y = f(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x_j) + b. \tag{18}$$

Substituting the previously constructed self-constructed kernel function $k(x, x')$ into the LSSVM model expression, the LSSVM model according to the self-constructed kernel function is obtained.

3. Adaptive Particle Swarm Optimization Algorithm Optimizes Model Parameters

To enhance the optimization capability of model parameters, this study employs the APSO algorithm to determine the optimal parameters for the LSSVM model, thereby enabling automatic parameter tuning and enhancing the model's predictive accuracy.

Particle Swarm Optimization (PSO) is a swarm intelligence algorithm that simulates the foraging behavior of bird flocks to find optimal solutions [24]. PSO is known for its simplicity, fast convergence, and ability to locate global optima. However, since its inertia weight and learning factors remain fixed throughout iterations, PSO can become trapped in local optima, requiring careful parameter tuning and initialization. To overcome this limitation and enhance both global search capability and convergence speed, APSO introduces adaptive mechanisms that dynamically adjust the inertia weight and learning factors [25, 26]. These improvements help prevent premature convergence and allow the algorithm to explore the solution space more effectively. In comparison, while Genetic Algorithm (GA) offers strong global search capabilities, its slow convergence and tendency to get trapped in local optima limit its effectiveness [27]. Differential Evolution (DE), though robust in certain applications, struggles with global search efficiency when handling high-dimensional, complex problems [28]. Hybrid algorithms, despite their potential to improve optimization performance, are complex to implement and come with high computational costs, making them unsuitable for this study [29]. Considering these limitations, APSO was chosen for its superior global search capability, faster convergence, and computational efficiency, making it the most suitable choice for this application.

In each iteration of the algorithm, particles are assigned different fitness values to evaluate the convergence speed and stability of the model training process. The fitness function measures the reduction in error during each iteration, guiding the optimization process along the convergence path. Based on these fitness values, the weight of each particle is determined. Particles closer to the optimal solution are assigned smaller weights, resulting in smaller step sizes. Conversely, particles farther from the optimal solution are given larger weights, allowing them to take bigger steps in the search process. This adaptive weighting scheme helps guide particles toward regions of the search space that are more likely to contain the optimal solution.

$$w = \begin{cases} w_{\min} + (w_{\max} - w_{\min}) \dfrac{f - f_{\min}}{f_{\text{avg}} - f_{\min}}, & f \le f_{\text{avg}}, \\ w_{\max}, & f > f_{\text{avg}}, \end{cases} \tag{19}$$

where w represents inertia weight, $w_{\max}$ and $w_{\min}$ represent the maximum and minimum values of the inertia weight, respectively, set $w_{\min} = 0.4$, $w_{\max} = 0.9$, f represents the current fitness value, f_{avg} represents the average fitness value, $f_{\min}$ represents the minimum fitness value.

The learning factors C_1 and C_2 are critical in Particle Swarm Optimization (PSO), influencing the balance between individual experience and social experience, and reflecting the information exchange among particles. A larger C_1 encourages particles to focus on local search, while a larger C_2 can lead to premature convergence to local optima [30]. At the beginning of the search, we use a larger C_1 and a smaller C_2 to promote broad exploration of the search space, minimizing the influence of other particles and enhancing swarm diversity. As iterations progress, C_1 is gradually decreased while C_2 is increased linearly, improving the particles' convergence towards the global optimum. The formula for the dynamic adjustment of the learning factors is as follows:

$$C_1 = C_{1,\text{start}} + (c_{1,\text{end}} - C_{1,\text{start}}) \frac{t}{T}, \tag{20}$$

$$C_2 = C_{2,\text{start}} + (c_{2,\text{end}} - C_{2,\text{start}}) \frac{t}{T}, \tag{21}$$

where $C_{1,\text{start}}$ and $C_{1,\text{end}}$ represent the initial and termination values of individual learning factor C_1, respectively, and $C_{2,\text{start}}$ and $C_{2,\text{end}}$ represent the initial and termination values of the group learning factor C_2, respectively. This paper sets $C_{1,\text{start}} = 2.5$, $C_{1,\text{end}} = 0.5$, $C_{2,\text{start}} = 0.5$, $C_{2,\text{end}} = 2.5$.

The data collected from the flight test are first subjected to data preprocessing: First, an FIR bandpass filter with a

Hanning window was designed to address frequency leakage and enhance frequency selectivity. Next, a time vector was computed to ensure signal consistency. A second-order polynomial fitting was applied to remove trend components, reducing low-frequency interference. The signal was then centralized by subtracting the mean to eliminate zero offset. Bidirectional filtering was used to remove noise outside the target frequency bands, further cleaning the signal and eliminating phase distortion. The final output was a signal free of trends, zero offset, and frequency interference. After this, the data was normalized to eliminate the impact of differing units, allowing the model to better adapt during training. Upon completion of model training, the data was denormalized to restore it to its original scale, facilitating the interpretation and application of the results.

The identification samples were analyzed and extracted, resulting in six sets of data fragments. One set of data is used as the training set, and the other five sets of data are used as the test set for model training and modeling, respectively. predict. APSO algorithm is used to optimize the penalty coefficient parameter λ, kernel width σ, coefficient p and order q in the model. The flowchart is shown in Fig. 1.

Detailed steps are as follows:

Step 1: Initialize particle swarm parameters.

Step 2: Calculate the fitness value of each particle, and use the mean square error between the model prediction value and the actual value as the fitness value function [31]:

$$\text{fitness} = \frac{1}{m}\sum_{i=1}^{m}[y(x_i) - f(x_i)]^2, \tag{22}$$

where m represents the maximum size of the particle swarm, $y(x_i)$ represents the actual data value, and $f(x_i)$ represents the model prediction value.

Step 3: Find the best individual fitness value. Each particle fitness value is compared with its historical optimal position P_{id}, and the position is updated.

Step 4: Find the best fitness value of the group. Compare the fitness value of each particle with the historical optimal position P_{gd} of the group and update it.

Step 5: Calculate adaptive weight w, individual learning factor C_1 and group learning factor C_2 according to formula (19) and formula (20), (21).

Step 6: Update the velocity and position of each particle. The particle position and velocity are updated by the following two formulas [32]:

$$V_{id}^{n+1} = wV_{id}^{n} + C_1 r_1(P_{id} - X_{id}^{n}) + C_2 r_2(P_{gd} - X_{id}^{n}), \tag{23}$$

$$X_{id}^{n+1} = X_{id}^{n} + V_{id}^{n+1}, \tag{24}$$

where r_1 and r_2 represent numbers in the range (0,1) randomly generated by the rand function, X_{id}^{n} and V_{id}^{n} represent the current position and velocity of the particle, respectively, and X_{id}^{n+1} and V_{id}^{n+1} are the updated position and velocity.

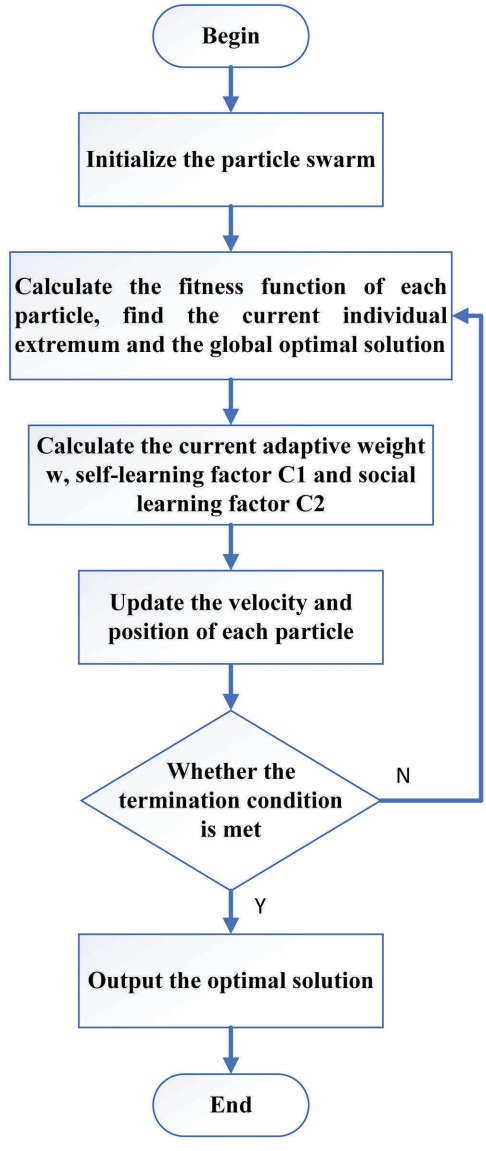


Fig. 1. APSO algorithm flowchart.

Step 7: Determine whether the termination condition is met. and output the optimal parameter if the termination condition is met, otherwise skip to Step 2.

Comparing the optimization effect of the APSO algorithm with the traditional PSO algorithm, the fitness curve comparison is shown in Figs. 2 and 3.

From the comparison of the fitness curves of the lateral channel and the longitudinal channel in Figs. 2 and 3, it can be seen that the APSO algorithm can significantly improve the model training speed and find the optimal parameter value faster. The optimized model parameter values at this time are shown in Table 1.

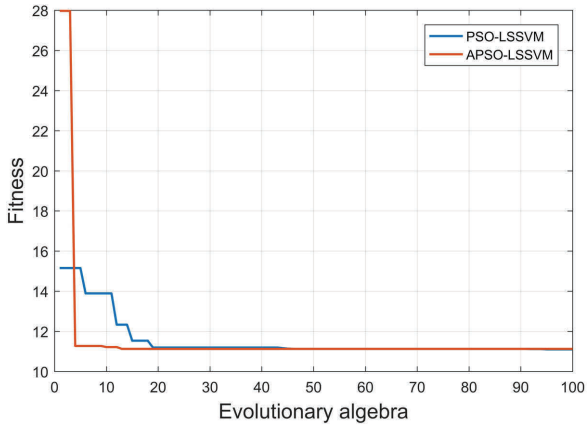


Fig. 2. Comparison of lateral channel fitness curves.

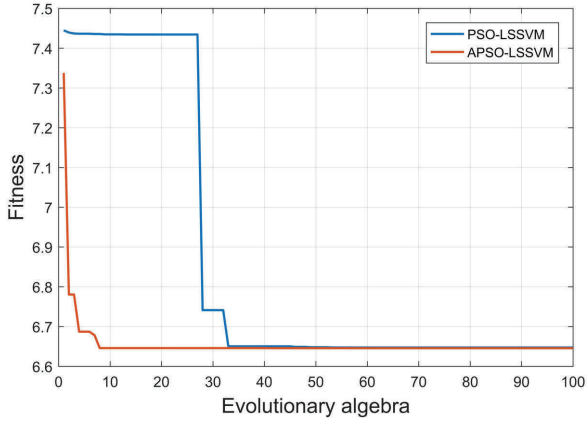


Fig. 3. Comparison of longitudinal channel fitness curves.

Table 1. Model parameter values.

Parameter	λ	σ	p	q
Lateral channel	0.0496	54.3144	4.8635	2.7455
Longitudinal channel	231.3614	907.3211	2.0468	1.3910

4. Model Validation and Analysis

Using the optimized parameters, an APSO-LSSVM model based on the custom kernel function was developed. This model was compared with two other models: the LSSVM model using a single RBF kernel and the LSSVM model that combines RBF and polynomial kernels in a linear weighted fashion, as employed in previous research. Test samples were used to evaluate the predictive accuracy of the three models, and the results are shown in Figs. 4–9.

Figure 4 shows the single-sided excitation PWM input values for the lateral channel, with negative PWM values in Figs. 4 and 7 resulting from data preprocessing, primarily due to detrending and centralization. Despite the preprocessing, the physical meaning of the PWM signals remains unchanged, and they are still within the normal range in the actual system. Figure 5 compares the prediction performance of three models for the lateral channel: the single-kernel LSSVM model, the hybrid-kernel LSSVM model, and the self-constructed Kernel LSSVM model. Figure 6 illustrates the prediction error curves for these models. From Figs. 5 and 6, it is clear that under single-sided excitation PWM input, the self-constructed Kernel LSSVM model achieves a closer fit between the predicted output and the flight test output compared to the single-kernel and hybrid-kernel models. Additionally, the self-constructed Kernel model exhibits significantly smaller prediction errors.

Figure 7 presents the single-sided excitation PWM input values for the longitudinal channel, while Fig. 8 compares the prediction performance of the single-kernel LSSVM, hybrid-kernel LSSVM, and self-constructed Kernel LSSVM models. Figure 9 illustrates the prediction error curves for the three models. From Figs. 8 and 9, it is clear that under single-sided excitation PWM input, the self-constructed Kernel LSSVM model achieves a closer fit between the predicted output and the flight test output compared to the single-kernel and hybrid-kernel models. Additionally, the self-constructed Kernel model exhibits significantly smaller prediction errors.

Furthermore, to validate the models, five additional validation samples were tested using the single-kernel, hybrid-kernel, and self-constructed Kernel LSSVM models. The model performance was evaluated using Mean Squared Error (MSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). The corresponding expressions are as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2, \tag{25}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i'|, \tag{26}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - y_i')^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{27}$$

where y_i represents the real data value output by the flight experiment, y_i' represents the model predicted data value, and $\bar{y}$ represents the average value of the output data value of the flight experiment. The smaller the MSE and MAE, the better the model, and the closer R^2 is to 1, the more accurate the model prediction is. The model prediction comparisons for the lateral channel and longitudinal channel are shown in Tables 2 and 3, respectively.

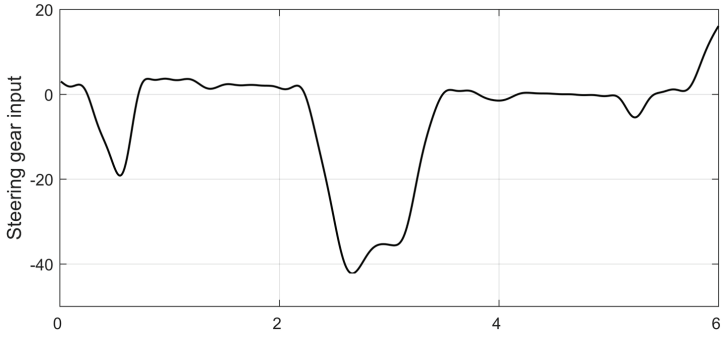


Fig. 4. Lateral channel input.

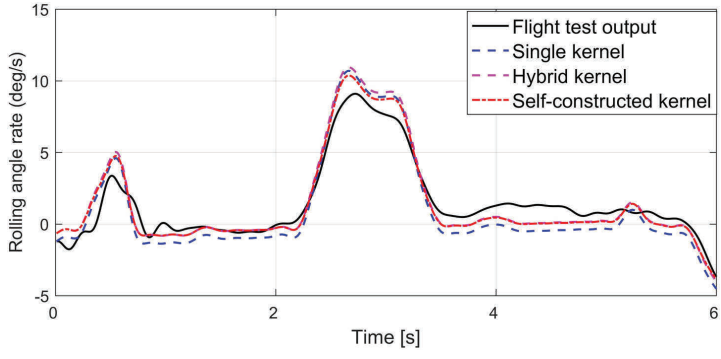


Fig. 5. Comparison of lateral channel identification effects.

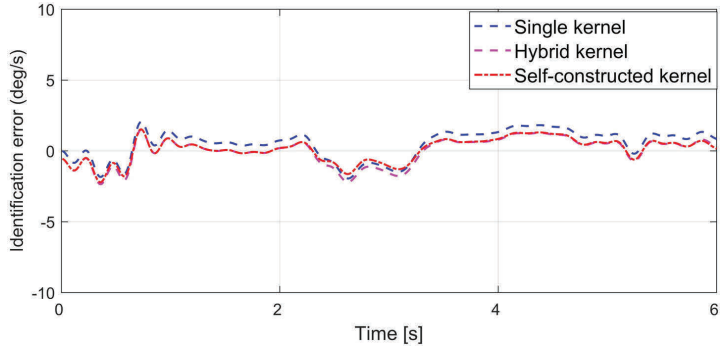


Fig. 6. Comparison of lateral channel model prediction errors.

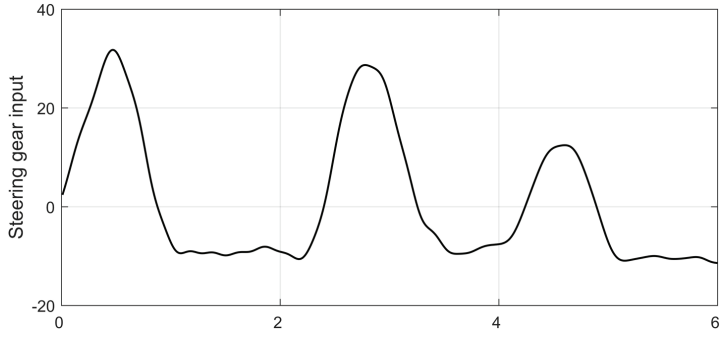


Fig. 7. Longitudinal channel input.

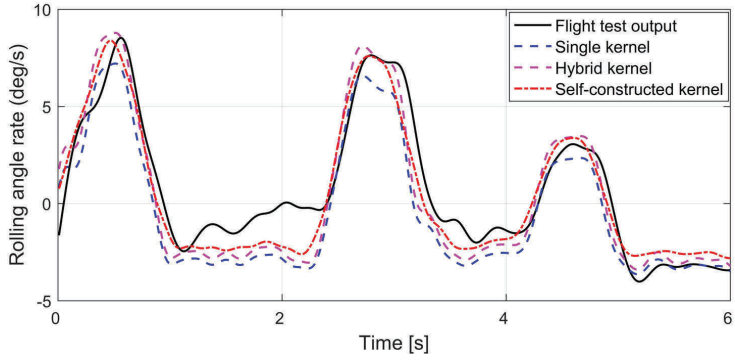


Fig. 8. Comparison of longitudinal channel identification effects.

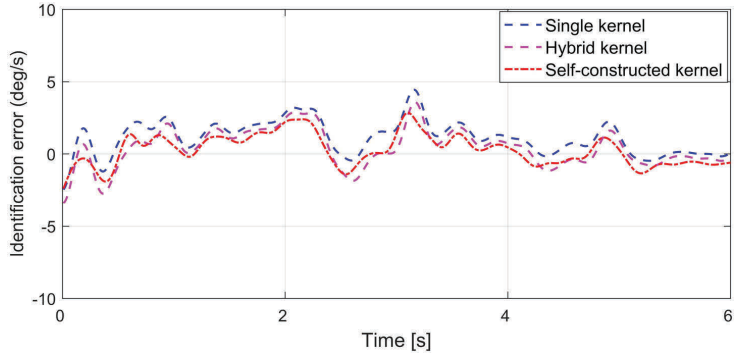


Fig. 9. Comparison of longitudinal channel model prediction errors.

Table 2. Comparative analysis of lateral channel model prediction.

No.	Method	MSE	MAE	R^2
1	Single kernel	5.8634	1.9718	0.8648
	Hybrid kernel	4.3856	1.7170	0.8935
	Self-constructed Kernel	4.2253	1.6277	0.9026
2	Single kernel	33.4178	4.0139	0.8777
	Hybrid kernel	25.3732	3.7064	0.9045
	Self-constructed Kernel	23.4857	3.5183	0.9140
3	Single kernel	1.3307	1.0277	0.9146
	Hybrid kernel	0.8546	0.9259	0.9416
	Self-constructed Kernel	0.7602	0.7343	0.9512
4	Single kernel	2.8919	1.3134	0.8644
	Hybrid kernel	2.2399	1.1757	0.8909
	Self-constructed Kernel	1.9893	1.0891	0.9067
5	Single kernel	2.0907	1.1840	0.8745
	Hybrid kernel	1.5878	0.9807	0.9127
	Self-constructed Kernel	1.3085	0.8151	0.9215

Table 3. Comparative analysis of longitudinal channel model prediction.

No.	Method	MSE	MAE	R^2
1	Single kernel	3.5334	1.6406	0.8367
	Hybrid kernel	2.6427	1.3141	0.8778
	Self-Constructed Kernel	2.2023	1.1966	0.8982
2	Single kernel	1.5973	1.0387	0.8601
	Hybrid kernel	1.1516	0.8230	0.8991
	Self-Constructed Kernel	0.8220	0.7330	0.9280
3	Single kernel	9.9811	2.4576	0.8771
	Hybrid kernel	8.4952	2.2365	0.8937
	Self-Constructed Kernel	7.6514	1.9940	0.9058
4	Single kernel	5.489	1.8316	0.9463
	Hybrid kernel	3.3205	1.4436	0.9675
	Self-Constructed Kernel	2.3156	1.2273	0.9773
5	Single kernel	1.9651	1.1902	0.8394
	Hybrid kernel	0.9178	0.8294	0.9313
	Self-Constructed Kernel	0.7276	0.6849	0.9405

Table 2 compares the MSE, MAE, and R^2 values for models built using three different kernel functions in the lateral channel. The Self-Constructed Kernel LSSVM model demonstrated significant improvements: MSE decreased by 30.32% compared to the single-kernel model and by 7.75% compared to the hybrid-kernel model. Similarly, the MAE decreased by 18.15% and 7.61% compared to the single-kernel and hybrid-kernel models, respectively. The R^2 value increased by 4.55% compared to the single-kernel model and by 1.16% compared to the hybrid-kernel model.

Table 3 compares the MSE, MAE, and R^2 values for the LSSVM models built using three different kernel functions in the longitudinal channel. The Self-Constructed Kernel LSSVM model showed notable improvements: MSE decreased by 39.21% compared to the single-kernel model and by 16.99% compared to the hybrid-kernel model. Similarly, MAE decreased by 28.47% compared to the single-kernel model and by 12.19% compared to the hybrid-kernel model. The R^2 value increased by 6.66% compared to the single-kernel model and by 1.76% compared to the hybrid-kernel model.

Based on the evaluation metrics in Tables 2 and 3, the following conclusions can be drawn: In both the lateral and longitudinal channels, the Self-Constructed Kernel LSSVM model demonstrated significantly higher predictive accuracy compared to the single-kernel and hybrid-kernel LSSVM models.

5. Conclusion

This paper addresses the modeling challenges of SUHs and proposes an LSSVM modeling method based on a custom kernel function, optimized with an APSO algorithm.

(1) A new custom kernel function, built on Mercer's theorem, combines the strong local search capabilities of the RBF kernel with the global search strengths of the polynomial kernel. Experiments demonstrated that models using this custom kernel show excellent learning and generalization performance.
(2) To mitigate the tendency of standard Particle Swarm Optimization (PSO) to get trapped in local optima, an improved PSO algorithm with adaptive inertia weight and linearly varying learning factors was introduced. This approach (APSO) significantly improved model training speed and addressed the limitations of traditional PSO.
(3) Furthermore, flight test results demonstrated that the APSO-LSSVM model with the custom kernel significantly improved predictive accuracy and accelerated parameter optimization, surpassing LSSVM models using single RBF kernels and RBF-Poly hybrid kernels. This approach offers valuable insights into enhancing the identification efficiency of SUH models and contributes to the development of high-precision models and autonomous flight control systems.

Acknowledgment

This work was supported by the China Scholarship Council.

ORCID

Jian Zhou https://orcid.org/0000-0002-1436-8119
Junyi Shi https://orcid.org/0009-0004-6346-9081
Weixin Wang https://orcid.org/0000-0001-9707-4572
Jian Lu https://orcid.org/0000-0003-0306-5101

References

[1] X. Lu, Z. Xiheng, M. Xianfeng, W. Heqiang, Z. Yang and C. Yang, ADRC based attitude control system design for unmanned helicopter, in *3rd Int. Conf. Unmanned Systems*, Harbin, China, 2020, pp. 309–313, doi: 10.1109/ICUS50048.2020.9274848.
[2] S. Tang, Z. Zheng, S. Qian and X. Zhao, Nonlinear system identification of a small-scale unmanned helicopter, *Control Eng. Pract.* **25** (2014) 1–15.
[3] W. Zhou, S. Chen, C.-W. Chang, C.-Y. Wen, C.-K. Chen and B. Li, System identification and control for a tail-sitter unmanned aerial vehicle in the cruise flight, *IEEE Access* **8** (2020) 218348–218359, doi: 10.1109/ACCESS.2020.3042316.
[4] M. H. Khalesi, H. Salarieh and M. S. Foumani, System identification and robust attitude control of an unmanned helicopter using novel low-cost flight control system, *Proc. Inst. Mech. Eng. I: J. Syst. Control Eng.* **234**(5) (2020) 634–645, doi: 10.1177/0959651819869718.
[5] T. Stachiw, J. Ricciardi and A. Crain, Combined time- and frequency-domain aircraft system identification using Pareto optimization, in *ASME 2021 Int. Mechanical Engineering Congress and Exposition*, 2021, doi: 10.1115/IMECE2021-68541.
[6] T. C. R. Theodore, J. D. Colbourne and M. B. Tischler, Rapid frequency-domain modeling methods for unmanned aerial vehicle flight control applications, *J. Aircraft* **41**(4) (2004) 735–743.
[7] B. Mettler, M. B. Tischler and T. Kanade, System identification modeling of a small-scale unmanned rotorcraft for flight control design, *J. Amer. Helicopter Soci.* **47**(1) (2002) 50–63.
[8] I. A. Raptis, K. P. Valavanis and W. A. Moreno, System identification and discrete nonlinear control of miniature helicopters using backstepping, *J. Intelli. Robot. Syst.* **55** (2009) 223–243, doi: 10.1007/s10846-008-9295-5.
[9] S. Setu, A. Abhishek and C. Venkatesan, Time-domain system identification of helicopters using nonlinear acceleration and jerk prediction model, *J. Aircraft* **56**(3) (2019) 1–9, doi: 10.2514/1.C035273.
[10] G. Tang, J. Ni, Y. Zhao and Y. Gu, A survey of object detection for UAVs based on deep learning, *Remote Sens.* **16**(1) (2024) 149, doi: 10.3390/rs16010149.
[11] F. Hu, Q. Wang, H. Shao, S. Gao and H. Yu, Anomaly detection of UAV state data based on single-class triangular global alignment kernel extreme learning machine, *Comput. Model. Eng. Sci.* **136**(3) (2023) 2405–2424, doi: 10.32604/cmes.2023.026732.

[12] J. Quiñonero-Candela and C. E. Rasmussen, A unifying view of sparse approximate Gaussian process regression, *J. Mach. Learn. Res.* **6** (2005) 1939–1959.
[13] J. Suykens and J. Vandewalle, Least squares support vector machine classifiers, *Neural Process. Lett.* **9** (1999) 293–300.
[14] Z. Xuegong, Introduction to statistical learning theory and support vector machines, *Acta Automatica Sinica* **26**(1) (2000) 32–42.
[15] L. Xiang, Z. Deng and A. Hu, Forecasting short-term wind speed based on IEWT-LSSVM model optimized by bird swarm algorithm, *IEEE Access* **7** (2019) 59333–59345, doi: 10.1109/ACCESS.2019.2914251.
[16] K. Li, G. Cheng, X. Sun and Z. Yang, A nonlinear flux linkage model for bearingless induction motor based on GWO-LSSVM, *IEEE Access* **7** (2019) 36558–36567, doi: 10.1109/ACCESS.2019.2905247.
[17] Z. Tian, Short-term wind speed prediction based on LMD and improved FA optimized combined kernel function LSSVM, *Eng. Appl. Artifi. Intelli.* **91** (2020) 103573, doi: 10.1016/j.engappai.2020.103573.
[18] W. L. Ma and H. Liu, Least squares support vector machine regression based on sparse samples and mixture kernel learning, *Inform. Technol. Control* **50**(2) (2021) 319–331.
[19] E. Guo *et al.*, Study on fault identification of mechanical dynamic nonlinear transmission system, *Nonlinear Eng.* **10**(1) (2022) 518–525, doi: 10.1515/nleng-2021-0042.
[20] M. Pinheiro Jr. and P. O. Dral, Kernel methods, in *Quantum Chemistry in the Age of Machine Learning*, (Elsevier, 2023), pp. 205–232, doi: 10.1016/B978-0-323-90049-2.00009-3.
[21] C. M. Bishop, *Pattern Recognition and Machine Learning* (Microsoft Research Cambridge, Cambridge, United Kingdom, 2006).
[22] M. Cai *et al.*, Integrating the LSSVM and RBFNN models with three optimization algorithms to predict the soil liquefaction potential, *Eng. Comput.* **38** (2022) 3611–3623, doi: 10.1007/s00366-021-01392-w.
[23] G. Ke *et al.*, Network traffic prediction based on least squares support vector machine with simple estimation of Gaussian kernel width, *Int. J. Inform. Comput. Secur.* **18**(1–2) (2022) 1–11, doi: 10.1504/IJICS.2022.122910.
[24] A. Karale, M. Lazarova, P. Koleva and V. Poulkov, A hybrid PSO-MiLOF approach for outlier detection in streaming data, in *43rd Int. Conf. Telecommunications and Signal Processing*, Milan, Italy, 2020, pp. 474–479, doi: 10.1109/TSP49548.2020.9163430.
[25] Q. Bian *et al.*, System identification method for small unmanned helicopter based on improved particle swarm optimization, *J. Bionic Eng.* **13** (2016) 504–514, doi: 10.1016/S1672-6529(16)60323-2.
[26] C. Huang, Y. Zhao, M. Zhang and H. Yang, APSO: An A*-PSO hybrid algorithm for mobile robot path planning, *IEEE Access* **11** (2023) 43238–43256, doi: 10.1109/ACCESS.2023.3272223.
[27] S. Dunn, S. Peucker and J. Perry, Genetic algorithm optimisation of mathematical models using distributed computing, *Appl. Intell.* **23** (2005) 21–32, doi: 10.1007/s10489-005-2369-1.
[28] R. Storn and K. Price, Differential evolution A simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.* **11**(4) (1997) 341–359, doi: 10.1023/A:1008202821328.
[29] R. P. Parouha and P. Verma, Design and applications of an advanced hybrid metaheuristic algorithm for optimization problems, *Artif. Intell. Rev.* **54**(8) (2021) 5931–6010, doi: 10.1007/s10462-021-09962-6.
[30] R. Eberhart and Y. Shi, Particle swarm optimization: Developments, applications and resources, in *Proc. Congress on Evolutionary Computation*, Vol. 1, Seoul, South Korea, 2001, pp. 81–86, doi: 10.1109/CEC.2001.934374.
[31] K. M. Hamdia, X. Zhuang and T. Rabczuk, An efficient optimization approach for designing machine learning models based on genetic algorithm, *Neural Comput. Appl.* **33** (2021) 1923–1933, doi: 10.1007/s00521-020-05035-x.
[32] T. Lawrence, L. Zhang, C. P. Lim and E.-J. Phillips, Particle swarm optimization for automatically evolving convolutional neural networks for image classification, *IEEE Access* **9** (2021) 14369–14386, doi: 10.1109/ACCESS.2021.3052489.

Jian Zhou received the B.S. and M.S. degrees from Northwestern Polytechnical University, Xi'an, China, in 2005 and 2009, respectively. He received the Ph.D. degree in Control Engineering from Northwestern Polytechnical University, Xi'an, China, in 2013. Since SEP 2013, he has been with Xi'an Polytechnic University, where he is currently a Lecturer in School of Electronics and Information. His research interests include robot modeling and control, mobile robot navigation and application of control theory.

Junyi Shi is a master's candidate at Xi'an Polytechnic University, focusing on frequency domain system identification of small unmanned helicopters. Her research aims to improve UAV control systems by analyzing dynamic behavior and identifying system parameters in the frequency domain.

Weixin Wang master's candidate at Xi'an Polytechnic University, his research interest covers mathematical modeling for small unmanned helicopter.

Jian Lu received the M.Sc. degree in Control Science and Engineering from the Xi'an Jiaotong University, Xi'an, China, in 2007, and the Ph.D. degree in Weapon Science and Technology from Northwestern Polytechnical University, Xi'an, in 2015. He is currently an Associate Professor at the School of Electronics and Information, Xi'an Polytechnic University, Xi'an, China. His main research interests include underwater robot location, cooperative localization, person re-identification, and small target detection.