DOI: https://doi.org/10.36910/6775-2524-0560-2026-63-37
УДК 004.932:004.8:629.7
Sadovnykov Borys, PhD in Engineering
https://orcid.org/0009-0009-4180-2863
Syvolovskyi Illia, PhD in Engineering
https://orcid.org/0000-0002-4592-0965
National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine.

REAL-TIME OBJECT DETECTION AND TRACKING METHOD FOR FLIGHT CONTROL SYSTEMS

Sadovnykov B., Syvolovskyi I. Real-time object detection and tracking method for flight control systems. The paper presents a method for object detection and tracking in video streams, oriented toward application in onboard flight control systems. The proposed approach is based on inter-frame difference analysis and is enhanced with coordinated processing mechanisms, which ensure robustness to disturbances, dynamic scene changes, and limited computational resources. The method takes into account the specific operating conditions of onboard systems, including illumination variations, platform motion, complex background dynamics, partial object occlusion, and real-time processing requirements. The algorithm is formalized as a sequence of interconnected processing stages, including preprocessing, motion compensation, inter-frame analysis, segmentation, morphological filtering, object classification, tracking, and integration of the results into the flight control loop. The developed algorithm provides coordinated parameter tuning, employs a neural network-based classifier adapted to resource-constrained environments, and incorporates tracking stabilization mechanisms under conditions of partial object loss. An analytical assessment of computational complexity and an experimental evaluation of algorithm performance have been conducted. It is substantiated that the proposed method ensures an effective balance between accuracy, robustness to disturbances, and computational efficiency under dynamically changing conditions.
Keywords: method; object detection and tracking; video streams; onboard systems; flight control systems; processing; computational complexity; real-time; robustness.

Садовников Б.І., Сиволовський І.М. Метод виявлення та відстеження об'єктів у реальному часі для систем управління літальними апаратами. У статті представлено метод виявлення та супроводу об'єктів у відеопотоках, орієнтований на застосування в бортових системах управління літальними апаратами. Запропонований підхід будується на аналізі міжкадрових змін та доповнений узгодженими механізмами обробки, що дозволяють забезпечити стійкість до завад, динамічних змін сцени та обмежених обчислювальних ресурсів. Метод враховує специфіку функціонування бортових систем, зокрема зміну освітлення, рух платформи, складну динаміку фону, часткову оклюзію об'єктів та необхідність роботи в режимі реального часу. Формалізацію алгоритму виконано як послідовність взаємопов'язаних етапів, що включають попередню обробку, компенсацію руху, міжкадровий аналіз, сегментацію, морфологічну фільтрацію, класифікацію об'єктів, їх супровід та інтеграцію результатів у контур управління. Розроблений алгоритм передбачає узгоджене налаштування параметрів обробки, використання нейромережевого класифікатора з урахуванням ресурсних обмежень, а також механізми стабілізації супроводу в умовах часткової втрати об'єкта. Проведено аналітичну оцінку обчислювальної складності та експериментальне дослідження продуктивності алгоритму. Обґрунтовано, що запропонований метод забезпечує ефективний баланс між точністю, стійкістю до завад та обчислювальною ефективністю в умовах змінного середовища.
Ключові слова: метод; виявлення та супровід об'єктів; відеопотоки; бортові системи; системи управління літальними апаратами; обробка; обчислювальна складність; реальний час; завадостійкість.

Statement of a scientific problem.

In onboard flight control systems, efficient processing of video streams is significantly complicated by strict constraints on computational resources, energy consumption, and real-time requirements. Such systems operate under dynamically changing conditions, including platform motion, illumination variability, background complexity, and the presence of disturbances, which directly affect the stability of object detection and tracking processes. At the same time, the need for timely decision-making imposes additional requirements on the balance between computational efficiency, robustness, and processing accuracy.

To address this scientific and practical problem, taking into account modern approaches to computer vision, real-time video processing, and neural network-based classification [1–19], a method for object detection and tracking in video streams is proposed, which involves:

– detection of dynamic objects based on inter-frame difference analysis;

– application of adaptive preprocessing, segmentation, and morphological filtering mechanisms to improve robustness to disturbances;

– classification of detected regions using neural network models adapted to resource-constrained environments;

– integration of detection and tracking results into the control loop of the flight system.

The proposed method is characterized by computational efficiency, adaptability to changing environmental conditions, and suitability for real-time onboard implementation. It ensures stable performance under varying scene complexity and limited computational resources, which makes it applicable to modern flight control systems.

Research analysis.

An analysis of recent research in the field of object detection and tracking in video streams shows that a wide range of approaches has been developed, including classical background subtraction methods, inter-frame difference techniques, and modern deep learning-based models.

The works [1, 3, 5] focus on methods based on inter-frame difference analysis and their mathematical formalization, demonstrating their computational efficiency for real-time applications. Studies [4, 6–8] consider classical and hybrid approaches to object detection, including motion-based and feature-based methods; however, their robustness under dynamic conditions remains limited.

Publications [9, 11, 12] analyze traditional and modern background modeling techniques, while works [13, 14, 16] investigate deep learning approaches, particularly YOLO- and SSD-based models, which provide high detection accuracy but are often computationally expensive for onboard systems.

Research [15, 17–19] is devoted to lightweight neural network architectures and their adaptation for resource-constrained environments. Although these approaches improve computational efficiency, they do not fully address the problem of integrating detection and tracking into real-time control loops under dynamic operating conditions.

Thus, despite the significant progress in this field, the problem of developing computationally efficient and adaptive methods for object detection and tracking in video streams for onboard flight systems remains relevant and insufficiently addressed.

The purpose of the work.

The aim of this study is to develop a method for object detection and tracking in video streams for onboard flight control systems, based on inter-frame difference analysis and adaptive processing mechanisms, ensuring a balance between computational efficiency, robustness to disturbances, and real-time performance under varying operating conditions.

Presentation of the main material and substantiation of the obtained research results.

In scientific and practical tasks of flight control systems, video stream processing is performed under strict constraints on computational resources, energy consumption, and real-time requirements. In this regard, it is important to evaluate the computational load of object detection and tracking algorithms in order to determine their suitability for onboard implementation.

To address this, an object detection and tracking method is proposed, the key feature of which is the combination of computationally efficient inter-frame difference analysis with adaptive processing mechanisms and its integration into the flight control loop. This enables robustness to disturbances, dynamic scene changes, and limited resources while maintaining real-time performance. The algorithm implementing the proposed method is presented in the form of a block diagram in Fig. 1.

The considered object detection and tracking method, based on inter-frame difference analysis, is implemented as a sequence of interconnected processing stages, each of which has a different impact on the overall performance of the onboard system. The main stages include image preprocessing, platform motion compensation, inter-frame analysis, segmentation, morphological filtering, object classification, temporal tracking, as well as target state estimation and control command generation.

Taking into account the constraints of computational resources and real-time operation requirements, it is important to perform a formalized evaluation of the computational load of each stage of the algorithm. This makes it possible to identify the most resource-intensive components and justify the feasibility of its practical implementation in onboard flight control systems.

For the analytical evaluation of the algorithm complexity, the following assumptions are adopted, corresponding to typical operating conditions of onboard video processing systems:

1. The input image has a fixed size of $W \times H$.
2. The number of detected (segmented) regions after morphological processing is denoted as K, and in typical scenarios its value ranges from 5 to 15.
3. For each detected region, classification and tracking are performed.
4. The size of the structuring element for morphological operations is fixed and equal to $k = 5$.

5. The depth of the neural network classifier is assumed to be constant, for example $d = 200$ (MobileNetV1).

6. The average area of a detected region after filtering is approximately $A_r \approx 4000$ pixels.

7. Processing is performed over a sequence of frames in real time, implying temporal dependency between consecutive frames.

8. Platform motion compensation is performed for each frame and has computational complexity proportional to the image size.

9. Object tracking is performed for each confirmed region and has complexity proportional to the number of tracked objects.

10. Target state estimation and control command generation are performed for each frame and have constant or linear complexity with respect to the number of objects.

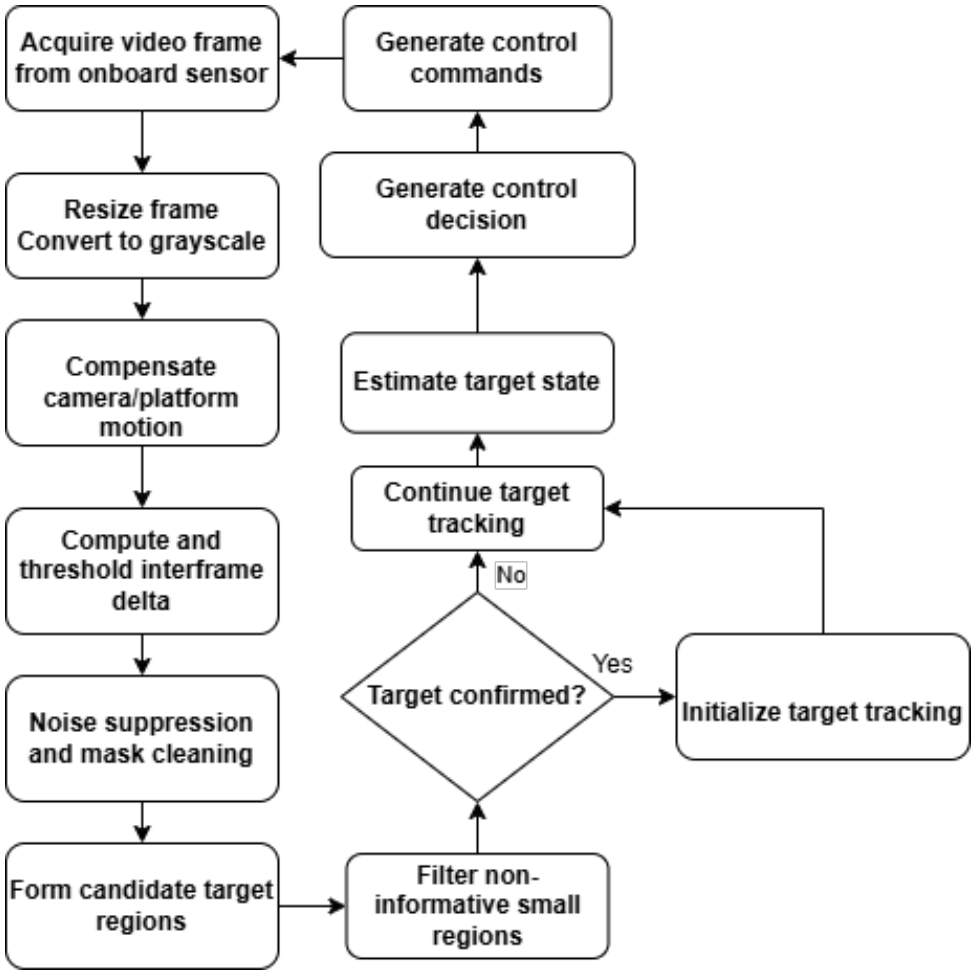

Fig.1 – Block diagram of the object detection and tracking algorithm integrated into an onboard flight control system

Taking into account the adopted assumptions, the overall computational complexity of the algorithm for processing a single frame is determined by the combination of operations performed over the entire image, as well as operations related to the processing of detected regions. The first group includes preprocessing, platform motion compensation, inter-frame analysis, segmentation, and morphological filtering, while the second group includes object classification, tracking, and state estimation.

In the typical case, the overall computational complexity of the algorithm can be expressed as:

$$\mathcal{O}(W \cdot H) + \mathcal{O}(K \cdot D) + \mathcal{O}(K), \quad (1)$$

where $\mathcal{O}(W \cdot H)$ corresponds to operations performed over the entire frame; $\mathcal{O}(K \cdot D)$ represents computations related to the classification of detected objects using a neural network model; $\mathcal{O}(K)$ corresponds to object tracking and state estimation operations.

Under more complex scene conditions, such as the presence of noise, illumination variations, or increased background dynamics, the computational load increases due to the higher complexity of spatial filtering and morphological processing. In this case, the algorithm complexity can be refined as:

$$O(W \cdot H \cdot (1 + s^2 + k^2)) + O(K \cdot D) + O(K), \quad (2)$$

where s is the spatial neighborhood radius and k is the size of the structuring element used in morphological operations.

In the worst-case scenario, corresponding to scenes with a large number of objects or complex structures, the algorithm complexity is determined by the maximum number of processed regions:

$$O(W \cdot H \cdot (1 + s^2 + k^2)) + O(K_{max} \cdot D) + O(K_{max}), \quad (3)$$

where $K_{\max}$ is the maximum number of objects in a frame.

The analytical estimates obtained from experimental modeling make it possible to identify the main components of the computational load and determine the stages that have the greatest impact on the overall performance of the algorithm. For a more detailed analysis, it is reasonable to consider the contribution of each processing stage separately, which is presented as a generalized complexity assessment in Table 1.

Table 1 – Computational complexity assessment of the object detection and tracking algorithm stages in an onboard system

Processing stage	Computational complexity	Description
Preprocessing (resizing, grayscale conversion)	$O(W \cdot H)$	Preparation of the frame for further processing
Platform motion compensation	$O(W \cdot H)$	Compensation of camera motion and scene stabilization
Inter-frame analysis (delta)	$O(W \cdot H)$	Detection of dynamic changes in the scene
Threshold segmentation	$O(W \cdot H)$	Extraction of candidate regions
Spatial filtering	$O(W \cdot H \cdot s^2)$	Noise suppression
Morphological processing	$O(W \cdot H \cdot k^2)$	Mask refinement and object formation
Region formation and filtering	$O(K)$	Selection of informative objects
Object classification	$O(K \cdot D)$	Object recognition using a neural network
Object tracking	$O(K)$	Updating object positions over time
Target state estimation	$O(K)$	Estimation of object motion parameters
Control command generation	$O(K)$	Generation of control inputs for the system

As shown in Table 1, the computational load of the algorithm is unevenly distributed across the processing stages. The largest contribution to the overall complexity is made by the stages of morphological processing and object classification, which is due to their dependence on the size of the structuring element and the depth of the neural network model, respectively. In particular, morphological operations exhibit a quadratic dependence on the parameter k, while classification scales proportionally with the number of objects K and the model complexity D.

At the same time, the preprocessing, inter-frame analysis, and threshold segmentation stages are characterized by a linear dependence on the frame size and have a relatively smaller impact on overall performance. Object tracking, state estimation, and control command generation also demonstrate linear dependence on the number of objects and do not impose critical computational load under typical operating conditions.

Thus, the results presented in Table 1 indicate that the primary performance limitations of the algorithm in onboard flight control systems are determined by the morphological processing and object classification stages. This highlights the need to focus optimization efforts on these components, for example, by reducing the size of structuring elements, applying more efficient filtering techniques, or using optimized neural network models.

For a more intuitive representation of the computational load distribution across the algorithm stages, a heatmap is used, which allows visualization of the intensity of computational resource utilization depending on the type of operations and processing characteristics (Fig. 2).

Fig. 2 presents a heatmap illustrating the distribution of computational load across the main stages of the algorithm in terms of computational time (Time), memory usage (Memory), and processing resource utilization (GPU/CPU). The horizontal axis represents these performance indicators, while the vertical axis corresponds to the processing stages, including preprocessing, motion compensation, inter-frame analysis, threshold segmentation, spatial filtering, morphological processing, region filtering, classification, tracking, state estimation, and control.

Stage	Time	Memory	GPU/CPU
Preprocessing	2	1	1
Motion compensation	2	2	2
Inter-frame analysis	2	1	1
Threshold segmentation	1	1	1
Spatial filtering	2	2	2
Morphological processing	4	3	2
Region filtering	1	1	1
Classification	5	4	5
Tracking	3	2	2
State estimation	1	1	1
Control	1	1	1

Fig.2 – Heatmap of computational load distribution across algorithm processing stages

The color scale reflects the intensity of resource utilization: green indicates low load, yellow corresponds to moderate load, orange represents increased load, and red denotes high computational cost.

As can be seen from the heatmap, preprocessing and threshold segmentation are characterized by low computational load due to their linear complexity. Motion compensation and spatial filtering demonstrate moderate resource utilization, as they involve operations over the entire image.

The most computationally intensive stages are morphological processing and object classification, as indicated by the predominance of orange and red colors. This is explained by the quadratic dependence of morphological operations on the structuring element size and the high complexity of neural network-based classification.

The tracking stage exhibits moderate load, while state estimation and control stages have minimal computational impact, which makes them suitable for real-time onboard implementation.

Thus, the heatmap visually confirms the analytical and experimental results and allows identifying the critical stages that determine the overall system performance, particularly morphological processing and classification.

To validate the analytical assessment of computational complexity, experimental profiling of the execution time for each stage of the proposed method was performed (Table 2). The timing results were obtained on a computing platform based on an Intel Core i7 processor with 32 GB of RAM and Nvidia 4080 GPU acceleration for video frames with a resolution of 1280×720.

The estimation of the number of operations (MOPs – millions of operations) was carried out analytically in accordance with the theoretical complexity of each stage of the algorithm, taking into account typical scene parameters relevant to onboard video processing tasks. In particular, the following values were assumed: the number of regions K = 8, the size of the structuring element for morphological operations k =5, the depth of the neural network classifier d=200, the spatial neighborhood radius s=2, the average region area Ar ≈4000 pixels, and the tracking window size w_t, which corresponds to the size of the detected object.

As shown in Table 2, the highest computational load is associated with the stages of morphological processing (31,52%) and object classification (36,14%), which together account for more than 69,63% of the total frame processing time. This confirms the analytical complexity assessment and is consistent with the results presented in the heatmap.

Table 2 – Time and computational characteristics of video processing algorithm stages

Processing stage	Average	Share of total	Resource	Number of operations

	execution time, ms	time, %	type	(estimate)
Preprocessing (resizing, grayscale conversion)	0,42	7,78	CPU	~0,903 MOPs
Platform motion compensation	0,38	7,04	CPU	~1,204 MOPs
Inter-frame analysis (delta)	0,31	5,74	CPU	~0,902 MOPs
Threshold segmentation	0,11	2,04	CPU	~3,00 MOPs (s=2)
Spatial filtering	0,28	5,19	CPU	~4,55 MOPs (s=2)
Morphological processing	1,68	31,11	CPU	~23,05 MOPs (k=5)
Region formation and filtering	0,22	4,07	CPU	~0,23 MOPs (K=8)
Object classification	2,08	38,52	GPU/CPU	~1,61 MOPs (K=8, d=200)
Object tracking	0,69	12,78	CPU	~0,62 MOPs (K=8)
Target state estimation	0,17	3,15	CPU	~0,11 MOPs
Control command generation	0,08	1,48	CPU	~0,05 MOPs
Total	**5,42**	**100,00**	—	—

The preprocessing, inter-frame analysis, and threshold segmentation stages have a relatively minor impact on overall performance and together account for less than 15% of the execution time. At the same time, the tracking stage demonstrates a moderate computational load (12,78%) and ensures stable object tracking without significantly affecting system performance.

Thus, the results of the experimental profiling confirm that the computational complexity of the method is primarily determined by morphological processing and neural network-based classification. This should be taken into account when optimizing the algorithm for implementation in onboard flight control systems.

To evaluate the scalability of the proposed method, an analysis of the computational load variation was performed by changing the key parameters of input data and algorithm components. The study was conducted for three representative scenarios, the results of which are summarized in Table 3:

1. Increasing the frame resolution up to 1920×1080.
2. Increasing the number of objects K up to 30, corresponding to dynamic scenes with high object density.
3. Using more complex neural network classifiers (MobileNetV2, EfficientNetB0).

Table 3 – Scalability analysis of computational load under varying scene and algorithm parameters

Scene/Algorithm parameters	Total time, ms	Frame-level processing, ms	Classification, ms
Baseline (1280×720, K=8, d=200)	5,42	3,02	2,40
FullHD (1920×1080)	9,36	6,95	2,41
K = 30 (dynamic scene)	9,18	3,15	6,03
MobileNetV2 (d = 300)	6,88	3,02	3,86
EfficientNetB0 (d = 400)	7,34	3,02	4,32

In addition, an experimental analysis of the impact of frame resolution on algorithm performance was conducted, which is critical for onboard systems operating under varying computational constraints. The results are presented in Table 4.

Table 4 – Scalability of the proposed method with respect to frame resolution

Frame resolution	Number of pixels	Average processing time, ms	FPS
640×360	230,400	3,28	304,88
1280×720	921,600	5,42	184,50
1920×1080	2,073,600	9,36	106,84

As shown in Tables 3 and 4, increasing the frame resolution leads to a significant rise in computational load, primarily due to operations performed over the entire image, particularly morphological processing and spatial filtering. This behavior is consistent with the analytical complexity model, where these operations scale proportionally to $\mathcal{O}(W \cdot H \cdot k^2)$.

An increase in the number of objects results in a substantial growth of the classification time, confirming its linear dependence on K. In this scenario, the classification stage becomes the dominant contributor to the overall computational load.

Furthermore, the use of more complex neural network models directly increases the classification time while leaving frame-level processing nearly unchanged. This highlights the critical role of classifier selection in achieving a balance between accuracy and real-time performance.

To provide a visual interpretation of the impact of scene and algorithm parameters on computational load, a graphical representation of the execution time of the main algorithm components was constructed (Fig. 3). This approach allows not only quantitative evaluation of the results presented in Table 3, but also a visual analysis of the scalability behavior of the proposed method.

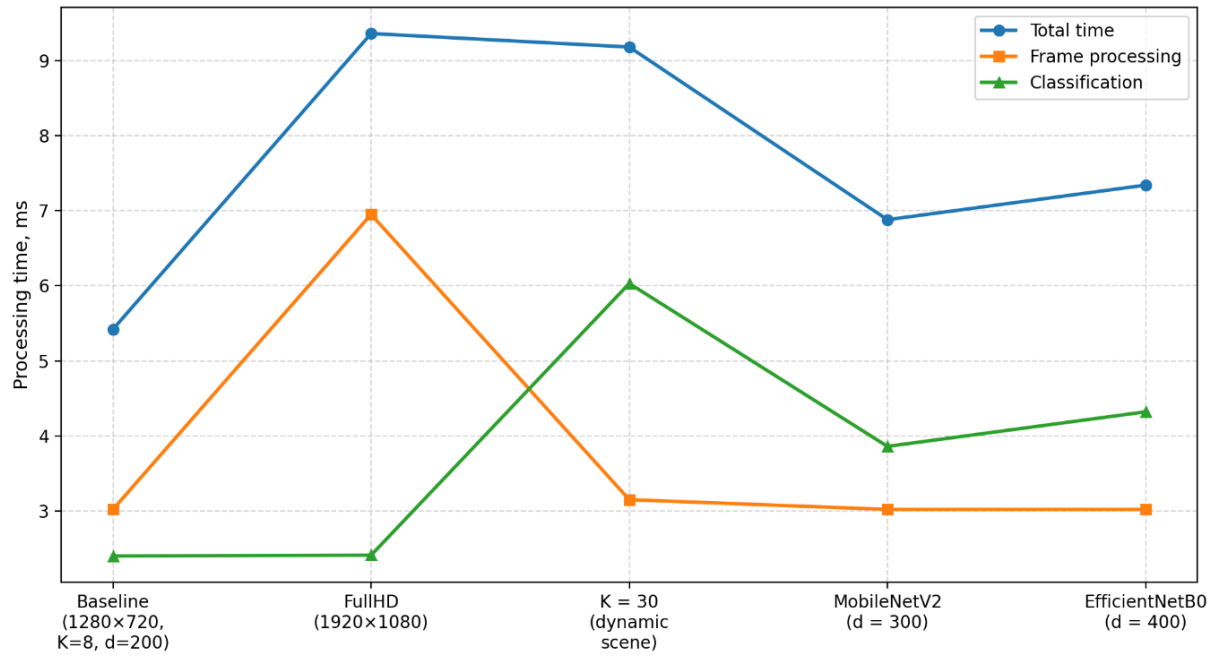

Fig. 3 – Processing time dependence on scene parameters and classifier complexity

As shown in Fig. 3, the total processing time (blue curve) significantly increases when the frame resolution is raised to FullHD, which is caused by the increased computational load of image-wide processing stages. In this scenario, the linear processing components (orange curve) demonstrate the most noticeable growth.

In the case of increasing the number of objects to K =30, a sharp increase in classification time (gray curve) is observed, while the contribution of frame-level processing remains nearly unchanged. This confirms the linear dependence of classification complexity on the number of objects.

When more complex neural network models are used (MobileNetV2, EfficientNetB0), the increase in computational load is again primarily associated with the classification stage, while other components of the algorithm remain relatively stable.

Thus, the graph in Fig. 3 demonstrates that different parameters affect different components of the algorithm: frame resolution primarily impacts image processing stages, whereas the number of objects and model complexity mainly influence the classification stage. This makes it possible to identify critical points for algorithm optimization depending on operating conditions, confirming the scalability and practical applicability of the proposed method.

Conclusions and prospects for further research.

The paper proposes an improved method for object detection and tracking in video streams, oriented toward onboard flight control systems and based on computationally efficient inter-frame difference analysis combined with adaptive processing mechanisms.

1. It has been established that the computational complexity of the method is primarily determined by morphological processing and object classification stages, which together account for approximately 70% of the total frame processing time (31,11% and 38,52%, respectively). This confirms their dominant impact on overall system performance.

2. Experimental profiling has shown that the method ensures real-time processing with an average frame processing time of 5,42 ms, corresponding to approximately 184, 5 FPS, while maintaining stable performance with deviations not exceeding ±7%, which indicates high temporal stability.

3. The scalability analysis demonstrated that increasing the frame resolution from 1280×720 to 1920×1080 leads to a rise in processing time by approximately 72,7%, mainly due to the increased computational load of image-wide operations.

4. It has been shown that increasing the number of objects from K=8 to K=30 results in an increase in classification time by more than 2,5 times, confirming its linear dependence on the number of detected regions.

5. The use of more complex neural network models (MobileNetV2, EfficientNetB0) increases classification time by approximately 60–80%, while the processing time of other algorithm stages remains nearly unchanged, highlighting the critical role of classifier selection.

6. The heatmap analysis confirmed the uneven distribution of computational load across processing stages and provided a visual identification of the most resource-intensive components, which is consistent with analytical and experimental results.

Thus, the obtained results confirm the scalability, computational efficiency, and practical applicability of the proposed method for real-time onboard video processing in flight control systems.

Further research will focus on optimizing the most computationally intensive stages of the algorithm, particularly morphological processing and object classification, through the use of adaptive filtering techniques and lightweight neural network models. In addition, it is promising to develop dynamic resource allocation and parameter tuning mechanisms depending on scene complexity, as well as to validate the proposed method in real onboard flight control systems under varying environmental and operational conditions.

References

1. Sadovnykov B. I., Zhuchenko O. S. A Method for searching and recognising objects in a video stream by calculating interframe deltas. Control, Navigation and Communication Systems, vol. 2, no. 80, pp. 249–254, 2025. DOI: https://doi.org/10.26906/SUNZ.2025.2.249.
2. Ladonia M. S. Research on the impact of Non_Maximal Suppression threshold value on YOLO's ability to recognize objects in low-quality images. Problems of Informatization and Control, vol. 2, no. 74, pp. 68–73, 2023. DOI: https://doi.org/10.18372/2073-4751.74.17884.
3. Sadovnykov B. I., Zhuchenko O. S. Mathematical model for object detection and recognition in video streams using inter-frame difference analysis. Science-Intensive Technologies, vol. 66, no. 2, pp. 181–189, 2025. DOI: https://doi.org/10.18372/2310-5461.66.20281.
4. Lebedeva O. Yu., Byrchenko T. O., Lebiha V. M. Development of an algorithm for object detection and tracking in video. Informatics and Mathematical Methods in Modeling, vol. 9, no. 1–2, pp. 59–68, 2019. URL: http://nbuv.gov.ua/UJRN/Itmm_2019_9_1-2_8.
5. Sadovnykov B., Lysechko V. Adaptive behaviour tuning of a neural network-based method for moving object recognition in video streams. Computer-Integrated Technologies: Education, Science, Production, no. 60, pp. 53–62, 2025. DOI: https://doi.org/10.36910/6775-2524-0560-2025-60-05.
6. Marchuk D. Analysis of modern algorithms for object detection and recognition from video streams for real-time parking systems. Herald of Khmelnytskyi National University. Technical Sciences, vol. 321, no. 3, pp. 17–23, 2023. DOI: https://doi.org/10.31891/2307-5732-2023-321-3-17-23.
7. Podchashynskyi Yu. O., Luhovykh O. O., Chepiuk L. O. Analysis of video image processing methods for measurement data obtained from thermal and spectral cameras. Technical Engineering, no. 1(91), pp. 214–221, 2023. DOI: https://doi.org/10.26642/ten-2023-1(91)-214-221.
8. Puida V. Ya., Stoian A. O. On Methods of Object Detection in Video Streams. Computer Systems and Networks, vol. 2, no. 1, pp. 80–87, 2020. DOI: https://doi.org/10.23939/csn2020.01.080.
9. Ahmed I. et al. Efficient top-view person detector using point-based transformation and lookup table. Computer Communications, vol. 147, pp. 188–197, 2019. DOI: https://doi.org/10.1016/j.comcom.2019.08.015.
10. Bouwmans T. Traditional and recent approaches in background modeling for foreground detection: An overview. Computer Science Review, vols. 11–12, pp. 31–66, 2014. DOI: https://doi.org/10.1016/j.cosrev.2014.04.001.
11. Brunetti A. et al. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. Neurocomputing, vol. 300, pp. 17–33, 2018. DOI: https://doi.org/10.1016/j.neucom.2018.01.092.

12. Chan K. L. Detection of foreground in dynamic scenes via two-step background subtraction. Machine Vision and Applications, vol. 26, pp. 723–740, 2015. DOI: https://doi.org/10.1007/s00138-015-0696-8.
13. Delleji T., Slimeni F. RF-YOLO: A modified YOLO model for UAV detection and classification using RF spectrogram images. Telecommunication Systems, vol. 88, article 33, 2025. DOI: https://doi.org/10.1007/s11235-025-01264-4.
14. Gheibi-Fetrat A. et al. A survey of SSD simulators and emulators. Journal of Supercomputing, vol. 81, article 592, 2025. DOI: https://doi.org/10.1007/s11227-025-07078-0.
15. Howard A. G. et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861, 2017. DOI: https://doi.org/10.48550/arXiv.1704.04861.
16. Jiang X. et al. Swimming-YOLO: A drowning detection method in multi-swimming scenarios based on improved YOLO. Signal, Image and Video Processing, vol. 19, article 161, 2025. DOI: https://doi.org/10.1007/s11760-024-03744-7.
17. Kamath V., Renuka A. Investigation of MobileNet-SSD on a human follower robot for stand-alone object detection and tracking using Raspberry Pi. Cogent Engineering, 2024. DOI: https://doi.org/10.1080/23311916.2024.2333208.
18. Kamath V., Renuka A. VireNet-SSD: Object detection model for resource-constrained applications based on self-organized operational neural networks. Neural Computing and Applications, vol. 37, pp. 8547–8569, 2025. DOI: https://doi.org/10.1007/s00521-025-10986-0.
19. Khandakar A. et al. DSPNet: A self-ONN model for robust DSPN diagnosis from temperature maps. IEEE Sensors Journal, (99):1-1, 2023. DOI: https://doi.org/10.1109/JSEN.2023.3235252.

Історія статті:
Отримано: 22.04.2026 Доопрацьовано: 13.05.2026 Прийнято до друку: 23.05.2026 Опубліковано: 29.05.2026