DOI: 10.18372/2310-5461.66.20281
УДК 621.391: 004.932.2: 004.93

B. Sadovnykov, Post-Graduate Student
Ukrainian State University of Railway Transport.
https://orcid.org/0009-0009-4180-2863
e-mail: borys.sadovnykov@gmail.com;

O. Zhuchenko, PhD. Associate Professor
Ukrainian State University of Railway Transport
orcid.org/0000-0003-3275-810X
e-mail: n030201@gmail.com

MATHEMATICAL MODEL FOR OBJECT DETECTION AND RECOGNITION IN VIDEO STREAMS USING INTER-FRAME DIFFERENCE ANALYSIS

Introduction

Modern computer vision systems offer extensive capabilities for the automatic analysis of visual information in real time. These technologies are increasingly being integrated into mission-critical domains – ranging from transportation safety to precision agriculture. In particular, object recognition in images or video streams serves as a key component in driver assistance systems, where automatic detection of traffic signs, pedestrian crossings, and road limitations helps reduce the risk of accidents. In agriculture, similar algorithms are employed to monitor crop conditions, detect plant diseases, or identify pests based on drone imagery.

In general, the task involves detecting, segmenting, and classifying objects within an image or a sequence of video frames. These operations must be robust against variations in lighting, perspective, scale, and partial occlusions. Such challenges necessitate the development of a mathematically formalized model that defines the full sequence of image processing stages – from initial transformations and extraction of key regions to the final classification and temporal tracking of objects. Formulating such a model enables the clear specification of functional relationships between the input data (frames), intermediate computations, and the final outcome – object identification within the scene. This, in turn, provides a foundation for theoretical analysis, stage-by-stage optimization, and experimental validation of the algorithm in real-world application scenarios.

Analysis of recent research and publications

To perform a proper mathematical modeling [1] all the operations and algorithms used inside the method should be deeply investigated.

A review of existing methods for processing video streams [1–16] highlights the proven effectiveness of convolutional neural networks (CNNs) in object recognition tasks involving visual data. While research such as [2, 3, 4] demonstrates high detection accuracy–especially when applied to large-scale datasets–these approaches are predomi-nantly designed for static image analysis or require significant computational resources, limiting their practicality in real-time or resource-constrained environments.

Real-time object detection approaches such as SSD (Single Shot MultiBox Detector) [5] and advanced YOLO variants provide competitive performance; however, they typically place a heavy load on the GPU and demand considerable video memory, which restricts their applicability in environments with limited computational resources [6].

Several studies [7–9] explore integrated frameworks that combine detection, tracking, and classifycation in video streams, but few incorporate early-stage optimization strategies such as frame differencing. For instance, although [8] presents a robust automated surveillance system, it lacks targeted techniques for reducing computational overhead.

Importantly, approaches leveraging inter-frame delta analysis (frame differencing) have demonstrated their effectiveness as lightweight preprocessing methods preceding classification. Research in [7, 10–12] confirms that analyzing pixel-wise differences between successive frames enables efficient and accurate identification of moving objects while significantly lowering processing requirements. Specifically, [7] introduces an enhanced differencing method that addresses issues such as noise and illumination shifts, further

reinforcing the viability of such techniques for real-time applications.

In parallel, advances in morphological image processing [13] contribute to refining the quality of regions extracted through frame differencing. The combination of these morphological filters with neural network-based classifiers, as shown in [10, 14, 16], enables a favorable trade-off between computational efficiency and recognition accuracy.

At the same time, existing reviews on thresholding and segmentation techniques [12, 15] often neglect the temporal complexity inherent to video streams, such as background variability, ambient interference, or partial object occlusion–factors that limit their effectiveness in practical monitoring and security scenarios.

Taken together, the reviewed methods do not provide a universal solution that guarantees both high-speed performance and low resource consumption. This underscores the necessity of developing a new approach that integrates inter-frame motion analysis (delta filtering) with adaptive neural classification, specifically tailored for real-time execution on hardware with constrained processing power.

The purpose of the work

The purpose of this work is to develop a mathematical model for real-time object detection and recognition in a video stream and to conduct an experimental research to confirm the model and find potential areas to improve.

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

The proposed method is comprehensive and consists of a sequence of preliminary linear and morphological transformations applied to the input image. The implementation algorithm is presented in Fig. 1 and includes the following sequence of steps.

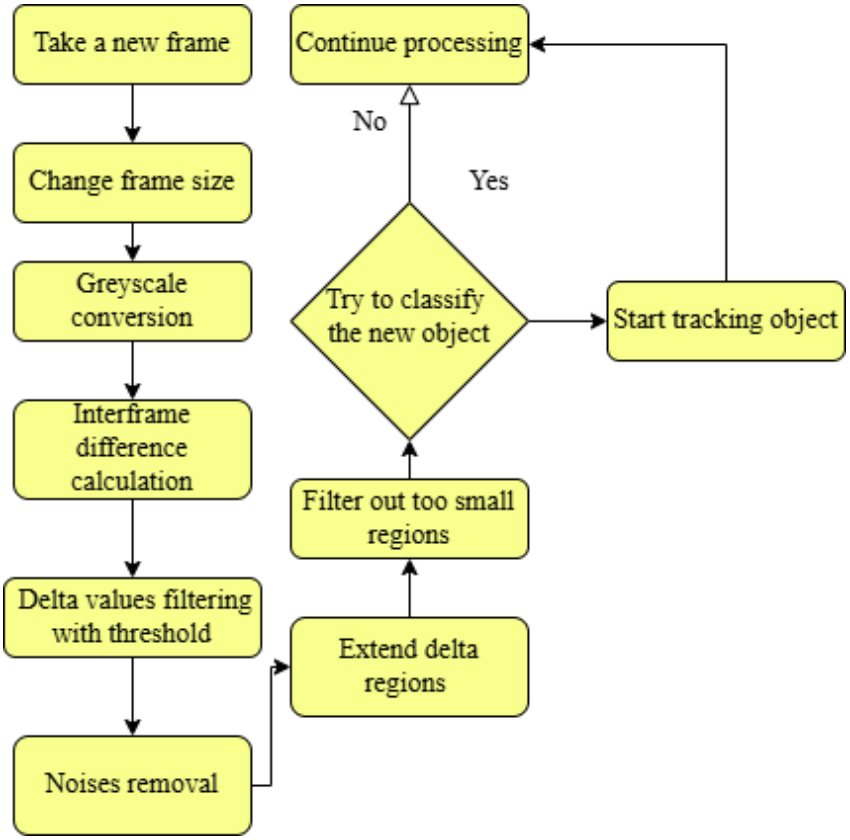

Fig. 1. Block diagram of the object detection and classification algorithm in a video stream

Step 1. Capturing the current frame

At the initial stage of processing, a new frame is extracted from the video stream, serving as input data for subsequent transformations. This process is performed continuously in real time or at fixed intervals, ensuring stable monitoring of scene changes and enabling prompt responses to the appearance of new objects or events.

The reliability and consistency of frame acquisition determine the system's overall ability to respond to changes within the field of view – the more accurate and stable this procedure is, the more effective the following stages will be.

Step 2. Rescaling the image to a unified format

After capturing the frame, it is rescaled to a standard resolution of 1280×720 pixels (720 p. format).

This unification simplifies subsequent processing, as the execution speed of most opera-tions directly depends on the number of pixels in the frame. Moreover, it helps avoid performance varia-bility caused by differing input video resolutions.

Most neural network models are also designed for a fixed input size, which improves their computational efficiency. Maintaining a consistent frame size is particularly important when comparing the performance of different methods and for building scalable solutions in resource-constrained environments. Furthermore, a unified image size is essential for batch processing and deployment on low-performance devices.

Step 3. Converting the image to grayscale

To reduce computational load, the captured image is converted to grayscale. This transfor-mation removes excess color information, which is often non-essential for object recognition. Reducing the number of channels from three (RGB) to one significantly accelerates processing, lowers memory usage, and improves overall system performance.

Additionally, grayscale conversion enhances the reliability of analysis by eliminating the influence of color artifacts and improving the detection of object contours and structural features within the frame. Grayscale images also provide better consistency when comparing frames, which is particularly important under varying or mixed lighting conditions.

Step 4. Change detection through inter-frame differencing

To detect the appearance of new objects or motion within the scene, the absolute difference between the current frame and the previous one is calculated. This approach is based on a fundamental property of video streams: a sequence of frames reflects temporal changes. If an object gradually enters the scene, the frame difference allows for accurate localization of its position.

In some cases, instead of comparing with the immediately preceding frame, a frame with a larger temporal offset (e.g., five frames earlier) is used. This enables better detection of slowly emerging or partially moving objects. The method is particularly effective under the condition of a stable, stationary camera.

An additional advantage of this approach is its ability to identify not only new objects but also those that have just disappeared from the scene, thereby enhancing the algorithm's overall functionality.

Step 5. Thresholding the frame difference

To eliminate minor changes caused by noise or lighting fluctuations, the computed frame difference undergoes threshold filtering. All pixels with changes below a defined threshold are ignored. This helps reduce the number of false positives and focuses processing only on significant areas within the frame.

By adjusting the threshold, the system's sensitivity to changes can be fine-tuned, which is essential for adapting to various application scenarios. It also allows better balancing between accuracy and system performance, ensuring sufficient result quality without overloading computational resources.

Step 6. Morphological cleaning and shape refinement

After thresholding, the resulting regions may still be fragmented, contain holes, or retain residual noise. To correct such artifacts, morphological transformations are applied to improve the structure of detected areas, close contour gaps, and remove small disturbances.

This results in coherent regions corresponding to real objects in the scene. A properly selected shape and size of the structuring element ensures a balance between processing accuracy and result stability.

These operations also make the image more suitable for subsequent analytical procedures by reducing the number of false or unstable regions. The operations are mathematically defined as follows:

Opening:

$$A \circ B = (A \ominus B) \oplus B,$$

Closing:

$$A \cdot B = (A \oplus B) \ominus B,$$

where A – the binary image, B the structuring element (kernel), $\ominus$ – erosion, $\oplus$ – dilation.

Step 7. Expansion of detected regions

Sometimes an object moves only partially, so the automatically detected region may not cover its entire area. To compensate for this and ensure complete coverage, an additional region expansion procedure is applied. Adjacent pixels that potentially belong to the object, but remained static – are added to the already identified region.

This step preserves the object's shape integrity before passing it to the classification stage. Such an approach reduces the risk of partial recognition or misclassification, which is critical in many real-world applications.

Step 8. Filtering out insignificant regions by size

After morphological processing, the frame may still contain regions too small to represent meaningful objects. These regions are excluded from further processing because they are unlikely to contain actual objects of interest.

This improves the overall efficiency of the system, reduces computational load, and enhances recognition accuracy. Reducing the number of false or non-informative regions contributes to the

system's stability and processing speed in practical conditions.

Step 9. Classification of detected objects

Regions that have successfully passed all previous filtering stages are passed to the classifi-cation module. At this stage, the system determines whether the detected region corresponds to an object of a specific class.

The choice of a particular neural network may vary depending on requirements for accuracy, performance, and available computational resour-ces. Classification enables the identification of objects and assigns appropriate labels for further analysis or actions.

Importantly, classification is not just a decision-making step – it also serves as a feedback mechanism to refine earlier processing stages if necessary.

Step 10. Object Tracking Over Time

Once an object is recognized, the system switches to tracking mode. This eliminates the need to run the full detection procedure on every frame, significantly reducing resource consumption. The system periodically checks whether the object is still present within the predicted region and, if necessary, re-initiates detection. The purpose of this stage is to ensure continuous monitoring of the object, even when its motion is minimal or absent. This supports analytical consistency in dynamic environments and reduces the number of missed objects in the video stream.

The proposed algorithm operates on a cyclic (iterative) processing principle: for each new frame, a complete sequence of steps is executed – from initial capture to object classification (from Step 1 to Step 9). If the object is not detected or fails classification, the system proceeds to the next frame, repeating the full cycle. However, once an object is successfully recognized, the tracking mode (Step 10) is activated. The main loop still continues in the background, periodically verifying the tracking's accuracy. If the object is lost or the prediction is unreliable, the system automatically reverts to detection mode for new frames.

Thus, the method combines the high processing speed of basic operations, such as linear and morphological transformations, with the accuracy provided by neural network classification. One of the main advantages of this approach is its predictable and stable processing time, achieved through frame size standardization and optimization of preproces-sing steps. This allows for quick and efficient localization of moving objects while consuming significantly fewer GPU and memory resources compared to fully neural-based solutions like SSD.

The proposed approach is particularly effective in systems operating under limited hardware resources – for instance, in real-time embedded solutions, surveillance cameras, or other compact devices with constrained computational capacity. In such cases, priorities include not only accuracy, but also rapid response, minimal system load, and overall operational stability under varying conditions.

Development of the mathematical model

Since the proposed method comprises multiple steps, constructing a complete mathematical model requires deriving expressions for each stage and integrating them into a unified framework.

The first step of the algorithm involves capturing a video frame. There are various sources of frames such as video files, live cameras, video streams, or inter-process communication. The origin of the frame is not fundamentally significant to the method, so we generalize the video input as a frame generator. The expression for capturing a frame is as follows:

$$F_t = V(t),$$

where: F_t – the current video frame at time t,, V – frame generation function, t – current time.

Depending on the frame generator, the time required to obtain a frame may vary. However, for the purpose of evaluating the algorithm's performance efficiency, the time spent acquiring the frame is typically excluded from the measurement.

The second step of the algorithm is scaling the image to a standard resolution.

Scaling involves recalculating the position of each pixel according to the new image dimensions. Each coordinate of the new pixel is determined through a proportional transformation of the original coordinates, taking into account the ratio between the input and target image sizes. The following model describes this transformation:

$$I_{out}(x', y') = I_{in}(\left\lfloor \frac{x' \cdot W_{in}}{W_{out}} \right\rfloor, \left\lfloor \frac{y' \cdot H_{in}}{H_{out}} \right\rfloor),$$

where $I_{out}(x', y')$ – pixel value at coordinates (x', y') - in the output (resized) image, W_{in}, H_{in} – original width and height of the image, W_{out}, H_{out} – target width and height of the image, I_{in} – input image, $\lfloor \cdot \rfloor$ – floor operator, used to round down to the nearest integer

The procedure preserves the proportions of objects in the image, enabling standardized processing regardless of the original resolution. This is critically important for ensuring consistent input data before passing it on to subsequent analysis stages.

In practical implementation, interpolation methods such as bicubic or bilinear interpolation are often used to fill in pixel values during scaling. However, in the general mathematical model of scaling, these can be omitted, focusing only on coordinate correspondence.

The third step of the algorithm is the conversion of the image to grayscale.

The conversion from the RGB format to grayscale involves reducing the three-dimensional color vector space to a one-dimensional intensity scale. For each pixel, a new value is calculated based on a weighted sum of its red, green, and blue components. The formula for this transformation is:

$$I_{gray}(x,y) = \alpha \cdot R(x,y) + \beta \cdot G(x,y) + \gamma \cdot \cdot B(x,y),$$

where $\alpha, \beta, \gamma \in [0,1]$ – are the weighting coefficients assigned to each color channel during brightness calculation, and they must satisfy the condition $\alpha + \beta + \gamma = 1$, $I_{gray}(x,y)$ – represents the grayscale intensity value of the pixel at coordinates (x,y), $R(x,y), G(x,y), B(x,y)$ – are the original red, green, and blue channel values of that pixel in the RGB color model.

The use of different weighting coefficients is based on the physiological characteristics of human vision: the green channel has the greatest influence on perceived brightness, the red channel has a moderate effect, and the blue channel contributes the least. This weighting preserves the realistic appearance of scene illumination after the conversion.

The result is a single-channel grayscale image $I_{gray}(x,y)$, which is much simpler to process further since each pixel is represented by a single brightness value rather than three separate color components.

The fourth processing step involves calculating the absolute difference between the current and previous video frames:

$$D(x,y) = \left| I_{t(x,y)} - I_{\{t-1\}(x,y)} \right|,$$

where $D(x,y)$ is the absolute difference at position (x, y), $I_{t(x,y)}$ is the pixel value at that position in the current frame, $I_{\{t-1\}(x,y)}$ is the corresponding pixel value in the previous frame.

The operation of computing the absolute difference between corresponding pixels of two consecutive frames makes it possible to detect changes in the scene at a specific point. If the value of $D(x,y)$ is close to zero, it indicates that no significant changes have occurred in that area of the image. Conversely, high difference values suggest the appearance, disappearance, or movement of an object.

Thus, this operation allows identifying regions of activity based on local changes in pixel intensity over time. The use of the absolute value ensures that changes – regardless of their direction (increase or decrease in brightness) are treated uniformly.

Computing the difference is a basic yet highly effective preprocessing technique for detecting moving objects, as it allows for the rapid localization of areas within the frame where significant changes have occurred between two points in time.

The fifth step, following the computation of the absolute difference between frames, is threshold filtering, which can be formally described as follows:

$$B(x,y) = Threshold(D(x,y), \theta) = = \begin{cases} 1, & if\ D(x,y) > \theta \\ 0, & if\ D(x,y) \leq \theta \end{cases},$$

where $B(x,y)$ is the result of the thresholding operation at point (x, y), i.e., the binary image;, $D(x,y)$ is the absolute pixel difference obtained from the previous step; θ is the threshold value that defines the system's sensitivity to changes..

Thresholding transforms the absolute difference map $D(x,y)$ into a binary image $B(x,y)$, д where each pixel is assigned a value of 1 (active) if the brightness change exceeds the defined threshold θ or 0 (inactive) otherwise.

Thus, only those regions of the scene are selected where sufficiently significant changes have been recorded, potentially indicating the appearance or movement of objects. The choice of threshold θ allows for adjusting the system's sensitivity:

– a lower value of θ leads to the detection of even minor changes but may increase the number of false positives caused by noise or lighting variations;

– a higher value of θ reduces false detections but might miss weak or partially obscured objects.

Threshold filtering is a crucial stage of data refinement before further processing such as morphological operations, segmentation, and classification.

The next step in processing involves the use of morphological transformations to refine the shape of the detected objects:

$$B_{morph}(x,y) = C(O(B(x,y)),$$

where $B_{morph}(x,y)$ is the binary image after morphological transformations, $B(x,y)$ is the binary image after thresholding, C denotes the closing operator, O denotes the opening operator.

After constructing the binary change mask, morphological processing is performed using a sequence of opening and closing operations, which are fundamental techniques in mathematical morphology that help clean the regions from artifacts and enhance the structure of the detected objects.

The opening operation smooths contours, removes small noise spots, and eliminates isolated pixels that do not correspond to actual objects.

The closing operation fills small holes within the region and connects broken or fragmented parts of objects.

Applying both operations in succession produces a cleaned and topologically stable mask that is better suited for the next steps of classification.

After the binary mask is built, morphologically refined, and expanded, the system extracts connec-ted components, which represent potential objects. However, some of these regions may be too small to realistically contain meaningful structures or objects.

To eliminate such interferences, size-based filtering is applied: from the set of all detected regions, only those whose area exceeds a predefined threshold A_{min}. are selected. This allows for the exclusion of fragments that may result from residual noise or uneven illumination.

Thus, the system focuses only on significant, statistically relevant regions of the frame, optimizing both the subsequent classification step and the overall computational load. This is formally described by the following expression:

$$R_{valid} = \{R_i \in R \mid A(R_i) \geq A_{min}\},$$

where R – is the set of all binary regions extracted after morphological processing, $A(R_i)$ - denotes the area (i.e., the number of pixels) of regionR_i, A_{min} – is the predefined minimum area threshold (in pixels) below which a region is considered noise and excluded from further analysis. The classification step is then performed according to the following formula:

$$l_i = C(R_i), \ \forall R_i \in R_{valid},$$

where R_{valid} – is the set of valid regions after filtering, C – is the classification function, l_i – s the class label of the object, R_i – is the current region.

After the extraction and preliminary filtering of regions, the system proceeds to the classification stage, where each region R_i, which potentially contains an object, is passed to the classifier func-tion C, implemented using a neural network or another machine learning algorithm. The classifica-tion function returns a class label l_i corresponding to the type of object identified in the region. This step enables the system not only to detect the presence of an object but also to recognize its cate-gory, which is critically important for practical applica-tions such as monitoring, security, and analytics.

At the final stage, the recognized object – specifically, the region with its class label – is trans-ferred to the tracking algorithm, which enables monito-ring its position in subsequent frames without the need for repeated detection and classifycation. The corresponding mathematical expression is:

$$T_t^{(i)} = T(S_i), \forall i = 1, \dots, k,$$

where S_i – s the region and class of the object, T – is the tracking function, $T_t^{(i)}$ – is the set of objects passed to the tracker at frame t, k – is the number of recognized objects in frame t.

This approach significantly reduces computa-tional load while ensuring continuous object tracking over time. The tracking system updates the object's position based on new frames, maintaining its identity even under conditions of partial occlu-sion, noise, or brief loss of visibility.

Tracking is initiated immediately after classify-cation but is periodically validated for accuracy. If tracking reliability decreases or the object is no longer present in the expected region, the tracker is reset and full detection is reinitiated.

Experimental Setup

To validate the effectiveness of the proposed mathematical model, a series of experiments was conducted using video data captured from a static camera. During the experiments, the following performance metrics were collected.

1. Average processing time per frame (in ms)
2. Detection accuracy (in %)
3. Minimum processing time per frame (in ms)
4. Maximum processing time per frame (in ms)
5. Average time spent on linear transformations per frame (in ms)
6. Average time spent on object classification per frame (in ms)

The consolidated experimental results, including refined statistical values, are presented in Table 1.

Table 1

Comparative performance metrics of object detection methods

Parameter	Unit	Value	Comment
Average frame processing time	ms	5.4 ± 0.2	Sufficient for real-time performance
Accuracy	%	71.2 ± 1.0	Adequate precision for practical algorithm usage
Minimum frame processing time	ms	5.0 ± 0.2	Dependent on input data; execution time may vary due to linear operations
Maximum frame processing time	ms	6.2 ± 0.2	Dependent on input data; execution time may vary due to linear operations

The end of the table 1

Parameter	Unit	Value	Comment
Average time for linear transformations per frame	ms	2.88 ± 0.2	Total time for all linear transformations
Average classification time per frame	ms	2.52 ± 0.2	Time spent on classifying all detected regions

As shown in Table 1, morphological transformations take more processing time than object classification.

Although neural network computations are commonly considered the most resource-intensive, the results of experimental profiling indicate that a significant portion of the total processing time is consumed by earlier stages – such as image scaling, grayscale conversion, morphological processing, and region filtering. This is due to the fact that these operations are applied to the entire image or full binary mask, whereas classification is performed only on a limited number of localized objects.

The measurement results were validated through an additional series of 25 experimental trials.

Figures 2–3 visualize the key performance metrics based on the data presented in Table 1.

Figure 2 illustrates a comparison of minimum, maximum, and average frame processing times, constructed from the corresponding measurements.

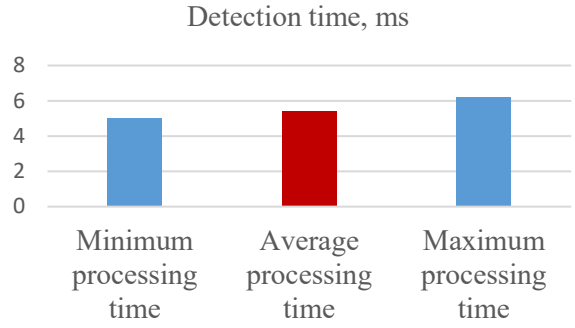

Fig. 2. Frame processing time

The visualization shows that frame processing time may fluctuate. This variability is caused by several factors.

1. Scene complexity. The number and size of regions detected in each frame directly affect processing duration. Frames without motion are processed significantly faster than those containing multiple objects or complex backgrounds.

2. Uneven system load: Since shared hardware resources are used (especially in CPU-based systems), the execution time of some operations may vary depending on the activity of other proces-ses running in parallel.

3. Mode switching conditions: For example, in frames where the object is already being tracked, only a partial processing cycle is executed. However, if an object is lost, the full detection pipeline is reactivated, leading to spikes in computational time.

4. Memory and caching behavior: Repeated processing of similar frames may be partially optimized through CPU caching or pre-allocated memory buffers, which affects the minimum observed execution time.

Another important factor is the ratio between the processing time required for linear image transformations and that required for object classification (Fig. 3).

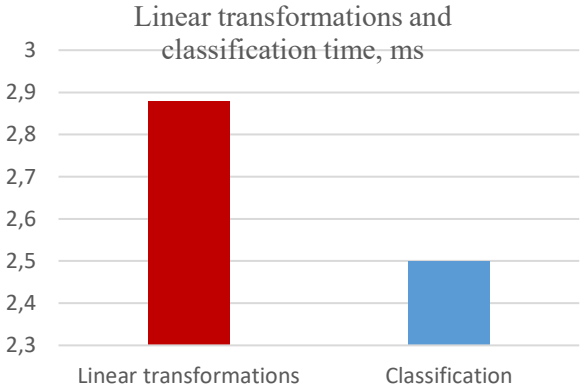

Fig. 3. Comparison of time consumption for linear transformations and classification

As shown in Fig. 3, linear transformations require more processing time than object classification due to their application across the entire image frame.

Experimental measurements show that linear transformations (such as scaling and grayscale conversion), along with morphological operations (erosion, dilation, opening, and closing), may consume more time than the classification stage itself. This is primarily because these transforma-tions are applied to every pixel of the entire frame, regardless of the presence of changes or objects. In contrast, the neural network performs computations only on a limited number of regions of interest, which have already been localized and filtered in advance.

Conclusions and prospects for further research

As part of this study, a mathematical model of an algorithm for object detection and classification in video streams was developed and substantiated. The model combines inter-frame difference computa-tion (delta analysis), threshold filtering, morpholo-gical processing, and subsequent neural classifica-tion of the extracted regions. It enables the formal

representation of the processing sequence as a set of functional transformations applied to each video frame, thus providing a structured foundation for implementation, analysis, and optimization of the algorithm.

A key feature of the model lies in the clear separation between low-level preprocessing and high-level classification stages, allowing the system to be effectively adapted to hardware-constrained environments. By incorporating operations such as image scaling, grayscale conversion, absolute difference calculation, and morphological transformations, the model achieves a flexible balance between accuracy, processing speed, and computational efficiency.

The model encompasses both spatial and temporal aspects of visual data analysis, making it suitable for real-time applications, including embedded platforms and video surveillance systems. Its formalization through mathematical operators supports further analytical investigation and facilitates comparison with alternative approaches.

Future research may focus on adaptive tuning of threshold parameters, dynamic selection of structu-ral elements for morphological filtering, and exten-ding the model to account for camera motion, vari-able lighting conditions, and multiclass object classifycation. Additionally, further optimization and acceleration of linear transformations through hardware support and more efficient algorithms repre-sent a promising direction.

REFERENCES

[1] Yue W., Liu S., Li Y. (2023) Eff-PCNet: An Efficient Pure CNN Network for Medical Image Classification, *Applied Sciences* 13(16):9226, https://doi.org/10.3390/app13169226.

[2] Cui W., Zhang Y., Zhang X., Li L., Liou F. (2020) Metal Additive Manufacturing Parts Inspection Using Convolutional Neural Network, *Applied Sciences* 10(2), 545; https://doi.org/10.3390/app10020545

[3] Simonyan K., Zisserman A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition, https://doi.org/10.48550/arXiv.1409.1556

[4] Zhao, Z. et al. (2019). Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11), 3212–3232. DOI: 10.1109/TNNLS.2018.2876865.

[5] Liu W., Anguelov D., Erhan D., Szegedy C., Reed S., Fu C., Berg A. (2016) SSD: Single Shot MultiBox Detector, Computer Vision and Pattern Recognition, P. 21–37, https://doi.org/10.1007/978-3-319-46448-0_2

[6] Ravi N., El-Sharkawy M. (2022) Real-Time Embedded Implementation of Improved Object Detector for Resource-Constrained Devices. Journal of Low Power Electronics and Applications 12(2):21, April 2022, DOI:10.3390/jlpea12020021.

[7] Huang W., Kang Y., Zheng S. (2017) An improved frame difference method for moving target detection. 2017 IEEE 16th International Conference on Cognitive Informatics & Cognitive Computing, DOI: 10.1109/ICCI-CC.2017.8109746.

[8] Mohana, Ravish Aradhya H.V. (2022) Design and Implementation of Object Detection, Tracking, Counting and Classification Algorithms using Artificial Intelligence for Automated Video Surveillance Applications, Conference: 24th International Conference on Advanced Computing and Communications, 2022.

[9] Singh, B., et al. (2014). Motion Detection for Video Surveillance. In Proceedings of the 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT) (pp. 592–597). IEEE. DOI:10.1109/ICSPCT.2014.6884919.

[10] Lysechko V. P., Sadovnykov B. I., Komar O. M., Zhuchenko O. S. (2024) A research of the latest approaches to visual image recognition and classification. 2024, *National University «Zaporizhzhia Polytechnic». Radio Electronics, Computer Science, Control,* 1(68). P. 140–147, DOI 10.15588/1607-3274-2024-1-13.

[11] Warren L., (2025) Mathematical and Computational Modeling, https://www.researchgate.net/publication/389880837_Mathematical_and_Computational_Modeling

[12] Lysechko V., Syvolovskyi I., Komar O., Nikitska A., Cherneva G.: Research of modern NoSQL databases to simplify the process of their design. Academic journal: *Mechanics Transport Communications*, 2023, vol. 21, issue 2, article № 2363, ISSN 2367-6620.

[13] Rodriguez, J.; Ayala, D. (2001) Erosion and Dilation on 2D and 3D Digital Images: A new size-independent approach. In Proceedings of the Vision Modeling & Visualization Conference, Stuttgart, Germany,

[14] Lysechko V., Zorina O., Sadovnykov B., Cherneva G., Pastushenko V.: Experimental study of optimized face recognition algorithms for resource – constrained. Academic journal: *Mechanics Transport Communications*, 2023, vol. 21, issue 1, article №2343, ISSN 2367-6620.

[15] Guruprasad P. (2020) Overview of different thresholding methods in image processing, Conference: TEQIP Sponsored 3rd National Conference on ETACC

[16] Пуйда В. Я., Стоян А. О. (2020) Дослідження методів виявлення об'єктів на відеозображеннях. *Комп'ютерні системи та мережі*. Vol. 2, No. 1, С. 80–87, 2020, https://doi.org/10.23939/csn2020.01.080.

Садовников Б. І., Жученко О. С.

МАТЕМАТИЧНА МОДЕЛЬ ПОШУКУ ТА РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У ВІДЕОПОТОЦІ ІЗ ЗАСТОСУВАННЯМ АНАЛІЗУ МІЖКАДРОВИХ ЗМІН

У статті представлено математичну модель пошуку та розпізнавання об'єктів у відеопотоці в реальному часі, що базується на поетапному аналізі міжкадрових змін. Запропонований підхід поєднує базові лінійні та морфологічні операції з ефективною процедурою міжкадрового порівняння, що дозволяє виявляти об'єкти шляхом локалізації змін у послідовності кадрів, а також класифікувати їх із застосуванням нейронних мереж. Формалізація алгоритму на рівні математичної моделі охоплює усі ключові етапи: масштабування зображення, переведення у відтінки сірого, обчислення абсолютної різниці між кадрами, порогову фільтрацію, морфологічне очищення, виділення регіонів інтересу, класифікацію виявлених об'єктів та їх подальше відстеження у часі.

Модель побудована як послідовність функціональних перетворень, які охоплюють як просторові, так і часові аспекти обробки відеоінформації. Обґрунтовано доцільність використання міжкадрової різниці як базового детектора активності, що дозволяє значно зменшити навантаження на систему порівняно з повністю нейромережевими рішеннями (наприклад, SSD або YOLO). Для уточнення форми та структури об'єктів застосовано класичні морфологічні фільтри (відкриття, закриття), а процедура фільтрації за розміром дозволяє виключити шумові області з подальшої обробки. На завершальному етапі модель передбачає передачу відібраних регіонів на модуль класифікації, що забезпечує розпізнавання типу об'єкта та його подальше відстеження без потреби у повторному виявленні.

Проведено експериментальну перевірку працездатності моделі на прикладі відео з фіксованої камери. Отримані результати демонструють, що середній час обробки одного кадру становить 5,4 мс, що відповідає вимогам реального часу, а точність розпізнавання досягає 71,2 %. Профілювання показало, що найресурсоємнішими є етапи морфологічної обробки, тоді як класифікація охоплює менше половини загального часу обробки. Це свідчить про ефективність комбінованого підходу, де лінійні й прості операції попередньої обробки дозволяють зменшити обсяг даних для класифікації без істотної втрати точності.

Ключові слова: математична модель, машинне навчання, комп'ютерний зір, обробка зображень, згорточні нейронні мережі, візуальне розпізнавання зображень, класифікація візуальних зображень, алгоритми, телекомунікаційні системи.

Sadovnykov B., Zhuchenko O.

MATHEMATICAL MODEL FOR OBJECT DETECTION AND RECOGNITION IN VIDEO STREAMS USING INTER-FRAME DIFFERENCE ANALYSIS

The paper presents a mathematical model for real-time object detection and recognition in video streams, based on stepwise analysis of inter-frame changes. The proposed approach integrates basic linear and morphological operations with an efficient inter-frame differencing procedure, enabling the localization of moving or newly appearing objects across consecutive frames, followed by their classification using neural networks. The formalized algorithmic structure of the model covers all essential stages: image scaling, grayscale conversion, absolute difference computation, threshold filtering, morphological cleanup, extraction of regions of interest, object classification, and subsequent temporal tracking.

The model is structured as a sequence of functional transformations addressing both spatial and temporal aspects of video data processing. The use of inter-frame differencing as a core activity detector is justified as it significantly reduces the computational burden in comparison with fully convolutional deep learning models such as SSD or YOLO. Classical morphological filters (opening and closing) are employed to refine object contours, while size-based region filtering helps exclude noisy or irrelevant areas. At the final stage, validated regions are passed to a classification module, allowing identification of object types and enabling tracking without repeated detection.

An experimental evaluation was conducted using footage from a static camera to assess the model's effectiveness. The results demonstrate an average frame processing time of 5.4 ms, meeting real-time operational requirements, and a recognition accuracy of 71.2%. Profiling indicates that the most computationally intensive operations are associated with morphological processing, whereas classification accounts for less than half of the total processing time. This highlights the efficiency of the hybrid approach, where simple linear preprocessing significantly reduces the data load for classification without substantial accuracy loss.

Keywords: mathematical model, machine learning, computer vision, image processing, convolutional neural networks, visual recognition, image classification, algorithms, telecommunication systems.

Стаття надійшла до редакції 20.05.2025 р.
Прийнято до друку 11.06.2025 р.