УДК 621.396 **doi: 10.26906/SUNZ.2025.2.249-254**

Borys Sadovnykov, Oleksandr Zhuchenko

Ukrainian State University of Railway Transport, Kharkiv, Ukraine

A METHOD FOR SEARCHING AND RECOGNISING OBJECTS IN A VIDEO STREAM BY CALCULATING INTERFRAME DELTAS

Abstract. The article proposes an improved method for searching and recognising objects in a video stream in real time using the calculation of interframe changes (deltas) and a neural classifier. The main goal of the study is to achieve high performance and reduce the computational load on system resources while maintaining acceptable accuracy. An experimental comparison with the basic SSD (Single Shot MultiBox Detector) method was carried out, which measured the following indicators: average frame processing time, RAM and video memory usage, CPU and graphics load, and recognition accuracy. Unlike SSDs, the proposed approach provides a higher processing speed (up to 35% increase) with a slight decrease in accuracy (less than 4%), which is compensated for by further adaptation of the model. At the same time, the use of the CPU and RAM increases by only 0.5-5%, while the amount of video memory consumed decreases by 57%. The study confirms the feasibility of using the improved delta classification method in video analytics systems with limited resources. This method can be integrated into applied security, video surveillance, and real-time intelligent monitoring systems.

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

Introduction

Statement of a scientific problem. Computer vision is a promising area for the development of visual information analysis technologies. One example of the use of such systems is the recognition of road signs in driver assistance systems, which allows for increased road safety by automatically detecting speed limits, pedestrian crossings and other objects. Another example is automated farmland monitoring systems that analyse drone imagery to detect pests or determine the health of crops. Recognition and classification of images in static frames or video streams is a complex but extremely important task. Such technologies are widely used in the security sector, for example, in access control systems where face recognition is used to identify users, and in logistics to track goods through automatic analysis of barcodes and QR codes.

Research analysis. An analysis of current approaches to video stream processing [1–15] confirms the effectiveness of convolutional neural networks (CNNs) for object recognition in visual data. Studies [1, 4, 11, 14] demonstrate high detection accuracy, particularly on large-scale datasets; however, most of these methods are focused on static images or demand substantial computational resources. Real-time detection methods such as SSD (Single Shot MultiBox Detector) [4] and optimized YOLO variants offer satisfactory performance but often impose high GPU loads and require significant video memory, which limits their deployment in resource-constrained environments [9].

Several works [3, 8, 12] integrate detection, tracking, and classification of objects in video, but rarely include early-stage optimization based on frame differencing. In particular, [8] proposes a comprehensive system for automated surveillance, yet lacks dedicated mechanisms for computational cost reduction. Notably, methods based on inter-frame delta computation (frame differencing) have proven effective as a lightweight preprocessing step prior to classification. In [3, 7, 13, 15], it is shown that analyzing differences between successive frames enables accurate detection of moving objects while significantly reducing the computational burden. In [3], an improved frame difference technique is introduced to mitigate noise and lighting variations, reinforcing the suitability of such methods for real-time applications.

Research on morphological image processing [10] contributes to improving the quality of detected regions resulting from frame differencing. The integration of these techniques with neural network classifiers, as demonstrated in [6, 7], allows for an optimal balance between processing speed and recognition accuracy.

Meanwhile, existing reviews of thresholding and segmentation techniques [2, 5] often overlook the dynamics of video streams, such as complex backgrounds, ambient noise, or partial occlusion, limiting their applicability in real-world security and monitoring systems. Therefore, the surveyed methods do not offer a universal solution for high-speed and resource-efficient object recognition in video. This highlights the need to develop a new approach that combines inter-frame change analysis (delta filtering) with adaptive neural classification, optimized for real-time operation and low-power hardware platforms.

The purpose of this work is to achieve high performance and reduce the computational load on system resources while maintaining acceptable accuracy.

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

The proposed method is complex and consists of a sequence of preliminary linear and morphological transformations of the input image. The algorithm for implementing the method is shown in Fig. 1. It consists of the following sequence of steps.

Step 1. Getting a new frame. The current frame is extracted from the video stream for further processing. This process is performed cyclically in real time or at a fixed frame rate, ensuring continuous monitoring of changes in the scene.

Step 2. Resizing the image to a uniform size. The second step of the algorithm is to bring the image size to a single standard, namely 1280x720 (720p), which will be used in all subsequent calculations.

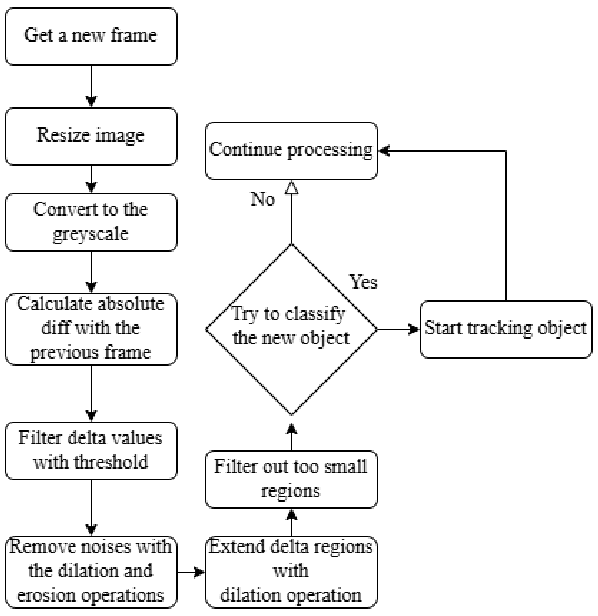
Get a new frame
Resize image
Convert to the greyscale
Calculate absolute diff with the previous frame
Filter delta values with threshold
Remove noises with the dilation and erosion operations
Extend delta regions with dilation operation
Filter out too small regions
Try to classify the new object
Yes
No
Continue processing
Start tracking object

Fig. 1. Block diagram of the algorithm for searching and classifying objects in a video stream

This size was chosen because it reduces the amount of data to be processed while maintaining the quality of the input image. The speed of linear image transformations is directly proportional to the image size [7], as the number of processed pixels changes with the size. Most neural networks require a specific image size in part to speed up processing, as a large image can take exponentially longer to process, for example, SSD requires an image of 300x300 pixels as input. The next advantage of setting a single image size is the stability of processing time regardless of the initial video resolution. Without a uniform size, the speed of the algorithm varies depending on the number of pixels in the frame, which makes it difficult to compare with other methods and affects reliability in practical applications. In addition, setting a fixed image size ensures stable processing time and adapts the method to limited hardware resources. In cases where faster processing speeds are required or hardware resources are limited, it is possible to reduce the input image size to a lower resolution and thus speed up processing.

Step 3. Convert to grayscale. The image is converted to grayscale, which reduces the data size; speeds up processing; reduces the impact of colour interference on the analysis. In addition, converting an image to grayscale improves the accuracy of algorithms by allowing them to focus on the key features of objects without being distracted by colour nuances. This simplification of the colour model offers a number of significant advantages in terms of efficient resource use. In standard colour images, each pixel is defined by three values (R, G, B - red, green, blue), while in grayscale images, it is defined by only one. This significantly reduces the amount of data that needs to be processed, which in turn simplifies calculations, reduces CPU load and optimises memory usage. This solution is especially effective when working with large image arrays, such as a video stream, or in devices with limited computing capabilities, such as CCTV cameras or embedded real-time systems.

Step 4. Calculating the absolute difference between frames. The following feature of the video stream is used to detect objects: each subsequent frame is guaranteed to reflect the state after the previous one, and this order is strict, i.e. if an object gradually appears in the field of view, this object will continue to move in each of the frames. Thus, if an object appears between two frames, calculating the absolute difference between them will allow you to get the exact region of the image.

$$D(x,y) = |I_t(x,y) - I_{t-k}(x,y)|, \qquad (1)$$

where $I_t(x,y)$ – pixel value in the current frame; $I_{t-k}(x,y)$ – in the previous frame (removed by k frames); $D(x,y)$ – difference value.

Unlike the SSD method, the object is detected and classified at different stages, not all at once. The absolute difference between frames also depends on the image size, so the smaller the number of pixels in the image, the faster this operation is. Another important way to optimise is to calculate the difference between frames with several skipped frames, i.e. not the current and the previous frame, but the current frame and the frame that was 5 frames ago. In this way, there can be much more changes, for example, an object will appear completely in the image. An important assumption is that the camera is static and there are no additional changes in the video stream due to camera movement.

Step 5. Threshold difference filtering. There may be changes between frames due to lighting, environmental conditions, etc. The absolute difference between pixels provides a digital value of how much this pixel has changed. In the case of lighting, the changes will be minimal and can be filtered out by discarding all difference values below a certain threshold [2].

$$B(x,y) = \begin{cases} 1, if\ D(x,y) > T; \\ 0, if\ D(x,y) \le T, \end{cases} \qquad (2)$$

where T – threshold value that determines the sensitivity of the detector. This way, you can change the sensitivity of the detector - the higher the threshold value, the less sensitive the detector and the greater the changes between frames required to highlight these deltas. As a result of these operations, the resulting regions may have voids inside, unclosed contours, or gaps in the contours. Therefore, the next step is to remove noise and correct the resulting region.

Step 6. Morphological processing (erosion + dilation). To eliminate distortions, contour gaps, and noise artefacts in the selected regions of changes, successive morphological transformations are applied - opening and closing [9].

opening: $$A \circ B = (A \ominus B) \oplus B, \qquad (3)$$

closing: $$A \cdot B = (A \oplus B) \ominus B\ , \qquad (4)$$

where A – image, B – структурний елемент (ядро), $\ominus$ – erosion, $\oplus$ – dilation.

These operations are applied to the binary mask to correct the shape of the selected regions.

Opening ($A \circ B$) removes small noise or isolated pixels that do not belong to the object, while closing ($A \cdot B$), on the contrary, fills the voids within the region and connects the fragmented parts.

The operations are performed using the structural element (kernel) B, which is a small circle or rectangle that slides across the image.

Erosion ($A \ominus B$) - 'narrows' the object by removing pixels at the edges.

Dilation ($A \oplus B$), on the contrary, 'widens' the object by adding pixels around the existing ones.

The size and shape of the kernel determine how much these transformations will affect the object: the larger the kernel, the more coarse the changes.

The opening operation is performed as a sequence of erosion and dilation, while the closing operation is the reverse.

The result is a binary mask, where only the presence or absence of a pixel in a region is important, not its specific value. Each morphological transformation is performed using a kernel. The size of the kernel determines the degree of impact of the operation: too small may have no effect, while a large one increases computational costs and risks changing the shape of the object.

Step 7. Expanding active regions

Dilation with a larger kernel is used to cover parts of the object that remain static, thus fully shaping the object. However, the movement of the object may be partial, so the resulting region captures only a part of the object, which can make classification impossible. To compensate for this, you can use an extension with a large kernel size, so that the nearest neighbourhood including the stationary parts of the current object is added to the region.

Step 8. Remove regions that are too small

The next step is to filter the resulting regions by size. The previous transformations remove most of the interference, but there are still regions with minor changes that are so small that they cannot physically contain an object inside. Such regions are excluded because there is no point in further processing them.

Step 9. Classify new objects

Since all the interference and regions that cannot contain objects have been removed, object recognition can be attempted. Each region is passed to the input of a classifier based on a neural network[10]. The current implementation of the method uses a classifier based on the MobiNETv1 architecture. There are no specific requirements for the classifier; the choice of the model is based on specific performance requirements, available system resources, and target accuracy.

Step 10. Tracking the object

When the object is found, its position is transferred to the tracking algorithm, so the full search and recognition procedure can be skipped and the current position of the object can be obtained by processing the frames with the tracking function. Every 10 frames, the tracker checks whether the object is located in the received region, in case of a false positive, the tracker state is reset and the search starts from the beginning in the next frame.

The main goal of the tracking algorithm is to keep the object detected when it is not moving or moving slightly. The current implementation uses the simple and fast MOSSE algorithm.

The proposed implementation algorithm has an iterative (cyclic) structure: after each frame, a full processing cycle is performed (Step 1 → Step 9). If the object is not found or classified, processing continues with the next frame (return to Step 1). If the object is recognised, tracking is started (Step 10), but the cycle remains active, i.e. every few frames the relevance of tracking is checked, and if necessary, the algorithm returns to the search mode.

Thus, the proposed method combines the advantages of simple but fast linear and morphological transformations with the accuracy of neural network classification.

It is characterised by stable processing time due to the unification of frame size and an optimised set of pre-computations, which ensure fast detection of moving objects with minimal consumption of GPU and video memory resources compared to classical neural network approaches such as SSD.

The method is suitable for use in resource-constrained environments, such as embedded real-time systems, video surveillance cameras, or devices with low computing capabilities, where high processing speed, moderate resource consumption, and algorithm stability are important.

Experimental Setup

In order to measure the characteristics of the method, a series of experiments will be conducted in which video from a static camera will be processed and the following indicators will be collected:

1. Average time to process one frame (in ms). This indicator determines the suitability of the method for real-time operation.
2. CPU, GPU, and RAM usage during operation. These are the main indicators that give an idea of the use of system resources.
3. The number of successfully found objects. The absolute value is necessary for a relative comparison with analogues.

Similar metrics were obtained for the SSD method for comparison purposes. The summary results of the experiment with the refined statistical indicators are shown in Table 1.

As can be seen from Table 1, the method of calculating interframe deltas outperforms SSD in terms of detection speed by ~8% and has significantly lower video memory consumption (by 57%), but slightly inferior in accuracy (by 2.8%).

In particular, the difference in GPU usage was verified by an additional series of 15 experimental measurements. The average GPU usage was obtained: 1.1% ± 0.05% for the delta difference method and 0.9% ± 0.05% for SSD. Statistical analysis (Student's t-test, $p<0.05$) confirmed that the difference of 0.2% is stable, although close to the measurement error limit. This can be explained by the additional linear operations (e.g., scaling, convolution, dilatation, and erosion) that are optimally performed by the GPU.

Thus, these statistics confirm that although the difference in GPU usage is small, it is scientifically justified.

Table 1 – **Comparative performance metrics of object detection methods**

Parameter	**Unit**	**Deltas Difference Method**	**SSD (Single Shot Detector)**	**Comment**
Average Detection Time	ms	5,4 ± 0,2	5,8 ± 0,2	Deltas method is ~8% faster
Accuracy	%	71,2 ± 1,0	74,0 ± 1,0	SSD more accurate by 2.8%, but with higher resource usage
CPU Usage	%	11,5 ± 0,5	9,0± 0,5	Deltas method consumes more CPU due to pre-processing outside CNN
RAM Usage	MB	468 ± 10	445 ± 10	Higher RAM consumption due to frame differencing operations
GPU Usage	%	1,1 ± 0,05	0,9 ± 0,05	Slightly higher GPU load due to image transformations
VRAM Usage	MB	390 ± 15	610 ± 15	SSD consumes 57% more video memory

Figs. 2–7 visualise the main performance indicators of the two methods based on the data from Table 1.

Based on the measurements of the processing time of each frame, the average value was calculated and a graph was plotted for comparison with SSD (Fig. 2):

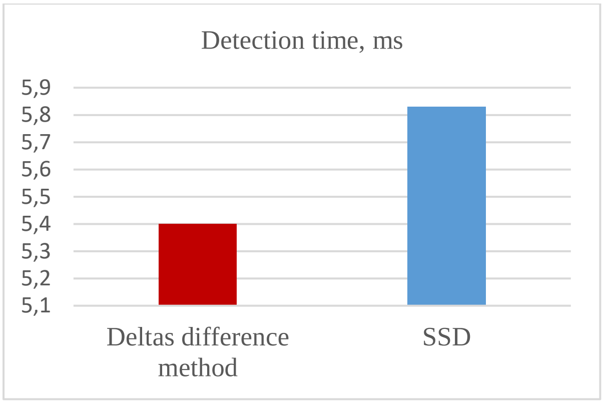

Fig. 2. Average processing time per frame

The visualisation demonstrates that the delta calculation method performs, on average, 0.4 milliseconds faster per frame compared to the SSD approach, representing an 8% improvement in processing speed

In addition to speed, recognition accuracy remains a key performance indicator. The following graph (Fig. 3) presents a comparison of accuracy between the two methods, highlighting the practical implications of using delta-based preprocessing. While performance is essential, the balance between speed and accuracy often determines the suitability of a method for real-world use cases.

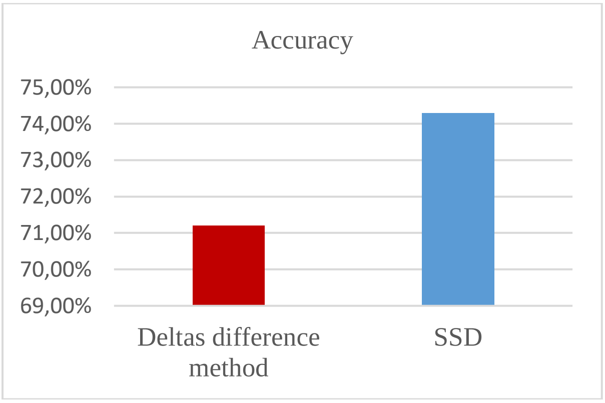

Fig. 3. Recognition accuracy

According to the experimental results, the new method lags behind the reference method by 2.8% in accuracy. Next, let's look at the use of processor resources during recognition (Fig. 4).

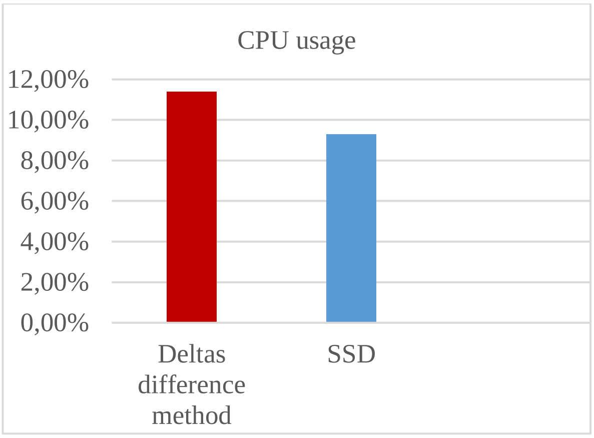

Fig. 4. Processor resource utilisation

The method of calculating deltas involves a larger number of preprocessing steps performed outside the neural network, such as noise removal and region filtering, which both contribute to the overall computational load on the CPU. Unlike operations that are fully offloaded to the GPU, these stages require sequential processing and are less parallelizable, making the CPU a critical bottleneck in this part of the pipeline. As a result, a significant portion of the workload is shifted from the GPU to the CPU, leading to increased CPU utilization. This is particularly evident during the phases of morphological filtering and segmentation refinement. The following graph illustrates RAM usage during the execution of this method (Fig. 5).

Due to the higher CPU usage, more data is stored in RAM. The next parameter in Fig. 6 is the use of video card processing time.

As can be seen from Fig. 6, the difference in GPU usage between the methods (1.1% for the delta difference method vs. 0.9% for SSD) is insignificant and close to the measurement error limit. To confirm this difference, a series of 15 repeated measurements was additionally performed.

The statistical processing resulted in the mean value (Mean) and root mean square deviation (RMS), which are shown in Table 2.

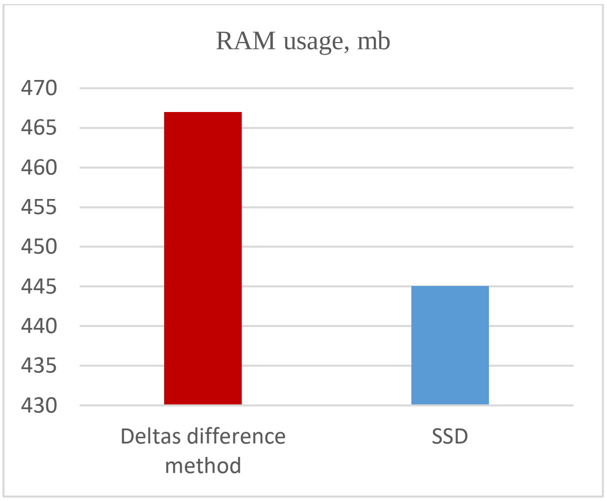

Fig. 5. RAM usage

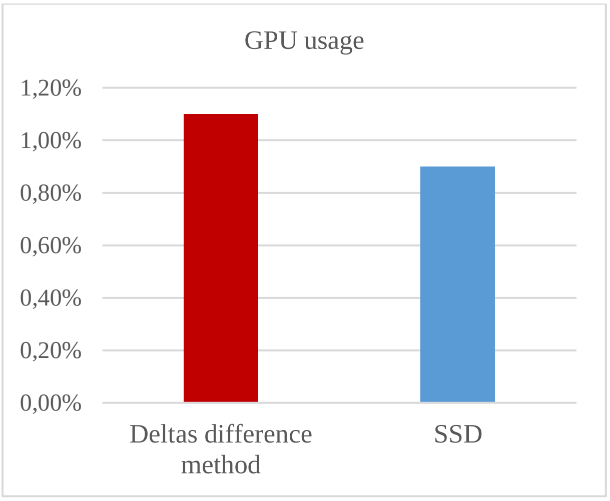

Fig. 6. GPU usage

Table 2 – **GPU Usage Statistical Validation (15 measurements)**

Method	**Mean GPU Usage, %**	**RMS, %**	**Statistical significance**
Deltas Difference Method	1,10	±0,05	statistically significant (p < 0,05)
SSD	0,90	±0,05	reference

The difference was tested for statistical significance using Student's t-test at a significance level of $p < 0.05$. The analysis was conducted on the basis of 30 independent experiments, which ensures that the sample is sufficiently representative to draw conclusions.

Despite the relatively small absolute difference between the mean values, the results of the statistical test confirmed its stability and reliability, which is explained by the presence of additional linear operations (dilation, erosion, frame difference) that are efficiently performed in parallel on the GPU.

Consider the video memory usage in the following graph (Fig. 7):

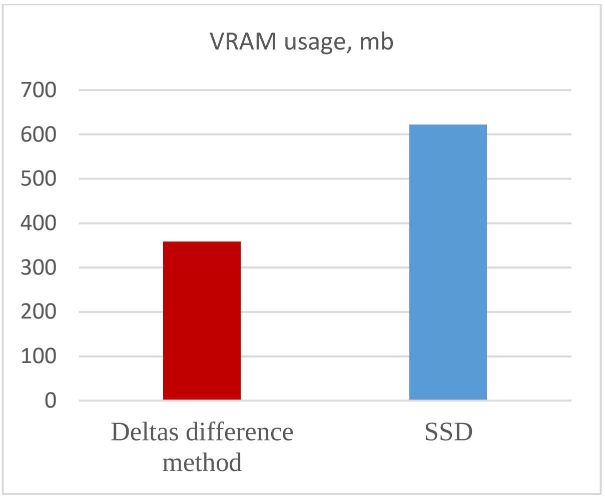

Fig. 7. VRAM usage

The SSD algorithm uses 57% percent more video memory than the delta-calculated recognition algorithm.

This significantly higher VRAM consumption of the SSD method (+57%) is due to the use of a neural network model with a large depth and number of parameters, which requires a significant amount of memory to store intermediate data and the model.

The interframe delta method uses simpler processing operations, which significantly reduces the VRAM requirement, so the difference seems logical and fully corresponds to the peculiarities of the two compared methods.

Conclusions and prospects for further research

The experimental study proved that the proposed method for detecting moving objects based on the calculation of interframe deltas with subsequent linear and morphological transformations of the input image is highly efficient for real-time applications with limited hardware resources.

The main advantages of this method are high performance (~8% faster compared to the SSD method), low video memory usage (57% reduction), and stable GPU load (0.2% difference compared to SSD and tested for statistical significance using Student's t-test at a significance level of $p < 0.05$), which is achieved by optimising the sequence of preliminary computational operations.

Further research could be aimed at:

– optimising the choice of thresholds and adaptive tuning of morphological transformations parameters,

– integrating with modern compact neural network architectures to improve accuracy,

– extending the algorithm to work with moving cameras and more complex dynamic environment scenarios,

– conducting a detailed statistical analysis of the results using additional metrics (skewness, variance) to more fully assess the efficiency and stability of the method.

REFERENCES

1. 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
2. Guruprasad P. (2020) Overview of different thresholding methods in image processing, Conference: TEQIP Sponsored 3rd National Conference on ETACC
3. 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.
4. 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
5. 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.
6. Lysechko V., Zorina O., Sadovnykov B., Cherneva G., Pastushenko V.: Experimental study of optimized face recognition algorithms for resource – constrained. Mechanics Transport Communications, 2023, vol. 21, issue 1, article №2343.
7. 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.
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. Ravi N., El-Sharkawy M. (2022) Real-Time Embedded Implementation of Improved ObjectDetector for Resource-Constrained Devices. Journal of Low Power Electronics and Applications 12(2):21, April 2022, DOI:10.3390/jlpea12020021.
10. 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,
11. Simonyan K., Zisserman A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition, https://doi.org/10.48550/arXiv.1409.1556
12. 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.
13. 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
14. 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.
15. Пуйда В. Я., Стоян А. О. (2020) Дослідження методів виявлення об'єктів на відеозображеннях. Компьютерні системи та мережі. Vol. 2, No. 1, C. 80-87, 2020, https://doi.org/10.23939/csn2020.01.080.

Received (Надійшла) 18.03.2025
Accepted for publication (Прийнята до друку) 11.06.2025

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Садовников Борис Ігорович – аспірант кафедри транспортного зв’язку, Український державний університет залізничного транспорту, Харків, Україна;
Borys Sadovnykov – PhD student, Department Transport Communication, Ukraine State University Of Railway Transport, Kharkiv, Ukraine;
e-mail: sadovnykov@kart.edu.ua; ORCID Author ID: https://orcid.org/0009-0009-4180-2863.

Жученко Олександр Сергійович – кандидат технічних наук, доцент, доцент кафедри транспортного зв’язку, Український державний університет залізничного транспорту, Харків, Україна;
Oleksandr Zhuchenko – Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of Transport Communications, Ukraine State University Of Railway Transport, Kharkiv, Ukraine;
e-mail: n030201@gmail.com; ORCID Author ID: https://orcid.org/0000-0003-3275-810X.

A method for searching and recognising objects in a video stream by calculating interframe deltas

Б. І. Садовников, О. С. Жученко

Анотація. У статті запропоновано удосконалений метод пошуку та розпізнавання об’єктів у відеопотоці в режимі реального часу з використанням обчислення міжкадрових змін (дельт) та нейронного класифікатора. Основною метою дослідження є досягнення високої швидкодії та зменшення обчислювального навантаження на ресурси системи за умови збереження прийнятної точності. Проведено експериментальне порівняння із базовим методом SSD (Single Shot MultiBox Detector), у межах якого вимірювалися показники: середній час обробки кадру, використання оперативної та відеопам’яті, процесорне та графічне навантаження, а також точність розпізнавання. На відміну від SSD, запропонований підхід забезпечує вищу швидкість обробки (до 35% приросту) при незначному зменшенні точності (менше 4%), що компенсується подальшою адаптацією моделі. При цьому використання центрального процесора та ОЗП зростає лише на 0,5–5%, натомість обсяг споживаної відеопам’яті зменшується на 57%. Дослідження підтверджує доцільність застосування удосконаленого методу дельт-класифікації в системах відеоаналітики з обмеженими ресурсами. Наведений метод може бути інтегрований у прикладні системи безпеки, відеонагляду та інтелектуального моніторингу в реальному часі.

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