Methods of Capturing and Tracking Objects in Video Sequences with Subsequent Identification by Artificial Intelligence

Mariia Nazarkevych[1,2], Vitaly Lutsyshyn[1], Vasyl Lytvyn[2], Maryna Kostiak[2], and Yaroslav Kis[2]

[1]*Lviv Ivan Franko National University, 1 Universytetska str., Lviv, 79000, Ukraine*
Lviv Polytechnic National University, 12 Stepan Bandera str., Lviv, 79013, Ukraine

Abstract

In video surveillance systems, the task of capturing an object and following it is relevant. An even more urgent task is the process of identification with subsequent authentication of the person. This makes it possible to organize the appropriate security of a private person or property, as well as the security of particularly important objects. A method of capturing the object has been developed. The use of the best among the considered methods in automated video surveillance systems will allow increasing the efficiency of intelligent video surveillance systems with a sufficient level of reliability. Footage from capturing the object is recorded in files. Implemented object capture using the MediaPipe Face Mesh library. In the future, it is planned to carry out the identification of a person using machine learning.

Keywords

Tracking objects, identification, artificial intelligence

1. Introduction

Tracking technology [1] is used in practice in automated video surveillance systems, traffic monitoring, motion-based recognition, such as gait-based identification, video indexing in large data catalogs, navigation of unmanned vehicles, Augmented Reality (AR), the film and advertising industry, for creating computer graphics and visual effects, in research in medicine and sports [2], medical imaging, human-computer interaction (gesture recognition, virtual keyboard, mouse, pupil tracking), crowd behavior analysis, video compression.

The rapid introduction and deployment of ubiquitous video sensors have resulted in the collection of massive amounts of data. However, indexing and searching large video databases remains a very difficult task. Augmenting media clips [3] with metadata descriptions are very useful for engineering. In our previous work, we proposed the notion of a visible scene model obtained by combining location and direction sensor information with a video stream. Such geo-referenced media streams are useful in many applications and, most importantly, they can be searched efficiently. The result of a geo-referenced video query will usually consist of the number of video segments that satisfy the query conditions but with more or less relevance.

The widespread availability of digital video sensors has led to a variety of applications, ranging from casual video recording to professional multi-camera surveillance systems [4]. As a result, a large number of video clips are collected, which creates complex data processing problems [5–7].

Many institutions deploy video surveillance systems with the primary function of protecting property rather than monitoring public traffic. Nowadays, there is cloud-based video surveillance that, in addition to security issues, solves the role of property protection. In addition, the main advantage of a cloud-based video surveillance system is that it is located outside the object of observation and has a flexible storage capacity.

A visual surveillance system [8] consists of two main parts:

- Target representation object.

CPITS 2023: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, February 28, 2023, Kyiv, Ukraine
EMAIL: mariia.a.nazarkevych@lpnu.ua (M. Nazarkevych); vitalylutsyshyn@gmail.com (V. Lutsyshyn); vasyl.v.lytvyn@lpnu.ua (V. Lytvyn); kostiak.maryna@lpnu.ua (M. Kostiak); yaroslav.p.kis@lpnu.ua (Y. Kis)
ORCID: 0000-0002-6528-9867 (M. Nazarkevych); 0009-0008-1229-6706 (V. Lutsyshyn); 0000-0002-9676-0180 (V. Lytvyn); 0000-0002-6667-7693 (M. Kostiak); 0000-0001-6552-7303 (Y. Kis)

- Localization object.
- Filtering.
- Data Association.

Target object representation and localization is mostly a bottom-up process, it is sequential and its subsequent steps do not affect the previous ones. The typical computational complexity of these algorithms is quite low.

Standard algorithms for finding and localizing [9] a target object include:

- Blob tracking: segmentation of the object's interior, including blob detection, block-based correlation, and optical flow.
- Kernel-based tracking (mean-shift tracking): an iterative localization procedure based on maximizing the similarity criterion.
- Contour tracking: search for the boundary of an object Feature matching: image registration.
- Point feature tracking: given a sequence of images of a certain scene captured by a moving or stationary camera. You need to get a set of the most accurate sequences of projection coordinates of some points in the scene in each frame.

Capturing and encoding digital images should lead to the creation and rapid dissemination of a huge amount of visual information. Therefore, efficient tools for searching and retrieving visual information are essential. Although effective search engines for text documents exist today, there are no satisfactory systems for retrieving visual information.

Due to the growth of visual data both online and offline and the phenomenal success of web search, expectations for image and video search technologies are increasing.

Hidden Markov Models (HMM) [10]. The method is based on a statistical comparison of an object with a database of templates. Hidden Markov models use the statistical properties of signals and take into account their spatial characteristics. Model elements: the initial probability of states, set of observed states, set of hidden states, transition probability matrix. During human recognition, all generated Markov models are checked and the highest probability that the sequence of observations for an object is generated by the corresponding model is searched.

To end our set of experiments, we should try PRNU SCI over video streams. 270 Basically, PRNU is equally valid for all frames that are captured in a video. Normally, a 271 video is made of 25 or 30 frames per second. Cameras usually reduce individual frames to 272 resolution when capturing a video: 720×576 is called standard definition, 1280×720 is 273 called enhanced definition, and 1920×1080 for high definition (or sometimes Full HD). 274 Nowadays there exist higher definitions like 3840×2160 (4K) or even 7680×4320 (8K). See 275 that even 4K is less than 8 Mpixels per frame.

The disadvantages include low processing speed and low resolution.

The system can only optimize the data processing time and response time of its model, but cannot minimize the time of searching other models.

Principal component analysis [11]. One of the most well-known and developed methods is the principal component analysis (PCA) based on the Karunen Loewe transformation [12]. Initially, the principal component method was used in statistics to reduce the feature space without significant loss of information. In face recognition, it is used mainly to represent a face image with a low-dimensional vector, which is then compared with reference vectors stored in a database.

The main goal of the principal component method is to significantly reduce the dimensionality of the feature space so that it describes "typical" images belonging to a large number of faces as well as possible. Using this method, it is possible to detect various variations in the training set of face images and describe this variability based on several orthogonal vectors.

The set of eigenvectors obtained once on the training set of face images [12] is used to encode all other face images, which are represented in a weighted combination of eigenvectors.

Using a limited number of eigenvectors, a compressed approximation of the input face image [13] can be obtained, which can then be stored in the database as a vector of coefficients, which is also a search key in the face database.

Also, the purpose of the PCA method is to reduce the feature space without significant loss of information so that it best describes the "typical" images belonging to a set of faces. In face recognition, it is used mainly to represent a face as a low-dimensional vector, which is then compared with reference vectors from the database.

PCA has proven itself in many applications. However, when there are significant changes in facial expressions or lighting in the face image, the effectiveness of the method drops significantly. This is because the principal component method selects a subspace to

maximize the approximation of the input data set, rather than to discriminate between classes of faces.

The Support Vector Machine (SVM) [14] method is a set of similar learning algorithms used for classification and regression analysis tasks. The essence of the support vector method is to find a hyperplane in the feature space that separates a class of face images from other images (the class of "no faces" images). In this case, out of all possible hyperplanes that divide into two classes, it is necessary to choose a hyperplane whose distance from each class is maximal.

The advantages of this method are high resistance to overtraining; high speed compared to neural networks; the ability to reduce sensitivity to noise by reducing accuracy.

The disadvantages include the fact that the accuracy of the method is inferior to many other methods.

Neural network methods [15]. Quite common methods include about a dozen different algorithms. The main feature of such networks is their ability to learn from a set of ready-made examples entered into the database in advance. During the training of neural networks, the network automatically extracts key features and builds relationships between them. After that, the trained neural network applies the experience gained to recognize a previously unknown object. Neural network methods show some of the best results in the field of recognition but are considered the most difficult to implement.

There are about a dozen different Neural Networks (NNs) [16]. One of the most widely used and popular instances is a network built on a multilayer perceptron, which allows you to classify an image or signal given as input according to the preliminary settings and training experience of the neural network.

Neural networks are trained on a set of training examples. The essence of training is to adjust the weights of inter-neuronal connections in the process of solving an optimization problem using the gradient descent method. During the training process, the neural network automatically extracts key features, determines their importance, and builds relationships between them.

It is assumed that the trained network will be able to apply the experience gained during training to unknown images using generalizing abilities.

2. Mathematical Model of the Object Tracking Algorithm

Let $I(x,t)$ be the brightness of the image frame with time t at point x, where x is a vector. The movement of the image far from the limits of visibility is described using an equation of the form: $I(x,t)=I(\text{delta}(x), t + t_1)$ (*), where delta(x) is the movement of point x when moving from the frame (t) to (t + 1). The movement of features from frame to frame is described by balancing for all points x from the surrounding features W [17].

In this case, the lighting of the points of the scene corresponding to the features remains constant.

With small changes in the image from frame to frame, you can read that the feature window is simply contained, and the delta(x) movement takes the form delta(x) = $x + d$. However, when the duration of tracking increases, the scene point image is distorted. This image can be approximately described by an affine transformation, therefore the movement of points is described by an affine transformation delta(x) = $Ax + d$, where A is a matrix of dimension 2×2.

The task of the tracker is to track the delta(x) movement values for all points of the feature window W. Since (*) is never performed in real conditions, the search is for such a movement that minimizes the difference between the windows at the current and next position on the frame, i.e. is the delta(x) at which the minimum is reached

$$e = min \sum_{W} delta(x), t + t_1 - I(x,t) \qquad (1)$$

or it is the norm of the image difference L2.

3. Detect Video Objects in Real-Time

Recurrent neural networks are based on sequential data and are well-suited for video object detection. A collaboration between the Georgia Institute of Technology (GIT) and Google Research proposed an RNN-based video object detection method [18] that achieved recognition rates of up to 15 frames per second, even running on a mobile processor.

One approach to detecting video objects is to split the video into its component frames and apply an image recognition algorithm to each frame. This rejects all the potential benefits of extracting temporal information from the video. Consider, for example, the problem of occlusion

and recapture: if a video object recognition system identifies a person who is then briefly covered by a passing pedestrian, it will take time for the system to realize that it has "lost" the object. An object can be lost not only due to occlusion, but also due to motion blur, in cases where camera movement or object movement (or both) causes enough disruption to the frame that elements become streaky or out of focus, and this is impossible for the recognition structure to identify.

If the system analyzes frames, this understanding is not possible because each frame is treated as a complete and closed episode. The computational cost of a video object detection system is less for object identification and tracking than re-registering the same object on a frame-by-frame basis.

Optical flow estimation generates a two-dimensional vector field that represents the pixel displacement between one frame and the neighboring frame (previous or next).

The optical flow can show the progress of groupings of individual pixels along the entire length of the video footage, providing guidance on which to perform useful operations. It has been used for a long time in traditional data-intensive video environments, such as video editing suites, to provide motion vectors along which filters can be applied, and for animation purposes. From the perspective of video object recognition, optical flow enables the computation of discontinuous object trajectories because it can compute mean trajectories in a way that is not possible with older methods such as the Lucas-Canade approach. This approach only considers the constant flow between grouped frames and cannot form a comprehensive relationship between multiple groups of actions, even though these groups may represent the same event interrupted by factors such as occlusion and motion blur.

4. Research of Software Products for Object Recognition

4.1. Google Video Intelligence

As a leading FAANG (Facebook, Amazon, Apple, Netflix, and Google) investor in computer vision research, Google's Video Intelligence system offers out-of-the-box features including object recognition in video, real-time OCR (optical character recognition) capabilities, logo detection, and face detection. Cloud Video Intelligence comes with a huge offering of pre-trained models, although customized training is available [19].

4.2. Hand Detection

The ability to perceive the shape and movement of hands can be a vital component in enhancing the user experience across a variety of technology domains and platforms. For example, it can form the basis for understanding sign language and controlling hand gestures, and it can enable the overlay of digital content and information on top of the physical world in augmented reality. Although it is natural for humans, reliable real-time hand perception is an extremely challenging task for computer vision because hands often cover themselves or each other (e.g., finger/palm occlusions and hand tremors) and lack high-contrast patterns. MediaPipe Hands is a highly accurate hand and finger tracking solution. It uses Machine Learning (ML) to identify 21 3D hand landmarks from just one frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments.

4.3. Palm Detection Model

To detect the initial location of the hand, there is a detector model optimized for real-time mobile use, similar to the face detection model in MediaPipe Face Mesh. Hand detection is an extremely challenging task: The simplified model and the full model have to work with different hand sizes with a large scale range (~20×) relative to the image frame and be able to detect closed and self-closed hands. While faces have high-contrast patterns, such as in the eye and mouth area, the lack of such features on hands makes it difficult to detect them reliably based on visual characteristics alone. Instead, providing additional contexts, such as hand, body, or facial features, helps to accurately localize the hand.

The method solves the above problems using different strategies. First, we train a palm detector instead of a hand detector because estimating the bounding boxes of solid objects such as palms and fists is much easier than detecting hands with articulated fingers. Furthermore, since palms are smaller objects, the non-maximal suppression algorithm works well even in cases of two-handed self-occlusion, such as handshakes. Furthermore, palms can be modeled using square bounding

rectangles (anchors in ML terminology), ignoring other aspect ratios, and thus reducing the number of anchors by a factor of 3–5. Secondly, an encoder-decoder feature extractor is used to increase the awareness of the scene context even for small objects. Finally, we minimize the focal loss during training to support a large number of anchors resulting from the high-scale dispersion.

With the above methods, an average accuracy of 95.7% in palm detection can be achieved. Using the conventional cross-entropy loss without a decoder gives a baseline of only 86.22%.

4.4. Hand Reference Model

After detecting the palm in the entire image, the subsequent reference hand model accurately localizes the key points of the 21 3D hand and finger coordinates within the detected hand region using regression, i.e. direct coordinate prediction. The model learns a consistent internal representation of the hand pose and is robust even to partially visible hands.

To obtain the data, the MediaPipe authors manually selected 21 coordinates (Figs. 1 and 2).

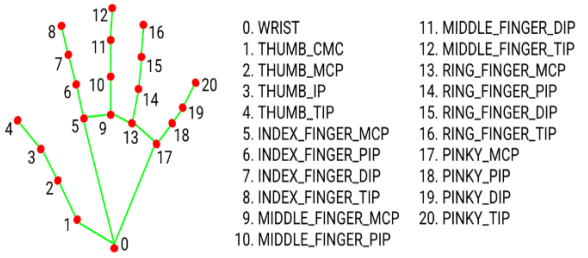

Figure 1: Palm skeleton search results

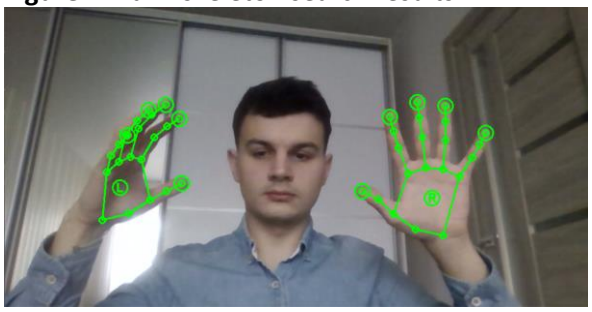

Figure 2: Implementation of palm capture in the video track

4.5. BlazePose Detector

BlazeFace provides two additional virtual key points that clearly describe the center of the human body, rotation, and scale as a circle. Using Leonardo's Vitruvian Man, the authors predicted the middle of the person's thighs, the radius of the circle surrounding the person, and the angle of the line connecting the middle of the shoulders and hips.

The landmark model in MediaPipe Pose provides the location of 33 pose landmarks (see Figs. 3 and 4) [20].

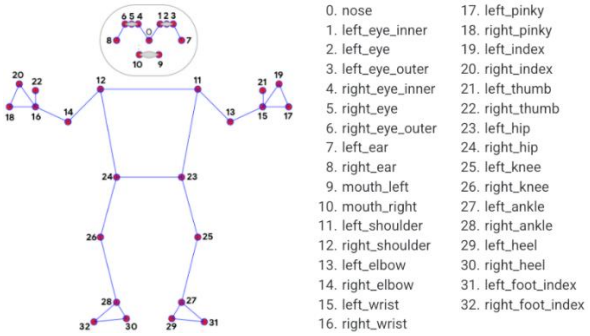

Figure 3: Creating a skeleton with MediaPipe

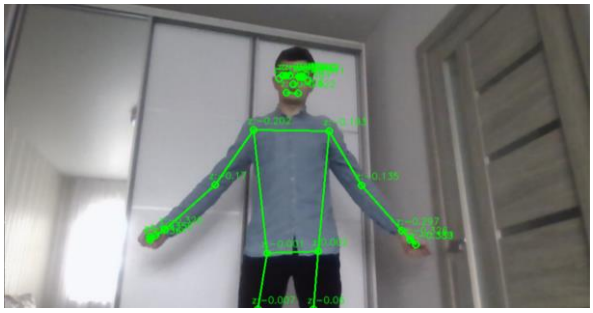

Figure 4: Implementation of capture in the skeleton video track using MediaPipe

4.6. Objectron

MediaPipe Objectron is a mobile solution for real-time 3D object detection. It detects objects in 2D images and estimates their position using an ML model trained on the Objectron dataset.

Object detection is a widely studied problem in computer vision, but most research has focused on 2D object prediction. While 2D prediction only provides 2D bounding boxes, extending the prediction to 3D can capture the size, position, and orientation of an object in the world, leading to a variety of applications in robotics, unmanned vehicles, image retrieval, and augmented reality. Although 2D object detection is relatively mature and widely used in industry, 3D object detection from 2D images is a challenging problem due to the lack of data and the variety of object appearances and shapes within a category.

When the model is applied to each frame captured by a mobile device, it may suffer from jitter due to the ambiguity of the 3D bounding box estimated at each frame [21]. To mitigate this, the same detection+tracking strategy is used in the 2D

object detection and tracking pipeline in MediaPipe Box Tracking. This reduces the need to run the mesh on each frame, allowing for heavier and therefore more accurate models while keeping the pipeline real-time on mobile devices. It also preserves subject identity in frames and ensures that predictions are consistent over time, reducing jitter.

The Objectron 3D object detection and tracking pipeline are implemented as a MediaPipe graph that internally uses a detection subgraph and a tracking subgraph [22]. The detection subgraph performs logical inference only once every few frames to reduce the computational burden and decodes the output tensor into a FrameAnnotation that contains nine key points: the center of the 3D bounding box and its eight vertices. The tracking subgraph runs each frame using the window tracking tool in MediaPipe Box Tracking to track a 2D box that tightly spans the projection of the 3D bounding box and upscales the tracked 2D key points to 3D using EPnP. When a new detection becomes available from the detection subgraph, the tracking subgraph is also responsible for consolidation between detection and tracking results based on the overlap region [23, 24].

5. Experimental Results

MediaPipe Selfie Segmentation can work in real-time on both smartphones and laptops.

Capturing an object in a video stream starts with the initialization of the ObjectTracker class, which is found in the cv2 library. Turn on the video camera. We perform scaling and set the interpolation characteristics. We create a window for the frame. We can enter a video recording and capture the track. We write the tracking window selection function. Set the size of the window. We capture the object. The algorithm analyzes the location of the object. Fig. 5 shows a fragment of the object tracking program.

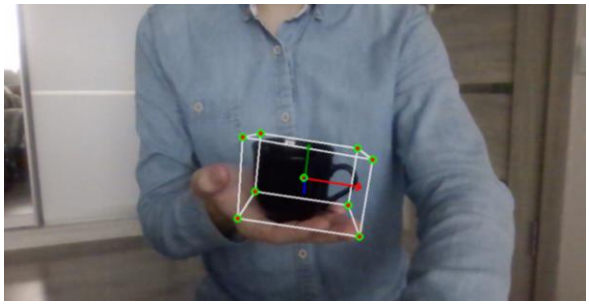

Figure 5: Capturing a 3-D object

```
import cv2
    # Method to start tracking the object
    def start_tracking(self):
        # Iterate until the user presses the Esc key
        while True:
            # Capture the frame from webcam
            _, self.frame = self.cap.read()
            #self.frame = cv2.rotate(self.frame,cv2.ROTATE_0)
            # Resize the input frame
            self.frame = cv2.resize(self.frame, None,
                    fx=self.scaling_factor, fy=self.scaling_factor,
                    interpolation=cv2.INTER_AREA)
            # Create a copy of the frame
            vis = self.frame.copy()
            # Convert the frame to HSV colorspace
            hsv = cv2.cvtColor(self.frame, cv2.COLOR_BGR2HSV)
            # Create the mask based on predefined thresholds
            mask = cv2.inRange(hsv, np.array((0., 60., 32.)),
                        np.array((180., 255., 255.)))
            # Check if the user has selected the region
            if self.selection:
                # Extract the coordinates of the selected rectangle
                x0, y0, x1, y1 = self.selection
                # Extract the tracking window
                self.track_window = (x0, y0, x1-x0, y1-y0)
                # Extract the regions of interest
                hsv_roi = hsv[y0:y1, x0:x1]
                mask_roi = mask[y0:y1, x0:x1]
                # Compute the histogram of the region of
                # interest in the HSV image using the mask
```

```
                hist = cv2.calcHist( [hsv_roi], [0], mask_roi,
                        [16], [0, 180] )
                # Normalize and reshape the histogram
                cv2.normalize(hist, hist, 0, 255, cv2.NORM_MINMAX);
                self.hist = hist.reshape(-1)
                # Extract the region of interest from the frame
                vis_roi = vis[y0:y1, x0:x1]
                # Compute the image negative (for display only)
                cv2.bitwise_not(vis_roi, vis_roi)
                vis[mask == 0] = 0
            # Check if the system in the "tracking" mode
            if self.tracking_state == 1:
                # Reset the selection variable
                self.selection = None
                # Compute the histogram back projection
                hsv_backproj = cv2.calcBackProject([hsv], [0],
                        self.hist, [0, 180], 1)
                # Compute bitwise AND between histogram
                # backprojection and the mask
                hsv_backproj &= mask
                # Define termination criteria for the tracker
                term_crit = (cv2.TERM_CRITERIA_EPS |
cv2.TERM_CRITERIA_COUNT,
                        10, 1)
                # Apply CAMShift on 'hsv_backproj'
                track_box, self.track_window = cv2.CamShift(hsv_backproj,
                        self.track_window, term_crit)
                # Draw an ellipse around the object
                cv2.ellipse(vis, track_box, (0, 255, 0), 2)
            # Show the output live video
            cv2.imshow('Object Tracker', vis)
            # Stop if the user hits the 'Esc' key
            c = cv2.waitKey(5)
            if c == 27:
                break
        # Close all the windows
        cv2.destroyAllWindows()
 if __name__ == '__main__':
   # Start the tracker
    ObjectTracker().start_tracking()
```

During the implementation of this experiment, the issue of object recognition in the video stream was considered. The main libraries of the Python language that can be used for the recognition and classification of objects from videos are covered.

MediaPipe methods for achieving a particular result in recognition are clearly described.

Since this experiment requires a significant number of images of objects from different viewing angles, a sample obtained using computer graphics and face generation by the MediaPipe program was used for its implementation. Face recognition testing was performed by matching the Viola-Jones method, an SVM classifier combined with histogram computation of oriented gradients and convolutional networks trained on the ImageNet sample. The trained models were provided by the Caffe libraries with OpenCV.

6. Conclusions

The result of the study was the study of the recognition algorithm. Experiments on capturing objects were conducted. Capture frames of objects were recorded in files. In the future, it is planned to identify these frames according to whether the face belongs to the given person or not. The MediaPipe library performed best.

A neural network for object recognition in a video stream was created and trained. An experiment was conducted in which the efficiency of the developed system was compared with the indicators of alternative known recognition methods. Recognition accuracy increases when using the proposed method. The developed recognition system is more resistant to local noise:

for images subject to blurring and occlusion, the recognition accuracy of the developed system drops. The research results are of practical interest in the design of management and information processing systems in the field of computer vision and image recognition, for those tasks where there is a need to determine the spatial parameters of the depicted objects. To use the system for real-time monitoring, it is necessary to analyze and design a system based on distributed computing for parallel analysis of frames because the results of local experiments show difficulties with many frames per second, activation of detectors of many levels requires significant computing resources.

[1] P. Moriggl, et. al., Touching Space: Distributed Ledger Technology for Tracking and Tracing Certificates, 56th Hawaii International Conference on System Sciences, January, 2023.

[2] M. Mashxura, I. Siddiqov, Effects of the Flipped Classroom in Teaching Computer Graphics, Eurasian Research Bulletin, 16 (2023) 119-123.

[3] A. Najmi, W. Alhalafawy, M. Zaki, Developing a Sustainable Environment Based on Augmented Reality to Educate Adolescents about the Dangers of Electronic Gaming Addiction, Sustainability, 15(4) (2023) 3185.

[4] R. Hyodo, et. al., Video Surveillance System Incorporating Expert Decision-making Process: A Case Study on Detecting Calving Signs in Cattle, arXiv, 2023, preprint.

[5] O. Iosifova, et al., Analysis of Automatic Speech Recognition Methods, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 2923 (2021) 252–257.

[6] K. Khorolska, et al., Application of a Convolutional Neural Network with a Module of Elementary Graphic Primitive Classifiers in the Problems of Recognition of Drawing Documentation and Transformation of 2D to 3D Models, Journal of Theoretical and Applied Information Technology 100(24) (2022) 7426–7437.

[7] V. Sokolov, P. Skladannyi, A. Platonenko, Video Channel Suppression Method of Unmanned Aerial Vehicles, in: IEEE 41st International Conference on Electronics and Nanotech-nology (2022) 473–477. doi: 10.1109/ELNANO54667.2022.9927105

[8] W. Lu, et. al., Blind Surveillance Image Quality Assessment via Deep Neural Network Combined with the Visual Saliency, Artificial Intelligence: Second CAAI International Conference, CICAI 2022, Beijing, China, August 2022, 136-146.

[9] S. Yu, J. Zhu, C. Lv, A Quantum Annealing Bat Algorithm for Node Localization in Wireless Sensor Networks, Sensors, 23(2) (2023) 782.

[10] R. Glennie, et al., Hidden Markov Models: Pitfalls and Opportunities in Ecology, Method. Ecol. Evol. 14(1) (2023) 43–56.

[11] A. Maćkiewicz, W. Ratajczak, Principal Components Analysis (PCA), Comput. Geosci. 19(3) (1993) 303–342. doi:10. 1016/0098-3004(93)90090-R.

[12] S. Adlersberg, V. Cuperman, Transform Domain Vector Quantization for Speech Signals, ICASSP'87. IEEE International Conference on Acoustics, Speech, and Signal Processing, 12 (1987) 1938–1941.

[13] A. Boyd, et. al., CYBORG: Blending Human Saliency Into the Loss Improves Deep Learning-Based Synthetic Face Detection, IEEE/CVF Winter Conference on Applications of Computer Vision, 2023, 6108–6117.

[14] A. Kurani, et. al., Comprehensive Comparative Study of Artificial Neural Network (ANN) and Support Vector Machines (SVM) on Stock Forecasting, Annals. of Data Sci. 10(1) (2023) 183–208.

[15] T. Tanantong, P. Yongwattana, A Convolutional Neural Network Framework for Classifying Inappropriate Online Video Contents, IAES Int. J. Artificial Intell. 12(1) (2023) 124.

[16] P. Wen, et. al., Fusing Models for Prognostics and Health Management of Lithium-Ion Batteries Based on Physics-Informed Neural Networks, arXiv, 2023, preprint.

[17] J. Zhang, D. Li, Research on Path Tracking Algorithm of Green Agricultural Machinery for Sustainable Development, Sustainable Energy Technologies and Assessments, 55 (2023) 102917.

[18] F. Azimi, et. al., Rethinking RNN-Based Video Object Segmentation, Computer Vision, Imaging and Computer Graphics Theory and Applications, CCIS, 1691 (2021) 348–365. doi:10.1007/978-3-031-25477-2_16

[19] D. Teece, Big Tech and Strategic Management: How Management Scholars Can Inform Competition Policy, Academy of Management Perspectives, 37(1) (2023) 1–15. doi:10.5465/amp.2022.0013
[20] S. Shaikh, et. al., Kinematic Pose Tracking for Workout App Using Computer Vision, Int. Res. J. of Modernization in Eng. Technol. Sci. doi:10.56726/irjmets33856
[21] M. Medykovskyy, et. al., (2015, September). Methods of Protection Document Formed from Latent Element Located by Fractals, 2015 Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies" (CSIT), 2015, 70–72. doi:10.1109/STC-CSIT.2015.7325434
[22] M. Logoyda, et. al., Identification of Biometric Images using Latent Elements, CEUR Workshop Proceedings, 2019.
[23] M. Nazarkevych, et. al., The Ateb-Gabor Filter for Fingerprinting, Conference on Computer Science and Information Technologies, 2019, 247–255.
[24] V. Hrytsyk, A. Grondzal, A. Bilenkyj, Augmented Reality for People with Disabilities, 2015 Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies (CSIT), 2015 188–191.