ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 7, Num. 2, 2022

SOFTWARE SYSTEM FOR END-PRODUCTS ACCOUNTING IN BAKERY PRODUCTION LINES BASED ON DISTRIBUTED VIDEO STREAMS ANALYSIS

Yurii Ivanov[1], *Borys Sharov*[2], *Nazar Zalevskyi*[1], *Ostap Kernytskyi*[1]

[1]*Lviv Polytechnic National University, 12, Bandera Str, Lviv, 79013, Ukraine.*
[2]*Electronic Systems Co. Ltd. 7, Zaliznychna Str., Lviv, 79018, Ukraine.*
Authors' e-mail: *yuriy.s.ivanov@lpnu.ua*

https://doi.org/10.23939/acps2022.__.___

Submitted on 10.10.2022

Abstract: **Among the main requirements of modern surveillance systems are stability in the face of negative influences and intellectualization. The purpose of intellectualization is that the surveillance system should perform not only the main functions such as monitoring and stream recording but also have to provide effective stream processing. The requirement for this processing is that the system operation has to be automated, and the operator's influence should be minimal. Modern intelligent surveillance systems require the development of grouping methods. The context of the grouping method here is associated with a decomposition of the target problem. Depending on the purpose of the system, the target problem can represent several subproblems, each of which usually accomplishes by artificial intelligence or data mining methods.**

***Index Terms*: Centroid, morphology, segmentation, tracking, video stream.**

I. INTRODUCTION

Video surveillance is a software-hardware system designed to monitor moving objects in video streams, continuously record, and quickly respond to alarm events in real time. Based on their structure, video surveillance systems are divided into centralized and distributed. Systems designed based on a centralized structure transmit network data for further processing, analysis, and storage to the central server. In distributed systems, data are distributed across the network to any workstation. Data that operate by the surveillance systems can be categorized as follows:

- configuration data: system structure, access rights and user authentication information, UI settings, surveillance devices configuration;
- video data: real-time video streams, archive video streams, alarm events, etc.

In a centralized video surveillance system, each camera in each subnetwork must regularly exchange data with the server to receive video streams and archive recordings, monitor for configuration updates, and configure user access rights. The main disadvantages of this structure are high cost, low reliability, fault tolerance, and scalability.

In distributed video surveillance systems, configuration data and video streams are distributed among each workstation, and in the operator's workstation, only the system cache and database cache can be stored. Although distributed system configuration parameters can be changed pretty often, it is possible to synchronize them: by automated schedule, request, or when changes were made. In case of server failure or database connection losses, users continue to work with the system using a backup copy of the database on the object server or in cache memory when a connection to the object server is missing. In distributed systems, video data decoding parameters and video analytics functionality are stored directly on surveillance cameras.

Video analysis in surveillance systems is usually represented as additional modules. Such modules contain typical interface functions to initialize the module, change settings, get the final results, and obtain the input frame. This approach to development allows the use of video analysis modules not only in particular but in any other system. A software system for object accounting is developed in the scope of such a module, which allows use in any distributed video surveillance systems.

II. PROBLEM STATEMENT

Among all the problems related to automated video surveillance systems, the issue of objects accounting in video streams stands out. Today, video data analysis is used in many areas, including bakery production lines in tunnel ovens.

Although in recent years a significant amount of research papers has been published, today objects accounting problem can't be considered solved completely. In many cases, there are factors like irregular lighting conditions, micro-movements, partial intersections, speed changes of moving objects, instability of geometric shapes, etc. All these factors, in many cases, significantly complicate video stream analysis.

Moreover, if there are several moving objects in the video stream at once, there is a problem with their interaction. That is why the development of object motion identification and accounting methods, which considers such factors as micromotions, partial intersections, changes in speed and stopping, and the parametric proximity of moving objects, is still an actual problem.

III. PURPOSE OF THE WORK

The work aims to develop and implement the algorithms that allow to provide detection, tracking, and accounting of bakery products in tunnel ovens. Algorithms should take into account such distortions as unstable lighting conditions, the presence of micro-movements, and partial objects occlusion. Algorithms should be implemented as a software module integrated into the existing distributed surveillance system as an additional video analysis module. The system functioning should be wholly autonomous and ensure the accuracy of accounting for at least 99 % of the total object number. Also, the system should keep track of the bakery conveyor's idle time, making it possible to optimize production operations. Video streams being analyzed are obtained from video surveillance cameras installed above the entrances or exits of the tunnel ovens. System settings should be changed remotely depending on the type of counting object. Results should be stored in the database and should be available in the form of histograms as well as in the form of data tables.

IV. RESEARCH PAPERS ANALYSIS

The solution to the moving objects accounting problem can be separated into two subtasks: object representation in video streams and object tracking over video frames. The representation process consists of object segmentation and calculation of the object's geometric features, which are invariant to transformations. Segmentation is one of the critical stages in the analysis of digital video streams. The segmentation process requires creating and periodically updating the background model. From the accuracy of this model, all subsequent stages of video stream processing are dependent, as well as the computational resources necessary. In general, all segmentation methods can be divided into the following groups: background extraction methods, probabilistic methods, time difference methods, and optical flow methods.

Background extraction methods are based on pixel-wise comparison of the current frame with the background model, a video frame where moving objects are removed, and periodical updating of the background model. The main disadvantages of these methods are the errors in pixels classification with periodic changes in the background and delays in model updating. In probabilistic methods, the scene is created by modeling the pixel process. An intensity change between current and previous frames is considered a linear combination of one-dimensional normally-distributed random variables for each pixel. More sophisticated methods are creating a pixel-wise model for an entire frame, using separate Gaussian mixtures for background and moving objects [1]. Based on the current time and variance of each Gaussian in a mixture, it is possible to determine which of them belongs to the background. Pixel values not in background distributions are considered moving objects until a Gaussian, which allows segment pixels to the scene with sufficient accuracy, will not appear. This approach finds slow changes in brightness in the video frame by adjusting the Gaussian parameters. Time difference methods [2, 3] provide background and moving object segmentation by calculating the differences in pixel values between two or more frames. The disadvantage of these methods is that they cannot wholly segment all pixels within similar brightness, which leads to gaps in a segmented object. In addition, these methods cannot detect partial object stops, so these methods are usually combined with those based on optical flow properties. Optical flow methods [4, 5] are based on the fact that each pixel's direction and motion speed can be calculated for every fragment containing the object. Optical flow information is used for spatial image segmentation: a group of adjacent pixels moving at the same speed can be considered the moving object. Also, information about this area's location, size, and other parameters can be obtained. The disadvantages of these methods are that they are resource-intensive and noise-sensitive.

Among moving object tracking methods, those based on contour extraction are the most effective for real-time video processing. [6] provided the tracking process, in which active contours determination is conducted within the segmentation method based on graphs opening. The resulting active contour in the previous frame is initialized in the current one. The new object contour is found using the intensity information in the current frame and the difference between the current and previous frames. Method [7] localizes objects within complex conditions of observation: partial intersections, objects exaggeration, etc. At the localization stage, the object's initial position is estimated using the Kalman filter and the Bhattacharyya coefficient. Applying this method has the advantage of multiple object extraction simultaneously.

Deep learning is also widely used for moving object tracking. For example, [8] describes modern CV and DL methods that allow the recognition of vehicles on the road and traffic violations of rules. The proposed method can recognize vehicles, track their speed, and help with precise object accounting.

V. MOTION DETECTION IN VIDEO FRAMES BASED ON A MIXTURE OF NORMAL DISTRIBUTIONS

An algorithm creates a pixel-wise model of the scene (Fig. 1) using a mixture of normal distributions, and with each new frame updates the model and classifies each pixel to background or foreground:

$$S \square \sum_{p=1}^{k} w_p \cdot \frac{1}{\sqrt{2 \cdot \pi \cdot \delta}} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \delta^2}}. \quad (1)$$

For each element in sum (1) corresponds to a process in pixel (Gaussian), which is characterized by normal distribution parameters (mean and variance), and coefficient w represents weight (an indicator of how often this process was in the field of view). Also, the U and V components of the YUV color space must be calculated to eliminate moving shadows. The parameter k in formula (1) is total Gaussians; it can be selected based on computational resources and is usually in the range [3; 5]. An algorithm that handles such things as micromotions, changes in the speed of objects, and their parametric proximity can be provided using the above model. We assume that gray-scale images come from the camera (Fig. 1).

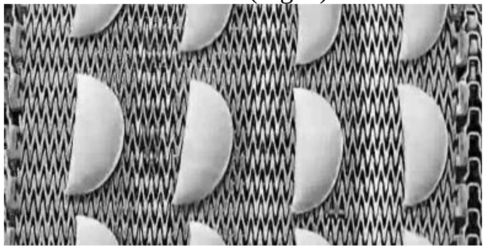


Fig. 1. Frame from camera to be processed

In the case of color images, they can be transformed to a gray scale using RGB to YUV converting formula. The algorithm consists of the following steps:

Step 1. A pixel-wise model is initialized at the first video frame, assuming all the frame's pixels are the background. Also, a pixel process for each pixel is created. Pixel process is created with the following parameters:

$$w = 1, \mu = I\left(x_0, y_0, 1\right), \delta^2 = \delta_{std}{}^2. \quad (2)$$

where δ_{std}^2 - default statistical mean can be chosen from the range [50; 65].

Step 2. Pixel process searching corresponds to the current value of the pixel brightness I(x_0, y_0, t). For every process, determine the threshold value:

$$\frac{\mu \cdot I\left(x_0, y_0, i\right)}{\delta} \le \varepsilon. \quad (3)$$

If the current pixel brightness value I(x_0, y_0, t) meets the requirement (3), then this process becomes current, and its statistic is updated.

Step 3. Creating a new pixel process. Mean and variance estimation are similar to pixel model initialization (2). If the number of processes in the model is equal to k, then the process with the lowest weight is required to be found. At the same time, the current weight does not change, and other parameters become identical to new process parameters; after that, an algorithm needs to be proceeded to step 6. If the number of processes for the current pixel has reached a maximum, then another process with zero weight value should be added, and the algorithm should be proceeded to step 5.

Step 4. Updating current process statistics. Mean, and variance values are updated using a low-frequency recursive smoothing filter. Let's denote μ_{t-1}, δ_{t-1}^2 as current process parameters in the previous step, and μ_t, δ_t^2 as process parameters in the current step, then:

$$\mu_t = \left(1 - \alpha_1\right) \cdot \mu_{t-1} + \alpha_1 \cdot I\left(x_0, y_0, t\right). \quad (4)$$

$$\delta_t^2 = \left(1 - \alpha_2\right) \cdot \delta_{t-1}{}^2 + \alpha_2 \cdot \left(\mu_t - I\left(x_0, y_0, t\right)\right)^2. \quad (5)$$

where α_1, α_2 – filter parameters that allow changing model updating speed.

Step 5. Updating weights of all processes. Let's denote $w_{i,t-1}$ as the weight of the process in the previous step and $w_{i,t}$ as the process in the current step, then

$$w_{i,t} = \left(1 - \alpha_3\right) \cdot w_{i,t-1} + \alpha_3 \cdot M\left(i\right). \quad (6)$$

where α_3 – is a parameter responsible for the process weight rate of change, M(i) – a function that returns 1 when argument i is equal to the current process index, and 0 – in all other values of argument i.

Step 6. Pixel classification and binary mask formation. The classification process assumes that the weight of the current process is compared with the threshold value, which can be determined from a range [0; 1]. If the weight of the current process is greater than the threshold value, the pixel is classified as background, otherwise as a moving object.

The result binary mask obtained after applying steps 1-6 of the algorithm is shown in Fig. 2.

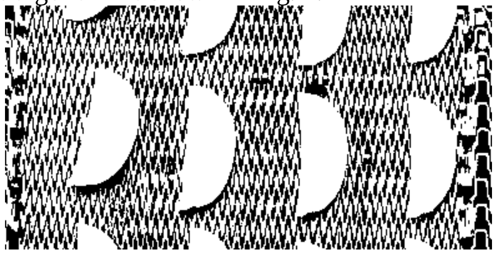


Fig. 2. Motion detection and binary mask processing result

VI. SHADOW REMOVAL AND FILTERING BASED ON MORPHOLOGICAL OPERATIONS

Accurate moving shadow detection and removal is one of the main challenges in dynamic scene segmentation algorithms since every moving object casts shadows. This can lead to a significant distortion of object shapes and different objects merging into one. These issues are significantly complicating further analysis and processing. A suppression algorithm that considers shadows' local (per-pixel) properties should be implemented to eliminate shadows.

Let's consider a frame pixel with R, G, and B components. After shadow falls on this pixel, it will have the color α·(R, G, B), where α determines how much pixel illumination has been decreased. This fact is quite enough to build a robust shadow removal algorithm. This algorithm will be presented for the RGB implementation of the algorithm with a single normal distribution. Still, by analogy, it can be integrated with a threshold and mixture of normal distributions. Let’s denote a color (R_1, G_1, B_1) of a background pixel (that is, the mathematical expectation estimation of a three-dimensional random variable that models the background at a given pixel) and current value (R_2, G_2, B_2) at a given pixel. First, it is advisable to switch to YUV color space. In YUV, the Y component represents intensity, and the U and V components represent chroma. When a shadow falls on an object, the value of Y should decrease significantly, and U and V values should not change much. So, the current pixel value should be classified as a shadow if:

$$|U_1 - U_2| \le \alpha, |V_1 - V_2| \le \beta. \quad (7)$$

where α and β – threshold values determine shadow removal sensitivity.

Parameter Y stands for brightness, and parameters U and V are calculated by the following expressions:

$$U = 0.492 \cdot (B - Y), V = 0.877 \cdot (R - Y). \quad (8)$$

The motion shadow detection algorithm is based on properties of shadow reflection in color images and is performed after the background segmentation algorithm. On the one hand, it is possible to significantly reduce computational resources since moving object shadow detection is conducted only at those pixels that were assigned to the foreground by the segmentation algorithm; on the other hand, filtering must precede all other actions since the purpose of the filter is to smooth out the results of all stages of the video stream segmentation work, without going into the specifics of its work.

This algorithm is executed for each pixel marked as foreground when forming the mask. The parameters required for the algorithm are calculated in the main algorithm. The algorithm for detecting shadows of foreground (moving) objects contains the following steps:

Step 1. When forming the mask, the algorithm is executed for each foreground pixel.

Step 2. Among the pixel processes, the process with the highest weight, $w_{i,t}$ is studied. Let’s assume that if this process has the highest weight and it is not foreground (the background pixel must have a significant weight), then this pixel color denotes the state of the background, as it was before the shadow fell on it. If this process is current, it is not marked as a shadow. The algorithm exits (so the rest of the processes that happened even less often, and therefore with a lower probability, can be the color of the background before the shadow falls on it; in this case, an element is likely a dynamic background, or for this pixel, the weight threshold is too low), otherwise we proceed to step 3.

Step 3. The relative modulus deviations of U and V values (YUV) for the current process from the process with the highest DU and DV weight are calculated:

$$DU = \frac{|U_c - U_{\max}|}{U_c}, DV = \frac{|V_c - V_{\max}|}{V_c}. \quad (9)$$

where U_c, V_c – U and V values of current process;

$U_{\max}$, $V_{\max}$ – U and V values of the process with the highest weight;

Step 4. If the values of the relative deviations calculated in the previous step are less than the threshold values and the brightness value of the process with the maximum weight is greater than the brightness value of the current process, then the pixel is declared as a moving shadow pixel and segmented back to the background; otherwise, it remains to be marked as a foreground element. That is, a pixel is declared as a shadow for which the following system of conditions is met:

$$\begin{cases} DU < A, \\ DV < B, \\ \mu_{\max} > \mu_c. \end{cases} \quad (10)$$

Step 5. Applying conditional morphological operations to binary mask (Fig 2). Let’s denote M_t^* as a binary mask on the current frame, and $M_{t,0}^* = M_t^*$ and where S – rectangular structuring element, then for the binary mask on the frame with index m, an operation (7) is applied:

$$\left(M_t^* \square \left| \begin{matrix} & S \\ M_i & \end{matrix} \right. \right) \cup \left(M_t^* \oplus \left| \begin{matrix} & S \\ M_i & \end{matrix} \right. \right) = M_{t,m}^*, \quad (11)$$

where $\square$ - morphological erosion, which is some set of elements z in which image M_t^* completely contains structuring element S. Applying the erosion operation, means that structuring element S is also gradually applied to all pixels in M_t^* image with the logical addition of the central pixel in S to the corresponding pixel in M_t^*.

$\oplus$ - morphological dilation. With this operation, a central mapping of structuring element S relative to its center and shifting the obtained set to the pixel z from image M_t^* is determined. The dilation operation determines the set of all shifts on the z-set for which M_t^* and S coincide in at least one element.

VII. BINARY-LARGE MOVING OBJECTS TRACKING BASED ON AREAS ESTIMATION AND CENTROIDS POSITION PREDICTION

This algorithm allows us to predict binary-large object (BLoB) position and area through all frames and build a track of objects by using centroid positions on all previous frames where BLoB was visible. The algorithm is applied to morphologically processed binary mask Fig. 3 (colors are inverted).

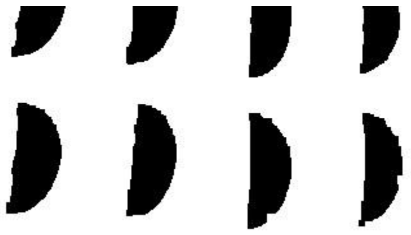


Fig. 3. Binary mask morphological processing result

The algorithm consists of the following steps:

Step 1. A contour vector is created for each binary mask in each frame and each BLoB. When defining external contours, horizontal, vertical, and diagonal segments are compressed, leaving only their endpoints. For example, the rectangular silhouette at the top right is encoded as four points. For each element in the contour vector, convex hulls are calculated, and a new convex hulls vector is created; the elements of this vector are sets of points that convex hulls contain. For each convex hull, a BLoB object is created. This object has the vector of coordinates of the convex hull bounding rectangle. This vector contains the object centroids on previous frames, a flag that indicates whether the object is in the tracking process, a vector of object areas, and a current object area.

Step 2. If the number of BLoBs is greater than zero, then the filter by diagonal size and area is applied to the vector of existing BLoBs; otherwise, the algorithm returns to step 1.

Step 3. For each BLoB in the vector of existing BLoBs in the current frame, coordinates of the predicted centroid position are estimated with the following expressions:

$$C_x^{\langle i+1\rangle} = \sqrt{\left(C_x^{\langle i\rangle}\right)^2 + \left(\frac{\sum_{k=i-1}^{i-5}\left\{\left(C_x^{\langle k\rangle} - C_x^{\langle k-1\rangle}\right)\cdot(k-1)\right\}}{n}\right)^2},$$

$$C_y^{\langle i+1\rangle} = \sqrt{\left(C_y^{\langle i\rangle}\right)^2 + \left(\frac{\sum_{k=i-1}^{i-5}\left\{\left(C_y^{\langle k\rangle} - C_y^{\langle k-1\rangle}\right)\cdot(k-1)\right\}}{n}\right)^2}. \quad (12)$$

where $C_x^{<i+1>}$ – centroid's x-coordinate predicted for i+1 frame, $C_y^{<i+1>}$ – centroid's predicted y-coordinate predicted for i+1 frame, k – previous frames count, based on these frames information prediction is accomplished, n – coefficient, that can be determined using the following condition:

$$n = \begin{cases} 2\cdot i, 0 \le i \le 5, \\ 10, i > 5. \end{cases} \quad (13)$$

If the predicted distance C is a less predefined threshold value, BLoB is considered in the tracking process or a new one. In this case, BLoB is added to the vector of existing BLoBs.

Step 4. Predicted BLoB's area estimation on the next frame by the following expression:

$$\frac{\sum_{k=i-1}^{i-5} A_j^{\langle k\rangle}}{5} \le Th_A. \quad (14)$$

In Formula 14, $A_j^{<k>}$ is j-BLoB's area on frame k, Th_A is area threshold.

If condition (14) is not met, BLoB is no longer in the tracking process and is removed from the vector of existing BLoBs.

Step 5. Checking whether the BLoB was in the camera's field of view during the last ten frames, i.e., centroid coordinates in centroid were changed, and the size of areas vector changed.

Step 6. Checking whether the BLoB's centroid crossed the line. In case if centroid y-coordinate is greater than the horizontal line position, BLoB is considered to be counted.

The tracking algorithm result is shown in Fig. 4 (colors are inverted). For each BLoB that meets the diagonal size and area filter conditions, area and diagonal size are also displayed.

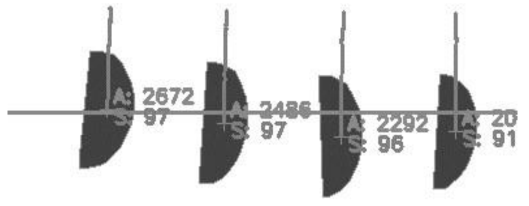


Fig. 4. Objects tracking result

VIII. DATABASE IMPLEMENTATION AND RESULTS REPRESENTATION

The settings UI (Fig. 5) are implemented using Qt tools such as QWidget and QWindow, which allows the creation of a simple and straightforward interface for the application.

The main window (Fig. 6) allows for monitoring accounting progress, performing the necessary settings, connecting video streams, and managing them. The interface allows it to manage all program modules and get the result.

When changing the current camera, the required CameraProcessor object is selected, through which the user can start the stream to account objects, stop and change settings. Settings changing is provided within SettingsWidget, which contains several QSliders and QSpinBoxes, for changing the settings for the parameters: dilation iterations, erosion iterations, threshold, cross-line position, maximal, and minimal blob size. Due to the interaction of signals and slots, it is possible to change settings in real-time and immediately view new results with updated settings.

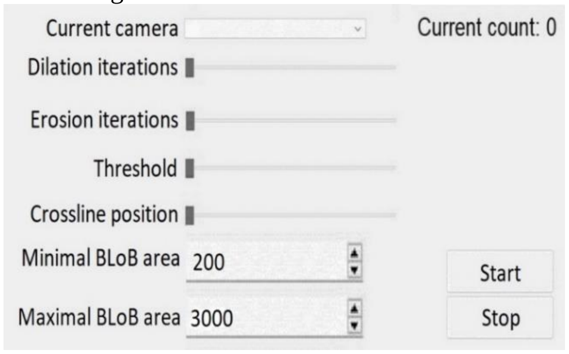

Current camera
Current count: 0
Dilation iterations
Erosion iterations
Threshold
Crossline position
Minimal BLoB area 200
Maximal BLoB area 3000

Fig. 5. Settings UI

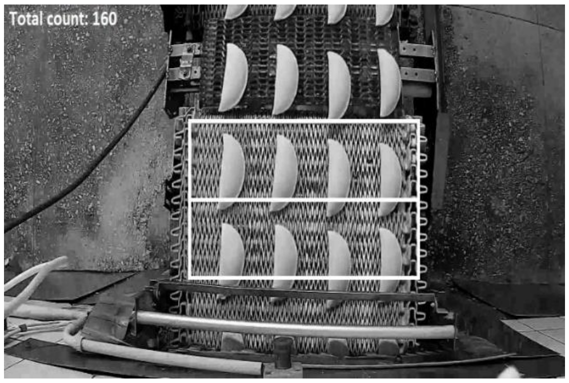


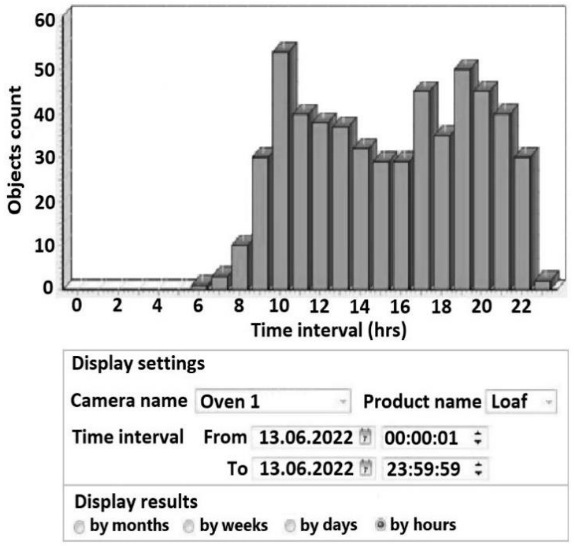
Fig. 6. Video stream window with the region of interest

A database based on SqlLite is used to store the obtained results. This database works without additional settings and stores the processing results on the user's PC. The database contains two tables for directly storing the objects' accounting results and a list of cameras registering certain products. The camera list is used to provide information about video streams and to navigate between them. New records can be entered into it to create new processes, for example, when changing products, and vice versa to connect to existing records, providing information on specific products.

Several classes are used, which simplify interaction with the data to manage the database. The Qt framework provides tools for developing databases, simplifying their setup and maintenance with the QSqlDataBase and QSqlQuery classes. It contains ready QSqlQuery queries necessary for creating tables, adding, and retrieving data. The DBTableController class is designed to manage a separate table. Through it, all fields, types, and the name of the table are specified. They are stored directly in the class body, minimizing all possible errors related to incorrect column names or other data specifications. DBTableController is a wrapper for simplified interaction with a separate database table. In the methods of this class, all simpler operations with any table are performed, and QSqlQuery queries that were previously prepared are executed. In the end, we receive ready-to-use data. This class is primarily universal and works with any data.

The DBController class completes the DB control unit. The DBController class connects to the database, or creates it if there is none on the user's PC, creates the necessary tables, and customizes everything to the application's needs. The functions implemented in this class are required for receiving and recording object account data, providing a list of cameras, and creating a new camera record. All interaction with the database takes place through the controller, and all functionality is encapsulated into this class.

Accounting results for each product can be represented in histograms, as it is shown in Fig. 7. By default, result data are displayed for the current day. Users can display data for any time interval with histograms steps: by months, weeks, days, hours, and for any video channel from which results were recorded.



Display settings
Camera name: Oven 1
Product name: Loaf
Time interval From 13.06.2022 00:00:01
To 13.06.2022 23:59:59
Display results
by months by weeks by days [✓ by hours]

Fig. 7. Moving objects accounting result representation

Also previewing data module contains some functions that allow to save displayed data to XLS, HTML, CSV, and image files.

The software implementation of the algorithms was tested on the three bakery production lines in factory conditions. Moving objects on which the testing took place had different shapes, areas, and textures. In addition, partial occlusions were observed. The test results are shown in Table 1.

As we can see from Table 1, the counting accuracy for different types of objects is more than 99 %; the counting accuracy does not depend on the shape of the object, its texture, and its geometric dimensions.

Table 1

Examples of objects accounting results

Video stream	Average BLoBs' area, px	Accuracy, %
Production line 1	2480	99.2
Production line 2	1560	99.5
Production line 3	1440	99.8

IX. CONCLUSIONS

This paper provided the development of moving objects identification, tracking, and accounting algorithms, as well as the software implementation of these algorithms in the form of a software system for product accounting on bakery production lines. Software implementation and testing have shown that developed algorithms effectively overcome such negative factors as temporary micro-movements, partial occlusions, and speed changes of moving objects. Also, algorithms consider such negative factors as unstable lighting conditions and the presence of moving objects' shadows. This makes video stream processing, in general, robust and stable.

The functioning of the software system is entirely autonomous, and at the same time, the accuracy of accounting is ensured to be over 99 % of the overall number of objects. The software system can also monitor bakery conveyors' idle time, making it possible to optimize production. The software system results are stored in the database and can be presented in the viewing module as histograms or data tables. It is possible to filter results depending on the selected video channel and product type and display results daily, weekly, monthly, and yearly. The developed user interface for changing settings and the database of the automated system are simple enough and do not require additional personnel training to use them.

References

[1] Shi, H., Liu, C. (2018). "A New Global Foreground Modeling and Local Background Modeling Method for Video Analysis", *Machine Learning and Data Mining in Pattern Recognition. MLDM 2018. Lecture Notes in Computer Science, vol 10934. Springer, Cham,* pp. 49-63, DOI: 10.1007/978-3-319-96136-1_5.

[2] Bin, Luo & Bo, Liu & Yu, Zhu. (2020). "The Video Detection of Human Respiratory Motion Based on Sequential Images", *International Conference on Applications and Techniques in Cyber Intelligence ATCI 2019*, pp. 992-997, DOI: 10.1007/978-3-030-25128-4_122.

[3] Firas Hamid (2020). "Study and Analysis Real-Time Methods of Tracking Objects on GPS and Mobile Telephone Towers Stations", *Proceedings of the 1st International Multi-Disciplinary Conference Theme: Sustainable Development and Smart Planning, IMDC-SDSP 2020,* pp. 20-29, DOI: 10.4108/eai.28-6-2020.2298219.

[4] Wang, Y., Idoughi, R., Heidrich, W. (2020). "Stereo Event-Based Particle Tracking Velocimetry for 3D Fluid Flow Reconstruction", *Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science(), vol 12374. Springer, Cham,* pp. 36-53, DOI: 10.1007/978-3-030-58526-6_3.

[5] Jung, K., Kim, Y., Lim, H., Myung, H. (2021). "ALVIO: Adaptive Line and Point Feature-Based Visual Inertial Odometry for Robust Localization in Indoor Environments", *RiTA 2020. Lecture Notes in Mechanical Engineering. Springer, Singapore,* pp. 171-184, DOI: 10.1007/978-981-16-4803-8_19.

[6] Hemalatha, R. *et al.*, (2018). "Active Contour Based Segmentation Techniques for Medical Image Analysis", *in R. Koprowski (ed.), Medical and Biological Image Analysis, IntechOpen, London,* pp. 17-34, DOI: 10.5772/intechopen. 74576.

[7] Shaikh, S., Saeed, K., Chaki, N. (2014). "Moving Object Detection Using Background Subtraction", *SpringerBriefs in Computer Science. Springer, Cham,* pp. 15-23, DOI: 10.1007/978-3-319-07386-6.

[8] Fastiuk Y., Bachynskyy R., Huzynets N. (2021). "Methods of Vehicle Recognition and Detecting Traffic Rules Violations on Motion Picture Based on OpenCV Framework", *ACPS. 2021*; Volume 6, Number 2, pp. 105-111, DOI: 10.23939/acps2021.02.105.

Yurii Ivanov is a Ph.D. and associate professor. In 2004 he graduated from the institute of Power Engineering and Control Systems, and in 2009 from the Institute of Computer Sciences and Information Technologies of Lviv Polytechnic National University. He has an M.Sc. degree in Electrical Machines and Apparatus and Computer Science. In 2005 he obtained his Ph.D. degree at Lviv Polytechnic National University. His research interests are digital video stream processing, web-oriented applications, and database development.

Borys Sharov is a Ph.D. and director of "Electronic Systems" Co. Ltd. He received an engineering education after graduating from Lviv Polytechnic Institute in 1979. In 1992 he obtained his Ph.D. degree at Saint Petersburg Electrotechnical University. He deals with the problems of organizing various real-time systems using video analysis of input signals. He is the author of more than 60 patents and research papers.

Nazar Zalevskyi is a sixth-year student at Lviv Polytechnic National University. His interests include developing applications in C# and Python programming languages using computer vision libraries. In addition, he effectively uses computer vision to achieve significant results in process automation.

Ostap Kernytskyi is a sixth-year student at Lviv Polytechnic National University. He is interested in developing web applications with service-oriented architecture using C# and .NET Framework. He has experience in building REST APIs and self-describing services.