JOURNAL OF UNIVERSAL SCIENCE RESEARCH
ISSN (E): 2181-4570 ResearchBib Impact Factor: 6,4 / 2023SJIF 2024 = 5.073/
Volume-2, Issue-8, 87-101

CAMShift Algorithm for Human Tracking in the Collaborative Robot Working Area

Dmytro Gurin[1], Vladyslav Yevsieiev[1], Svitlana Maksymova[1], Ahmad Alkhalaileh[2]

[1]Department of Computer-Integrated Technologies, Automation and Robotics, Kharkiv National University of Radioelectronics, Ukraine

[2]Senior Developer Electronic Health Solution, Amman, Jordan

Abstract: This article considers the complex implementation of the CAMShift algorithm for human tracking in the collaborative robot working area. The study covers both the algorithmic and mathematical underpinnings of CAMShift, detailing the underlying principles and mathematical models used to improve tracking accuracy. A Python program was developed in the PyCharm environment to effectively implement this algorithm, taking into account aspects such as real-time processing and integration with robotic systems. The research conducted a comprehensive assessment of the tracking speed, studied how effectively the algorithm works in different conditions and how it affects the overall sensitivity of the system. The results demonstrate the effectiveness of the CAMShift algorithm in providing accurate and timely tracking, highlighting its suitability for dynamic and interactive environments. This work helps to optimize the performance of collaborative robots by improving tracking capabilities, enabling better interaction and safety in shared work areas.

Key words: Industry 5.0, Collaborative Robots, Work Area, Computer Vision, CAMShift Algorithm, Tracking People.

Introduction

In today's manufacturing environment, the concept of Industry 5.0 aims to integrate advanced technologies to create more flexible, more efficient and safer work environments [1]-[6]. One of the key aspects of this transformation is ensuring seamless interaction between humans and collaborative robots, which requires accurate and reliable tracking of

human objects in work areas [7]-[17]. Various methods and approaches can be used here [18]-[39].

A software implementation of the Continuously Adaptive Mean Shift (CAMShift) algorithm for human tracking is critical to achieving these goals, as it provides high accuracy and adaptability in dynamic environments. This algorithm allows you to effectively track human movements, which contributes to increasing the safety and optimization of the work of robots, as well as improving the interaction between people and robots. In the conditions of Industry 5.0, where the integration of robots into production processes is key, the research and development of such software solutions are gaining more and more relevance, opening new horizons for improving automation and increasing the overall efficiency of production systems.

Related works

Detecting a person and tracking his behavior in the robot's work area is an extremely important task in collaborative work between a robot and a person. It is natural that research on this topic is constantly updated. Let us consider some of them.

Authors in [40] note, that human safety must be granted avoiding possible collisions with the robot. They propose their own system that is implemented by a camera network system positioned around the robot workspace, and thoroughly evaluated in different industry-like settings in terms of both tracking accuracy and detection delay.

Zaccaria, M., and co-authors in [41] presented for people detection and tracking in automated warehouses. Experiments performed in a real warehouse show the viability of the proposed approach.

The paper [42] examines the reliability of existing state-of-the-art detectors such as Faster R-CNN, YOLOv4, RetinaNet, and Cascade R-CNN on a VisDrone benchmark and custom-made dataset SARD build to simulate rescue scenes.

Researchers in [43] first track persons in the robot coordinate space using Unscented Kalman filter with the ground plane information and human height estimation. Then, they identify the target person to be followed with the combination of Convolutional Channel Features and online boosting.

The article [44] presents a modular detection and tracking system that models position and additional properties of persons in the surroundings of a mobile robot. The proposed system introduces a probability-based data association method that besides the position can incorporate face and color-based appearance features in order to realize a re-identification of persons when tracking gets interrupted.

In crowded human scenes with close-up human-robot interaction and robot navigation, a deep understanding of surrounding people requires reasoning about human motion and body dynamics over time with human body pose estimation and tracking [45]. In this paper [45], authors introduce JRDB-Pose, a large-scale dataset and benchmark for multi-person pose estimation and tracking.

Eppenberger, T., & et al. in [46] present a system for accurate and reliable detection and tracking of dynamic objects using noisy point cloud data generated by stereo cameras. The proposed approach identifies individual objects in the robot's surroundings and classifies them as either static or dynamic. The dynamic objects are labeled as either a person or a generic dynamic object.

De Langis, K., & Sattar, J. in [47] describes a technique that enables autonomous underwater robots to track divers in real time as well as to reidentify them.

CAMShift (Continuously Adaptive Mean Shift) algorithm mathematical representation of the

The CAMShift algorithm is an extension of the Mean Shift algorithm and is used for object tracking in video. CAMShift adapts Mean Shift to changing tracking conditions by updating the search area and scale. Within the framework of these studies, it is proposed to use the following interpretation of the CAMShift algorithm, which is presented in Figure 1.

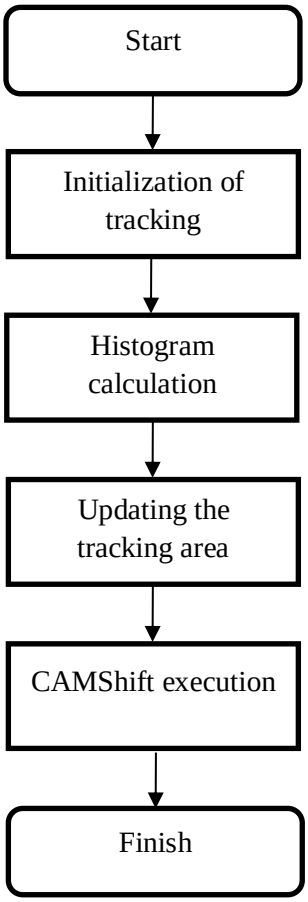

Figure 1: Interpretation of the CAMShift algorithm for use in the identification system of the presence of a person in the collaborative robot working area

Let us describe the purpose of each step of interpretation of the CAMShift algorithm for use in the identification system of the presence of a person in the collaborative robot working area (Figure 1):

- initialization of tracking, at the beginning of tracking, an area (ROI - Region of Interest) is defined, which contains the object for tracking. The color histogram of this object is defined;

- a color histogram calculation is generated based on the ROI area and used for further comparison with the current frames. CAMShift usually uses a histogram in the HSV (Hue, Saturation, Value) color model;

- updating the tracking area, for each new frame a reverse projection image is calculated based on the histogram of the object. This allows you to determine where the object is located in the new frame;

- CAMShift execution (adaptive Mean Shift) adapts the search area according to the change of scale and orientation of the object. It uses an elliptical area that can change its size and orientation to better fit the size and shape of the tracked object.

Le us describe the mathematical representation of the working principle of the CAMShift algorithm, which will be integrated into the developing system for identifying the presence of a person in the collaborative robot working area. Let *H* be the color histogram for the Region of Interest (ROI). A histogram is defined as a frequency distribution for different colors in terms of Hue, Saturation and Value (HSV):

$$H(h) = \frac{count(h)}{total_{count}} \quad (1)$$

H - color histogram for the initial ROI region;
h - color value;
count(h) - number of pixels with color *h*;
$total_{count}$ - total number of pixels.
A reverse projection image *P* is calculated for each new frame *I*:

$$P(x,y) = H(I(x,y)) \quad (2)$$

P - reverse projection image;
(x,y) – pixel coordinates;
I(x,y) - the color value of a pixel in an image.

The Mean Shift algorithm in CAMShift searches for the maxima of the probability distribution in the reverse projection image. It defines the new center of the search region using the following expression:

$$m = \frac{\sum_{x,y} p(x,y) * c(x,y)}{\sum_{x,y} p(x,y)} \quad (3)$$

m - the new center of the search area after performing Mean Shift;
p(x,y) - inverse projection value for a pixel *(x,y)*;
c(x,y) – pixel coordinates.

CAMShift adapts the size and orientation of the search area based on the received coordinates of the center and distribution of the object. This ensures that the search area better conforms to changes in the size and shape of the object:

$$eclipse_{params}=FitEllipse(contours) \quad (4)$$

FitEllipse - the FitEllipse function in OpenCV uses the contours of an object to approximate its shape as an ellipse. It returns several key parameters: center of the ellipse (center), radii (axes), angle (angle).

When performing the CAMShift algorithm, these parameters are used to determine the elliptical region that best fits the object. This allows the algorithm to adjust the shape and size of the search area depending on the dynamics of the object in the frame.

Software implementation of the CAMShift algorithm for human tracking in the collaborative robot working area in Python

The choice of the Python programming language and the PyCharm environment to implement the CAMShift algorithm for human tracking in the workspace of a collaborative robot is justified by several key factors. Python provides simplicity and convenience in writing code thanks to its clear syntactic constructs and powerful image processing libraries such as OpenCV and TensorFlow, which greatly simplifies the implementation of complex computer vision algorithms. Additionally, Python has a large community and plenty of training and support resources to help with any technical issues that arise during development. The PyCharm environment offers advanced tools for debugging and testing code, including integration with version control systems and the ability to work with various libraries. This environment also provides a user-friendly interface and a powerful set of project management tools, which simplifies the development process and allows you to focus on achieving results. The combination of Python and PyCharm provides an efficient and convenient platform for implementing and optimizing the CAMShift algorithm, making it an ideal choice for developing a human tracking system in a complex work environment.

We will give an example of a software implementation of the CAMShift algorithm for human tracking in the collaborative robot working area in Python.

```
import cv2
import numpy as np
import tensorflow as tf
```

Allows you to import libraries necessary for image processing, numerical calculations and work with neural networks. cv2 provides functions for image and video processing, numpy provides capabilities for working with numeric arrays, and tensorflow provides tools for building and training neural networks.

```
model = tf.saved_model.load(r"C:\Users\Vladyslav\.cache\kagglehub\models\
tensorflow\ssd-mobilenet-v2\tensorFlow2\fpnlite-320x320\1")
```

Allows you to load a saved TensorFlow model from the specified path. This allows the already trained model to be used for further prediction or estimation without the need for retraining.

```
    if tracking_window is None:
        for i in range(num_detections):
            if detection_scores[i] > 0.5 and detection_classes[i] == 1:  # Class 1 corresponds to a person
                box = detection_boxes[i]
                y1, x1, y2, x2 = box
                y1, x1, y2, x2 = int(y1 * frame.shape[0]), int(x1 * frame.shape[1]), int(y2 * frame.shape[0]), int(
                    x2 * frame.shape[1])
                tracking_window = (x1, y1, x2 - x1, y2 - y1)
                break
```

This piece of code checks if the tracking scope is initialized. If not, it finds the first object with class "Person" among the detections and sets the tracking area based on the coordinates obtained for that object, converting them to image pixels.

```
    if tracking_window is not None:
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        roi = frame[tracking_window[1]:tracking_window[1] + tracking_window[3],
              tracking_window[0]:tracking_window[0] + tracking_window[2]]
```

This piece of code checks if the tracking scope is initialized. If so, it converts the image to HSV format to facilitate color processing and extracts a region of interest (ROI) from the frame according to the defined tracking area.

dst = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)

ret, tracking_window = cv2.CamShift(dst, tracking_window, (255, 0, 0))

This piece of code performs a color backprojection to determine the tracking area in the new frame using the color histogram from the ROI. The cv2.CamShift function then adapts the tracking area to the new data, updating its position and dimensions for more accurate tracking.

An example of the software implementation of the CAMShift algorithm for tracking a person in the collaborative robot working area in Python is shown in Figure 2.

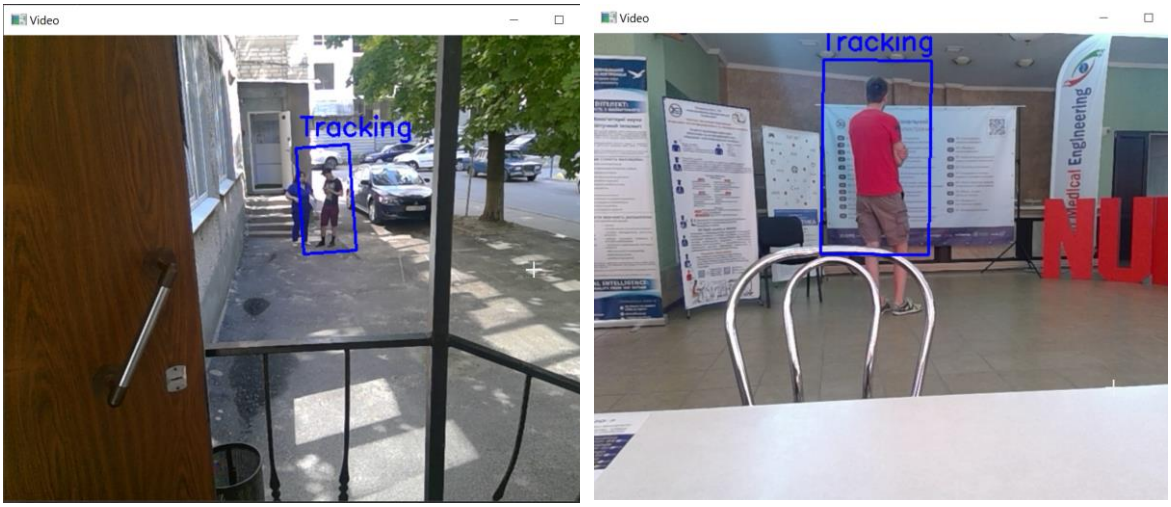

Figure 2: An example of a software implementation of the CAMShift algorithm for human tracking in the collaborative robot working area in Python

Based on the developed program, we will test the speed of human tracking in the collaborative robot working area using the CAMShift algorithm, the results of which are presented in Table 1, and the visualization of the obtained data is presented in Figure 3.

Table 1: The obtained results of testing the speed of human tracking in the collaborative robot working area using the CAMShift algorithm.

Test case	Processing time (ms)	Tracking area size (pixels)	Number of frames per second	Assessment of tracking accuracy
Test 1	45	200x150	22	95%
Test 2	50	250x200	20	92%
Test 3	40	180x130	25	97%
Test 4	55	220x180	18	90%
Test 5	48	210x160	21	94%

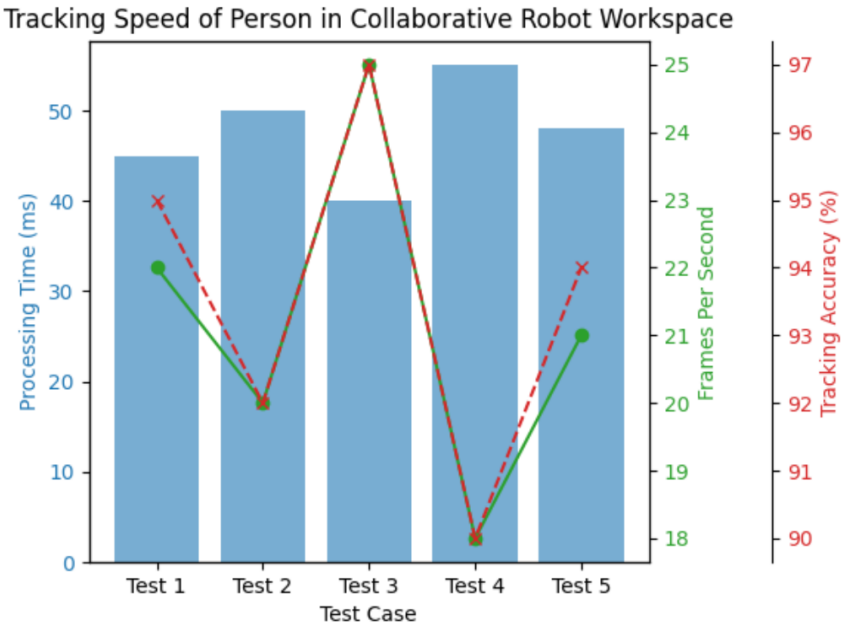

Figure 3: Graph of the obtained results of testing the speed of human tracking in the collaborative robot working area based on the CAMShift algorithm

Analysis of the graph (Figure 3) showing processing time, frames per second, and tracking accuracy score for each test case shows that processing time ranges from 40 to 55 milliseconds, with the lowest value in Test 3 and the highest in Test 4. Accordingly, the number of frames per second varies from 18 to 25, which is the highest in Test 3 and the lowest in Test 4. The tracking accuracy score remains relatively high, ranging from 90% to 97%, with the highest in Test 3 and the lowest in Test 4

.These results indicate an inverse relationship between processing time and the number of frames per second, where longer processing times are often accompanied by a decrease in the number of frames per second, which can affect the overall tracking accuracy. Tests with shorter processing times show higher frame rates and better tracking accuracy, highlighting the importance of algorithm optimization to achieve better real-time results.

Conclusion

Analysis of the results of testing the software implementation of the CAMShift algorithm for tracking a person in the collaborative robot working area showed that the effectiveness of tracking directly depends on the optimization of processing time and the number of frames per second. A graph displaying these parameters along with the tracking accuracy score demonstrates an inverse relationship between processing time and frame rate, where a decrease in processing time is accompanied by an increase in the number of frames per second, which positively affects tracking accuracy. The best results were achieved in tests with shorter processing times, which allows you to maintain high frame rates and tracking accuracy while achieving maximum performance in dynamic environments. This highlights the importance of optimizing the CAMShift algorithm to ensure fast and accurate human tracking, which is critical for the safety and performance of collaborative robots. The test results confirm that in order to achieve optimal results, it is necessary to ensure a balance between processing speed and tracking accuracy, which can be achieved through further adjustment of the algorithm parameters and improvement of the software implementation.

References:

1. Nevliudov, I., Yevsieiev, V., Baker, J. H., Ahmad, M. A., & Lyashenko, V. (2020). Development of a cyber design modeling declarative Language for cyber physical production systems. J. Math. Comput. Sci., 11(1), 520-542.
2. Abu-Jassar, A. T., Al-Sharo, Y. M., Lyashenko, V., & Sotnik, S. (2021). Some Features of Classifiers Implementation for Object Recognition in Specialized Computer systems. TEM Journal: Technology, Education, Management, Informatics, 10(4), 1645-1654.

3. Al-Sharo, Y. M., Abu-Jassar, A. T., Sotnik, S., & Lyashenko, V. (2021). Neural networks as a tool for pattern recognition of fasteners. International Journal of Engineering Trends and Technology, 69(10), 151-160.
4. Matarneh R., & et al. (2017). Speech Recognition Systems: A Comparative Review. Journal of Computer Engineering (IOSR-JCE), 19(5), 71–79.
5. Nevliudov, I., Yevsieiev, V., Lyashenko, V., & Ahmad, M. A. (2021). GUI Elements and Windows Form Formalization Parameters and Events Method to Automate the Process of Additive Cyber-Design CPPS Development. Advances in Dynamical Systems and Applications, 16(2), 441-455.
6. Mustafa, S. K., Yevsieiev, V., Nevliudov, I., & Lyashenko, V. (2022). HMI Development Automation with GUI Elements for Object-Oriented Programming Languages Implementation. SSRG International Journal of Engineering Trends and Technology, 70(1), 139-145.
7. Yevsieiev, V., & et al. (2024). The Canny Algorithm Implementation for Obtaining the Object Contour in a Mobile Robot's Workspace in Real Time. Journal of Universal Science Research, 2(3), 7–19.
8. Abu-Jassar, A., & et al. (2024). The Optical Flow Method and Graham's Algorithm Implementation Features for Searching for the Object Contour in the Mobile Robot's Workspace. Journal of Universal Science Research, 2(3), 64-75.
9. Yevsieiev, V., & et al. (2024). The Sobel algorithm implementation for detection an object contour in the mobile robot's workspace in real time. Technical Science Research in Uzbekistan, 2(3), 23-33.
10. Nevliudov, I., & et al. (2023). Mobile Robot Navigation System Based on Ultrasonic Sensors. In 2023 IEEE XXVIII International Seminar/Workshop on Direct and Inverse Problems of Electromagnetic and Acoustic Wave Theory (DIPED), IEEE, 1, 247-251.
11. Yevsieiev, V., & et al. (2024). Object Recognition and Tracking Method in the Mobile Robot's Workspace in Real Time. Technical science research in Uzbekistan, 2(2), 115-124.
12. Samoilenko, H., & et al. (2024). Review for Collective Problem-Solving by a Group of Robots. Journal of Universal Science Research, 2(6), 7-16.

13. Nikitin, V., & et al. (2023). Traffic Signs Recognition System Development. Multidisciplinary Journal of Science and Technology, 3(3), 235-242.
14. Nevliudov, I. S., & et al. (2023). Conveyor Belt Object Identification: Mathematical, Algorithmic, and Software Support. Appl. Math. Inf. Sci. 17, 6, 1073-1088.
15. Maksymova, S., & et al. (2024). The Lucas-Kanade method implementation for estimating the objects movement in the mobile robot's workspace. Journal of Universal Science Research, 2(3), 187-197.
16. Yevsieiev, V., & et al. (2024). Building a traffic route taking into account obstacles based on the A-star algorithm using the python language. Technical Science Research In Uzbekistan, 2(3), 103-112.
17. Abu-Jassar, A., & et al. (2023). Obstacle Avoidance Sensors: A Brief Overview. Multidisciplinary Journal of Science and Technology, 3(5), 4-10.
18. Deineko, Zh., & et al.. (2021). Color space image as a factor in the choice of its processing technology. Abstracts of I International scientific-practical conference «Problems of modern science and practice» (September 21-24, 2021). Boston, USA, pp. 389-394.
19. Putyatin, Y. P., & et al.. (2016) The Pre-Processing of Images Technique for the Material Samples in the Study of Natural Polymer Composites. American Journal of Engineering Research, 5(8), 221-226.
20. Lyashenko, V., & et al.. (2021). Wavelet ideology as a universal tool for data processing and analysis: some application examples. International Journal of Academic Information Systems Research (IJAISR), 5(9), 25-30.
21. Kobylin, O., & Lyashenko, V. (2014). Comparison of standard image edge detection techniques and of method based on wavelet transform. International Journal, 2(8), 572-580.
22. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 665-670). IEEE.
23. Baker, J. H., Laariedh, F., Ahmad, M. A., Lyashenko, V., Sotnik, S., & Mustafa, S. K. (2021). Some interesting features of semantic model in Robotic Science. SSRG International Journal of Engineering Trends and Technology, 69(7), 38-44.

24. Lyashenko, V., Kobylin, O., & Ahmad, M. A. (2014). General methodology for implementation of image normalization procedure using its wavelet transform. International Journal of Science and Research (IJSR), 3(11), 2870-2877.
25. Гиренко, А. В., Ляшенко, В. В., Машталир, В. П., & Путятин, Е. П. (1996). Методы корреляционного обнаружения объектов. Харьков: АО "БизнесИнформ, 112.
26. Babker, A. M., Abd Elgadir, A. A., Tvoroshenko, I., & Lyashenko, V. (2019). Information technologies of the processing of the spaces of the states of a complex biophysical object in the intellectual medical system health. International Journal of Advanced Trends in Computer Science and Engineering, 8(6), 3221-3227.
27. Vasiurenko, O., Lyashenko, V., Baranova, V., & Deineko, Z. (2020). Spatial-Temporal Analysis the Dynamics of Changes on the Foreign Exchange Market: an Empirical Estimates from Ukraine. Journal of Asian Multicultural Research for Economy and Management Study, 1(2), 1-6.
28. Mustafa, S. K., Ayaz, A. M., Baranova, V., Deineko, Z., Lyashenko, V., & Oyouni, A. A. A. (2020). Using wavelet analysis to assess the impact of COVID-19 on changes in the price of basic energy resources. International Journal of Emerging Trends in Engineering Research, 8(7), 2907-2912.
29. Matarneh, R., Tvoroshenko, I., & Lyashenko, V. (2019). Improving Fuzzy Network Models For the Analysis of Dynamic Interacting Processes in the State Space. International Journal of Recent Technology and Engineering, 8(4), 1687-1693.
30. Khan, A., Joshi, S., Ahmad, M. A., & Lyashenko, V. (2015). Some effect of Chemical treatment by Ferric Nitrate salts on the structure and morphology of Coir Fibre Composites. Advances in Materials Physics and Chemistry, 5(1), 39-45.
31. Kuzemin, O., & Lyashenko, V. Microsituation Concept in GMES Decision Support Systems/A. Kuzemin, V. Lyashenko. Intelligent Data Processing in Global Monitoring for Environment and Security (pp. 217–238).–2011.–P, 217-238.
32. Lyashenko, V., Matarneh, R., & Kobylin, O. (2016). Contrast modification as a tool to study the structure of blood components. Journal of Environmental Science, Computer Science and Engineering & Technology, 5(3), 150-160.
33. Lyashenko V., & et al. (2023). Automated Monitoring and Visualization System in Production. Int. Res. J. Multidiscip. Technovation, 5(6), 09-18.

34. Lyubchenko, V., & et al.. (2016). Digital image processing techniques for detection and diagnosis of fish diseases. International Journal of Advanced Research in Computer Science and Software Engineering, 6(7), 79-83.
35. Lyashenko, V. V., Matarneh, R., Kobylin, O., & Putyatin, Y. P. (2016). Contour Detection and Allocation for Cytological Images Using Wavelet Analysis Methodology. International Journal, 4(1), 85-94.
36. Sotnik, S., & Lyashenko, V. (2022). Prospects for Introduction of Robotics in Service. Prospects, 6(5), 4-9.
37. Al-Sharo Y., & et al. (2023). A Robo-hand prototype design gripping device within the framework of sustainable development. Indian Journal of Engineering, 20, e37ije1673.
38. Abu-Jassar, A. T., Attar, H., Lyashenko, V., Amer, A., Sotnik, S., & Solyman, A. (2023). Access control to robotic systems based on biometric: the generalized model and its practical implementation. International Journal of Intelligent Engineering and Systems, 16(5), 313-328.
39. Al-Sharo, Y. M., Abu-Jassar, A. T., Sotnik, S., & Lyashenko, V. (2023). Generalized Procedure for Determining the Collision-Free Trajectory for a Robotic Arm. Tikrit Journal of Engineering Sciences, 30(2), 142-151.
40. Terreran, M., & et al. (2020). Low-cost scalable people tracking system for human-robot collaboration in industrial environment. Procedia Manufacturing, 51, 116-124.
41. Zaccaria, M., & et al. (2021). Multi-robot multiple camera people detection and tracking in automated warehouses. In 2021 IEEE 19th International Conference on Industrial Informatics (INDIN), IEEE, 1-6.
42. Sambolek, S., & Ivasic-Kos, M. (2021). Automatic person detection in search and rescue operations using deep CNN detectors. Ieee Access, 9, 37905-37922.
43. Koide, K., & et al. (2020). Monocular person tracking and identification with on-line deep feature selection for person following robots. Robotics and Autonomous Systems, 124, 103348.
44. Müller, S., & et al. (2020). A multi-modal person perception framework for socially interactive mobile service robots. Sensors, 20(3), 722.
45. Vendrow, E., & et al. (2023). Jrdb-pose: A large-scale dataset for multi-person pose estimation and tracking. In Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 4811-4820).

46. Eppenberger, T., & et al. (2020). Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 10528-10535.
47. De Langis, K., & Sattar, J. (2020). Realtime multi-diver tracking and re-identification for underwater human-robot collaboration. In 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 11140-11146.