1. Andriy MALYAR, 2. Maksym VIHURO, 3. Valerii MISIURENKO

ORCID: 1. 0000-0001-7735-0592

DOI: 10.15199/48.2025.04.37

Computer vision algorithm for identifying main beer fermentation stages

Algorytm komputerowego widzenia do identyfikacji głównych etapów fermentacji piwa

***Abstract**: The article examines the current issues of beer production related to the yeast foam formation during fermentation, and discusses the importance of controlling the fermentation stages to ensure the proper product quality. Computer vision technologies were applied to identify the stages of the main fermentation. Based on the analysis of the computer vision algorithms, the K-means method was used for image clustering. The systematic description of the algorithm for detecting contaminated foam based on the K-means method is provided.*

***Streszczenie:** W artykule omówiono bieżące problemy produkcji piwa związane z powstawaniem piany drożdżowej podczas fermentacji oraz omówiono znaczenie kontrolowania etapów fermentacji w celu zapewnienia odpowiedniej jakości produktu. Zastosowano technologie wizji komputerowej w celu zidentyfikowania etapów głównej fermentacji. Na podstawie analizy algorytmów wizji komputerowej do klasteryzacji obrazów zastosowano metodę K-means. Przedstawiono systematyczny opis algorytmu wykrywania zanieczyszczonej piany w oparciu o metodę K-means.*

Keywords: computer vision, fermentation, segmentation, K-means method, identification
Słowa kluczowe: widzenie komputerowe, fermentacja, segmentacja, metoda K-means, identyfikacja

Introduction

Machine learning and computer vision technologies [1] are actively used in various manufacturing industries to automate processes and improve product quality. For example, in the automotive industry, the computer vision is applied for checking the quality of welds, detecting defects in parts, and controlling the correctness of assembly. Similar technologies are also widely used in the food industry to monitor product quality at all stages of production, from primary processing to final packaging.

The use of computer vision makes it possible to automatically analyze images obtained with the help of cameras and carry out high-precision quality control without the involvement of specialists. Machine learning algorithms, in particular image segmentation methods, allow quick detection of defects, anomalies or contamination on products, which significantly improves the efficiency of production processes and ensures high quality. This is important for modern high-tech industries.

Beer production is one of the most widespread sectors of the food industry. This is an intricate process comprising multiple stages, each of which is critical for obtaining a high-quality beverage [2]. There are two distinct beer fermentation methods: either utilizing open fermentation vats or closed cylindrical conical tanks. For the purposes of this discussion, we will focus solely on the open vat fermentation. At this stage, it is imperative to maintain optimal fermentation conditions through regulating temperature, pH, etc. [2,3].

It is also important to remove spent dead yeast from the surface of the beer wort to prevent the presence of fermentation by-products in the final product and to ensure high quality of the beer. In this context, the detection and removal of contaminated foam that forms on the surface of the beer at different stages of fermentation is of particular importance.

After completing the brewing process, yeast is introduced to the wort resulting in what is called young beer. Then the primary fermentation commences, during which the yeast transforms monosaccharides into alcohol and carbon dioxide. Young beer undergoes several stages of fermentation, characterized by changes in the metabolic processes of the yeast, in particular in the release of carbon dioxide and heat energy, changes in pH and the degree of fermentation. One of the metrics for controlling the fermentation process is the appearance of the beer wort surface on which foam is formed. Four main stages of fermentation are identified, each of which is characterized by a certain percentage of the contaminated foam [3].

1) Initial. A sign of a successful start is the whitening of the beer. The surface is covered with a thin layer of fine foam. The amount of contaminated foam relative to the total surface area is ≤5% (Figure 1,a).

2) Young krausen. The layer of fine foam becomes deeper and has brown caps. The temperature of the beer rises from 6 °C to 11 °C. The amount of the contaminated foam increases and reaches 30% of the total surface area (Figure 1,b).

3) High krausen. Lasts 1-3 days. The ridges or crests in the foam become higher and the bubbles coarser.. The contaminated foam that appears at this stage should be removed daily to avoid it getting into the beer in the form of flakes. The relative area of contaminated foam that forms daily after regular cleaning is in the range of 30-60% (Figure 1,c).

4) Krausen collapsing. The temperature is gradually reduced by 1 °C per day until it reaches 5 °C. At this stage, the high crests slowly collapse as less carbon dioxide is formed. The foam becomes heavily contaminated and looks browner. The contaminated foam must be removed. Despite daily foam collection, the relative area of contamination is in the range of 60-100%. At this stage, it is important to carefully remove all contaminated foam from the beer wort surface so that it does not get into the next production stage. (Figure 1,d)

Therefore, the development of automated systems capable of an efficient and accurate detection of the main fermentation stages and contaminated foam for its subsequent removal is of topical importance.

Segmentation of the wort surface into clean and contaminated foam is an important metric for determining the current stage of fermentation. Detection of the contaminated foam can serve as a signal for its collection, and can also be part of an automated system for removing the contaminated foam at different stages of fermentation. Such a system will help to ensure a high level of beer quality control and optimize the beer production process.

(a)

(b)

(c)

(d)

Fig. 1: Main fermentation stages: (a) Initial, surface whitening; (b) Young krausen; (c) High krausen; (d) Krausen collapsing, end of fermentation

Overview of Computer Vision Methods

The automation and development of projects aimed at detecting and cleaning contaminated foam in open fermentation tanks is an important step in improving the quality and efficiency of beer production. Traditional methods of quality control in this process require a significant involvement of the human factor [4], which leads to a certain risk of error and lack of efficiency. However, recent developments in computer vision [5] have made it possible to use automated systems to detect and control the quality of beer in open fermentation vats.

There are a number of modern computer vision technologies that can be used to solve the problem of detecting beer fermentation stages based on images of the beer wort surface. Several areas can be distinguished:

1. Semantic segmentation based on deep neural networks [6-13]. The two popular architectures are U-Net [6] and Mask R-CNN [7]. The U-Net architecture is effective in the tasks of segmenting microscopic images, medical images, and other biomedical research. The Mask R-CNN architecture is an improved algorithm that extends the popular Faster R-CNN algorithm [8] for object detection and segmentation tasks. However, Mask R-CNN requires additional computational resources.
2. Motion tracking methods [14] are designed to detect and track moving objects. They are widely used in robotics, video analysis, auto piloting, video games, etc. These methods can determine the speed of movement of objects, which is useful for analysing the course of the fermentation process.
3. Object localisation based on object detection methods. YOLO (You Only Look Once) [15] and SSD (Single Shot MultiBox Detector) are popular [16]. The YOLO algorithm uses a slightly different approach to detection than the traditional ones. The idea is that the neural network processes the input image only once. The SSD algorithm, like YOLO, processes the image once. However, it uses different feature maps with different resolutions, which increases the resolution of the image and therefore the complexity of the real-time computation.
4. The analysis of texture features and image structure is based on the use of methods such as Local Binary Patterns and Histogram of Oriented Gradients [12, 17]. These methods are designed to detect an object in an image and check whether it belongs to a particular class.

All the methods described above can be used for input image identification using certain features. However, foam formation during fermentation is a random process and therefore there are no identical images for the same stage, which complicates the initial training of the convolutional neural network and makes the application of the above algorithms difficult.

In our opinion, the task of recognising the fermentation stages can be solved by segmenting the surface of the beer wort into contaminated and white foam, the percentage of which characterizes each stage. For this task, the K-means algorithm was selected [9-11]. It allows the image to be divided into clusters according to the similarity of the colours that characterise the contaminated foam on the surface of the wort. In addition, the algorithm is simple and can work effectively with large data sets without prior training of the network. This enables a fast adaptation of the algorithm to new conditions and different types of wort in real time, which is an important factor in the beer production process.

Algorithm for detecting the main fermentation stages

The use of the open fermentation vat in beer production has certain disadvantages. One of them is the direct contact

of the wort with the environment, which imposes special hygiene requirements to the process of contaminated foam removal. A high quality and timely cleaning of the wort can be ensured only with the help of intelligent automated systems [18]. Such a system should contain several components:

- A computer vision algorithm is required to detect and evaluate contaminants in the wort and activate the cleaning system promptly.
- A robotic cleaning system should be designed to remove the contaminated foam from the surface of the wort in various vat sizes and operating conditions with ease.
- A control system for electric drives should offer an accurate and coordinated control of the manipulator movement and ensure safe operation. The control ought to rely on information gathered from the computer vision algorithm and consider the particulars of the beer production setting.

To successfully automate the beer wort cleaning during the main fermentation process, an efficient algorithm for detecting each stage needs to be devised.

To address this issue, an algorithm utilising the Open CV library [19], with a range of standard image processing functions, was developed in the Python programming language. The algorithm requires an image of the beer wort surface during fermentation as an input.

The algorithm comprises the subsequent steps:

1. Uploading and preparing the image.

The input files can consist of JPEG, PNG or TIFF images depicting the surface of young beer (Figure 2, a). The images must have a resolution of 960×1280 pixels and must be in the RGB colour model. Consequently, each image is depicted by a 960×1280 matrix with each element containing an array of three colour channel values, ranging from 0 to 255. After reading a colour image, it is converted to black and white for further processing. This conversion is implemented using the cv2.cvtColor() function with the cv2.COLOR_BGR2GRAY parameter. The resulting matrix is of the same size as the original one and consists of single values ranging from 0 to 255. These values represent the intensity of the pixel, as opposed to the original three RGB values per pixel.

2. Define the boundaries of the elements in the image.

To prepare the beer wort surface image for further processing, the background is removed, leaving only the surface in view. To achieve this, we apply a Gaussian filter [20] and a median filter [21] to blur the monochrome image. The outcome is an image with blurred edges. The function cv2.absdiff() computes the absolute difference between every corresponding matrix element of a grayscale image and a grayscale image which has been blurred. When examining the middle of an object within the image, the absolute difference will be almost zero, despite the blurring. However, near the edges of the objects, as a result of differences in their colour intensity, the cv2.absdiff() function will output non-zero values. The processed image is binarised using the cv2.threshold() function. This function sets all values less than the experimentally determined threshold "9" to black "0" and those greater than the threshold to white "255". Consequently, an image with a black background and white object boundaries is generated.

3. Separating the surface area of the young beer.

From the binary image obtained in the previous step of the algorithm, object contours need to be defined. To achieve this, the cv2.findContours() function is utilized. Afterward, each object's area is calculated. The object exhibiting the largest area corresponds to the surface of the young beer. Next, a black mask is produced to match the size of the input image. The outline of the beer wort surface is then transferred onto it and subsequently filled with white. Next, a black output image of the same dimensions is produced. The pixels from the input image that correspond to the location of the white pixels in the black-and-white mask image are then transferred to it. This enables highlighting the surface of the young beer against the black background. To improve image segmentation, the black background is replaced with a green one (Figure 2, b).

(a)

(b)

Fig. 2. (a) Surface of beer wort (input image); (b) surface of beer wort without background

4. Colour segmentation.

The image produced is transformed into an array wherein each element corresponds to a single pixel composed of three channel values from the Pi(Ri, Gi, Bi) colour model. Subsequently, these values are represented through floating point numbers.

The original image requires segmentation into three clusters: the green background (the metal vat and the wall), white uncontaminated foam, and brown contaminated foam. To accomplish K-means segmentation, we developed an algorithm (Figure 3) that includes the following steps:

I. Three centroids are selected randomly, denoted as $C_k(R_{ck}, G_{ck}, B_{ck})$, where k = 0,1,2 defines the ordinal position of each centroid. R_{ck} corresponds to the red channel of the centroid's colour, G_{ck} to the green channel, and B_{ck} to the blue channel, respectively.

II. The RGB model is formed by adding three colours and can be presented as a coordinate system having three axes representing these colours. This colour space's coordinates are bounded evenly, with the values ranging from 0 to 255. Then, the distance between each pixel's coordinates and the three centroids is measured using the formula (1).

$$d_{c_{ki}} = \sqrt{\left(R_{c_k} - R_i\right)^2 + \left(G_{c_k} - G_i\right)^2 + \left(B_{c_k} - B_i\right)^2} \quad (1)$$

where i= [0, 1, ... 1254399] is the current pixel number.

The outcome is an array comprising the ordinal numbers of the centroid with which the distance d_{ck} is the shortest

III. The mean average of all the pixels that belong to a cluster with each centroid in the three colour channels is calculated using the formula (2). These will be the updated centroids.

$$R_{c_k} = N_k^{-1} \sum_{i \in k} R_{i \in k} \; ; \quad G_{c_k} = N_k^{-1} \sum_{i \in k} G_{i \in k} \; ; \quad B_{c_k} = N_k^{-1} \sum_{i \in k} B_{i \in k} \; , \quad (2)$$

where N_k is the number of pixels associated with centroid k.

IV. By repeating steps II and III, the algorithm improves the accuracy of clustering until the stopping criterion is met. For the K-means method, the stopping criterion is the fulfilment of the inequality $(C_{ka} - C_{ka-1}) \leq \varepsilon$ or the performance of the maximum number of iterations a_{max} (Figure 3).

Next, applying the formula (3), minimum distances to the corresponding S_k clusters and their sum (4) need to be determined.

$$S_k = \sum_{j=0}^{N_k} \left(C_k - P_i\right)^2 \; , \quad (3)$$

$$S_{new} = \sum_{j=0}^{k} S_j \; , \quad (4)$$

where S_k is the aggregate distances from pixels P_j to the closest centroid C_k;

P_j is the vector of elements that pertain to the k-th centroid;

S_{new} is the sum of the minimum distances to the centroids C_k.

As clustering is a non-linear programming problem, step IV results in the first local minimum calculated using the formula (4). The S_{new} value of this minimum serves as the initial value for the first iteration or is compared with the value obtained in the previous iteration. If the current minimum is smaller, the previous value is replaced with it.

As the centroids' initial centres are chosen randomly, the algorithm calculates the local optimal centroid value in one cycle (steps I÷IV). Consequently, image clustering should be carried out multiple times for the centroid values to become stable. It was determined experimentally that the total of 10 different randomly selected sets of centroids are needed to achieve the global minimum of the sum of distances S_{min}. Therefore, we chose a constant number of cycles p=10. The set of centroids and the pixels belonging to them are chosen according to the smallest sum of distances S_{min} and do not depend on the iteration at which this sum was achieved. As a result, we obtain the centroids and an array of elements with their ordinal numbers, where the total distances correspond to the global minimum (4).

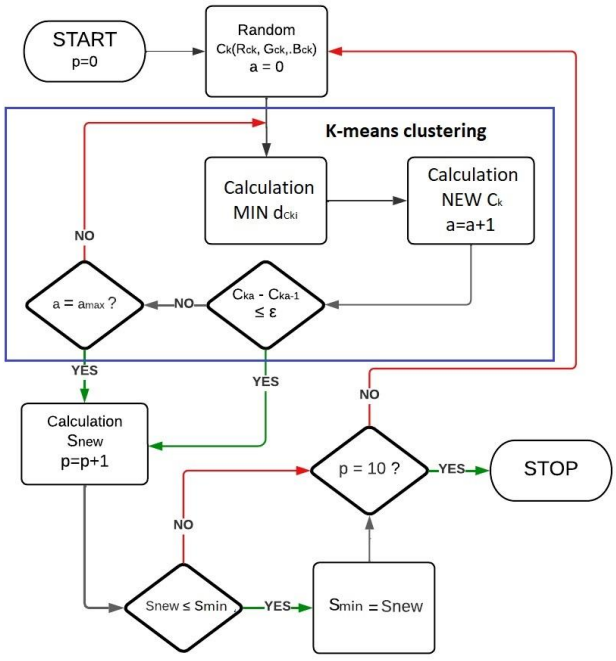

Fig. 3a: Segmentation algorithm: block diagram

```
-----------------------------------------------
    SEQUENCE
      // Step 1: Uploading and preparing the image.
      READ image (input_image)
      CONVERT image to grayscale (grayscale_image)
      // Step 2: Define the boundaries of the elements in the image
      APPLY Gaussian blur to grayscale_image (blurred_image)
      APPLY Median filter to blurred_image (background_image)
      COMPUTE the difference between grayscale_image and background_image (diff_image)
      APPLY thresholding to diff_image with a threshold (threshold_image)
      // Step 3: Separate the surface area of the "young beer".
      FIND contours in threshold_image (contours)
      SELECT the largest contour corresponding to the surface of the beer wort (largest_contour)
      CREATE a mask (mask) for the beer wort surface
      APPLY mask to input_image to isolate the beer surface (output_image)
      // Step 4: Colour segmentation (K-means clustering).
      FOR iteration = 1 to 10:
          // Step 4.1: New random centroids
        INITIALIZE 3 random centroids C0, C1, C2 (random_centroids)
          REPEAT (until centroids stabilize or max iterations reached):
        // Step 4.2: Assign each pixel to the nearest centroid
        FOR each pixel in output_image:
          CALCULATE the distance to each centroid using formula (1):
          ASSIGN the pixel to the nearest centroid (C0, C1, or C2)
        END FOR
        // Step 4.3: Compute the new centroids for each cluster
        FOR each centroid Ck:
          COMPUTE the new centroid Ck using formula (2):
        END FOR
       // Check stopping criteria
        IF |Ck_new - Ck_old| ≤ ε OR maximum iterations reached THEN:
          BREAK REAPEAT // Exit the loop
        // Step 4.4: Compute the new distances to the centroids for each pixel
        COMPUTE the distance to the new centroids for each pixel using formula (3):
        COMPUTE the total sum of distances (S_new) using formula (4):
        // Step 4.5: If the sum of distances is smaller than the previous iteration, update centroids
        IF S_new < smallest_sum THEN:
          UPDATE previous_centroids = current_centroids
          UPDATE smallest_sum = S_new
        END IF
      // Step 4.6: Continue to the next iteration if max iterations are not reached
      IF iteration < 10 THEN:
        CONTINUE // Move to the next iteration
      END FOR
      // Step 5: Processing the results of the segmentation algorithm.
      CONVERT centroids from floating point to integer format (np.uint8)
      SEGMENT the image using the updated centroids
      DISPLAY segmented_image
      // Calculate the number of pixels for each class
      CALCULATE green_count = Count the green pixels
      CALCULATE brown_count = Count the brown pixels
      CALCULATE white_count = Count the white pixels
      // Calculate the percentage of contaminated foam
      CALCULATE brown_area = (brown_count / (brown_count + white_count)) * 100
      // Step 6: Check if the contaminated foam exceeds the threshold
      IF brown_area > threshold THEN:
        PRINT "Contaminated foam area:" brown_area "%"
        ACTIVATE cleaning system
      ELSE:
        PRINT "Contaminated foam area within acceptable limits:" brown_area "%"
      ENDIF
    -----------------------------------------
```

Fig. 3b: Segmentation algorithm: pseudocode

V. Processing the results of the segmentation algorithm.

As the K-means method's centroid coordinates are in floating point values, they need to be converted into an integer format to display the image. The conversion process employs the np.uint8() function, which transforms the input centroid coordinates into unsigned 8-bit integers.

Results

The experiment aimed to analyse the surface of the beer wort shown in Figure 2a. This image, which was obtained experimentally during the technological process, shows the stage of active fermentation of beer, which is characterized by the ratio of contaminated foam to the total surface area in the range of 30-60%. The algorithm developed (Figure 3) has generated a multidimensional array that comprises three colour clusters: red, green and blue.

The resultant array was then transformed into a one-dimensional array using the labels.flatten() function, which accurately represents the output image. Each element of this array represents a pixel displayed in the format (Rk, Gk, Bk), where Rk, Gk, Bk are the colours of the pixel

corresponding to the centroid of the k-th cluster in the image. This enables effective processing, storage and utilisation of such images. Thus, a segmented image is obtained (Figure 4), wherein every pixel is depicted by the colour of its associated cluster.

To ascertain the corresponding stage of the primary fermentation, the number of pixels that correspond to each of the three designated colours in the segmented image needs to be calculated. Then, the percentage of the brown pixels representing the colour of the contaminated foam, compared to the sum of the brown and white pixels, is determined. This method enables the calculation of the ratio of the contaminated foam area to the young beer surface area, expressed as a percentage.

For the given example (Figure 4), 48.52% of the surface area is contaminated. This percentage of foam corresponds to the stage of active fermentation of beer, and therefore the result of the algorithm has been confirmed experimentally.

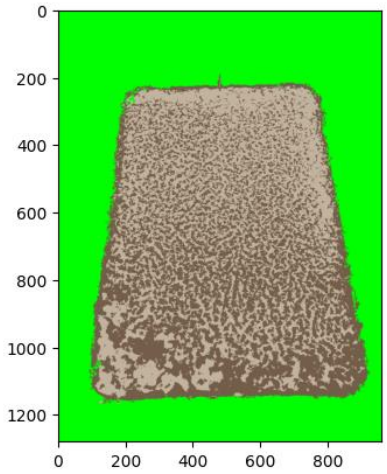

Fig. 4: Segmentation algorithm

Based on expert analysis, the maximum allowable level of contaminated foam of young beer has been set at 40%. If the quantity exceeds this limit, the contaminated foam must be removed by an automated system.

Conclusions

Using the computer vision technology, an algorithm was developed to detect the various stages of primary beer fermentation by segmenting the images of the surface of young beer. The accuracy and speed of input image processing are high due to the employment of the K-means method.

The algorithm facilitates the determination of the level of beer contaminated foam by calculating the relative area of the contaminated foam. As demonstrated in the example provided, this area accounted for 48.52% of the surface. The algorithm is a reliable tool for automating the process of wort cleaning and analysis of fermentation stages. This reduces the risk of beer contamination with yeast foam and improves beer production quality.

This algorithm is planned to employed as an integral part of the beer production control system. Specifically, it can be utilised to detect and prevent emergencies and gather statistical data on the course of fermentation processes in the production environment.

The implementation of such algorithms will promote innovation and implementation of modern technology in the beer industry.

***Authors**: Andriy Malyar, D.Sc, prof., Lviv Polytechnic National University, Lviv, Ukraine. E-mail: andrii.v.maliar@lpnu.ua; Maksym Vihuro, PhD student, Lviv Polytechnic National University, Lviv, Ukraine. E-mail: maksym.i.vihuro@lpnu.ua, Valerii Misiurenko, PhD, Associate Professor, Lviv Polytechnic National University, Lviv, Ukraine. E-mail: valerii.o.misiurenko@lpnu.ua*

REFERENCES

[1] John D. Kelleher, Brian Mac Namee, Aoife D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies, MIT Press, (2015), 624.
[2] H. M. Esslinger, Handbook of brewing: Processes, technology, markets, Wiley-VCH, Weinheim, (2009), 779.
[3] W. Kunze, Technology brewing and malting, 3rd. ed., VLB, Berlin, (2004), 948.
[4] Jevšnik, M., & Raspor, P. The human factor as a trigger of food safety culture within food networks: the review. *Acta Microbiologica Bulgarica*, (2020), no. 36(4), 115-131.
[5] Xin Feng, Youni Jiang, Xuejiao Yang, Ming Du, Xin Li, Computer vision algorithms and hardware implementations: A survey, *Integration*, vol.69, (2019), 309-320, doi: 10.1016/j.vlsi.2019.07.005.
[6] O. Ronneberger, P. Fischer, T. Brox. U-Net: convolutional networks for biomedical image segmentation, in: Lecture notes in computer science, Springer International Publishing, Cham, (2015), 234-241. doi:10.1007/978-3-319-24574-4_28.
[7] K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask R-CNN, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2020), no.42(2), 386-397. doi:10.1109/tpami.2018.2844175.
[8] Ren, Shaoqing, Kaiming He, Ross B. Girshick and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no.39, (2015), 1137-1149. doi: 10.48550/arXiv.1506.01497.
[9] Ahmed, A. S., Basheer, O. N., & Salah, H. A. Breast Tumors Diagnosis Using Fuzzy Inference System and Fuzzy C-Means Clustering. *International Journal of Computing*, (2021), no.20(4), 551-559. doi.org/10.47839/ijc.20.4.2443
[10] Lucchese, L., & Mitra, S. K. Colour image segmentation: a state-of-the-art survey, *Proceedings-Indian National Science Academy*, Part A, (2001), no.67(2), 207-222.
[11] Malav, Amita, Kalyani Kadam, and Pooja Kamat. Prediction of heart disease using k-means and artificial neural network as hybrid approach to improve accuracy, *International Journal of Engineering and Technology*. vol.9(4), (2017), 3081-3085. doi: 10.21817/ijet/2017/v9i4/170904101.
[12] C.-C. Liu, Y.-C. Zhang, P.-Y. Chen, C.-C. Lai, Y.-H. Chen, J.-H. Cheng, M.-H. Ko, Clouds classification from sentinel-2 imagery with deep residual learning and semantic image segmentation, *Remote Sensing*, vol.11(2), (2019), 119. doi:10.3390/rs11020119.
[13] Z. Wu, Y. Xiong, S. X. Yu, D. Lin, Unsupervised feature learning via non-parametric instance discrimination, in: 2018 IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, 2018. doi:10.1109/cvpr.2018.00393.
[14] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. V. D. Smagt, D. Cremers, T. Brox, FlowNet: learning optical flow with convolutional networks, in: *2015 IEEE international conference on computer vision (ICCV)*, IEEE, (2015). doi:10.1109/iccv.2015.316
[15] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, IEEE, (2016). doi:10.1109/cvpr.2016.91.
[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, SSD: single shot multibox detector, in: *Computer vision – ECCV 2016*, Springer International Publishing, Cham, (2016), 21-37. doi:10.1007/978-3-319-46448-0_2.

[17] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, IEEE, (*2005*). doi:10.1109/cvpr.2005.177.
[18] A. Malyar, A. Andreishyn, B. Kaluzhnyi, I. Holovach. Study of the hamming network efficiency for the sucker-rod oil pumping unit status identification, *Computational Problems of Electrical Engineering*, vol.7(1), (2017), 45-50. doi: 10.23939/jcpee2017.01.045
[19] A. Kaehler, G. Bradski, Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library, O'Reilly Media, Incorporated, (2016), 99-180.
[20] H. Strasburger, M. Bach, S. P. Heinrich, Blur Unblurred. A Mini Tutorial, *i-Perception,* vol.9(2), (2018), 1-15. doi; 10.1177/2041669518765850.
[21] Boateng, Kwame & Weyori, Benjamin & Laar, David. Improving the Effectiveness of the Median Filter, *International Journal of Electronics and Communication Engineering*, no.5, (2012), 85-97.