Electronic Letters on Computer Vision and Image Analysis 20(2):113-143, 2021

Application of computer vision to egg detection on a production line in real time.

Maciej Ulaszewski*, Robert Janowski* and Andrzej Janowski+

* *Warsaw School of Computer Science, Lewartowskiego 17, Warsaw, Poland*
+ *Independent Scholar, Czaplin 55, Poland*

Received 3rd of February 2021; revised 1st of October 2021; accepted 26th of December

Abstract

In this paper we investigate the application of computer vision to the problem of egg detection on a production line in real-time. For this purpose a dedicated software using Python3 and specialized libraries was designed and implemented that exploited the advantages of neural networks or template matching approaches. For neural network 3 models were investigated: SSD-Mobilenetv2, FR-CNN and YOLOv3. The effectiveness and performance of the egg detection, tracking and counting process were evaluated for these models and the template matching method. The sensitivity of the effectiveness and performance to the number of objects in a video frame, to film resolution and to size differentiation among the eggs were within the scope of the evaluation. An improvement was proposed relying on processing only every n-th video frame and optimal value of n that raised the performance without loss of effectiveness was investigated. In case of template matching method, the impact of cascade templates on the effectiveness and performance were also verified. Finally, the recommendation was done regarding the methods and their configuration applicable to be utilized on a production line in a real industrial environment with strict real-time requirements.

Key Words: Computer vision, OpenCV, neural networks, template matching, LaTeX.

1 Introduction

In recent years computer vision has gained a lot of attention and popularity among business or industry clients. It is not only due to the research achievements in the area of neural networks, machine learning or deep learning [1] but more of the other the result of the wide availability of software tools that aid the process of exploiting the computer vision techniques in practice. To name a few it suffices to mention TensorFlow for machine learning [2], [3], [4], Darknet [5] for training neural networks and OpenCV offering a library of functions to be used for developing a computer vision applications in a number of languages [6], [7]. There are also ready to be used neural networks structures for object detection or recognition featuring different vision methods like single step analysis [8] e.g. Single Shot Detector (SSD) [9] or double step analysis [8] e.g. Region Based Convolutional Neural Networks (R-CNN). These tools make the software development much more accessible for a wide range of programmers. They do not necessarily have to dive deeply into the theoretical aspects of machine learning

Correspondence to: rjanowski@poczta.wwsi.edu.pl

Recommended for acceptance by Angel D. Sappa
https://doi.org/10.5565/rev/elcvia.1390
ELCVIA ISSN:1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain

or neural networks areas and create applications from the scratch by developing their every element e.g. neural network structures but can exploit existing tools with handy Application Programming Interface (API). This lets them focus on the basic functionality of the developed applications making the development process faster and more comfortable. However, the correct choice of tools and methods that could be exploited might be crucial for the effectiveness and performance of a computer vision application. This choice depends on the industry or business segment an application is being developed for because it implies different requirements toward input data and expected results. In our work we have chosen to develop an application for detecting, tracking and counting eggs in camera videos taken on a production line in a breeder farm. The aim of our work is to investigate what methods are applicable for that problem to assure high effectiveness of the visual detection, tracking and counting eggs and on the other hand some minimum performance level imposed by the requirement of providing the output in real-time. The outline of the remaining part of the paper is as follows: in section 2 we formulate the research problem. In section 3 we provide the details of the application development process. In section 4 we describe the experiments that were conducted and their results covering different aspects of object detection and tracking on a moving production line. In section 5 we summarize the paper formulating some valuable conclusions regarding the applicability of different neural networks structures and picture detection methods to successfully achieve previously stated targets.

2 Statement of the problem

The idea to develop an application for visual egg detection, tracking and counting on a production line has been inspired by the fact that today, there are many breeder farms producing more than million eggs every month and the counting process of this huge amount of eggs for obvious reasons is supported by the hardware counters placed at the end of a production line. However, there are at least two constraints of hardware counters. First of all, they provide only current information that is displayed on Liquid-Crystal Display (LCD) screens so memorizing it requires writing this information by a human operator. The second constraint is related to the placement of the hardware counter at the end of a production line in a so called aggregation point. This precludes counting the eggs that enter the production line from particular poultry houses that is a highly desired information letting to establish the real productivity of hens in particular poultry houses (see Figure 1).

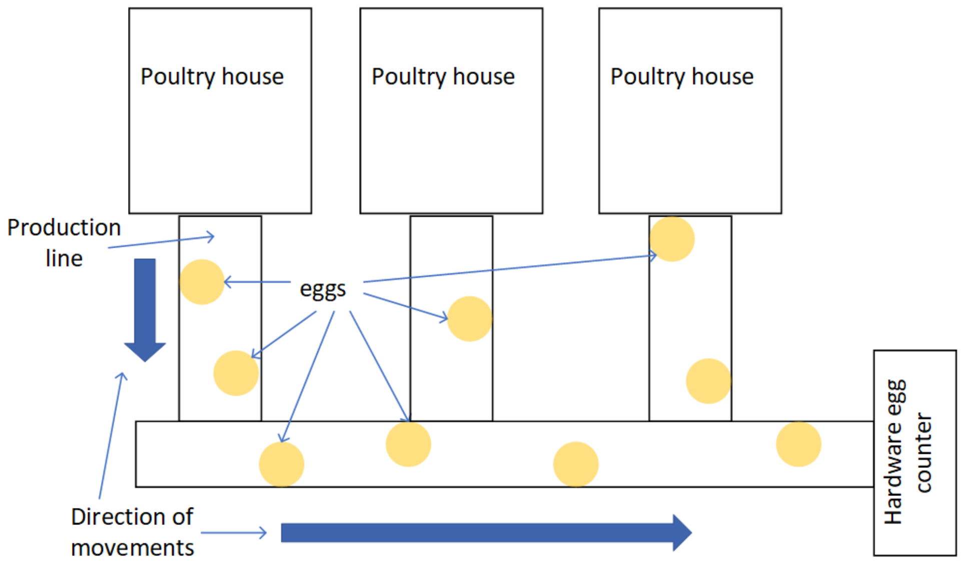


Figure 1: The organization scheme of a main production line with branches coming from particular poultry houses.

The proposed application for counting eggs based on the visual detection in camera videos taken from a production line alleviates the above problems. It lets for counting the eggs and storing the results in a database with desired time granularity for further analysis. It also lets to count the eggs in any point of a production line where the camera is currently pointing. Thus the eggs leaving a particular poultry house and entering the aggregate production line can also be counted. For the application to be useful it must provide the results understood as the number of detected eggs in a real-time. It means that analyzing the current video frame should not last more than the inter-frame period. Otherwise the new frame arrives before the analysis of the previous one is finished and a backlog builds up. This is unacceptable from the point of view of its application as a supporting tool for the production process. Commonly used inter-frame periods in videos from a camera are 25-40ms and this formulates the minimum requirement for the performance level. Inter-frame period of 40ms implies 25 frames per second as a minimum performance level.

The above are the most important business drivers for developing a software application for visual egg detection, tracking and counting on a production line. However, we have also defined few research targets for our work. They are as follows:

- Determination of what methods (with and without the involvement of the neural networks) and models (of neural networks) are suited for the visual detection of eggs on the production line.
- Evaluation of the performance vs the number of objects to be detected on a picture (video frame).
- Evaluation of the performance vs processor type (Central Processor Unit or Graphics Processor Unit).
- Evaluation of the performance vs detection method (different types of neural networks vs template matching).
- Verification of the effectiveness of template matching method with cascaded filters (templates).
- Verification of the effectiveness and performance of neural network methods while skipping the analysis of some video frames.

Before starting our work we have analyzed the state of the art in this area. We were looking for similar applications or the research to understand what was already done, discovered and documented and to what extent our work can shed a new light on this topic. A direct motivation for our work comes from the fact that the automation of everyday tasks in agriculture industry, especially in the poultry business, is desirable yet feasible thanks to development of computer vision methods. The paper [10] summarizes applications of the computer vision in poultry business and among the others it mentions egg production and collection system as one of them. For the choice of methods we followed the conclusions pointed out in [11] that stated the CNN methods were widely applicable and effective in detecting different types of objects including animals. In [12] the authors developed a faster R-CNN detector to detect dairy goats in a surveillance video and obtained very good results in terms of precision. In [13] the authors applied CNN networks to pig standing and lying posture detection under commercial farm conditions also with very satisfactory results in terms of detection precision. Since the problem of egg detection seems to be an easier task due to the more regular shape of the eggs than animals, we expect the promising results applying the same methods to the problem of egg detection. This reasoning was apparently a guideline for the authors of [14] where the problem of egg detection was studied. In [14] the authors evaluated few CNN networks to quantify their ability and performance of detecting hen eggs lying on the floor. The feature in common with our work is the application of SSD or faster R-CNN networks. However, that paper focused on the problem of egg detection without the need to track them that made a difference with our work. The work described in [15] has some similarities to our work as its aim was to detect, count and track hens bred in cages. The tracking algorithm was similar to ours and it used the distance between the hens detected on the consecutive video frames to decide whether it was the same or another hen. Moreover, the detection of hens was accomplished using CNN networks, the same approach that we used. Besides mentioned literature, we have found 3 more works that treated similar problem as we did i.e. the egg detection [16], detection and counting [17] or detection, tracking and counting on a production line [18]. In [16] the authors have investigated the application of computer vision to determine the dimensions of hen eggs. The investigated method covered 5 steps: 1) Inserting the image into the program and cropping it, 2) Converting an image with RGB (Red Green Blue) color model to grayscale, 3) Converting a grayscale image to a black and white image, 4) Calculation of geometric parameters of eggs such as surface size, length, width, radius and circumference, 5) Determining the size of the egg on the basis of geometric parameters. The authors conducted research for 120 photos. The effectiveness of size determination was 94%. For comparison, determination of egg sizes based on their weight according to the authors is only 44% effective. This shows the potential of image detection in the manufacturing or production processes. However, it is worth noting that in the photos used for the research the eggs were placed on a uniform background which made it possible to easily determine the shape of the objects. In the case of the videos used in our work, typically the conveyor belt of the production line has bulges and indentations to reduce the movement of the eggs along the belt. This could cause a problem in the program to extract the exact egg shape. Moreover in the research presented in [16] eggs were only recognized in photos so they were motionless. There are several videos available in the Internet on how to recognize and count eggs on the production line. However, for any of them the authors did not provide details of the method used or the source code. The pictures from these videos are presented below with brief comments.

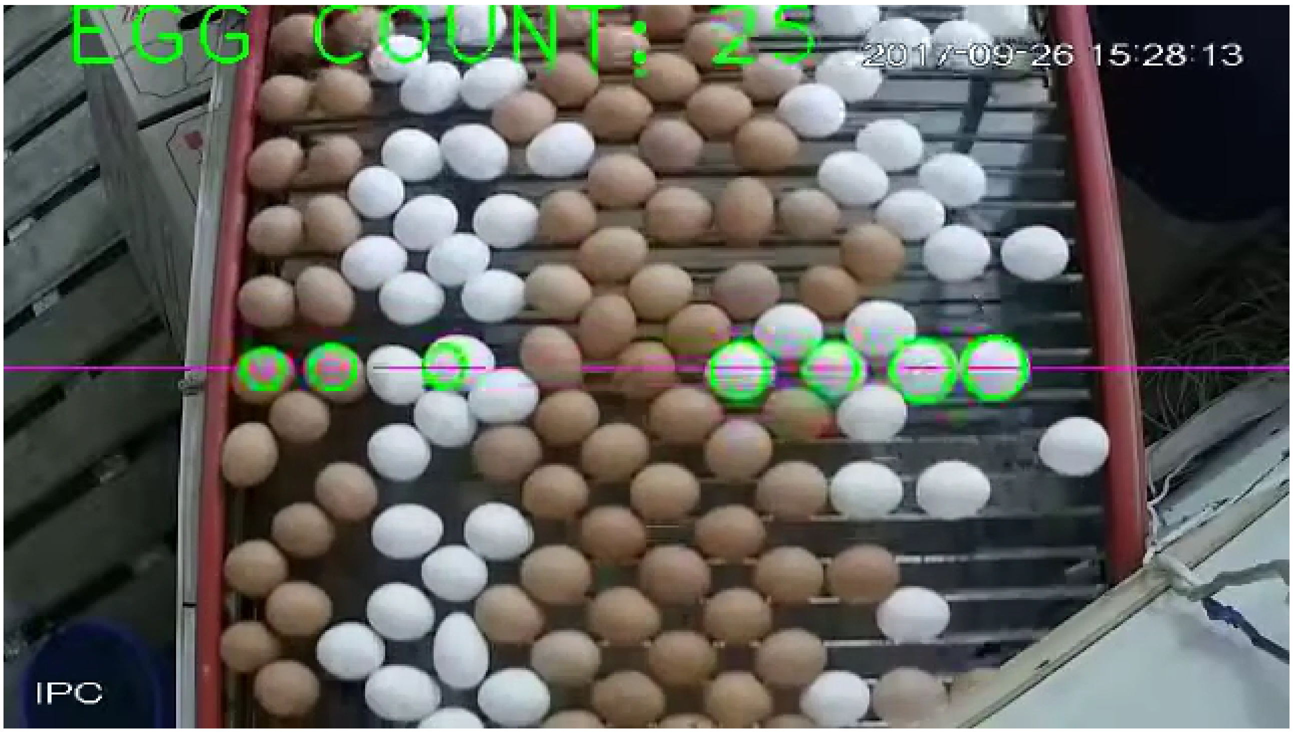


Figure 2: The video 'Test of egg detection with opencv' [17].

According to the title of the film [17] the egg detection process exploited OpenCV library. As the eggs were marked and counted in the center of the screen this might suggest that there was no on-screen tracking module. The detected eggs were marked with envelopes similar to the shapes of the objects they described.

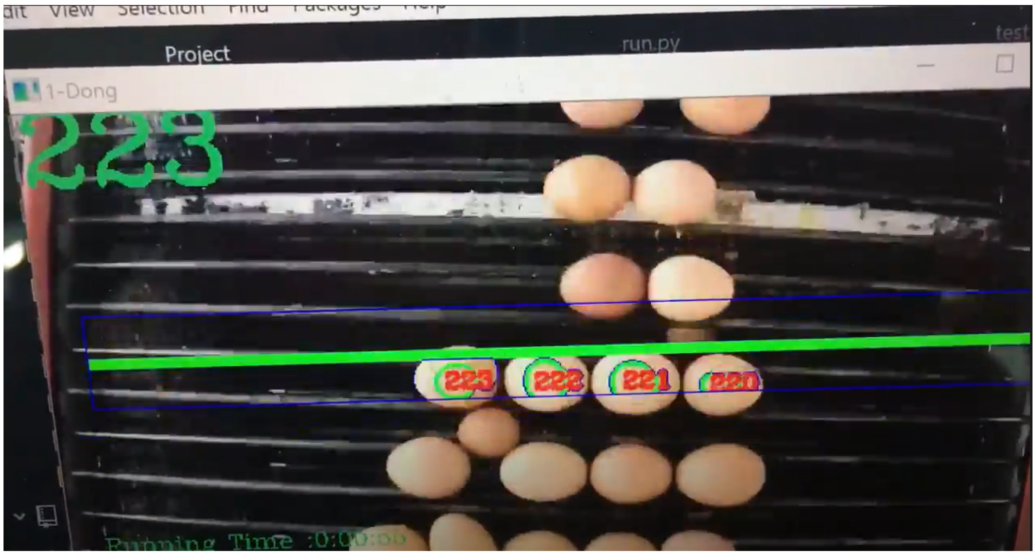


Figure 3: The video 'Egg counter with computer vision on the conveyor' [18].

The detected eggs in the video shown in Figure 3 were also marked when the eggs were near the green line on the counter. In this case the eggs were marked with a circle and an ordinal number without more precise boundary of the object. In addition, Figure 3 shows a small egg that was not counted by the program. Probably the program can detect only standard size eggs.

3 Application for egg detection on a production line

Our application consists of two main modules written in Python [19], [20] with the support of OpenCV framework [7]. The first module (called Detector) is responsible for egg detection using a selected computer vision method. The second module (called ObjectsTracker) is responsible for tracking and counting the eggs as they change their position on the consecutive video frames due to the movements of the production line. Since Detector module was developed using different methods (neural networks or template matching) and different models of neural networks (single step and two step) the implementation included: 1) preparation of data set for training neural networks, 2) preparation of templates for template matching detection method, 3) training the neural networks, 4) implementation of the modules itself. Each of these steps is described in the following subsections.

3.1 Data set for training and testing neural networks

Neural networks will be the more accurate, the larger and more diverse the data set is [1]. The diversity is especially significant for objects that may look different and represent the same object e.g. dogs may be in different breeds yet they are all dogs [21]. In this study we aim to detect chicken eggs i.e. not very diverse objects. Therefore at the beginning we tried to train neural networks on a small set of data - 200 unique photos which included about 600 eggs. However, it turned out to be not sufficient with a very first neural network we tried to train (the results of egg detection were unsatisfactory). Thats why an attempt was made to increase the size of the data set beyond the recommended minimum number of unique photos of objects usually stated to be 2000 [1]. As a result, 2500 unique photos were collected with about 9000 eggs. The collected photos differed in resolutions, color of the background and the degree of cropping of the eggs. The next step was to manually position all 9000 eggs on each of the 2500 photos. For this purpose, LabelImg program [22] was used which let to indicate a rectangular boundary of an object and thus to determine its coordinates. The information about the position of objects was exported in PascalVOC or YOLO (You Only Look Once) [22] format dependent on the type of neural network it was to be used with. It was a time consuming activity to mark all 9000 eggs and export this information to the files that finally took about 8 hours.

3.2 Templates for template matching method

Template matching is a method of finding information in images that uses a pattern of the searched object [23]. Therefore, for each of tested videos from the production line an example of an egg cropped from a frame of this video was used as a template. Several different shots were tried but they did not change the effectiveness of the object detection. Figure 4 shows selected video stills against which the tested methods were evaluated. The images of the eggs in vertical and horizontal orientations were chosen as the most common layout of the eggs on the production line. The process of selecting and preparing the appropriate templates took about 15 minutes.

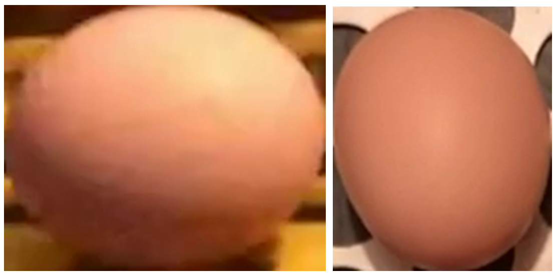


Figure 4: Templates selected for the object detection using the template matching method.

There are a number of algorithms that can be used in this method. In the research presented in this work one of the algorithms implemented in the OpenCV library was used. It is a normalized square difference matching method. In the case of template matching algorithms, the normalization means a greater tolerance for the difference in lighting between the input image and the template [24].

3.3 Training and testing neural networks models

There are a number of ready-to-use models of neural networks usually trained to recognize several dozen of the most common types of objects such as: people, dogs, cats, cars, planes, cell phones and other everyday objects [21], [25]. However, there is no neural network that is trained to detect or recognize hen eggs. Thats why training the neural network to detect hen eggs was a part of the implementation of our solution.

We have conducted the process of training neural networks in accordance with the state of the art. In order to assess the training process we adopted two measures: loss function [21] and mean Average Precision (mAP). Loss function is defined as the difference between the values calculated by the neural network and the actual values. Ideally, the value of the loss function should be zero. However, we noticed that at the beginning of the training process the value of the average loss function collapsed significantly but after some steps it started to fluctuate around the value of 1. The further proceeding with the training process didnt lower the value of the loss function. Thats why we decided to stop the training process while loss function achieves the value of 1. Various formulas are used to calculate the value of the loss function with the most common being the mean square error. The loss function only occurs with supervised learning [26]. Mean Average Precision is a value specifying the average effectiveness of assigning data to specific classes. It is calculated as the sum of average precision of each class divided by the number of classes in a given neural network according to the formula given in [26]. However, in our case we dealt with one class only. With the help of the above defined metrics we paid the attention to avoid underfitting problem [26], [27]. On the other hand, to eliminate the threat of overfitting we applied the hold-out cross-validation technique that is a statistical method relaying on dividing a data set into subsets: a training subset and a testing subset [27]. The dataset consisted of 2500 pictures divided into training and validation set in proportions 20/80%. The egg templates for template matching method were prepared based on the egg pictures from the same dataset.

For our research we have chosen one model from SSD, YOLO and R-CNN [8], [28], [29] neural networks families. These networks were selected because they are currently the most popular, they have a predisposition to be used in real-time and indicate the position of the object through a rectangular envelope which is sufficient information for the research on selected videos [21], [25]. Moreover, all of the selected models of neural networks can be generated in a format that can be imported by OpenCV library making the comparison of their features more reliable.

As an example of SSD family models we chose SSD-MobileNet v2. The architecture of this model [9] is depicted in Figure 5. The ssd_mobilenet_v2_coco model and the corresponding configuration file [25] were downloaded from the official TensorFlow database containing ready-made neural networks. After the appropriate editing of the configuration file (changing the number of classes to 1, changing the paths to the data set and including a random selection of photos from the test set), the training of this neural network was started. The batch values equal 24 were left as recommended by the authors because this is the value that the GPU (Graphic Processor Unit) used to train the network could handle. The maximum number of steps at which the training process was terminated was set to 200000 as being sufficient according to the recommendations of the model authors. The resolution of the processed images was set to 300x300 pixels because the network trained in this way would run faster than with a resolution of 512x512 pixels making the object counting program more likely to be able to run in real-time.

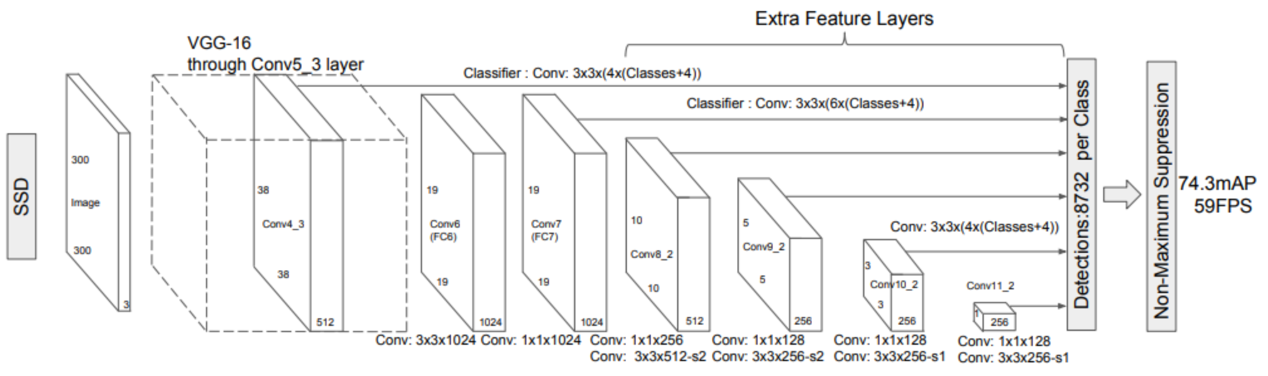


Figure 5: The architecture of SSD-MobileNet v2 model [9].

The choice of image resolution (300x300 in this case) was caused by the fact that for ssd-mobilenet v2 the authors announced the nominal performance was equal to 46 fps (Frames Per Second) while it drop to only 19 fps when the resolution of images was 512x512. As we plan our experiments having in mind the required performance at least of 25 fps, we decided to use films with the resolution of 300x300. This choice doesnt impose any special constraints on camera as the resolution of 300x300 is commonly available in such equipment.

The revision of the training process in TensorBoard [30] revealed that the images from the test set were detected with satisfactory effectiveness from a threshold of about 25000 steps (see Figure 6)

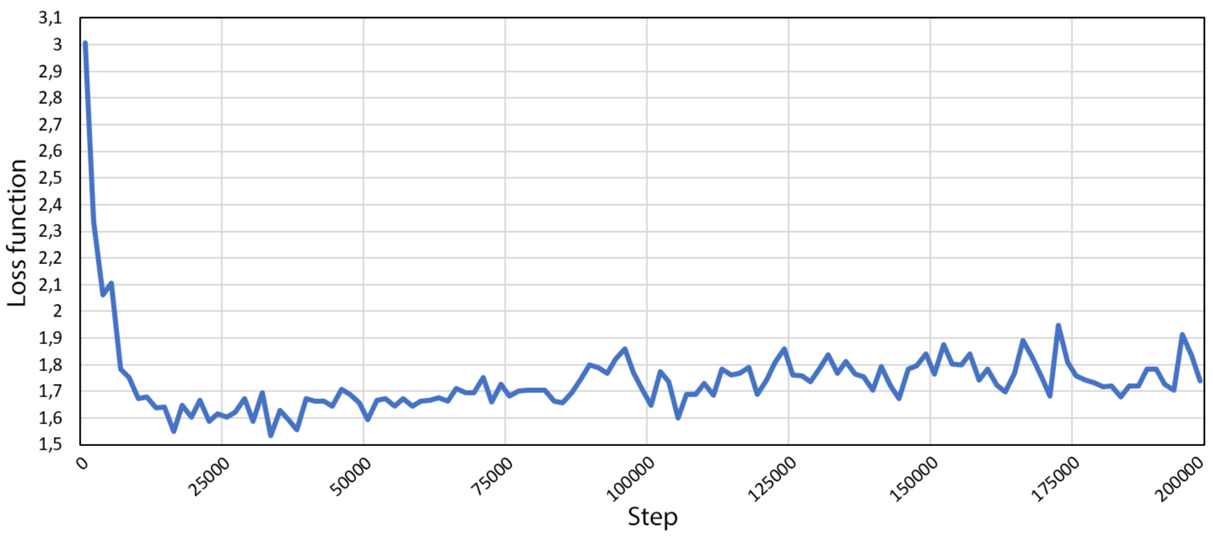


Figure 6: The value of the loss function while training SSD-MobileNet v2 neural network.

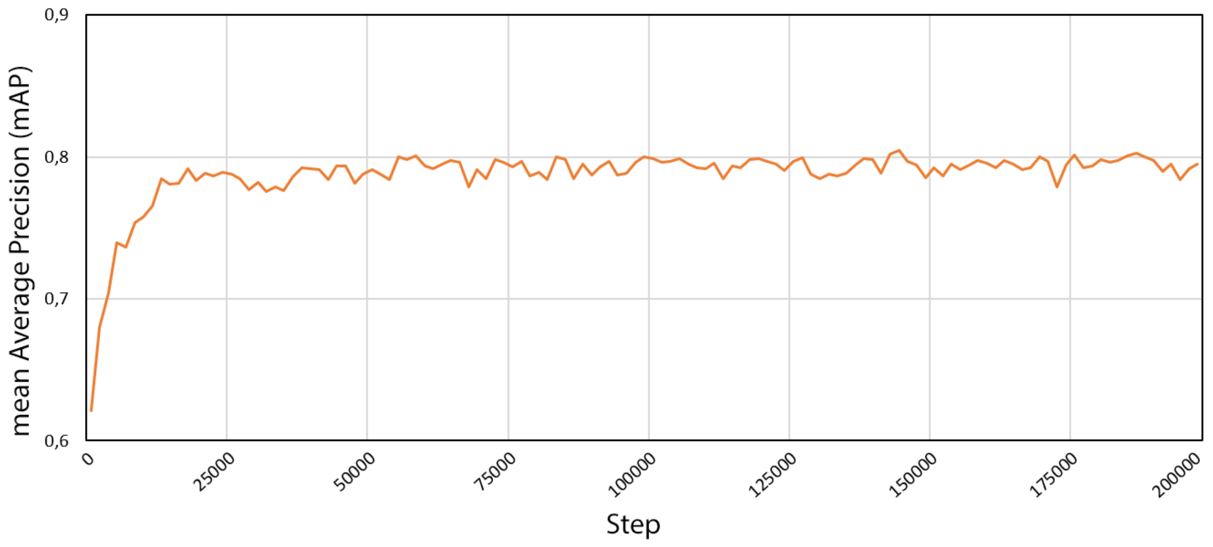


Figure 7: Mean Average Precision while training SSD-MobileNet v2 neural network.

However, neural networks recorded after 125000 and 175000 steps showed the highest effectiveness of object detection. It was confirmed during the tests carried out with the neural network trained after 125000 or 175000 steps where the degree of detection confidence was very high (95% - 100%) and the envelopes indicating the location of the objects were accurate. The neural networks trained with a number of steps lower than 125000 revealed lower detection confidence for data outside our test sets and as a result they were excluded from final considerations. Finally, the network trained after 125000 steps was chosen because the loss function had a similar error and for fewer iterations the risk of over-fitting was lower. The training was carried out using the central processor unit (CPU) and the graphics processor unit (GPU). The comparison revealed the training process using GPU was about 11 times faster than using CPU. This is due to the nature of neural network topologies that require a large amount of simultaneous computations [31]. Additionally, the choice of 125000 steps for training neural network also shorten the whole process that finally took about 13 hours instead of 21 in case of performing 200000 steps.

As an example of R-CNN neural network family models we chose Faster R-CNN (FR-CNN) due to its potential toward working in real-time [29], [32]. The architecture of this model is depicted in Figure 8. The network model named faster_rcnn_inception_v2_coco and the corresponding configuration file were downloaded from the official TensorFlow database containing ready-made neural networks [25]. The configuration file was modified in the same way as for SSD-MobileNet v2. The maximum number of steps was left as recommended by the authors of the model (200000 steps) bearing in mind that in the case of unsatisfactory performance of the trained network the training process could be continued. The graphs of the loss function and the mean Average Precision created during training had the shapes similar to those created during training of SSD-MobileNet v2 network. The loss function decreased sharply at the beginning of training to keep the value of around 0.7 starting from 25000 steps. Meanwhile, the mean Average Precision increased to a value of about 0.84 and maintained this value until the end of the training process. Finally, for further use, we selected the network trained with 200000 steps that took 6.5 hours.

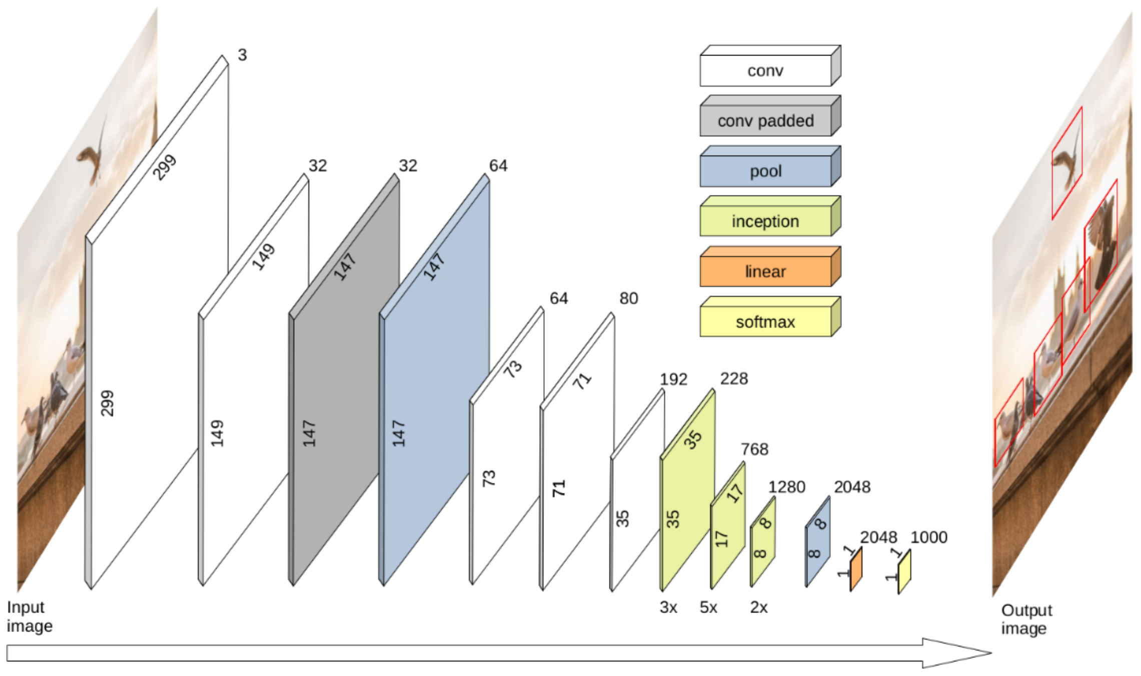


Figure 8: The architecture of Faster R-CNN (FR-CNN) model created based on the description from [33].

As an example of YOLO neural network we chose YOLOv3 model [28]. The architecture of this model [34] is depicted in Figure 9. The training process was carried out using Darknet framework [5]. As in the case of neural networks previously trained with TensorFlow system, training this network to detect hen eggs consisted in re-training the ready-to-use YOLOv3 neural network structure [35]. The training configuration file was mostly configured according to the recommendations outlined in [35]. However, due to technical reasons (insufficient GPU memory), it was necessary to reduce the batch value from the recommended 64 to 24. When training one class, it is suggested to use 4000 iterations [5]. However, due to the need to reduce the value of the batch, it was decided to increase the number of iterations proportionally that is to 11000. The size of the input image was set to 418x418 pixels i.e. one of the three standard values suggested in [5]. Darknet framework does not have a tool for visualizing the neural network training process in real-time, therefore the correctness of egg detection was manually checked on the images from the test set and on images from outside our data set after the process was finished. The comparison was done for neural networks recorded after 10000 and 11000 steps. It revealed that both networks could detect the eggs with almost the same satisfactory confidence of 95% 100% and a precise indication of the position of the objects. Their loss function and mean Average Precision values were also highly similar to each other and were about 0.2 and 0.96, respectively. For this reason we decided to use the neural network recorded after 11000 steps that lasted 9 hours without the need for further training the network.

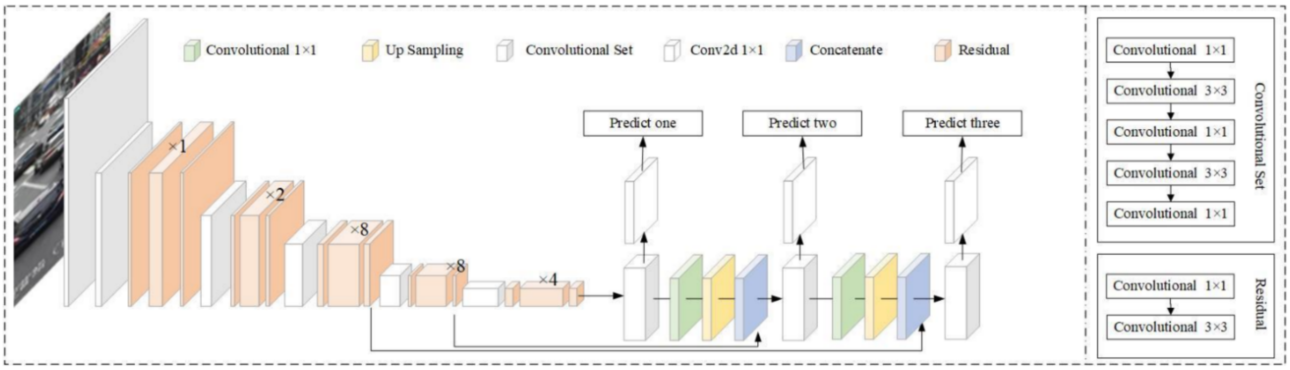


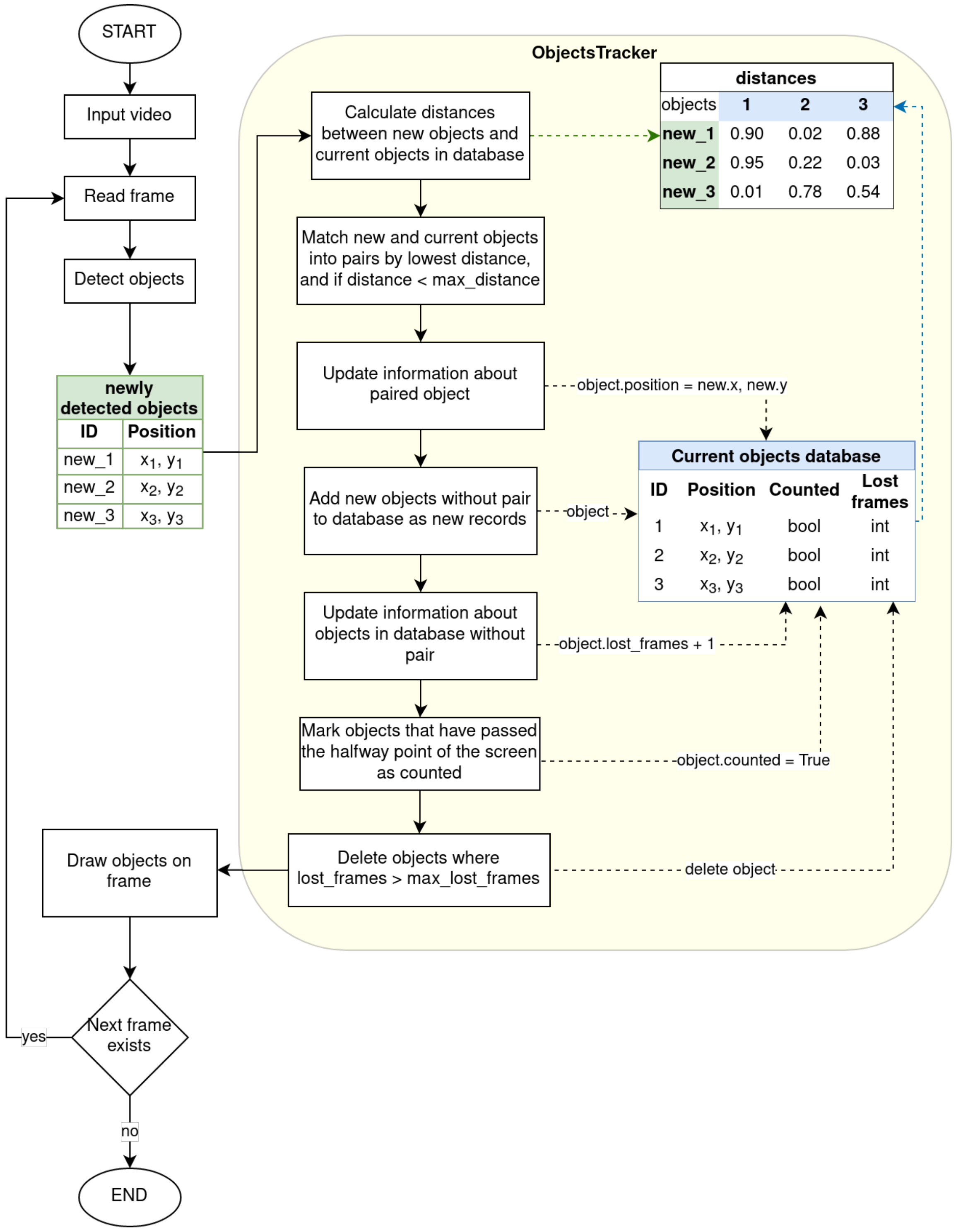
Figure 9: The architecture of YOLOv3 model [34].

3.4 Implementation of tracking and detection modules

Since we aimed at detecting and tracking the eggs while they were moving with the production line as well as counting them, our software had to implement all these functions. Thats why the developed software consisted of two scripts written in Python 3: i) ObjectsTracker and ii) Detector. ObjectsTracker is a Python 3 script that allows one to track and count objects that change their position from the top to the bottom on a film from a production line. It takes a list containing the coordinates of the centers of detected objects as an input. Then, it compares the newly entered data to the previously entered data and based on the distance between the points determines whether this object has already been observed and counted or whether it is a new object. The object is counted when it is in the lower half of the image (crosses the horizontal green line). The program gives each object a random color from a defined color base and when the object is counted i.e. it is placed at the bottom of the image, its color is changed to green. The program returns the information about objects currently in the video frame, whether they have already been counted or not and objects that have been lost before reaching the bottom of the screen. This script was used by Detector script to track and count objects that were detected using the previously described computer vision methods. The actions accomplished by ObjectTracker and shortly described above are illustrated on the Figure 10 in a convention of a flowchart. Detector is a Python 3 script that enables the detection of objects using selected computer vision methods as well as tracking and counting them using the imported ObjectsTracker module. As an input it takes files relevant for a given detection method (neural network configuration or a template) and a video containing the eggs to be detected and counted. It has three versions supporting different computer vision methods:

- tensorflow_detector.py (for detection using a neural network saved in the TensorFlow format),
- darknet_detector.py (for detection using a neural network saved in the Darknet format,
- matchtemplate_detector.py (for detection using a selected template matching method).

It can be run using both CPU or GPU. The result of the program is a movie with detected objects marked with appropriate colors from ObjectsTracker module as well as the information about the number of counted and lost eggs, the elapsed time and the average number of frames per second. In this work Detector program was used to determine the effectiveness and performance of the tested computer vision methods.



START
Input video
Read frame
Detect objects
newly detected objects
ID
Position
new_1
x1, y1
new_2
x2, y2
new_3
x3, y3
ObjectsTracker
Calculate distances between new objects and current objects in database
distances
objects
1
2
3
new_1
0.90
0.02
0.88
new_2
0.95
0.22
0.03
new_3
0.01
0.78
0.54
Match new and current objects into pairs by lowest distance, and if distance < max_distance
Update information about paired object
object.position = new.x, new.y
Current objects database
ID
Position
Counted
Lost frames
1
x1, y1
bool
int
2
x2, y2
bool
int
3
x3, y3
bool
int
Add new objects without pair to database as new records
object
Update information about objects in database without pair
object.lost_frames + 1
Mark objects that have passed the halfway point of the screen as counted
object.counted = True
Delete objects where lost_frames > max_lost_frames
delete object
Draw objects on frame
Next frame exists
yes
no
END

Figure 10: Flowchart illustrating the principle of operations of ObjectsTracker.

4 Results of the experiments

To answer the questions cited in section 2 as the research targets of our work we have designed a set of experiments. The tests were carried out on the same set of video files from the production line using the proprietary Detector program described in section 3.4 and in the same hardware environment:

- CPU: Intel Core i7-9750H (6 cores, 12 threads, 2.60-4.50 GHz, 12 MB cache),
- GPU: NVIDIA GeForce GTX 1660 Ti, 6GB GDDR6
- RAM: 32 GB (SO-DIMM DDR4, 2666MHz)
- Operating System: Ubuntu 18.04

Since we aimed at evaluating the process of visual egg detection, tracking and counting we defined a number of events and related metrics that helped us to assess its effectiveness. For events the natural choice was as follows: a correctly detected egg, an unrecognized object, a false positive egg detection. However, since after a successful egg detection we track the position of this egg, the following events related to the effectiveness measure were introduced: a loss of an egg being tracked, a multiple detection of the same egg. The relevant effectiveness metrics were defined as the number of these events occurred in the tested video recordings. In order to correctly understand these metrics we introduce the concrete definitions below:

- ”Correctly detected egg” means an egg that was detected and tracked at least until it passed the half of the screen (crossed the green horizontal line) and was counted only once,
- ”Multi-detected egg” means a previously detected egg not detected on a number of consecutive videos frames and then detected again. Actually, this might happen when an egg is not detected on the next video frame while it was detected on the previous one or when an egg is detected but not correctly associated with itself on the previous video frame. The last case is due to the misplaced detection and shifted placement of a bounding-box thus misleading ObjectsTracker module. Such an egg is counted again as a new egg,
- ”Lost egg” means a previously detected egg that is not detected on consecutive video frames till it passes the green horizontal line,
- ”False positive” means a detection of an egg in place where it is not actually present,
- ”Unrecognized object” means an object that has not been detected nor counted or is only detected after it passed the half of the screen (green line).

Thus, the effectiveness we report for an evaluated film is defined as a ratio of the correctly detected, tracked and finally counted eggs to all eggs present in this film. According to this definition, any of the events mentioned above, except correctly detected egg, contributes to a reduction of the effectiveness. If we define the detection effectiveness for a single video frame as :

$$detection\_effectiveness = \frac{TP}{TP + FN}, \tag{1}$$

then our effectiveness metric might be defined as the minimum of detection effectiveness over the set of video frames of a given recording:

$$\min_{\forall frames \in recording} \frac{TP}{TP + FN}, \tag{2}$$

The performance metric is defined as the number of video frames the program is able to generate per 1 second while it is reading the input frame, detecting the objects, tracking them, counting and finally creating the output video frame where bounding boxes, egg numbering and the green line are added. To evaluate the achievable performance of the selected methods (CNN and template matching) i.e. the maximum achievable frame generation rate, we read the video frames from a file as a bunch. In this way the output frame generation is a bottleneck and its rate is a value of the performance metric.

4.1 Performance evaluation of the egg detection process supported with neural networks techniques.

In this section we present the results that assess the effectiveness and performance of egg detection, tracking and counting process accomplished with the support of neural network models. Figures 11 and 12 show frames from the film generated by Detector program during the egg detection and counting process. All eggs in both videos were detected and counted correctly when the detections were carried out with CPU or GPU. The numbers and dots marking the centers of the detected eggs were placed in the center of them when they were fully visible. On the generated movies these numbers and dots were moving smoothly across the screen aligned with the eggs they were marking. This means that rectangular boundaries precisely positioned eggs and the eggs themselves were detected continuously so ObjectsTracker module did not mark any egg as ”lost”.

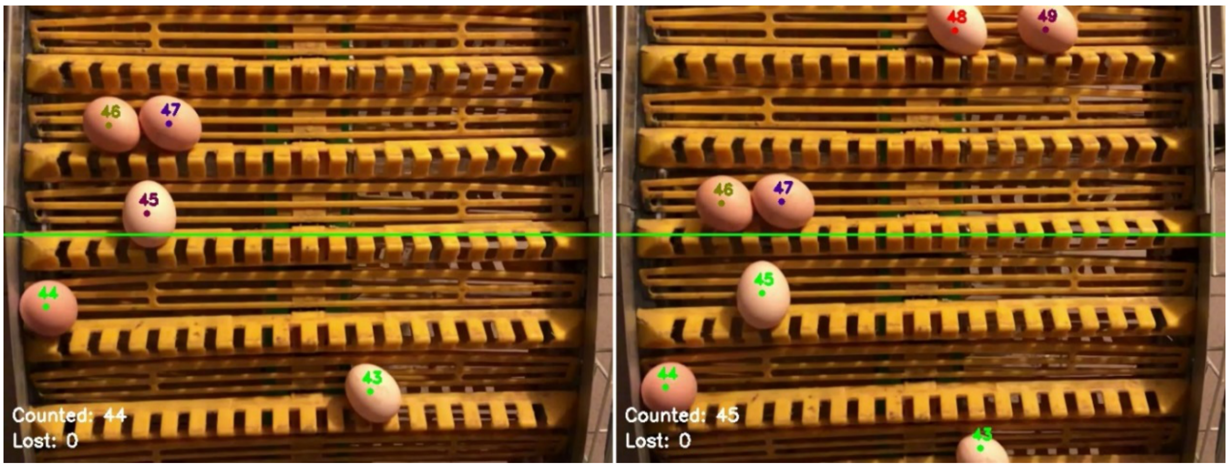


Figure 11: Correct detection of eggs with SSD-MobileNet v2 neural network. Stills from film No.1.

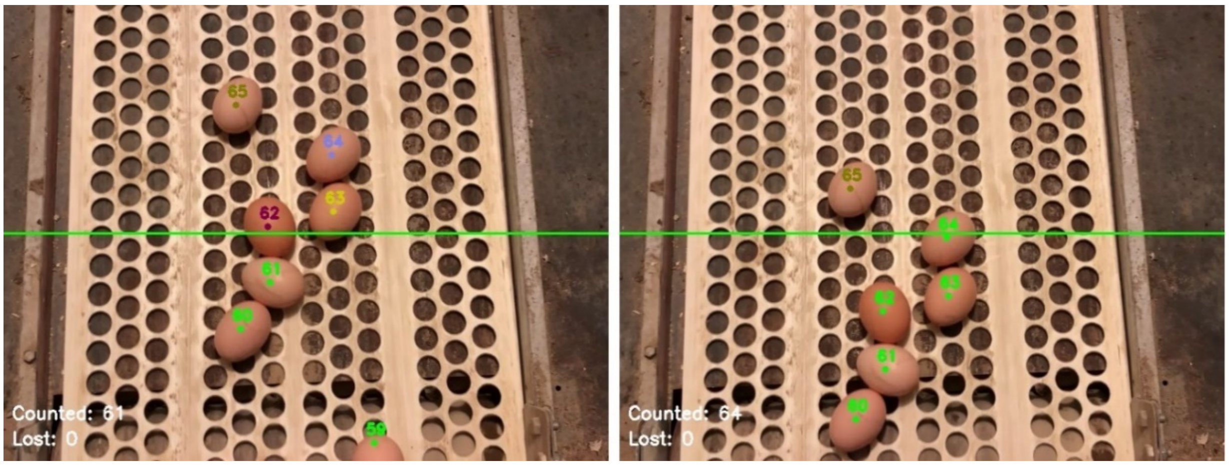


Figure 12: Correct detection of eggs with SSD-MobileNet v2 neural network. Stills from film No.2.

In the Figure 11 one can notice the correct detection of an egg numbered 45. After it crossed the green horizontal line the egg counter increased its value from 44 to 45. The number and the dot marked on this egg in violet color (on the left picture) changed its color to green (on the right picture) thus confirming the correct counting. The similar situation might be observed in the Figure 12. There, one can see the eggs number 62, 63 and 64 that are correctly counted thus they changed the color of their numbers and dots after they crossed the green horizontal line (the picture on the right). Also the counter has been updated accordingly.

Next, we studied the effect of skipping ”n” frames and processing only every (n+1)-th frame on the effectiveness and performance of egg detection. Detector program allows to define the number of video frames to be skipped for detection e.g. setting this parameter to 3 will make the eggs to be detected in video frames number 1, 5, 9, 13, etc. i.e. every fourth video frame will be processed. The tests were carried out separately for the program running with CPU or GPU. The number of skipped video frames ranged from 0 to 6. The effectiveness and performance of egg detection, tracking and counting process for 5 different films (an average) are shown in Figure 13 and Figure 14.

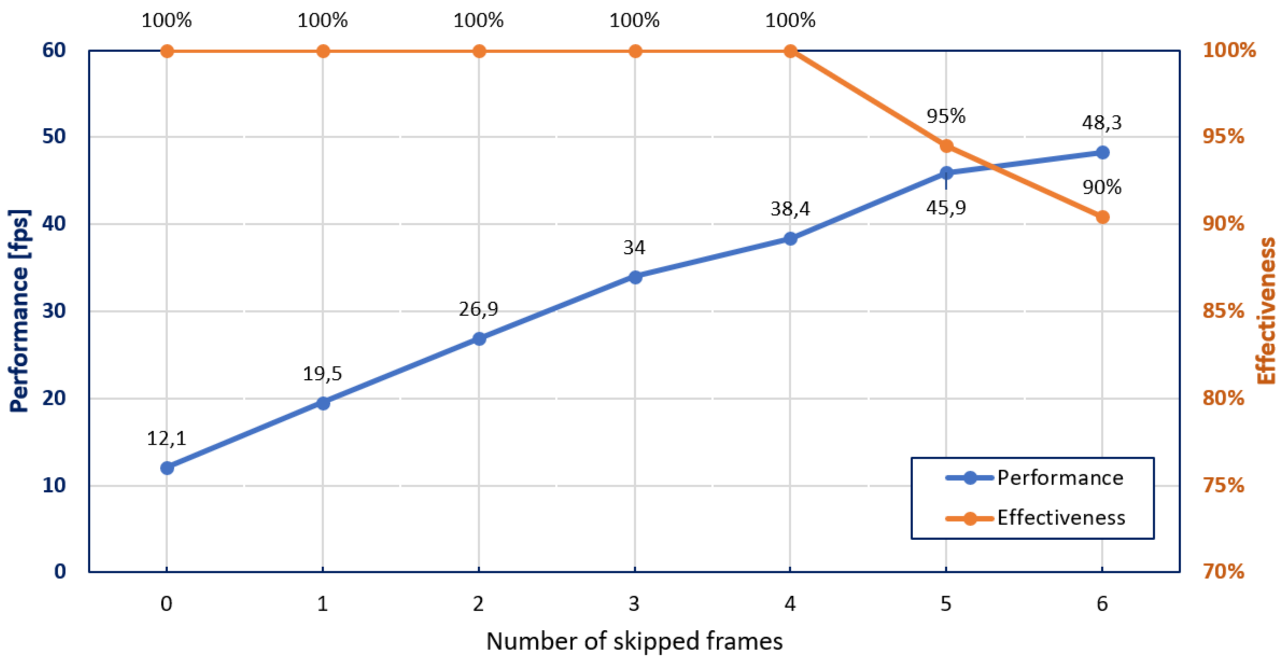


Figure 13: The impact of every (n+1)th video frame processing on effectiveness and performance of egg detection using SSD-MobileNet v2 neural network and CPU.

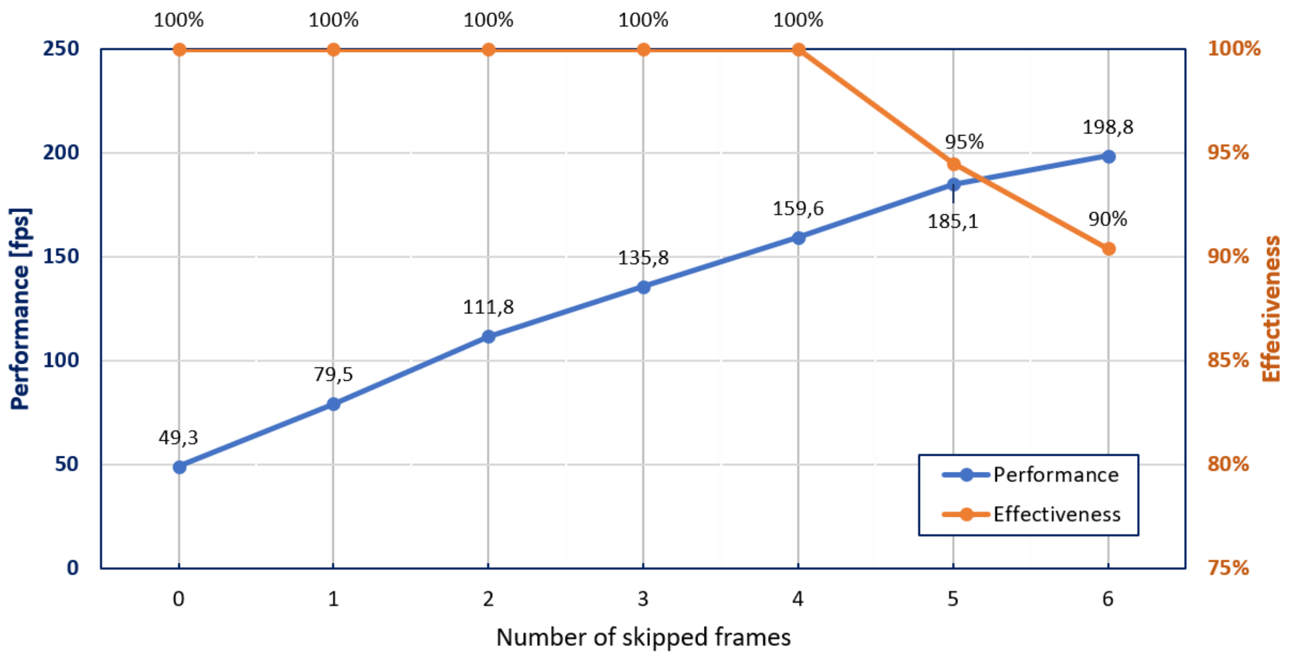


Figure 14: The impact of every (n+1)-th video frame processing on effectiveness and performance of egg detection using SSD-MobileNet v2 neural network and GPU.

For both processor types the effectiveness was maintained at 100% until the number of skipped video frames ”n” reached 4, then dropped by 5% for ”n” equal 5 and 6 consecutively. On the other hand, each increase in the number of skipped frames raised the performance by several frames per second for CPU and several dozen frames per second for GPU. For the processing of the video recordings using the CPU this change is of great importance because skipping 24 frames made it possible to achieve performance greater than 25 frames per second without any loss of the effectiveness i.e. the video could be processed in real-time. Using the processing power of GPU and processing every fifth frame we managed to achieve the performance of 159.6 frames per second while maintaining the effectiveness of 100%.

Finally, we verified the impact of the number of eggs in a video frame i.e. the objects to be detected on the program performance metric. Figure 15 shows a graph containing information on the number of video frames and the average number of detected objects per frame during successive seconds of processing the video recording number 1

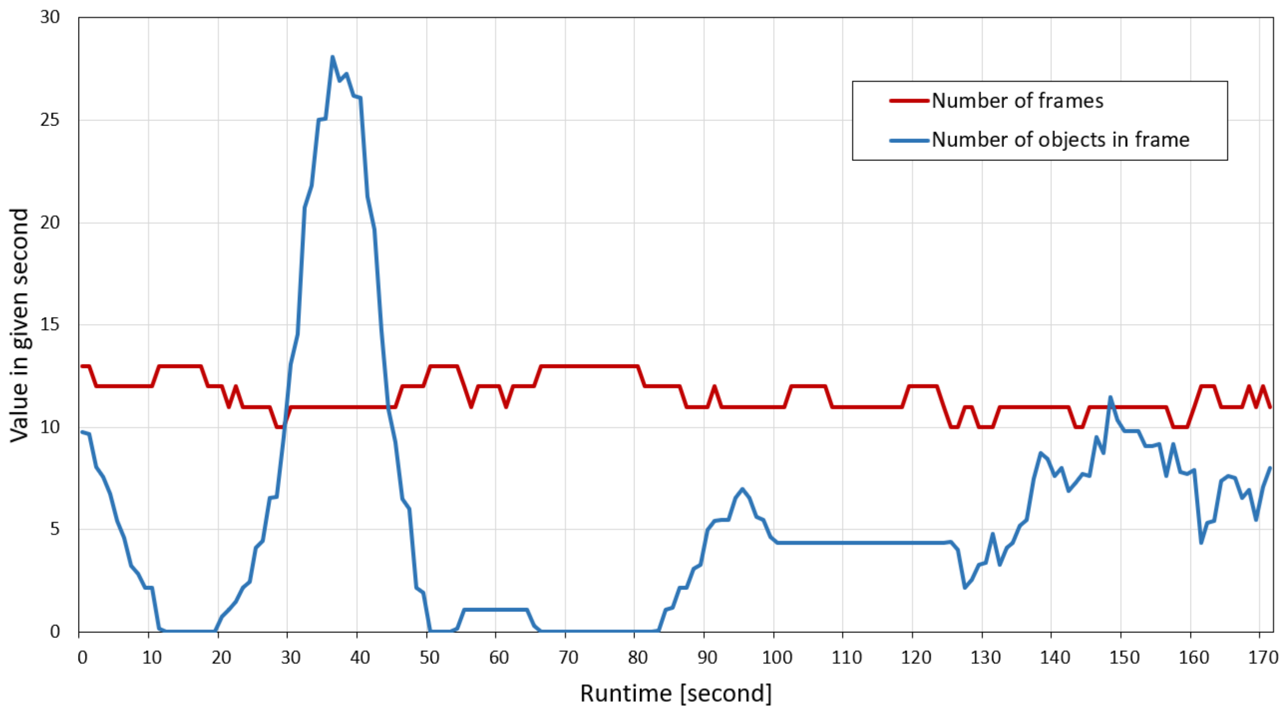


Figure 15: Program performance vs the number of eggs to be detected per video frame for SSD-MobileNet v2 neural network using CPU (film No.1).

In the graph from Figure 15 one can observe the frame rate is almost constant and varies between the values 10 and 13 while the number of eggs varies between 0 - 28. This means a very slight correlation between the number of processed frames per second and the number of eggs - when there is no egg in the frame the number of frames per second increases to 13 while it drops to 10-12 frames per second when there are more than 5 eggs. However, this is not a difference that significantly affects the performance of this method.

In Figures 16 and 17 we present the results of the effectiveness and performance of egg detection process for FR-CNN neural network when video frame skipping is applied, for CPU and GPU respectively.

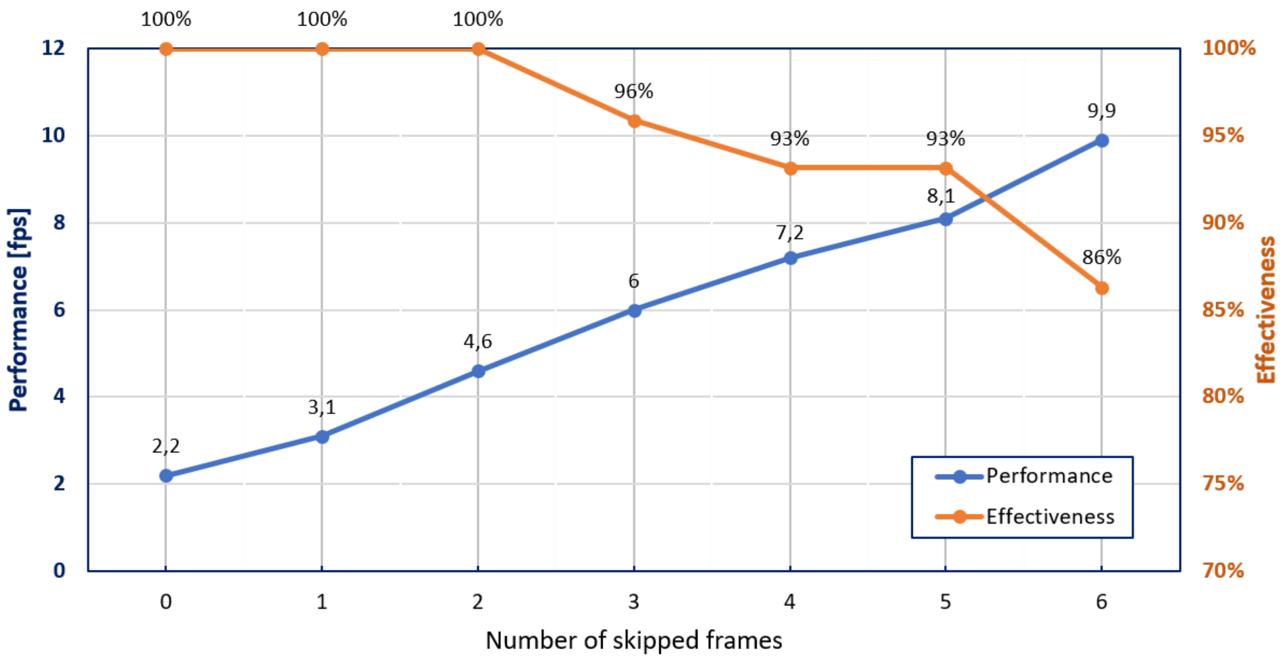


Figure 16: The impact of every (n+1)-th video frame processing on effectiveness and performance of egg detection using FR-CNN neural network and CPU.

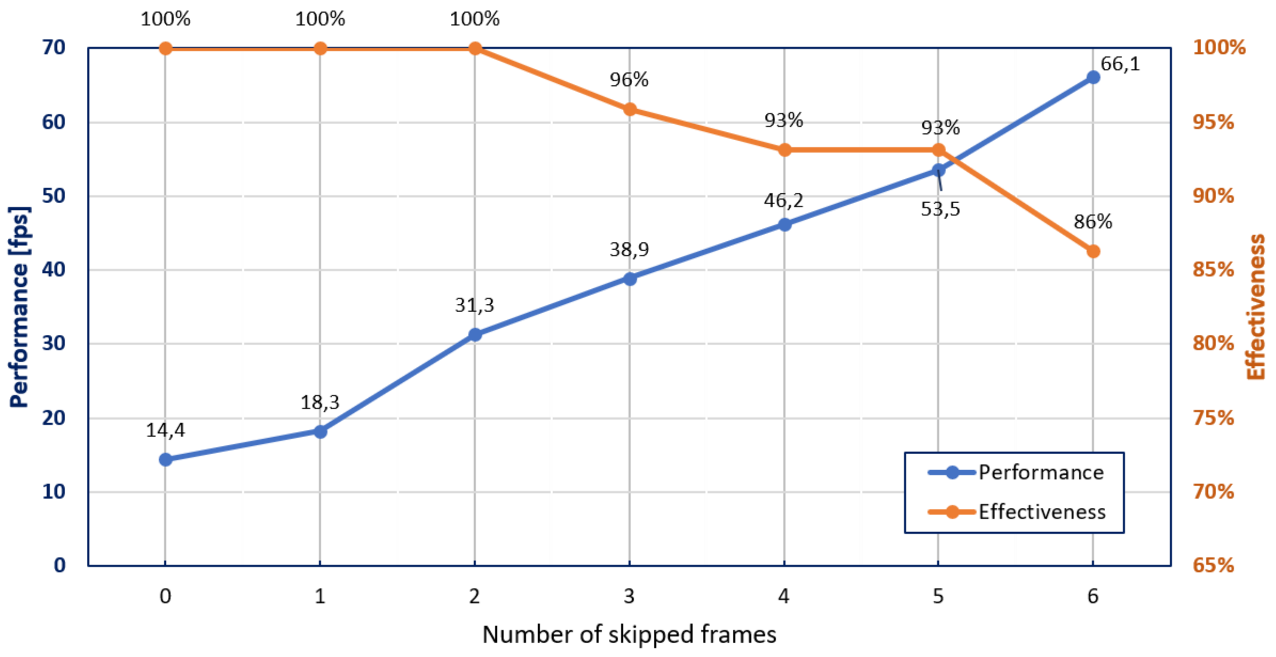


Figure 17: The impact of every (n+1)-th video frame processing on effectiveness and performance of egg detection using FR-CNN neural network and GPU.

The egg detection in every (n+1)-th video frame had an impact on both the effectiveness and performance of the program. The detection effectiveness had the same values for CPU and GPU processing. The 100% effectiveness level was maintained for skipping one or two frames which is equal to the effectiveness while processing each frame i.e. without skipping. When CPU processing power was used performance increased on average 1.3 fps for each skipped frame. However, even when skipping 6 frames the performance of 25 frames per second was not achieved whereas the detection effectiveness dropped to 86%. In the case of using GPU and setting the detection on every third frame it was possible to achieve a performance of 31.3 fps which allowed for real-time processing of the video recordings.

In case of investigating the impact of the number of eggs in a video frame on the program performance metric we obtained similar results as for SSD-MobileNet v2 neural network. In the case of FR-CNN neural network no correlation was noticed between the number of objects in a given frame of the video recording and the number of processed frames per second.

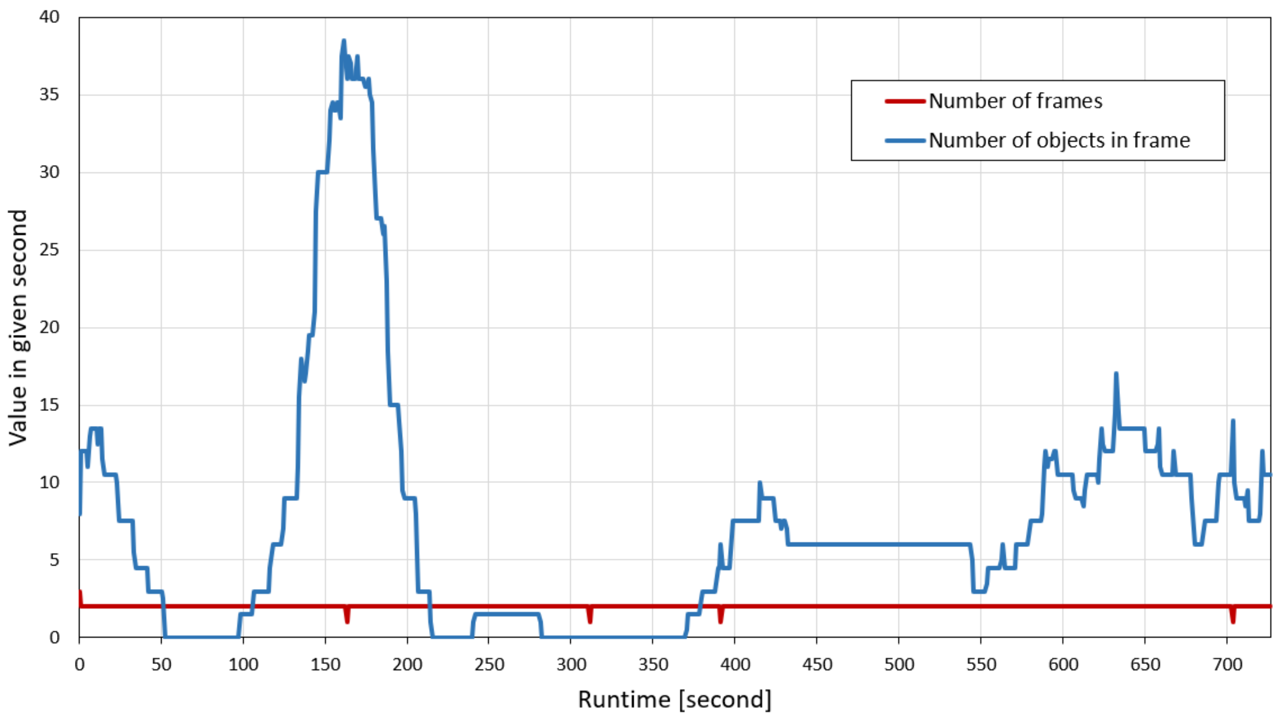


Figure 18: Program performance vs the number of eggs to be detected per video frame for FR-CNN neural network using CPU (film No.1).

In Figure 19 and Figure 20 we present the graphs illustrating the program performance and effectiveness vs the number of skipped frames for YOLOv3 neural network, for CPU and GPU respectively.

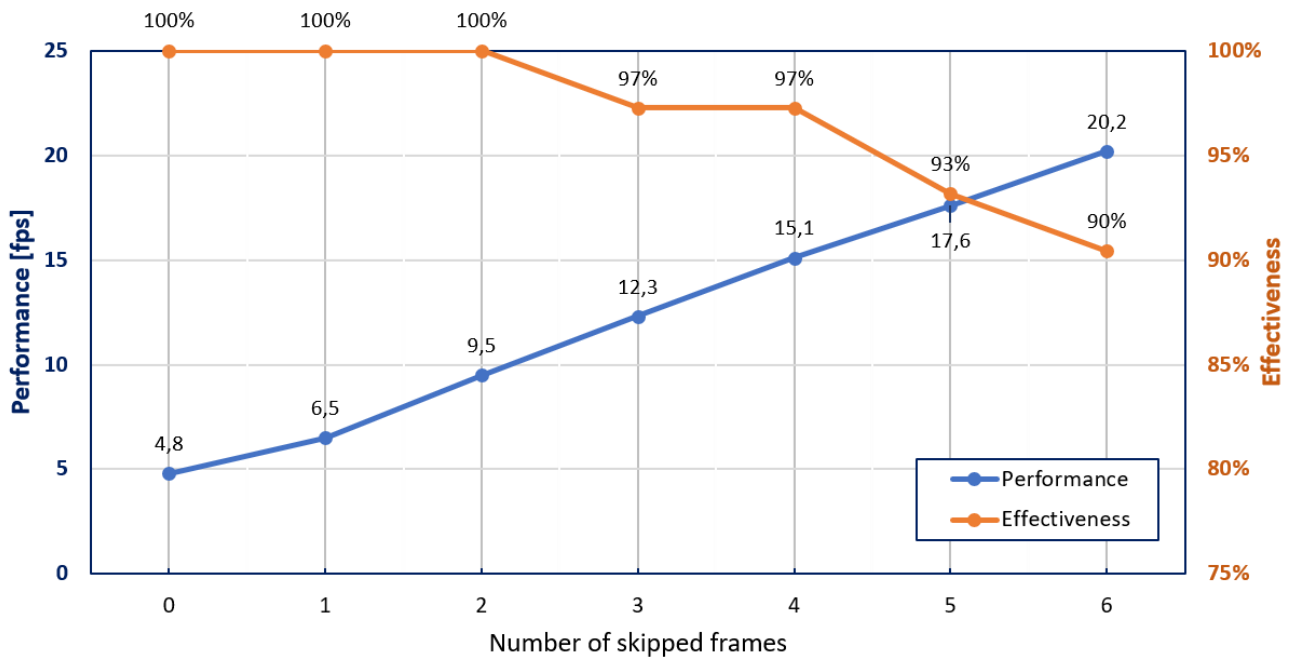


Figure 19: The impact of every (n+1)th video frame processing on effectiveness and performance of egg detection using YOLOv3 neural network and CPU.

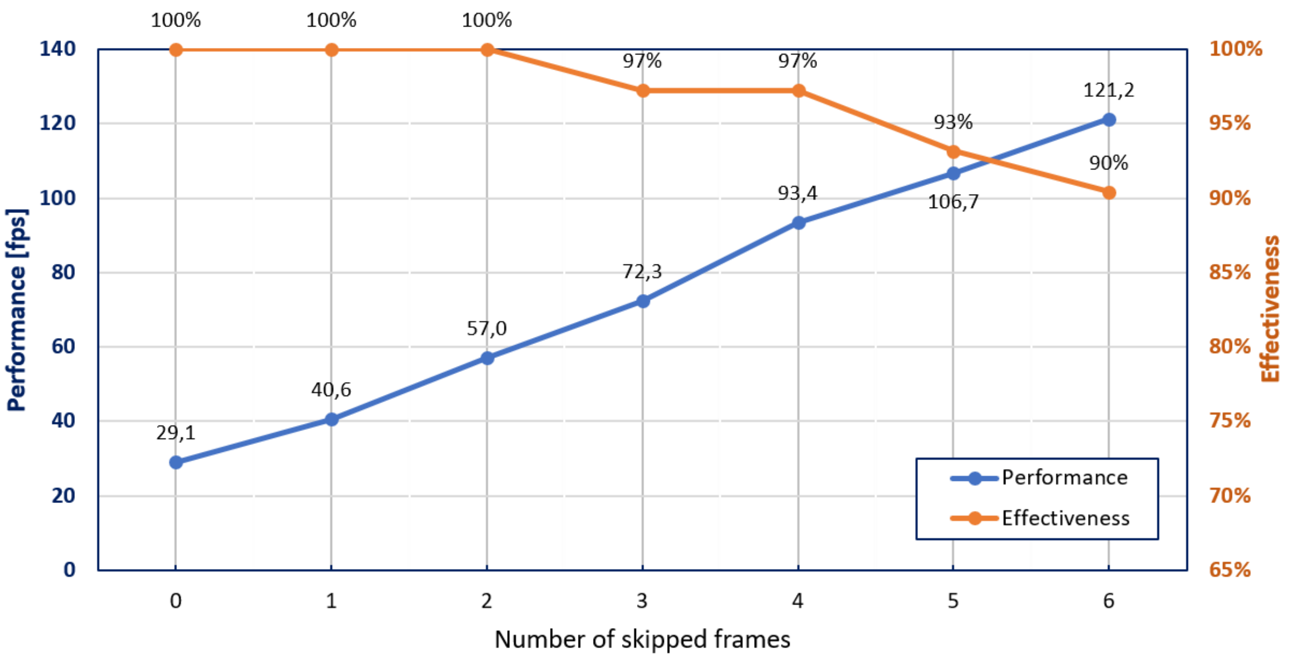


Figure 20: The impact of every (n+1)-th video frame processing on effectiveness and performance of egg detection using YOLOv3 neural network and GPU.

In the case of YOLOv3 neural network the processing of every (n+1)-th video frame had a similar effect on effectiveness and performance as in the case of two previously presented models. The limit to maintain 100% effectiveness was achieved for processing every third video frame (Figure 19). However, with n=2 and CPU the performance increased only to 9.5 fps which was not sufficient for real-time processing of movies. On the other hand, for GPU processing only every third video frame the performance increased to 57 fps. When the number of skipped frames was greater than 2, the effectiveness of detection and tracking of eggs was decreasing by 3-4% for each additionally skipped frame (Figure 20).

Figure 21 shows the result of examining the impact of the number of skipped frames on the program performance expressed in frames per second. The red line representing the program performance is almost flat (at the level of 4 fps) exhibiting some accidental increases to 5 fps during the intervals when the consecutive video frames contain no eggs.

Summarizing the results, we can conclude that skipping video frames was beneficial to the value of the performance metric for each type of investigated CNN network. This is due to the fact that the object detection in CNN networks requires a lot of effort related to processing the input image by many layers. Skipping "n" frames is almost as dividing the video input rate by "(n+1)" since only 1/(n+1)-th of frames are processed for detection, tracking and counting. However, this is almost but not exactly the same as lowering the video input rate by (n+1)-th since for skipped video frames we still perform some actions like reading the input video frame, drawing egg numbering, drawing the green line and the values of counters. Thats why the raise in the performance is not doubled with doubling the value of "n". On the other hand, one can observe that independently of the type of CNN network, from a particular value of "n", the value of the effectiveness metric is going down. This is the effect of the problems with mutual matching of the same egg in the consecutive processed video frames. The greater the value of "n" the bigger the distance between the position of the same egg on the consecutive processed video frames that finally leads to the problems with tracking. Thus some optimal value of "n" can be chosen.

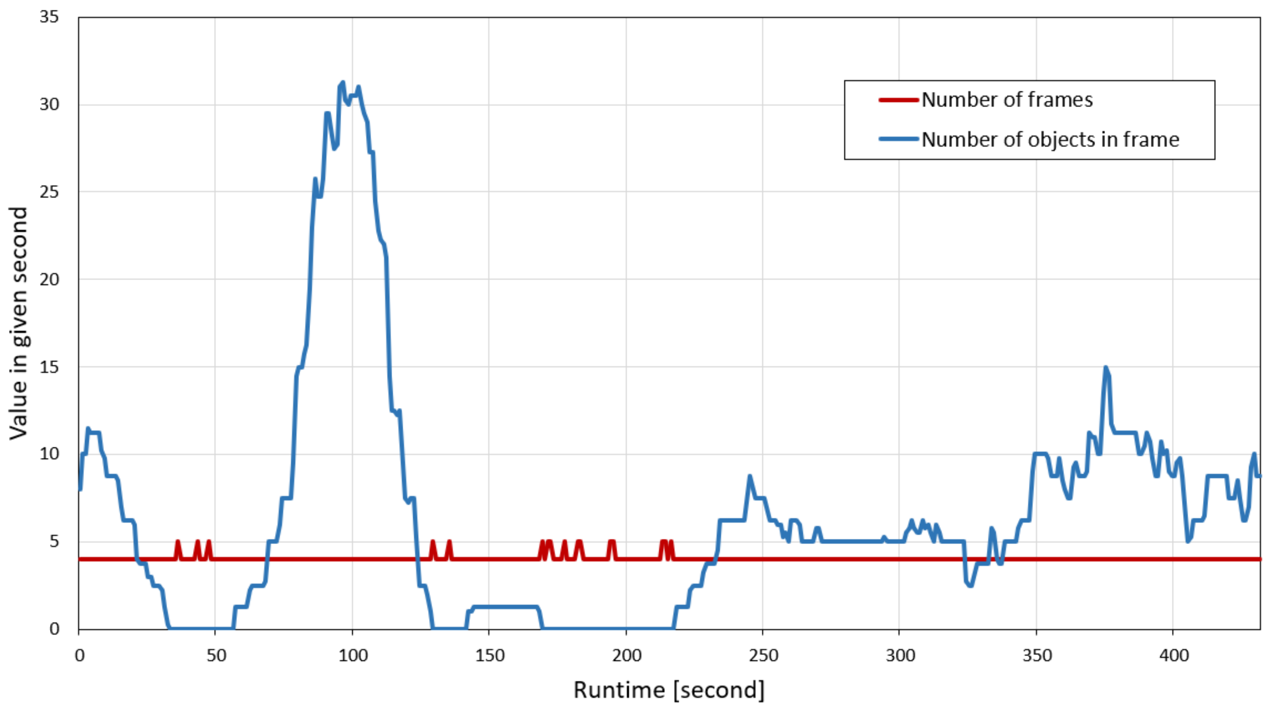


Figure 21: Program performance vs the number of eggs to be detected per video frame for YOLOv3 neural network using CPU (film No.1).

4.2 Performance evaluation of the egg detection process supported with template matching technique.

In contrast to the egg detection using the examined neural networks, in the case of the template matching method the effectiveness of egg detection and tracking was much more sensitive to the individual values of Detector configuration parameters. As noted in section 3.2 we have used a normalized square difference matching method that is implemented in OpenCV library. The library function returns a value that describes the level of similarity of an object and the template being compared. Because of the square difference component of this method, the lower value returned the better similarity of objects [24]. Since this value was characterizing the level of similarity of the object and the template we referred to it as a confidence of egg detection. After many attempts in order to obtain the best detection effectiveness, the following threshold values were selected: for the film number 1 - the confidence of egg detection equal 0.14, for the film number 2 and 4 this value was 0.11, for the film number 3 - 0.12 and for the film number 5 - 0.13. These values denote the thresholds below which an object is detected as an egg. However, with this method the best settings do not always mean 100% success of egg detections.

Table 1 shows the results of the examination of the effectiveness of the egg detection, tracking and counting process. The overall effectiveness is not the same for all recordings with the average of 82% meaning that on average one in five objects has been detected incorrectly. In the case of films number 1, 4 and 5 the effectiveness was 82%, 81% and 85%, respectively. These values are close to the average value of effectiveness for all tested films. In the case of the film number 2, the effectiveness is the lowest and equal to 63% although in this film the eggs are lit evenly which makes them visually more similar than for example in the case of the film number 1. More interestingly, the effectiveness for the film number 4 showing the same production line as in the film number 2 was larger and equal to 81%.

Film no	Real number of eggs	Eggs detected correctly	Lost eggs	Multiple detected eggs	False positives	Not detected eggs	Detection Effectiveness
1	73	60	1	7	0	5	82%
2	93	59	4	3	2	25	63%
3	24	24	0	0	0	0	100%
4	32	26	2	2	0	2	81%
5	72	61	0	0	0	11	85%

Table 1: Effectiveness of detection, tracking and counting eggs using template matching method.

For the film number 3 as the only one the effectiveness was 100%. A characteristic feature of this film is that all eggs are of the same white color, the lighting and scaling are the same throughout the film. These conditions made it possible to achieve 100% detection effectiveness. An example of egg detection in the film number 3 is shown in Figure 22.

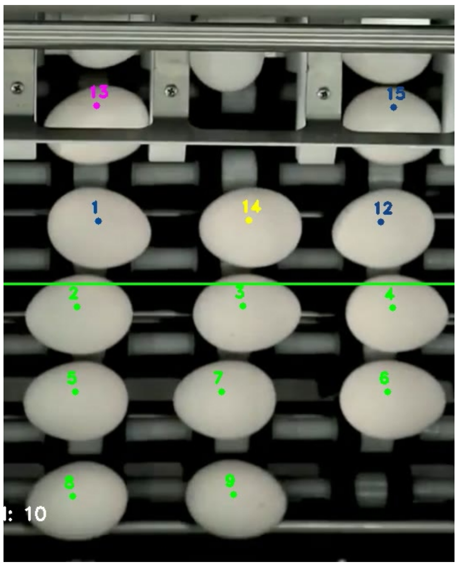


Figure 22: Egg detection and counting process using template matching method. A still from the film No.3.

Template matching method uses a template to find eggs in an image. All videos were tested with several different templates however these changes did not affect the total effectiveness of detection and tracking. In the case of the film number 2, it was surprising that some eggs were only detected when they reached the bottom of the video frame. An attempt was made to change the template to a cropped image of the egg while it was in the upper part of the video frame but this did not change the moment of egg detection. Attempts to change the template have shown that the correct detection of individual eggs differs depending on the template used. This means that one template may allow the detection of completely different objects than another from the same movie. For this reason an additional study was carried out on the effectiveness of detection and program performance vs the number of templates used in a cascade filtering. The study was carried out for film number 1 and for the number of templates from 1 to 4. The results are visualized in Figure 23.

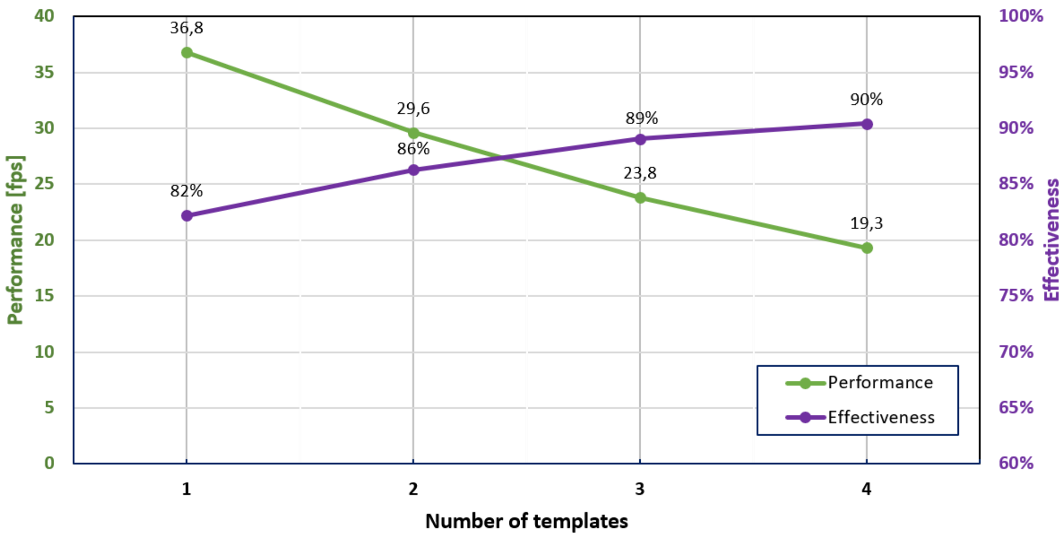


Figure 23: The impact of the number of templates used on the effectiveness and performance of egg detection using template matching method.

With the increase in the number of used templates, the detection effectiveness was increasing and the performance was decreasing. When using 3 or 4 templates, the performance dropped below 25 fps so the video recordings would not be processed in real-time. Comparing the results with the use of one and four templates, the detection effectiveness increased by 8% and the performance decreased by 46%. This means a large loss in detection effectiveness with a slight improvement in performance. In the case of the template matching method, it is not possible to use the GPU to detect objects. However, the performance on the CPU using one template is high and almost equal 37 fps for the tested films which is enough to process them in real-time.

Next we studied the effect of skipping n video frames and processing only every (n+1)-th frame on the effectiveness and performance of egg detection, tracking and counting with template matching method. The results depicted in Figure 24 reveal that processing every (n+1)-th video frame increases the performance without losing the effectiveness for the initial values of n. However, for the subsequent values of n the performance increases further but at the cost of a decreased effectiveness. In the case of skipping each subsequent 2 or 3 frames, the detection effectiveness remained at the level of 90% and the performance increased to 27.7 fps and 34.2 fps, respectively i.e. to values sufficient for real-time processing.

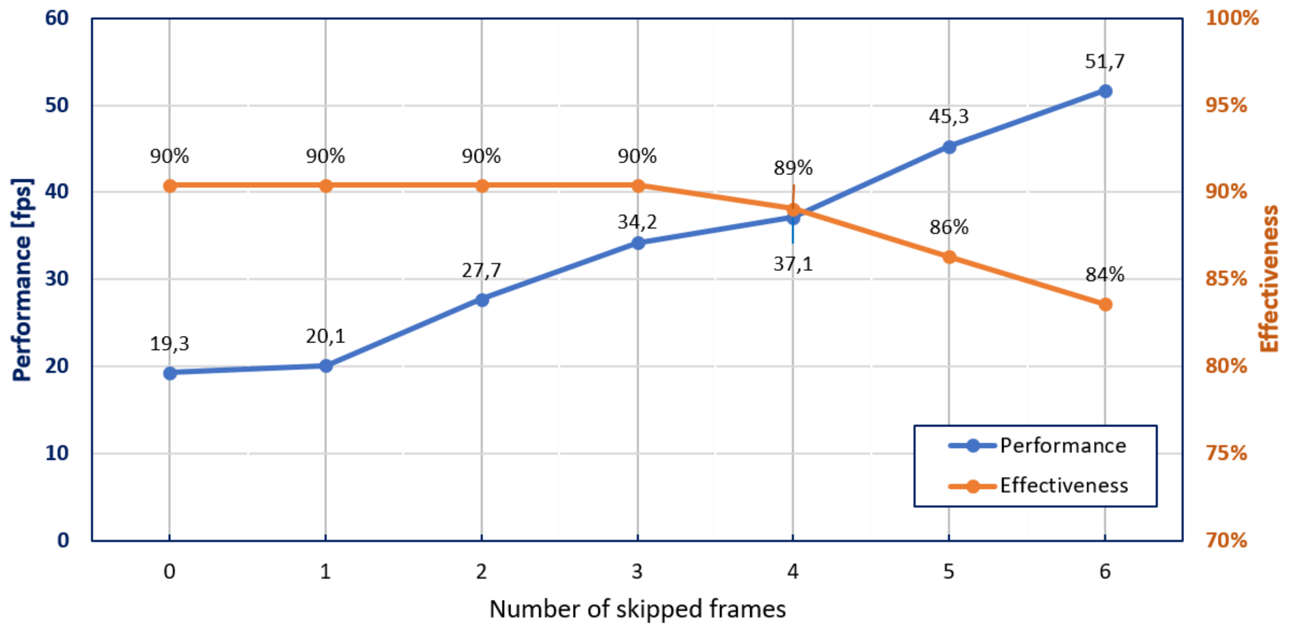


Figure 24: The impact of every (n+1)-th frame processing on the effectiveness and performance of egg detection using template matching method.

The final study was to determine the dependence of the number of processed video frames per second on the number of eggs present in these frames. The analysis was performed for egg detection using one template without skipping any video frames. In the graph from Figure 25 one can see a strong correlation between the number of eggs present in a frame and the performance. The difference between the minimum and maximum performance level is over 20 fps. This means that the speed of the template matching method is highly sensitive to the number of objects in the video frame.

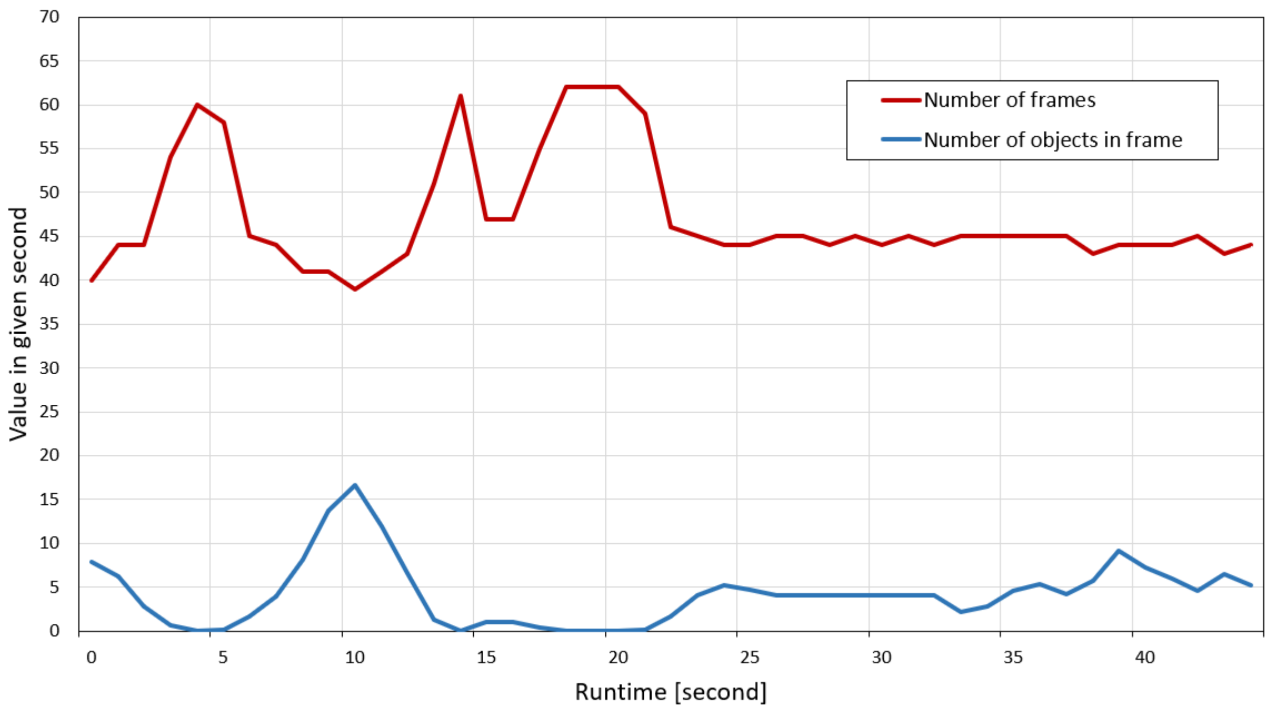


Figure 25: Program performance vs the number of eggs to be detected per video frame for template matching method (film No.1).

4.3 Summary of the effectiveness and performance of egg detection process.

In the previous section we have presented the performance and the effectiveness of egg detection, tracking and counting process for different methods (neural network vs template matching) and models (YOLOv3 vs FR-CNN vs SSD-MobileNet v2). The sequential description could make it difficult for a reader to find out the mutual performance of the methods. In this section we present the summary of the results for methods and models investigated in section 4.2 to identify those of them that allow for being applied on a production line to support a real-time operation. However, we would like to emphasize that our only aim was to verify which of widely applicable CNN or template matching methods will be appropriate for egg detection on the moving industrial tape. We didnt aim at conducting comparative study but decide if the selected methods could be used in industrial application where the clearly defined requirements toward effectiveness and performance (speed) hold. In addition, we present some supplementary results regarding the impact of the film resolution or different egg sizes on the detection, tracking and counting effectiveness.

In Figure 26 we have presented comparison of different methods and models in term of the performance expressed in fps units. The red horizontal line indicates the threshold performance level that determines the possibility to use a given method for real-time operations on the production line. This means that only in three cases (neural networks: YOLOv3 or SSD-MobileNet v2 or template matching method) a real-time operation is possible without the need for configuration and environment changes.

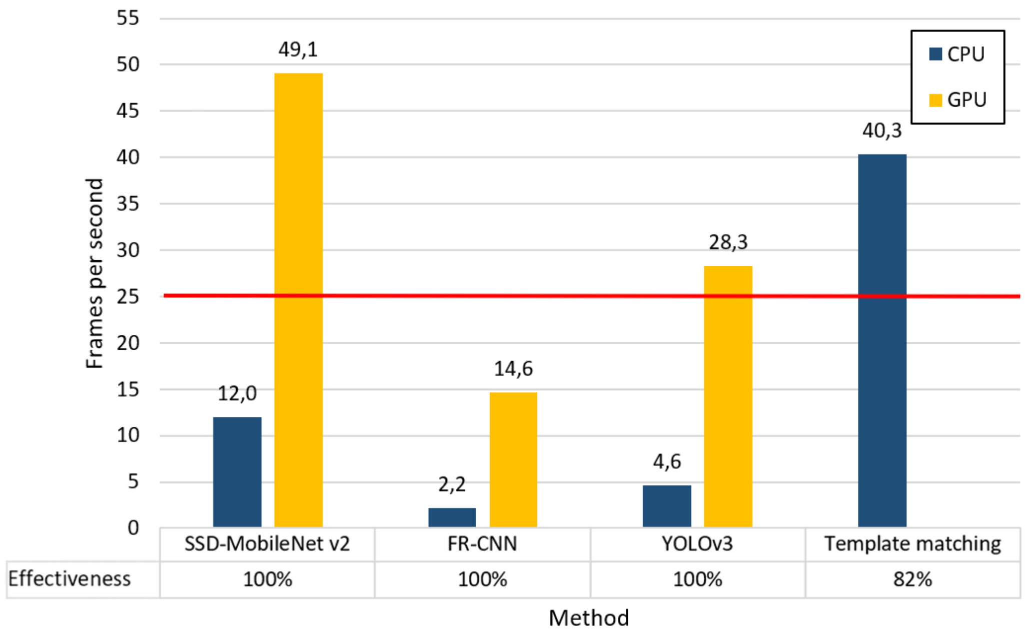


Figure 26: Performance of different methods and models with the usage of CPU or GPU expressed in fps units.

The performance results can vary significantly depending on the hardware environment and various ways to speed up the program. An example of configuration modification aimed at increasing program performance is the detection of eggs on every n-th video frame. Additionally for the template matching method the configuration changes (cascaded templates) can increase the effectiveness of the egg detection process. In this way we can obtained an optimal performance and effectiveness of the egg detection process for a particular environment and method.

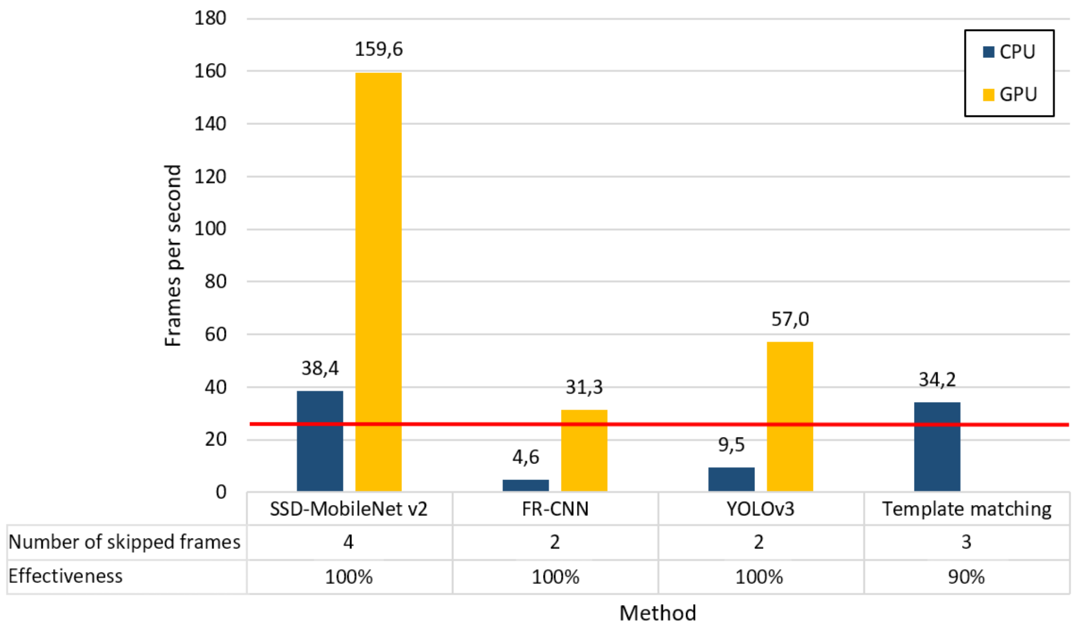


Figure 27: Effectiveness and performance of different methods and models with the usage of CPU or GPU expressed in fps units for the optimal configuration.

In Figure 27 we presented comparison of different methods and models in terms of the performance and the effectiveness of egg detection, tracking and counting process for their optimal configurations. For GPU processing there are 3 models for the neural network method that are applicable for the real-time operation. On the other hand, for CPU processing there are only two methods possible to be applied: template matching or SSD-MobileNet v2 neural network. However template matching method even in optimal configuration could only achieve 90% detection effectiveness. Moreover, as proved in section 4.2, the performance of this method is highly dependent on the number of objects to be detected on a frame. Thats why the neural network based methods are preferred since they exhibit high insensitivity of the processing speed vs the number of objects to be detected on a video frame. Concerning the effectiveness of the egg detection and tracking process, template matching method is quite sensitive to a choice of pictures for the templates. The neural networks based methods do not exhibit such a sensitivity once they are trained correctly on a wide set of images. These facts make the neural network based method more predictable and confident and thus preferable in commercial applications. Table 2 summarizes our results and includes the conclusions about applicability of a given method to egg detection, tracking and counting in real-time.

Method	Processor type	Complexity of training process	Data set preparation time[h]	Standard configuration (processing each frame, usage of single template)		Optimal configuration (processing every (n+1)-th frame, usage of multiple templates)		Does processing speed depend on the number of objects to be detected	Applicability for real-time processes
				Effectiveness	Speed [fps]	Effectiveness	Speed [fps]		
SSD-MobileNet v2	CPU	high	29.5	100%	12.0	100%	38.4	No	Partial
SSD-MobileNet v2	GPU	high	29.5	100%	49.1	100%	159.6	No	Full
FR-CNN	CPU	high	23	100%	2.2	100%	4.6	No	Full
FR-CNN	GPU	high	23	100%	14.6	100%	31.3	No	Partial
YOLOv3	CPU	high	25.5	100%	4.6	100%	9.5	No	No
YOLOv3	GPU	high	25.5	100%	28.3	100%	57.0	No	Full
Template matching	CPU	n/a	1.25	82%	40.3	90%	34.2	Yes	No

Table 2: Summary of all studied aspects of computer vision methods.

Below we have presented supplementary results obtained with low resolution films or different egg sizes. We have called them as supplementary results since they were obtained provoking not typical or very rare conditions. A low resolution of the films is not a typical condition that moreover, could be eliminated be using a better camera.

All of the results presented so far were obtained by using the films with resolution 300x300 that matched the resolution of images used for training neural networks. However, we also verified the effectiveness of egg detection, tracking and counting on the films with a lower resolution. For this purpose we chose the following film resolutions: 150x112, 200x150, 400x300. In 3 tables below we present the results.

Method	Real number of eggs	Lost eggs	Multiple detected eggs	False positives	Not detected eggs	Detection Effectiveness
SSD-Mobilenetv2	73	1	36	0	2	47%
FR-CNN	73	0	0	1	73	0%
YOLOv3	73	0	5	11	0	93%
Template matching	73	0	3	0	1	95%

Table 3: Summary of egg detection and tracking effectiveness on films with 150x112 resolution.

Method	Real number of eggs	Lost eggs	Multiple detected eggs	False positives	Not detected eggs	Detection Effectiveness
SSD-Mobilenetv2	73	0	3	0	0	96%
FR-CNN	73	4	31	0	25	18%
YOLOv3	73	0	0	0	0	100%
Template matching	73	1	0	0	1	97%

Table 4: Summary of egg detection and tracking effectiveness on films with 200x150 resolution.

Method	Real number of eggs	Lost eggs	Multiple detected eggs	False positives	Not detected eggs	Detection Effectiveness
SSD-Mobilenetv2	73	0	0	0	0	100%
FR-CNN	73	0	1	0	0	99%
YOLOv3	73	0	0	0	0	100%
Template matching	73	1	1	0	1	96%

Table 5: Summary of egg detection and tracking effectiveness on films with 400x300 resolution.

According to the presented results, the template matching method seems to be insensitive to the resolution of the processed films. On the other hand, it is evident that the detection and tracking effectiveness for CNN methods is dependent on the resolution of the processed films. This dependency is especially high for SSD or faster R-CNN methods. This is in line with the observations made by other researchers and documented in [36]. They also showed that a CNN exhibited lower detection effectiveness when processing the images with the resolution lower than the ones used during the training process.

Another supplementary result is the investigation how size differentiation among eggs impacts the effectiveness of detection, tracking and counting. In the production environment where we performed the tests, the large variety of sizes was a quite rare situation. This is because the egg size depends on the age and breed of hens. In our experiments all the eggs came from the hens of the same breed and age. However we created extraordinary conditions where the egg sizes differ from L to S size which means 30% difference. An example of provoked size differentiation among eggs on a tape is visible in Figure 28 on eggs number 57, 58, 59.

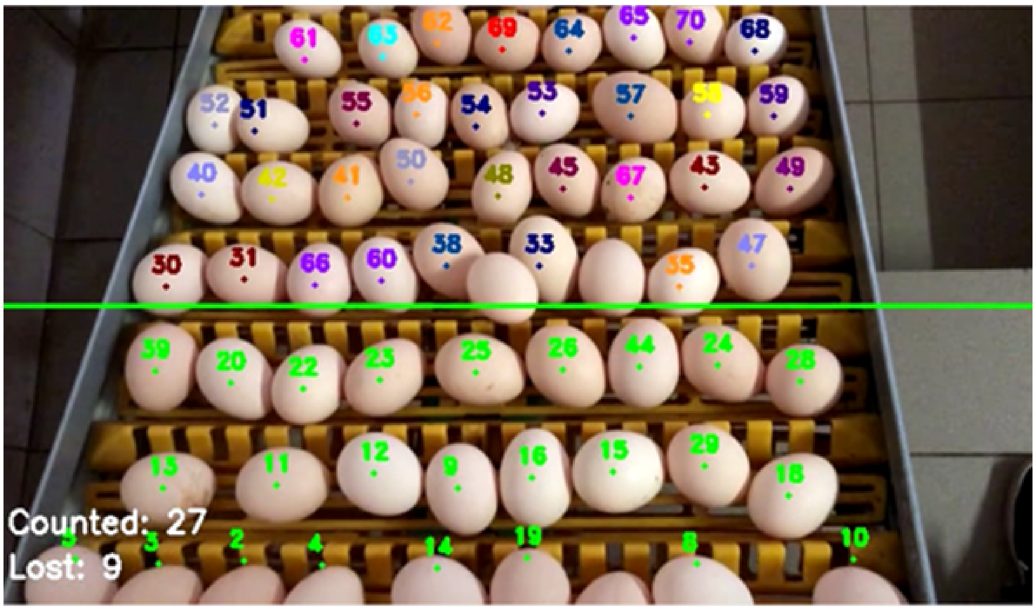


Figure 28: Differentiation of egg sizes visible on eggs number 57, 58, 59.

The summary of the results regarding the effectiveness of detection, tracking and counting process with eggs of different sizes is presented in Table 6.

Method	Real number of eggs	Lost eggs	Multiple detected eggs	False positives	Not detected eggs	Detection Effectiveness
SSD-Mobilenetv2	335	1	18	0	0	94.33%
FR-CNN	335	0	5	0	0	98.51%
YOLOv3	335	0	4	0	0	98.81%
Template matching	335	41	15	0	31	74.03%

Table 6: Summary of egg detection and tracking effectiveness on films different sizes of eggs.

The results reveal that there are mainly multi-detected eggs errors so the problems are related to continuity of the detection and tracking process. From this point of view, template matching method with the effectiveness of 74% seems to be unacceptable for industrial application. However, relatively high number of multi-detected eggs events was also observed for SSD method. It is worth to remind that multi-detected eggs event relates to the situation where the egg was already detected and then not detected on a number of the consecutive video frames and detected again. All these events contribute to the loss of effectiveness which makes this measure very strict. This explains the relatively low values of effectiveness reported in Table 6.

5 Conclusions

Our work aimed at providing the answer which computer vision methods are suitable to be applied for egg detection, tracking and counting on a production line in the real-time. During the investigation of this problem we performed a set of experiments that let to acquire the answers to many other research issues related to the performance and the effectiveness of egg detection, tracking and counting process. We evaluated the performance and the effectiveness of different methods (template matching vs neural network based) and models (YOLOv3 vs FR-CNN vs SSD-MobileNet v2). We studied the impact of adjusting the values of configuration parameters in order to obtain optimal operation of the examined methods and models including such techniques like skipping the processed video frames or applying cascade templates. These experiments let us obtain tangible results that helped to decide which methods are applicable in commercial production lines.

Based on the conducted experiments we can draw a number of conclusions. First of all, template matching method is not acceptable for on-line application on a production line since it has too low effectiveness, something around 80%. Even with multiple templates applied in a cascade, the effectiveness of detection, tracking and counting process is not more than 90%. In addition, it is achieved at the cost of a two-fold loss in the performance. The performance of template matching method is even poorer with eggs of different size and it is dependent on the number of eggs to be detected on the frame. Surprisingly, this method is quite insensitive to the changes of film resolution. The second, CNN methods exhibit good detection, tracking and counting effectiveness, especially with eggs of similar size and good resolution films. When the size of eggs differs significantly (S vs L size of eggs) the best performing were FR-CNN and YOLOv3 while the SSD-MobileNet v2 was a little bit worse which we quantified as 5% less. For low resolution films, we identified that YOLOv3 was the best performing with 93% effectiveness of detection, tracking ad counting process. However, regarding the performance it appeared that the best one was SSD-MobileNet v2 and it was the only network being able to work on-line on CPU for our application. Moreover, the important feature of CNN methods is their performance insensitivity to a number of eggs to be detected on the video frame. The third, the performance of any method depends on the parameters of the hardware that is used. Our experience indicates that it is preferable to install and compile the software environment to be used with dedicated GPU instead of CPU t is possible to improve the performance by skipping a number of frames without a loss in the effectiveness (optimal value of n), however it improves the performance for CPU or GPU in the same way. Because of presented arguments and obtained results we plan to deploy industrial version of computer vision detection, tracking and egg counting process based on GPU and YOLOv3 network. As a final word we would like to remark that the eggs were the objects under detection in our tests. The eggs feature similar and regular shapes. Taking into account this fact a little bit surprising might be the differences in performance among the models of a neural network based method as well as low effectiveness of egg detection for template matching method. We hope our work shed new light on the applicability of computer vision methods for real-time egg detection on a production line. Our results might be useful for developers as well as system designers applying innovative technologies.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press. 2016.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, *TensorFlow: A System for Large-Scale Machine Learning*, USENIX, 2016.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng. *TensorFlow: Large-scale machine learning on heterogeneous distributed systems*, arXiv preprint, 2016.

[4] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, OReilly Media, 2017.

[5] https://github.com/pjreddie/darknet Accessed 25 Apr 2020.

[6] https://docs.opencv.org/ Accessed 30 Aug 2020.

[7] https://opencv.org Accessed 18 Aug 2020.

[8] P. Rajeshwari, P. Abhishek, P. Srikanth, T. Vinod, "Object Detection: An Overview", *International Journal of Trend in Scientific Research and Development (IJTSRD)*, 1663-1665, 2019.

[9] Liu W. et al., "SSD: Single Shot MultiBox Detector.", *In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision ECCV 2016*, Lecture Notes in Computer Science, vol 9905. Springer, 2016.

[10] Guoqiang Ren, Tao Lin, Yibin Ying, Girish Chowdhary, K.C. Ting, "Agricultural robotics research applicable to poultry production: A review", *Computers and Electronics in Agriculture*, Volume 169, 2020, 105216, ISSN 0168-1699, https://doi.org/10.1016/j.compag.2020.105216.

[11] Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.;Wojna, Z.; Song, Y.; Guadarrama, S., "Speed/accuracy trade-os for modern convolutional object detectors." *In Proceedings of the IEEE Conference.*

[12] Wang, D.; Tang, J.; Zhu, W.; Li, H.; Xin, J.; He, D., "Dairy goat detection based on Faster R-CNN from surveillance video.", *Comput. Electron. Agric.*, 154, 443449, 2018.

[13] Nasirahmadi, A.; Sturm, B.; Edwards, S.; Jeppsson, K.-H.; Olsson, A.-C.; Mller, S.; Hensel, O., "Deep Learning and Machine Vision Approaches for Posture Detection of Individual Pigs.", *Sensors* , 19, 3738, 2019.

[14] Li G, Xu Y, Zhao Y, Du Q, Huang Y., "Evaluating Convolutional Neural Networks for Cage-Free Floor Egg Detection.", *Sensors (Basel)*, Jan 7;20(2):332, 2020, doi: 10.3390/s20020332. PMID: 31936028; PMCID: PMC7013917.

[15] O. Geffen, Y. Yitzhaky, N. Barchilon, S. Druyan, I. Halachmi, "A machine vision system to detect and count laying hens in battery cages", *Animal*, Volume 14, Issue 12, Pages 2628-2634, 2020, ISSN 1751-7311, https://doi.org/10.1017/S1751731120001676.

[16] A. Fakhri A. Nasir, S. Sabarudin, A. Majeed, A. Shahrizan, A. Ghani, "Automated egg grading system using computer vision: Investigation on weight measure versus shape parameters", *IOP Conference Series Materials Science and Engineering*, 342(1):012003, 2018

[17] https://www.youtube.com/watch?v=FY3OhQceZO0 Accessed 01 Sep 2020.

[18] https://www.youtube.com/watch?v=3wJ0T371O8o Accessed 01 Sep 2020.

[19] M. Lutz, *Learning Python*, OReilly Media, 2013.

[20] https://python.org Accessed 18 Aug 2020.

[21] A. Rosebrock, *Deep Learning for Computer Vision with Python*, PyImageSearch, 2017.

[22] https://github.com/tzutalin/labelImg Accessed 30 Aug 2020.

[23] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*, John Wiley & Sons, 2009.

[24] G. Bradski, A. Kaehler, *Learning OpenCV. Computer Vision with the OpenCV Library*, OReilly Media, 2008.

[25] "https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md", Accessed 20 Apr 2020.

[26] C. Sammut, G. Webb, *Encyclopedia of Machine Learning and Data Mining*, Springer, 2017.

[27] D. Conway, J. White, *Machine Learning for Hackers*, OReilly Media, 2012.

[28] J. Redmon, A. Farhadi, YOLOv3: *An Incremental Improvement*, University of Washington, 2018.

[29] Ren S, He K, Girshick R, Sun J, ”Faster R-CNN: towards real-time object detection with region proposal networks”, *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems*, Volume 1, December, Pages 9199, 2015.

[30] https://www.tensorflow.org/tensorboard Accessed 30 Aug 2020.

[31] J. Liu, J. Liu, W. Du, D.Li, ”Performance Analysis and Characterization of Training Deep Learning Models on Mobile Devices”, 2019, arXiv:1906.04278v2.

[32] J. Dai, Y. Li, K. He, J. Sun, ”R-FCN: Object Detection via Region-based Fully Convolutional Networks”, arXiv:1605.06409v2, 2016.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, ”Rethinking the inception architecture for computer vision”, *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 2818-2826, Jun. 2016.

[34] Q.-C. Mao, H.-M. Sun, Y. B. Liu, and R.-S. Jia, ”Mini-YOLOv3: Realtime object detector for embedded applications”, *IEEE Access*, vol. 7, pp. 133529133538, 2019.

[35] https://pjreddie.com/darknet/yolo/ Accessed 25 Apr 2020.

[36] Kannojia S P, Jaiswal G, ”Effects of Varying Resolution on Performance of CNN based Image Classification: An Experimental Study”, *International Journal of Computer Sciences and Engineering*, Vol.6, Issue.9, pp.451-456, 2018.