International Journal of Mathematical, Engineering and Management Sciences
Vol. 10, No. 4, 874-895, 2025
https://doi.org/10.33889/IJMEMS.2025.10.4.042

Ram Arti Publishers

The Formation of Artificial Data based on a Conveyor Enterprise

A. V. Zaripov
Department of Data Processing,
Tomsk State University of Control Systems and Radio Electronics, Tomsk, Tomsk State, Russia.
Corresponding author: aleksei.v.zaripov@tusur.ru

R. S. Kulshin
Department of Data Processing,
Tomsk State University of Control Systems and Radio Electronics, Tomsk, Tomsk State, Russia.
E-mail: roman.s.kulshin@tusur.ru

A. A. Sidorov
Department of Data Processing,
Tomsk State University of Control Systems and Radio Electronics, Tomsk, Tomsk State, Russia.
E-mail: anatolii.a.sidorov@tusur.ru

(Received on October 25, 2024; Revised on December 27, 2024; Accepted on February 13, 2025)

Abstract
The research is devoted to the analysis and development of methods for generating artificial data in order to solve computer vision problems that arise during the operation of conveyor-type technological lines. The paper substantiates the relevance of this problem, as traditional approaches to data collection, such as video recording and manual markup, are not only time-consuming but also ineffective in terms of cost and efficiency. These methods are often not applicable in situations where a large amount of data needs to be processed or in cases where efficiency is required. As an alternative approach, the paper proposes a generalized model for synthetic data generation using modern game engines and 3D modeling techniques. This model allows for a high degree of variability and realism in the generated data, making it possible to simulate various scenarios for conveyor lines and specific computer vision tasks. The effectiveness of the proposed approach was experimentally confirmed using the example of the problem of determining laminate coatings. Synthetic data generated using the proposed model were used to train the YOLOv8 neural network as part of the experiment. The test results showed a high accuracy of the trained model, with an *mAP*50 of 0.95, indicating the significant potential of synthetic data for improving the quality of machine learning models. These results confirm the possibility of using synthetic data when access to real datasets is limited. This opens up opportunities for optimizing neural network learning processes and improving the efficiency of computer vision solutions, particularly in industrial applications.

Keywords- Data generation, Neural network, Synthetic data, Computer vision, Unity.

1. Introduction

As part of the implementation of a large number of production processes that involve multi-stage transformations of raw materials into finished products, it is necessary to ensure the quality control of intermediate and/or final results. This involves identifying and rejecting objects that do not meet certain standards (regulations, quality standards, etc). To control and evaluate the quality of products during production, companies usually rely on the visual inspection of the product by a human, which can be challenging for several reasons:

- the low rate of assessment and classification, which may not correspond to the quality parameters, is due to the physiological characteristics of the human visual system;
- subjectivity of perception (omission of objects of inadequate quality or incorrect classification related to the qualification of a quality assurance specialist, his fatigue, loss of concentration, etc.);

• the cost of this stage in the production process does not directly add value to the product, and is expressed through the need to employ a specialized professional.

To solve these problems, companies are increasingly turning to the introduction of various digital solutions based on computer vision systems (CV), which are used in various industries and for individual production tasks. These solutions include traffic control (Serrano et al., 2005; Blackledge and Dubovitskiy, 2011; Greeshma et al., 2019; Liu et al., 2021), metallurgy (Rusanovsky et al., 2022; Harikrishna et al., 2023; Sarrionandia et al., 2023), territory protection and safety (Gupta and Garima, 2014; Aydin and Othman, 2017; García et al., 2017; Lan et al., 2024), agriculture (Arakeri, 2016; Costa et al., 2011; Dhanya et al., 2022), and more. The technology is used to automate various routine and attention-intensive tasks.

For example, Esteva et al. (2021) discusses possible applications of technology for the detection and classification of diseases in medical images, such as ultrasound images and images of tissue samples under a microscope.

CV can also be used in the printing industry to improve the efficiency of quality control tasks for printing equipment, ultimately reducing the cost of products. As part of the manufacturing process, separate components for production equipment are created - printed cylinders with engravings designed to apply necessary images or text onto paper. In Villalba-Diez et al. (2019), deep neural networks are trained to detect various types of defects in printing machine parts. These defects include dents, scratches, inclusions, bends, dis-placement, excessive or missing printing, and color errors that can occur during the engraving process. Photographs of printing machine parts were used as training data, with the operator classifying them as either suitable for printing or unsuitable due to defects. During testing, the model achieved an accuracy of 97.85% for truly negative responses, meaning it correctly identified parts that did not have any defects. The model also had a high number of truly positive responses, at 99.01%. This high accuracy value is significant considering the complexity of the task and the variety of devices that can occur in printing machine parts.

Even a brief analysis of the previous implementations of the CV allowed us to identify two types of tasks: continuous (production using a conveyor belt) and discrete (product evaluation, only, if necessary, with an indefinite interval) visual inspection. In the continuous visual inspection task, additional challenges arise related to collecting training data:

• it is not always possible to record a video of the technological process in order to create a dataset of the desired size and variety. This may be due to limitations related to the production process, such as a ban on recording confidential stages of product manufacturing;
• the process of obtaining permission to install video cameras and collect data can take a considerable amount of time, as it is necessary to obtain approval from relevant authorities in order to comply with specific regulatory requirements;
• high costs for manual labeling of training images are associated with the occurrence of error due to human error, which in turn affects the efficiency of the neural network.

To address the issues described, a solution is proposed that relies on the creation of synthetic datasets. This approach would significantly reduce the time and effort re-quired to obtain training data (images of the production process and associated annotations), while still meeting the necessary volume and variety requirements.

This study was conducted to fill a gap in the scientific literature regarding the effectiveness of synthetic datasets in visual control tasks in streaming environments. Although synthetic data has been widely used

in related fields, its potential for more complex production processes with specific limitations remains underexplored. The lack of empirical evidence on the practicality and limitations of these approaches hinders the development of generalizable solutions that can be adapted to real-world conditions.

The research aims to develop a methodology that not only validates the applicability of synthetic data but also demonstrates its benefits in situations where accessing real data is limited. By addressing this gap, we hope to contribute to the advancement of more efficient and effective decision-making processes in various industries.

2. The Concept of Data Generation

Data scarcity has been a long-standing challenge in machine learning, and several solutions have been proposed to address it, including the use of synthetic datasets. For example, Kiefer et al. (2022) describes the process of creating artificial data to simulate the operation of unmanned aerial vehicles (UAVs), with the aim of detecting and classifying them. The dataset was created using the GTA V video game, which features high-quality graphics with high-resolution textures, volumetric lighting, and simulated weather conditions. It also has low system requirements, making it accessible to a wide range of users. The game was used in conjunction with the DeepGTAV framework to generate annotated data.

As part of the experiments, nine models were trained using a combination of three different neural network architectures and three datasets. The quality of the classification and object boundary (bounding box) allocation was evaluated using the mean average precision metric (*mAP*) with a threshold value of 0.5 (*mAP*50). Based on the results of testing, estimates of the performance of the models are presented in **Table 1**.

Table 1. Model quality estimates based on *mAP*50 metrics.

Neural network model	Syntetic	Real	Mixed
EfficientDet-D0	1.2 – 29.2	24.6 – 78.4	27.2 – 85.8
Faster R-CNN	2.4 – 38.8	48.6 – 90.5	51.2 – 91.5
YOLOv5	10.2 – 64.2	43.9 – 88.8	45.0 – 86.9

The *mAP*50 results for models trained only on synthetic data were unsatisfactory, as the environment did not allow for the generation of classes that were present in the test set. This is why the percentage values of the metric ranged from 1.2% to 64.2%, while for models trained on real data, the *mAP*50 values were between 24.6% and 90.5%. If synthetic and real samples are combined during training, there is a noticeable improvement in the accuracy of the detection model. The percentage of average accuracy for models obtained using this approach varies between 27.2% and 91.5%. Therefore, synthetic data can be used to augment the dataset with real images.

At the same time, computer games do not provide for variability in creating narrow-profile systems, as they lack advanced tools for designing models and configuring the three-dimensional space of the "virtual world", also known as a scene. In such cases, it is recommended to use 3D modeling software, such as Blender, or game engines like Unity and Unreal Engine.

If we consider the use of 3D modeling tools for generation, then, for example, a data generation system was developed for the task of detecting pipe defects at a thermal power plant using UAVs as part of work (A synthetic data generator. Training of neural networks for industrial flaw detection. Available online). The Blender tool was used as the development tool, which allows the creation of objects with photorealistic graphics, and Python was used for writing scripts. As part of the detection process, the main types of defects are metal corrosion, runaway colors (rainbow-like patterns on metal), and cracks. To create textures that

resemble these defects, Perlin noise generation algorithms were employed. The resulting images were annotated using a third-party pro-gram written in Python, as the 3D editor did not provide the ability to mark up the images. To solve this issue, it was necessary to create two images: the original image of the pipes and a defect mask. Then, the Python script used recursive techniques to determine the coordinates of defects and write them to a markup file for the original image. Despite the photorealistic quality of the images, this solution has one significant drawback - low generation speed. Since rendering does not occur in real time, it can take about half an hour to create a one-minute video at a good speed of one frame per second. To address this disadvantage, game engines can be used that allow for real-time rendering and processing of more than 60 frames per second, eliminating this limitation.

For instance, the Unity game engine provides an experimental Unity Perception package that enables you to automate the process of labeling images for tasks like segmentation, two-dimensional and three-dimensional object tracing, counting past objects, and more (The Perception Camera Component. Available online).

In Reutov et al. (2022), the Unity game development platform and the Unity Perception package were used to generate data for training a neural network that detects manufactured metal bolts on a conveyor belt. This approach allowed for a frame rate of 6 frames per second, significantly reducing time costs. However, during the process, testing for the accuracy of neural network training was not conducted.

As part of the system testing, three models were trained. The first model was trained on 1,200 real images, the second model was trained on 120,000 generated images, and the third model was trained on a mix of real and generated data. Additionally, to compare the speed of creating the data sets, the time it took to create each dataset and the accuracy of the models trained on these datasets were provided in the paper (**Table 2**).

Table 2. Data collection time and model performance indicators.

Metric	Syntetic	Real	Mixed
Time of data collection, h.	7.25	185.00	192.25
Sample size, pcs.	120000	1200	121200
mAP	0.67	0.76	0.85

The first model achieved a *mAP* of 0.76, while the second achieved a result of 0.67. These results are not comparable, given the time lost for data collection. As expected, the third model achieved the highest result with a *mAP* of 0.85. While the extended sample had a positive impact on the training of the neural network, it should be noted that it took the longest time to create a mixed data set.

The options considered solve the problem in each case of data shortage, but each has its own advantages and disadvantages. For instance, the method of generating images using a 3D editor may not provide the ability to create data quickly, as image rendering occurs at a relatively slow speed. However, it does provide photorealistic images. The solution that uses GTA V does not provide the ability to generate data for other tasks, such as production processes, which is a significant drawback. However, it does allow you to generate real-time data from UAVs. The metal bolt production data generation solution is a simple graphic implementation that does not utilize the rendering capabilities of the game engine. As a result, the generation rate is only 6 frames per second. The traffic sign detection data generation system currently does not have streaming generation capabilities, as each frame is generated separately. This significantly reduces the speed of image acquisition, while providing a variable selection of different objects, including background objects.

To increase the speed of data generation and improve the visual quality of images, as well as to achieve the realism of synthetic data, a new solution based on a generalized list of tasks is proposed. This solution will be used to build a synthetic data generator for the task of identifying objects in various industries.

3. Generalized Model of the Synthetic Data Generator

The task of identification is to detect objects of interest that need to be recognized and categorized by the computer vision system. For example, in order to control the quality of bricks in production, the object of interest would be brick defects. Similarly, if the task is to detect vehicles with UAVs (Unmanned Aerial Vehicles), then the objects of interest would be vehicles of different types.

To effectively solve identification problems within the framework of conveyor production, a generalized model of generator formation is proposed. It consists of several interrelated stages that simulate the production process:

- formation of a digital twin of the conveyor system (creation of 3D models of the conveyor band, creation of 3D models of the sides of the conveyor, texturing of the models of the belt and limits);
- creation of the object of interest (creation of 3D models of the desired object, texturing the models of the object, creation of algorithms for manipulating the object);
- simulation of a real technological process (creation of algorithms for moving objects along the digital twin conveyor system, creation of algorithms to modify the digital twin and apply effects, creation of algorithms that change the angle of the production chamber).

The consistent implementation of this set of tasks will form the basis for a synthetic data generator for any production line. As an example, later in the paper, we will con-sider a special case of forming a laminate production model. As part of the general list of tasks, we developed specialized software, the diagram of components of which is shown in **Figure 1**.

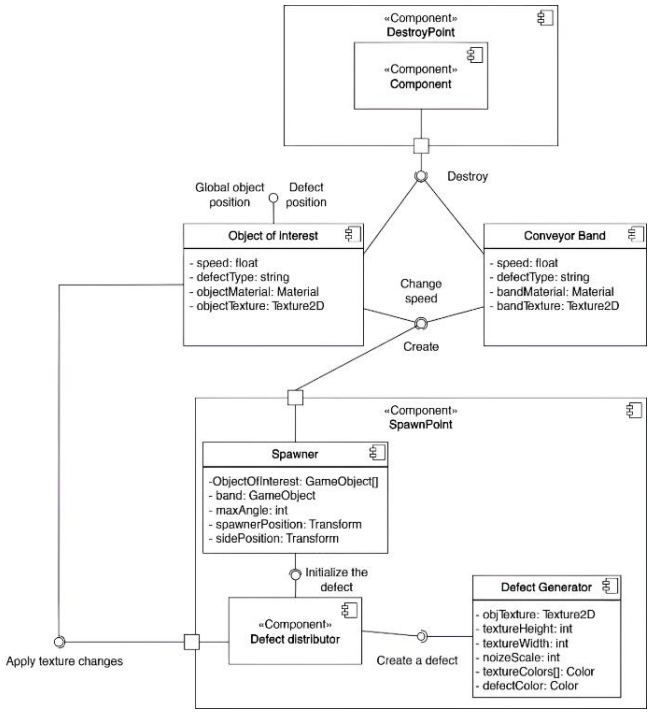


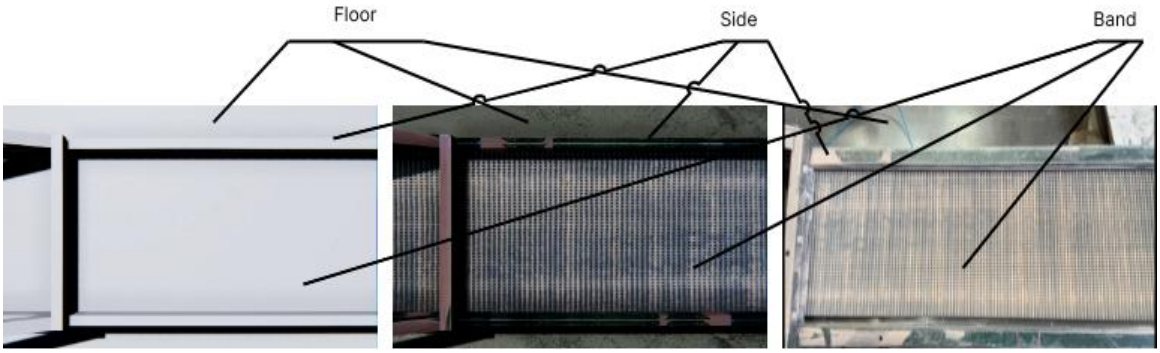
Figure 1. Diagram of software components.

Situations are generated by the work of 8 main components of the scene responsible for the creation and destruction of objects of interest, the imposition of defects, the simulation of the conveyor line, and the expansion of the image sample.

3.1 Formation of the Digital Twin of the Conveyor Line

The system uses the Unity game engine, which allows for real-time scene rendering, the accompanying C# programming language, and the High-Definition Render Pipeline (HDRP) technology, which provides high-quality images.

Since the generalized list assumes the creation of a digital twin of production, a representation of the pipeline was formed from the primitives of the game engine (**Figure 2(a)**), which are parallelepipeds. Next, high–resolution textures were superimposed on the generated representation (**Figure 2(b)**), which were created based on a real prototype - a conveyor line from an enterprise producing floor coverings (**Figure 2(c)**).



(a) (b) (c)

Figure 2. Conveyor line: (a) representation of primitives; (b) textured model; (c) real prototype.

As a result of the first stage of the generalized model, a three-dimensional model of the conveyor has been created, which will be used to simulate the manufacturing process.

3.2 Formation of an Object of Interest

The object of interest in the framework of the process under consideration is a laminate board, which in the future will need to be detected using a short circuit system. Accordingly, it is important to reflect the maximum number of variations of an object in a set of synthetic data.

To create objects of interest, 10 high-resolution textures of laminate board and their corresponding normal maps were created. The choice of this number of laminate textures was based on the need to cover the main visual and textural variations found in real-world operating conditions, as this amount allows us to create a sufficient amount of synthetic data for the correct training of the CV system. These textures were then superimposed onto 3D models of parallelepipeds, resulting in the final set of models shown in **Figure 3**.

Next, algorithms for modifying the boards were developed. To create a sample range of variations, it was decided to increase the number of elements of interest by introducing defects. For this purpose, the components "Defect Distributor" and "Defect Generator" were created.

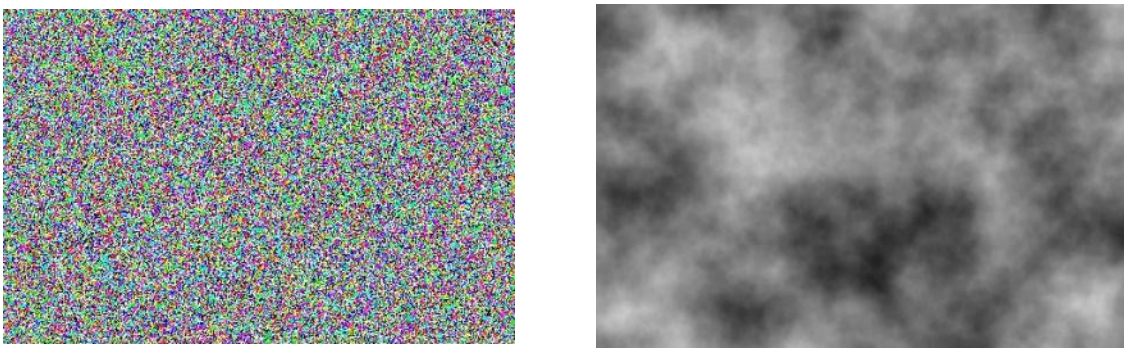
The "defect distributor" component is responsible for initiating the process, pre-paring the object, selecting a current defect, and passing the task onto the "defect creator" component. The latter is responsible for creating and installing the defect on the designated object. The "defect creator" can create three types of defects:

- "printing defect" is a separate section of the printed pattern on a part that differs in color and texture from an identical coating;
- "glue spot" is a light gray spot on the outer surface of the cladding, formed as a result of the penetration of glue from the adhesive layer located under it;
- "clipping" of the edge material along the edges of the part – chipping of the edge material along the entire length of the part or part of it.

The defect selection algorithm involves obtaining a random number between 0 and 1, which is used to distribute defects as a percentage in order to ensure the balance of classes in the training sample.

The "printing defect" and "glue stain" are generated according to an algorithm based on Perlin noise, which is a type of random number generator that produces smooth transitions between values. The algorithm is implemented using the Unity game engine. Perlin noise differs from standard random noise **Figure 4(a)** in that it provides more gradual changes between values, as shown in **Figure 4(b)**. This allows for more realistic and visually appealing results when generating these effects.

Figure 3. Formed models of laminate boards.



(a) (b)

Figure 4. Noise visualization: (a) random noise; (b) Perlin noise.

The "printing defect" and "adhesive stain" are formed according to the algorithm shown in **Figure 5**.

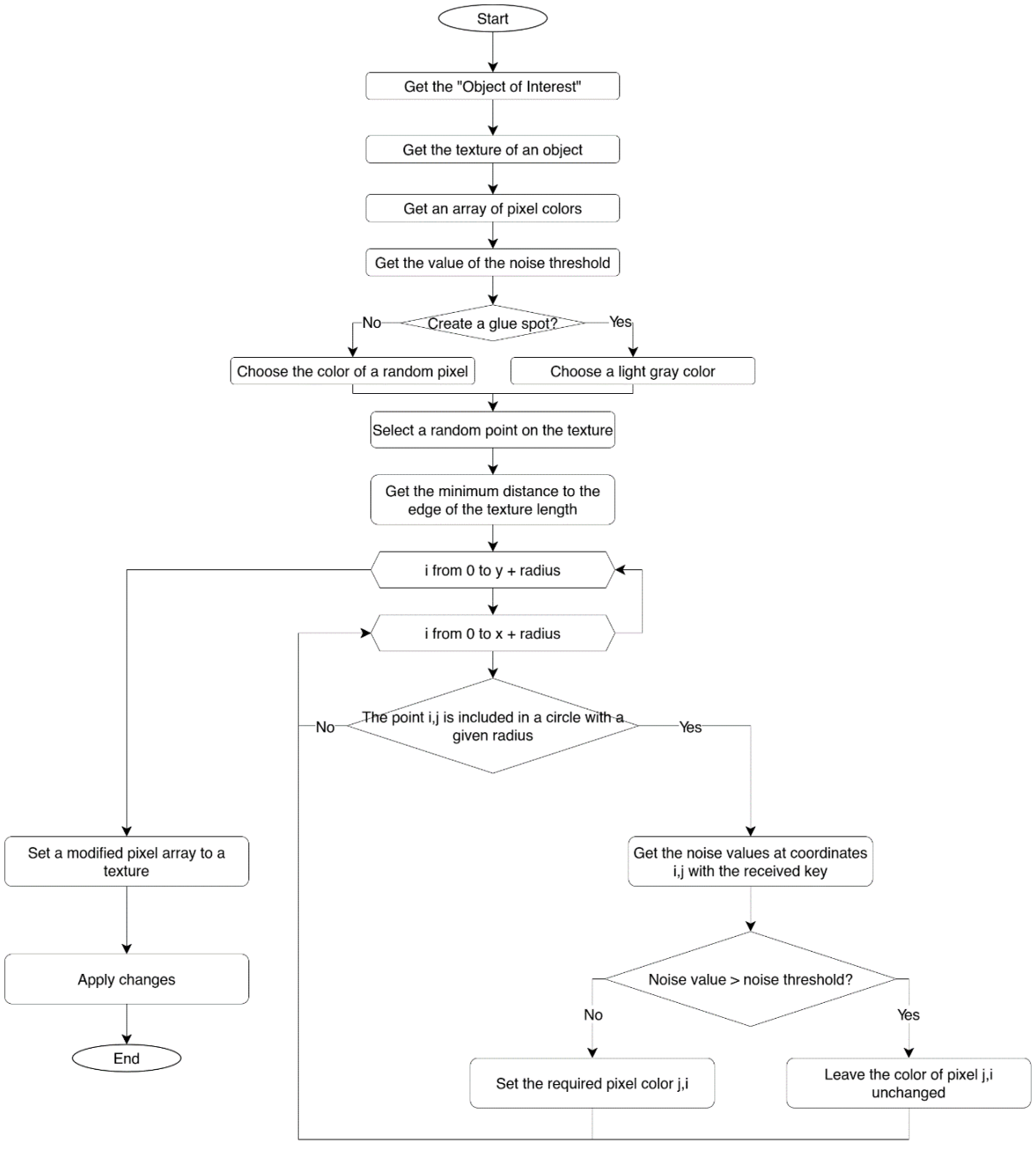
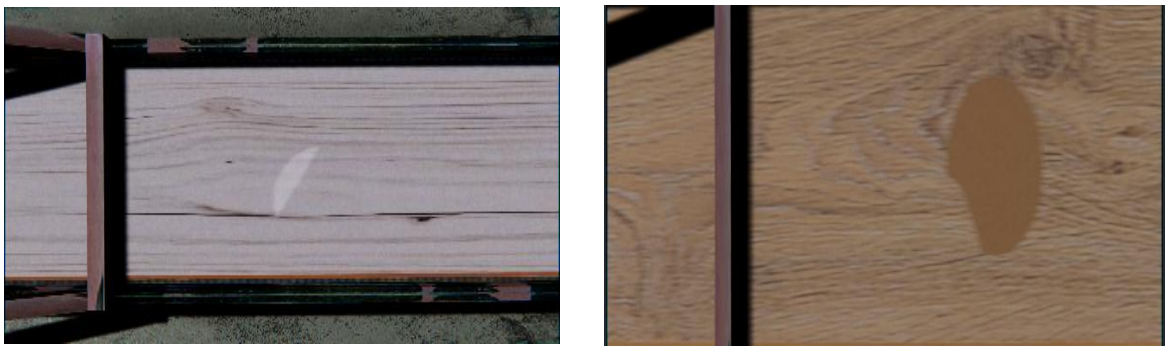


Figure 5. Algorithm for the formation of a "printing defect" and " glue spot".

Using Perlin noise, a smoothed texture can be created from a random template, providing greater realism. This technique is used in various fields, including medicine (Dustler et al., 2015), information security (Bazuhair and Li, 2020), and relief generation (Li et al., 2015). Perlin noise is chosen as the basis for the

spot formation algorithm because it allows for the creation of smooth pseudo-random textures on the surface of an object of interest. Initially, a spot with a random size is selected on the laminated board. The spot cannot exceed the minimum distance from the center to the nearest edge of the object. Once the spot area has been determined, the noise value for each pixel in the texture is calculated. The noise value is in the range of [0, 1]. If the value is below a user-defined threshold, the color of the pixel is determined by the spot color specified. To avoid repetition of the spot, the coordinate value is shifted by a randomly generated seed number. The result of the spot generation process is shown in **Figure 6**.



(a) (b)

Figure 6. Examples of defect formation: (a) "glue spot"; (b) "printing defect".

The images of laminate boards (**Figure 6**) show that the proposed algorithm allows for the creation of pseudo-random textured patterns on the surface of the board. This greatly expands the range of objects of interest and helps train the neural network with a wider variety of data. Consequently, the quality of training improves, which positively impacts the subsequent identification of objects in manufacturing processes.

The chip generation algorithm shown in **Figure 7** is based on changing the transparency of a selected pixel by lowering the value of its alpha channel from 100 to 0.

Initially, the algorithm randomly selects a number of clips between 1 and 4, which corresponds to the number of corners on the board. Based on this number, the corners that need to be removed are randomly selected and the process of creating the defect begins.

After selecting the angle for chipping, the size of the chip is randomly chosen. This is the radius of a circle centered on the extreme pixel of the angle. For example, if the board texture has a size of 2048x512 pixels, pixels with coordinates (0, 0), (0, 512), (2048, 0) and (2048, 512) could act as the center of the chip. The chip itself is a sector of the board that enters the area from the edge of the texture, defined by the circle.

Figure 8 shows a circle with a radius of 100 pixels in the corner of a board with dimensions of 4096 pixels wide and 980 pixels high. The hatching indicates the portion of the circle where the values of the alpha channel in pixels are being changed, while the area without hatching represents the portion that is unaffected.

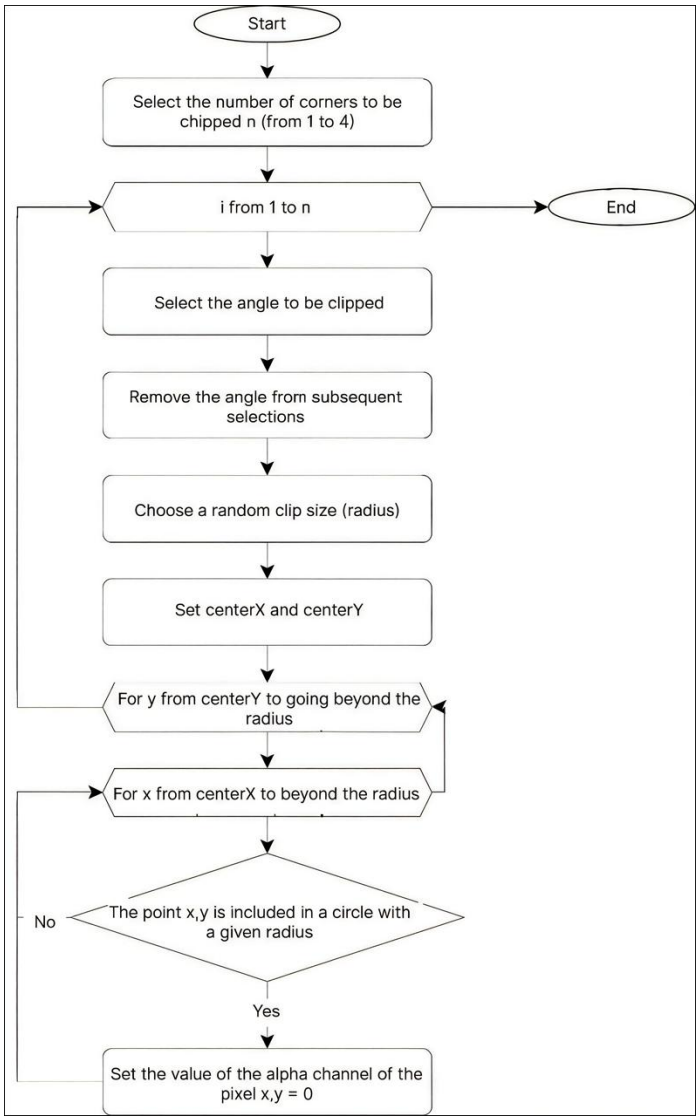


Figure 7. Chip generation algorithm.

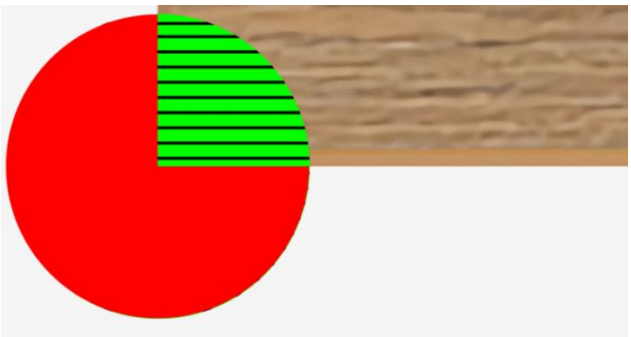


Figure 8. The sector of the circle defining the chip.

The next step is to modify the pixels themselves. Based on the size of the clip, the pixels can be altered. In accordance with the above instructions, it is necessary to use the circle equation.

$$(x - x_0)^2 + (y - y_0)^2 = radius^2 \quad (1)$$

where, x and y are the coordinates of the current pixel; x_0 and y_0 are the coordinates of the angular pixel; radius is the radius of the circle (the size of the chip).

If a pixel with the given x and y coordinates falls within the radius of the circle, its alpha value is set to 0. Otherwise, the pixel remains unchanged. This is illustrated in **Figure 9**.

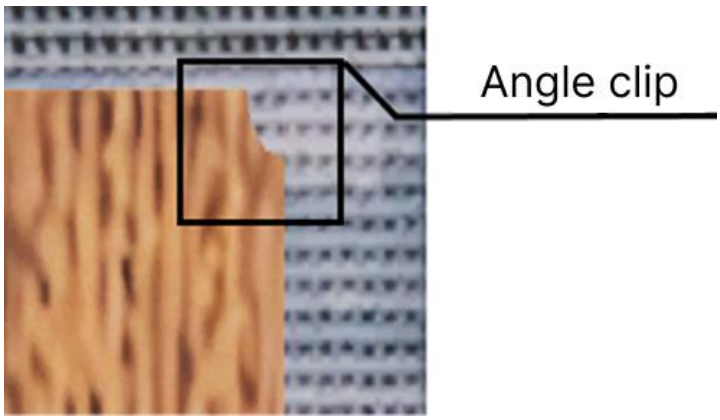


Figure 9. An example of the result of the formation of a "Chipped angle" defect.

As a result of these algorithms, the number of possible board variations increases many times over. This means that we will have a much larger sample of different objects to train neural network on, without having to use identical images. This will have a positive impact on the neural network's generalizing ability and accuracy in detecting objects.

3.3 Simulation of a Real Technological Process

To simulate conveyor production, we created the following components: "Object of Interest", "Conveyor Belt", and "Spawner". The "Spawner" is responsible for generating a variation of the "Object of Interest" and adding it to the scene, along with the "Conveyor Belt". Before placing an "Object of Interest" on the stage, its position is determined randomly, taking into account the dimensions of the conveyor belt. Based on these dimensions, the displacement of the "Object" from the center of the belt is set within a range (-*distance*, *distance*). The distance is calculated using a formula:

$$distance = |pos_1 - pos_2| \quad (2)$$

where, pos_1 is the position of the "Spawner" object; pos_2 is the position of the "Side" object.

After selecting the offset of the object, the angle of rotation in space is randomly chosen from the range [-max_angle, max_angle] to prevent collisions with other objects. The maximum rotation angle is calculated using a formula:

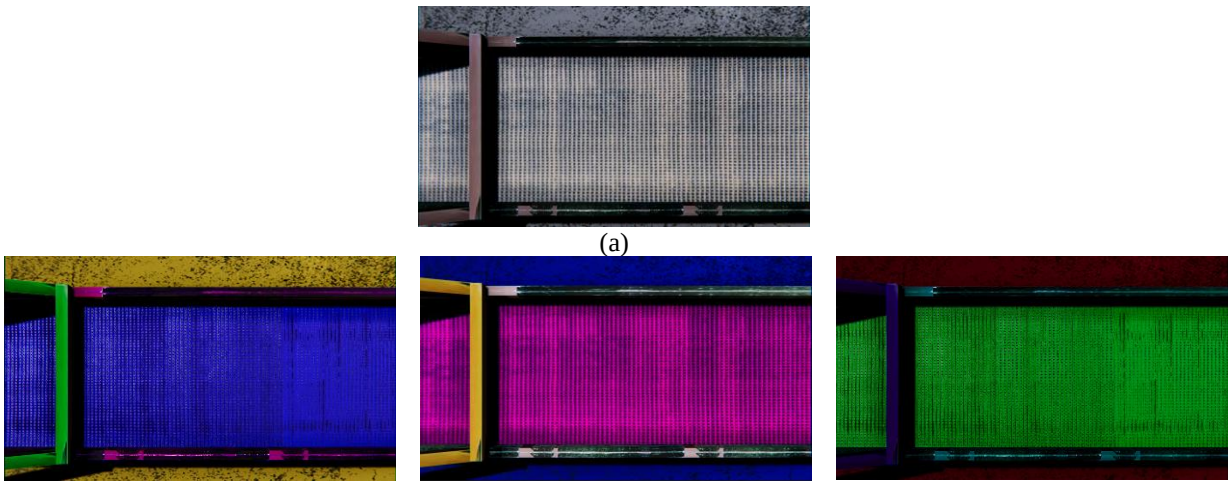
$$max\_angle = arctg\left(\frac{w}{l/2}\right) \quad (3)$$

where, w is the minimum distance from the object of interest to the edge of the conveyor belt; l is the length of the object of interest.

The "Object of Interest" and "Conveyor Belt" components are responsible for storing information about the object's position and movement in space, from its creation to its destruction, at a predetermined speed. This speed is set during the preparation phase, in the "Parameters" section.

To achieve a variety of images, a color-changing algorithm for the conveyor line has been developed. This helps to avoid overfitting the model on similar images, which can lead to a loss of generalization ability and a rapid decrease in the accuracy of predictions on data other than the training sample (Ying, 2019).

During the generation process, the operation time is recorded. After the user-defined time has elapsed, one background object is randomly selected, and its color characteristics are changed. The color of the background object is modified by sequentially changing the values of the three-color channels (red, green, blue) of the RGB color space between 0.0 and 1.0. The initial background image is shown in **Figure 10(a)**. The result of applying the color changes to the background object can be seen in **Figure 10(b)**.



(a)

(b)

Figure 10. Changing the color characteristics of background objects: (a) conveyor line without color change; (b) examples of conveyor line with changed colors.

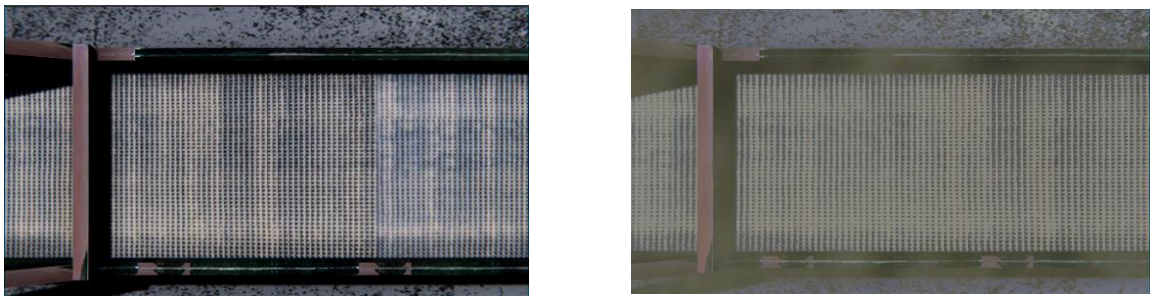
To simulate work on a conveyor line and capture images of the object from different angles, we need to adjust the camera's movement. Unity provides tools for this in its Cinemachine package. This package is designed to automate the process of creating virtual camera movements and reduce the time needed to create custom solutions. The module contains algorithms for random generation of motion maps and tracking of objects of interest. These tools make it easier to change camera angles and eliminate the need for manual manipulation (About Cinemachine, Available online).

Accordingly, as part of the work on generating synthetic data, the tool described above was used to implement a chaotic camera movement and obtain frames from different angles, as shown in **Figure 11**. To

create this effect, the Noize module was utilized, which implements a camera movement based on a depth map generated using the Perlin noise algorithm. This allows for the simulation of realistic movement and the "shaking" of a virtual camera in real time.

Figure 11. Examples of changing the camera angle.

To create a simulation of a dusty environment, the Unity Particle System module was used in the development process. The documentation for the game engine states that the particle system was designed to create realistic effects from objects that lack a clear shape and can change in real-time, such as smoke, fire, and liquids. This description clearly fits the effect of "industrial dust", Accordingly, the use of a particle system module is proposed to implement the dustiness of the room (Particle system, Available online). **Figure 12** shows a dusty room with a conveyor belt.



(a) (b)

Figure 12. A room filled with industrial dust: (a) a room without dust; (b) a room filled with dust.

The developed system will allow to generate various scenarios in production and repeatedly expand the final sample of images. This will help to avoid the problem of having to retrain the model every time we want to make changes.

4. Testing and Results

To test the data generation method, two neural networks with the YOLOv8 architecture were trained, which is due to its high accuracy in the object detection task (Sohan et al., 2024). The presence of two models is associated with the need to compare a neural network trained exclusively on synthetic data with a variant trained only on real data. In this regard, two corresponding datasets have been created. The real dataset was created based on video recordings from production, a fragment of which is shown in **Figure 13**, it consists of 371 images that were annotated using the Roboflow tool. Further, augmentation methods were applied to this set of images to expand the dataset. The final size of the dataset was 815 images.

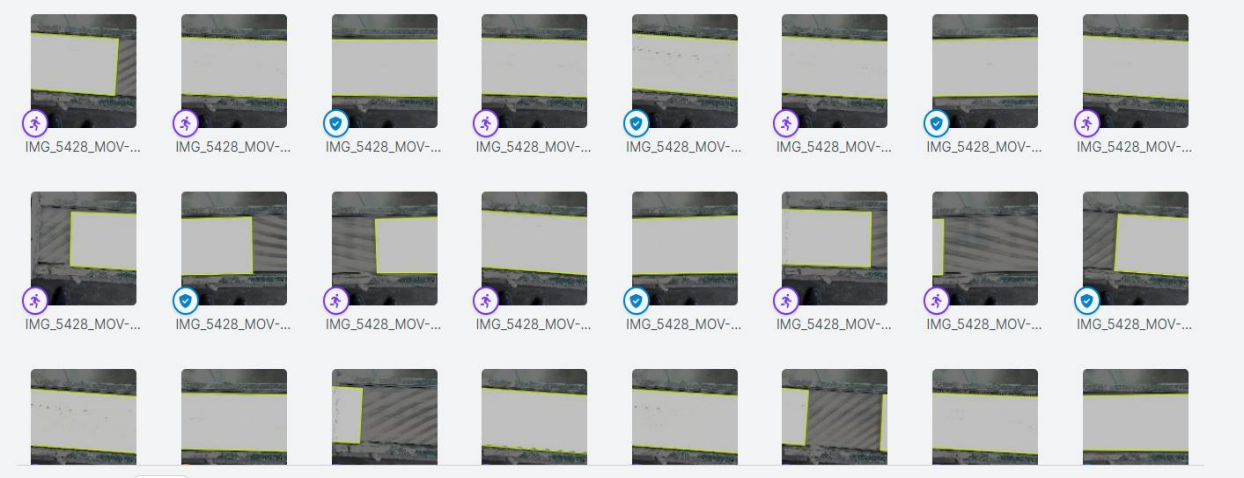


Figure 13. A fragment of a dataset of real data.

The second data set is artificially generated and contains the same number of synthetic images that were automatically annotated using the Unity Perception tool (**Figure 14**).

Figure 14. A fragment of a dataset of synthetic data.

After the data sets were prepared, neural network models were trained in the same environment using Google Colab. The models were trained for 100 epochs with a batch size of 16.

To evaluate the performance of the models, we used the mean Average Precision (*mAP*) metric at a threshold of 0.5 (*mAP*50). This metric is composed of several components, including *precision* (*p*) and *recall* (*r*), as well as the degree of intersection over union (*IoU*), and average precision (*AP*).

Precision is the proportion of correctly identified objects among all objects that the model has detected. In other words, it is a measure of how many of the model's pre-dictions were correct.

$$precision = \frac{TrueDetect}{AllDetect} \tag{4}$$

where, *TrueDetect* is the number of true detections; *AllDetect* is the number of all detections.

Recall is the proportion of correctly detected objects among all objects that are actually present in the image:

$$recall = \frac{TrueDetect}{AllObjects} \tag{5}$$

where, *TrueDetect* is the number of true detections; *AllObjects* is the number of all marked objects.

IoU shows how well the predicted bounding box overlaps with the real one:

$$IoU = \frac{S(A \cap B)}{S(A \cup B)} \tag{6}$$

where, *A* is the area of the predicted bounding box; *B* is the area of the true bounding box; *S* is the area of intersection or union of the bounding boxes.

The value of 50 in the *mAP*50 metric indicates that the predictions of the model are considered accurate if the intersection over union (*IoU*) between the predicted and actual bounding boxes is greater than 50%. In other words, for a prediction to be deemed successful, it must overlap with the actual object by at least 50%.

Next, a precision-recall curve (PR curve) is plotted for each class of object, which shows how accuracy changes depending on completeness. After that, the average precision (*AP*) value for each class is calculated by taking the area under the PR curve:

$$AP = \int_0^1 p(r)\,dr \tag{6}$$

As a result, after obtaining the *AP* scores for all classes, a final *mAP* score is calculated. This is the average AP score for all object classes:

$$mAP = \frac{1}{k}\sum_i^k AP_i \tag{7}$$

where *k* is the number of classes.

As a result, if the model's *mAP*50 is 0.75, it means that the average detection ac-curacy for all object classes (at an *IoU* of > 0.5) is 75%. This is considered a good result. of changes in the *mAP*50 metric are presented for models trained on synthetic (**Figure 15**) and real (**Figure 16**) data.

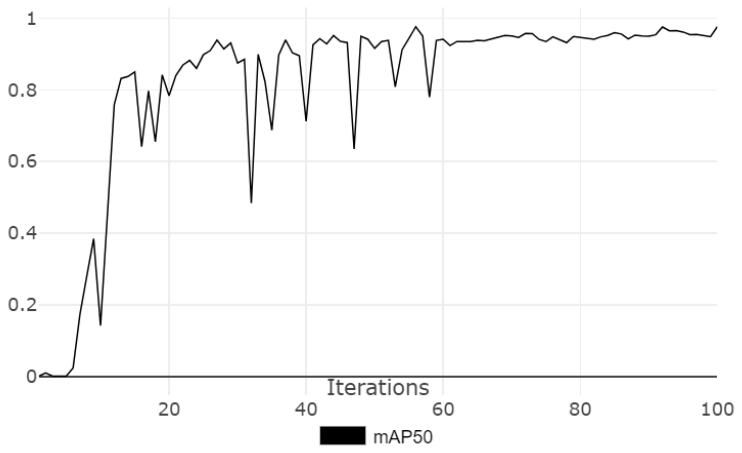


Figure 15. The graph of the average accuracy of classification and detection of the object of interest for a model trained on synthetic data.

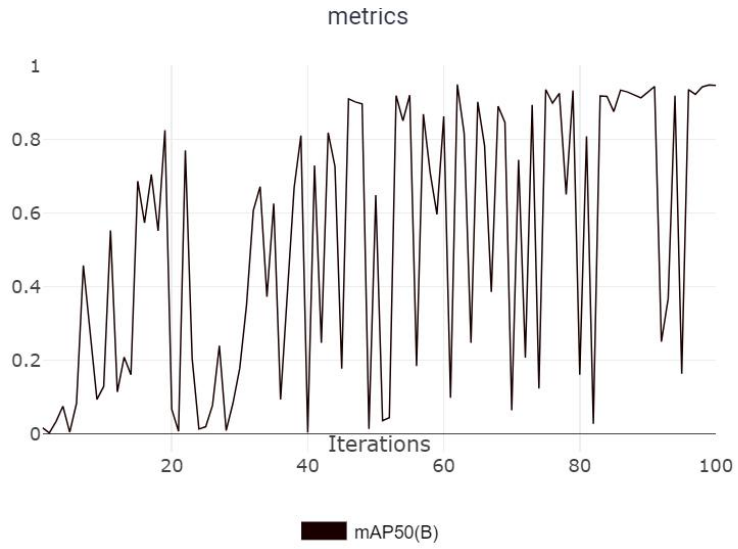


Figure 16. The graph of the average accuracy of the classification and detection of the object of interest for a model trained on real data.

From **Figures 15** and **16**, we can see that the model trained on real data made more mistakes in its predictions and achieved an *mAP*50 of 0.93 at the end of training. This is good, but it doesn't guarantee adequacy, as the uneven training schedule suggests a low ability to generalize. In contrast, the model trained using synthetic data has a smoother metric graph during training, indicating a better generalization ability. The final *mAP*50 value for the model trained on synthetic data was 0.94.

After training the models, they were tested on a video clip from the enterprise, which served as the basis for development. However, the video was not used to create a dataset of real data, as this would have distorted the test results.

As a result of the testing, a model trained on real data achieved a *mAP*50 score of 0.36 on a test dataset from the enterprise. From this, it can be concluded that with the same hyper parameters and image segmentation methods, a model based on a set of real data is not adequate. This was evident from the graph of the *mAP*50 metric during training, which showed that the model had difficulty generalizing from the presented images. This suggests that there was a lack of variety in the collected data, leading to monotony.

While the variety of synthetic data has made it possible to avoid this problem and the model has demonstrated success in the test video sequence, the *mAP*50 metric value is 0.95, indicating the adequacy of the resulting neural network.

As a result of testing the models, it was revealed that the model trained on synthetic data showed high accuracy rates in contrast to the model trained on real data, from which it can be concluded that the data generator for solving production tasks is an effective tool for creating short-circuit systems.

4. Discussion

Based on the testing data obtained from the two models, we can draw conclusions about the performance of the synthetic data generator compared to similar solutions.

First, the proposed method for generating synthetic data stands out by providing a generalized list of tasks, significantly speeding up the development process for image generation software.

Second, compared to the methodology described in Kiefer et al. (2022), our solution offers greater sampling variability thanks to the capabilities of the game engine. This allows us to recreate any desired scenarios, increasing the accuracy of models by generating images for specific tasks. In contrast to solutions that use game engines with limited capabilities like GTA V, which do not allow changing the in-game code or adding additional scenarios, our generator offers more flexible settings for modeling.

The third key aspect of the proposed system is the speed at which images are generated. The system is able to generate approximately 1,500 annotated images per minute, significantly exceeding the rendering speeds of 3D modeling tools like Blender. Achieving 1 frame per second in 3D modeling is considered a good result, but the proposed solution outperforms this in terms of both speed and flexibility. In addition to the above, compared to solutions based on the Unity game engine, the proposed system demonstrates significantly better performance. Considering the speed of image generation as a single metric - the number of images generated per second - the current generator produces 25 images per second, while solutions from (Pchelintsev et al., 2022; Reutov et al., 2022) produce 6 and 4.6 images per second respectively. Therefore, the performance of the proposed system is at least four times better than similar solutions.

Furthermore, it should be noted that generating synthetic data significantly speeds up the development of computer vision systems in all cases considered. This allows creating models with high accuracy using only synthetic data and further improving them by combining synthetic and real data.

At the next stage, it is planned to focus on training models that are not aimed at detecting the board as an object, but rather at identifying and classifying defects on its surface. This is because during the development process, there was a challenge of insufficient availability of test data necessary for the functioning of the computer vision system.

To address this issue, it will be necessary to develop and adapt computer vision models that work with labeled images to identify and classify defects on boards. Additionally, it will be essential to collect real-

world data on the objects we are studying - board defects - which is one of the primary goals of the next stages of our work. Without this real data, it is impossible to reliably assess the effectiveness of the artificially generated data and test our system for accuracy and quality.

Additionally, research is planned on the generation of artificial data for bulk materials and minerals, using coal products as an example. Currently, a test algorithm has been developed for generating coal rocks of various shapes, which ensures the creation of a diverse sample of rocks with unique shape and size characteristics**. Figure 17** shows an example of testing the algorithm, demonstrating the process of coal rock generation and real-time annotation.

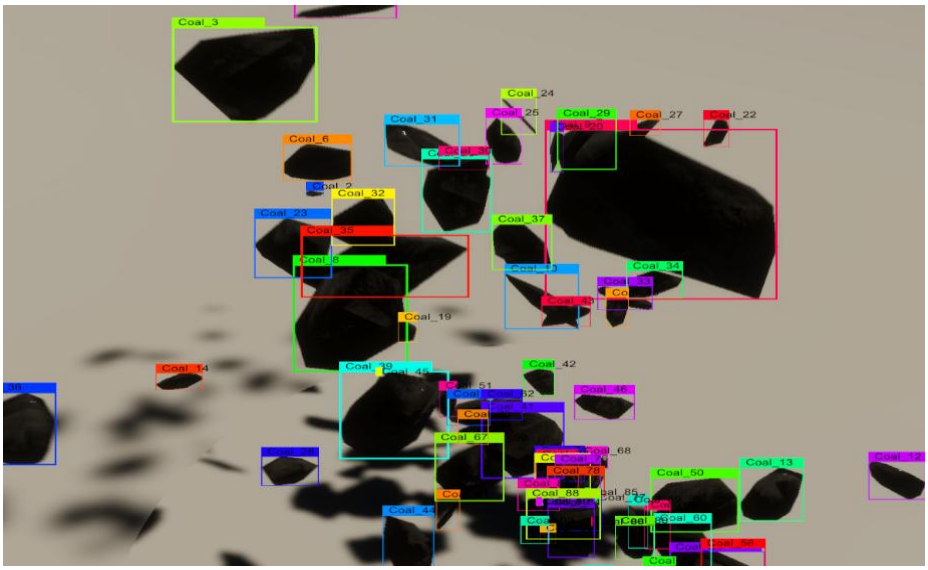


Figure 17. Example of test generation for coal rock.

This approach allows to generate up to 1,000 unique rock shapes. To further improve the process, it is necessary to refine the stone formation algorithm to ensure a wider variety of samples. It is also need to develop an algorithm to combine all coal objects into one set in order to solve the problem of measuring the height of bulk materials on a conveyor belt. Additionally, it is planned to create additional lighting settings for the scene to increase the realism of the generated images.

As part of the planned research into the field of synthetic data generation, it is planned to conduct a series of experiments with the aim of comprehensively analyzing and optimizing the use of such data for machine learning tasks. Specifically, one of the key goals is to determine the optimal proportion of synthetic and real-world data in a training sample. This will reveal the impact that varying the proportion of synthetic data has on the quality of trained models and their ability to accurately analyze real-world samples. It is expected that the results will make it possible to formulate recommendations on how to integrate synthetic data most effectively to improve the accuracy and reliability of algorithms.

In addition to data composition studies, special attention will be paid to the performance analysis of various neural network architectures. It is planned to evaluate the effectiveness of training models such as YOLO, R-CNN, Fast R-CNN, Faster-RCNN, and others in order to identify their strengths and weaknesses when working with synthetic data.

Special emphasis will be placed on studying the features of different versions of the YOLO architecture. During the experiments, learning parameters, adaptation to synthetic data, and evaluation of results on mixed datasets.

5. Conclusion

An overview of the different areas in which CV applications can be used was conducted. The analysis showed that traditional methods of data collection, such as video recording and manual image marking, are often costly and inefficient. Additionally, these methods are subject to human error, which can decrease the accuracy of neural network models. Synthetic data, on the other hand, is an alternative that can significantly reduce the time and financial cost of creating training samples.

To address the issue of image collection, we explored modern approaches to generating synthetic data for automatic object detection and classification tasks on production lines using CV technologies. The primary goal of this review was to develop an efficient and cost-effective solution for creating training datasets when real data is limited or insufficient. An analysis of existing solutions has revealed that the use of synthetic data is becoming a more significant and in-demand tool in various industries. Various tools and technologies for generating such data have been considered, including game engines, 3D modeling tools, and specialized packages for automatic image annotation. Each of these solutions has its own advantages and disadvantages. For example, the use of game engines allows you to generate real-time data and process large volumes of images at high speed, which significantly speeds up the process of creating datasets. At the same time, the use of 3D modeling provides a higher level of realism, but requires significant computing resources and rendering time.

The generalized model of the synthetic data generator, proposed in the paper, was applied to the implementation of a technological process for quality control of floor coverings on a conveyor line. As part of this process, a system was developed to simulate a real production environment, considering all its features, such as changing camera angles and applying various defects to objects of interest. Additionally, effects such as industrial dust were added to create diverse and realistic images. These images can then be used to train neural network models for quality control purposes.

After the implementation of the data generator, two YOLOv8 neural network models were compared. Testing carried out as part of this work has shown that the model trained on synthetic data demonstrated superior accuracy compared to the model trained on real data, which was limited in volume and variety. Using synthetic data avoids the overfitting, providing higher generalization ability for models. This was confirmed by testing results, where the model trained on synthetic data achieved an *mAP*50 value of 0.95, a high indicator for industrial object detection.

The conducted research has demonstrated that the developed tool has a number of significant advantages, allowing it to efficiently generate data whose characteristics are comparable to those of real data. This opens up prospects for its application in various computer vision tasks.

Particular attention should be paid to the fact that the use of synthetic data significantly speeds up the process of model development. In all the analyzed cases, it was confirmed that the use of this generator can significantly reduce the time and resource costs for collecting and processing real data. Moreover, synthetic data makes it possible to create models with high accuracy even in conditions of limited access to large amounts of real information.

Additionally, it should be noted that combining real and synthetic data allows for better results. This approach helps to increase the stability and generalizing ability of models, which, in turn, provides better and more reliable forecasts in various conditions. Thus, the results of the work carried out allow us to conclude that the generation of synthetic data is a valuable tool for the development of computer vision systems, which helps to reduce development time and improve their quality.

Conflict of Interest
The authors confirm that there is no conflict of interest to declare for this publication.

AI Disclosure
The author(s) declare that no assistance is taken from generative AI to write this article.

Acknowledgments
This research was funded by Ministry of Science and Higher Education of the Russian Federation; project FEWM-2023-0013.

References

Arakeri, M.P. (2016). Computer vision based fruit grading system for quality evaluation of tomato in agriculture industry. *Procedia Computer Science*, *79*, 426-433. https://doi.org/10.1016/j.procs.2016.03.055.

Aydin, I., & Othman, N.A. (2017). A new IoT combined face detection of people by using computer vision for security application. In *2017 International Artificial Intelligence and Data Processing Symposium* (pp. 1-6). IEEE. Malatya, Turkey. https://doi.org/10.1109/IDAP.2017.8090171.

Bazuhair, W., & Lee, W. (2020). Detecting malign encrypted network traffic using perlin noise and convolutional neural network. In *2020 10th Annual Computing and Communication Workshop and Conference* (pp. 0200-0206). IEEE. Las Vegas, NV, USA.

Blackledge, J., & Dubovitskiy, D.A. (2011). Quality control system using texture analysis in metallurgy. *Third International Conferences on Pervasive Patterns and Applications* (pp. 122-127). Rome. https://doi.org/10.21427/D7GZ4Q.

Costa, C., Antonucci, F., Pallottino, F., Aguzzi, J., Sun, D.W., & Menesatti, P. (2011). Shape analysis of agricultural products: a review of recent research advances and potential application to computer vision. *Food and Bioprocess Technology*, *4*(5), 1192-1215. https://doi.org/10.1007/s11947-011-0556-0.

Dhanya, V.G., Subeesh, A., Kushwaha, N.L., Vishwakarma, D.K., Kumar, T.N., Ritika, G., & Singh, A.N. (2022). Deep learning based computer vision approaches for smart agricultural applications. *Artificial Intelligence in Agriculture*, *6*, 211-229. https://doi.org/10.1016/j.aiia.2022.09.007.

Dustler, M., Bakic, P., Petersson, H., Timberg, P., Tingberg, A., & Zackrisson, S. (2015). Application of the fractal Perlin noise algorithm for the generation of simulated breast tissue. In *Medical Imaging 2015: Physics of Medical Imaging* (Vol. 9412, pp. 844-852). SPIE Medical Imaging, Orlando, Florida, United States.

Esteva, A., Chou, K., Yeung, S., Naik, N., Madani, A., Mottaghi, A., Liu, Y., Topol, E., Dean, J., & Socher, R. (2021). Deep learning-enabled medical computer vision. *NPJ Digital Medicine*, *4*(1), 5. https://doi.org/10.1038/s41746-020-00376-2.

García, C.G., Meana-Llorián, D., G-Bustelo, B.C.P., Lovelle, J.M.C., & Garcia-Fernandez, N. (2017). Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in smart cities, smart towns, and smart homes. *Future Generation Computer Systems*, *76*, 303-313. https://doi.org/10.1016/j.future.2016.12.033.

Greeshma, C.A., Nidhindas, K.R., & Sreejith, P. (2019). Traffic control using computer vision. *International Journal of Advanced Research in Computer and Communication Engineering*, *8*(4), 39-47.

Gupta, S., & Garima, E. (2014). Road accident prevention system using driver's drowsiness detection by combining eye closure and yawning. *International Journal of Research*, *1*(6), 839-842.

Harikrishna, K., Davidson, M.J., & Reddy, G.D. (2023). New method for microstructure segmentation and automatic grain size determination using computer vision technology during the hot deformation of an Al-Zn-Mg powder metallurgy alloy. *Journal of Materials Engineering and Performance*, *34*, 121-131. https://doi.org/10.1007/s11665-023-09025-7.

Kiefer, B., Ott, D., & Zell, A. (2022). Leveraging synthetic data in object detection on unmanned aerial vehicles. In *2022 26th International Conference on Pattern Recognition* (pp. 3564-3571). IEEE. Montreal, QC, Canada. https://doi.org/10.1109/ICPR56361.2022.9956710.

Lan, R., Awolusi, I., & Cai, J. (2024). Computer vision for safety management in the steel industry. *AI*, *5*(3), 1192-1215. https://doi.org/10.3390/JMMP7010017.

Li, H., Tuo, X., Liu, Y., & Jiang, X. (2015). A parallel algorithm using Perlin noise superposition method for terrain generation based on CUDA architecture. In *International Conference on Materials Engineering and Information Technology Applications* (pp. 967-974). Atlantis Press. https://doi.org/10.1007/s11947-011-0556-0.

Liu, G., Shi, H., Kiani, A., Khreishah, A., Lee, J., Ansari, N. (2021). Smart traffic monitoring system using computer vision and edge computing. *IEEE Transactions on Intelligent Transportation Systems*, *23*(8), 12027-12038. https://doi.org/10.1109/TITS.2021.3109481.

Pchelintsev, S., Yulyashkov, M.A., & Kovaleva, O.A. (2022). A method for creating synthetic datasets for training neural network models to recognize objects. *Information Management Systems*, *3*, 9-19. https://doi.org/10.31799/1684-8853-2022-3-9-19.

Reutov, I., Moskvin, D., Voronova, A., & Venediktov, M. (2022). Generating synthetic data to solve industrial control problems by modeling a belt conveyor. *Procedia Computer Science*, *212*, 264-274. https://doi.org/10.1016/j.procs.2022.11.010.

Rusanovsky, M., Beeri, O., & Oren, G. (2022). An end-to-end computer vision methodology for quantitative metallography. *Scientific Reports*, *12*(1), 4776. https://doi.org/10.1038/s41598-022-08651-w.

Sarrionandia, X., Nieves, J., Bravo, B., Pastor-López, I., & Bringas, P.G. (2023). An objective metallographic analysis approach based on advanced image processing techniques. *Journal of Manufacturing and Materials Processing*, *7*(1), 17. https://doi.org/10.3390/JMMP7010017.

Serrano, Á., Conde, C., Rodríguez-Aragón, L.J., Montes, R., & Cabello, E. (2005). Computer vision application: real time smart traffic light. In *Computer Aided Systems Theory–EUROCAST 2005: 10th International Conference on Computer Aided Systems Theory* (pp. 525-530). Springer Berlin Heidelberg, Las Palmas de Gran Canaria, Spain.

Sohan, M., Sai Ram, T., Reddy, R., & Venkata, C. (2024). A review on yolov8 and its advancements. In *International Conference on Data Intelligence and Cognitive Informatics* (pp. 529-545). Springer, Singapore. https://doi.org/10.1007/978-981-99-7962-2_39.

Villalba-Diez, J., Schmidt, D., Gevers, R., Ordieres-Meré, J., Buchwitz, M., & Wellbrock, W. (2019). Deep learning for industrial computer vision quality control in the printing industry 4.0. *Sensors*, *19*(18), 3987. https://doi.org/10.3390/s19183987.

Ying, X. (2019). An overview of overfitting and its solutions. In *Journal of Physics: Conference Series* (Vol. 1168, p. 022022). IOP Publishing. https://doi.org/10.1088/1742-6596/1168/2/02202.

A synthetic data generator. Training of neural networks for industrial flaw detection. Available online: https://medium.com/phygitalism/synthetic-data-generator-a052d347468 (accessed on May 5, 2024).

About Cinemachine. Available online: https://docs.unity3d.com/Packages/com.unity.cinemachine@2.8/manual/index.html (accessed on May 26, 2024).

Particle system. Available online: https://docs.unity3d.com/ru/530/Manual/ParticleSystems.html (accessed on May 26, 2024).

The perception camera component. Available online: https://docs.unity3d.com/Packages/com.unity.perception@1.0/manual/PerceptionCamera.html (accessed on May 20, 2024).