Idle Identification of Construction Machinery through a Deep Learning-Based Algorithm Embedded in Surveillance Camera Systems

XANDER KÜPERS, University of Twente, The Netherlands

This study proposes a lightweight deep learning-based algorithm for idle identification of construction machinery, which can be embedded in surveillance camera systems. The construction industry faces severe challenges. Efficient utilization of construction machinery is crucial. Monitoring the utilization rate of construction machinery can identify inefficiencies and idle times, allowing for optimization of equipment use. The proposed algorithm consists of an object detection model, tracking algorithm, and idle state identification method. It is designed to run on a CPU and on the edge. The embedding in the existing surveillance infrastructure has several advantages, such as leveraging existing hardware to reduce costs and minimizing bandwidth usage and latency by enabling edge deployment. Performance findings indicate that the algorithm can effectively monitor idle states of construction machinery, achieving high accuracy and a high harmonic mean of precision and recall overall.

Additional Key Words and Phrases: construction site - edge computing - object detection - idle identification - vision based - construction machinery

1 INTRODUCTION

In the Netherlands, the construction industry faces several severe challenges, including a nitrogen crisis, a housing shortage, and high housing prices. In 2021, there was a housing shortage of 279,000 houses, and an additional 900,000 houses need to be built by 2030 [20]. An effective construction industry is key to addressing these challenges. A critical part of every construction project is the efficient utilization of construction machinery, such as excavators, bulldozers, cement mixer trucks, and dump trucks. This machinery is one of the constructor's biggest expenditures [11].

Monitoring the utilization rate of construction machinery can identify inefficiencies and idle times, allowing for optimization of equipment use. This leads to improved productivity, which can reduce fuel consumption and CO2 emissions, thereby saving costs. A study referenced in [25] shows that increasing the average operational efficiency of a dump truck from 40% to 50% by reducing idle time by 6 minutes per hour can reduce hourly fuel consumption and CO2 emissions by 10%. Manual monitoring is costly, tedious and prone to errors [3]. There are however monitoring systems that are used on construction sites that rely on Artificial Intelligence (AI), which include telematics, IoT sensors, drones, RFID, and augmented reality systems [18]. There are also case studies that use video for monitoring the construction environment, but many of these systems require high computing power and have limited capabilities to be deployed on the edge [3, 6, 12] High computing power requirements can necessitate the use of expensive hardware and may also lead to increased energy consumption. Also, without edge deployment, there might be latency issues and bandwidth constraints as there is a need to send data to a server and receive responses [22].

Many construction sites are monitored by mobile surveillance camera systems to secure construction sites against theft, vandalism, and unauthorized access. However, these surveillance cameras could be useful for wider algorithms through embedding an AI algorithm in the mobile surveillance system. This embedding in the existing surveillance infrastructure has several advantages, such as leveraging existing hardware to reduce costs and minimizing bandwidth usage and latency by enabling edge deployment.

This study aims to develop a lightweight AI algorithm integrated into mobile surveillance camera systems to monitor the idle status of construction machinery to understand their utilization rates. It will focus on supervised deep learning, tracking and logistic regression. With this study, there is a potential for project-managers and contractors to gain a deeper insight into the utilization rate of construction machinery and can help in completing construction projects under tight schedule and budget restrictions.

This study is structured as follows. First, a literature review explores the existing literature on computer object detection, tracking algorithms, and idle identification methods. Second, the design and implementation choices behind the proposed AI algorithm will be explained. Third, a performance evaluation of the proposed algorithm will be presented. Fourth, the conclusion will summarise the findings, discuss the implications for practice, and highlight the current limitations. Finally, potential future work will be outlined.

2 LITERATURE REVIEW

This literature review explores the existing literature of computer object detection, tracking algorithms, and idle identification methods. This review provides an understanding of how this study builds upon previous research and extends the field of construction site monitoring. It also offers a foundational overview for the rest of this study.

2.1 Computer Object Detection

Computer object detection involves detecting and localizing objects that appear in images. The traditional approach for object detection includes rule-based detectors and human-tuned feature descriptors [15, 23]. Both studies address similar challenges that need to be overcome to extend their applicability and enhance their effectiveness in real-world settings. These directions include improved feature extraction techniques, real-time processing capabilities, and better handling of environmental variability.

Some of these challenges can be overcome with the help of deep learning techniques. In recent years, there has been an increased number of algorithms on construction sites that involve deep learning. Many of these techniques involve Convolutional Neural Networks (CNNs). In a study by Nath and Bezhdan [17], YOLOV2 and

TScIT 41, July 5, 2024, Enschede, The Netherlands

YOLOV3-based CNN models were employed for the detection of construction objects. In another study by Kim et al. [12], a Faster R-CNN-based model was used. Both types of CNN models are real-time detection architectures. A comparison of YOLOV3 and Faster R-CNN performances was conducted in a study on deep learning [27]. The study concluded that both algorithms have their trade-offs: compared to YOLOV3, Faster R-CNN's average iterations of frame images per second are slower. However, the mean average precision over all classes is higher.

MobileNets is another CNN that is deployed in other industries but, to our knowledge, not yet in construction domain. MobileNets is a family of algorithms focused on mobile and embedded vision algorithms, designed for resource-constrained devices [7]. However, MobileNets has the same Frames Per Second (FPS) on a single-board computer compared to YOLOV5, while YOLOV5 has higher accuracy [10].

As discussed in the literature, YOLO is a popular choice for real-time object detection inference. The latest version, YOLOv8, was released in May 2023. Although there are limited scientific papers directly from the creators, several case studies and articles highlight YOLOv8's performance. For instance, a study reported improvements in detection accuracy and processing speed when comparing YOLOv8 to previous versions such as YOLOv3 and YOLOv5 [14]. These findings suggest that YOLOv8 could be a promising candidate for this study, given its demonstrations and results in recent case studies [4, 26].

2.2 Tracking Algorithms

Tracking in AI involves monitoring a detected object that potentially moves. When the object moves, the tracking algorithm follows the object in the correct direction. As discussed in the study of Kumar and Rawal [13], there are three techniques for tracking: point based, kernel based and silhouette based. Point based tracking is used for tracking objects that are well-defined and have relatively easy distinguishable features. Kernel based tracking involves using a shape to represent the object and see in iterative frames if there is movement of the object. This is often used if objects do not have distinct points but can be distinguished by their overall representation. Silhouette based tracking is used for tracking objects with variable shapes and sizes, where the outline can dynamically change.

Construction machinery can be seen as well-defined objects with distinguishable features. Therefore, the focus will be on point-based tracking algorithms. Simple Online and Realtime Tracking (SORT) is a well-known easy-to-compute tracking algorithm that can track multiple objects in a frame. It works by combining concepts of the Kalman filters and Hungarian algorithm to track objects [5]. In a case study about excavators, the SORT algorithm was utilized to identify distinct excavators and ensure that their unique IDs were maintained and tracked even if they moved. This contributed to the accurate generation of statistics, preventing any confusion regarding which excavator the statistics referred to [6].

After the creation of the SORT algorithm an enhanced algorithm was created named called Deep SORT. It is the SORT algorithm with a deep association metric. This makes it possible to track objects through longer periods of occlusions, while still keeping a high performance for real-time usage. Another advantage of Deep SORT compared to SORT is that Deep SORT performs well even if objects that are tracked do collide or overlap [28]. A downside of Deep SORT is that it needs more computing power and decreases the speed of the algorithm significantly [16].

The last explored tracking algorithm is the ByteTrack tracking algorithm. While not being used in construction industry until now, ByteTrack has high potential in other fields. In a study about vehicle tracking in highway videos, ByteTrack outperforms SORT and DeepSORT significantly. It achieved the highest multiple object tracking accuracy, highest multiple object tracking precision and had the fewest ID switches. It also had the lowest false positives and false negatives, with the fastest processing speed at 171 FPS [1].

2.3 Idle Identification Methods

Idle identification on the construction site has extensively been researched for productivity analysis of excavators. One proposed approach involves identifying the idle state using data of the bounding box [6]. After the centroid, height and width of the bounding box is computed, a sliding window mechanism stores the data and performs intensive computational tasks to compute statistics about the bounding box. These statistics are used to predict if the status of the object.

Another method for identifying idle states is via the frame difference method [9]. This method of idle identification works via binarization of the pixels in the frame. If there is a certain amount of white pixels that is beyond a certain threshold, motion is detected. The threshold is adaptive in this study, meaning that it starts at 15% and changes if needed to prevent threshold instability.

The last explored method for idle identification is a non-vision based approach, but via an on-board system [31]. Via on-board sensors, data can be gathered on vehicle speed, location changes, fuel consumption rate, and power-take-off status. This data can determine idle states. For instance, if the vehicle is stationary with the ignition on for at least 5 minutes, the vehicle is idle. If during these 5 minutes the steering wheel is turned or the location changes, the timer resets and starts over.

While the second and third idle identification methods are promising in their use cases, they have challenges regarding implementation within the context of this study. The second method works via determining the total difference in pixels, however if there is movement in front of the machine from for example another machine or a worker, the algorithm will detect movement, even if the movement is not from the machine in frame. The third method will not be utilized in this study, as the focus is on a vision-only approach for idle identification. However, the heuristics about when a vehicle is idle or not can still be applied. The first explored idle identification method is promising, as it is already used in the construction domain and tested on excavators. This method can be extended to make it also work for other types of machinery and to make it less computational intensive.

3 DESIGN AND IMPLEMENTATION

This section outlines the design and implementation choices behind the AI algorithm for detecting, classifying, tracking, and identifying idle states of construction machinery. The steps of the AI algorithm are depicted in Figure 1. The methodology used for designing and implementing the proposed AI algorithm is based on the machine learning workflow, which consists of nine stages that can be encapsulated into four main stages: data collection, data preprocessing, model development, and model evaluation [2].

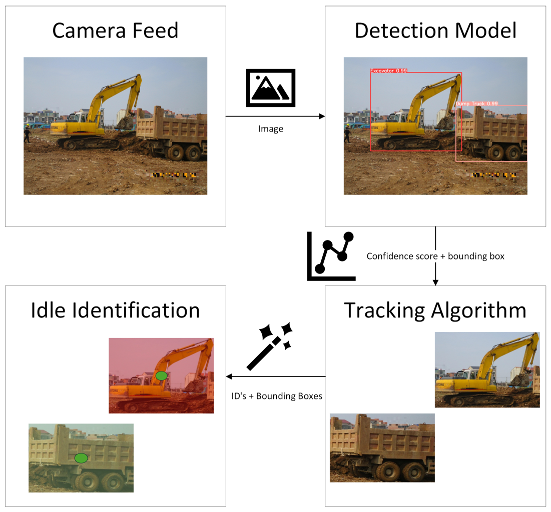


Fig. 1. Simplified overview of proposed algorithm for this study

3.1 Data Collection

In this study, three stages require data. However, the data for the second and third stages depends on the data from the first stage. The first stage, which involves the detection model, requires an image dataset for training purposes. Several public image datasets for machine learning in construction are available. These machine learning datasets do not only consists construction machinery, but also involves images and labels about construction buildings, safety equipment and construction material. Among these datasets, two prominent and extensive ones are the *Alberta Construction Image Dataset* (ACID) [30] and *Moving Objects in Construction Sites* (MOCS) image dataset [29].

Both datasets contain images captured from various poses, viewpoints, illumination conditions, weather conditions, and levels of occlusion. This ensures that the models trained on these datasets are robust and capable of performing effective under various real-world circumstances. The *ACID* dataset consists of three types of machinery: excavator, dump truck, and concrete mixer truck. The *MOCS* dataset consists of thirteen categories, six of which are types of machinery: roller, bulldozer, excavator, truck, loader, and concrete mixer truck. Only three of these six machinery categories will be used to keep it consistent with the *ACID* dataset. This means only the excavator, dump truck and concrete mixer truck will be used.

For this study, a subset of the two mentioned datasets will be extracted and combined into a single dataset. The final dataset will include a randomized selection of images, while still ensuring representation of the different types of construction machinery and their various appearances. This combination will potentially improve the performance metrics of the trained model and decrease the false positive rate [21]. Another potential improvement from using a subset of both datasets is enhanced generalization, due to the different geographic locations of the images and the varied appearances of construction machinery.

3.2 Data Preprocessing

The *MOCS* and *ACID* dataset are already labeled. However, preprocessing is needed before the detection model can be trained. The *MOCS* dataset contains thirteen categories, of which three are used for this study. Therefore, preprocessing is necessary to exclude images and labels that do not contain information about excavators, dump trucks, or concrete mixer trucks. Additionally, the *MOCS* dataset annotation file is formatted in the Common Objects in Context (COCO) format. This COCO JSON file includes info, licenses, images, annotations, categories, and segment info about the dataset. However, the YOLOv8 annotation format only requires the class ID, the coordinates of the bounding box center, and the width and height of the bounding box. Through a conversion script, the required annotations can be extracted from the COCO format and reformatted into the YOLOv8 format, ensuring that only the required categories are included.

An other preprocessing step is image resizing. All the images in the dataset have different dimensions. The YOLOv8 needs images that have a width and height of 640 pixels. Nearest neighbor interpolation is used to resize, as it can achieve this in a relative fast and computationally efficient way.

At this stage of the data preprocessing, the dataset contains labels and images of excavators, dump trucks, and concrete mixer trucks, all resized to the 640x640 dimensions required by the YOLOV8 model. Before splitting the data, it is needed to validate the bounding boxes and labeled vehicles. This validation is performed using the Computer Vision Annotation Tool (CVAI). By randomly reviewing labels of the images in the CVAI software, efforts are made to ensure the validation of the labels and depicted machinery.

The data set is randomly split into a training and test set according to a 6:4 ratio. A larger training set compared to the test set prevents overfitting [32]. Also, for every class in the data set, the same amount of images is used (1508 for train set and 1005 for test set). This is to prevent class inbalance.

3.3 YOLOV8 Detection Model

As discussed in the Section 2.1, YOLOV8 is a promising candidate to be the detection model of this study. As a detection model, the architecture of YOLOV8 is capable of classifying and localizing specific objects on which it has been trained, such as construction machinery. YOLO is a single-stage detection model. The single-stage detection model makes sure that data gets through the neural network in a single forward pass. This is different compared to a two-stage detection model, were multiple parts of the data gets reused for further processing. This single-stage approach makes YOLO better to use for real-time processing and low computational resources, although the accuracy is lower compared to two-stage detection models [8].

In this study, YOLOv8 will not be modified or improved for the construction machinery use case. Instead, YOLOv8 will be used as a tool for real-time detection of construction machinery. In this context, real-time detection refers to object identification and localization within 150 ms on the CPU of a general-purpose computer.

The YOLOV8 architecture comprises two CNNs for automatic feature extraction, bounding box, and object classification [26]. Based on a study [19], the YOLO algorithm can be divided into the following abstracted steps:

(1) Input gets divided into an $S \times S$ grid.
(2) Every cell in the $S \times S$ grid produces two outputs:
    - The cell produces bounding boxes and an associated confidence score for every bounding box that it actually contains a class and has the correct bounding box dimensions.
    - The cell produces a class probability map, giving every cell in the $S \times S$ grid a color based on the class that has the highest probability of being contained inside the cell. If the probability is below a certain threshold, the cell can also be colored to indicate that it contains no class.
(3) The bounding boxes, associated confidence scores, and the class probability map into a final decision.

The YOLO model can be trained from scratch or using a pre-trained model. The benefit of using a pre-trained model is that the weights have already been defined based on a large dataset. This approach is comparable to transfer learning, in which the model already knows how to identify basic shapes such as edges and circles, as well as other generalized properties, before the custom model gets trained. Therefore, the YOLOv8n pre-trained model will be used. This is the smallest available pre-trained model available.

Manual feature extraction is not necessary as it is a deep learning model. However there are certain hyperparameters that can be adjusted. These hyperparameters help to control the training process of the YOLOV8 model. The used hyperparameters are shown in Table 1. If a hyperparameter retains its default value as specified in the YOLOv8 architecture, it is not shown in table.

Table 1. Custom Hyperparameters for YOLOV8 Model

Hyperparameter	**Value**
epochs	100
batch size	16
image size	640
optimizer	SGD
learning rate	0.01
momentum	0.937
weight decay	0.0005

The hyperparameters listed in Table 1 have been adjusted multiple times, with variations in epochs, batch sizes, optimizers, and other parameters to achieve the best performance observed in this study. After a trial-and-error approach of parameter tuning, which involved testing multiple models, the hyperparameters shown above yielded the best results. To improve the performance of the model, it is beneficial to benchmark the model on multiple machine learning frameworks. In Table 2, the benchmark results are shown.

Table 2. YOLOV8 benchmark in machine learning frameworks

Format	**mAP50-95**	**FPS**
PyTorch	0.5498	4.00
TorchScript	0.5510	4.78
ONNX	0.5523	15.12
OpenVINO	**0.5550**	**33.14**
NCNN	0.5510	14.78

During this benchmark, techniques such as half-precision inference and INT8 quantization were not used, as these could increase speed but can potentially decrease mean average precision. The benchmark revealed a significant winner: the OpenVINO model. OpenVINO, an Intel-created toolkit, is designed to optimize inference on Intel hardware. Since the benchmark was run on an Intel CPU, its performance is understandable. If the model is run on an ARM CPU, NCNN might deliver the best performance, as it is a neural network inference framework optimized for ARM CPUs.

3.4 ByteTrack Tracking Algorithm

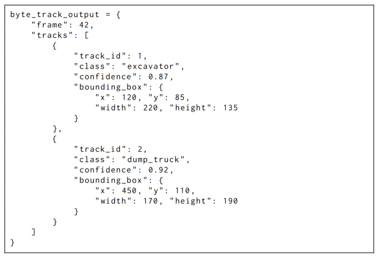
As discussed in Section 2.2, ByteTrack has the potential to outperform state-of-the-art tracking algorithms. This, in combination with its fast processing speed is the reason why this tracking algorithm is chosen as the tool to use in this study. The purpose of the tracking algorithm is to track and maintain distinct identifications of the detected machinery, even in cases of occlusion or overlapping within the frame. This aims to collect specific statistics about each object's idle time while minimizing any confusion if the focus has switched to the wrong object.

ByteTrack operates via a track management system that processes the detection output of the detection model per frame [33]. These detections include the bounding boxes, confidence scores, class labels and frame information. Detections are divided into two groups: high-confidence detections (HCDs) above a high-confidence threshold and low-confidence detections (LCDs) between the high-confidence and minimal-confidence thresholds.

For each HCD, the Hungarian algorithm is used to find the best match with existing tracks. If a match is found, the track gets updated with the new detection information. If there is no match found, a new track gets initialized with a unique track ID. For each LCD, a cost matrix calculation is performed based on the features of the LCD. The result of the cost matrix is compared with each track, and the LCD gets added to the closest matching existing track.

There are several mechanism that are included for the track maintenance and termination. A track becomes active after it has been detected in several consecutive frames. A track gets terminated after it is not updated for a predefined number of frames. However, a re-identification mechanism allows objects that reappear after occlusion to obtain their previous track ID if the features match closely enough.

For the proposed algorithm of this study, no hyperparameters are changed. This means that the default values for the thresholds, buffers, and minimal box areas are based on the default YOLO tracker settings for the ByteTrack tracker. The output does differ from the standard output. The format of the used output is shown in Figure 2.



```
byte_track_output = {
    "frame": 42,
    "tracks": [
        {
            "track_id": 1,
            "class": "excavator",
            "confidence": 0.87,
            "bounding_box": {
                "x": 120, "y": 85,
                "width": 220, "height": 135
            }
        },
        {
            "track_id": 2,
            "class": "dump_truck",
            "confidence": 0.92,
            "bounding_box": {
                "x": 450, "y": 110,
                "width": 170, "height": 190
            }
        }
    ]
}
```

Fig. 2. ByteTrack output format

3.5 Idle Identification Algorithm

The final step of the algorithm is the idle identification of construction machinery. This idle identification algorithm takes as input the bounding boxes and track IDs provided by the ByteTrack. The idle identification algorithm processes frames. This means that a hash map is provided to the idle identification algorithm per frame. Each entry in the hash map contains a Track ID and metadata. An overview of the idle algorithm is presented in the flowchart in Figure 3.

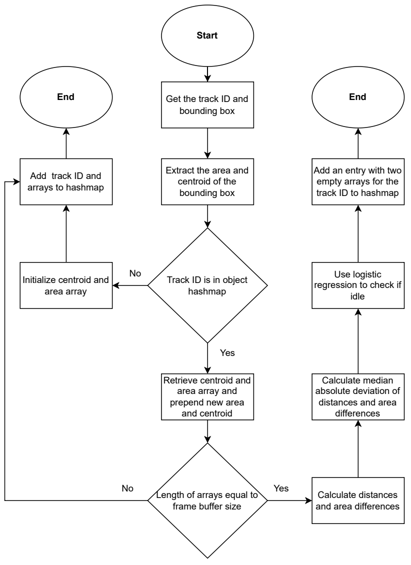

Start
Get the track ID and bounding box
Extract the area and centroid of the bounding box
Track ID is in object hashmap
No
Yes
Initialize centroid and area array
Add track ID and arrays to hashmap
End
Retrieve centroid and area array and prepend new area and centroid
Length of arrays equal to frame buffer size
No
Yes
Calculate distances and area differences
Calculate median absolute deviation of distances and area differences
Use logistic regression to check if idle
Add an entry with two empty arrays for the track ID to hashmap
End

Fig. 3. Flow chart idle identification algorithm

Each entry in the hash map represents a distinct detected object, identified by its tracking ID. The idle identification algorithm uses only one statistic that can be derived from the metadata of each hash map entry, the bounding box. The bounding box is composed of 4 variables, which include the x and y coordinates of the top-left corner, as well as the width and height of the bounding box. This data is sufficient for the idle identification algorithm to function.

When an entry passes through the idle algorithm, the area and centroid of the bounding box are calculated and stored in two separate arrays associated with the tracking ID. There is a buffer mechanism in the idle identification algorithm that monitors the number of elements in the arrays. When the buffer is full, calculations can be performed on the elements, after which the buffer clears itself and becomes empty again. The number of elements refers to the frames that can be kept in the buffer before calculations are performed. For example, if the buffer has a size of 20 and the frame rate of the video is 10, then the idle calculations are done over a period of 2 seconds. This means that the algorithm will predict whether the machine was idle or not during those 2 seconds.

The algorithm utilizes four formulas for its calculations, as defined in Table 3.

Table 3. Formulas used in the algorithm

Calculation	**Formula**
Area Difference	$AD = \lvert A_i - A_{i+1} \rvert$
Centroid Difference	$CD = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$
Median Abs. Deviation	$MAD = \frac{\sum_{i=1}^{n} \lvert D_i - \text{Median} \rvert}{n}$
Logistic Regression	$p = \frac{1}{1+e^{-(\beta_0+\beta_1 \cdot \text{MAD\_AD}+\beta_2 \cdot \text{MAD\_CD})}}$

The area difference formula is used to determine the area difference between frame i and frame i + 1. The centroid difference is the Euclidean distance formula, which is based on the difference in centroid position between frame i and frame i + 1. The median absolute deviation (MAD) is chosen as a measure of the variability of the data. The MAD is chosen over the standard deviation due to the non-normal distribution of the data. The distribution of the data was tested through a Shapiro-Wilk Test and showed a p-value of 0.007 that is significantly less than 0.05, meaning that the data is not normally distributed. MAD is a robust measure of data variability, meaning that it can still be used if the distribution is unknown. With the MAD, the variance in area difference and centroid distances can be calculated. The last formula is the logistic regression formula. Logistic regression is a supervised learning method for binary classification. It determines whether a machine is idle using a probability ranging from 0 to 1. If $p \geq 0.5$, it means that the machine is not idle and if $p < 0.5$, it means that the machine is idle. The further p is from 0.5, the more confident the algorithm is in its prediction.

A logistic regression model is trained for determining if the machine is idle or not. The model is trained with 200 gathered data points. These are trained with a buffer size of 15 with a FPS of 10, meaning that each classification of being idle covers a duration of 1.5 seconds. The final logistic regression model outputs two coefficients based on the predictors and one intercept. The following coefficients and intercept have been computed by the model: $\beta_0 = 2.4613463131$, $\beta_1 = -0.00136793$, $\beta_2 = -0.36581202$.

The frame buffer size and desired FPS are two parameters that can be adjusted to customize the algorithm according to the specific goal. A larger frame buffer size means that the idle status prediction is based on a longer video recording. If the frame buffer size is too large, small or rapid changes of the machine may not be detected. If the frame buffer size is too small, the algorithm may be overly sensitive. If the desired FPS is too high, too much computational resources are needed, meaning that it is unable to predict the idle state. If the desired FPS is low, it means that fewer frames are processed per second. Meaning that the computational load gets reduced, but it also lead to a less responsive video.

4 PERFORMANCE EVALUATION

The proposed AI application in this study can be integrated within the existing surveillance camera infrastructure. Several experiments and benchmarks are conducted to obtain results that will be shown in the following subsections. These experiments and benchmarks are performed on hardware similar to existing surveillance camera infrastructures. The computer hardware configuration typically includes an Intel i3 CPU, 8 or 16 GB of RAM, a SSD, and no dedicated GPU. Their computer software environment usually consist of a Docker environment running on Ubuntu or the Windows operating system. In this section, several metrics will be used to examine the performance.

4.1 Detection Model Evaluation

The performance of the detection model during the 100 training epochs is shown in Figure 4. Table 4 presents the model's performance including the final precision, recall, F1 score, mAP50 and mAP50-95.

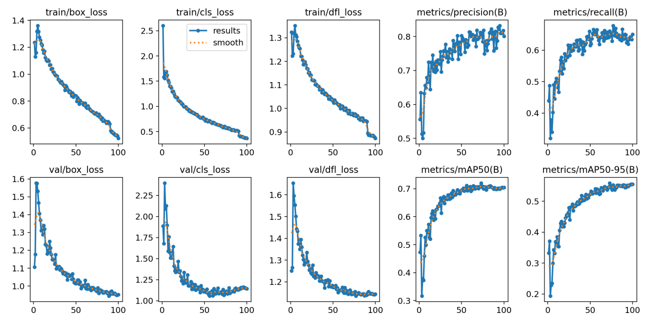


Fig. 4. Detection model performance improvement during training

In Table 4, three different losses for both the train and test data can be analyzed: the box loss, class loss and distribution focal loss. The losses for the test data set is much higher compared to the train data set. This may have to do with wrongly labeled objects in the test set. Certain images lack object annotations, causing the model to be penalized for detecting real objects that are not labeled and thus seen as noise/background.

The overall performance of the detection model is promising, as the F1 score shows a balance between precision and recall, meaning that it correctly detects objects while minimizing false positives. The result of 70.44% mean Average Precision at 50% intersection over union (mAP50) means that it can predict bounding boxes with at least 50% overlap with the ground truth box in most situations. However, the lower percentage for the mAP50-95 indicates that there are still some fluctuations in the bounding boxes. This could mean that if a machine is idle in the application, different bounding boxes could still be inferred during sequential frames. This is why the threshold values of the idle identification algorithm must be taken into account, and the values should allow this randomness within the interval.

4.2 Tracking Model Evaluation

ByteTrack can be evaluated using the following metrics: multiple object tracking accuracy, multiple object tracking precision, ID F1 score, ID precision, ID recall, false positives, false negatives, ID switches and more. By using a benchmark that includes video footage and annotations with bounding boxes, object IDs and class labels, the evaluation of the metrics can be performed. Due to time constraints, no video annotation has been executed. Therefore, no evaluation of the tracking model can be done. However, it can be suggested that the performance would be semi-consistent with the results reported by the creators, given that no parameters have been changed compared to the original.

4.3 Idle Identification Evaluation

The idle identification algorithm can be evaluated using supervised learning methods. First, 150 frames functioning as data points from 8 instances are gathered and labeled using CVAI. This annotated data includes the ground truth bounding box, attached class, and track ID. This annotated data is used as the test set for the idle identification algorithm. After running the algorithm, predictions are made on the test set. The output labels (idle or not idle) are then used to calculate the accuracy, precision, recall, F1 score. The results are shown in Table 4.

The idle identification algorithm performs well based on the results in Table 4. The algorithm identifies idle states of machinery while avoiding false positives and accurately predicting actual idle states. Another performance advantage of the idle identification algorithm is its low computational overhead. Because it depends solely on a logistic regression formula, it runs efficiently. However, if the buffer size is increased, it will require more computational resources. How much the buffer size depends on the computational overhead has not been tested for the performance evaluation.

Table 4. Evaluation Metrics

Metric	**Detection**	**Idle Identification**
Precision	80.08%	73.33%
Recall	64.99%	79.66%
F1 Score	71.75%	76.42%
mAP50	70.44%	N/A
mAP50-95	55.50%	N/A
Accuracy	N/A	78.99%

4.4 Overall Application Evaluation

The stages of the application (detection, tracking and idle identification) have been tested on surveillance's videos of a company in construction surveillance. While the videos are not annotated, a preliminary emperical evaluation of the application can be discussed.

The AI application was able to detect excavators and dump trucks, while cement mixer trucks were not included in the surveillance camera videos. The excavators operated most of the times on the edge of the video, meaning that the bounding boxes were small. This made detection of the idle status difficult as there was small variation in the centroid and area of the bounding boxes. However, because the bounding box was placed accurately around the excavator, idle identification was possible.

The tracking of distinct dump trucks was doable for the application, there were no unnecessary track ID switches and there was a smooth transition of the bounding box across the frames. The tracking of the excavators was less accurate, track ID changes occurred sometimes, even when it was still the same excavator. ByteTrack was unable to re-identify excavators after occlusion occurred by leaves of trees moved by the wind or if the excavator was behind a pile of sand for more than about 10 seconds.

The idle detection system generally worked well and accurately, even for small movements. However, there were times when motion was not detected, causing the excavator to be considered idle for a few seconds while it was actually active.

5 CONCLUSION

5.1 Summary of Findings

This study presented a potential algorithm for idle identification of construction machinery on the edge via a lightweight deep learning algorithm that can be embedded in surveillance camera systems. The algorithm consists of object detection, tracking, and idle state identification that can be used to monitor the utilization rates of construction machinery.

The key findings of this study are as follows. The trained YOLOV8 detection model was selected for its balance between accuracy and speed. The F1 score of the model was 71.75%, and the mAP50 was 70.44%. Both metrics indicate a reliable performance in detecting construction machinery. ByteTrack effectively tracked the machinery while maintaining high accuracy and fast processing speed. The idle identification algorithm, based on logistic regression, accurately predicted the idle states of machinery while having a low computational overhead.

The AI algorithm, therefore, has the ability to perform idle identification of construction machinery and can be used to monitor the utilization rate of construction machinery. This can all be done in real-time, on the edge, and with only the use of a CPU.

5.2 Implications for Practice

There is a potential for project-managers and contractors with this study to gain deeper insights in the utilization rate of construction machinery. These insights can be captured in assessing if there is an improvement of the Overall Equipment Effectiveness (OEE) of the machine. OEE is a metric that measures how effective a production process is utilized compared to its maximum potential [24]. It can be split in the proportion of time the machine is running compared to the scheduled operating time (availability), the speed and efficiency of operations (performance) and the number of task that meet the required standard (quality). The algorithm can contribute in improving:

- **Availability**: The algorithm can identify idle machines and help reduce unplanned downtime of the machines. If there is unplanned downtime, the algorithm could alert the project manager or contractor, who can then take proactive measures to address the downtime.
- **Performance**: The algorithm can track and detect machines, and identify patterns or inefficiencies in the machine tasks.
- **Quality**: The algorithm can prevent overuse or misuse of machinery.

The algorithm does not only have implications for the improvement of OEE. There are also other implications for practice. This includes: cost savings, as minimizing idle time could reduce fuel consumption and labor costs if more tasks could be performed in the same time; reduced carbon emissions, as idle times are reduced; making informed decisions regarding machinery deployment.

5.3 Limitations

The proposed algorithm in this study contains several parts that could potentially limit the effectiveness of the goal of the algorithm.

5.3.1 Surveillance Camera Systems Environment. One potential limitation of the proposed algorithm is the lack of experiments conducted in a Docker environment. A Docker environment runs several containers, and one of these containers can be the algorithm. However, this means that not all the CPU and RAM can be utilized for the algorithm, which could potentially impact its performance. This has not been tested. Another limitation is that only the most light-weighted pretrained model of YOLOV8. This pretrained model has a lower mAP compared to heavier pretrained models.

5.3.2 Tracking. The proposed algorithm assumes that the input of the camera is static, while most surveillance cameras have the capability to pan, tilt, and zoom. The effectiveness of the tracking algorithm decreases significantly if any of these actions are performed. This is due to the Kalman filters included in ByteTrack. Kalman filters estimate the next position of the bounding box. However, if there is pan, tilt, or zoom, this estimate makes no sense anymore. Even though the object remains the same, it obtains a different track ID. The machine also obtains a different track ID if it switches to another camera, as there is no re-identification over multiple cameras is implemented.

5.3.3 Idle Identification Machine Learning. The idle identification works via logistic regression. The logistic regression outputs a bounding line on a Cartesian plane for determining the idle state. If the coordinate is above the bounding line, it is idle, and if it is below the bounding line it is not idle. However no other binary classification algorithms have been tested to see if they perform better. Also, only two variables of the metadata is used for determining idle state, while more data could potentially result in better performance of the idle identification.

6 FUTURE WORK

There are still many challenges that needs to be researched in order for making the AI algorithm robust and easy-to-use. Currently, it has the potential to form the basis for many other algorithms. For example, video action recognition could be the next step of the algorithm, as this can easily interpret machine activities in real-time. However, current implementations require expensive high-end video cards [12, 18]. Research is needed to reduce computational load to enable the deployment of video action recognition in the portrayed surveillance camera environment.

Dynamic tracking capabilities with the pan, tilt and zoom properties of camera's could also be a future research direction. This, in combination with re-identification of machines over multiple camera's could enhance the current algorithm implementation significantly. This could positively impact the accuracy of the machine statistics, such as usage and location.

The last mentioned future work direction for this study is the enabling of heuristics for the idle identification step. Currently the algorithm only predicts if a machine is idle over the buffer size. With adding heuristics such as a machine is definitely inactive if there is an idle prediction for 5 minutes or more, gives potentially a better overview about the utilization of machinery.

7 ACKNOWLEDGEMENTS

The author wishes to express gratitude to Rob Bemthuis for his guidance and support throughout this research, and also thanks Jeroen Klein Brinke for his helpful contribution.

During this research, the author utilized ChatGPT-4 for assistance in debugging code, correcting grammatical errors, and structuring the LaTeX file. All outputs of the prompts were thoroughly reviewed and edited as needed. The author takes full responsibility for the final outcome. This research is part of the ECOLOGIC project, which was funded by the Dutch Ministry of Infrastructure and Water Management and TKI Dinalog (case no. 31192090).

REFERENCES

[1] Mahmoud Abouelyazid. 2023. Comparative evaluation of SORT, DeepSORT, and ByteTrack for multiple object tracking in highway videos. https://vectoral.org/index.php/IJSICS/article/view/97

[2] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. *International Conference on Software Engineering: Software Engineering in Practice* (5 2019). https://doi.org/10.1109/icse-seip.2019.00042

[3] Ehsan Rezazadeh Azar, Sven Dickinson, and Brenda McCabe. 2013. Server-Customer Interaction Tracker: Computer Vision–Based system to estimate Dirt-Loading cycles. *Journal of construction engineering and management* 139, 7 (7 2013), 785–794. https://doi.org/10.1061/(asce)co.1943-7862.0000652

[4] Murat Bakirci. 2024. Utilizing YOLOV8 for enhanced traffic monitoring in Intelligent Transportation Systems (ITS) applications. *Digital signal processing* (5 2024), 104594. https://doi.org/10.1016/j.dsp.2024.104594

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple online and realtime tracking. *International Conference on Image Processing* (9 2016). https://doi.org/10.1109/icip.2016.7533003

[6] Chen Chen, Zhenhua Zhu, and Amin Hammad. 2020. Automated excavators activity recognition and productivity analysis from construction site surveillance videos. *Automation in construction* 110 (2 2020), 103045. https://doi.org/10.1016/j.autcon.2019.103045

[7] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: efficient convolutional neural networks for mobile vision applications. https://arxiv.org/abs/1704.04861

[8] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. 2016. Speed/accuracy trade-offs for modern convolutional object detectors. https://arxiv.org/abs/1611.10012

[9] A M Husein, None Calvin, David Halim, Raymond Leo, and None William. 2019. Motion detect application with frame difference method on a surveillance camera. *Journal of physics. Conference series* 1230, 1 (7 2019), 012017. https://doi.org/10.1088/1742-6596/1230/1/012017

[10] Rakkshab Iyer, Kevin Prabhulal Bhensdadiya, and Priyansh Shashikant Ringe. 2021. Comparison of YOLOV3, YOLOV5s and MobileNet-SSD v2 for Real-Time Mask Detection. *ResearchGate* (7 2021). https://www.researchgate.net/publication/353211011_Comparison_of_YOLOv3_YOLOv5s_and_MobileNet-SSD_V2_for_Real-Time_Mask_Detection

[11] Hyunsoo Kim, Changbum R. Ahn, David Engelhaupt, and SangHyun Lee. 2018. Application of dynamic time warping to the recognition of mixed equipment activities in cycle time measurement. *Automation in construction* 87 (3 2018), 225–234. https://doi.org/10.1016/j.autcon.2017.12.014

[12] In-Sup Kim, Kamran Latif, Jeonghwan Kim, Abubakar Sharafat, Dong-Eun Lee, and Jongwon Seo. 2022. Vision-Based Activity Classification of excavators by bidirectional LSTM. *Applied sciences* 13, 1 (12 2022), 272. https://doi.org/10.3390/app13010272

[13] Chinthakindi Kiran Kumar and Kirti Rawal. 2022. A brief study on object detection and tracking. *Journal of physics. Conference series* 2327, 1 (8 2022), 012012. https://doi.org/10.1088/1742-6596/2327/1/012012

[14] Jun Ha Lee and Su Jeong You. 2024. Balancing Privacy and accuracy: Exploring the impact of data anonymization on deep learning models in computer vision. *IEEE access* 12 (1 2024), 8346–8358. https://doi.org/10.1109/access.2024.3352146

[15] Milad Memarzadeh, Mani Golparvar-Fard, and Juan Carlos Niebles. 2013. Automated 2D detection of construction equipment and workers from site video streams using histograms of oriented gradients and colors. *Automation in construction* 32 (7 2013), 24–37. https://doi.org/10.1016/j.autcon.2012.12.002

[16] Mohammad Hossein Nasseri, Hadi Moradi, Reshad Hosseini, and Mohammadreza Babaee. 2021. Simple online and real-time tracking with occlusion handling. https://arxiv.org/abs/2103.04147

[17] Nipun D. Nath and Amir H. Behzadan. 2020. Deep convolutional networks for construction object detection under different visual conditions. *Frontiers in built environment* 6 (8 2020). https://doi.org/10.3389/fbuil.2020.00097

[18] Aravinda S. Rao, Marko Radanovic, Yuguang Liu, Songbo Hu, Yihai Fang, Kourosh Khoshelham, Marimuthu Palaniswami, and Tuan Ngo. 2022. Real-time monitoring of construction sites: Sensors, methods, and applications. *Automation in construction* 136 (4 2022), 104099. https://doi.org/10.1016/j.autcon.2021.104099

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2015. You only look once: Unified, Real-Time Object Detection. https://arxiv.org/abs/1506.02640

[20] Algemene Rekenkamer. 2022. Aanpak woningtekort. https://www.rekenkamer.nl/publicaties/rapporten/2022/06/23/aanpak-woningtekort

[21] Saleh Shahinfar, Paul Meek, and Greg Falzon. 2020. "How many images do I need?" Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. *Ecological informatics* 57 (5 2020), 101085. https://doi.org/10.1016/j.ecoinf.2020.101085

[22] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: vision and challenges. *IEEE Internet Of Things journal* 3, 5 (10 2016), 637–646. https://doi.org/10.1109/jiot.2016.2579198

[23] Madhuri Siddula, Fei Dai, Yanfang Ye, and Jianping Fan. 2016. Unsupervised feature learning for objects of interest detection in cluttered construction roof site images. *Procedia engineering* 145 (1 2016), 428–435. https://doi.org/10.1016/j.proeng.2016.04.010

[24] Ranteshwar Singh, Dhaval B. Shah, Ashish M. Gohil, and Milesh H. Shah. 2013. Overall Equipment Effectiveness (OEE) Calculation - Automation through Hardware amp; Software Development. *Procedia engineering* 51 (1 2013), 579–584. https://doi.org/10.1016/j.proeng.2013.01.082

[25] Banu Sizirici, Yohanna Fseha, Chung-Suk Cho, Ibrahim Yildiz, and Young-Ji Byon. 2021. A Review of Carbon Footprint Reduction in Construction Industry, from Design to Operation. *Materials* 14, 20 (10 2021), 6094. https://doi.org/10.3390/ma14206094

[26] Mupparaju Sohan, Thotakura Sai Ram, and Ch. Venkata Rami Reddy. 2024. A review on YOLOV8 and its advancements. *Algorithms for intelligent systems* (1 2024), 529–545. 39 https://doi.org/10.1007/978-981-99-7962-2\{_

[27] Shrey Srivastava, Amit Vishvas Divekar, Chandu Anilkumar, Ishika Naik, Ved Kulkarni, and V. Pattabiraman. 2021. Comparative analysis of deep learning image detection algorithms. *Journal of big data* 8, 1 (5 2021). https://doi.org/10.1186/s40537-021-00434-w

[28] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple Online and Realtime Tracking with a Deep Association Metric. https://arxiv.org/abs/1703.07402

[29] Bo Xiao and Shih-Chung Kang. 2021. Development of an image data set of construction machines for deep learning object detection. *Journal of computing in civil engineering* 35, 2 (3 2021). https://doi.org/10.1061/(asce)cp.1943-5487.0000945

[30] An Xuehui, Zhou Li, Liu Zuguang, Wang Chengzhi, Li Pengfei, and Li Zhiwei. 2021. Dataset and benchmark for detecting moving objects in construction sites. *Automation in construction* 122 (2 2021), 103482. https://doi.org/10.1016/j.autcon.2020.103482

[31] Kin Yen, Travis Swanston, Vic Reveles, Bahram Ravani, and Ty A. Lasky. 2011. Identifying excessive vehicle idling and opportunities for off-road fuel tax credits for stationary operations in the Caltrans fleet, phase 1. https://rosap.ntl.bts.gov/view/dot/27703

[32] Xue Ying. 2019. An Overview of Overfitting and its Solutions. *Journal of physics. Conference series* 1168 (2 2019), 022022. https://doi.org/10.1088/1742-6596/1168/2/022022

[33] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2021. ByteTrack: Multi-Object tracking by associating every detection box. https://arxiv.org/abs/2110.06864