Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS) pp. 277–285
DOI: 10.15439/2025F5518
ISSN 2300-5963 ACSIS, Vol. 43

Real-time Container Tracking and Damage Detection at Seaports Using Deep Learning

Sotiris Vasileiadis*, Sheraz Aslam*, Kyriacos Orphanides†, Alessandro Cassera†,
Eduardo Garro Crevillen‡, Alvaro Martinez-Romero‡, Michalis P. Michaelides* and Herodotos Herodotou*
*Cyprus University of Technology, 3036 Limassol, Cyprus
ORCID: 0009-0002-4479-1169, 0000-0003-4305-0908, 0000-0002-0549-704X, 0000-0002-8717-1691
Email: sr.vasileiadis@edu.cut.ac.cy; sheraz.aslam@cut.ac.cy; michalis.michaelides@cut.ac.cy; herodotos.herodotou@cut.ac.cy
†EUROGATE Container Terminal Limassol Ltd, 3045 Limassol, Cyprus
ORCID: 0009-0009-4510-3939, 0009-0005-5616-149X
Email: kyriacos.orphanides@eurogate-limassol.com; alessandro.cassera@eurogate-limassol.com
‡Prodevelop S.L, 46001 Valencia, Spain
ORCID: 0000-0002-8160-0125, 0009-0002-2154-0906
Email: egarro@prodevelop.es; amromero@prodevelop.es

***Abstract*—Efficient container handling and early damage detection are critical for minimizing operational delays, reducing costs, and ensuring safety in global maritime logistics. This work presents a deep learning-based methodology for real-time container tracking and automated damage detection during crane unloading operations at container terminals. We develop and deploy two specialized YOLOv12-based object detection models: one for identifying containers in motion and another for detecting structural damages such as bents, dents, and holes. Our models are trained and evaluated on a real-world dataset curated from video feeds captured at the EUROGATE Container Terminal in Limassol, Cyprus. The system is designed for robust performance under realistic terminal conditions, including variable lighting and motion. Our models achieve high detection accuracy, with a mAP50 of 0.99 for container detection and 0.75 for damage detection, substantially outperforming existing benchmarks. These results highlight the practical potential of our method for improving efficiency and safety in automated maritime logistics.**

I. Introduction

CONTAINERS play a pivotal role in global maritime commerce by supporting the movement of incoming, outgoing, and transshipped goods. Due to the significant increase in international seaborne containerized trade volumes, port operators must improve and automate various terminal operations to meet the growing demand and reduce safety risks due to human-computer cross-operations [1]. Approximately 20 million active containers are in use worldwide, completing over 200 million trips annually. Ports serve as central hubs where cargo from around the world converges as it enters or exits. These terminals are responsible for securely loading, unloading, and handling containers. However, the high volume of containers passing through makes it challenging to track damaged ones. Most damage detection methods are based on manual inspection around the globe [2], [3], which causes low efficiency, low accuracy, slow speed, high cost, and safety risks. Hence, an efficient inspection of the container body is essential to reduce disputes between container terminals and transportation enterprises due to damage to containers [4]. The common damaged types of containers include dents, scratches, distortions, holes, convex, damage to container doors, etc. [4]. These damages can result from various factors, including harsh weather conditions, poor transportation, or inadequate storage at the terminal. If the port fails to identify damaged containers upon unloading, the shipping company may impose fines, claiming the damage occurred within the port rather than beforehand.

In current research studies, computer vision and machine/deep learning methods are commonly used to detect damages in containers at seaports [2]. Machine learning methods are extensively adopted for object detection in images and videos in various fields, including medical, security, military, construction, and transportation [5]. To deal with damage detection in maritime containers at the ports, several maritime companies, stakeholders, and researchers started investing in automatic methods to detect damaged containers [4], [6]. For instance, the authors of [7] propose an image processing-based method (e.g., looking at hue, saturation, and intensity) using sensors and cameras. Another study [8] employs a phase correlation approach to detect damage in containers. Both studies rely on manual detection and employ cloud-based processing and storage. Any information that must be accessed is transmitted from the cloud to the terminal, which consumes significant time and resources. Due to the high real-time requirements of container detection, this approach is impractical. To address this limitation of cloud computing, the authors of [9] proposed the concept of edge computing, which enables effective data processing at the source, thereby meeting real-time requirements.

To improve the efficiency and accuracy of container damage detection, the current study develops a deep learning-based method for performing two key tasks in real time: 1) detection of containers and 2) detection of damages on containers using video data. We opt for the YOLOv12 model, a state-of-the-art object detection algorithm designed for fast and accurate real-time applications. It is trained initially on the COCO128

dataset for 100 epochs, optimizing parameters using techniques like stochastic gradient descent (SGD) or its variants. The models were then fine-tuned using the transfer learning methodology with real, manually-labeled datasets. Our extensive experimental evaluation demonstrates our proposed approach's efficacy in terms of higher accuracy regarding container and damage detection.

The remaining manuscript is organized as follows. Related work is presented in Section II. Section III presents comprehensive details of the proposed methodology along with a description of the employed datasets. Section IV shows simulation settings along with results and discussion. Finally, Section V concludes the study.

II. Related Work

Several pieces of research focus on developing automatic container and damage detection approaches. The most relevant ones are reviewed and analyzed in this section.

A study disclosed in [10] proposes a system for container damage detection using the correlation coefficient method. The system can be set at the port gate to detect damage in the arriving containers. Another study [11] develops a container damage detection model by proposing an Fmask-RCNN model. It is based on Mask-RCNN, introducing the Res2Net101 framework and adding path fusion augmentation, multiple fully connected layers, and fusion upsampling. The Fmask-RCNN model is applied to the identification of port container damage. Simulation results reveal that their proposed model achieves a miss rate of 4.59% and an error rate of 18.88%. The authors of [6] developed an automatic system by employing CNN and transfer learning from MobileNetV2 [12] and InceptionV3 [13] models to detect various types of damage in maritime containers. They also develop a dataset considering nine typical types of container damage. The proposed model and the existing methods are trained on developed data and then validated on test data. Results from experiments demonstrate higher accuracy in damage detection compared to prior approaches.

In [14], an automatic system (Spatial Structure Window plus texture clustering by K-Means algorithm) is used to recognize ISO-codes, improving the automatic methods that rapidly identify containers. On the one hand, this method increases the efficiency with which containers are registered after being discharged (thus increasing the supply chain and international trade efficiency). On the other hand, it eliminates the human-led errors likely to occur before the process's digitalization. In [15], an automatic image recognition system is used together with an adaptive control system based on neural networks for container identification and its position for smooth container landing on the platform. The benefits these authors mention are more effective control, reduced waste, and more precise measurements. A study presented in [16] develops a computer vision-based model to minimize total costs at terminals and reduce delay time by avoiding wrong container unloading. For this, an alarm system is designed based on container color detection. The proposed method alerts the quay crane operator if the detected color of a container does not comply with the correct color. It is concluded from several experiments at the Hong Kong Port that the proposed method can help in saving resources, especially total annual costs up to 85%.

The authors of [17] propose a computer vision-based method to determine the container location during the loading operation and record it. This process is typically performed by the human eye, and sometimes small miscalculations cause dangerous consequences. For accurate location prediction, the proposed method uses coastline, container edge line, and container number as features to locate and analyze. Experimental results from Ningbo-Zhoushan Port, China, show the proposed method's efficacy in accurate location prediction with minimum processing time (less than 0.6 seconds). Another study [18] also deals with container position prediction for gantry cranes at the container terminals by proposing a computer vision method. The primary objective of the newly developed system is to provide precise parameters for container lifting operations. The developed method utilizes cameras for collecting container information, then the corners of the containers are detected by employing a traditional image processing algorithm and a convolutional neural network (CNN). Finally, the offset distance and deflection angle are measured by precise corner position. Results from simulations demonstrate that the detection rate of the proposed system reaches 94%. Huang *et al.* propose an accident prevention system based on vision tracking during container lifting operations [19]. The developed system uses a camera to detect and track the movement of the truck wheel hub and the corners of the container to be loaded. The proposed algorithm combines CNN, traditional image processing, and a multi-target tracking algorithm to calculate the displacement and posture information of the truck during the operation. Experiment results show that the proposed method's measurement accuracy reaches 52mm.

III. Adopted Methodology and Dataset Description

The current study develops two models for container detection and damage detection for the Port of Limassol, Cyprus. Both models utilize a pre-trained YOLOv12 model [20], [21], which we fine-tuned on custom datasets created specifically for this research. Separating the tasks into two models provides key advantages: damage detection is only performed when a container is first identified, reducing unnecessary computation and enabling the second model to be more specialized and focused. Further details regarding the datasets and the chosen methodology are presented next.

A. Dataset for Container and Damage Detection

This study utilizes a real-world dataset from EUROGATE Container Terminal Limassol, Cyprus. The dataset comprises video data recorded by IPTV cameras installed on the quay cranes responsible for loading/unloading containers to/from vessels. The data from these videos was converted into image frames, which were manually labeled when they contained containers and container damage.

1) Data Preparation: To prepare the data for the container detection model, we collected videos taken throughout the year 2024, at different times of day and night, and under different weather conditions (e.g., sunny, cloudy, raining). We converted these videos into frames, after which we manually reviewed each frame, defined a bounding box around the containers using an open-source tool called LabelImg (also known as Label Studio) [22]. A total of 1927 images with containers were labeled, containing containers of different sizes (e.g., 20 ft, 40 ft), types (e.g., regular, reefer, open top), and colors (e.g., yellow, blue, red).

For the damage detection model, we utilized images from the container dataset. We followed the same labeling procedure with video data as the first model, but for the types of damages instead, as shown in Figure 1. We recorded three main types of container damage that are of interested to the container terminal: (a) *holes*, which represent visible holes on the container surface; (b) *bents/dents*, which represent deformation features such as concave damage, indentations, or arching on the container surface; and (c) *normal wear*, which represents minor damage such as cuts, cracks, rust, and other similar features on the container surface. A total of 732 images with damage were labeled.

2) Data Augmentation: Due to the limited amount of usable data (especially for damaged containers), we employed data augmentation techniques to expand our datasets. For both the container and the damage datasets, we added images using horizontal flips. In addition, we added custom-created damage images, specifically for holes, as the available data was severely limited due to it being a rare occurrence. By using these methods, we successfully increased our datasets from 1927 and 732 images to 3854 and 1764 images, for container and damage, respectively. Both datasets were then randomly split into training, validation, and testing. The containers dataset contains 2470, 598, and 786 images for training, validation, and testing, respectively, while the damage dataset includes 1442, 182, and 140 images for training, validation, and testing, respectively.

(a) Container Dataset

(b) Damage Dataset

Fig. 1: Sample of labeled images from (a) the container and (b) the damage datasets

B. Model Development

YOLOv12 is a cutting-edge object detection algorithm engineered to deliver speed and accuracy for real-time applications [23]. Trained on the COCO128 dataset over 100 epochs, it refines parameters through techniques such as stochastic gradient descent (SGD) or its derivatives. The YOLOv12 model architecture is constructed upon a deep convolutional neural network (CNN) that divides the image into grids and predicts objects in each grid cell. Unlike traditional two-stage detectors (e.g., R-CNN), YOLO is a single-stage detector, making it fast and suitable for real-time applications like video surveillance, autonomous driving, and robotics [24].

For the container detection model, we utilize our dataset of container images to train a customized YOLOv12 model. The training process involved fine-tuning several key training parameters to optimize performance: training for 80 epochs, using an image size of 1024 pixels to capture detail, and setting a batch size of 24 for efficient processing. We also built on the YOLOv12n pre-trained model, chosen for its strong balance of speed and capability. This setup allowed us to achieve accurate and reliable container detection suited to our application.

For the damage detection scenario, we trained the same pre-trained YOLOv12n model on our damage dataset. We also perform model optimization, like the one performed in the first stage for the container identification model. The model was trained on 1442 images and validated on 182 images. The model is able to classify the type of damage into three categories: Holes, Bents/Dents, and Normal Wear.

To improve our models' performance, we tested different configuration options provided by the YOLOv12 model, a very common practice known as hyperparameter tuning. This included adjusting the hue and saturation of the input images and tuning the IoU threshold to better detect multiple containers and damages that appear close together. The parameter ranges and optimal values are listed in Table I, which are the same for both the container and damage detection models.

Fig. 2: Tracking experiment with reassignment box visible

TABLE I: Parameter ranges and optimal values for model hyperparameter tuning

Parameter	Tested Range	Optimal Value
IoU Threshold	0.3 – 0.7	0.5
HSV Saturation	0.1 – 0.9	0.7
HSV Value	0.1 – 0.9	0.4
Image Resolution	640px, 1024px	1024px

C. Tracking Capabilities

YOLO includes integrated real-time tracking capabilities by implementing BoT-SORT [25], an advanced multi-object tracking algorithm that assigns a unique ID to each detected container. This allows us to track individual containers over time, recording details such as when each one was first and last seen, how long it was present, and whether any damage was detected. The model's streamlined design and adjustable tracking parameters make it adaptable to a wide range of deployment scenarios, from edge devices to cloud-based APIs [24].

One of the biggest difficulties faced was the occasional view impairment due to other external factors (trucks, straddle carriers, etc.) beyond our control, which could cause multiple IDs of the same container. To combat this, we implemented some extra checks needed before a container gets a new ID assigned, like a minimum time between detections, using detection coordinates before and after the object was lost, with a dynamic confidence threshold that works in tandem with the minimum time. In basic terms, if a container is lost for a few frames, the model will reassign the same ID to the container as the confidence is very high. But as time passes, the confidence is reduced, and so does the probable location box of the object.

Figure 2 shows the tracking algorithm in action. In the first frame, we have the detection of a container with an ID of 1. In the second frame, the container is lost due to a straddle carrier passing over the container. The green outline represents the confidence of a re-assignment if an object is found again. The confidence is high because the container is lost for only a few frames. In the third frame, some time has passed, so the possible location of a reassignment converges towards the last known position of the last detection. The container is found again in the last frame, but only half of what it was. Because the new detection is inside the reassignment box, we reassign the same ID to the new detection. This algorithm has helped us solve many problems related to losing objects due to such external factors.

Fig. 3: Validation results for container detection

IV. Results and Discussion

This section presents the experimental evaluation results of the container and damage detection models using the YOLOv12 model. The study uses real-world datasets from the EUROGATE Container Terminal Limassol, Cyprus, as discussed in Section III-A. The training and testing of the models were performed on a server with two AMD EPYC 7352 24-core processors, 128 GB RAM, and an NVIDIA Tesla T4 GPU running Ubuntu 22.04.4 LTS. We first present the results for the container and damage detection models, and then compare them against other state-of-the-art approaches.

A. Container Detection Model Evaluation

Figure 3 shows some validation results for the container detection model, illustrating the model's ability to detect and bound containers in images, including scenarios with multiple containers per image or partially-viewed containers. To concretely evaluate the performance of the developed models, we employ the mAP50 and mAP50-95 metrics, as these are commonly used metrics in object detection models [26]. mAP50 measures the mean average precision at an intersection over union (IoU) threshold of 0.5, indicating how accurately the model detects objects with at least 50% overlap between

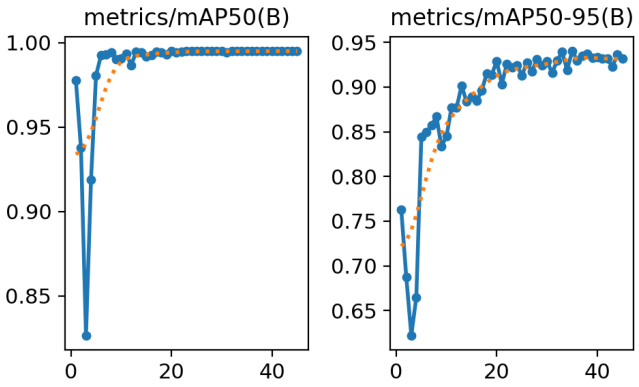

Fig. 4: mAP50 and mAP50-95 for container detection during training (the x-axis denotes the number of epochs)

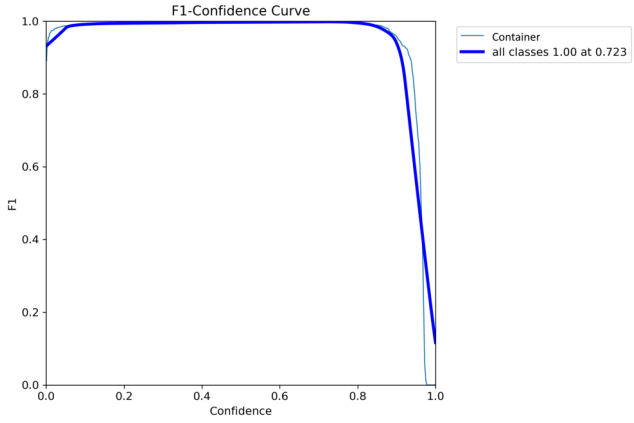

Fig. 5: F1 score for container detection across varying confidence thresholds

predicted and ground truth boxes. mAP50-95 measures the mean average precision across multiple IoU thresholds ranging from 0.5 to 0.95. We also investigate the F1 score, which represents the harmonic mean of the precision and recall. The selection of these evaluation metrics collectively contributes to a robust and comprehensive evaluation of the model's efficacy.

The evaluation of the container detection model, based on the mAP50 and mAP50-95 metrics, indicates remarkable performance. As illustrated in Figure 4, the model achieved mAP50 scores ranging from 0.96 to 0.99 after 10 epochs and mAP50-95 scores from 0.91 to 0.93 after 20 epochs. Notably, the model reached its peak performance at the 35th epoch, where it recorded a mAP50 score of 0.9921 and a mAP50-95 score of 0.9331. The initial drops in both figures are common and reflect sensitivity to noise or random initialization, but both metrics stabilize quickly. These high scores demonstrate the model's robustness and precision, highlighting its efficacy and reliability in container detection. Even with the relatively low amount of data, our developed model performs well in detecting containers.

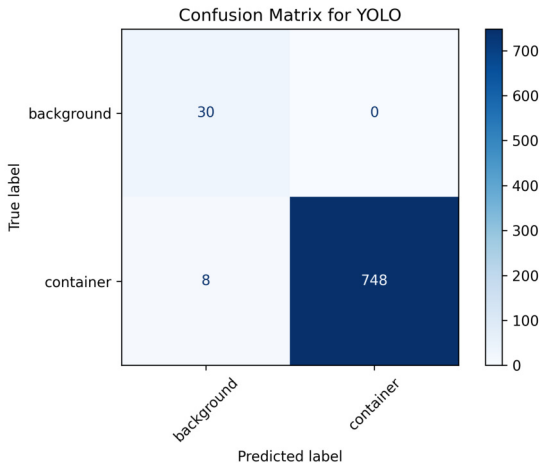

Fig. 6: Confusion matrix for container detection

The results depicted in Figure 5 show the F1 scores of the container detection model across varying confidence thresholds, providing insights into the model's balance between false positives and false negatives. The model achieves high performance, with a near-perfect F1 score at the confidence thresholds between 0.5 and 0.8. This indicates a strong balance between precision and recall, confirming the model's reliability in detecting containers accurately. The consistently high F1 scores across thresholds further demonstrate the robustness of the detection system.

This study also constructs a confusion matrix to validate the performance of the container detection model, shown in Figure 6. A confusion matrix serves as a valuable tool to assess the prediction/detection accuracy of any model, and it supports visual and robust validation of the approach. This matrix enables the comparison of actual and predicted values for each class within the dataset, providing valuable insights into key metrics such as true positives, true negatives, false positives, and false negatives. Figure 6 illustrates that the container detection model only misclassified 8 images (as false negatives) out of the 786 images of the test set, validating the strong detection power of the model.

B. Damage Detection Model Evaluation

Figure 7 shows some results for the damage detection model, illustrating that the model can detect and differentiate various types of damages, as well as detect multiple damages per container (if any).

Figure 8 shows the results of the container damage detection model. The damage detection model achieved a final mAP50 of 0.685 and mAP50-95 of 0.458 after 80 training epochs, indicating moderate detection accuracy. The metrics improved steadily throughout training, suggesting a smooth training process. The main reasons for the lower performance compared to the container detection model are the limited size of the dataset, as well as the far more complicated detection problem, since container damages are often small

Fig. 7: Validation results for damage detection

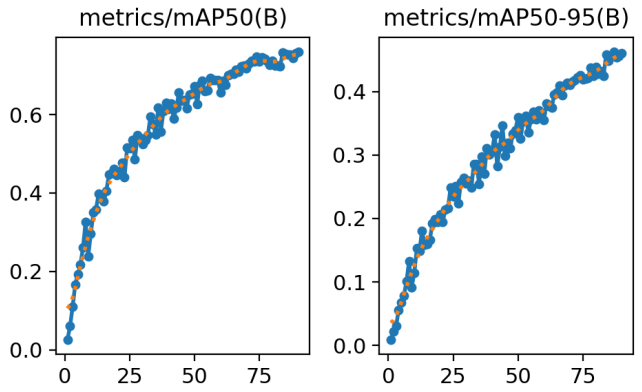

Fig. 8: mAP50 and mAP50-95 for damage detection during training (the x-axis denotes the number of epochs)

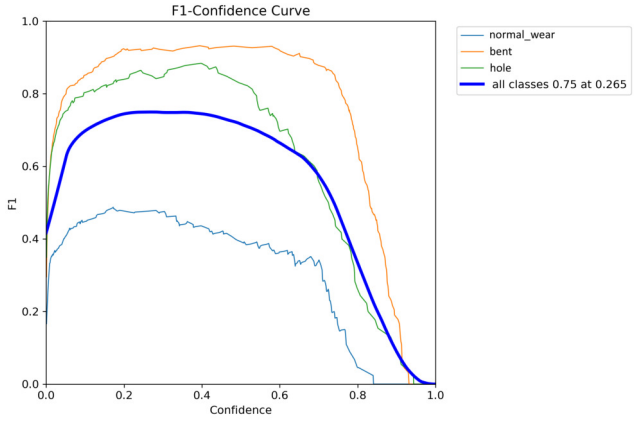

Fig. 9: F1 score for damage detection across varying confidence thresholds

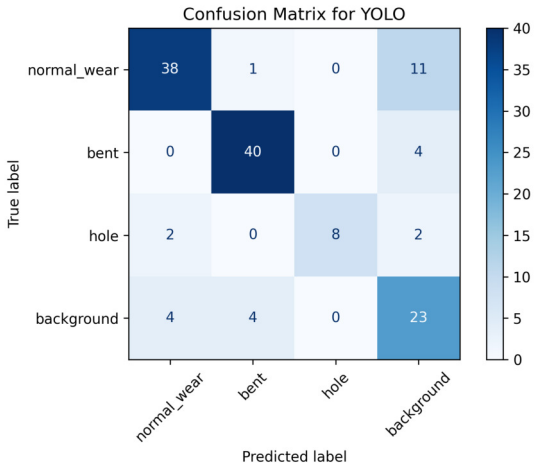

Fig. 10: Confusion matrix for multi-class damage detection

in size (compared to a container) and hard to distinguish from the rough container surface. However, this performance is significantly better than the damage detection achieved by other state-of-the-art models, as we will see in Section IV-D.

The results depicted in Figure 9 show the F1 score of the damage detection model across varying confidence thresholds. While the average F1 score peaks at 0.75 at a 0.27 confidence threshold, the model achieves much higher F1 scores for detecting the more important damages of bents/dents (0.92) and holes (0.88) at the 0.5 confidence threshold. The main difficulty for the model is the detection of normal wear, which is typically much more subtle compared to the other damages. The lower F1 scores are attributed to the limited dataset, as well as the more complex nature of damages (different sizes, types, locations, etc.).

Figure 10 shows the confusion matrix of the damage detection model based on the testing data, which again shows the good performance of the model in detecting and classifying the types of damages found. The confusion matrix reveals that the model achieves high accuracy for the bent/dent damages (40/44 correct) and performs reasonably well for normal wear (38/50 correct). The most frequent misclassifications occur when the model fails to detect some minor normal wear areas. The model also exhibits some confusion between the hole and the other damage types, mainly due to the small training and testing sizes. These results indicate some areas for improvement in discriminative learning and class boundary clarity.

Driven by the above results and the small difficulties observed in detecting specific damage types, we also developed a binary damage detection model, which detects whether damage is present without attempting to categorize it. The key motivation behind this approach was to evaluate whether combining all damage types into a single class could improve the overall detection performance by simplifying the task and leveraging a larger combined dataset.

Figure 11 shows the mAP50 and mAP50-95 metrics for the binary damage detection model. When comparing Figures

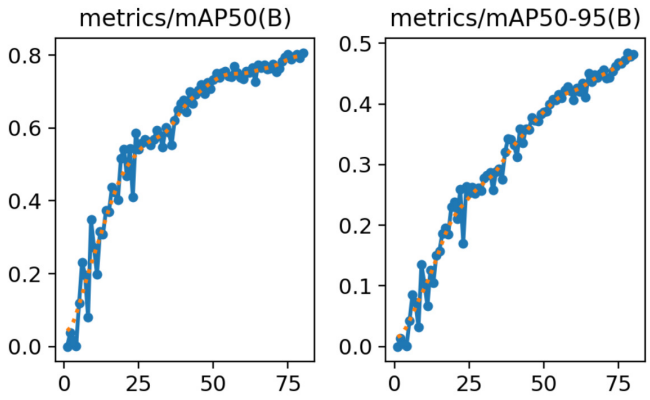

Fig. 11: mAP50 and mAP50-95 for binary damage detection during training (the x-axis denotes the number of epochs)

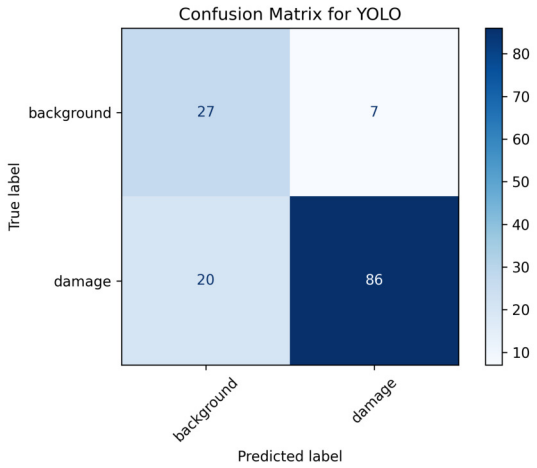

Fig. 12: Confusion matrix for binary damage detection

8 and 11, we observe that the binary model achieved higher mAP50 and mAP50-95 scores. Specifically, the binary model reached a final mAP50 of 0.79 and a mAP50-95 of 0.48 after 80 training epochs, as opposed to 0.69 and 0.46, respectively, for the multi-class model. This indicates that a simplified binary formulation benefits from greater sample density per class and suggests that our multi-class model could also benefit from a larger, more diverse dataset.

The confusion matrix for the binary damage detection is shown in Figure 12, revealing that the model achieved a precision of 92.5%, a recall of 81.1%, and an F1-score of 86.4%. These values reflect high confidence in positive predictions, but also reveal that 20 out of 106 damaged instances were missed. While the binary damage/no-damage model achieved high performance, the multi-class model revealed important distinctions in model performance across specific damage types. In particular, the model performed very well on bent/dent detections but struggled to identify hole instances reliably. This suggests that while binary classification may be sufficient for general screening, multi-class classification is essential when damage type differentiation is operationally important.

TABLE II: Inference speed and latency breakdown on two hardware setups

Metric	Jetson AGX Orin	Server Machine
GPU	Integrated	NVIDIA T4
Container Detection (ms)	37.5	25.0
Damage Detection (ms)	18.8	12.5
Other Processing (ms)	6.2	4.2
Total Frame Time (ms)	62.5	41.7
Achieved FPS	16	24

C. Model Inference Speed

Minimizing latency was a key requirement for these models, given their intended use in real-time applications. To evaluate performance, we benchmarked the tracking system along with container and damage detection on two different machines: (i) the server machine that was used to train the models, and (ii) a Jetson AGX Orin 64GB device that will be deployed on the quay crane for processing the video stream from the IPTV cameras in real time.

Table II shows the average latency (in ms) required to detect a container, detect damages on a container, and other required processing (e.g., split the frame, track the container, output result, etc.). Latency values (ms) are approximate and estimated based on the proportional contribution of each component to the total frame processing time at 1024×1024 input resolution. Overall, even the more resource-constrained device, i.e., the Jetson AGX Orin, can fully process a frame in 62.5 ms, which showcases the lightweight nature of the developed models. Note that the input video streams are recorded at six frames per second (FPS). As shown in Table II, both the Jetson AGX Orin and the server equipped with an NVIDIA T4 GPU achieved inference speeds significantly higher than the input rate, at 16 FPS and 24 FPS, respectively. This performance margin enables the system to process multiple video streams concurrently while maintaining real-time responsiveness.

D. Comparison with Other Work

This section compares our approach with the latest state-of-the-art models proposed in [6], specifically MobileNetV2 [12] and InceptionV3 [13]. To ensure a fair and meaningful comparison, we made several adjustments to our YOLOv12 model. Since MobileNetV2 and InceptionV3 are classification models that assign a single label to each image, we modified YOLOv12 to output only the most important class per image (holes, bents/dents, normal wear), effectively aligning its output format with that of the classification models. This allowed us to directly compare classification performance across all models under similar evaluation conditions.

Table III summarizes the classification performance of MobileNetV2, InceptionV3, and YOLOv12 in terms of precision, recall, and F1 score. The results clearly show that YOLOv12 significantly outperforms both MobileNetV2 and InceptionV3 across all classes and evaluation metrics. YOLOv12 achieved

TABLE III: Multi-class damage detection performance comparison across models

Class	MobileNetV2				InceptionV3				YOLOv12			
	Prec.	Rec.	F1	Pred/Real	Prec.	Rec.	F1	Pred/Real	Prec.	Rec.	F1	Pred/Real
Normal Wear	0.43	0.26	0.33	10/38	0.56	0.13	0.21	5/38	0.85	0.70	0.77	28/38
Bent/Dent	0.56	0.18	0.27	10/56	0.66	0.52	0.58	29/56	0.91	0.89	0.90	50/56
Hole	0.10	0.33	0.15	4/12	0.10	0.42	0.16	5/12	0.90	0.75	0.82	9/12
No Damage	0.00	0.00	0.00	0/34	0.20	0.33	0.25	1/34	0.69	0.73	0.71	29/34
Macro Avg	0.27	0.19	0.19	–	0.38	0.35	0.30	–	0.84	0.77	0.80	–
Weighted Avg	0.45	0.22	0.27	–	0.55	0.37	0.40	–	0.84	0.80	0.82	–

Note: *Prec.* = Precision, *Rec.* = Recall, *F1* = F1-score, *Pred/Real* = Number of predicted samples over ground truth count for each class.

TABLE IV: Binary damage detection performance comparison across models

Class	MobileNetV2				InceptionV3				YOLOv12			
	Prec.	Rec.	F1	Pred/Real	Prec.	Rec.	F1	Pred/Real	Prec.	Rec.	F1	Pred/Real
Damage	0.72	0.59	0.65	63/106	0.78	0.66	0.71	70/106	0.89	0.82	0.85	87/106
No Damage	0.17	0.26	0.21	9/34	0.28	0.41	0.33	14/34	0.55	0.68	0.61	23/34
Macro Avg	0.44	0.43	0.43	–	0.53	0.54	0.52	–	0.72	0.75	0.73	–
Weighted Avg	0.58	0.51	0.54	–	0.66	0.60	0.62	–	0.82	0.79	0.79	–

Note: *Prec.* = Precision, *Rec.* = Recall, *F1* = F1-score, *Pred/Real* = Number of predicted samples over ground truth count for each class.

a macro-average F1-score of 0.80 and a weighted average of 0.82, while InceptionV3 reached 0.30 and 0.40, and MobileNetV2 only managed 0.19 and 0.27, respectively. A closer look reveals that both MobileNetV2 and InceptionV3 struggled to correctly identify images with no visible damage, often misclassifying them as one of the damage types. YOLOv12, on the other hand, maintained a relatively high F1-score in these cases by correctly predicting the absence of damage in most of the 34 images.

An important factor influencing these results is the input resolution of each model. MobileNetV2 operates on 224×224 pixel images, InceptionV3 on 299×299, and YOLOv12 on 1024×1024. The performance of the models shows a clear correlation with input size: larger inputs allow more detailed spatial information to be retained, which is especially important for detecting subtle or localized damage features. This linear relationship between input resolution and classification performance highlights the importance of high-resolution processing in damage detection tasks.

To further confirm our results, we ran the same tests by creating models for container detection and binary damage detection. The performance metrics followed the same pattern shown in Table IV for the binary damage detection models. MobileNetV2 and InceptionV3 achieved better results in the binary case compared to the multi-class case, but they were still inferior to the results yielded by our YOLOv12 models.

In summary, YOLOv12 not only provides stronger performance due to its architecture but also benefits from higher input resolution, making it more reliable in distinguishing between different damage types and identifying undamaged cases correctly. In addition, MobileNetV2 and InceptionV3 can only detect the presence of damage on a container, whereas our model can (i) detect multiple damage areas on the same container (if there are any), and (ii) localize the damage by outputting a bounding box around each damaged area. Hence, our proposed model not only offers higher detection performance but also offers additional useful functionality.

V. Conclusion

In this study, YOLOv12-based container detection and damage detection models have been proposed to enhance optimization at the container terminal. The primary aim of this study is to reduce manual operations by humans in a risky environment by proposing automated systems to perform the same tasks. The proposed models are trained using real-world data collected from the EUROGATE Container Terminal Limassol, Cyprus. Due to the low amount of available data, this study also adopts data augmentation to increase the dataset. Finally, the proposed models are validated on test data, and the results show the high performance of the proposed models in terms of high accuracy and low error rate. The proposed models offer significant advantages over the existing state-of-the-art approaches, both in terms of higher detection accuracy as well as providing new functionalities, including tracking multiple containers in the same video frame and detecting multiple damages (of potentially different types) on the same container.

In future work, we aim to leverage the current damage detection model in a feedback loop that will automatically identify and incorporate new instances of container damage,

thereby expanding our dataset and improving model performance over time. In addition, we plan to extend the system's capabilities to detect whether container seals are intact or broken. Finally, we intend to integrate the detection models into a real-time monitoring system that operates through port IP network cameras and interfaces directly with the port management infrastructure.

ACKNOWLEDGMENT

This research was funded by the European Union's Horizon Europe program for Research and Innovation through the aerOS project under Grant No. 101069732.

REFERENCES

[1] S. Aslam, M. P. Michaelides, and H. Herodotou, "A Survey on Computational Intelligence Approaches for Intelligent Marine Terminal Operations," *IET Intelligent Transport Systems*, vol. 18, no. 5, pp. 755–793, 2024. doi: 10.1049/itr2.12469

[2] H. Wang, Q. Liu, and G. Zhang, "Container Damage Detection Algorithm Based on Fast-Solo," in *Chinese Intelligent Systems Conference*. Springer, 2022. doi: 10.1007/978-981-19-6226-4_13 pp. 119–131.

[3] S. Aslam, H. Herodotou, E. Garro, Á. Martínez-Romero, M. A. Burgos, A. Cassera, G. Papas, P. Dias, and M. P. Michaelides, "IoT for the Maritime Industry: Challenges and Emerging Applications," in *18th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2023. doi: 10.15439/2023F3625 pp. 855–858.

[4] X. Li, X. Huang, and Q. Liu, "Container Damage Identification Based on RP-FCN," in *39th Chinese Control Conference (CCC)*. IEEE, 2020. doi: 10.23919/CCC50068.2020.9189392 pp. 7031–7034.

[5] A. R. Pathak, M. Pandey, and S. Rautaray, "Application of Deep Learning for Object Detection," *Procedia Computer Science*, vol. 132, pp. 1706–1717, 2018. doi: 10.1016/j.procs.2018.05.144

[6] Z. Wang, J. Gao, Q. Zeng, and Y. Sun, "Multitype Damage Detection of Container using CNN based on Transfer Learning," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–12, 2021. doi: 10.1155/2021/5395494

[7] B.-S. Jeng, Q.-Z. Wu, Y.-P. Chen, and W.-Y. Cheng, "Image-based Container Defects Detector," Oct. 3 2006, US Patent 7,116,814.

[8] C. Tang, P. Chen, and Y. Li, "Automatic damage-detecting system for port container gate based on ai," in *Proceedings of the 2020 9th International Conference on Computing and Pattern Recognition*, 2020. doi: 10.1145/3436369.3436480 pp. 146–151.

[9] W. Shi, X. Zhang, Y. Wang, and Q. Zhang, "Edge Computing: State-of-the-art and Future Directions," *Journal of Computer Research and Development*, vol. 56, no. 1, pp. 69–89, 2019. doi: 10.7544/issn1000-1239.2019.20180760

[10] O. J. Ho, H. S. Woo, C. G. Jong, K. M. Ho, and A. D. Sung, "Development of the Container Damage Inspection System," *Journal of the Korean Society for Precision Engineering*, vol. 22, no. 1, pp. 82–88, 2005.

[11] X. Li, Q. Liu, J. Wang, and J. Wu, "Container Damage Identification based on Fmask-RCNN," in *Neural Computing for Advanced Applications: First International Conference, NCAA 2020, Shenzhen, China, July 3–5, 2020, Proceedings 1*. Springer, 2020. doi: 10.1007/978-981-15-7670-6_2 pp. 12–22.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[14] K.-m. Koo and E.-y. Cha, "A Novel Container ISO-code Recognition Method using Texture Clustering with a Spatial Structure Window," *International Journal of Advanced Science and Technology*, vol. 41, no. 4, pp. 83–92, 2012.

[15] A. Andziulis, T. Eglynas, M. Bogdevičius, T. Lenkauskas, and M. Jusis, "Development of an Adaptive Intermodal Container Handling Control Subsystem based on Automatic Recognition Algorithms," *European International Journal of Science and Technology*, vol. 5, no. 3, pp. 21–28, April 2016.

[16] R. Yan, X. Tian, S. Wang, and C. Peng, "Development of Computer Vision Informed Container Crane Operator Alarm Methods," *Transportmetrica A: Transport Science*, vol. 20, pp. 1–22, 2022. doi: 10.1080/23249935.2022.2145862

[17] M. Dai, Q. Liu, and J. Wang, "An Auxiliary Container Loading Location Algorithm based on Computer Vision," in *34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 2019, pp. 280–284.

[18] Y. Zhang, Y. Huang, Z. Zhang, O. Postolache, and C. Mi, "A Vision-based Container Position Measuring System for ARMG," *Measurement and Control*, vol. 56, p. 596–605, 2023.

[19] Q. Huang, Y. Huang, Z. Zhang, Y. Zhang, W. Mi, and C. Mi, "Truck-Lifting Prevention System Based on Vision Tracking for Container-Lifting Operation," *Journal of Advanced Transportation*, vol. 2021, pp. 1–9, 2021.

[20] Y. Tian, Q. Ye, and D. Doermann, "YOLOv12: Attention-centric Real-time Object Detectors," *arXiv preprint arXiv:2502.12524*, 2025.

[21] M. A. R. Alif and M. Hussain, "YOLOv12: A Breakdown of the Key Architectural Features," *arXiv preprint arXiv:2502.14740*, 2025.

[22] Tzutalin, "LabelImg," https://github.com/HumanSignal/labelImg, 2022.

[23] H. Lou, X. Duan, J. Guo, H. Liu, J. Gu, L. Bi, and H. Chen, "Dc-yolov8: small-size object detection algorithm based on camera sensor," *Electronics*, vol. 12, no. 10, p. 2323, 2023. doi: 10.3390/electronics12102323

[24] "Ultralytics - Revolutionizing the World of Vision AI," May 2024, https://www.ultralytics.com/.

[25] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "BoT-SORT: Robust Associations Multi-pedestrian Tracking," *arXiv preprint arXiv:2206.14651*, 2022. doi: 10.48550/arXiv.2206.14651

[26] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time Flying Object Detection with YOLOv8," *arXiv preprint arXiv:2305.09972*, 2023. doi: 10.48550/arXiv.2305.09972