Pipeline for Visual Container Inspection Application using Deep Learning

Guillem Delgado[a], Andoni Cortés[b] and Estíbaliz Loyo[c]
Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Mikeletegi 57, 20009, Donostia-San Sebastián, Spain

Keywords: Deep Learning, Shipping Container Inspection, Ship to Shore Inspection, Port Application.

Abstract: Containerized cargo transportation systems are associated to many visual inspection tasks. Especially during the process of loading and unloading containers from and to the vessel. More and more of these tasks are being automatized in order to speed up the overall process of transportation. This need for optimized processes calls for new vision systems based on the latest technologies to reduce operation times. In this paper, we propose a pipeline and a complete study of each of its parts in order to provide an end-to-end system that solves and automatizes the process of inspection of a loading or unloading freight container from and to the vessel. We outline all the components involved in a separated way. Tackling from the acquisition of the images at the beginning of the process, to visual inspection tasks such as containers' id detection, text recognition, damage classification or International Maritime Dangerous Goods (IMDG) detection. In addition, we also propose a heuristic algorithm that is capable of managing all the information from the multiple tasks in order to provide as much insights as possible out of the system.

1 INTRODUCTION

Containerized freight transport is the predominant method used nowadays to transport cargoes around the world. Its ability to store and organize provides an advantage against other methods of transportation. Moreover, the development of efficient hardware and software lowered the time on the procedure of goods exchanges and lowered the cost of it. This leads to the desirability of this kind of transportation and reduces the final consumer price of goods. This is especially important in maritime transportation as it is responsible for approximately 90% of global transportation, consequently becoming a backbone of global trade (Filom et al., 2022).

Automation is nothing new in this field, since it is becoming more and more common to automate the many tasks associated with each link in the transportation chain. Specially in port environments where the main tasks regarding the inspection of containers are usually reviewed by a port operator during the loading and unloading of the containers with a Ship To Shore (STS) crane. These tasks include the detection of the IMDG markers, container marker label recognition and seal presence detection, among others. In order to avoid conflicts between transportation companies and wharf, the visual inspection of containers must be carried out focusing on potential structural defects. Currently, two operators work in parallel. One person enters all these data into a terminal device, while the other one does the container inspection. This takes about 30 seconds per container. However, delays related to unnecessary waiting times means multiple people blocked for their respective tasks.

Due to the latest achievements of Deep Learning techniques in computer vision, it allowed advances in research regarding visual task of inspection of containers, such as segmentation of damages, IMDG marker detection, text recognition, corrosion inspection and so on.

Nevertheless, this automation has several drawbacks that has not been tackled yet, to the best of our knowledge. First of all, these deep learning algorithms require a big amount of data in order to work properly. Acquiring this data is an arduous challenge considering the fact that there are several privacy and confidentiality constraints in ports. In addition, there is a lot of time and cost that it is needed to devote in the data collection process which makes applying

[a] https://orcid.org/0000-0003-2240-9723
[b] https://orcid.org/0000-0002-7158-0175
[c] https://orcid.org/0000-0001-8232-2828

Delgado, G., Cortés, A. and Loyo, E.
Pipeline for Visual Container Inspection Application using Deep Learning.
DOI: 10.5220/0011590900003332
In *Proceedings of the 14th International Joint Conference on Computational Intelligence (IJCCI 2022)*, pages 404-411
ISBN: 978-989-758-611-8; ISSN: 2184-3236

and comparing state-of-the art techniques a difficult task. Secondly, the communication between the operation center and the STS crane tends to be located far from each other. This leads the operation center to be isolated making difficult to perform an effective data transmission. Thus, it is difficult to remotely analyse the information, due to bandwidth constrains. Finally, different inner visual task modules must be synchronized in order to associate correctly the retrieved information with a logical heuristic process.

In this work, we present an architecture that faces globally the visual inspection process of containers, grouping and solving multiple visual inspection tasks. Furthermore, we propose a custom heuristic algorithm where we interpret and associate retrieved data in order to provide detailed information within a video stream transmitted via 5G. We train the state-of-the art models and validate the whole pipeline using synthetic labelled data for a multi-camera system in a STS crane.

The paper is organized as follows: Section 2 describes the state of the art of the different visual tasks; Section 3 describes the proposed architecture, explaining the setup, the synthetic data generation, the visual inspection tasks solved and the heuristic algorithm; Section 4 explains the experimentation and the results obtained; finally, Sections 5 and 6 present the conclusion and future lines of work.

2 RELATED WORK

Deep Learning is being employed more and more in all computer vision tasks, including the visual inspection of shipping containers in port environments where neural networks have proven to provide more accurate results compared to traditional methods. Variations of state-of-the art networks has been proposed in order to solve different tasks. W Zhiming et al. (Zhiming et al., 2019) used Faster Region-based Convolutional Neural Networks (Faster-RCNN) with a binary search tree to find the container identifier. X Li et al. (Li et al., 2020) proposed a variation of the Mask-RCNN for damage detection called Fmask-RCNN which introduces changes in the backbone, fusions in the Feature Pyramid Network (FPN) and multiple fully connected layers. Furthermore, A Bahrami et al. (Bahrami et al., 2020) introduced neural networks architectures to detect corrosion in containers using different state-of-the art models such as Faster R-CNN, Single-Stage object Detection (SSD) Inception V2 and SSD-MobileNet. They included an anchor box optimizer in order to localize and detect the corrosion from the containers. Wang et al. (Wang et al., 2021) also proposed a lightweight neural network to only classify damages from containers using mobile phone devices. In addition, Verma et al. (Verma et al., 2016) addressed the problem of text recognition and put up a proposal for a solution in the form of an end-to-end pipeline that combines Region Proposals for detection with Spatial Transformer Networks for text recognition. Bu et al. (Bu et al., 2018) also proposed a method for text recognition in order to recognize container numbers in arbitrary directions and complex backgrounds. Other methods proposed in X. Feng et al. used You Only Look Once (YOLO) detector and Convolutional recurrent neural network (CRNN) and Efficient and Accurate Scene Text Detector (EAST) algorithms for text recognition (Feng et al., 2020b) (Feng et al., 2020a).

Nevertheless, all these methods use data that is typically not public, making the process of comparing the state of the art difficult. Therefore, synthetic data generation has received a lot of attention lately as it has proven to be a good approach to tackle the lack of data (Cortés et al., 2022). It already exists methods for creating synthetic data for Deep Learning models to train on, using the 3D components and description file of the scene as the input data (Aranjuelo et al., 2021). However, these methodologies have not been applied yet to the containerized freight transportation field, to the best of our knowledge.

3 PIPELINE

In this work, we present a pipeline that includes multiple modules, from modules for the purpose of detection and classification used in visual tasks, to modules focused on the connectivity and delivery of information, as well as a heuristic process for extracting information. The proposed setup can be seen in Figure 1. The pipeline consists in multiples cameras that transmit a video stream through 5G. Single camera setups could be used also in this pipeline. Then, a visual inspection tasks module uses deep learning state-of-the art models in order to solve multiple proposed tasks. The results are coded and sent via broker to another client that process temporally the data and provides information given a sequence of loading or unloading containers with an STS crane. The aim of the pipeline is to generate information regarding these containers and compare them between a list of the containers that should be loaded and unloaded for each day. Before anything else, synthetic data had to be generated in order to overcome the lack of data. Thus, allowing to train multiple inspection tasks with minimal effort. In this section we present the physical setup used in the

port, describing the architecture of the cameras and processes to automatize. Next, we present the process of creating the synthetic data in order to train state-of-the art models and validate the entire pipeline as real data is scarce on this field. We also introduce the different visual tasks and how they are used to extract multiple features from the container and how they are sent through a broker. Finally, we present a heuristic algorithm that is in charge of receiving the data and extract useful readable information for operators in order to automatize their process.

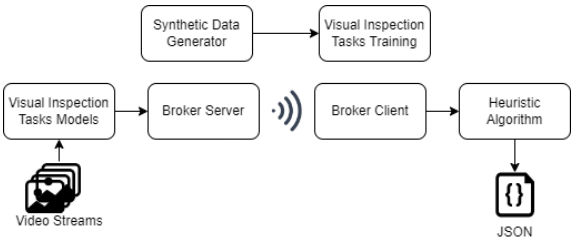

Figure 1: Overall pipeline used for visual inspection of containers.

3.1 Setup

Our setup is based in Luka Koper's port, where we installed four cameras placed in different parts of the STS crane so we can monitor the process of loading and unloading the container on each side of it. We define them as Camera A, Camera B and Cameras Cl and Cr, as we can see in Figure 2. Cameras A and B will be at the bottom of the crane, on the lower horizontal beam and will be always point at the Back and Front side of the container, that means towards land and sea sides. Meanwhile, Cameras Cl and Cr will be located in the upper horizontal beams, at 13m of height. Those cameras will point at, what we refer as, the Door and the No Door of the container.

The process we aim to automatise is the loading and unloading of containers from ship to shore or from shore to ship. The unloading process mainly consists of using the STS crane to discharge vessels continuously. Once they are moved from the ship to the shore, they are placed in one of the available rails and manual inspection is performed. Operators give emphasis on possible structural damages in order to prevent disputes between transportation companies and wharf. It takes a minimum of 30 seconds and multiple operators to perform all manual checks. In addition, delays might occur related to unnecessary waiting times, such as STS crane idling or uneven distributed movements on the yard. Once the container is successfully inspected, it is transported through the rails to its destination. The loading process consists of the same steps as stated previously, but the other way around.

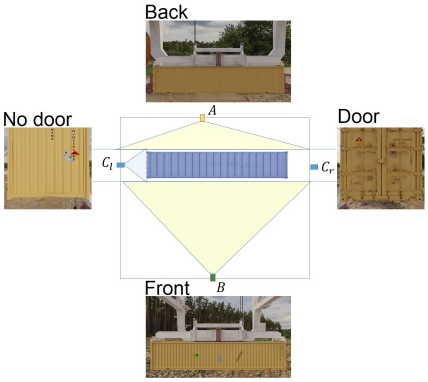

Figure 2: Configuration of the virtual cameras inside the 3D scenario.

The port has available 5G connection which will be used to transmit four videos' streams at high resolution (3840x2160) in order to process the images without any latency. However, we will not delve in further details about the 5G connection as it is not within the scope of the paper.

3.2 Synthetic Data

One of the main problems found recurrently in the state of the art is lack of publicly accessible port data and is one of the major constraints on port-related research. Since ports are important nodes in nations' supply chains, most of them withhold specific information about their operations in order to preserve their competitive advantage. Furthermore, there isn't a trustworthy benchmark for well-known port-related issues that may be used as a performance evaluation indicator. That is why in order to validate this pipeline we also generate our own synthetic data to train multiple visual tasks and test the entire proposed pipeline including the heuristic algorithm. The synthetic data, called SeaFront (Delgado et al., 2022), is publicly available and it can be used as a baseline for other research. This datasets consist of two splits of training and validation and it contains a total of 9888 images. For training there are 7910 images and 1978 for validation. In addition, a test set is also available with 2480 images. We can train multiple tasks with this data as it allows us to train detection and/or segmentation of damages, more specifically perforations, dents, and bents of the container. It also provides detection information regarding IMDG markers, container's position, and text identifier. We also generate annotations for text recognition tasks. Finally, it offers the possibility to train Cl and Cr Cameras for a binary classification of door or no door. In addition, we generate an extra 20 sequences of 30 images for

each camera in order to validate the entire pipeline and provide results and metrics.

On the other hand, we create a synthetic dataset (a sample can be seen in Figure 3) for optical character recognition (OCR) tasks using multiple backgrounds and placing the 26 upper case letters with 10 different numbers using multiple fonts. This dataset consists of images with vertical and horizontal labelled character sequences. Characters inside the images are generated randomly and placed in such a way that the sequence of characters occupies the entire image. Several augmentation processes are also applied to the generation of each character. These include the following processes: scale changes, light changes, shadows addition, padding, elastic transformations, rotation, blurring and a last step to add a random image as background. Although a subset of five system fonts have been defined to generate character's glyphs, this could be changed according to the context of the application.

Figure 3: Detection patches from real images (left). Synthetic character sequence generated image (right).

3.3 Visual Inspection Tasks

In this paper, we focus mainly on four visual inspection tasks that are found recurrently in the state of the art and aims to automatize the manual inspection of the containers. First of all, the damage detection and segmentation where we aim to predict the location and the label of its four possible damages (dents, bents, bents on axis and perforations) and the container. The location of these five classes can be via detection or segmentation. By using detection, we provide bounding boxes with a set of coordinates and its class confidence. By using semantic segmentation, we predict a class per-pixel of all the image. Secondly, the IMDG detection where we aim to detect multiple elements in the image. This includes the container, any vertical or horizontal text and the different 26 IMDG markers. Another visual inspection task is to perform a binary classification whether a door from a container is present or not. This is especially useful when loading and unloading containers such that do not follow the same orientation and we need to know where exactly is the door. Finally, the text recognition task, where given some crops of images where text appears, we aim to provide sequences of recognized characters. These texts can have multiple orientations such as horizontal or vertical writing and can have occlusions.

All these visual tasks will be processed independently in each camera defined previously. Each camera's instance reads from its stream and starts analysing the image before sending a message through a broker with its results. Thus, we propose the following algorithm to process multiple tasks.

```
Input: image
damages = Segmentation.infer(image)
detections = Detection.infer(image)
textsDetection = detections.getTextDet()
containers = NonMaxSup(damages, detections)
IF no containers
  sendEmptyMessage()
ENDIF
IF Camera is Cl or Cr
  door = Classifier.infer(image)
ENDIF
FOR container in containers
  croppedImage = CropFromContainer(image,
                                   container)
  textsDet = TextDetection.infer(croppedImage)
ENDFOR
texts = textsDetection + textsDet
FOR bbox in texts
  IF bbox not visited
    btree = Node(bbox)
  ENDIF
  FOR bbox+1 in texts
    IF bbox+1 not visited
        inserted = btree.insert(bbox+1)
    ENDIF
    IF inserted
        bbox+1 is visited
    ENDIF
    btreelist.append(btree)
  ENDFOR
  FOR btree in btreelist
    bbox = btree.getbbox()
    cropText = CropFromText(image, bbox)
    letters = CharRecog.infer(cropText)
  ENDFOR
sendMessage(detections, damages,
            door, letters)
```

First, we infer the bounding boxes location of the container using two different models. If the same container is detected from both models, we perform a non maximum suppression in order to remove those similar containers. On the other hand, if no container is found, we prepare and send an empty message. Additionally, we infer the IMDG and the texts' location, we classify the image as Door or NoDoor and we provide the segmentation of the damages. Once we have all the predictions ready, we include them in the message to be sent only if the container's confidence is above a given threshold. We also perform the recognition of the detected text. In order to do so, we use

the CRAFT (Baek et al., 2019) detector in addition to our trained character detector to generate a list of text bounding boxes. The reason we use CRAFT along with our trained model is because it is more robust on horizontal orientation rather than vertical orientation. As it is needed to provide solid results in both cases, we decided to merge both models' detections using a binary tree. Due to visual distortion or model's precision, the predictions do not line precisely in the same orientation. Therefore, to solve these issues we group these bounding boxes either horizontally or vertically. Using a binary tree allows to be efficient while being able to customize the insertions within the tree. We traverse each bounding box while creating a binary tree if it has not been previously inserted in another tree. Subsequently, we get the bounding box for each tree and use our character detector network in order to provide a single text for each box. In order to insert a bounding box inside the binary tree, certain tolerance angle is used, and they are determined according to empirical experience.

Finally, once all the tasks has been processed, we encode a message in Base64 and send it via our Kafka broker as a JSON message which will be read by the heuristic algorithm.

3.4 Heuristic Algorithm

The main objective of the heuristic algorithm is to process the raw information from the multiple detections in the video stream and provide some useful insights. As we can see in Figure 4, the algorithm is composed by multiple modules that handle the information along time and once the container has disappeared, it fuses the information extracted from the sequence. The proposed modules used are the Preprocess, the imFusion, the checkIso, the seqFusion and the sendContainer. The algorithm works as follows. After the visual inspection tasks are executed, some rule decisions must be defined so that the data detected can be post analysed. In order to process the data, we must take into consideration sequences of containers. This means that when a container is detected for the first time in a camera, a possible report will start to be considered. However, due to the proposed setup, a minimum of one JSON message will be received from the Kafka client up to a 4 in total from all the multiple video streams. We will consider that the container sequence has ended when no detection has been provided during a given time. First of all, $M = max(1, m_{cameras})$ received JSON messages will be pre-processed. We check first if multiple containers are next to each other, and their Intersection over Union (IoU) is less than a certain threshold. If this condition does not apply, it will remove the one with lower confidence. This is done to avoid multiple false positive detections and to detect multiple containers that can be carried in a single journey. Next, it checks if all the detected damages are inside its correspondent container. Regarding the imFusion module, it restructures the data and aggregates the four independent streams to one, so it makes it clear and easier to work with. It also changes the text confidence according to the face. We weight more Cl and Cr cameras as they are nearer the trajectory of the container tend to have more resolution on the text. Thus, providing a better prediction of the detected text. The checkISO module is in charge of proving that the different proposed text comply with the ISO6346 regulation (ISO, 2022). The ISO6346 code consists of three parts: four capital letters, six digits, and one check digit. There may be extra characters, but these eleven ISO characters are defined by the ISO standard are considered unique. To do so, we iterate over all the candidates that have 11 characters and check the text using the corresponding steps proposed in the ISO (ISO, 2022). If the last control digit is correct after the calculations, then the proposed text is validated. Otherwise, the proposal is discarded. In addition, the Type and Size code are also provided and checked by using look up tables. The seqFusion module, is in charge of storing and aggregating all the information within a frame and to provide the final information over time. We define then a sequence as a set of frames-timesteps that refers to the same element(s) (one or two containers). Therefore, it fuses all the data from the different instances and provides a single JSON, sets the timestamp range, the identifiers, the IMDG, the doors, the damages and finally outputs a JSON file with all the information. The detailed algorithm is the following:

```
function seqFusion(timestamp, data):
    checkFinished(timestamp,data)
    IF finished
        setTimestampRange()
        setIDS()
        setIMDG()
        setDoor()
        setDamages()
        resetVariables()
        return data
    ENDIF
function checkFinished(timestamp,data):
    FOR i, k in enumerate(data)
        sequencedata[k] += data[k]
    ENDFOR
    IF sequencedata is empty
        begin_timestamp = timestamp
        last_timestamp = timestamp
    ENDIF
    IF timestamp - last_timestamp > th_time
        return True
```

```
ELSE
    last_timestamp = timestamp
    return False
```

The modules used inside seqFusion are:

- setTimestampRange: Set the timestamp from the beginning to the end.
- setIDS: Set one or more identification numbers that passed the ISO check. If no text is detected set it as UNKNOWN. If multiple ID are available, it checks the coordinates and decide which one is on the left and on the right.
- setIMDG: Get the unique IMDG detections and set the face where it was located. In case multiple containers are present, location will determine in which containers they should go.
- setDoor: For Cl and Cr cameras, set door face as the maximum occurrences predicted. In case there are multiple containers, and their doors are not visible, set to NO_VISIBLE.
- setDamages: Similar to IMDG, get the unique damage for all detections and set the face where located. In case of multiple containers, location will determine in which containers they should go.

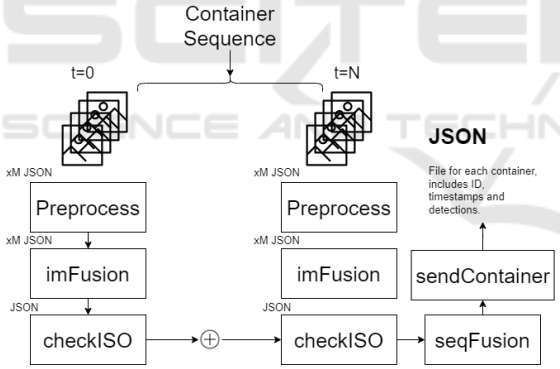

Figure 4: Overall pipeline for the heuristic algorithm defining the different modules.

4 EXPERIMENTS AND RESULTS

In this section, we demonstrate how training a model using precisely created synthetic images can yield results that are competitive with those of real datasets, which are challenging to get due to private entities in port environments and the difficulties of annotating them. In addition, we show how our proposed pipeline and heuristic algorithm can provide useful information from multiple streams of videos. For training and evaluation purposes, we used a single NVIDIA Tesla V100-SXM2 of 32 GB. All the tasks have been validated and tested using a synthetic dataset previously mentioned that has not been seen during training.

4.1 Visual Container Tasks

Using a Cascade Mask R-CNN (Cai and Vasconcelos, 2019), we solved detection and segmentation simultaneously. We utilize the detection power of cascade regression using a mix of Cascade R-CNN and Mask R-CNN. After that, we segment the detections using the Mask R-CNN. We trained the model for 20 epochs with a decreasing learning rate at 16 and 19 epochs. We used the MMDetection framework (Chen et al., 2019) in Pytorch, with a batch size of 16 samples and a ResNet50 (He et al., 2016) backbone with an input image of 1333x800. We provide the average precision (AP) for an intersection over the union (IOU) threshold of 0.5 (AP_{50}), 0.75 (AP_{75}) and 0.5:0.05:0.95 (AP). We also provide these metrics for small, medium and large objects (AP_S, AP_M, AP_L), for detection in Table 1 and segmentation in Table 2. As we can see, we have solid results, especially in large objects but the network has troubles detecting smaller objects. Nevertheless, regarding the detection and segmentation we have a viable method.

Table 1: Average precision of detection of damages.

AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
71.5	89.9	79.2	12.5	50.3	77.8

Table 2: Average precision of segmentation of damages.

AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
67.5	88.6	74.4	12.6	45.8	73.3

Regarding the detection of the IMDG markers, we used YOLOv5 using Ultralytics Pytorch framework (Jocher et al., 2021). We trained the model for 150 epochs with learning rate of 0.01 using all 28 classes. We used a batch size of 16 and an input image resolution of 640x640 letterbox scaled. As we can see in Table 3, some classes are simpler to distinguish from others. Some classes have greater inter-class correlation than others, making it more difficult to distinguish between them. Other classes, however, have more pronounced and robust visual traits, and the detector can quickly detect them. With a mAP of 74.9, the detector's overall precision reaches a precision of 67.9 and a recall of 84.0. Even if the results can potentially be improved, we believe that these results present also a baseline to help further research in this field.

The door/no door classification has been solved using ResNet-50. We trained the model for 100

Table 3: Metrics for items detection model.

Class	*P*	*R*	AP_{50}	*AP*
text	89.0	87.9	91.5	73.2
C1.1	54.5	59.3	65.2	59.0
C1.2	29.3	42.8	28.9	26.0
C1.3	28.9	49.9	30.3	27.4
C1.4	30.6	54.7	34.5	31.3
C2.1	50.4	65.4	53.3	47.8
C2.2	46.4	71.6	47.7	42.4
C2.3	87.3	98.7	98.9	87.6
C2.4	80.5	96.5	97.0	87.8
C2.5	43.1	72.7	46.3	40.7
C3.1	45.7	86.3	47.8	42.8
C3.2	49.2	83.1	51.5	45.7
C4.1	85.0	97.7	98.8	89.9
C4.2	90.8	95.6	97.4	88.0
C4.3	89.0	96.6	98.3	89.2
C4.4	85.5	97.5	97.9	88.7
C5.1	90.4	97.1	98.4	89.6
C5.2	91.4	96.0	97.6	89.3
C5.3	83.8	97.1	97.9	89.3
C6.1	47.2	69.0	49.0	44.1
C6.2	65.3	78.3	80.7	72.6
C7.1	78.0	94.6	95.0	85.1
C7.2	47.1	88.4	52.1	46.4
C7.3	85.1	88.0	92.6	83.7
C7.4	51.9	91.5	52.6	46.0
C8.1	88.7	98.4	98.5	89.3
C9.1	87.5	97.3	97.3	86.9
container	100	100	99.5	99.5

epochs. We carried out two main experiments. The first one was using a subset from cameras C_l and C_r and the second one, containing more negatives, we used the camera C_r as positives images and C_l, A and B cameras as negative images. We can see excellent outcomes in Table 4. Although some false positives have surfaced in experiment two due to the fact that we added more negatives, the classifier keeps the accuracy and recall quite high.

Table 4: Precision and recall of door classification.

	Precision	*Recall*
Experiment 1	100	100
Experiment 2	90.4	96.0

Finally, the text recognition task has been solved training a character detector which can handle the horizontal and vertical text recognition. We trained a YOLOv5 for 150 epochs with learning rate of 0.01 and used all Latin letters from A to Z and numbers from 0 to 9. We used a batch size of 16 and an input image resolution of 640x640 letterbox scaled. Model was trained with 9.600 images and validated with 2.400 images. The data used was entirely synthetically generated, trying to minimize the domain gap effect between real and synthetic using an iterative error analysis procedure. Regarding the text recognition, we used crops from the SeaFront synthetic data. We achieve a character precision and recall of 83.16% and 66.82% respectively. In addition, we calculated the Normalized Edit Distance (1-N.E.D) with a value of 62.78%. It seems that we might have low values, but this dataset is pretty challenging as multiple occlusions might occur due to IMDG markers placed in text, deformations or perforations removing part of the text or the same structure can hide parts of the text.

4.2 Pipeline Testing

In order to validate the entire pipeline, we follow the same methodology as stated previously using synthetic data. Thus, we created 20 sequences that consists of a single container being loaded with an STS crane. We calculate the accuracy of the different visual tasks for the outcomes of all sequences. As we can see in Table 5, we have solid results in most of the tasks. We are able to perform really well in the ID recognition, the IMDG detection and the Door/NoDoor classification. However, we do not perform that great on damage detection as it leads to lots of false positive due to unseen negative samples, such as when there is no container visible on one of the cameras.

Table 5: Accuracy of the different tasks within the sequence.

ID	*IMDG*	*Damages*	*Doors*
75.0	71.1	33.1	95.0

5 CONCLUSIONS

This study presents a pipeline with state-of-the art algorithms that allows the first step towards automatizing port tasks. We overcome the restrictions imposed by the capture of real tagged images using synthetic data and provide results and baselines for multiple visual container inspection tasks.

By combining multiple tasks within the presented pipeline, we are able to automatize a loading and unloading process of an STS by using deep learning and computer vision techniques. These tasks have been validated using synthetic data. In addition, we proposed an algorithm that tackle the issue of ID recognition by using multiple deep learning algorithms and a custom binary tree.

We also presented a heuristic algorithm that it is a set of rules that are connected logically and are designed to solve the presented issue. We are able to process multiple tasks and extract useful insights from multiple video streams. Therefore, proposing a

pipeline that can be used for automatizing the current STS cranes.

6 FUTURE WORK

Future work includes extending the inspection tasks in order to automatize even further the current manual tasks and increase the performance of the current ones, as some still have room for improvements. Also, we would like to investigate state-of-the art deep learning models using temporal information in order to provide better results to the heuristic algorithm. In addition, we would like to study with an extended research regarding the potential domain gap between synthetic data and real data.

ACKNOWLEDGEMENTS

This work has been partially done under the frame of the project 5GLOGINNOV (Grant agreement ID: 957400) funded by the European Comission under the H2020-ICT-2018-20 programme, within the topic ICT-42-2020 - 5G PPP – 5G core technologies innovation.

REFERENCES

Aranjuelo, N., García, J., Unzueta, L., García, S., Elordi, U., and Otaegui, O. (2021). Building synthetic simulated environments for configuring and training multi-camera systems for surveillance applications. In *VISIGRAPP (5: VISAPP)*, pages 80–91.

Baek, Y., Lee, B., Han, D., Yun, S., and Lee, H. (2019). Character region awareness for text detection. In *CVPR*, pages 9365–9374.

Bahrami, Z., Zhang, R., Rayhana, R., Wang, T., and Liu, Z. (2020). Optimized deep neural network architectures with anchor box optimization for shipping container corrosion inspection. In *2020 IEEE SSCI*, pages 1328–1333. IEEE.

Bu, W., Yan, S., Chen, J., Yang, C., and Liu, C. (2018). Visual recognition of container number with arbitrary orientations based on deep convolutional neural network. In *2018 10th IHMSC*, volume 2, pages 204–207. IEEE.

Cai, Z. and Vasconcelos, N. (2019). Cascade r-cnn: high quality object detection and instance segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1483–1498.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. (2019). MMDetection: Open mmlab detection toolbox and benchmark.

Cortés, A., Rodríguez, C., Vélez, G., Barandiarán, J., and Nieto, M. (2022). Analysis of classifier training on synthetic data for cross-domain datasets. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):190–199.

Delgado, G., Cortés, A., García, S., Aranjuelo, N., Loyo, E., and Berasategi, M. (2022). Seafront: Synthetic dataset for visual container inspection. https://datasets.vicomtech.org/di21-seafront/readme.txt.

Feng, X., Wang, Z., and Liu, T. (2020a). Port container number recognition system based on improved yolo and crnn algorithm. In *2020 AIEA*, pages 72–77. IEEE.

Feng, X. Q., Liu, Q., and Wang, Z. W. (2020b). Port container number detection based on improved east algorithm. In *Journal of Physics: Conference Series*, volume 1651, page 012088. IOP Publishing.

Filom, S., Amiri, A. M., and Razavi, S. (2022). Applications of machine learning methods in port operations–a systematic literature review. *Transportation Research Part E: Logistics and Transportation Review*, 161:102722.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778.

ISO (2022). *ISO 6346:2022 Freight containers — Coding, identification and marking*. International Organization for Standardization.

Jocher, G., Stoken, A., Chaurasia, A., Borovec, J., NanoCode012, TaoXie, Kwon, Y., Michael, K., Changyu, L., Fang, J., V, A., Laughing, tkianai, yxNONG, Skalski, P., Hogan, A., Nadar, J., imyhxy, Mammana, L., AlexWang1900, Fati, C., Montes, D., Hajek, J., Diaconu, L., Minh, M. T., Marc, albinxavi, fatih, oleg, and wanghaoyang0106 (2021). ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support.

Li, X., Liu, Q., Wang, J., and Wu, J. (2020). Container damage identification based on fmask-rcnn. In *International Conference on Neural Computing for Advanced Applications*, pages 12–22. Springer.

Verma, A., Sharma, M., Hebbalaguppe, R., Hassan, E., and Vig, L. (2016). Automatic container code recognition via spatial transformer networks and connected component region proposals. In *2016 15th ICMLA*, pages 728–733. IEEE.

Wang, Z., Gao, J., Zeng, Q., and Sun, Y. (2021). Multitype damage detection of container using cnn based on transfer learning. *Mathematical Problems in Engineering*, 2021.

Zhiming, W., Wuxi, W., and Yuxiang, X. (2019). Automatic container code recognition via faster-rcnn. In *2019 5th ICCAR*, pages 870–874. IEEE.