htw saar Hochschule für Technik und Wirtschaft des Saarlandes University of Applied Sciences

evocortex ROBOT ENGINEERING & SENSOR FUSION

Object Detection System Implementation for an Autonomous Mobile Robot using a Synthetic 2D-RGB Data Generation Approach

Bachelor Thesis

Ivan Eric Díaz Arenas

Degree Course: Mechatronics/Sensor Technology

Examiners:	Dr. -Ing. Kai Haake
	Dr. -Ing Steffen Knapp
Advisor:	M. Eng. Waldemar Haag

Conducted at Evocortex GmbH

Nürnberg, 31.03.2023

Declaration of Authorship

“I affirm that this Bachelor thesis was written by myself without any unauthorized third-party support. All used references and resources are clearly indicated. All quotes and citations are properly referenced. This thesis was never presented in the past in the same or similar form to any examination board. I agree that my thesis may be subject to electronic plagiarism check. For this purpose, an anonymous copy may be distributed and uploaded to servers within and outside the Hochschule für Technik und Wirtschaft des Saarlandes (HTW Saar).”

[illegible]

Ivan Eric Díaz Arenas

Nürnberg, 31.03.23

Table of Contents

List of figures

1 Introduction

An Artificial Neural Network is a massively parallel distributed processor that acquires and stores knowledge from its environment through a learning process [1]. In the context of the supervised learning approach, the learning process, also known as training, adjusts internal parameters of the network by mapping input-output pairs of information within a dataset [2]. By training specialized networks such as Convolutional Neural Networks (CNN), problems that were exclusively solved in the realm of human expertise are now being tackled and even surpassed [3].

Mobile robotics is one of the main fields incorporating CNNs. While mapping techniques based on the geometric representation of the space are enough for the robot to navigate autonomously, high-level tasks such as interacting with objects may require an interpretation of the situation [4]. To deal with this semantic understanding challenge, robots rely on object detection, which is a combined task of localizing an object and classifying it, i.e., assigning it the category it belongs to [5]. Taking a step closer for object detection in real-time applications, Redmon and Ali [6] introduced the *You Only Look Once* (YOLO) algorithm that uses a single CNN to look once at the image. Since then, new state-of-the-art detectors have been developed relying on the YOLO model, including the popular YOLOv5 and the newest YOLOv7, which has currently surpassed all known object detector algorithms [7].

Datasets contain images and labels, i.e., files containing coordinates of bounding boxes enclosing objects and their corresponding class. However, datasets found within the industrial domain do not fulfill every use case and consequently are manually collected and annotated in an expensive and time-consuming process [8]. Therefore, the synthetic-data generation approach represents an alternative to the real-world limited data challenge since it is not only computer-generated but also automatically labelled. The domain gap, however, between synthetic and real-world industrial data poses a limitation to the performance among object detectors [9]. In light of the non-existent consensus of which real-world features should be emphasized more on synthetic data, the Domain Randomization (DR) technique addresses the domain gap by randomizing multiple simulation parameters until the real world becomes another scope of the generated synthetic data [10].

In the context of an industrial environment, this thesis explores various synthetic-data generation strategies based on the Domain Randomization technique and determines the most suitable approach to detect *Small Load Carriers* (SLC). To achieve this goal, the YOLOv5s and the YOLOv7 object detection algorithms are trained and evaluated.

2 Objective

Intralogistics, i.e., the connection and interaction of different logistic functions, has increased in complexity, leading to new challenges within companies [11]. To address these problems, some corporations intend to shift to dynamic and networked autonomous systems, such as Autonomous Mobile Robots (AMR). Through sensor fusion and Simultaneous, Localization and Mapping (SLAM) techniques, AMRs have traditionally increased transportation productivity by avoiding obstacles and navigating freely in its environment [12].

The EvoCarrier, currently being developed by Evocortex GmbH, is among the AMRs that provide an automation solution within the intralogistics field. By integrating new technologies such as the *Evocortex Localization Module*, EvoCarrier offers a millimetric-accurate positioning that helps him play a leading role on the object transportation task [13]. As depicted on *Fig. 2.1*, EvoCarrier integrates a lifting unit that typically fits into a four-wheeled platform called trolley. Among the most common objects loaded are stacks of SLCs.

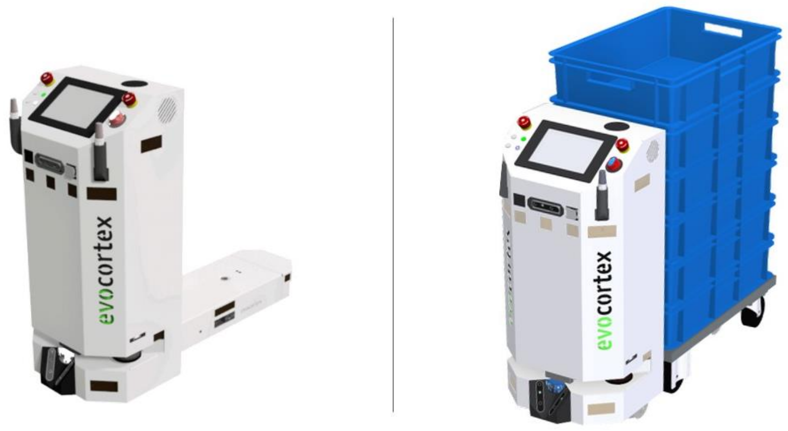


Figure 2.1 EvoCarrier Robot [13]

The SLC is one of the most used object-transportation means within the industrial field; defined as a durable and standardized plastic container, SLCs maximize the use of space in the transportation and storage process [14]. Given that SLCs models differ among the manufacturers, this thesis uses as a basis the RL-KLT 6147 model with a weight of 1.82 kilograms and dimensions of $600x400x147$ $[mm]$ [15]. *Fig. 2.2* shows the SLC.

Figure 2.2 Small Load Carrier (SLC) [16]

EvoCarrier currently lacks the semantic understanding of the environment, specifically information regarding the nature of its load and the role it plays on the scene [17]. To fully comprehend the intricacies of the scene, EvoCarrier should gather several details from an image, for example:

- The trolley may be loaded with objects that do not concern the robot.
- SLCs may have different dimensions and may contain complementary objects attached to them, e.g., labelling cards inserted at one side of them.
- The height of an SLC stack depends on the number of SLCs stacked.
- SLCs and SLC stacks are usually placed on trolleys.

To bridge the gap in the semantic understanding of every scene, a combination of an RGB camera and a real-time object detection algorithm can be used. *Fig. 2.3* depicts on the left side an example faced by EvoCarrier; on the right side, YOLOv7 is localizing and classifying each SLC presented on the scene.

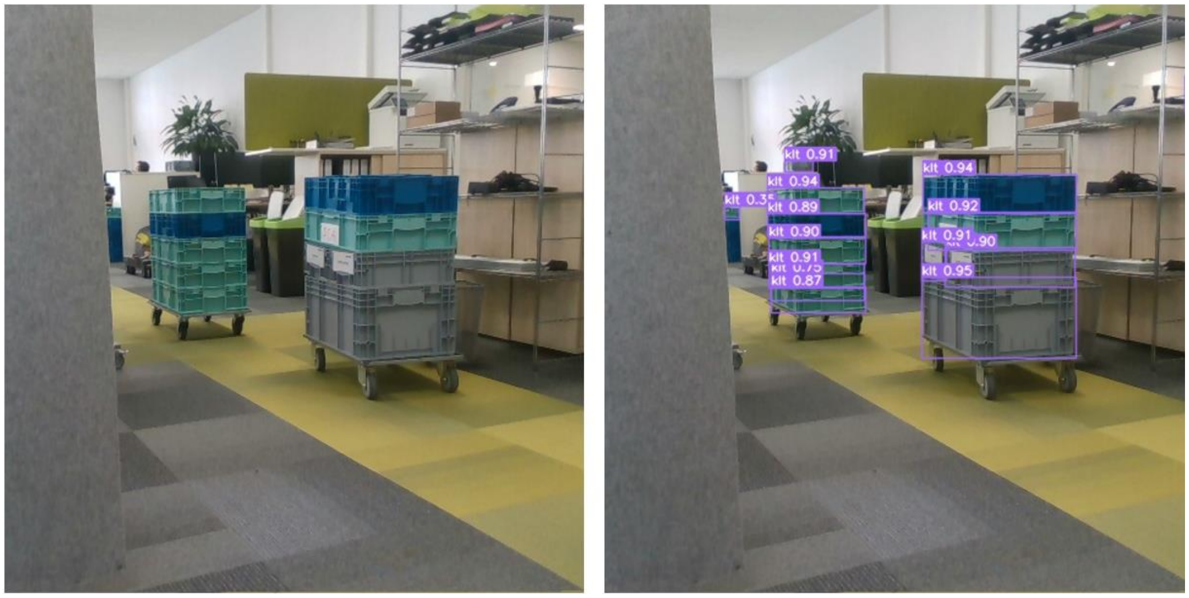


Figure 2.3 Semantic information from the environment.

Based on the previous discussion, the present thesis aims to address two main objectives:

(1) Determine the most appropriate synthetic generation strategy that meets the real-world detection of Small Load Carriers.
(2) Evaluate the YOLOv5s and YOLOv7 object detection algorithms and determine the most suitable for the EvoCarrier use case.

3 Theoretical Background

In this chapter, the foundational knowledge for understanding the thesis is presented. Subchapter 3.1 explains the Supervised Learning. Subchapter 3.2 provides an overview of the Convolutional Neural Network (CNN). Subchapter 3.3 delves into the general object detection task using the YOLO algorithm. Subchapter 3.4 shows the metrics used to evaluate object detection models. Finally, subchapter 3.5 explains in detail the synthetic data.

3.1 Supervised Learning

During the supervised learning approach, an Artificial Neural Network learns a mathematical model that maps input values, e.g., images, and target values, specifically by adjusting learnable parameters that store the learning knowledge. This process is commonly referred as training and it attempts to make the network able to predict target values given unseen images [18]. As explained by [19], the target values, also known as labels, contain information about the object within an image, specifically the coordinates of a bounding box enclosing the object and its corresponding class. This information is usually kept into a training and a validation dataset, where the first one is used to adjust the learnable parameters of the network and the second one serves as a benchmark to validate the model during training. To conclude the process prior deployment, the network is evaluated on a completely unseen environment via a test dataset. A formal description of the training process is explained next.

Training

As described by [20], the training process updates the learnable parameters following a backpropagation method that relies on an objective loss function, which estimates the error between the current prediction and the target value, following an output-to-input direction. Once the gradients are computed, the method employs an optimization algorithm that updates the learnable parameters, called weights, to determine the minimum point of the loss function, also known as global minimum. In the context of supervised learning, optimization algorithms based on the Gradient Descent method are widely used, wherein the weights are updated following the gradient to the local or global minimum of the loss function. To achieve this process, the process typically starts first by setting random values to the weight parameters Θ. Afterwards, the algorithm adjusts per training cycle of parameter Θ to decrease the main objective loss function $J(\Theta)$. *Fig. 3.5* depicts an example graph of the loss function $J(\Theta)$ during the parameter Θ update, where the gradient goes from a high value (red) to a low value (blue). The training process is thus finished when the minimum error is reached.

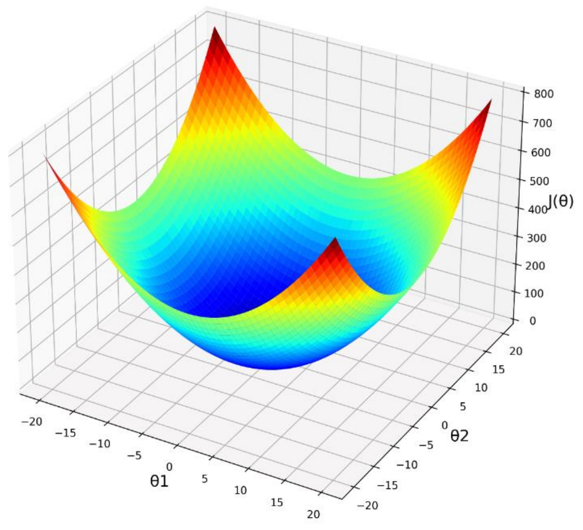


Figure 3.1 Gradient Descent [20]

[21] argues that the seed should be considered as another hyperparameter, i.e., a parameter that directly affects the training process. The seed induces a random parameter initialization, such as weights, and therefore may lead to a consistent performance among training models.

Transfer Learning

Neural networks that follow the supervised learning approach are highly dependable on the amount of data. However, gathering and labelling the training data is time consuming and, in most of the cases, unfeasible [22]. To address this challenge, multiple techniques such as the Transfer Learning (TL) have been followed. [23] demonstrates that TL improves the learning process on a target task by leveraging knowledge from one or more source tasks. TL has mainly three advantages: 1) the initial performance achieved with TL is higher, 2) the amount of time required to learn the target task is less than learning from scratch and 3) the final performance is higher with the TL knowledge. On the scope of this thesis, the transfer learning technique is followed via the Common Objects in Context (COCO) dataset, which contains more than 200,000 labeled images with objects in everyday scenes [24].

3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are defined as a specialized Artificial Neural Network that processes grid-pattern data, such as images, to search for specific features. By first extracting simple features, such as edges or corners, CNNs gather enough information to enable high-order feature extractors, which are responsible of gaining an overall receptive field of the image [25]. Mainly composed of convolutional, pooling and fully connected layers, a typical CNN architecture is depicted on *Fig. 3.1*.

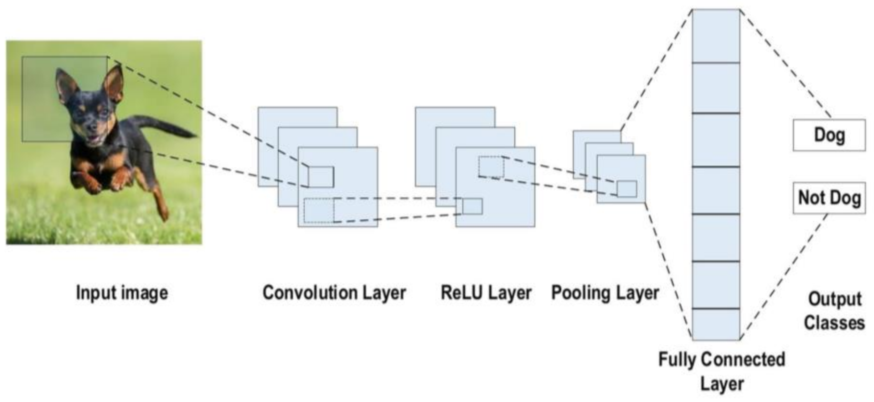


Figure 3.2 Architecture of a typical CNN on the classification task [22]

Convolutional layer

As noted by [26], the convolution operation describes an element-wise product and summation between an input tensor and a kernel*.* The kernel, which is defined as an array of discrete numbers called kernel weights, is constantly adjusted during the training process. The aim of the whole layer is to extract specific information of the input tensor in a process called feature extraction; as a result, an output vector called feature map is computed. *Fig. 3.2* shows the generation of a feature map (right) by applying an outline-extractor kernel (middle) to an input tensor (left).

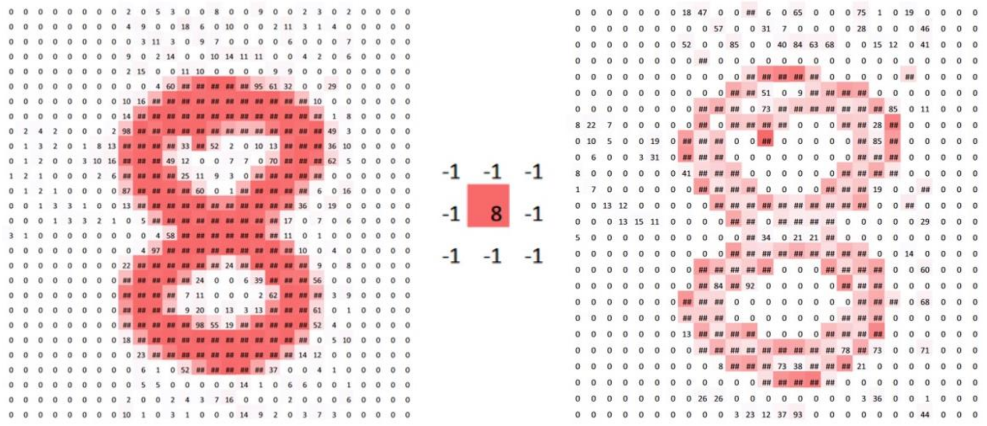


Figure 3.3 Convolution operation. [26]

Nonlinear Activation Function

According to [18], a nonlinear activation function provides a nonlinearity nature to CNNs by regulating the output of a neuron. *Fig 3.3* explains the difference between the linear and nonlinear behavior using a binary-classification example. To simplify, each point on the diagram would represent an image that has a label $y_i \in \{0,1\}$, represented by a color. The task is thus to classify each image by creating two different class regions denoted by the same colors. Given that the presented classes are not linearly separable (left), the nonlinearity behavior is introduced as a solution to enclose correctly the red-class images from the blue ones (right).

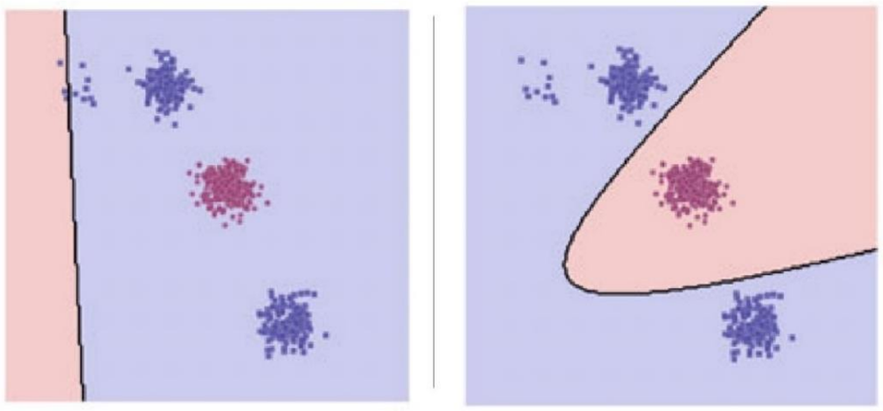


Figure 3.4 Linear and non-linear behavior at the classification task [18]

Pooling Layer

The pooling layer is commonly used for down sampling the in-plane dimensions of the feature maps. The most used types of pooling are shown on *Fig. 3.4*. The Max Pooling extracts patches from the feature maps and outputs the maximum value on that patch by disregarding the other values. The Global Average *Pooling* is an extreme type of Average Pooling where a feature map is reduced to a 1x1 array by taking the average value of all elements inside the feature map [22].

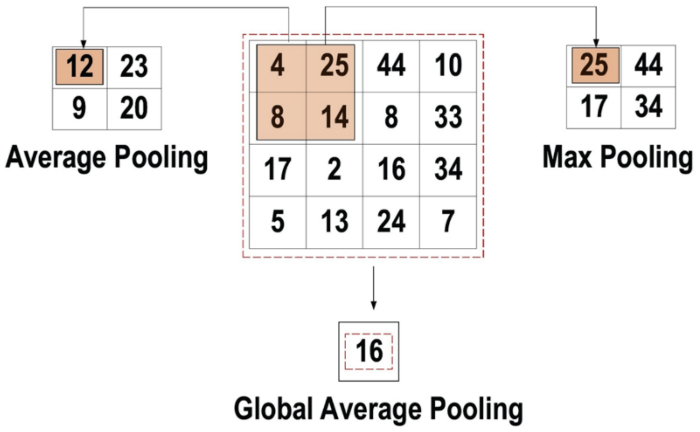


Figure 3.5 Average, Max and Global Average [22]

Fully Connected Layers

As described by [26], feature maps are usually transformed into a one-dimensional vector after the last convolution or pooling layer. The down-sampled features contained on the vector are then mapped via synaptic weights through fully connected layers to compute a final output tensor. These layers are usually linked to a special nonlinear activation function that normalizes the output. Unlike the kernels on the convolutional layer that are shared among the input tensor, these layers allocate synaptic weights between neurons and therefore the connections between adjacent layers are increased. *Fig. 3.6* shows an example of an CNN with the task of classifying numbers, wherein the fully connected layer is located at the end.

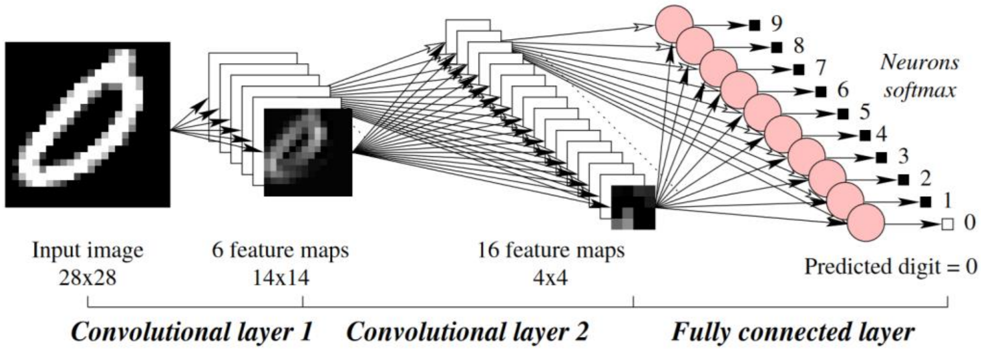


Figure 3.6 Fully Connected Layers [27]

3.3 Object Detection

While localization task attempts to assign on the image the coordinates of the object and its extent, the classification task assigns the class belonging an object. The combination of these two tasks is called object detection, and it may be resolved either a regression or a classification. The main difference, however, is the type of target values, i.e., regression employs continuous real numbers and classification uses discrete numbers [5]. The object detection problem was first addressed repurposing classifiers to detect objects, e.g., the sliding window method that run a classifier at evenly spaced locations over the entire image [28]. To solve the challenges faced by traditional methods, the region-based concept was introduced on CNNs by [29] and its R-CNN, where a reduced number of regions are proposed to shorten the number of classifications. As described by [5], despite the improvement on the classifier approach, redefining the object detection task as a regression problem provides clear advantages. For instance, detectors employing a combination of the latter and the region-based approach of the classifier approach, have made possible the introduction of one-step frameworks that can map straightly the image pixels into bounding boxes and class probabilities, which reduce time expense. One of these frameworks is called You Only Look Once and is explained as follows.

You Only Look Once (YOLO)

Introduced by [28], the YOLO algorithm reframed the object detection task as a regression problem by using a single neural network to look once at the input image before predicting bounding boxes. The algorithm starts by dividing the input image into a $S \times S$ grid, where each grid cell becomes responsible for predicting via a bounding box the object whose center falls on it. Moreover, the bounding box prediction is composed of six elements: (x, y) coordinates of the object's center, width and height dimension of the bounding box enclosing the object, a confidence score, and a class probability. *Fig. 3.6* depicts the detection procedure given an input image.

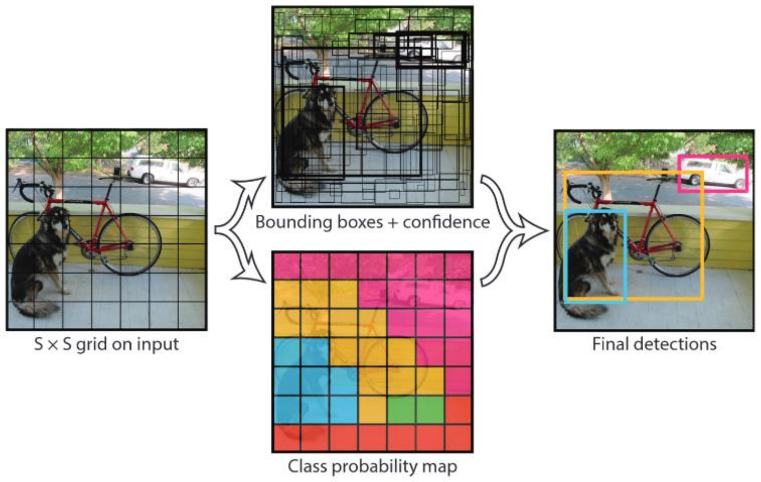


Figure 3.7 Object detection done by YOLO [28]

After [6] made improvements to the original algorithm, newer YOLO versions took the Faster R-CNN network approach of predicting anchor boxes and confidence scores instead of directly predicting bounding boxes using FC layers. Several anchor boxes, defined as pre-defined bounding boxes, are predicted per grid cell and during training are adjusted to the actual object to generate the final detection.

Fig. 3.7 shows the bounding box prediction based on an anchor box. According to [30], the network computes four predictions t_x, t_y, t_w, t_h and then computes relative-to-the-grid bounding boxes. The next equations define the actual bounding box dimensions with respect to the image dimensions, where each grid cell has an offset of (c_x, c_y), and (p_w, p_h) represent the width and height of the anchor box.

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

Three aspects should be noted: the sigmoid function $\sigma(\cdot)$ constrains the relativity to the grid cell, (b_x, b_y) represent the center coordinates, and (b_w, b_h) the width and height of the predicted bounding box.

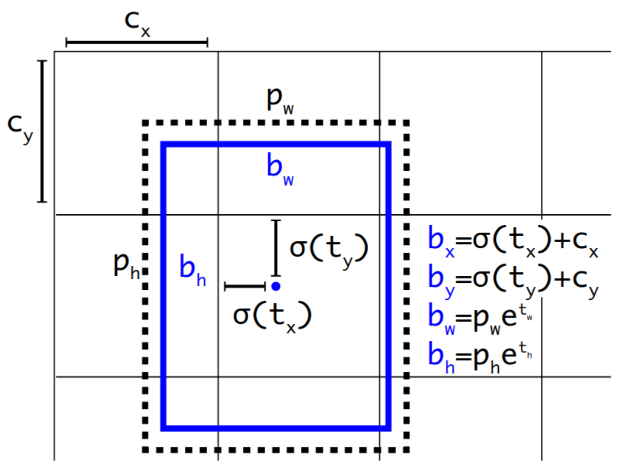


Figure 3.8 Bounding Box prediction [6]

In order to train a YOLO-based model, the hyperparameters, which describe the parameters that directly affect the training process [31], must be defined. *Table 3.1* defines the most important hyperparameters.

Table 3.1 Hyperparameters

Hyperparameter	Definition
Epochs	It defines the number of times the complete training data is shown to YOLO.
Batch size	Number of images propagated at the same time by YOLO.
Class categories	Object categories to train. On this thesis just one class is detected: small load carrier.
Image	Width and height of the input image.
weights_file.pt	Defines the architecture and weights of the model being trained.

3.4 Evaluation Metrics

This subchapter presents the necessary information to understand how to evaluate the performance of the network.

Intersection Over Union and Confidence Threshold

Intersection Over Union (IoU) shows the overlap of two bounding boxes, specifically between the ground-truth label and the predicted bounding box. It is formally described as $IoU = \frac{Intersection}{Union}$. The confidence threshold, on the other hand, sets the minimum value accepted of the IoU to consider a prediction as positive.

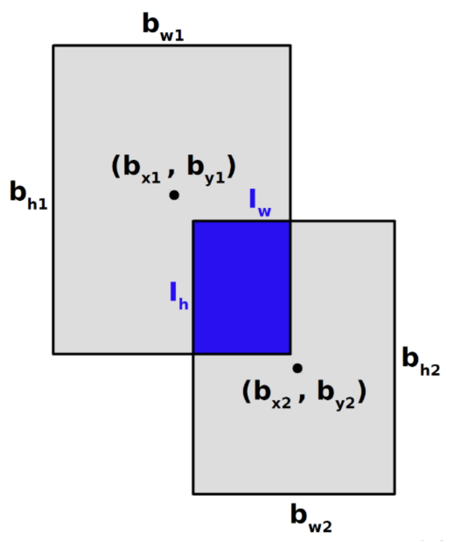


Figure 3.9 Intersection Over Union [32]

Confusion Matrix

Considering the ground-truth label and the prediction done by the classifier, four possible outcomes are possible and are denoted on *Fig. 3.8*.

	Actual 1	Actual 0
Predicted 1	True Positive	False Positive
Predicted 0	False Negative	True Negative

Figure 3.10 Confusion Matrix

The figure is explained as follows:

i) True Positive (TP): the existence of an object is predicted, and it does exist.
ii) False Positive (FP): the classifier predicts the existence of an object that does not exist.
iii) True Negative (TN): the classifier predicts the non-existence of an object and that does certainly not exist.
iv) False Negative (FN): the classifier predicts the non-existence of an object, but it does exist.

Precision, Recall and F1-Score

Recall computes the proportion of positive patterns that are predicted correctly; thus, in absence of FNs, a model estimates a recall of 1.0. On the other hand, Precision estimates the proportion of positive classes from all positive predicted classes, so a model that does not predict FPs has a precision of 1.0. Finally, the F1-score (F1) denotes the harmonic mean between precision and recall. In other words, it is the optimum point at the recall-precision tradeoff [33].

$$Recall = \frac{TP}{TP+FN}$$

$$Precision = \frac{TP}{TP+FP}$$

$$F1 = \frac{2 \times precision \times recalll}{precision + recall}$$

YOLO Graphs

YOLO computes its output graphs against the current epoch and follows the next procedure. First, different confidence thresholds are used to compute the TP, FP, TN, FN alongside with its respective Recall and Precision values. Second, the *F1-score vs different confidence thresholds* is calculated as depicted on the next figure.

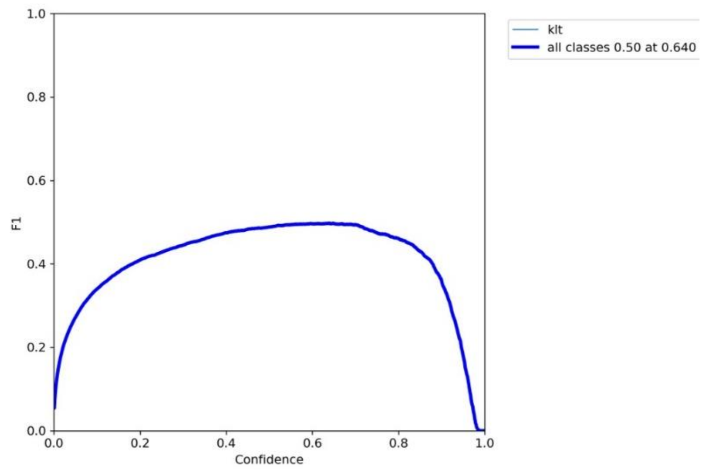


Figure 3.11 F1-score vs confidence threshold

Third, YOLO takes the optimal confidence threshold that maximizes the F1-score, which in the example refers to 0.640. Finally, the precision and the recall with the chosen confidence threshold is computed and plotted on their respective graphs as the corresponding epoch. The output graphs from YOLO are shown on *Fig 3.11.* Although YOLO does not display the x-axis, it should be clear that this axis represents the training epochs.

Average Precision

The Average Precision (AP) is defined as the area under the $Precision \times Recall$ curve [34]. On *Fig 3.11*, YOLO uses the notation @0.5 to indicate an interpolation done by a confidence threshold of 0.5, i.e., the precision at $recall = 0.5$ substitutes any other precisions values computed at $recall > 0.5$. Given that the current thesis only considers one class, mean Average Precision (mAP) is the same as the AP.

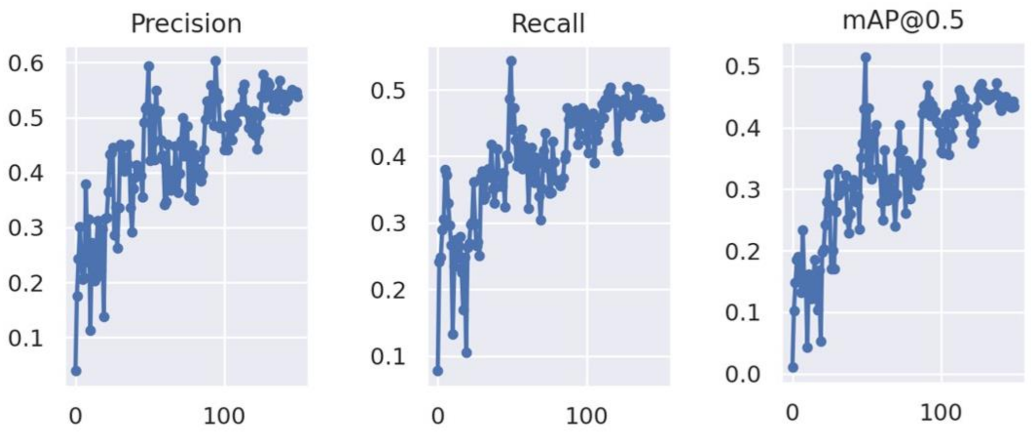


Figure 3.12 Precision, Recall and mAP@0.5 against Epochs

3.5 Synthetic Data

Real-world datasets face a data shortage challenge mainly due to scalability issues and privacy concerns [35]. On top of that, real-world data is manually collected and labelled on a human-driven process that eventually leads to low-quality labels and errors among the datasets [36]. For this reason, synthetic data attempts to address these challenges using a computer-based generation approach that is faster, cheaper, and more accurate [37]. The following subtopics explain the image synthesis and the general challenges faced by synthetic data.

Image synthesis

According to [38], the initial phase on image synthesis is modelling, in which the virtual scene, 3D object models and additional simulation parameters are defined. Essentially made up of a mesh-like structure containing clusters of triangles with shared vertices, an object model enhances its visual appearance by the application of an image texture onto its surface. To achieve this task, [39] introduces the UV mapping process that flattens the 3D object model into a 2D representation, allowing a mathematical mapping of pixels between the image texture and the surface of the model. *Fig. 3.12* shows the result (right) of the UV mapping process (middle) applied to raw 3D object (left).

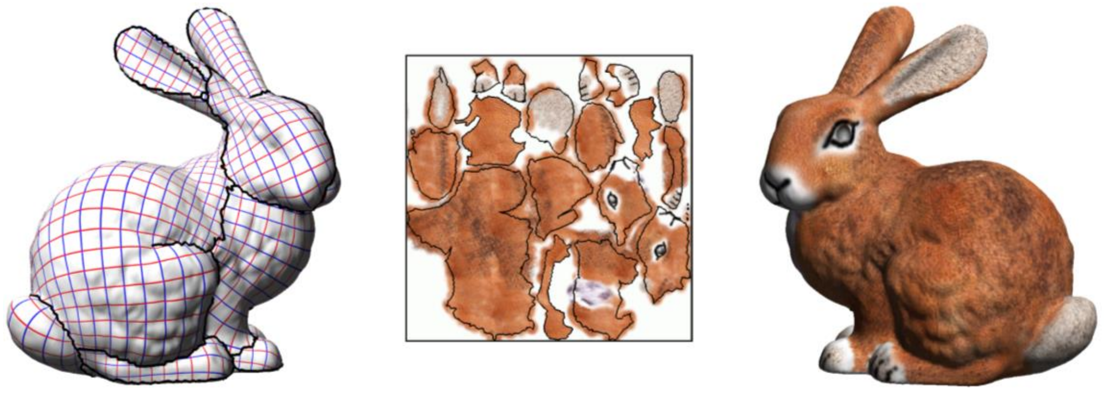


Figure 3.13 Texture added to a model [40]

To complete the image synthesis, the rendering process simulates the light behavior from a defined viewpoint and generates an image out of the 3D scene [38]. One of the main rendering methods is the Photorealistic Rendering explained by [41], where the synthetized image appears indistinguishable from a real-world scene. This approach is based on the ray-tracing algorithm, which essentially follows the path of a ray light in an environment as it strikes and bounces off an object. This ray traces, usually coming from a simulated camera, determine how is the scene viewed and therefore synthetized.

Procedural Modelling and Rendering

[42] states that the increasing complexity of the level of detail in a scene makes the manual modelling unfeasible; for this reason, procedural modelling addresses this challenge by executing algorithms that automate the scene generation. Introduced by [43] as a fully configurable procedural pipeline for generating and rendering photorealistic scenes, BlenderProc is presented as a tool that runs on top of Blender, an open-source 3D creation suite. By automating the generation process via python scripting, images, also known as frames, achieve simulated complex behaviors usually done by a digital artist, such as sampling the pose among objects, texture modification, mesh deformation, and, most importantly, rendering and generating pixel-perfect bounding box labels.

Reality Gap

[44] illustrates that neural networks trained on synthetic data suffer from a domain gap challenge, in which the network's performance is dropped while tested on real-world data. Explained as the inability to realistically simulate the real-world, the reality gap of the synthetic data is bridged by the Domain Randomization (DR) technique [45]. DR attempts to increase the generalization capabilities of the network by employing highly randomized data, allowing the network to interpret reality as just an additional instance of the synthetic data [46]. In order to implement this technique, a randomization space is initially established, containing predefined randomizations parameters and their corresponding range of possible values. By being sampled within the boundaries of the interval, the randomization parameters generate a new simulated environment per rendered frame [10].

4 Method

The current chapter gives a methodological procedure to establish the relationship between training synthetic data, specifically based on a randomization space, and the performance of an object detection algorithm. Hence, Subchapter 4.1 defines the simulation scene and the randomization parameters, and Subchapter 4.2 presents the procedure to train two versions of the YOLO algorithm: YOLOv5s and YOLOv7.

4.1 Synthetic Data Generation

This subchapter presents eleven different generation strategies based on the Domain Randomization technique. Each of these approaches focuses on a specific randomization parameter within the overall randomization space to synthetize a dataset of 500 images. To establish a baseline behavior across datasets, the isolated randomization parameter is applied on top of a control group, which is created by excluding the randomization space.

4.1.1 Control Group

The control group stays as the most-basic dataset presented on this thesis, since it completely discards the randomization space. However, the current thesis works under the assumption that synthetic data with a null variation leads to a poor performance and, therefore, the control group addresses the challenge of creating variations on the data without affecting enough to be considered as part of a bigger randomization space. The key to achieve the desired baseline behavior lies on the enumeration of the frame being rendered. With this enumeration, BlenderProc is able to sample in a controlled way the next benchmarks: camera and light pose, light power, and sampling of the rendered background.

Procedural Modelling and Rendering

The procedural modelling within the world scene is based not only on the frame enumeration, but also on the spherical polar coordinate system depicted on *Fig. 4.1*, where r is the distance from the origin to point P, θ is the colatitude and ϕ represents the longitude [47]. It should be noted that the current thesis does not delve into the effect of the light positioning and thus employs a single point P to represent both the camera and the light. Furthermore, the procedural modelling pipeline attempts to replicate the real-world perspective experienced by the EvoCarrier by locating the SLCs on the $x - y$ ground plane and positioning the point P via a three-step cycle. During the first two-thirds of the cycle, point P is localized from above the ground using a colatitude $\theta \in [30°, 90°]$, ensuring that the synthetized images present the SLC from above the ground; during the last third, variability on the data is aimed to present a slight view from below the ground plane using a $\theta \in [90, -105]$. Finally, BlenderProc completes the positioning of point **P** by computing a rotation matrix based on a forward vector pointing to the center of the scene.

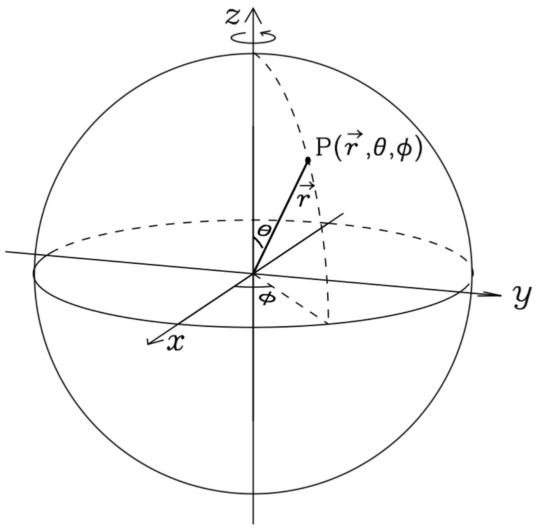
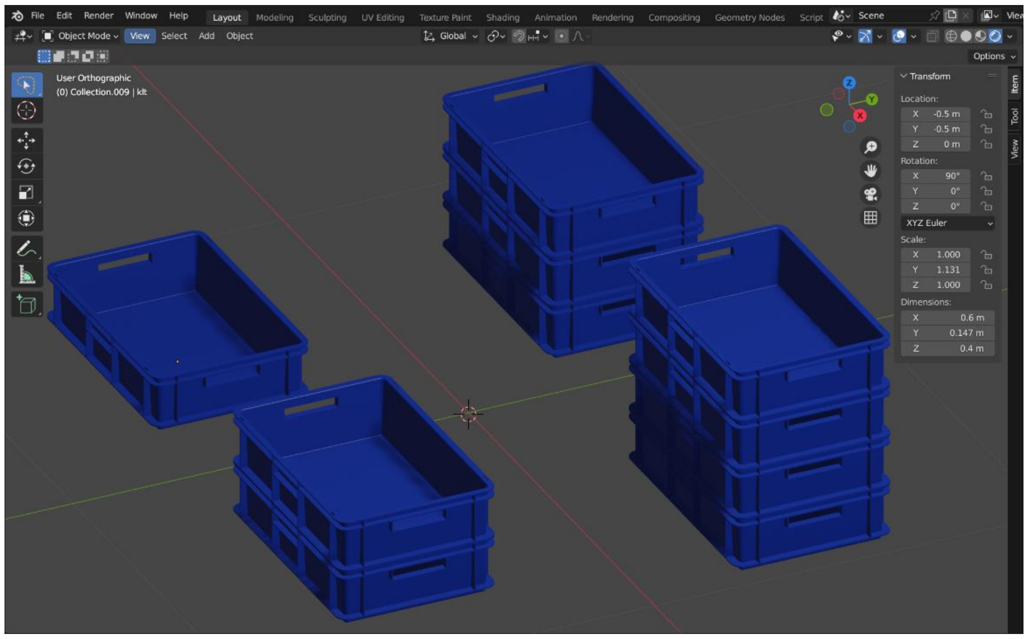


Figure 4.1 Spherical Polar Coordinate System [47]

Before the procedural modelling starts, the SLC model is added to a basic scene. The RL6147 SLC 3D model was asked to the retailer, and it was not given; for this reason, a simplified version of the SLC is imported into the scene and a blue color material is assigned to them. Based on the HSV color model, the blue material is given by: $[H: 0.655, S: 0.988, V: 0.448, Alpha: 1]$. Afterwards, the 3D model is manually copied and moved to form three stacks with two, three and four SLCs. *Fig. 4.2* depicts the piles positioning with an offset from the origin of the scene of $(\pm 0.5, \pm 0.5, 0.0)$.



Transform
Location:
X -0.5 m
Y -0.5 m
Z 0 m
Rotation:
X 90°
Y 0°
Z 0°
XYZ Euler
Scale:
X 1.000
Y 1.131
Z 1.000
Dimensions:
X 0.6 m
Y 0.147 m
Z 0.4 m

Figure 4.2 World Scene with small load carriers

The world scene contains no simulated floor or 3D scene enclosing the SLCs and hence renders a gray solid color on the background of each synthetized image. The COCO dataset is filtered out to address this challenge, wherein a subset of 500 images with a $640x640$ pixel resolution is gathered. The background is thus sampled from this subset based on the frame enumeration. Given that the simulated camera resolution is aimed to match the resolution of the background, the intrinsic parameters of the simulated camera are taken from the Intel RealSense Depth D435 [48] at an image resolution of $640x640\ [pixels]$.

The illumination is done via a single point light, which is defined as a simulated light source that emits the same amount of light in all directions [41]. This point light depicts among the rendered frames a constant white color defined on the RGB color scale as $[255,255,255]$. Its power, however, depends entirely on the enumeration of the rendered frame:

$$power(n) = power_{n-1} + \left(\frac{1000}{500} \times n\right), n \in [1,500],$$

where $power(n)$ is the light power on a particular frame n, $power_{n-1}$ is the power computed at the previous frame and $\frac{1000}{500}$ is the ratio between a maximum light power of $1000\ [watts]$ and the maximum frames to render $n = 500$. Fig. 4.3 shows four example images rendered with the control group setting.

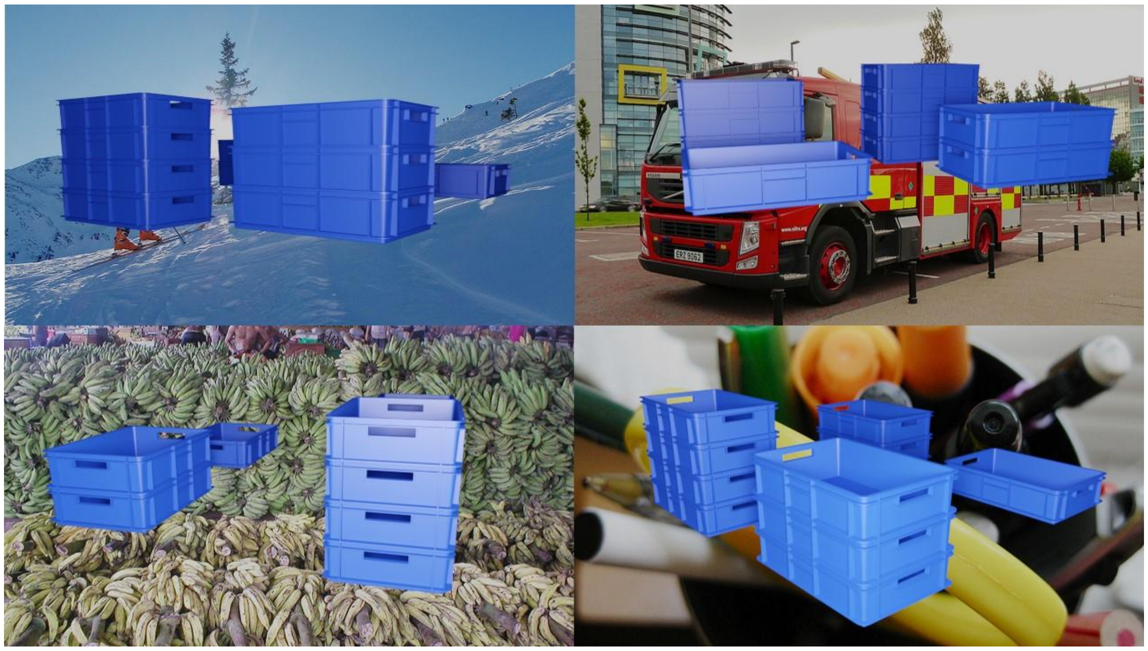


Figure 4.3 Control Group

4.1.2 Randomization Space

The thesis introduces a randomization space of ten randomization parameters, each with their respective sampling interval. While it is aimed to analyze the effect of isolating each randomization parameter, it should be clear that BlenderProc takes first the control group setting and afterwards targets the specific randomization parameter. Given this situation, the in-depth explanation of each randomization parameter is made without mentioning which control group's parameters were assumed. *Table 4.1* provides a brief summary of the randomization space.

Table 4.1. Randomization Space

Randomization Parameter	Description
High Dynamic Range Image (HDRI)	The background is randomly sampled though a set of HDRIs.
Dust	Dust is simulated as being settled on the SLCs.
Distractors	Five cube distractors are modelled and randomly placed on the scene.
Light Color	The light color is randomly sampled among the whole RGB color spectrum.
Depth of Field	The Depth of Field of the simulated camera is randomly modified based on the focus point and the aperture of the camera.
Noise	Noise templates are randomly chosen and overlapped to the camera's view.
Material	The material of each SLC is randomly modified.
Mesh Deformation	The mesh of each SLC suffers a deformation.
Location	The location of each SLC is randomly modified.
Scale	Each SLC suffers a scale modification on each dimension.

HDRI

Hodan et al. [49] rendered complete scenes with realistic lightning and materials, attempting to bridge the domain gap challenge. Despite their promising results, the computational power available for this thesis made unfeasible to replicate entirely their highly photorealistic image method. Consequently, a simplified version is presented, which focuses specifically on the use of High Dynamic Range Images (HDRI). According to [50], HDRIs overcome the challenge of simulating the exact lightning conditions of the real world by employing an image-based lightning method. To generate the HDRI, the lens captures a 360° real environment and additionally stores its respective light conditions. For this reason, the Poly Haven public 3D asset library [51] was used to gather a set of 556 HDRIs, which contain a 4K resolution image of a real-world scene and its light information. As a final comment, this randomization parameter is the only one presented on this thesis that discards the employment of the default point light described on the basic scene. Fig. 4.4 depicts one example image using the HDRI randomization parameter.

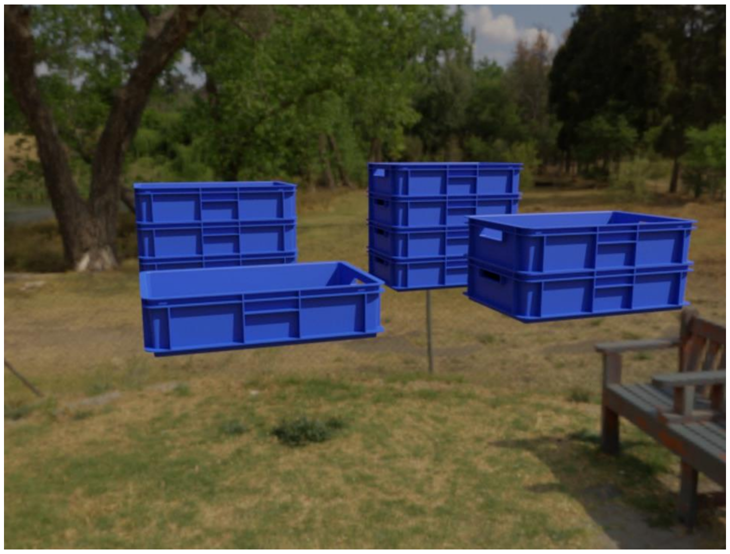


Figure 4.4 HDRI

Dust

No studies regarding the dust factor as a randomization parameter were found; however, the typical industrial environment presents external factors that may affect the appearance of the SLC, including the dust factor. For this reason, this dataset simulates the existence of dust within the scene by adding the particles to the objects via UV mapping. After multiple tests, the sampling interval of this randomization parameter was increased to non-realistic levels; for instance, the particle diameter and the predominance of dust in the scene were highly increased that it resembles a dusty environment. It should be noted that the dust particles were only simulated within the SLC and no attempt of dust particles in the air was aimed. A rendered image using BlenderProc is shown in Fig. 4.5.

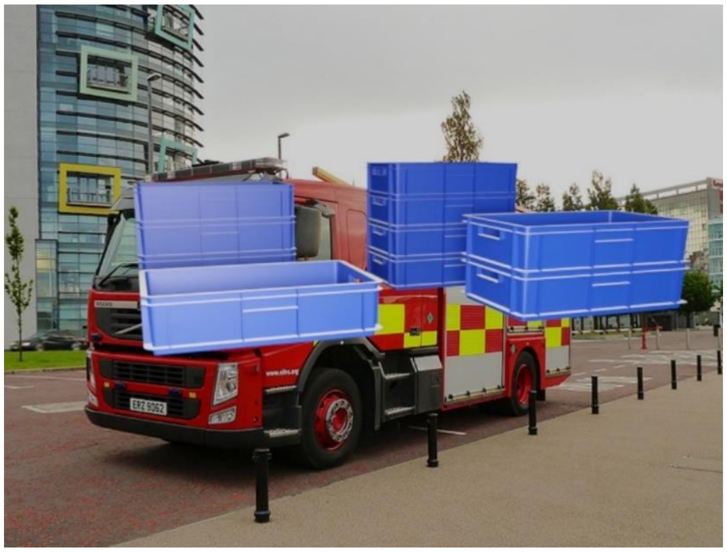


Figure 4.5 Dust

Distractors

Eversberg and Lambrecht [37] conducted an analysis on the impact of adding foreground objects as distractors in the scene, such as simple cubes and complex 3D object models. Their findings, however, did not show any significant benefits in rendering complex 3D object models when compared to simple cubes. In line with these findings, the current dataset explores the incorporation of five cubes in the scene, wherein the pose of the cubes is randomly sampled per frame. The sampling interval, nevertheless, does not consider the collision with the SLC mesh, and at times the cubes occlude or even invade the mesh. Finally, these cubes were selected from Blender's primitive mesh shapes with a default configuration and a dimension of $[0.1, 0.1, 0.1]$ meters. Fig. 4.6 depicts an example result.

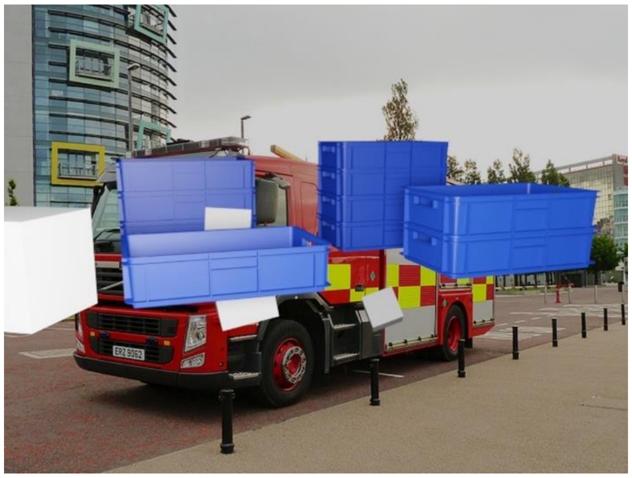


Figure 4.6 Distractors

Light Color

Building upon the approach proposed by Hinterstoisser et al. [46], this dataset aims to reaffirm that, despite being a straightforward and simple operation, sampling the light color improves the overall object detection performance. As a result, a slight modification to the control group behavior is presented, in which the position and power of the light is controlled, but the color of the light is randomly sampled over the entire color scale. It is worth mentioning that the sampled color only modifies the appearance of the SLC in the scope of its color hue. Fig. 4.7 illustrates this randomization parameter.

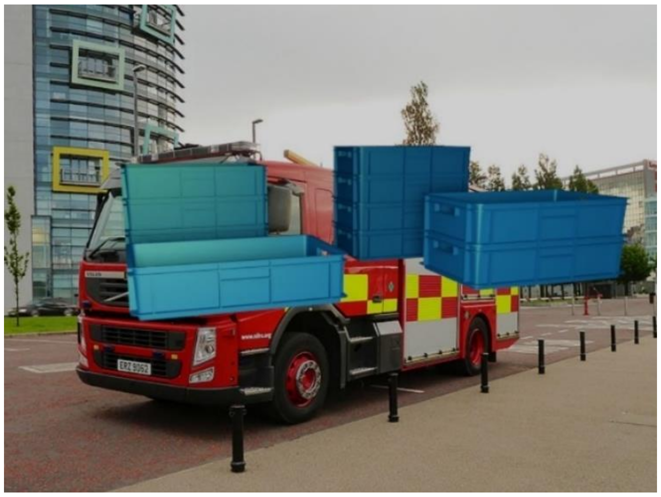


Figure 4.7 Light Color

Depth of Field

This approach is inspired by the work presented by Mayer et al. [9], wherein the lens distortion and blur among images is analyzed. According to their results, the network learns that the object boundaries may be blurry, which cannot be learned from images where crispy contours among the objects are presented. A simplified version of their approach is followed by randomly adjusting the Depth of Field, which is described as the area where the object appears to be in focus [52]. To achieve this task, BlenderProc assigns the focus point to the center of a randomly selected SLC and, afterwards, it randomly samples the aperture of the simulated camera lens between a value of $[0.2,1]$. Fig. 4.8 shows an example of the Depth of Field adjustment.

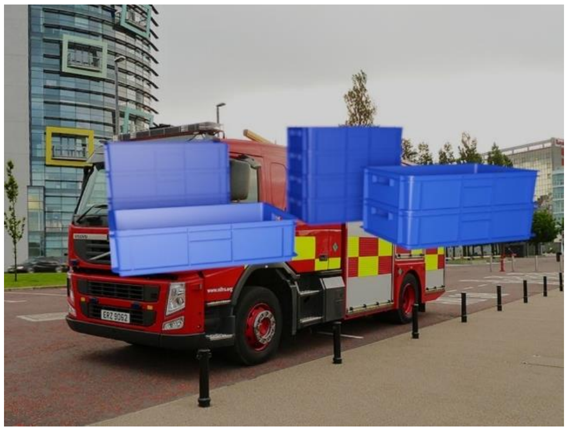


Figure 4.8 Depth of Field

Random Noise

Taking cues from [46], this dataset aims to allow the network to focus more on the key features of the SLC and not really on the basic geometric information. In pursuit of this objective, BlenderProc randomly overlaps a noise filter image onto the frame being rendered. In total, thirteen different noise templates were created from the noise filter options of the GIMP image editor, wherein different colors and random seeds, i.e., the random behavior of the filter, were used. Fig. 4.9 illustrates the result.

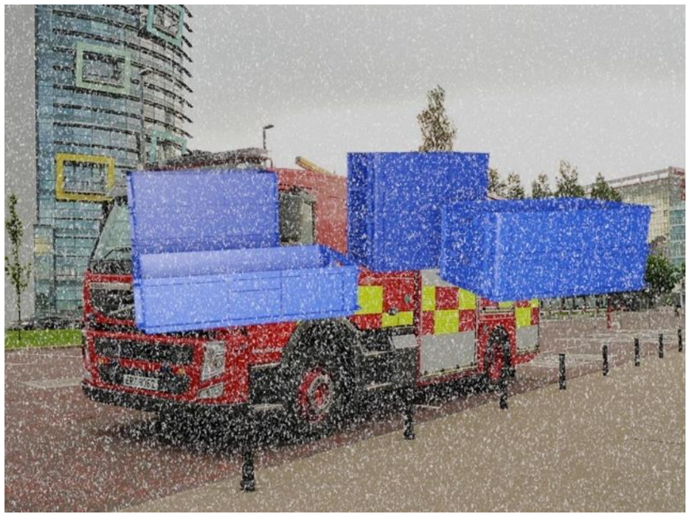


Figure 4.9 Noise

Mesh Deformation

Prior to the thesis process, several studies conducted at Evocortex GmbH generated heatmaps from a trained YOLO-based model to indicate the features of the SLC that received the most attention during object classification. As a result of this analysis, it was found that the YOLO-based model tended to give significant attention to edges and vertical lines within the SLC, which explained the occurrence of False Positives among cube-like real-world objects. Considering these findings, the present dataset explores the idea of training the network via a deformed-mesh SLC to allow YOLO to prioritize the overall SLC rather than a few specific characteristics. This mesh deformation is made via a displace modifier, wherein Blender [53] displaces the vertices of a mesh based on the intensity of an applied procedural texture, i.e., a texture mathematically generated. In this dataset, a cloud-like procedural texture is being applied via UV-mapping and then assigns a strength to it. The strength of the modifier was fine-tuned for finding the optimum point where the SLC is deformed but still seems a SLC. Fig. 10 depicts the mesh deformation.

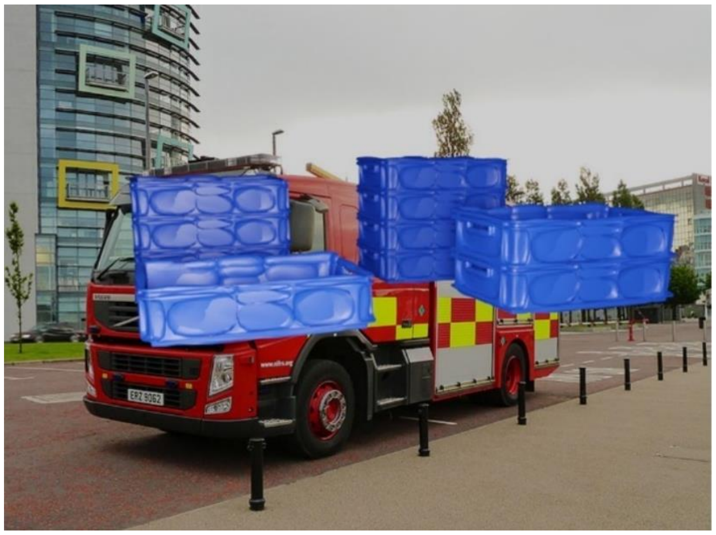


Figure 4.10 Mesh Deformation

Materials

Movshovitz-Attias et al. [54] determined the effect of the quality of the rendering process, specifically via the material and lightning factors. Their results suggest that aiming for a photorealism level via complex material and lightning reduces the error function among a neural network. Inspired by this approach, this randomization parameter explores the idea of randomly sampling realistic materials on SLCs from a pool of 290 image textures, each gathered from Poly Haven [51] with a 4k pixel resolution. This material assignation is done via the UV-mapping of each SLC and the texture of each SLC is randomly sampled per rendered frame. Fig. 11 illustrates an example image of this dataset.

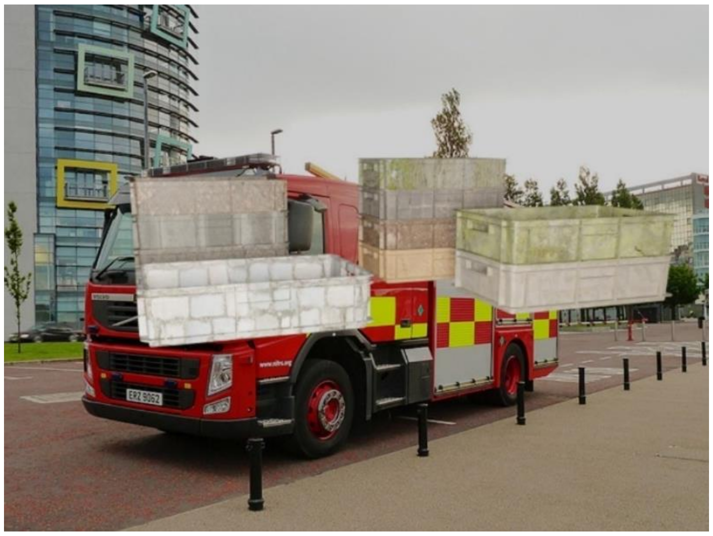


Figure 4.11 Material

Location

Fischer et al. [55] intended to analyze the use of CNN on a specific supervised learning task; however, the data shortage challenge led them to produce a synthetic dataset called Flying Chairs. This dataset consists of 3D chair object models that are transformed in various ways, resulting in chairs that appear to be suspended in the air and in non-common positions. Although present dataset initially attempted to follow the basic concept of the Flying Chairs method, it was found that a random rotation allowed the SLCs to be rendered in poses that the EvoCarrier would hardly experience. Therefore, the approach was modified to only sample the location of each SLC, with the base of the SLC being parallel to the x-y plane ground. More specifically, the location of each SLC is randomly sampled on each dimension from a range of [-0.4, 0.4] and, since no collisions among the SLCs are monitored, SLCs may occasionally overlap between each other occluding a percentage of one another. Fig. 4.12 depicts an example of the location sampling.

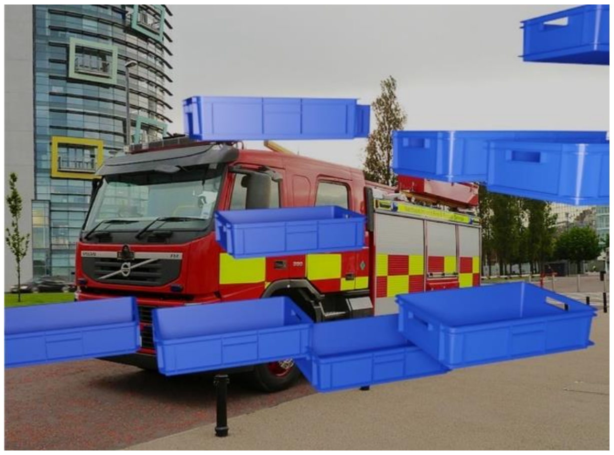


Figure 4.12 Location

Scale

[56] conducted experiments to investigate the effect of modifying the object size on the training synthetic data, in which they concluded that the performance dropped significantly as the object size decreased. In agreement with the previous study, this dataset proposes an extension of the object size approach by presenting a scale randomization parameter. This scale modification intends to make the network learn the fact that SLCs may come in different shape proportions. The process is made by randomly modifying the dimensions of the SLC between a scale range of [0.8, 1.1]. Fig. 4.13 shows the scale modification on each SLC.

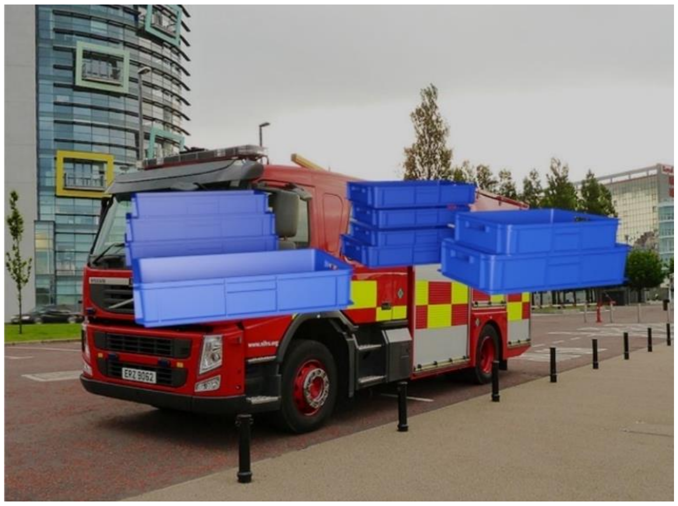


Figure 4.13 Scale

5 Results and Discussion

This chapter explains in-depth the results of training upon two YOLO-based object detection models. Subchapter 5.1 presents the training and evaluation of YOLOv5s and Subchapter 5.2 the results regarding the YOLOv7.

5.1 YOLOv5s

In this subchapter, the results of training a YOLOv5s model on synthetic datasets are presented. The model was trained using the next training hyperparameters: a batch size of 16, a pretrained yolov5s weight file, 100 training epochs, image size of 640 pixels, and a specific random seed per training. With this configuration, the approximate training time of YOLOv5s is 23 minutes.

Validation Dataset

Fig. 5.1 illustrates the distribution of the mAP@0.5 metric across all images of the validation dataset. The HDRI randomization, which has the highest upper quartile of all datasets, shows a wider interquartile range than others, suggesting that the detection performance is not consistent unlike the Distractors and the Light Color datasets. Furthermore, the Light Color presents a higher mean value, presented as an x on the graph, than the one presented on the HDRI. Moreover, the Distractors dataset's upper quartile lies into the 25th percentile of the HDRI and on the 75th percentile of the control group. These three datasets above performed better than the control group that, similarly to the HDRI, show a wider interquartile range.

The graph illustrates that YOLO encountered greater difficulties with the remaining datasets, as evidenced by their median performance being lower than that of the control group. For example, although the Mesh Deformation and the Material datasets have their upper 25th percentile falling within the second quartile of the control group, the lower quartile of both datasets lie on or below the minimum value of the control group. The same situation occurs with the Dust and Depth of Field datasets, whose upper quartile do not reach the first quartile of the control group. The remaining datasets, i.e., Noise, Scale and Location, perform below the expected behavior, where the lowest mAP@0.5 is reached by the Location.

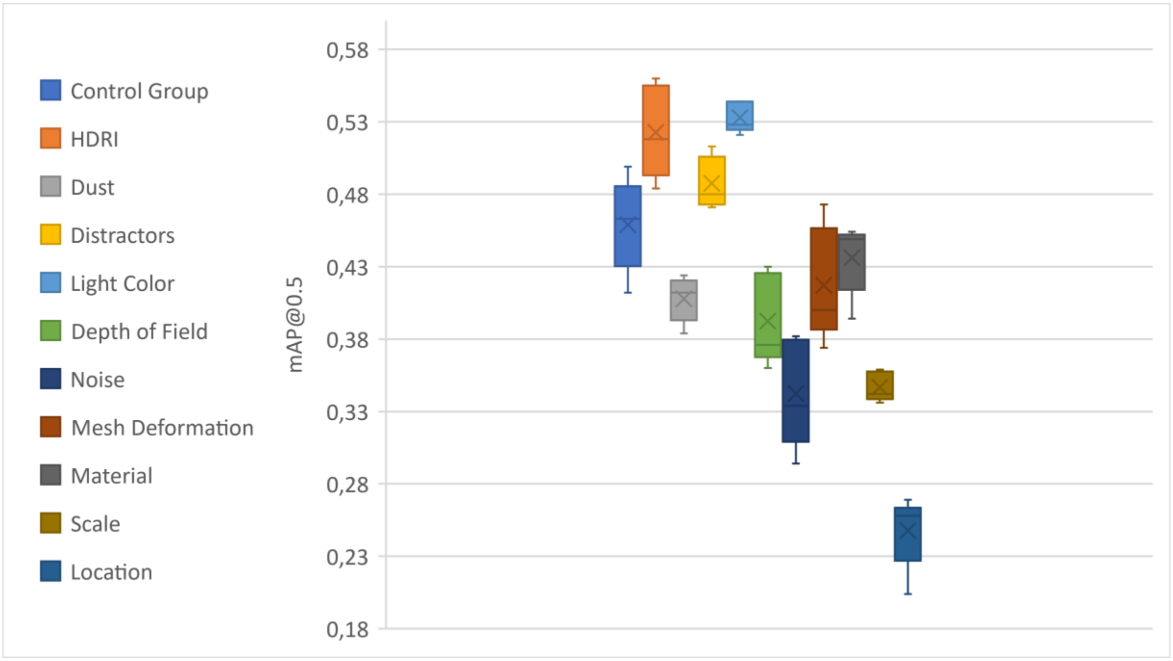


Figure 5.1 Box plot

A major observation can be made regarding the best-ranked mAP@0.5 datasets: HDRI, Distractors and Color Light. Recalling *Table 4.1*, it is depicted that these randomization parameters are not invasive to the mesh appearance of the SLC. Conversely, the remaining datasets that affected the SLC mesh appearance achieved a lower than the control group performance. It may be argued that the Location, the Depth of Field, and the Noise datasets do not affect the appearance of the mesh; nonetheless, Fig. reveal that sampling the localization disassembles the SLCs stacks and at times overlapped the meshes, and the out-of-focus and Noise occasionally randomized the data up to the point of not keeping the key features of the SLC, making it, even for the human-eye, confused or not recognized at all.

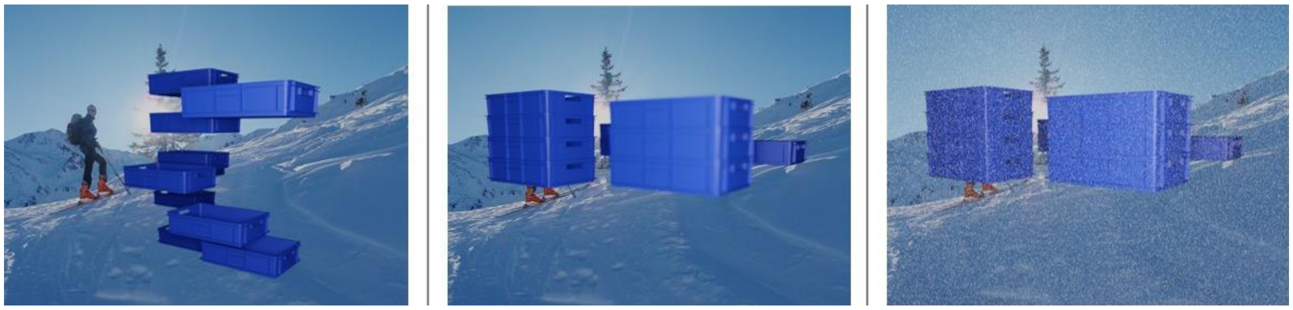


Figure 5.2 Location (left), Depth of Field (middle), Noise (left)

Visual Representation

Based on the preceding discussion, the HDRI and the Color Light datasets achieved the highest mAP@0.5. For this reason, a visual representation of the behavior on a world testing environment is shown. On the left, the ground-truth label is shown; on the middle, the Color Light randomization was used for training purposes; and on the right the HDRI randomization approach was followed. A significant observation is that the number of detections is lower in the Color Light compared to the HDRI. Additionally, these example images (left panel) do not show False Positives at all, but it does show False Negatives. Conversely, the HDRI increases the number of detections and True Positives (right), but a significant increase on the False Positives is seen.

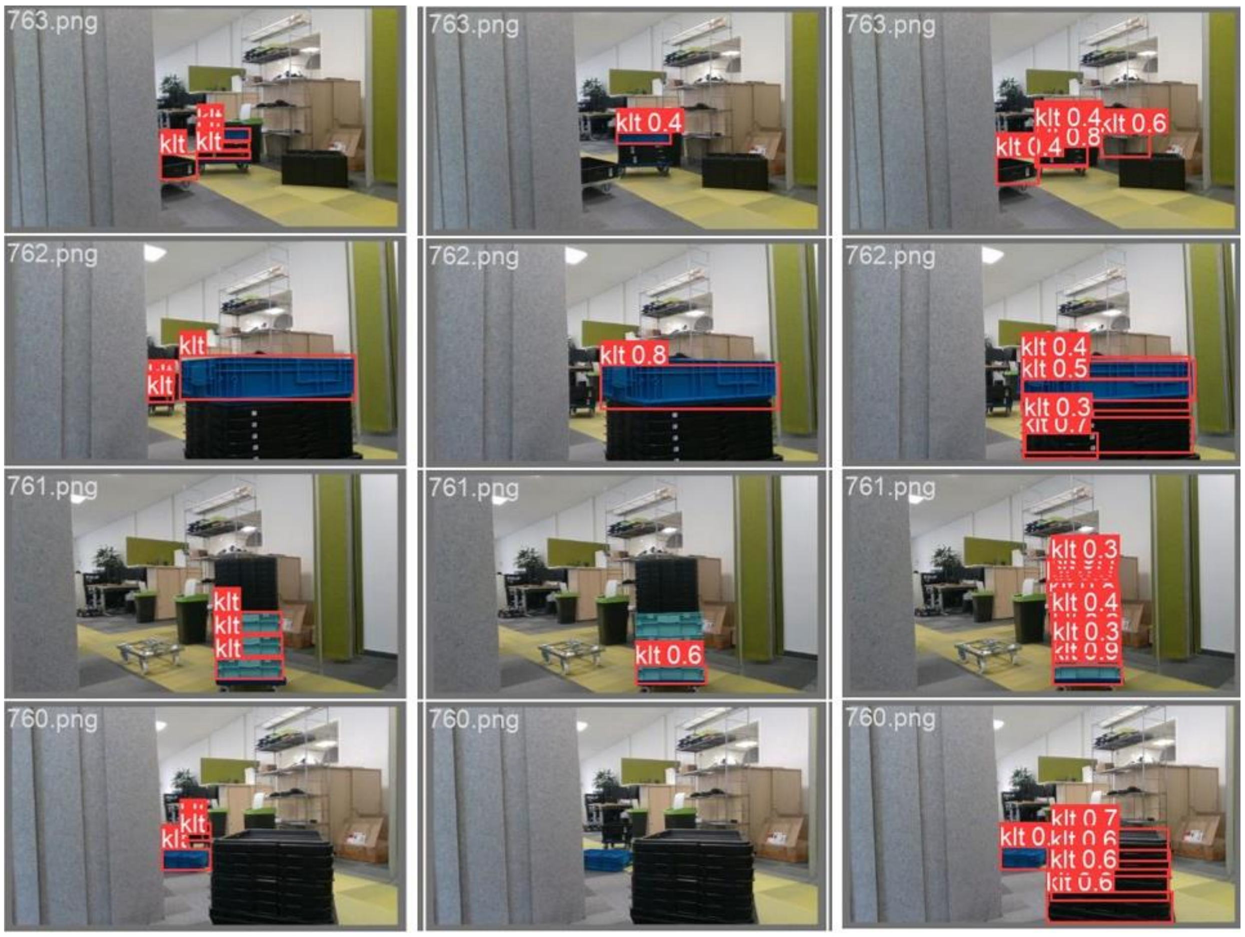


Figure 5.3 YOLOv5s being tested on a real-world environment. On the left, the ground-truth label is shown; on the middle, the Color Light randomization was used for training purposes; and on the right the HDRI randomization approach was followed.

5.2 YOLOv7

It was expected to replicate the YOLOv5s training pipeline using the YOLOv7 model. However, the increase in complexity of the YOLOv7 architecture [7] made infeasible to even start the training process with the available computational power. After a prior-exploration step, it was noted that the batch size hyperparameter needed to be drastically reduced from 16 to 4 to start the training. On top of that, YOLOv7 required approximately 6 hours per dataset, in contrast to the 23-minutes training time of YOLOv5s. For this reason, the five-times training pipeline used in YOLOv5s was not followed and consequently the YOLOv7 model was only trained only once per dataset. To sum up, the used hyperparameters are a batch size of 4, a pretrained yolov7 weight file, 120 training epochs, an image size of 640 pixels and the pre-default seed value.

Validation Dataset

Fig. 5.4 illustrates the recall, precision and mAP@0.5 performances denoted by YOLOv7. The HDRI, Light Color, Distractors and Control Group attained the highest mAP@0.5. Additionally, it can be observed that all datasets, except the Noise randomization, achieved a higher Precision than Recall. However, there is not significant improvement in mAP@0.5 in comparison to the YOLOv5s model. Therefore, it can be concluded that using YOLOv7 does not provide considerable advantages over YOLOv5s, especially since YOLOv5s requires less training time and less computational power.

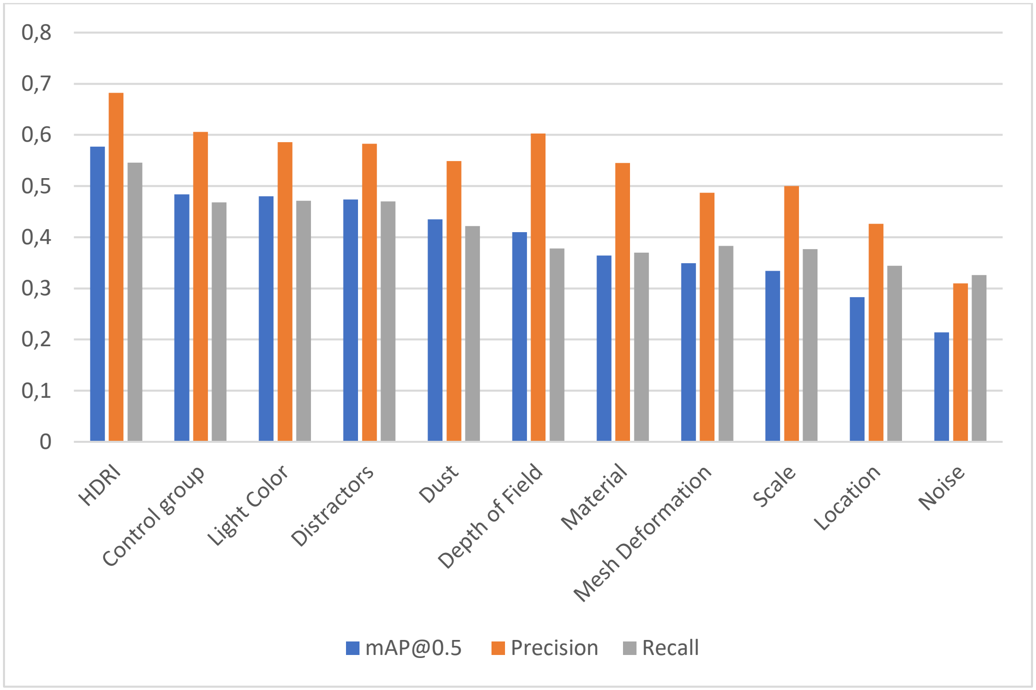


Figure 5.4 illustrates the recall, precision and mAP@0.5 performances denoted by YOLOv7.

Visual Representation

Based on the previous results, the HDRI dataset achieved a significantly higher performance on the mAP@0.5 metric than the control group. Fig. 5.5. depicts a visual representation of the Distractors, Light Color and Depth of Field datasets on a real-world testing environment is shown. The panel on the left shows the ground-truth label; the panel on the middle depicts the Light Color randomization; and, on the right the HDRI is illustrated. It can be observed that the Light Color dataset (middle panel) struggles with False Negatives on small objects. On the other hand, HDRI (right panel) decreased significantly the False Negatives on the image, but the False Positives increased.

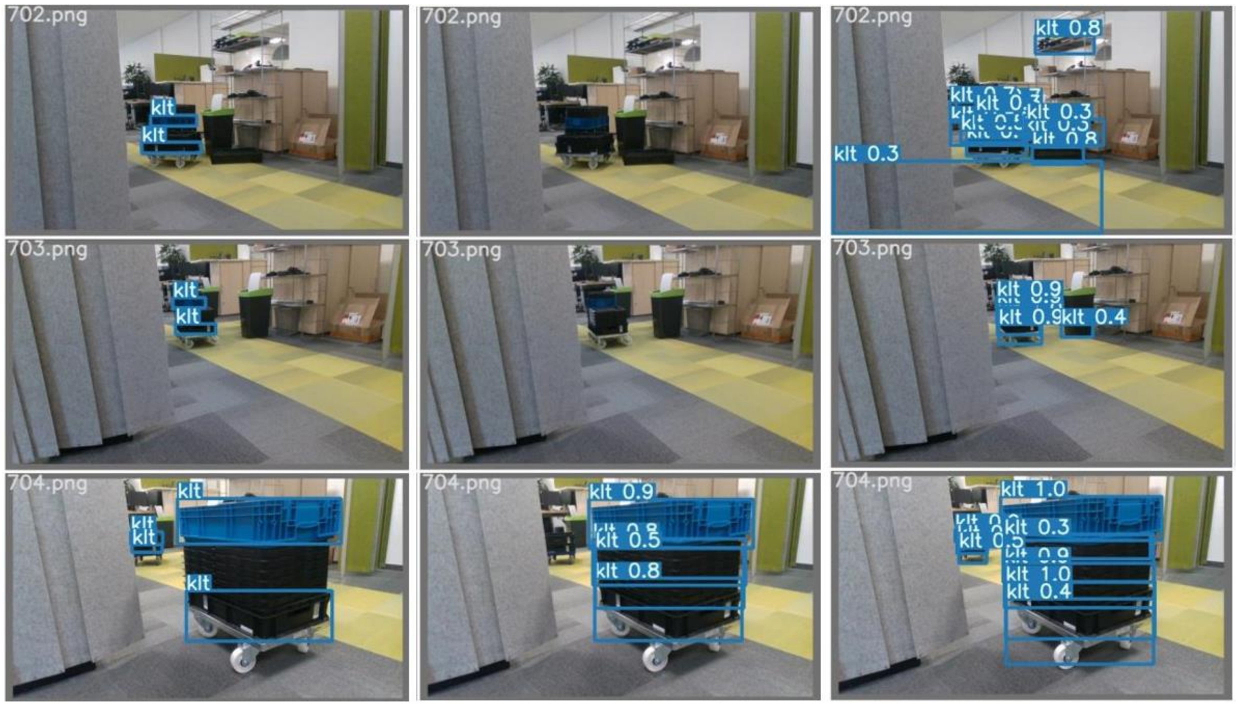


Figure 5.5 YOLOv7 being tested on a real-world environment. On the left, the ground-truth label is shown; on the middle, the Color Light randomization was used for training purposes; and on the right the HDRI randomization approach was followed.

6 A Further Comparison

After a deep visual analysis of the test and training datasets, it was discovered that the simplified SLC model missed multiple patterns shown on the real-world SLC. As illustrated in *Fig. 6.1*, the simulated SLCs (right) misses multiple aspects denoted on the real-world SLCs (left); for example, the real models contain a designed face with multiple patterns of vertical and horizontal lines, and the real SLC has no hollow for being gripped. In addition, the industrial domain typically adds a description of the carried load via a white label paper that may change the appearance of the SLC. For this reason, this chapter presents a further comparison was made to establish how much does the simplification of the SLC affected the overall object detection model.

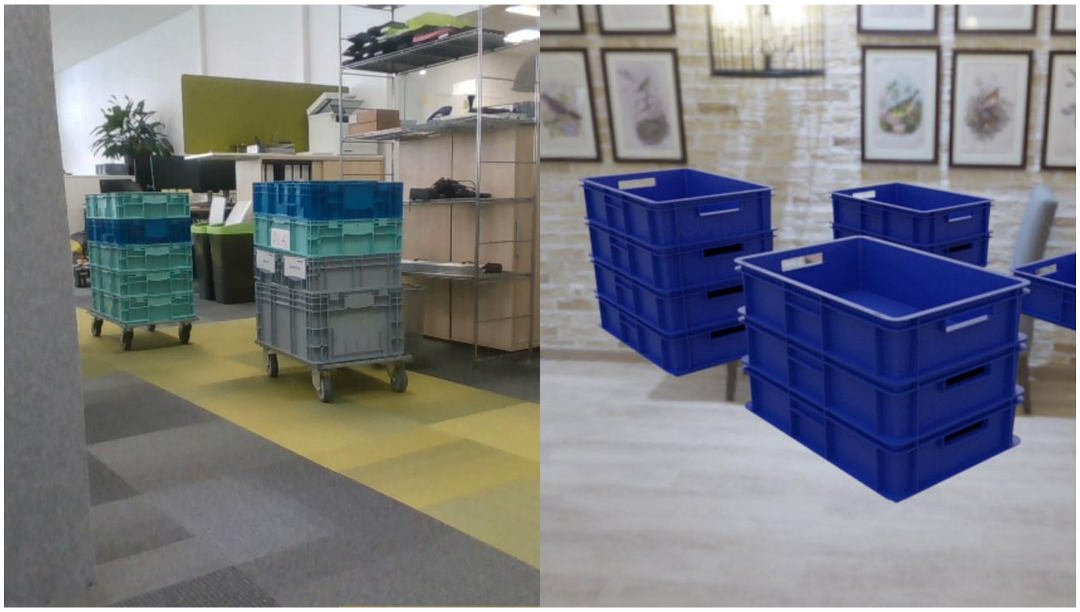


Figure 6.1 Real-world vs simulated SLC

Modelling and Rendering

A more robust design for the SLC was planned. The enhanced version takes inspiration of the complex design shown among the faces of the real-world SLC. The result is shown on *Fig 6.2*, where the first model (left) introduces a design pattern like the ones of the real-world SLC and the second uses the same enhanced model but adds the white label paper. Both, the enhanced SLC and the enhanced white-label-feature SLC, are generated by applying the HDRI randomization. 500 images were rendered per dataset to train a YOLOv7 model, which contain the hyperparameters described in the Subchapter 5.2, i.e., batch size of 4, pretrained yolov7 weight file, 120 training epochs, image size of 640 pixels and the pre-default seed value.

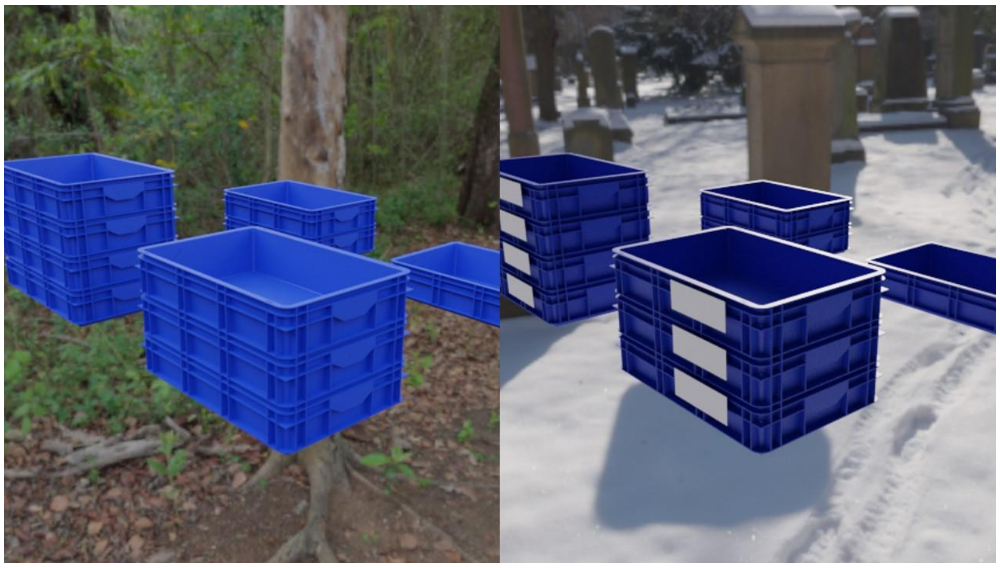


Figure 6.2 New version of the SLC (left) and the same model adding a white label paper (right)

Validation Dataset

Fig. 6.3 provides a summary of the training processes for the simplified SLC (left), enhanced SLC (middle) and enhanced SLC with the white labelling paper (right). It is important to note that YOLOv7 does not label the x-axis, but it should implied that the graph represents the mAP@0.5 vs training epochs. Based on the three graphs showing mAP@0.5 versus training epochs, a key observation can be made. While there is no significant improvement in the performance between the simplified SLC and the enhanced SLC, the enhanced SLC with the white labelling paper achieved a slightly higher mAP@0.5. With an mAP@0.5 score of 0.6, the model thus achieved the highest mAP of all generated datasets.

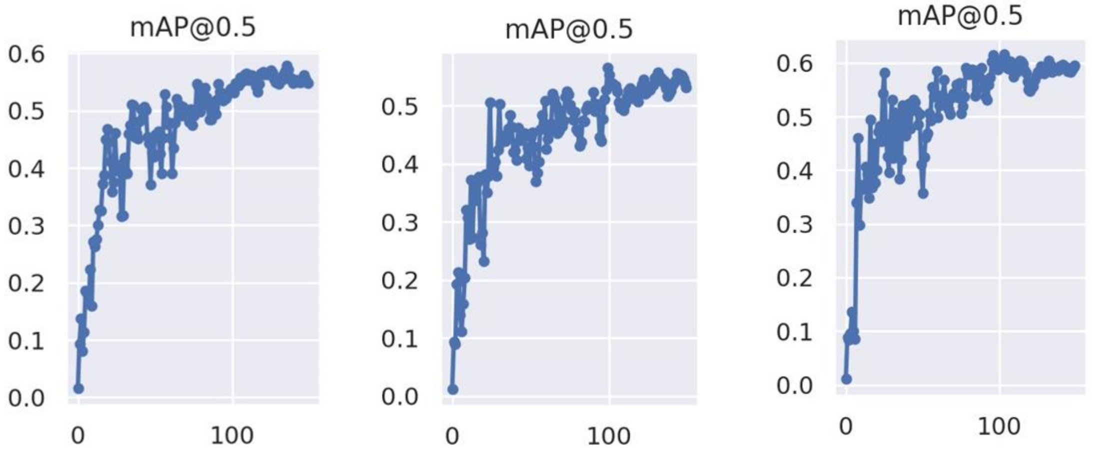


Figure 6.3 mAP@0.5. On the right, the simplified SLC, on the middle the enhanced SLC, and on the right, the enhanced SLC with the white label paper.

Visualization Representation

Multiple videos with SLCs were taken, in addition to the image test dataset, to visually evaluate the trained models. One such video, represented on Fig. 6.4 by the frame at second 0.55, it is depicted a challenging test environment for the YOLO model, resulting in lower-than-expected performance for all synthetic datasets so far generated. This can be attributed to three main factors: the camera angle depicts predominantly objects on an upper position, which is a behavior that was not reinforced on the training dataset; the distance from the camera to the shelf makes unclear the features of the SLCs; and there is a predominance of white papers used for labelling. A comparison between the enhanced SLC and the enhanced SLC with the white label papers revealed that training YOLOv7 with the latter not only considerably reduced the number of False Positives but also maintained a consistent classification of True Positives for some SLCs. Despite its advantages against the other datasets, the False Negatives are still predominant onto the video and thus stays below the desired performance.

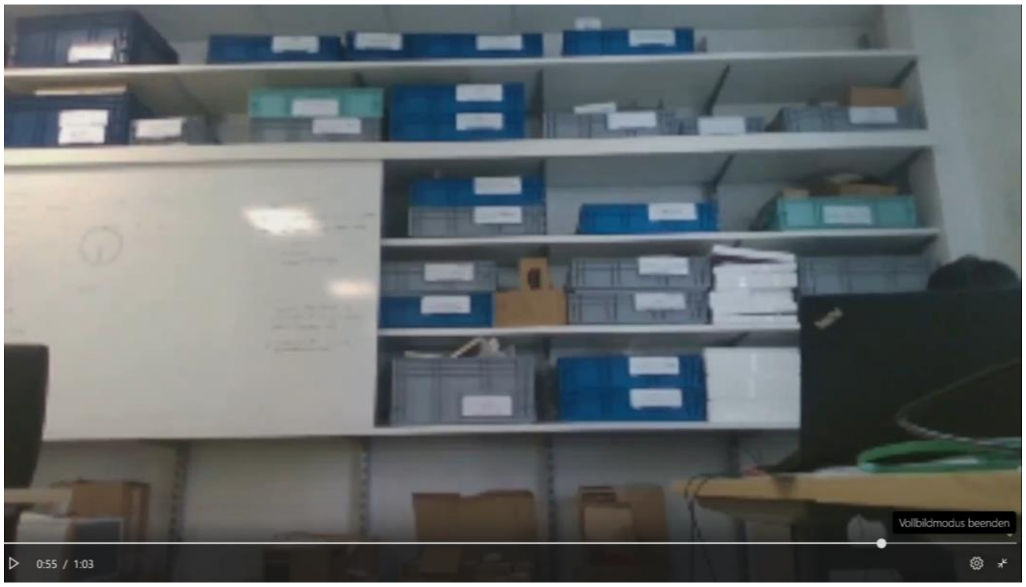


Figure 6.4. Test video

7 Conclusion

This chapter addresses the two primary objectives outlined on Chapter 2, which are listed below.

Objective 1: Determine the most appropriate synthetic generation strategy that meets the real-world detection of Small Load Carriers.

Based on the previous findings, it can be concluded that the domain gap between synthetic and real-world data is more pronounced when the SLC is not properly represented on the scene, regardless of the randomization being applied. Consequently, YOLOv5s exhibits a better generalization performance when the simulated SLC closely resembles its real-world counterpart. The HDRI and the Light's Color dataset were found to perform better on the mAP@0.5 metric, but the decision of selecting one over another required an extra visual representation of their detection behavior. The visualization suggests that scenarios that demand a higher accurate object detection, e.g., the EvoCarrier facing a SLC in a designated loading area, the color-light-based training dataset is the optimal choice, as the cost of a false positive is higher in such situations. Conversely, scenarios where missing a SLC carries a higher cost should prioritize a HDRI-based dataset. Finally, the chapter *A Further Comparison* illustrates the improvement of the mAP@0.5 by adding a special feature to the SLC, such as a white labelling paper. This result contributes to the idea that the best dataset should keep as realistic as possible the mesh appearance of the SLC and should sample the light´s nature.

Objective 2: Evaluate the YOLOv5s and YOLOv7 object detection algorithms and determine the most suitable for the EvoCarrier use case.

As explained on Subchapter 5.2, the available computation power made infeasible to train the YOLOv7 model with the pre-default hyperparameters; therefore, the batch size was reduced from 16 to 4. The implications of this decision are not delved on this thesis, but studies like [57] have shown that the larger the batch size, the higher the detector's performance. Given that the training time increased significantly, and no further training tests could be made, it is not clear how much impact does YOLOv7 received on its detection performance. However, YOLOv7 resembles YOLOv5s results by showing that the HDRI, Control Group, Light's Color and Distractors datasets outperformed the other ones. Considering the aforementioned, it can be concluded that YOLOv7 models do not show a significantly improvement over YOLOv5s, and thus should not be selected.

Future Work.

Multiple suggestions can be made to improve the results shown on the current thesis. The most important deals with the adjustments of hyperparameters, such as the batch size, to really conclude the behavior of YOLOv7 against YOLOv5s. Additionally, combining randomization parameters should be studied in the future, as studies like [45] present. Finally, a combination of multiple randomization datasets should be studied in the future.

8 Bibliography

[1] S. Haykin, Neural Networks and Larning Machines, New Jersey: Pearson, 2009.

[2] W. Ertel, Introduction to Artificial Intelligence, London: Springer, 2017.

[3] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *arXiv,* 2015.

[4] J. Velez, I. Posner and N. Roy, "Modelling Observation Correlations for Active Exploration and Robust Object Detection," *Journal of Artificial Intelligence Research,* no. 44, pp. 423-453, 2012.

[5] Z.-Q. Zhao, P. Zheng, S.-t. Xu and X. Wu, "Object Detection with Deep Learning: A Review," *Neural Networks and Learning Systems,* 2019.

[6] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv,* 2016.

[7] C.-Y. Wang, A. Bochkovskiy and H.-Y. Liao, "YOLOv7: Trainable bag-of-freebies," *arXiv,* 2022.

[8] F. Nowruzi, P. Kapoor, D. Kolhatkar, F. Hassanat, R. Laganiere and J. Rebut, "How much real data do we actually need: Analyzing object detection performance using synthetic and real data," *rXiv,* 2019.

[9] N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy and T. Brox, "What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?," *International Journal of Computer Vision volume,* no. 126, p. 942–960, 2018.

[10] B. Mehta, M. Diaz, F. Golemo, C. Pal and L. Paull, "Active Domain Randomization," *arXiv,* 2019.

[11] D. Arnold, Intralogistik: Potentiale, Perspektiven, Prognosen, Heidelberg: Springer, 2006.

[12] J. Fottner, D. Clauer, F. Hormes, M. Freitag, T. Beinke, L. Overmeye, S. Gottwald, R. Elbert, T. Sarnow, T. Schmidt, K. Reith, H. Zadek and F. Thomas, "Autonomous Systems in Intralogistics – State of the Art and Future Research Challenges," *Logistics Research,* 2021.

[13] Evocortex, "Evocarrier," [Online]. Available: https://evocortex.org/evocarrier/ . [Accessed 03 08 2023].

[14] DE-PACK, "KLT VDA," [Online]. Available: https://klt-vda.com/properties/small-load-carrier-system . [Accessed 08 03 2023].

[15] A. X. Isoco, "Alpina X Isoco," [Online]. Available: https://www.isoco-alpina-shop.com/products/rl-klt-6147?_pos=2&_sid=6800c592d&_ss=r. [Accessed 3 8 2023].

[16] S. Allibert, "VDA RL-KLT 6147," [Online]. Available: https://www.schoellerallibert.de/produkte/stackable-containers/euro-stacking-containers/vda-rrl-klt/vda-rl-klt-6147-kunststoffbox/. [Accessed 12 03 2023].

[17] C. W. Landsiedel, "Semantic Mapping for Autonomous Robots in Urban Environments," Technische Universität München, München, 2018.

[18] H. H. Aghdam and E. Heravi, Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign, Tarragona: Springer, 2017.

[19] M. Taylor, Neural Networks: A Visual Introduction for Beginners, Blue Windmill Media, 2017.

[20] R. Weiss, S. Karimijafarbigloo, D. Roggenbuck and S. Rödiger, "Applications of Neural Networks in Biomedical Data Analysis," *Biomedicines,* no. 10, 2022.

[21] P. Madhyastha and R. Jain, "On Model Stability as a Function of Random Seed," in *Proceedings of the 23rd Conference on Computational Natural Language Learning*, Hong Kong, 2019.

[22] L. Alzubaidi, J. Zhang, A. Humaidi, A. Al-Dujail, Y. Duan, O. Al-Shamma, J. Santamaría, M. Fadhel, M. Al-Amidie and L. Farhan, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data,* pp. 8-53, 2021.

[23] E. Soria, J. Martin, R. Magdalena, M. Martinez and A. Serrano, Handbook of Research on Machine Learning Applications, IGI Global, 2009.

[24] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. Zitnick and P. Dollár, "Microsoft COCO: Common Objects in Context," *European Conference on Computer Vision,* no. 8693, p. 740–755, 2014.

[25] Y. H. Liu, "Feature Extraction and Image Recognition with Convolutional Neural Networks," *Journal of Physics: Conference Series,* no. 1087, 2018.

[26] M. Nishio, K. Togashi, R. Do and R. Yamashita, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging,* no. 9, pp. 611-629, 2018.

[27] J.-F.¸. Coucho, C. Guyeux, Raphael Couturier and M. Salomon, "Steganalysis via a Convolutional Neural Network using Large Convolution," in *arxViv*, 2016.

[28] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016.

[29] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Region-based Convolutional Networks for Accurate Object Detection and Segmentation," *Pattern Analysis and Machine Intelligence,* no. 38, pp. 142-158, 2015.

[30] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv,* 2018.

[31] G. Cloud, "Overview of hyperparameter tuning," 13 03 2023. [Online]. Available: https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview. [Accessed 20 03 2023].

[32] T. Li, Y. Ma and T. Endoh, "A Systematic Study of Tiny YOLO3 Inference: Toward Compact Brainware Processor With Less Memory and Logic Gate," *IEEE Access,* no. 8, pp. 142931-142955, 2020.

[33] M. Hossin and M. Sulaiman, "A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS," *International Journal of Data Mining & Knowledge Management Process,* no. 5, 2015.

[34] M. Everingham, L. Gool , C. William, J. Winn and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision volume,* no. 88, pp. 303-388, 2010.

[35] J. Jordon, F. Houssiau , G. Cherubin, S. Cohen, L. Szpruch, M. Bottarelli, C. Maple and A. Weller, "Synthetic Data - what, why and how?," The Alan Turing Institute, London, 2022.

[36] C. G. Northcutt, L. Jiang and I. Chuang, "Confident Learning: Estimating Uncertainty in Dataset Labels," *Journal of Artificial Intelligence Research ,* vol. 70, pp. 1373-1411, 2021.

[37] L. Eversberg and J. Lambrecht, "Generating Images with Physics-Based Rendering for an Industrial Object Detection Task: Realism versus Domain Randomization," *Sensors,* no. 23, 2021.

[38] A. Tsirikoglou, "Synthetic data for visual machine learning: A data-centric approach," Linköping University Electronic Press, 2022.

[39] S. Marschner and P. Shirley, Fundamentals of Computer Graphics, NW: CRC Press, 2016.

[40] B. Lévy, S. Petitjean, N. Ray and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM Transactions on Graphics,* vol. 21, no. 3, pp. 362-371, 2002.

[41] M. Phar, W. Jakob and G. Humphreys, Physically Based Rendering From Theory To Implementation, Cambridge: Elsevier, 2016.

[42] L. Krecklau and L. Kobbelt, "Procedural Modeling of Interconnected Structures," *EUROGRAPHICS ,* vol. 30, 2011.

[43] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi and H. Katam, "BlenderProc: Reducing the Reality Gap with Photorealistic Rendering," in *Robotics: Science and Systems (RSS) Workshops*, 2020.

[44] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, 2017.

[45] S. Z. Valtchev and J. Wu, "Domain Randomization for Neural Network Classification," *Journal of Big Data,* no. 8, 2021.

[46] S. Hinterstoisser, O. Pauly, H. Heibel, M. Marek and M. Bokeloh, "An Annotation Saved is an Annotation Earned: Using Fully Synthetic Training for Object Detection," in *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, Seoul, 2019.

[47] M. P. D. Mauro, "Lecture Notes in Physics," in *Helioseismology: A Fantastic Tool to Probe the Interior of the Sun*, Berlin, Springer, 2003, pp. 31-67.

[48] A. Srivastava, "Projection in RealSense SDK 2.0," 02 08 2018. [Online]. Available: https://github.com/IntelRealSense/librealsense/wiki/Projection-in-RealSense-SDK-2.0. [Accessed 28 02 2023].

[49] T. Hodan, V. Vineet, R. Gal, E. Shalev, J. Hanzelka, T. Connell, P. Urbina, S. Sinha and B. Guenter, "Photorealistic Image Synthesis for Object Instance Detection," in *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019.

[50] A. Rafi, M. Izani and M. Tinauli, "High Dynamic Range Image (HDRI) Rendering A Technique for Architectural Pre-Visualisation," in *Proceedings of the International Conference on Environmental Design for Living Excellence (eDesign2004)*, Kuala Lumpur, 2005.

[51] Polyhaven, "Polyhaven," [Online]. Available: https://polyhaven.com/. [Accessed 03 03 2023].

[52] D. King, Depth of Field in Photography: Student Handout, 2004.

[53] Blender, "Blender 3.4 Manual," 15 02 2023. [Online]. Available: https://docs.blender.org/manual/en/latest/modeling/modifiers/deform/displace.html. [Accessed 26 03 2023].

[54] Y. Movshovitz-Attias, T. Kanade and Y. Sheikh, "How useful is photo-realistic rendering for visual learning?," in *ECCV 2016 Workshops*, Amsterdam, 2016.

[55] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, D. Cremers and T. Brox, "FlowNet: Learning Optical Flow with Convolutional Networks," in *2015 International Conference on Computer Vision (ICCV)*, Santiago, 2016.

[56] C. Mayershofer, T. Ge and J. Fottner, "Towards Fully-Synthetic Training for Industrial Applications," in *Proceedings of the 10th International Conference on Logistics, Informatics and Service Sciences (LISS 2020)*, Singapore, 2020.

[57] P. M. Radiuk, "Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets," *Information Technology and Management Science,* vol. 20, pp. 20-24, 20217.

[58] D. Cireşan, J. Schmidhuber and U. Meier, "Multi-column Deep Neural Networks for Image Classification," Dalle Molle Institute for Artificial Intelligence, Manno, Switzerland, 2012.

[59] P. J. Phillips and A. Toole, "Comparison of human and computer performance across face recognition experiments," *Image and Vision Computing,* pp. 74-85, 2014.

[60] O. Russakovsky, . J. Deng, . H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision,* no. 115, p. 211–252, 2015.

[61] A. Kolchinsky and D. Wolpert, "Semantic information, autonomous agency and non-equilibrium statistical physics," *Interface Focus,* vol. 8, no. 6, 2018.

[62] X. Ying, "An Overview of Overfitting and its Solutions," *Journal of Physics,* no. 1168, 2019.

[63] S. Sehra, D. Flores and G. Montanez, "Undecidability of Underfitting in Learning Algorithms," *2021 2nd International Conference on Computing and Data Science (CDS),* 2021.

[64] F. Yu, X. Xiu and Y. Li, "A Survey on Deep Transfer Learning and Beyond," *Mathematics,* no. 10, 2022.

[65] M. Deutschland, "Transform Spherical Coordinates to Cartesian Coordinates and Plot Analytically," [Online]. Available: https://de.mathworks.com/help/symbolic/transform-spherical-coordinates-and-plot.html. [Accessed 03 03 2023].

[66] Polyhaven, "Polyhaven," [Online]. Available: https://polyhaven.com/textures. [Accessed 03 03 2023].

[67] A. Norrstig, "Visual Object Detection using Convolutional Neural Networks in a Virtual Environment," Linköping University, Linköping, 2019.

[68] A. Constantin and R. Johnson, "Large gyres as a shallow-water asymptotic solution of Euler's equation in spherical coordinates," *Proc. R. Soc,* no. 473, 2017.

[69] D. J. Eck, Introduction to Computer Graphics, Geneva: Hobart and William Smith Colleges, 2021.

[70] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer and N. Lawrence, Dataset Shift in Machine Learning, Cambridge: MIT Press, 2009.

[71] K. Man and J. Chahl, "A Review of Synthetic Image Data and Its Use in Computer Vision," *Journal of Imaging,* vol. 310, no. 8, 2022.

[72] F. Harris, Mathematics for Physical Science and Engineering: Symbolic Computing Applications in Maple and Mathematica, ElSevier, 2014.

[73] X. Zhou, "Understanding the Convolutional Neural Networks with Gradient Descent and Backpropagat," in *IOP Conf. Series: Journal of Physics: Conf. Series 1004* , Suzhou, 2018.

A. Datasets

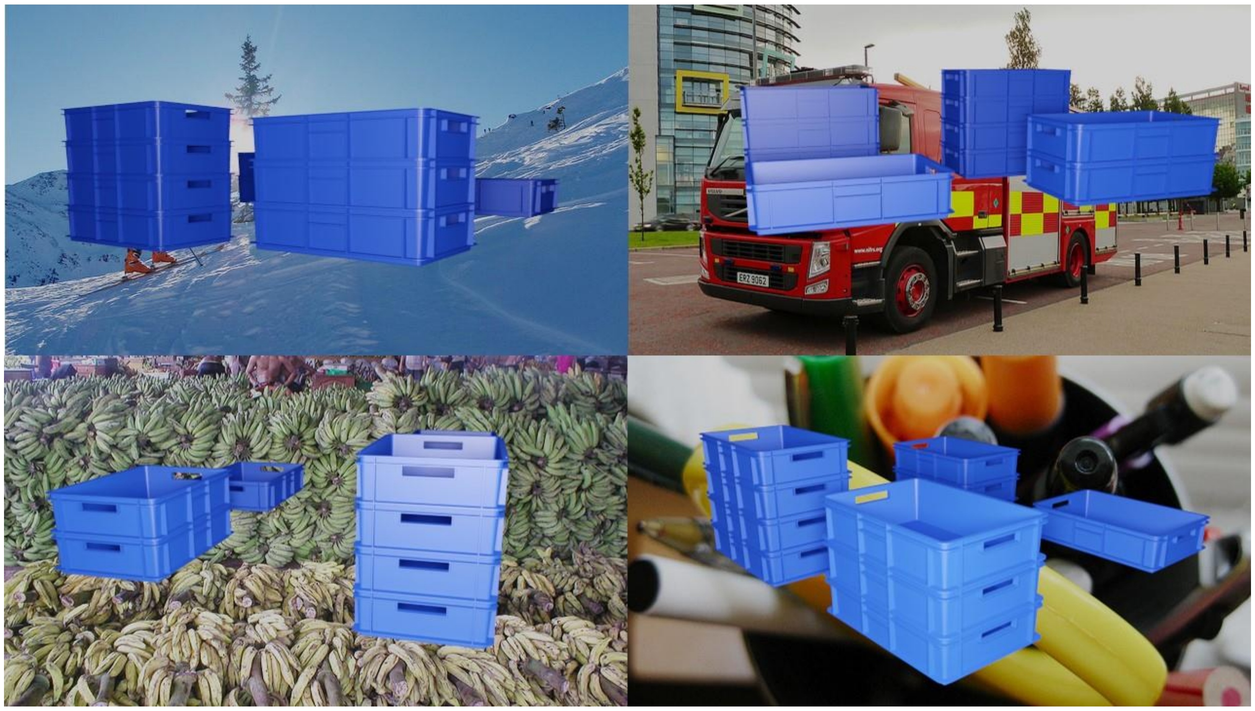


Figure A.1 Control Group

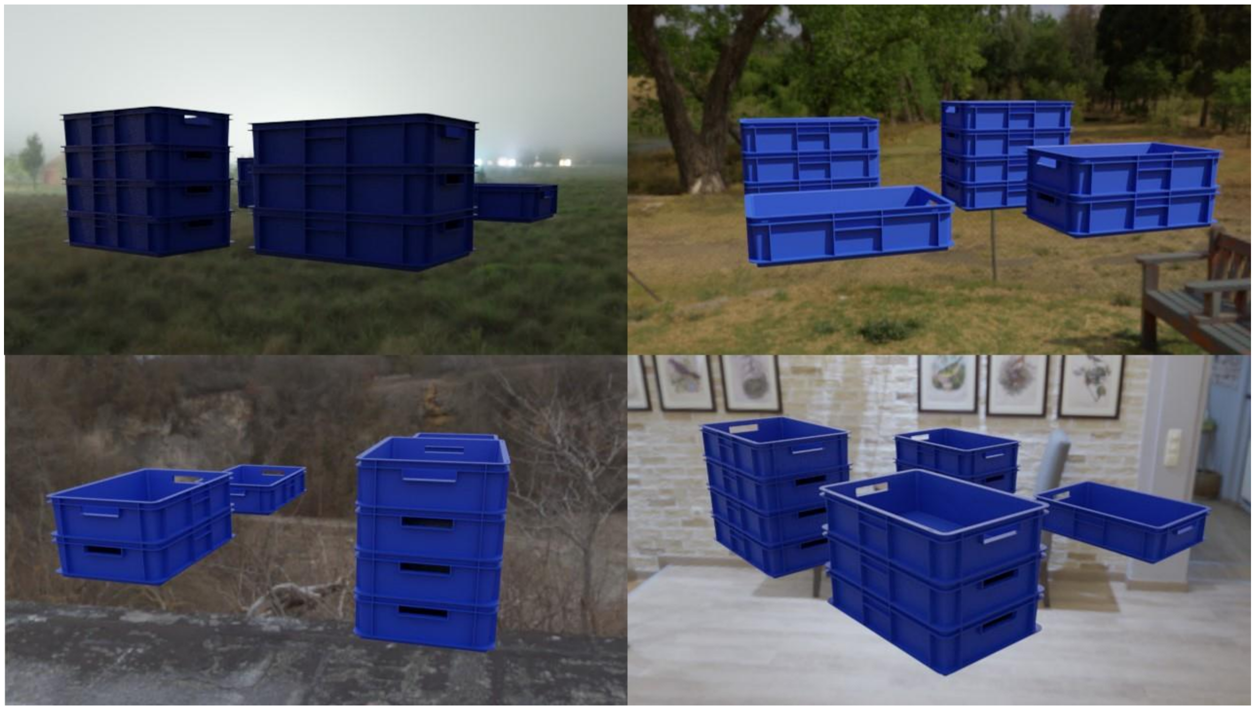


Figure A.2 HDRIs

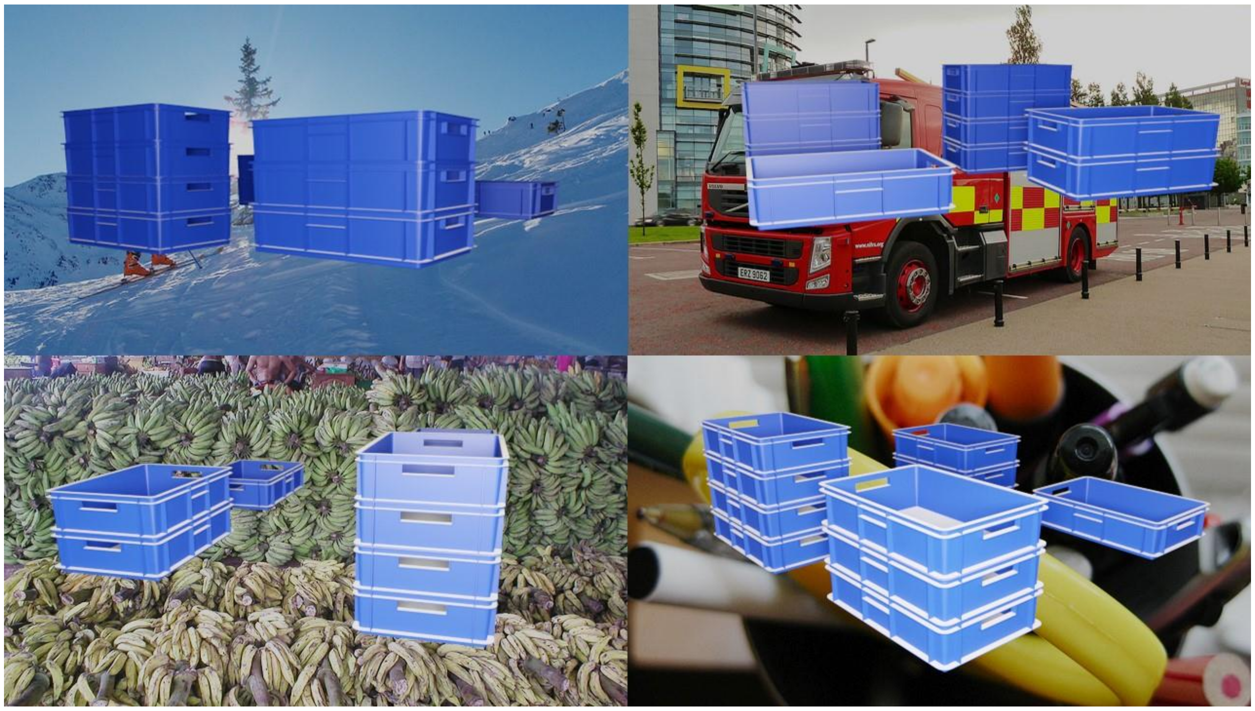


Figure A.3 Dust

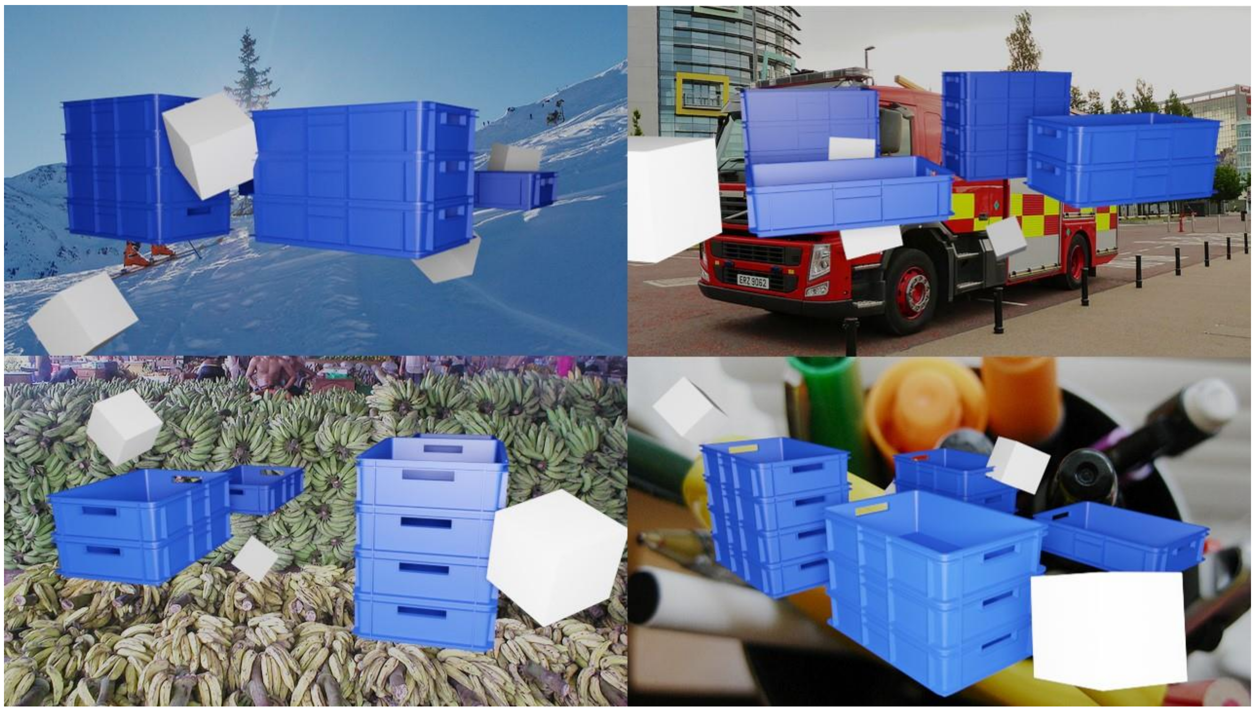


Figure A.4 Distractors

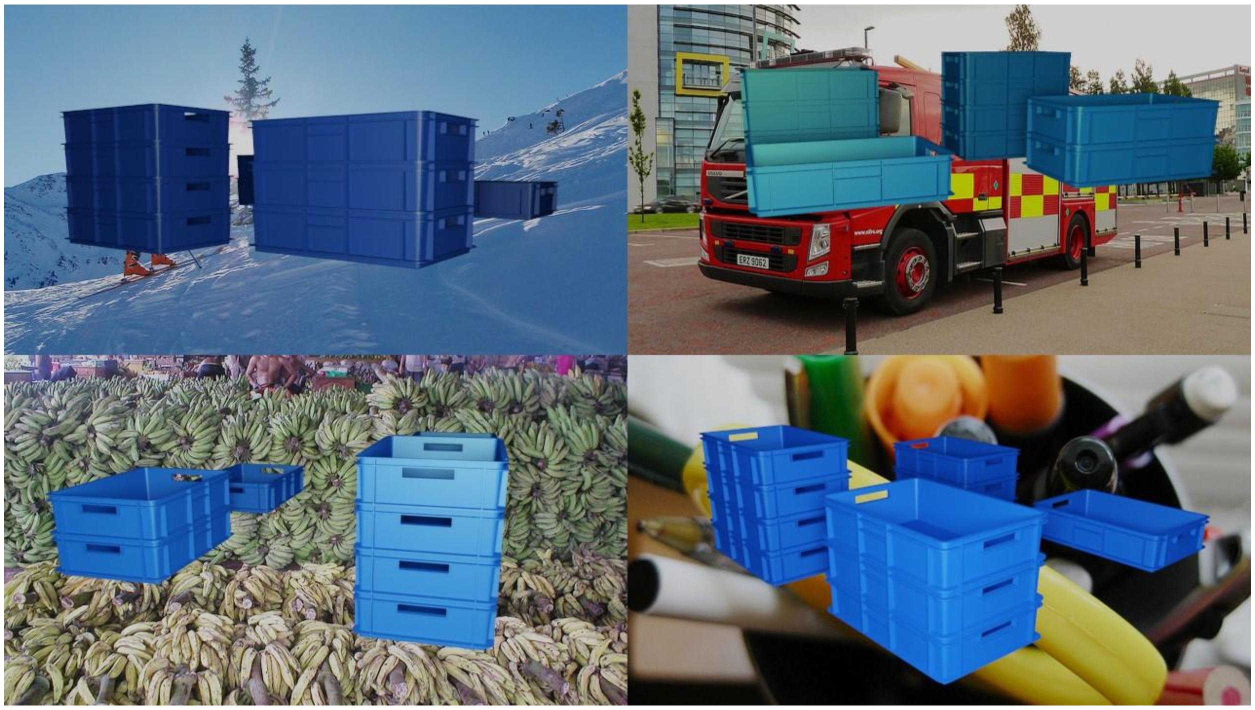


Figure A.5 Light Color

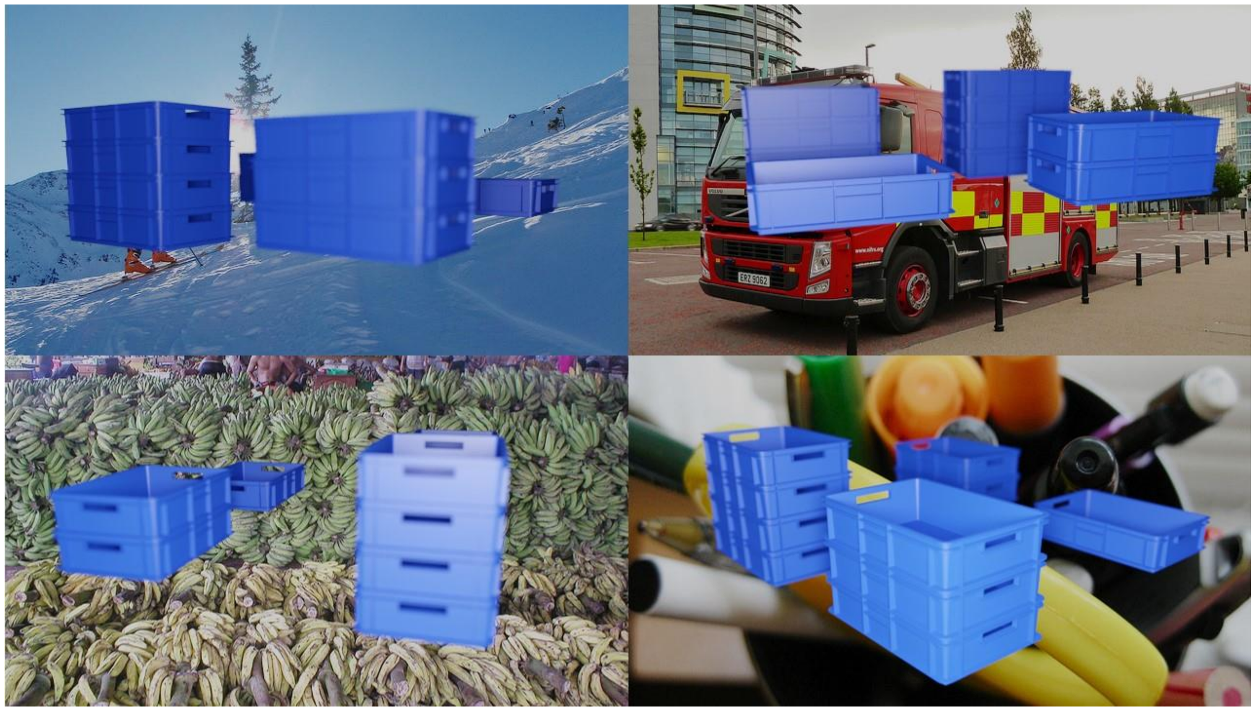


Figure A.6 Depth of Field

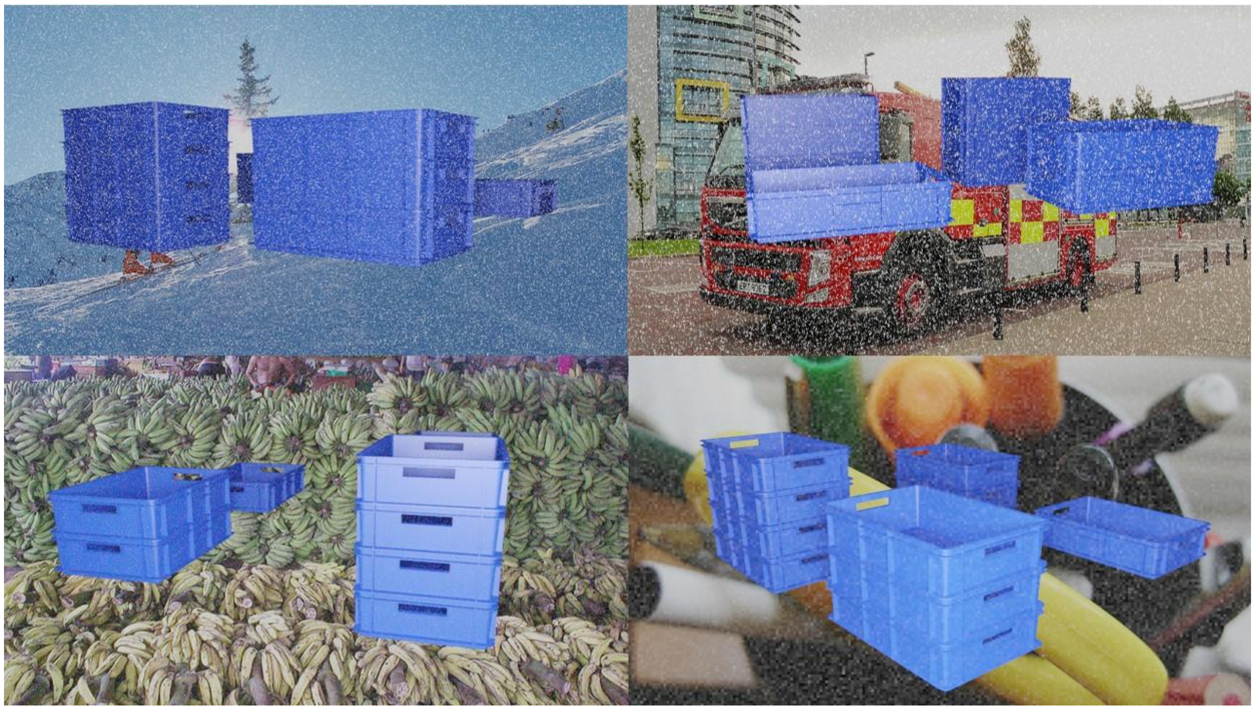


Figure A.7 Random Noise

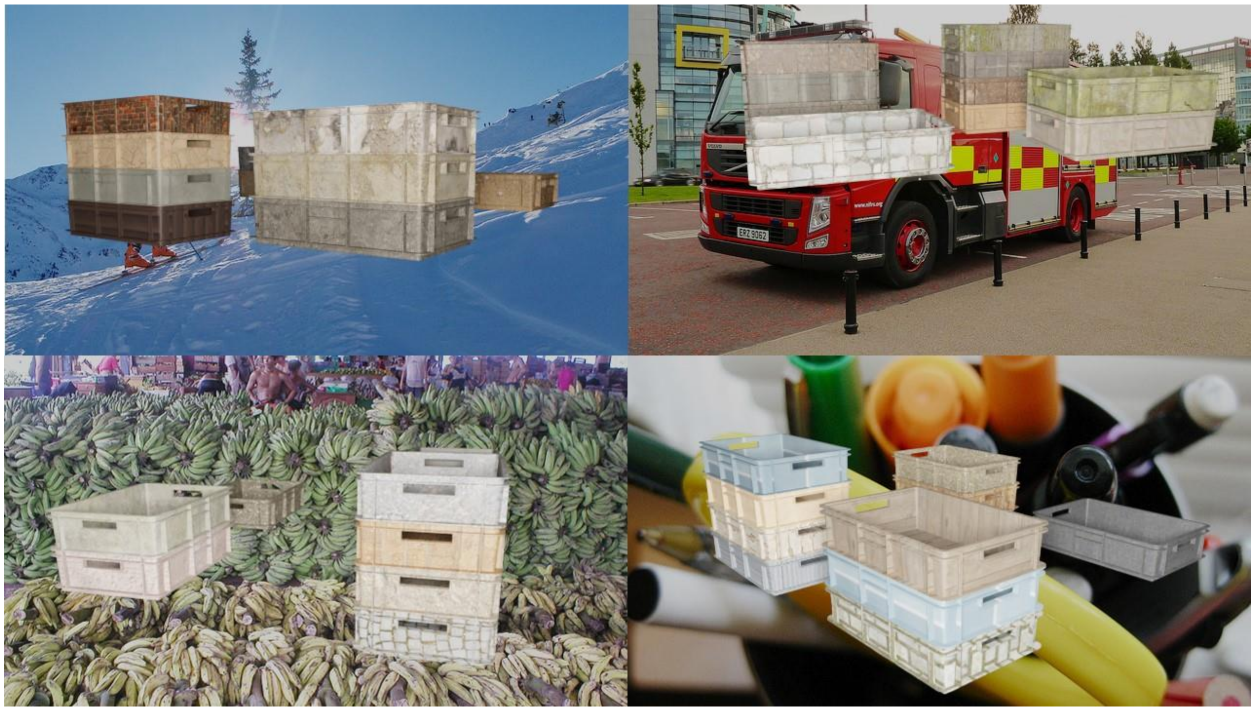


Figure A.8 Materials

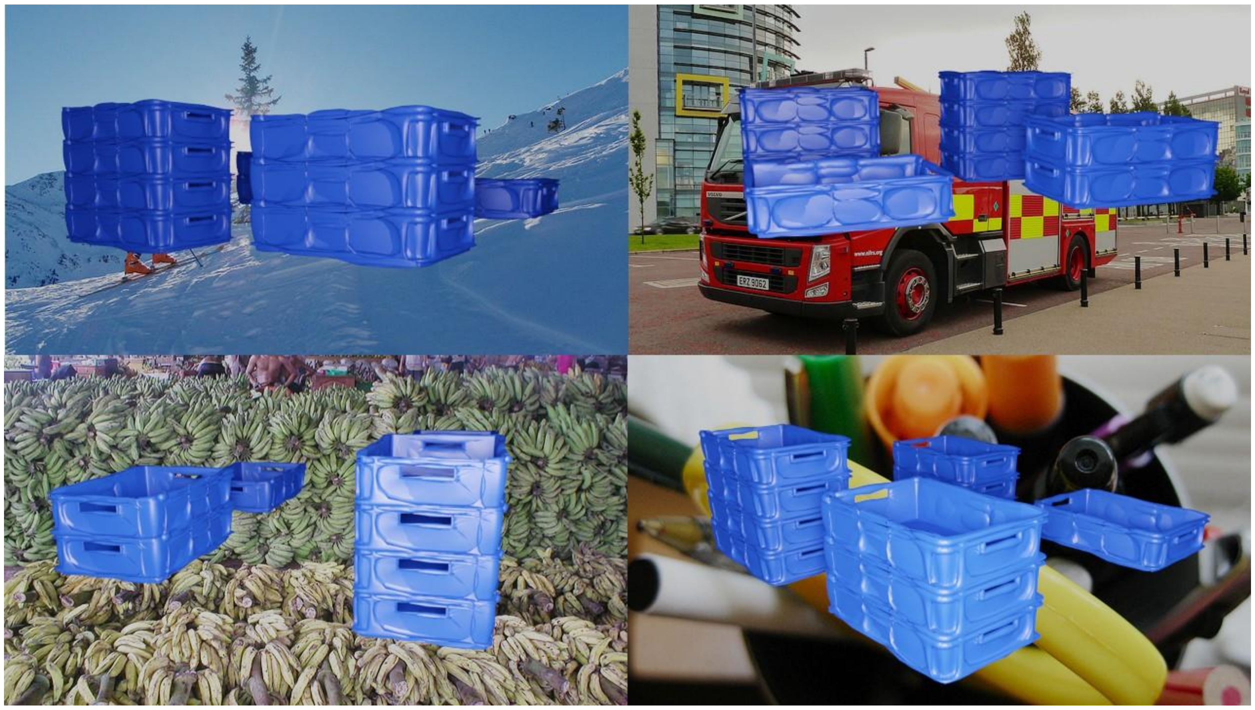


Figure A.9 Mesh Deformation

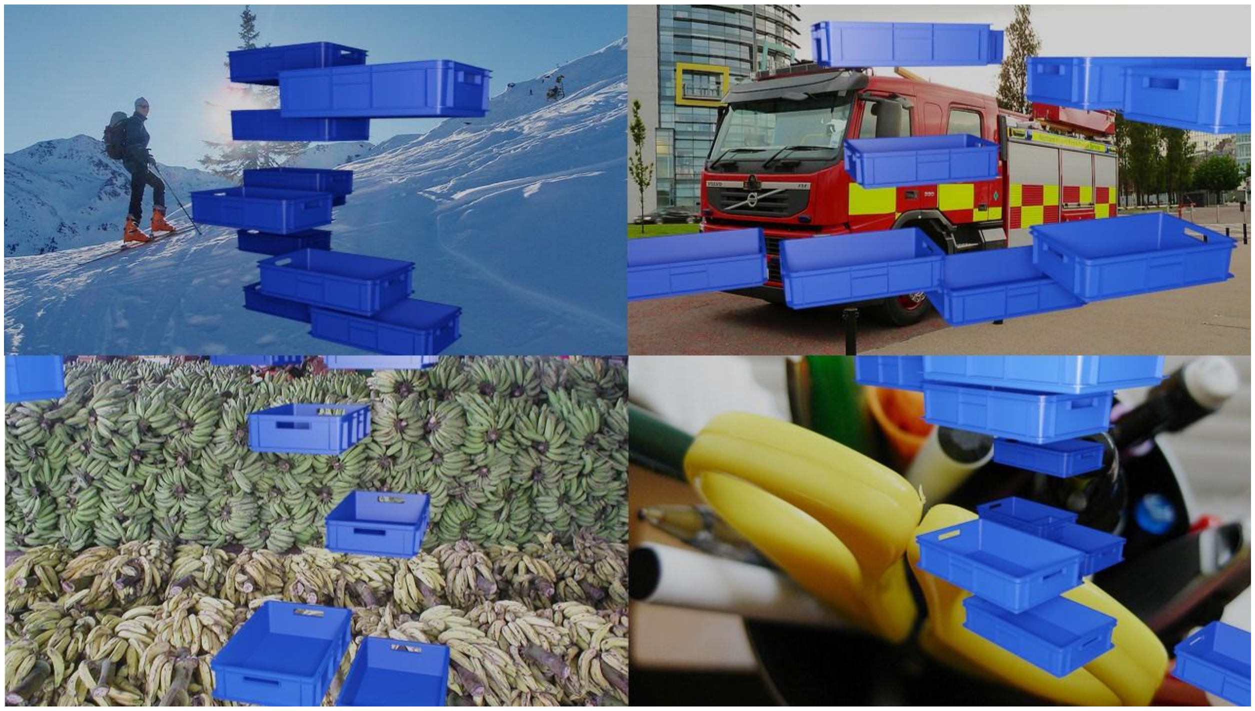


Figure A.10 Location

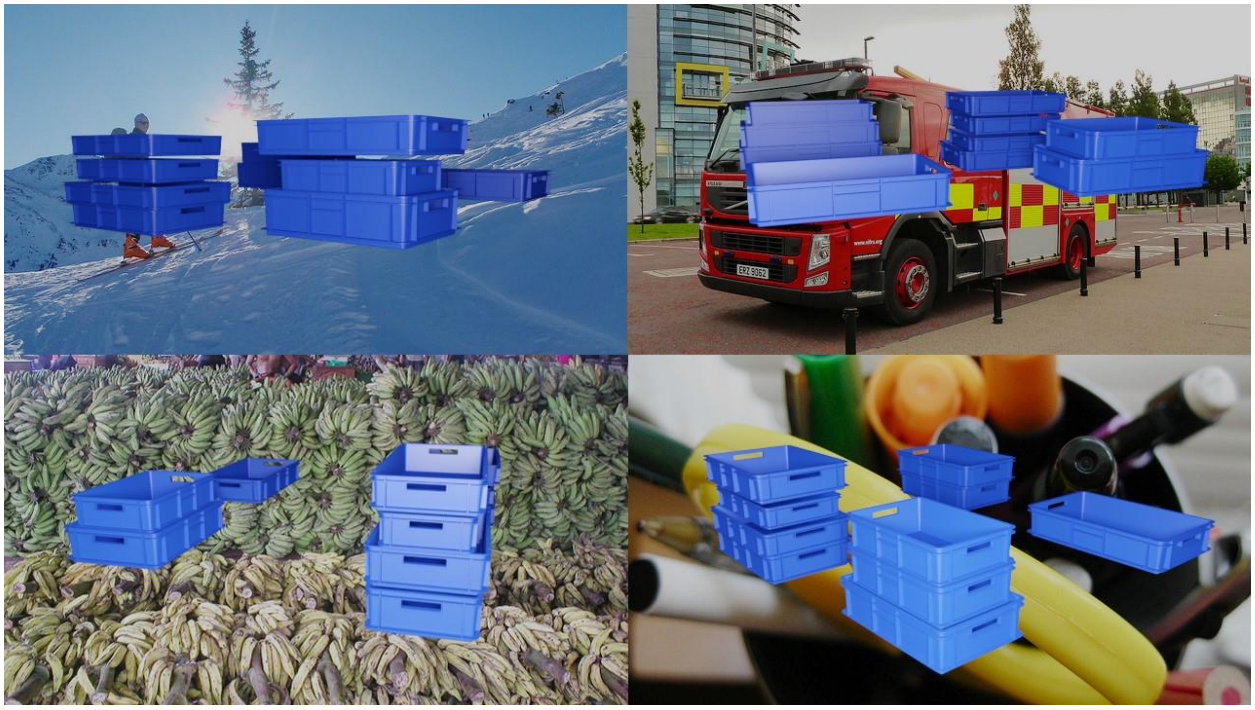


Figure A.11 Scale