h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Darmstadt University of Applied Sciences

– Faculty of Computer Science –

Evaluation of Synthetic Data Generation Using the "Cut-Paste" Method for Microorganism Detection

Submitted in partial fulfilment of the requirements for the degree of

Bachelor of Science (B.Sc.)

by

Sebastian Jörz

Matriculation number: 1116144

First Examiner	:	Prof. Dr. Eva Brucherseifer
Second Examiner	:	Prof. Dr. Kawa Nazemi

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

05. März 2025

Jörz

Sebastian Jörz

ABSTRACT

Training object detection models often relies on large annotated datasets. However, for niche areas such as the detection of microorganisms under a microscope, annotated datasets are scarce. Synthetic data generation methods offer a simple and efficient solution to expand these small datasets. However, the creation of realistic synthetic data remains challenging due to the difference between synthetic and real data distributions, known as the synthetic-to-reality gap. To investigate this gap, the “Cut-Paste” method is employed in combination with blending techniques to generate realistic synthetic data.

This thesis evaluates the “Cut-Paste” method for synthetic data generation in the context of microorganisms, specifically tardigrades, using the YOLOv11. Four individual blending methods — Alpha blending, Gaussian blending, Poisson blending and Pyramid blending — were assessed for their ability to seamlessly integrate cut-out objects into new backgrounds. Additionally, a multi-method blending approach was tested, and experiments were conducted with varying synthetic-to-real data ratios.

To quantify the synthetic-to-reality gap, three complementary metrics were used: (1) the difference in mean Average Precision at 50% Intersection over Union threshold (mAP@0.50) of the self-trained model, (2) the Fréchet Inception Distance (FID), and (3) the CLIP Maximum Mean Discrepancy (CMMD). While the mAP@0.50 directly measures the impact on object detection performance, FID and CMMD provide an independent assessment by quantifying the distributional distance between synthetic and real data.

The results demonstrate that the “Cut-Paste” method improves the performance of the self-trained model by approximately 0.18 mAP@0.50 when using multi-method blending and 0.16 mAP@0.50 with Pyramid blending. This suggests that incorporating synthetic data enhances the model’s robustness and generalization across different domains. Furthermore, the FID and CMMD metrics support these findings, indicating that synthetic images generated with Pyramid blending and multi-method blending are more similar to real images. These results underscore the importance of blending techniques in reducing the synthetic-to-reality gap and improving the effectiveness of synthetic data in training object detection models.

ZUSAMMENFASSUNG

Die Entwicklung von Objekterkennungsmodellen erfordert in der Regel große annotierte Datensätze. In spezialisierten Anwendungsbereichen, wie etwa der Erkennung von Mikroorganismen unter dem Mikroskop, sind derartige Datensätze jedoch selten verfügbar. Eine effiziente Lösung bietet die Generierung synthetischer Daten, wobei die realistische Erstellung solcher Daten eine Herausforderung darstellt. Diese Diskrepanz zwischen synthetischen und realen Daten ist auch als synthetic-to-reality gap bekannt. In dieser Arbeit wird die "Cut-Paste"-Methode in Kombination mit verschiedenen Blending-Methoden zur Generierung synthetischer Daten eingesetzt, um diese Lücke zu untersuchen und zu quantifizieren.

Evaluiert wird die "Cut-Paste"-Methode für die Erzeugung synthetischer Trainingsdaten bei der Erkennung von Mikroorganismen, insbesondere Bärtierchen, mithilfe des YOLOv11 Modells. Dabei wurden vier Blending-Methoden zur nahtlosen Integration von Objekten in neue Hintergründe untersucht: Alpha-Blending, Gaussian-Blending, Poisson-Blending und Pyramid-Blending. Zusätzlich wurde eine Kombination mehrerer Blending-Methoden getestet und Experimente mit verschiedenen Verhältnissen synthetischer zu realer Daten durchgeführt.

Zur Quantifizierung der synthetic-to-reality gap wurde eine Auswahl von drei unterschiedlichen Metriken herangezogen: (1) die Differenz der mean Average Precision bei einer Intersection over Union von 50% (mAP@0.50) des selbst trainierten Modells, (2) die Fréchet Inception Distance (FID) und (3) die CLIP Maximum Mean Discrepancy (CMMD). Während mAP@0.50 die Auswirkungen auf die Leistung der Objekterkennung direkt misst, bieten FID und CMMD eine unabhängige Bewertung, indem sie den Abstand zwischen den Verteilungen synthetischer und realer Daten quantifizieren.

Die Ergebnisse zeigen, dass die "Cut-Paste"-Methode die Leistung des Modells signifikant verbessert. Durch den Einsatz von Pyramid-Blending konnte eine Steigerung um 0,16 mAP@0.50 erreicht werden, während die Kombination mehrerer Blending-Methoden zu einer Verbesserung von 0,18 mAP@0.50 führte. Dies deutet darauf hin, dass synthetische Daten das Modell robuster machen und die Generalisierungsfähigkeit zwischen verschiedenen Domänen verbessern. Die Werte von FID und CMMD bestätigen diese Ergebnisse und zeigen, dass die Blending-Methode einen signifikanten Einfluss auf die synthetic-to-reality gap hat. Hierbei zeigten vor allem die Kombination der Methoden und Pyramid-Blending eine höhere Ähnlichkeit zu realen Daten. Diese Erkenntnisse unterstreichen die Relevanz von Blending-Methoden für die Reduzierung der synthetic-to-reality gap und die effektive Nutzung synthetischer Daten im Training von Objekterkennungsmodellen.

CONTENTS

ILLUSTRATIONS

Part I

Thesis

1 INTRODUCTION

Over the past decade, deep learning has transformed the field of object detection [2, 10]. A significant challenge in this domain is the scarcity of high-quality training data, for supervised learning [2, 30]. Labeling data is a time-consuming and expensive process, which can introduce human and experimental biases that negatively impact the model's performance during deployment [30].

Furthermore, models have been shown to extrapolate poorly beyond their training datasets and have limited out-of-distribution generalization behavior [58]. This phenomenon is also known as the domain gap, which occurs when there is a significant discrepancy between the distribution of the training data and the test data [23].

In order to address these issues, a number of data augmentation techniques have recently been employed with increasing frequency, particularly the generation of synthetic data [67], which has been shown to enhance the performance of object detection models [12]. A notable benefit of synthetic data is its scalability, which allows for the generation of extensive and varied datasets [67]. This is particularly advantageous in specialized domains where access to authentic data is limited or challenging to obtain.

Synthetic data can also facilitate the bridging of the domain gap between training and test data, thereby enhancing model generalization [23]. An effective approach involves combining cut-out objects with images of different background environments [51], which increased the models robustness by exposing it to a more extensive range of scenarios. However, seamlessly integrating foreground elements into new backgrounds can be challenging, as the synthetic images may appear unrealistic and fail to capture the complexity of real-world data [12]. This discrepancy is referred to as the synthetic-to-reality gap, which is defined as the difference between synthetic and real data [23]. The synthetic-to-reality gap builds on the domain gap, but focuses on the difference between synthetic and real data, while the domain gap describes the difference between domains in general. The difference between the domain gap and the synthetic-to-reality gap is explained in more detail in section 3.2.

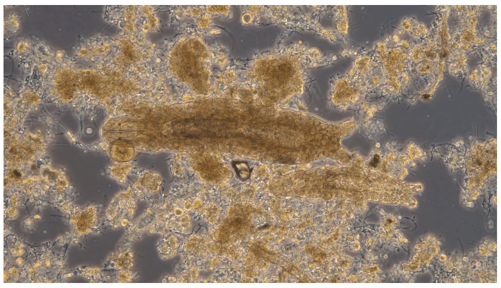
In order to bridge the synthetic-to-reality gap and enhance the model's performance, it is essential to generate synthetic images that are photorealistic and indistinguishable from real images [12]. This objective can be accomplished by employing blending methods, which merge synthetic images with real images to create more photorealistic results [12].

1.1 PROBLEM DEFINITION AND RELEVANCE

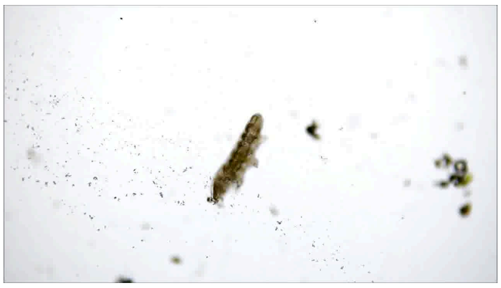
Access to clean water is a fundamental requirement for public health and sustainable development, as emphasized by the Sustainable Development Goal 6 (SDG 6) of the United Nations [50]. The SDG 6 aims to ensure the availability and sustainable management of water and sanitation. In this context, microorganisms play a crucial role, such as environmental monitoring, biotechnology, and medicine [47]. In particular, the detection of microorganisms in wastewater treatment is essential for monitoring and optimizing the biological purification process [33]. Among these microorganisms, tardigrades (water bears) are of particular interest, as they serve as biological indicators for the efficiency of the purification process. Their presence and condition provide insights into the health of the microbial community responsible for breaking down organic matter in sewage treatment [33].

The detection of these microorganisms can be a challenging task, as it requires a high level of expertise and can be time-consuming. Therefore, the automation of this process using machine learning models could improve the efficiency and accuracy of the detection process. However, the scarcity of high-quality training data poses a significant challenge for the development of such models. For instance, in the context of sewage plants, microorganisms are frequently observed to be surrounded by substantial amounts of mud, as seen in Figure 1.1a. Meanwhile, most public datasets for microorganisms are more clear and contain almost no mud, as seen in Figure 1.1b, since they are often taken in a laboratory environment. Furthermore the images are often taken with different microscopes, which can lead to different conditions, like illumination and contrast [46]. This represents the typical problem of the domain gap between the training and test data, as referenced by Dwibedi et al. [12] and Giakoumoglou et al. [23]. Consequently, when a model is trained with images of a clean environment and is then applied to images containing mud, it could result in the mud being identified as microorganisms, and vice versa. This could lead to false positives and negatives, which could have severe consequences in the context of sewage treatment, as it could lead to incorrect conclusions about the health of the microbial community.

In order to address this challenge, it is necessary to evaluate a solution capable of generating synthetic images of microorganisms in such environments, thereby mitigating the domain gap between the training and test data. Furthermore, it is essential that the synthetic images are generated in a way that they are as realistic as possible, thereby enabling the model to generalise effectively to real-world images.



(a) In a muddy environment. Source: own image



(b) In a clear environment. Source: [34]

Figure 1.1: Tardigrades under a microscope.

1.2 GOAL

The objective of this thesis is to evaluate the effectiveness of synthetic data augmentation for a limited, out-of-domain dataset of microorganisms — focusing particularly on tardigrades — in order to enhance object detection performance. This is achieved by training a state-of-the-art YOLOv11 model on both real and synthetic images and comparing the mean Average Precision (mAP) values. The mAP values serve as a primary indicator to quantify the synthetic-to-reality gap, which reflects how well the model generalizes when synthetic images are incorporated into the training process.

Synthetic images are generated using a variety of blending techniques, namely Alpha, Gaussian, Poisson, and Pyramid blending, with Pyramid blending being of particular interest due to its limited prior application in this context. The thesis not only evaluates the impact of each individual blending method on the model's performance but also investigates the effect of combining the best-performing methods to determine if a multi-method approach further reduces the synthetic-to-reality gap.

To validate the findings from the YOLOv11 mAP comparisons, two additional metrics are employed: the Fréchet Inception Distance (FID) and the CLIP Maximum Mean Discrepancy (CMMD). These metrics provide an independent assessment by quantifying the similarity between synthetic and real images from a distributional perspective, thereby offering a more comprehensive evaluation of the impact of blending methods on the syntoreal gap. In summary, three complementary metrics are employed to quantify the synthetic-to-reality gap: (1) the mAP values of the self-trained model, (2) the FID, and (3) the CMMD.

The goal can be summarized by the following research questions:

RQ1. How can synthetic generated images impact the model's ability to generalize between different environments?

RQ2. How does the selection of the blending method influence the synthetic-to-reality gap?

This thesis will contribute to the field of object detection by providing the following insights:

- Demonstrating that synthetic data augmentation can improve the generalization of object detection models in out-of-domain datasets, particularly in the context of microorganisms.
- Evaluating multiple blending methods — including the less-explored Pyramid blending — and their impact on the quality of synthetic images.
- Quantifying the synthetic-to-reality gap using YOLOv11 mAP values and validating these findings with FID and CMMD metrics, offering a comprehensive analysis of how blending methods affect the synthetic-to-reality gap.

1.3 STRUCTURE

In order to achieve the objectives of this thesis, the following structure outlines the methodology, experimental setup, and evaluation of the results.

First, the theoretical background of the research will be discussed in chapter 2, including the theoretical fundamentals of object detection, their metrics, and blending methods. It will also introduce the YOLOv11 object detection algorithm and its fundamentals. Furthermore, metrics to quantify the synthetic-to-reality gap will be introduced, including evaluations based on the performance of object detection models, as well as the FID and CMMD metrics. In chapter 3, the current state of the art for synthetic data generation, blending methods and the quantification of the synthetic-to-reality gap will be reviewed. This will provide a comprehensive overview of the current research landscape and identify potential research gaps. The methodology of the experiment will be outlined in chapter 4, which includes the synthetic data generation method as well as the augmentation and blending process. In chapter 5, the methods used for the evaluation and the test cases will be presented. This will include the datasets used in the experiment, along with the design of the tests. In addition, the parameters employed for the generation of synthetic data and the training of the model will be outlined. The results of the experiment will be presented in chapter 6, including the performance of the model on the synthetic data and a comparison with the performance on the real data. The results will be analyzed and discussed in the context of the research questions. Further, the quantification of the synthetic-to-reality gap using the self-trained model will be compaired with the FID and CMMD metrics. The results show that the synthetic data generation method is able to reduce the synthetic-to-reality gap between the training and test data and that the additional metrics are able to quantify the synthetic-to-reality gap. Finally, the conclusions of the study will be summarized in chapter 7, with a discussion of the findings, as well as recommendations for future research in this area.

2 THEORETICAL FOUNDATIONS

This chapter provides an overview of the theoretical foundations of object detection, synthetic data generation and quantification of the synthetic-to-reality gap. It begins with an introduction to object detection, how it works, and its challenges, especially in the field of microscopy. The chapter then introduces the YOLOv11 object detection algorithm and its fundamentals, along with the metrics used to evaluate object detection models. Moreover, the chapter concludes with an overview of synthetic data generation used in this work, including the "Cut-Paste" method. Finally, the evaluation metrics employed in this thesis to quantify the synthetic-to-reality gap will be explained. The concept of feature embeddings will be discussed in order to provide the theoretical background behind the application of these metrics in the context of the synthetic-to-reality gap.

2.1 OBJECT DETECTION

Object detection is a computer vision technique that involves identifying objects within an image or video and assigning them to predefined categories [38]. Some of these techniques include traditional computer vision algorithms, machine learning algorithms, as well as deep learning algorithms. The goal of these techniques is to identify and classify objects within an image or video, as well as to locate them within the image or video. As one of the primary tasks of computer vision, the ultimate goal of object detection is to detect the classes and locations of objects [46].

An object is a single entity that can be classified into a category, while an instance is a specific occurrence of an object in an image [12]. Object detection is the task of detecting and classifying objects in an image, while instance segmentation is the task of detecting and segmenting individual instances of objects in an image. Instance segmentation is a more challenging task than object detection, as it requires identifying and segmenting each instance of an object in an image. In this work, it should mainly be about traditional object detection. Object segmentation is not the focus of this work.

Object detection, as in the context of this work, is typically performed using machine learning algorithms, such as deep learning algorithms. These algorithms are trained on large datasets of labelled images, where each image is annotated with the class labels and bounding boxes of the objects present in the image [27].

2.1.1 NEURAL NETWORKS

To process the images, the images are fed into a neural network, which consists of multiple layers of neurons that process the image data and extract features from the images. A neural network is a mathematical model that is inspired by the structure and function of the human brain [32]. It is composed of multiple layers of neurons that process the input data and make predictions about the objects in the images. The neurons are connected to each other through weights and biases, which are used to adjust the parameters of the network during training [32]. A neuron can be thought of as a mathematical function that takes an input and produces an output, which is then passed to the next layer of neurons in the network. A function of a neuron can be for example $y = f(\sum_{i=1}^{n} w_i x_i + b)$, where y is the output of the neuron, f is the activation function, w_i are the weights, x_i are the inputs, and b is the bias. The weights and biases are adjusted during training to improve the accuracy of the predictions made by the network [20]. The activation function is a mathematical function that introduces non-linearity into the network, allowing it to learn complex patterns and make accurate predictions [20].

For the detection of objects in images, each pixel in the image is represented as an input to the first layer, the input layer of the neural network [49]. The neural network processes the image data by passing it through multiple layers of neurons, which extract features from the images and make predictions about the objects in the images. These layers are referred to as "hidden layers". An example of a neural network with one input layer containing three input neurons, two hidden layers, and one output layer containing one output neurons is shown in Figure 2.1. For object detection, the input to the neural network is an image, and the output is the class labels and bounding boxes of the objects detected in the image [59].

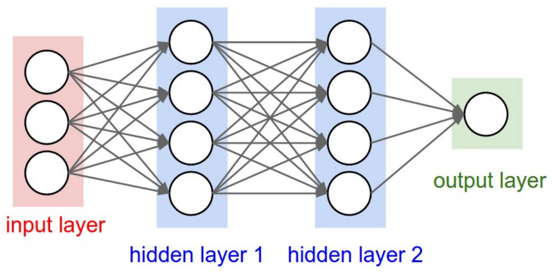


Figure 2.1: Neural Network with one input layer, two hidden layers and one output layer. Source: [69]

Convolutional Neural Networks (CNNs) are a type of neural network that is commonly used for object detection tasks. CNNs are designed to process images and extract features from the images by convolving filters over the image data to detect patterns and features in the images [10]. CNNs are composed of multiple layers of neurons, including fully connected layers, convolutional layers and pooling layers. An illustration for this can seen in Figure 2.2, where the input image is passed through multiple convolutional layers to extract features from the image. The result is the class labels (cls) and the location of the objects (loc) in the image. The application of these layers is twofold: first, they are employed to detect edges, textures and shapes in images; secondly, they are utilised to combine these features in order to detect objects in images. CNNs are trained on extensive datasets of labelled images, with the purpose of learning the features of objects in images and making predictions about objects in new images [10].

The bounding boxes indicate the location of the objects within the image, while the class labels indicate the category of the objects [10]. This is also known as the ground truth data, as it provides the correct labels and locations of the objects in the image. The bounding box is a 2D rectangle that encloses the object in the image, while the class label is the category of the object, such as car, person, or dog. The ground truth data is

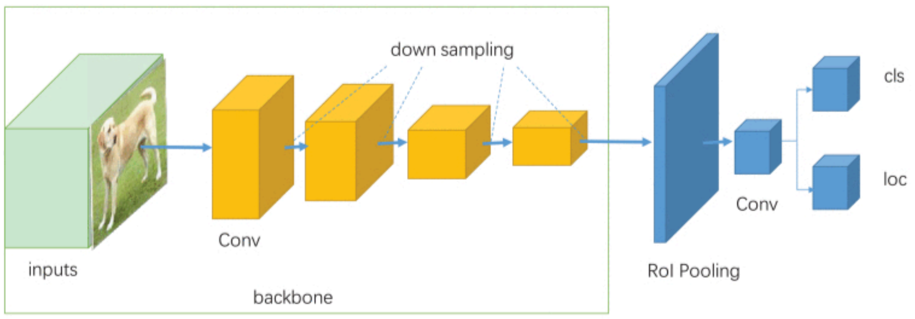


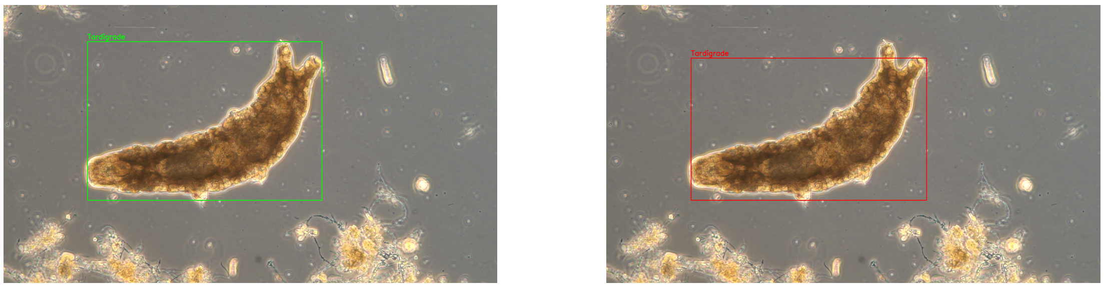
Figure 2.2: Single stage detector architecture. With the input image, the CNN extracts features and predicts the class labels and bounding boxes of the objects in the image. Source: [10]

used to train the object detection algorithm to accurately detect and classify objects in new images [27]. In contrast, the predictions made by the object detection algorithm are the class labels and bounding boxes of the objects detected in the image [10]. An example of the ground truth and a prediction is shown in Figure 2.3a and Figure 2.3b. During training, the algorithm predicts the class labels and bounding boxes of the objects in the image, and the predictions are compared to the ground truth labels to calculate the difference between them [10]. This difference, also known as the loss, is used to update the parameters of the algorithm to improve its performance. The goal of training is to minimize the loss and improve the accuracy of the predictions made by the algorithm [24]. This process is known as backpropagation, where the loss is propagated back through the network to update the weights and biases of the algorithm [20]. The learning rate affects the size of the updates made to the weights and biases during training, while the loss function determines how the loss is calculated and minimized during training [20]. A higher learning rate results in larger updates, while a lower learning rate results in smaller updates. The loss function is a mathematical function that measures the difference between the predicted and ground truth labels of the objects in the image [20].

Summarizing, object detection can be understood as a synthesis of classification, localisation, and segmentation tasks, as outlined by Diwan et al. [10]. The process of object detection entails the accurate classification and efficient localisation of one or more objects within an image. The classification task involves assigning a label to each object detected in the image, while the localisation task involves determining the precise location of each object within the image.

2.1.2 CHALLENGES

However, object detection presents several challenges, as discussed in [10]. A primary challenge is the inherent variation in object occupancy within an image, where objects may occupy a significant proportion of the pixels (e.g., 70–80%) or a very small proportion (e.g., 10% or less). Furthermore, the processing of low-resolution visual content presents a challenge, as it can reduce detection accuracy. Additionally, the presence of multiple objects of varying sizes in a single image adds complexity to the task. The scarcity of labelled data constitutes a significant impediment to the development of robust models. Additionally, the detection of overlapping objects within



(a) Ground Truth (b) Prediction

Figure 2.3: Example of object detection. (a) Ground truth with bounding boxes and class labels. (b) Prediction with bounding boxes and class labels.

visual content remains a substantial challenge, as it complicates both classification and localisation efforts.

Special challenges arise when applying object detection to microscopy images, as discussed by Ma et al. [46]. These challenges include low contrast, color overlap, and different illumination conditions. These challenges can make it difficult to detect objects in microscopy images. Additionally, the presence of noise and artifacts in microscopy images can further complicate the object detection process. The small size of objects in microscopy images can also pose a challenge, as it can be difficult to accurately detect and classify objects that are very small in size [58]. Furthermore, the presence of multiple objects in a single image can make it difficult to accurately detect and classify each object. These challenges highlight the need for robust object detection algorithms that can accurately detect and classify objects in microscopy images.

2.1.3 YOLOV11

In the context of object detection, YOLO (You Only Look Once) [59] represents a single-stage real-time detection system, as described by [10]. YOLO formulates object detection as a regression problem, allowing the detection of multiple objects in an image through a single forward pass of a convolutional neural network (CNN). The system divides the input image into an S $\times$ S grid, with each grid cell predicting B bounding boxes, confidence scores, and C class probabilities, as shown in Figure 2.4. The confidence score is predicated on two factors. First, the likelihood that a bound-

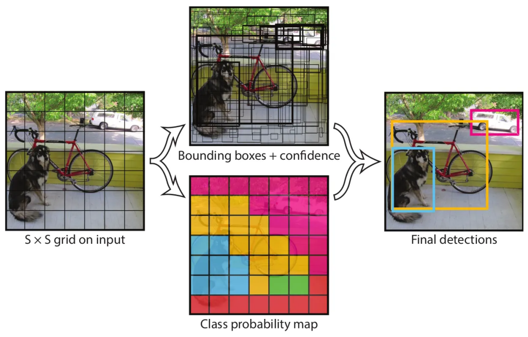


Figure 2.4: YOLO divides the image into a grid and predicts bounding boxes and class labels for each grid cell. Source: [59]

ing box contains an object is considered. Secondly, the accuracy of the bounding box localization is taken into account. The YOLO algorithm combines object localization and classification in a single step. This eliminates the need for region proposal methods, which are commonly used in two-stage detectors such as R-CNN [31].

YOLOv11 [40] is the latest iteration of the YOLO object detection algorithm. It builds on the previous versions of YOLO by incorporating new features and improvements to enhance its performance. YOLOv11 is designed to be faster and more accurate than its predecessors, making it an ideal choice for real-time object detection applications.

2.2 METRICS

The performance of object detection models is evaluated through the utilisation of various metrics. These metrics are employed to assess the model's precision in identifying objects within an image, as referenced in [52]. Each metric offers a distinct perspective on the model's performance, enabling the evaluation of its diverse object detection capabilities.

2.2.1 INTERSECTION OVER UNION

In order to identify a detection that is correct, a metric is utilised to measure the overlap between the predicted bounding box and the correct bounding box, which is referred to as the ground truth. This metric is termed the Intersection over Union (IoU) [10, 14]. The IoU is calculated as the ratio of the area of the intersection against the area of the union of the two bounding boxes. The formula for IoU is given by Everingham et al. [14] as:

$$\text{IoU} = \frac{\text{Area of Intersection } (A \cap B)}{\text{Area of Union } (A \cup B)} \tag{2.1}$$

The IoU is a number between 0 and 1, where a high IoU value indicates that the model's bounding box predictions are accurate, while a low IoU value indicates that the model's bounding box predictions are inaccurate [52]. An example of IoU is shown in Figure 2.5, where one of the boxes represents the ground truth, and the other box represents the predicted bounding box.

IoU is commonly used as a threshold for determining whether a detection is considered correct or not. For example, in the PASCAL VOC challenge [14], a detection is considered correct if the IoU value is above 0.5, indicating that the predicted bounding box overlaps with the ground truth bounding box by at least 50% [14].

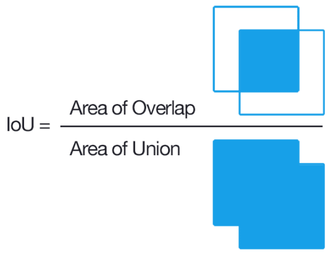


Figure 2.5: Intersection over Union (IoU) for object detection. Source: [62].

This threshold is used to classify the model's predictions into four categories: true positive (TP), false positive (FP), false negative (FN), and true negative (TN) [24]. A true positive occurs when the model correctly detects an object, while a false positive occurs when the model predicts an object that is not present. A false negative occurs when the model fails to detect an object that is present, and a true negative occurs when the model correctly does not detect an object. These categories are then used to calculate other metrics.

		Predicted Values	
		Positive	Negative
Actual Values	Positive	TP	FN
	Negative	FP	TN

Table 2.1: Classification of predictions into true positive (TP), false positive (FP), false negative (FN), and true negative (TN). Left-side: Predicted values, top: Actual values.

2.2.2 MEAN AVERAGE PRECISION

The mean average precision (mAP) is a metric employed to evaluate the performance of object detection models [24]. To calculate the mAP, the Precision and Recall metrics are utilised, and these are then used to calculate the AP for each class. Subsequently, the mAP is calculated as the mean of the AP values across all classes [46].

Precision and Recall

Precision is a measure of how many of the objects, detected by the model, are actually correct [24, 46, 52]. The formula for precision is given by [24]:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{2.2}$$

A high precision indicates fewer false positives, while a low precision suggests more false positives.

Recall quantifies the proportion of actual objects correctly detected [24, 52], given by [24]:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{2.3}$$

A high recall means fewer false negatives, whereas a low recall implies that some objects are missed.

Precision-Recall Curve

The precision-recall curve visualizes the trade-off between precision and recall at varying confidence thresholds [52]. Each prediction is assigned a confidence score, influencing detection accuracy. The Precision at a given threshold τ is defined by [52]

$$\text{Precision}(\tau) = \frac{\text{TP}(\tau)}{\text{TP}(\tau) + \text{FP}(\tau)} \tag{2.4}$$

and the Recall at a given threshold τ is represented by [52]

$$\text{Recall}(\tau) = \frac{\text{TP}(\tau)}{\text{TP}(\tau) + \text{FN}(\tau)}. \tag{2.5}$$

A high precision-recall curve signifies robust detections, while a lower curve suggests reduced accuracy. An example is illustrated in Figure 2.6.

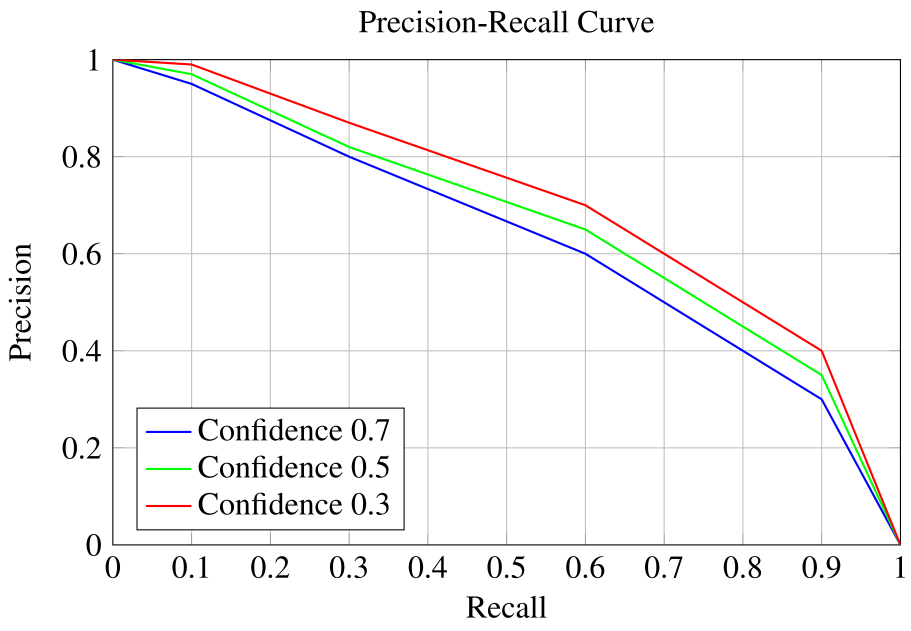


Figure 2.6: Example of a Precision-Recall Curve for three confidence thresholds.

Average Precision and Mean Average Precision

The Average Precision (AP) evaluates object detection performance as the area under the precision-recall curve [24, 52]. It is calculated as:

$$\text{AP} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \text{Precision}(r). \tag{2.6}$$

Instead of integration, precision is averaged at eleven recall levels to ensure a monotonic curve [14]:

$$\text{Precision}(r) = \max_{r' \geqslant r} \text{Precision}(r'). \tag{2.7}$$

The mean Average Precision (mAP) is calculated as the mean of the AP values across all classes [46]:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i, \tag{2.8}$$

where N is the total number of classes, and AP_i is the average precision for the i-th class. A higher mAP value indicates better performance of the object detection model. The IoU thresholds (e.g., mAP@0.5, mAP@[0.5:0.95]) further refine detection accuracy [14], where IoU values determine correct detections.

2.3 “CUT-PASTE” METHOD

The generation of synthetic data can be achieved through a variety of approaches. The method outlined in this thesis, referred to as the “Cut-Paste” method, involves cutting objects from one image and pasting them into another image [9, 12, 19]. In the majority of cases, the objects which are pasted into the new image are named the foreground, while the image into which the objects are pasted is named the background. The objects are first cut-out using a binary mask, which defines the boundaries of the objects. They are then pasted into new images, which serve as the background. This method allows for the generation of large datasets with minimal effort and can be used to create diverse training data for object detection models. The “Cut-Paste” method can mathematically be represented as:

$$I_{\text{final}}(x, y) = \begin{cases} I_{\text{foreground}}(x, y), & \text{if } (x, y) \in \text{foreground} \\ I_{\text{background}}(x, y), & \text{otherwise} \end{cases}, \tag{2.9}$$

where $I_{\text{foreground}}$ and $I_{\text{background}}$ are the pixel values of the foreground and background images, respectively.

2.4 BLENDING

Blending is the process of seamlessly integriating the foreground elements into the background environment, to prevent the emergence of pixel artifacts. The blending of objects into new images can be challenging due to the differences in lighting, color, and texture between the foreground and background images. Dwibedi et al. [12] and Giakoumoglou et al. [23] realised, that this can result in visible artifacts in the final image, which can negatively impact the performance of the model. To address this issue, traditional visual computing techniques, such as blending, can be used.

To achieve this, different blending methods can be used. The most common blending methods in the context of synthetic data generation are Alpha blending, Gaussian blending, Poisson blending. For blending of package boxes, approaches also use motion blurring [51]. Pyramid blending is used less often, but can also be used for blending objects into new images [19]. An example for the finished blending methods, in the case of this work, can be seen in Figure 5.1. The objects are outlined with their ground truth bounding boxes.

Other approaches, like motion blurring, can also be used to blend objects into new images [51]. Given the infrequency with which these methods are employed, this paper is constrained to the aforementioned methods.

2.4.1 ALPHA BLENDING

Alpha blending is a widely used and simple blending technique that combines the foreground and background images by assigning a transparency value to each pixel in the foreground image [12, 22]. This transparency value, known as the alpha channel (or alpha mask), determines the opacity of the pixel, allowing the background to show through. The alpha channel can either be binary or continuous, depending on the complexity of the blending effect required. Binary alpha channels are a value of 0 **or** 1, where 0 represents the background and 1 represents the object [17]. Continuous alpha channels are grayscale images with pixel values between 0 **and** 1, where 0 is transparent and 1 is opaque [55].

For a binary mask, the alpha channel is defined by Ghiasi et al. [22] as:

$$\alpha(x, y) = \begin{cases} 1, & \text{if } (x, y) \in \text{object} \\ 0, & \text{otherwise} \end{cases}. \tag{2.10}$$

An example of a mask can be seen in Figure 2.7a, where the object is represented by white pixels and the background by black pixels.

To calc the final image, the alpha channel is used to blend the object into the background image and compute the final image. The final formula for a two-dimensional image, can be expressed based on a rewritten formula from Zanella et al. [74]:

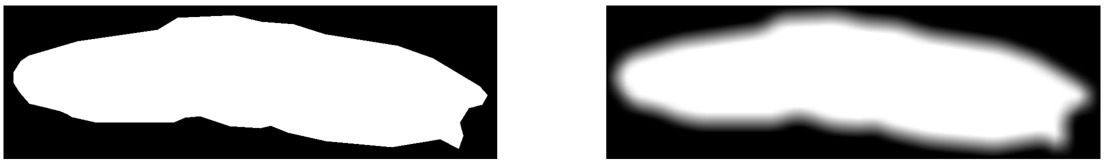
$$I_{\text{final}}(x, y) = I_f(x, y) \cdot \alpha(x, y) + I_b(x, y) \cdot (1 - \alpha(x, y)), \tag{2.11}$$

where (x, y) are the pixel coordinates of the image, and I_f and I_b are the pixel values of the foreground and background images, respectively. When using a binary mask, the final image is the pixel value of the foreground image if the mask is 1, and the pixel value of the background image if the mask is 0. For continuous alpha channels, the final image is a linear combination of the foreground and background images, weighted by the alpha channel. This method is computationally efficient and widely used for blending in synthetic dataset generation [12].

Sometimes the Alpha blending method is called *no blending* [12], linear combination [74] or *trivial blending* [19], because it is the simplest blending method. An example of Alpha blending can be seen in Figure 5.1a.

2.4.2 GAUSSIAN BLENDING

Gaussian blending smooths the boundaries of the pasted object by applying a Gaussian filter to the region of interest. Rather than being a standalone blending method, it functions as a blurring technique that is often used in combination with Alpha blending. For example, the mask responsible for the blending effect is smoothed using Gaussian blending, as seen in Figure 2.7b. This makes the edges of the mask more smooth.



(a) Mask (b) Blurred Mask

Figure 2.7: Example of Gaussian blending. (a) Mask with sharp edges. (b) Blurred mask with smoothed edges. The blurred mask is then used for blending the object into the background, for example with Alpha blending.

The Gaussian filter is defined by Lindeberg [45]:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.12}$$

where σ is the standard deviation of the Gaussian distribution, and x and y are the pixel coordinates. By convolving the object edges with the Gaussian kernel, this method reduces sharp transitions and integrates the object more naturally into the background. This technique is widely used in computer vision and image processing to enhance the visual quality of composite images [12].

2.4.3 POISSON BLENDING

Poisson blending [54] solves the problem of seamless integration by ensuring that the gradient of the composite image matches the gradient of the foreground object while blending it with the background. This approach adjusts the pixel intensities in the overlapping region to match the surrounding background, ensuring smooth transitions. The Poisson equation is defined as:

$$\Delta f = \nabla \cdot \mathbf{v}, \tag{2.13}$$

where f is the function representing the pixel intensities in the region of interest, Δ is the Laplacian operator, and $\mathbf{v}$ is the guidance vector field derived from the gradients of the source image [28, 54]. The boundary conditions are given by the pixel values of the target image on the boundary of the region. Therefore it is used by [12, 21, 23, 51]. Despite its high visual quality, Poisson blending is computationally expensive and may struggle with large intensity differences between foreground and background images [28].

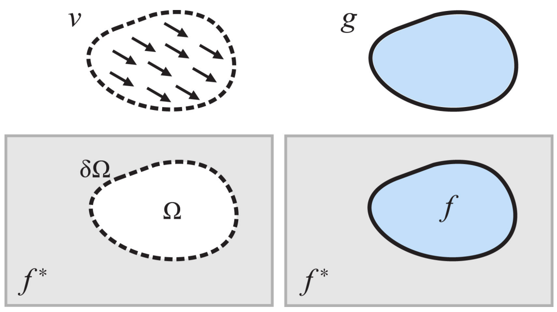
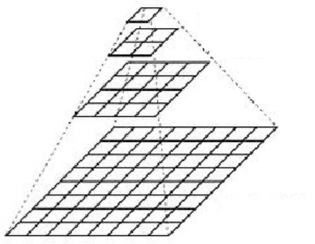


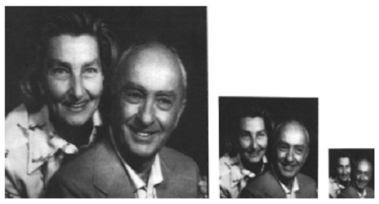
Figure 2.8: Poisson blending [54] positions a source patch g, along with its associated vector field v and support region Ω, onto a target background f^*. The boundary conditions $\delta\Omega$ are set to match the values of the target image. To achieve a seamless integration, f^* is composited with f, which is a modified version of g, computed by the algorithm. Source: [28]

2.4.4 PYRAMID BLENDING

Pyramid blending [1, 42] is an image blending technique designed to seamlessly combine two images by utilizing multi-scale representations. It addresses challenges such as visible seams, mismatched colors, and inconsistent textures that arise when merging images directly.

For this method, image pyramids are used to decompose the images into multiple levels of detail. Image pyramids are hierarchical representations of an image that capture different scales of the image, from low-frequency components to high-frequency details [1]. An illustration can seen in Figure 2.9 (a). The Gaussian pyramid captures the low-frequency components of the image, while the Laplacian pyramid preserves the high-frequency details [42].


(a) Pyramid representation


(b) Gaussian pyramid

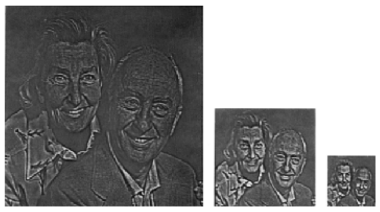

(c) Laplacian Pyramid

Figure 2.9: An example of Gaussian and Laplacian pyramids from the same input image: (a) is a visual representation of an image pyramid; (b) shows the first three levels of Gaussian pyramid; (c) shows the first three levels of Laplacian pyramid. Adapted from: [1]

First, the Gaussian pyramid for the mask and its complement is created [42]. This is done by repeatedly applying a Gaussian blur to the mask and downsampling it. This process captures the low-frequency components of the mask. An example of a Gaussian pyramid can be seen in Figure 2.9 (b).

Next, Laplacian pyramids are derived by computing the difference between successive levels of the Gaussian pyramid, preserving high-frequency details such as edges and textures [42]. This step needs the Gaussian pyramid for the images, but only as a step to get the Laplacian pyramid. A Laplacian pyramid can be calculated as the difference between the Gaussian pyramid and the upsampled version of the next level of the Gaussian pyramid [1]. The Laplacian pyramid is computed as:

$$L_i = G_i - \text{upsample}(G_{i+1}), \tag{2.14}$$

where L_i is the current level of the Laplacian pyramid, G_i is the current level of the Gaussian pyramid of the image, and $\text{upsample}(G_{i+1})$ is the upsampled version of the next level of the Gaussian pyramid. An example of a Laplacian pyramid can be seen in Figure 2.9 (c).

After that, the Laplacian components of the two images are combined using the mask's Gaussian levels as weights [42]. Mathematically, this step can be expressed as [42]:

$$L^k_{\text{blend}} = M^k \cdot L^k_{\text{image1}} + (1 - M^k) \cdot L^k_{\text{image2}}, \tag{2.15}$$

where L^k_{blend} is the blended Laplacian at level k, M^k is the Gaussian mask at level k, and L^k_{image1} and L^k_{image2} are the Laplacians of the two images at level k.

Finally, the blended Laplacian pyramid is collapsed to obtain the final blended image, starting from the smallest level of the pyramid and iteratively adding the upsampled Laplacian components [42].

2.5 SYNTHETIC-TO-REALITY GAP

This section introduces the concept of the domain gap and the synthetic-to-reality gap, followed by an explanation of how the synthetic-to-reality gap can be quantified.

The domain gap is a general term that describes the difference between two datasets, including the difference between various real-world datasets [56]. For instance, the difference between the domains in Figure 1.1, where one image is taken in a muddy environment and the other in a clear environment, represents a domain gap.

The synthetic-to-reality gap is a specific type of domain gap that describes the difference between synthetic and real data [23]. For instance, the difference between a synthetic image of a tardigrade, generated with the "Cut-Paste" method, and a real image of a tardigrade, represents the synthetic-to-reality gap. Other names for the synthetic-to-reality gap include the simulation-to-reality gap, the Syn2Real gap and reality gap [8, 56].

As Alhaija et al. [2] pointed out, "the difference in data distribution and pixel-value statistics between the real and virtual data prevents it from being a direct replacement to real training data" [2, p. 8]. This describes that the synthetic-to-reality gap is a challenge in the field of synthetic data generation, as the synthetic data generated by a model is often not realistic enough to be used as a direct replacement for real data [23].

2.5.1 QUANTIFICATION

The synthetic-to-reality gap can be quantified using various evaluation metrics [8]. On the one hand, the performance of object detection models can be used to assess the synthetic-to-reality gap [9]. On the other hand, image quality metrics, such as FID and CMMD, can be used to measure the similarity between synthetic and real images [5, 29]. To understand the calculation of the FID and CMMD metrics, the concept of feature embeddings is introduced, as they are used to compute the similarity between real and synthetic images.

Embeddings can be defined as mathematical representations of objects in a continuous vector space [3]. They are utilised to visualise the semantic similarities between objects using numerical distances. In the context of image analysis, feature embeddings can be considered as high-dimensional representations of data that capture features or patterns in the data [41]. These embeddings are frequently extracted from deep neural networks, such as CNNs that have been pre-trained on extensive datasets, as referenced by Kiela and Bottou [41] and Radford et al. [57].

For instance, the CLIP [57] model is trained on a large-scale dataset of 400 million images with associated textual descriptions. The model learns to map images and text into a shared embedding space, where the similarity between images and text is measured using their embeddings. As seen in Figure 2.10 (1), the Image and Text are both

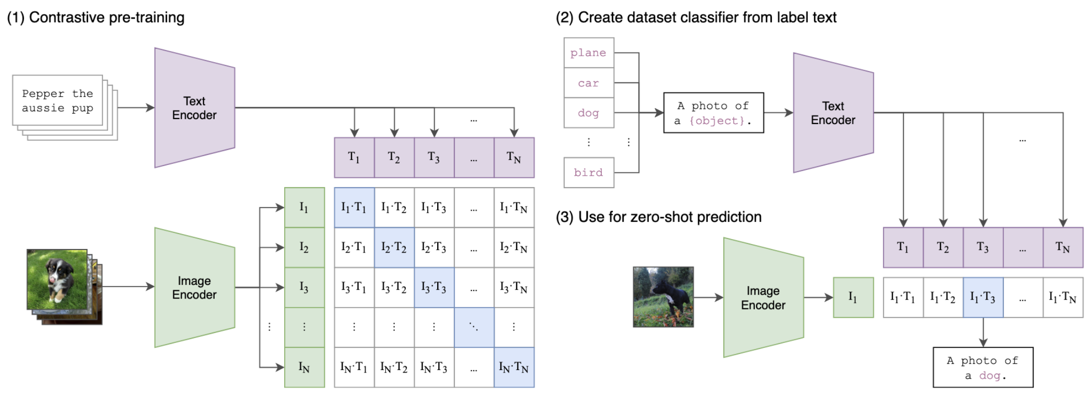


Figure 2.10: CLIP Embeddings. (1) Generating the embeddings for images and text. (2) Generating the encodings for the text classes. (3) Generating the encodings for the image classes. (2) and (3) are transformed into the shared embedding space, where the similarity between the image and text embeddings is computed. Source: [57]

encoded using a Text and Image Encoder, respectively. They are then projected into a shared embedding space, where the similarity between the image and text embeddings is computed [57]. The model is trained to maximize the similarity between corresponding image-text pairs and minimize the similarity between non-corresponding pairs. In the prediction process, seen in Figure 2.10 (2) and (3), the model uses the classes from the text encoder to predict the image embeddings and vice versa. The similarity between the predicted embeddings is then used to determine the correspondence between the image and text.

Object Detection Performance Using object detection performance to quantify the synthetic-to-reality gap is a common approach in the field of computer vision [8]. For example, multiple object detection models are trained separately on synthetic and real data and their performances are compared to measure the synthetic-to-reality gap [8].

In this thesis, this will be achieved by training multiple YOLOv11 object detection models on synthetic and real data, as outlined in chapter 4 and chapter 5. The performance, calculated as the mAP, will then be employed to quantify the synthetic-to-reality gap based on the difference in performance between the two domains.

Another approach to quantify the synthetic-to-reality gap is to use image quality metrics, such as FID and CMMD [5]. These two metrics were selected on the basis of section 3.2 and can also be used to quantify the synthetic-to-reality gap of synthetic images. In contrast to the requirement for object detection performance to be trained directly on the data, image quality metrics can be achieved through the use of pre-trained models. These models extract feature embeddings from the images, which are then utilised to compare and quantify the synthetic-to-reality gap [5].

Fréchet Inception Distance The Fréchet Inception Distance (FID) [29] is a widely used metric to evaluate the quality of images generated by models such as Generative Adversarial Networks (GANs). FID measures the similarity between the distributions of real and generated images by comparing their statistical properties, specifically their feature embeddings. It is based on the idea that high-quality generated images should have feature distributions that are similar to those of real images. FID utilizes feature representations extracted from a pre-trained Inception-v3 network, which is a deep convolutional neural network trained on the ImageNet dataset [29].

The FID score is calculated by first computing the means μ_P, μ_Q and covariances Σ_P, Σ_Q of the feature distributions for real and generated images, respectively. The P represents the distribution of real images and Q the distribution of generated images. These statistical moments are then used to compute the Fréchet distance (also known as the squared Wasserstein-2 distance) between the two multivariate Gaussian distributions. The formula for the FID score is given by [11, 29, 36]:

$$\mathrm{dist}_F^2(P, Q) = \|\mu_P - \mu_Q\|_2^2 + \mathrm{Tr}\left(\Sigma_P + \Sigma_Q - 2(\Sigma_P \Sigma_Q)^{1/2}\right), \tag{2.16}$$

where μ_P and μ_Q are the means, and Σ_P and Σ_Q are the covariances of the real and generated image distributions, respectively. The term $\|\mu_P - \mu_Q\|_2^2$ measures the squared Euclidean distance between the means of the two distributions. The term $\mathrm{Tr}\left(\Sigma_P + \Sigma_Q - 2(\Sigma_P \Sigma_Q)^{1/2}\right)$ uses the trace operator Tr to quantify the difference in the covariance structures of the distributions, which sums the diagonal elements of a matrix A as $\mathrm{Tr}(A) = \sum_i a_{ii}$ [16]. A lower FID score indicates that the generated images are closer to the real images in terms of both their means and covariances.

While FID is widely used, it makes the assumption that the feature representations of real and generated images follow multivariate Gaussian distributions [11]. This assumption may not always hold in practice, especially when the real and generated image distributions are complex or multimodal [36]. As a result, the FID score may fail to capture the full range of differences between the two distributions in some cases.

CLIP Maximum Mean Discrepancy The CLIP Maximum Mean Discrepancy (CMMD) is a metric designed to address certain limitations of the FID. CMMD uses feature embeddings from the above-mentioned CLIP model to quantify the similarity between real and generated images [36]. Unlike FID, CMMD measures the discrepancy between the distributions of real and generated images without assuming any specific statistical distribution. It leverages the Maximum Mean Discrepancy, which quantifies the distance between two probability distributions based on their feature embeddings. The formular for CMMD, where P represents the distribution of real images and Q the distribution of generated images, is given by [36]:

$$\mathrm{dist}^2(P, Q) = \mathbb{E}_{x,x' \sim P}[k(x, x')] + \mathbb{E}_{y,y' \sim Q}[k(y, y')] - 2\mathbb{E}_{x \sim P, y \sim Q}[k(x, y)], \tag{2.17}$$

where x are samples from the real image distribution P and y are samples from the generated image distribution Q. The $\mathbb{E}$ terms represent the expectation over the distributions of real and generated images, respectively. More specifically, the three terms in the equation represent the following:

- $\mathbb{E}_{x,x' \sim P}[k(x, x')]$: Similarity between pairs of real images, where x and x' are samples from the real image distribution P.
- $\mathbb{E}_{y,y' \sim Q}[k(y, y')]$: Similarity between pairs of generated images, where y and y' are samples from the generated image distribution Q.
- $\mathbb{E}_{x \sim P, y \sim Q}[k(x, y)]$: Similarity between real and generated images, where x is a sample from the real image distribution P and y is a sample from the generated image distribution Q.

To calculate the similarity between the feature representations of images, the Radial Basis Function (RBF) kernel is used, defined as [36]:

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right), \tag{2.18}$$

where σ is a hyperparameter that controls the bandwidth of the kernel. In the case of CMMD, it is set to 10 [36].

3 RELATED WORK

In this chapter, the synthetic data generation, the synthetic-to-reality gap and blending methods in the context of this thesis are discussed.

The chapter begins by introducing the “Cut-Paste” method as a common approach to generating synthetic data. For the sake of completeness, other methods, such as 3D models or machine learning-based methods, are also discussed briefly. This chapter thus establishes the “Cut-Paste” method as the most suitable approach for the use case of this work.

After that, the importance of the realism of the synthetic data will be examined, with a particular focus on its potential to reduce the synthetic-to-reality gap. Additionally, methodologies for quantifying this gap will be explored, along with additional metrics relevant to its measurement.

Finally, the discussion will move on to the various blending methods that have been developed for the purpose of combining synthetic and real images. This will also be discussed in the context of the “Cut-Paste” method and reducing the synthetic-to-reality gap.

3.1 SYNTHETIC DATA GENERATION

Different methods can be used to generate synthetic data. One common approach is to use 3D models or Computer-Aided Design (CAD) models to create images. Another approach is to use the “Cut-Paste” method to combine objects from different images. Additionally, there are machine learning (ML)-based approaches that attempt to generate images that appear realistic, for instance by using Generative Adversarial Networks (GANs) [25].

3D and CAD Methods Using 3D models or CAD based approaches, objects of interest are rendered in a 3D environment and then projected onto a 2D plane. This facilitates the generation of images from different viewpoints and with different lighting conditions. In contrast to the “Cut-Paste” method, 3D models are more complex to generate. They require to have models of the objects, which can be hard to get or to manually create. Furthermore, this often requires domain knowledge [15] to reduce the synthetic-to-reality gap [9].

Several studies have explored different ways to generate synthetic training data using 3D rendering. Hinterstoisser et al. [30] used OpenGL with Phong Shading and Gaussian noise to integrate 3D CAD models into background images. Johnson-Roberson et al. [37] demonstrated that video game environments, such as GTA V, can be used to generate training data for real-world applications. Prakash et al. [56] introduced structured domain randomization (SDR), maintaining scene context while varying object placements. Hybrid approaches, such as Alhaija et al. [2], combine real imagery with rendered objects for greater realism. In drone research, Dieter et al. [8] leveraged Unreal Engine and Microsoft AirSim to generate synthetic datasets, highlighting the risk of overfitting when relying solely on real data.

"Cut-Paste" Method The "Cut-Paste" method, as introduced in section 2.3, is easier to implement than 3D models and can be used with less computational power [4]. Especially for the use case of this work, under a microscope, the boundary conditions from Dirr et al. [9], where "the flat and slim parts typically have a small number of preferred orientations around the x- and y-axis, so that fewer perspective views of the parts occur during the application" [9, p. 8], are met. Therefore, the "Cut-Paste" method is a suitable approach for this work.

In their work, Dwibedi et al. [12] propose to train an object detector using only synthetic images generated with the "Cut-Paste" method. The objects they tried to detect were kitchenware. The objective was to detect distinct objects from a variety of viewpoints. They gained up to 21% in relative performance when combined with real images. They used a model pre-trained on the MSCOCO [44] dataset. The primary focus of the study centred on the methodologies employed for blending. Additionally, the researchers implemented diverse splits between real and synthetic data. Already with a mix of 10% real and synthetic data they observed a performance increase about 10 AP. The results indicate that the model exhibits superior performance in domain generalization. Additionally, the researchers posited that rendering with 3D models is not always an optimal approach, as the model may encounter difficulties generalizing to real data due to the alteration in image statistics. Furthermore, they employed data augmentation, which yielded gains in AP ranging from 3 to 10 per method. In their data augmentation strategy, they utilized 2D and 3D rotation, occlusion, truncation, and the introduction of distractor objects into the scene. They used 30%, 60%, and 90% and 100% of synthetic data in their experiments. A model trained exclusively on real data was used for comparison.

In addition, the study by Georgakis et al. [21] also addressed the "Cut-Paste" method, but they focused on the semantic and geometric context of the scene. For instance, the synthetic objects were positioned on tables. Furthermore, the scale of the objects was taken into account in relation to their depth within the scene. Blending was also employed as a technique. The approach yielded results that were nearly on par with those obtained from the real data. The researchers suggested that the semantic and geometric context of the scene plays an important role in the effectiveness of the method. They used 0%, 50%, 90% and 99% of synthetic data in their experiments. In addition, a comparison was made with a model trained exclusively on real data.

Khalil et al. [39] investigated whether synthetically generated images can improve the PASCAL VOC [14] image dataset performance. They compared the training with only the original data, with a fully generated dataset and also a mixed dataset. They showed that the synthetically generated images gained a similar performance as the training with original data. Their experiment suggests that "Cut-Paste" methods can be effectively used as a data augmentation technique when an accurate segmentation of the object is available. Furthermore, the researchers determined that the context is a crucial factor. A transition to backgrounds that are markedly distinct from those in the data set is inadvisable. This was observed in the "Bird" class, where the background was predominantly blue. Other backgrounds in this class introduced noise and were therefore unsuitable. Also the scale, illumination and the ratio between background and foreground proved to be important.

In his empirical study, Arcidiacono [4] considered various techniques for synthetic image generation, including those based on the “Cut-Paste” and CAD method, and analysed the performance increase between them. He used 20 times more synthetic data than real data, as in the study by Alhaija et al. [2], and the Open Image Dataset as background images. In his study, he observed that the performance of the model trained on synthetic data generated from CAD models outperformed that of the model trained on “Cut-Paste” data. However, it was also noted that when a limited dataset is available, the “Cut-Paste” method is sufficient and more straightforward to implement.

In their work, Dirr et al. [9] posit that the “Cut-Paste” technique is to be preferred to 3D modelling, on the grounds that the images are derived from a comparable environment, thereby reducing the discrepancy between the virtual and the real. Image augmentation was employed for two distinct purposes: initially, to modify the pre-existing objects; and subsequently, in conjunction with the newly generated images following their incorporation. The experiment was designed for flat and slim parts, with a ratio of 5:1, and the camera was placed in the top view. Furthermore, it was stated that the flat and slim parts typically exhibit a limited range of preferred orientations with respect to the x- and y-axis. Consequently, fewer perspective views of the parts are generated during the application process. This results in a significant reduction in the number of source images compared to existing techniques, where a multitude of views on volume objects are mapped in the source images.

As an approach, Naumann et al. [51] developed a complete pipeline, which also includes scraping the images from the web. The remaining “Cut-Paste” is based on [12], but the authors neglected all effects to make the images realistic. Consequently, they used no blending at all. For their approach, they pasted $1-4$ objects onto the background images and used $2-4$ distractor objects. Furthermore, they employed the use of augmentation techniques, including translation, rotation, and scaling. During the scaling of the objects, the researchers paid attention to maintaining a range between 15% and 40% of the entire image. Additionally, they applied a maximum of 20% upscaling and an intersection with other objects of a maximum of 0.5 IoU.

Garcia-Peraza-Herrera et al. [19] focused on the generation of synthetic data for the detection of surgical tools, such as endoscopes. These tools were subjected to random zooming, rotation, vertical and horizontal flipping and shifting, with all operations performed while maintaining connection to the image border. Augmentations were also applied to the background images, including horizontal and vertical flips, as well as to the blending images, such as cutouts, noise and blur. They used 1-3 objects and distractor objects, as well as a 0.5 IoU threshold. The conclusion drawn was that training with synthetic data is as effective as training with real data.

Furthermore, no work in the literature has been found using the “Cut-Paste” method for synthetic data generation in the context of microorganisms. This is why it should be evaluated in this work.

Other Methods Additionally, there are machine learning (ML)-based approaches that attempt to generate images that appear realistic. These methods are often based on Generative Adversarial Networks (GANs) [25], which can be used to generate images from text or other data. However, these methods require a lot of computational power and are more complex to implement than the “Cut-Paste” method.

Lin et al. [43] demonstrated the use of text-to-image GANs to create synthetic images for object detection. Similarly, Wang et al. [71] and Sixt et al. [65] used GANs to generate realistic images, such as license plates and barcode-like markers for honeybees, improving object detector performance. GANs can also be combined with other methods, as seen in Remez et al. [60], who integrated them with the “Cut-Paste” method, and Peng and Saenko [53], who combined them with 3D CAD models.

While generative methods can produce highly realistic images, they require significant computational power and are often unstable to train [23]. In contrast, the “Cut-Paste” method is more efficient and easier to implement, making it the focus of this work.

3.2 SYNTHETIC-TO-REALITY GAP

To further analyze the research questions, first, the importance of synthetic data realism in improving model performance on real data is discussed. Second, methods for quantifying the synthetic-to-reality gap are explored, along with additional metrics relevant to its measurement. Finally, the significance of the ratio between real and synthetic data in optimizing model performance is discussed.

Su et al. [66] demonstrated the importance of synthetic data realism in improving model performance on real data. Their work on object viewpoint estimation showed improved results when training data included real background textures and varying lighting conditions. Similarly, Hinterstoisser et al. [30] emphasized the importance of patch-level realism, referring to the realistic appearance of the content within the bounding box framing the rendered object, which is essential for effective model performance on real-world data. This relates directly to the blending methods discussed in section 3.3. Additionally, Prakash et al. [56] highlighted that discrepancies in texture and lighting between real and synthetic data can create a domain gap, significantly influencing model performance.

Furthermore, Hinterstoisser et al. [30] pointed out that 3D rendering approaches suffer from a domain gap between real and synthetic data. To mitigate this issue, they recommend the “Cut-Paste” method, as it reduces the domain gap since the objects are already embedded in the domain of real images. This is due to the fact that the objects are extracted from real images, reducing the difference between the synthetic and real data.

3.2.1 QUANTIFICATION

To quantify the synthetic-to-reality gap, various approaches exist. The different methods for quantifying the synthetic-to-reality gap has been introduced in subsection 2.5.1.

Object Detection Performance Using object detection models to quantify the synthetic-to-reality gap is used by Dirr et al. [9], Dwibedi et al. [12], and Prakash et al. [56]. These models are trained separately on synthetic and real data, and their performances are compared to measure the synthetic-to-reality gap. A model that performs equally well on both domains exhibits a low synthetic-to-reality gap, whereas a model that performs significantly better on real data indicates a higher synthetic-to-reality gap [56].

Image Quality Metrics Another approach to quantifying the synthetic-to-reality gap relies on image quality metrics such as the FID and CMMD, which were described in more detail in subsection 2.5.1. Originally designed to assess the quality of images generated by models like Generative Adversarial Networks (GANs), these metrics can also be applied to synthetic images generated by alternative methods [5], such as the "Cut-Paste" method.

Furthermore, the FID and CMMD are not the only available metrics for quantifying the synthetic-to-reality gap. The Inception Score (IS) [63], for example, is another widely used metric for evaluating the quality of synthetic images. The issue with IS is that it does not consider real-world data [5], thereby reducing the applicability of the score to the experiments that were intended in the experiments of this thesis. While FID and IS are considered standard for synthetic image evaluation, CMMD is a more recent metric that has been shown to outperform FID in certain cases [36]. Therefore, Jayasumana et al. [36] suggest that CMMD may be more robust than FID, as it does not assume any specific statistical distribution for the feature representations of images.

In consideration of the aforementioned factors, this thesis employs in total three metrics to quantify the synthetic-to-reality gap. On the one hand, the object detection performance is evaluated using the mAP score, which is further described in chapter 4 and chapter 5. On the other hand, the FID and CMMD metrics are additionally used to quantify the synthetic-to-reality gap of the synthetic images generated by the "Cut-Paste" method.

In conclusion, the synthetic-to-reality gap is a challenge in synthetic data generation, as the realism of synthetic data significantly impacts model performance on real data. To address this issue, it is essential to generate synthetic images that are photorealistic and indistinguishable from real images. This objective can be achieved by employing blending methods to merge synthetic images with real images, creating more photorealistic results [12].

3.3 BLENDING

The impact of blending on the realism of synthetic images is significant, as it determines the extent to which the object is integrated into the background. This, in turn, can affect the synthetic-to-reality gap, as previously discussed in section 3.2. This section reviews related work on blending methods used in synthetic dataset generation, analysing their impact on model performance.

Dwibedi et al. [12] compared various composing methods for their approach, including Alpha blending, Gaussian blending, and Poisson blending. They propose to synthesize every training image several times, using the same objects and background, but a different method to blend the objects into the background. Their findings demonstrated that using all blending methods simultaneously led to an 8 AP increase in performance metrics. They attributed this improvement to enhanced robustness against pixel artifacts, as the model encountered the same image with varying blending methods during training.

Giakoumoglou et al. [23] propose an approach to generate synthetic datasets for object detection that requires only a small dataset of target objects and a larger background dataset that fits the desired environment. The distinction between this approach and that described in [12] lies in the generation of objects from random noise rather than the simple pasting of objects from the foreground to the background. This allows for greater diversity and flexibility in the generated images. They used Gaussian and Poisson blending. As model, they compared different Yolo versions, all pre-trained with the MSCOCO [44] dataset. They also use geometical transformations, like translation and color transformation.

Ghiasi et al. [22] used Alpha blending and smoothed the edges of the alpha mask with a Gaussian filter and no geometical transformations. An important difference to other approaches, is that they used images, in which other objects are already included. Unlike [12], they found that simply composing without any blending has similar performance to blendig with Gaussian blending. They found that it is very effective and robust. It also performs well across multiple experimental settings and provides significant improvements on top of strong baselines. In contrast to this, [51] used Alpha blending, Gaussian blending, Poisson blending and Motion bluring for their "Cut-Paste" method. As [12] they generated a image for each blending method. Therefore, from a single foreground and background image, four images were generated, with each image employing a distinct blending method. They compared themself directly with [22], leading the conclusion, that blending is important for synthetic images, in their case. Nevertheless, as the Motion bluring is specifically designed for the purpose of simulating motion blur, it is not further addressed in this thesis.

Dirr et al. [9] and Zanella et al. [74] utilized Gaussian blending as the sole method for their cp approach. Their studies focused exclusively on this blending technique, without exploring or comparing alternative blending methods. This decision highlights a reliance on Gaussian blending as a standard approach but leaves the effectiveness of other blending methods unexplored.

In contrast, Giakoumoglou et al. [23], along with Naumann et al. [51] and Dwibedi et al. [12], used multiple blending methods in their studies. This approach allowed them to evaluate the impact of different blending techniques on the performance of their "Cut-Paste" pipeline, providing a more nuanced analysis of blending methods in synthetic data generation.

In the study by Garcia-Peraza-Herrera et al. [19], the researchers employed a range of blending methods, including Alpha blending, Gaussian blending and Pyramid blending. The experimental design involved individual testing of each blending method, as well as a combination of all methods. Notably, the study incorporated a novel approach, wherein a mixed blending of all methods was applied to the same image, referred to as Mixed Blending. The findings revealed that the performance of Pyramid blending was superior to that of the other individual blending methods, as well as the multiple blending methods from [12]. However, it was observed that Mixed Blending exhibited superior performance in comparison to Pyramid blending.

Summarizing, Alpha blending and Gaussian blending are among the most commonly used techniques due to their computational efficiency and effectiveness in blending objects into backgrounds [12, 22, 23, 51]. Poisson blending is used less frequently but has been explored in some puplications [12, 23]. Some approaches have used multiple blending methods simultaneously to enhance the robustness of the synthetic data [12, 23, 51].

Motion blur has been employed in specific cases [51], but as it is primarily intended to simulate motion artifacts rather than blending objects naturally, it is not considered further in this work. Instead, Pyramid blending has been incorporated due to its promising results in improving image realism and model performance [19].

4 APPROACH

To evaluate the “Cut-Paste” method and the different blending methods, an empirical approach is used. A pipeline has been designed to facilitate the extraction of objects from images, their augmentation, and subsequent integration into background images. This pipeline draws upon the methodologies outlined in [9, 12]. The pipeline is then utilised to generate synthetic images, which are employed for the training of the YOLOv11 model. The efficacy of the model is evaluated using different test data, and the results are compared to the baseline test results.

The mAP values are compared to quantify the synthetic-to-reality gap between the baseline test and the tests with the synthetic data. Additionally, these results are compared to the FID and CMMD metrics to further validate these findings. The mathematical calculation of the FID and CMMD has been introduced in subsection 2.5.1 and at the end of this chapter, in section 4.5, the adapted code for this thesis is described.

For the self-trained YOLOv11 model, the code is written in Python and uses the OpenCV library [6] for image processing. Additionally, the code is written to export the images in the YOLO format to train the YOLOv11 model. The entire process can be algorithmically described as follows, based on [12, 23, 51]:

Algorithm 1 Randomized Image Composition with Blending

```
Require: Foreground images set $F$, Background images set $B$, Transformation library
    $T$, Blending methods $BM$, number of foregorund images $n$
Ensure: Final composite image $I_{\text{composite}}$, YOLO $\mathrm{bboxes}$
 1: Randomly select $n$ foreground images $\{F_1, F_2, \dots, F_n\} \subset F$
 2: Randomly select one background image $B_{bg} \in B$
 3: Initialize $\mathrm{bboxes} \leftarrow \emptyset$
 4: for each foreground image $F_i$ do
 5:     Extract the polygon region from $F_i$ {Set non-polygon pixels to transparent}
 6:     Apply random augmentations to $F_i$ using $T$
 7:     Compute bounding box $\mathrm{bbox}$ from the polygon
 8:     Add $\mathrm{bbox}$ to $\mathrm{bboxes}$
 9: end for
10: Check for object occlusions or overlaps in $\mathrm{bboxes}$ and resolve them
11: Initialize $I_{\text{composite}} \leftarrow B_{bg}$
12: for each foreground image $F_i$ do
13:     Blend $F_i$ with $I_{\text{composite}}$ using blending method $bm \in BM$
14: end for
15: return $I_{\text{composite}}$, $\mathrm{bboxes}$
```

where the foreground images set F contains the objects, the background images set B contains the background images, the transformation library T contains the augmentation methods, the blending methods BM contains the different blending methods,

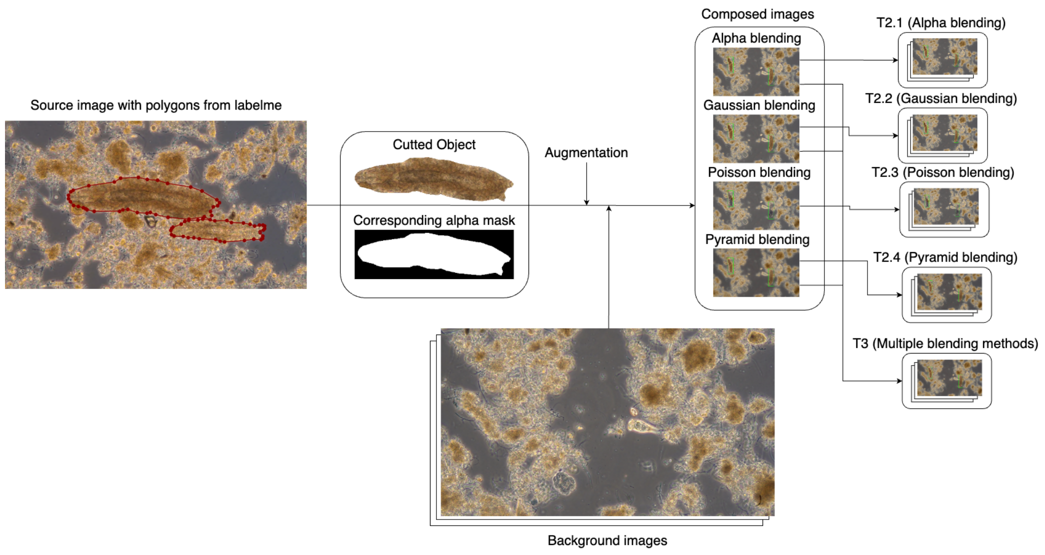


Figure 4.1: Basic procedure of the "Cut-Paste" approach. From the objects, which are cut-out via polygons, the alpha mask is extracted. The objects are then augmented and pasted into the background images using different blending methods. Adapted from: [9]

and n is the number of foreground images to pick. First, n images from the F dataset and one random background image from the B dataset are picked. After that, for each picked foreground image, the object is cut-out based on a annotated polygon, which borders the object, as described in section 4.1. Thereby the bounding box of the object is computed. Furthermore random transformations, as described in section 4.2, are applied to the objects. After that, the objects are checked for occlusions, as described in subsection 4.2.1. Then the objects are blended into the background image using the different blending methods, as described in section 4.3. The final image $F_{\text{composite}}$ and the bounding boxes of the objects, which are needed for the training of the object detection model, are returned. A visualization of the procedure is shown in Figure 4.1.

The following sections describe the individual steps of the pipeline in more detail.

4.1 CUT OBJECTS

First, the images containing the objects are masked with a polygon. These polygons are created manually with LabelMe [70], an annotation tool for images. To simplify the annotation process, EfficientSam [73] is integrated in LabelMe, which can be used to automatically create polygons around objects in images. As in the study by Garcia-Peraza-Herrera et al. [19], the polygons are manually validated to ensure that the objects are correctly cut-out, as this step is crucial for the entire process. As discussed in the study by Dirr et al. [9], the alpha masks from the objects can then be extracted once they have been cut-out. This mask can then be utilised for the blending of the objects into the background images, as demonstrated in section 4.3. The objects are saved as PNG images, accompanied by the relevant alpha channel. The aforementioned process can be delineated through the following algorithm:

Algorithm 2 Cutting out the objects

```
Require: Image $I_{foreground}$, Polygon $P$
Ensure: Object with alpha channel $I_{object}$
1: Create a binary mask $M$ from the polygon $P$
2: Create an alpha channel $A$ from the mask $M$
3: Create a new image $I_{object}$ with the alpha channel $A$
4: return $I_{object}$
```

where the image $I_{foreground}$ is the image with the object, and the polygon P is the polygon around the object. The binary mask M is created from the polygon, and the alpha channel A is created from the mask. The object image I_{object} is then created with the alpha channel A.

4.2 AUGMENTATION

Before pasting the objects into the background images, the objects are augmented to generate more diverse synthetic images, as by [9, 23, 51]. The process involves a series of geometric transformations, including rotations, translations, and scalings, which collectively generate a position for the object within the background image. The augmentation is done with the Albumentations library [7], which is a library for image augmentation in computer vision tasks. The used transformations can be configured in a YAML file. To ensure that the objects are not distorted too much, the transformations are limited to a certain range [9]. The transformations are applied to the objects, as shown in Listing 4.1. The *fg_image* is the foreground image with the object, and the *transformations* are the transformations from the Albumentations library.

Listing 4.1: Augmentation with Albumentations

```
def transform_foreground(fg_image, transformations):
    return Image.fromarray(transformations(image=np.array(fg_image))['image'])
```

4.2.1 OCCLUSION

To prevent object occlusions or overlaps, the bounding boxes of the objects are checked for intersections. If two objects overlap, the objects are moved to prevent occlusions. This is done by shifting the objects in the image, as in [12] and [23]. Same as the IoU metrics in subsection 2.2.1, the intersection of two bounding boxes is calculated. If the intersection is greater than a certain threshold, the objects are moved to prevent occlusions. For this approach, the threshold is set to 0.25, as in [12]. To calculate the intersection of two bounding boxes, an algorithm based on Mileff [48] and Rezatofighi et al. [61] can be seen in appendix A.

4.3 BLENDING

After the objects have been cut-out and augmented, they are pasted into the background images. This can be done with or without complex blending. The blending is used to seamlessly integrate the foreground elements into the background environment, to prevent the emergence of pixel artifacts, as described in section 3.3.

4.3.1 ALPHA BLENDING

Alpha blending is used to blend the foreground image with the background image, utilising the corresponding mask image. The mathematical process is described in subsection 2.4.1. The mask image is a binary image that indicates the opacity of the foreground image. The blending process is controlled by the opacity values in the mask image, which determine how much of the foreground image is blended with the background image.

For the Alpha blending, the mask images, extracted in section 4.1, are used to blend the objects into the background images. A simple algorithm for Alpha blending is shown in Algorithm 3, which takes the foreground image, background image, and alpha mask as input and returns the final blended image, using the formula from Equation 2.11. As seen, for each pixel in the image (x, y), the blended pixel is computed by combining the foreground and background pixels using the alpha value from the mask image.

Algorithm 3 Alpha Blending for Image Combination

```
Require: Foreground image $I_{\text{foreground}}$, Background image $I_{\text{background}}$, Alpha mask $\alpha$
Ensure: Final blended image $I_{\text{blended}}$
1: for each pixel $(x, y)$ in the images do
2:     Let $A \leftarrow \alpha(x, y)$
3:     Let $F \leftarrow I_{\text{foreground}}(x, y)$
4:     Let $B \leftarrow I_{\text{background}}(x, y)$
5:     Compute blended pixel:
           $I_{\text{blended}}(x, y) \leftarrow F \cdot A + B \cdot (1 - A)$
6: end for
7: return $I_{\text{blended}}$
```

4.3.2 GAUSSIAN BLENDING

Gaussian blending is similar to Alpha blending, but it uses a blurred mask image instead of a binary mask to blend the foreground and background. The mask is blurred using a Gaussian filter, which softens edges and creates a smooth transition, ensuring seamless object integration and reducing edge visibility.

The blending process is guided by the blurred mask, which determines the contribution of the foreground to the final image. The Algorithm 4 takes the foreground, background, mask image, kernel size, and sigma as inputs, producing the blended image. Gaussian blurring is performed using `cv2.GaussianBlur()` with the specified parameters.

Algorithm 4 Gaussian Blending for Image Combination

```
Require: Foreground image $I_{\text{foreground}}$, Background image $I_{\text{background}}$, Alpha mask $\alpha$,
    Kernel size $k$, Sigma $\sigma$
Ensure: Final blended image $I_{\text{blended}}$
1: Apply Gaussian blur to the alpha mask:
       $\text{blurred\_mask} \leftarrow \text{cv2.GaussianBlur}(\alpha, k, \sigma)$
2: Blend the foreground with the background using the blurred mask:
       $I_{\text{blended}} \leftarrow \text{alphaBlending}(I_{\text{foreground}}, I_{\text{background}}, \text{blurred\_mask})$
3: return $I_{\text{blended}}$
```

4.3.3 POISSON BLENDING

For Poisson blending, the `cv2.seamlessClone()` function from the OpenCV library [6] is used, as it provides an efficient implementation of Poisson blending [18]. The blending mode is set to `cv2.MIXED_CLONE`, as it provides a good balance between the source and destination images, as in [18, 26]. The position of the foreground image is determined by the augmentation process, as described in section 4.2.

Algorithm 5 Poisson Blending for Image Combination with Seamless Cloning

```
Require: Foreground image $I_{\text{foreground}}$, Background image $I_{\text{background}}$, Alpha Mask $\alpha$,
    Position $(x, y)$
Ensure: Final blended image $I_{\text{blended}}$
1: Blend the foreground with the background using seamless cloning:
       $I_{\text{blended}} \leftarrow \text{cv2.seamlessClone}(I_{\text{foreground}}, I_{\text{background}}, \alpha, (x, y))$
2: return $I_{\text{blended}}$
```

4.3.4 PYRAMID BLENDING

As in subsection 2.4.4, the code shows how to blend two images using the Pyramid blending technique. The images are first decomposed into Gaussian pyramids, and then the Laplacian pyramids are derived by computing the difference between successive levels of the Gaussian pyramid. The blending process involves combining the Laplacian components of the two images using the Gaussian mask levels as weights. Finally, the blended image is reconstructed by collapsing the pyramid levels. The code for Pyramid blending is shown in the appendix A.

Algorithm 6 Pyramid Blending with Gaussian and Laplacian Pyramids

```
Require: Foreground image $F$, Background image $B$, Alpha mask $\alpha$, pyramid levels $L$
Ensure: Blended image $I_{\text{blended}}$
 1: Construct Gaussian Pyramids:
 2: Create Gaussian pyramids $G_F$, $G_B$, and $G_\alpha$ for images $F$, $B$, and $\alpha$ with $L$ levels.
 3: Construct Laplacian Pyramids:
 4: Derive Laplacian pyramids $L_F$ and $L_B$ from $G_F$ and $G_B$ by subtracting successive
    levels after upsampling.
 5: Blend Pyramids:
 6: for each level $i$ from 1 to $L$ do
 7:     Combine corresponding levels of $L_F$ and $L_B$ using $G_\alpha[i]$:
            $L_{\text{blend}}[i] \leftarrow G_\alpha[i] \cdot L_F[i] + (1 - G_\alpha[i]) \cdot L_B[i]$
 8: end for
 9: Reconstruct Blended Image:
10: Reconstruct $I_{\text{blended}}$ by summing up levels of $L_{\text{blend}}$, starting from the smallest and
    progressively adding upsampled levels.
11: return $I_{\text{blended}}$
```

4.3.5 MULTIPLE BLENDING METHODS

When employing multiple blending methods, as illustrated in [12], the initial steps resemble those of variants utilising single blending methods. Initially, a random foreground object is selected and augmented to diversify its position and orientation. Subsequently, a random background is selected, which will be employed for all methods. Thereafter, for each employed blending method, a final image is generated using its single blending technique. To illustrate this process, consider a scenario where four blending methods are employed. In this instance, four images are generated, each featuring the same foreground objects positioned identically on a shared background. However, the objects are blended differently, resulting in a diverse array of images. Upon revisiting the Algorithm 1, line 11 can be adapted to include multiple blending methods. This can be succinctly described as an extension of the existing algorithm, replacing line 11 with the following:

Algorithm 7 Multiple Blending Methods

```
Require: Set of blending methods $\mathcal{BM} = \{bm_1, bm_2, \ldots, bm_n\}$, Selected and al-
    ready augmented foreground images $F_i$, Selected background image $B$
1: Initialize set of images $\{I_{bm_1}, I_{bm_2}, \ldots, I_{bm_n}\}$
2: for each blending method $bm \in \mathcal{BM}$ do
3:     for each foreground image $F_i$ do
4:         Blend $F_i$ with $B$ using blending method $bm$
5:     end for
6: end for
7: return Set of images $\{I_{b_1}, I_{b_2}, \ldots, I_{b_n}\}$
```

where the set of blending methods $\mathcal{BM}$ contains the employed blending methods, the selected and augmented foreground images F_i are the objects, and the selected background image B is the background image. These parameters are already defined in the Algorithm 1. The algorithm initiates the generation of a set of images, which will ultimately comprise the final images for each blending method. For each blending method, the foreground images are blended with the background image using the blending method. The final images are then returned as a set.

4.4 DATASET MIX

In order to evaluate the impact of the various blending methods on the object detection model, a combination of synthetic and real data is employed, with the ratios of these two types of data varying. In order to mix the real and synthetic data, the datasets are first undersampled to the size of the smallest dataset [72]. Then the datasets are mixed with the desired ratio.

For example, considering the following scenario: Assuming a dataset of 3000 real and 10000 synthetic images, the objective is to blend them with specific ratios, namely 10:90, 30:70, 50:50, 70:30 and 90:10. It is important to note that undersampling imposes a limitation on the maximal size of the new dataset, which is constrained to the size of the smallest existing dataset. Given the availability of 3000 images in the real dataset, the new dataset will contain 3000 images in total. For the 30:70 ratio, the new dataset will contain 900 real and 2100 synthetic images. For the 50:50 ratio, the new dataset will contain 1500 real and 1500 synthetic images. This pattern continues for all ratios.

4.5 FID AND CMMD

The FID values are calculated with code from Seitzer [64], while the CMMD values are calculated with code from [35]. Both FID and CMMD are calculated against the *in-domain* dataset of the test data, which consists of real images from the domain of interest. For each synthetic dataset of the tests, the FID and CMMD values are calculated, and then compared to the baseline to evaluate the realism of the synthetic images.

5 EVALUATION METHOD

The following chapter will outline the methodology for evaluating the proposed synthetic data generation approach, using the self-trained YOLOv11 model. The evaluation of the FID and CMMD metrics has previously been explained in the preceding section 4.5.

The evaluation is designed to assess the effectiveness of the synthetic data generation method in bridging the synthetic-to-reality gap between the training and test data. The evaluation consists of multiple tests, comparing different training configurations and data compositions. The tests utilise both real-world and synthetic datasets to determine how well the model generalises to various conditions. The chapter also provides comprehensive details on the datasets utilised, the experimental setup, the selected hyperparameters, and the evaluation criteria.

5.1 DATASETS

The tests are based on two datasets containing real-world images. Both datasets contain images of tardigrades taken under a microscope, as described in section 1.1, which are annotated with bounding boxes to define the position of the tardigrades in the images.

In-Domain Dataset The in-domain dataset is the main dataset for the evaluation of the tests in this approach. It reflects the real-world scenario for the aforementioned use case: the detection of tardigrades in context of a sewage plant. An example of these images is shown in Figure 1.1a. These images are taken by the "Zentralklärwerk Darmstadt" and manually annotated with bounding boxes. All images are taken with the same type of microscope, which has the same magnification and resolution. This dataset contains 223+57 images with tardigrades and 356 images without tardigrades. The 356 images without tardigrades are used as background images for the synthetic data generation. These images contain mud from the sewage plant and also other microorganisms, which are not the focus of this work.

The 223 images with tardigrades are used as the test set for all tests. The remaining 57 images are on the one hand used in the baseline test, mixed with the cross-domain dataset. On the other hand they are used for the training of the model in the second and third test, where the objects are cutted out and pasted onto the background images.

Cross-Domain Dataset The second dataset is a cross-domain dataset from Jaso [34] published on Roboflow [13], which is publicly available. An example of these images is shown in Figure 1.1b. The dataset contains 3356 images. The images are taken with different types of microscopes, having different magnifications and resolutions. The images are taken in primarily clean water. Barely any distractors are present in the images. The dataset is already annotated with bounding boxes.

This dataset is on the one hand used in the baseline test, mixed with the 57 images from the in-domain dataset. On the other hand, it is used for the training of the model in the second and third test. These images are used as they are and the objects are not used for the synthetic data generation.

5.2 TESTS

To evaluate the effectiveness of the synthetic data generation methods, three tests have been designed. The first test (T1) serves as a baseline and evaluates the model's performance on real data from the in-domain dataset. The second test (T2) and the third test (T3) involve the generation of synthetic data using the "Cut-Paste" method, in which objects from the in-domain dataset are cut and pasted into images from the same dataset. T2 uses one blending method, while T3 uses multiple blending methods.

5.2.1 T1 - BASELINE

The first test $T1_{\text{baseline}}$ is designed to evaluate the model's performance when trained on both real datasets. The model is trained on images from the in-domain dataset and the cross-domain dataset. Then its tested on images from the in-domain dataset. This test should show how well the model can generalize to different domains, when trained on both domains, but with significantly more images from the cross-domain dataset.

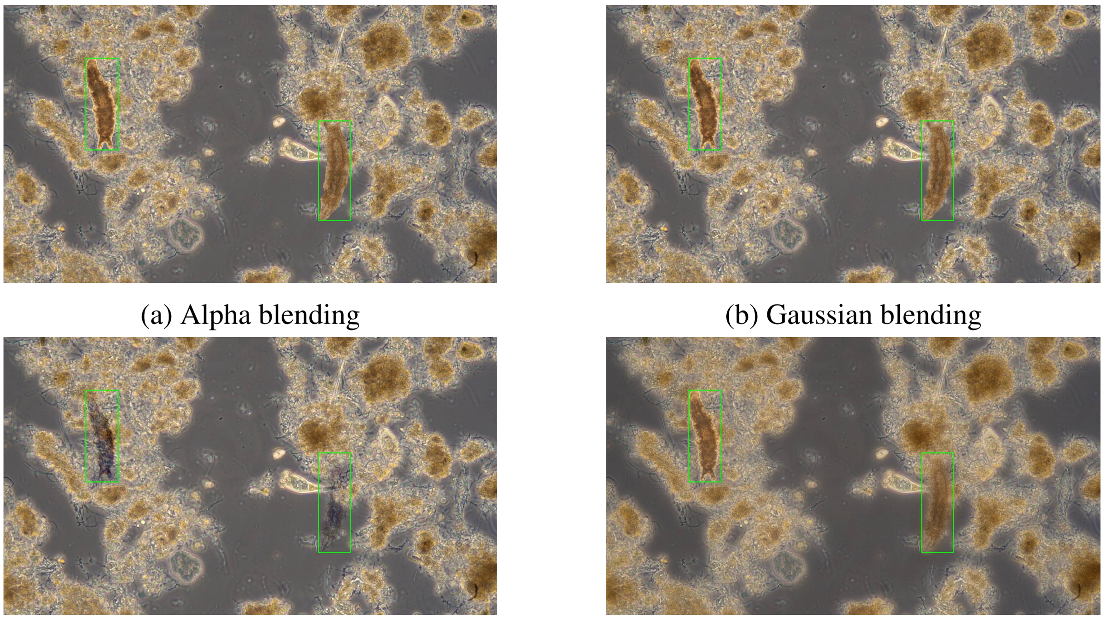
5.2.2 T2 - ONE BLENDING METHOD

The second test (T2) involves the generation of new synthetic images, in which microorganisms from the in-domain dataset are cut and pasted onto images also from the in-domain dataset. As real data, images from the cross-domain dataset are used. So a mix of real clean data and synthetic muddy data is used for the training of the model. For these test, the model is trained with different splits of synthetic and real data. Based on [12], the model is trained with different splits of 30%, 60%, 90% and 100% synthetic data. The remaining data is real data from the cross-domain dataset.

The test is divided into four different tests, each for another blending technique. As from section 4.3, Alpha blending, Gaussian blending, Poisson blending and Pyramid blending are used. The objects from the 57 images of the in-domain dataset are cutted out and pasted onto the background images of the in-domain dataset. An illustration for the different generated synthetic images of the second test can be seen in Figure 5.1.

5.2.3 T3 - MULTIPLE BLENDING METHODS

The third test (T3) also involves the generation of new synthetic images. Other as in the second test, all blending methods are used to blend the objects into the background images, as proposed by [12, 23, 51]. The blending methods are Alpha blending, Gaussian blending and Pyramid blending, as explained in section 4.3. Poisson blending is not used, because in $T2.3_{\text{poisson}}$, the blending method did not perform well. A more detailed explaination can be found in Table 6.1, where the results of the tests are presented and discussed. For each randomly generated set of foregounds and the corresponding



(a) Alpha blending

(b) Gaussian blending

(c) Poisson blending

(d) Pyramid blending

Figure 5.1: Examples for the different blending methods. Used for the training of the model. Including the ground truth.

background image, each blending method generates one image, so $n * 3$ images are generated. As in the second test, the objects are cut and pasted from the in-domain dataset to the background images from the in-domain dataset. However, the real data for this test is taken from the cross-domain dataset, like in the second test. So a mix of real clean data and synthetic muddy data is used for the training of the model. As with T2, illustrations of the various blending methodologies can be observed in Figure 5.1. However, for the third evaluation, all blending methodologies except for the Poisson blending method are utilised. As in T2, the model is trained with different splits of synthetic and real data: 30%, 60%, 90% and 100% of synthetic data. The remaining data is the real data from the cross-domain dataset.

5.3 PARAMETERS

For the tests, different Parameters have been defined. These include the split of the train and validation data, the hyperparameters for the model, as well as the evaluation criteria and the parameters for the synthetic data generation.

5.3.1 TRAIN/VALIDATION SPLIT

The training and validation datasets are divided using an 80:20 split. For dataset T1, consisting of $3356 + 57 = 3413$ images, at least 511 images are allocated for validation. Similarly, for dataset T2 and T3, comprising 3356 images, the validation subset contains at least 510 images, ensuring a confidence interval of 95% with a margin of error of 0.04. A detailed overview of the train, validation, and test sizes for each test is provided in Table 5.1. Due to the undersampling, explained in section 4.4, the train and validation sizes of the $T1_{\text{baseline}}$ test are different from the other tests, as the mixing of the datasets is done before the split. So the mixing process of T2 and T3 oriantes on the 3356 images of the cross-domain dataset and is then split into the train and validation set. In contrast, the mixing process of the $T1_{\text{baseline}}$ test is done with the 57 images of the in-domain dataset and the 3356 images of the cross-domain dataset. The split is then done with the mixed dataset.

Test	**Train size**	**Valid Size**	**Test size**
$T1_{\text{baseline}}$	2684+45=2729	672+12=684	223
$T2.1_{\text{alpha}}$	2684	672	223
$T2.2_{\text{gauss}}$	2684	672	223
$T2.3_{\text{poisson}}$	2684	672	223
$T2.4_{\text{pyramid}}$	2684	672	223
$T3_{\text{multip}}$	2684	672	223

Table 5.1: Train, valid and test sizes for the tests T1, T2 and T3

5.3.2 MODEL

To evaluate the experiments, a YOLOv11m model is utilized. Here, m indicates the medium size of the model, offering an optimal balance between computational complexity, training time, and detection performance [23, 40]. The model is initialized with pretrained MSCOCO weights [12, 23, 51], followed by fine-tuning with the experimental datasets.

5.3.3 HYPERPARAMETERS

The YOLOv11m model is trained over 15 epochs, as preliminary tests indicate convergence within this range. A learning rate of 0.002 is automatically set by the YOLOv11 optimizer [68], with the Adam optimizer employed for training. Batch size is set to 32. Default values from the YOLOv11 framework are used except for the parameters specified above.

5.3.4 EVALUATION CRITERIA

In order to ensure reproducibility and mitigate variance, each experiment is repeated 20 times. The evaluation metrics employed include the mAP and the IoU, as defined in section 2.2. These are combined to provide a comprehensive evaluation of the model's performance across different IoU thresholds. The following metrics are utilised to evaluate the model's performance, as they are commonly employed in object detection tasks [12, 23, 51]:

- mAP@0.50: The mean average precision at an IoU threshold of 50%.
- mAP@[0.50:0.95]: The mean average precision over a range of IoU thresholds from 50% to 95% in increments of 5%.

The mAP@0.50 metric is used to evaluate the model's performance on the primary task of detecting tardigrades, while mAP@[0.50:0.95] is used to assess the model's accuracy in localizing the objects.

5.3.5 SYNTHETIC DATA GENERATION

Synthetic datasets are generated by pasting 1–4 objects into background images. Object sizes are adjusted with a scaling factor ranging from 0.25 to 0.85, ensuring diversity. Gaussian blending with a kernel size of nine and Pyramid blending with six levels are applied to create seamless integrations.

6 RESULTS

This chapter presents the results of the experiment, which aimed to evaluate the performance of a self-trained object detection model on synthetic data. The results are analysed and discussed in the context of the research questions. The quantification of the synthetic-to-reality gap using the self-trained model is compared with the FID and CMMD metrics. The chapter is structured as follows: First, the results of the self-trained model are presented, followed by a comparison with the FID and CMMD values. The chapter concludes with a discussion of the findings in the context of the research questions.

The self-trained model was trained with the different training data, as described in chapter 5. All tests were evaluated on the same test data, which are from the in-domain dataset. Each model was trained 20 times and the mean of the mAP@0.50 and mAP@[0.50:0.95] was calculated. The baseline test $T1_{baseline}$ was trained with a mix of *cross-domain* and *in-domain* data. The tests $T2.1_{alpha}$, $T2.2_{gauss}$, $T2.3_{poisson}$ and $T2.4_{pyramid}$ were trained with different ratios of synthetic to real data, each with a different blending method. The test $T3_{multip}$ was trained with a combination of all blending methods, excluding the Poisson blending method, and again with different ratios of synthetic to real data. A table with the results of the tests is shown in Table 6.1. The mAP@0.50 represents the mean of the mAP with an IoU threshold of 0.5. The mAP@[0.50:0.95] is the mean of the mAP with an IoU threshold range from 0.5 to 0.95. The mAP values ranges from 0 to 1, where higher values indicate better results.

The baseline test $T1_{baseline}$ achieved a mAP@0.50 of 0.81, while the training with only one blending method in $T2.1_{alpha}$, $T2.2_{gauss}$, $T2.3_{poisson}$ and $T2.4_{pyramid}$ achieved a mAP@0.50 up to 0.97, which was achieved with the Pyramid blending method. The superior performance of the Pyramid blending method is consistent with the findings of Garcia-Peraza-Herrera et al. [19]. All single blending methods gained a mAP@0.50 between 0.92 and 0.97, except the Poisson blending method, which only achieved a mAP@0.50 up to 0.43. As stated by Henz et al. [28], the Poisson blending method also changes the color of images, which is not conducive to photorealistic images. This may be the reason why this method performed worst in all tests. Using multiple blending methods in the $T3_{multip}$ achieved a mAP@0.50 between 0.97 and 0.99, which is the highest mAP@0.50 of all tests. This could be attributed to the model's capacity to generalise the objects rather than simply mastering the blending technique, as asserted by Dwibedi et al. [12].

A comparison between the tests is shown in Figure 6.1, which shows the mAP@0.50 as a heatmap. As the basline test $T1_{baseline}$ has no synthetic data, it is a single box in the plot. The heatmap outlines, that the test $T3_{multip}$ achieved the best results, while the test $T2.3_{poisson}$, which used the Poisson blending method, achieved the worst results.

The results of the mAP@[0.50:0.95] are in line with the mAP@0.50. The test $T1_{baseline}$ achieved a mAP@[0.50:0.95] of 0.58, while the test $T3_{multip}$ achieved a mAP@[0.50:0.95] of 0.85. The test $T2.1_{alpha}$ to $T2.4_{pyramid}$ achieved a

Test	**Train Dataset**	**mAP@0.50**	**mAP@[0.50:0.95]**
T1$_{\text{baseline}}$	cross- & in-domain	0.81	0.58
T2.1$_{\text{alpha}}$	30% Syn + Real	0.93	0.75
	60% Syn + Real	**0.95**	**0.80**
	90% Syn + Real	0.94	0.78
	100% Syn	0.94	0.79
T2.2$_{\text{gauss}}$	30% Syn + Real	0.92	0.74
	60% Syn + Real	**0.95**	**0.79**
	90% Syn + Real	0.92	0.77
	100% Syn	0.93	0.77
T2.3$_{\text{poisson}}$	30% Syn + Real	0.30	0.18
	60% Syn + Real	**0.43**	**0.27**
	90% Syn + Real	0.26	0.16
	100% Syn	0.29	0.17
T2.4$_{\text{pyramid}}$	30% Syn + Real	0.95	0.78
	60% Syn + Real	**0.97**	**0.82**
	90% Syn + Real	0.93	0.76
	100% Syn	0.94	0.78
T3$_{\text{multip}}$	30% Syn + Real	0.98	0.82
	60% Syn + Real	0.98	0.85
	90% Syn + Real	**0.99**	**0.85**
	100% Syn	0.97	0.94

Table 6.1: Test results for T1, T2, and T3, including mAP@0.50 and mAP@[0.50:0.95]. The values are averaged over 20 runs. The values ranges from 0 to 1, where higher values indicate better results. The best results per test are highlighted in bold.

mAP@[0.50:0.95] between 0.74 and 0.82, except the Poisson blending method, which only achieved a mAP@[0.50:0.95] of 0.27.

The first research question "How can synthetic generated images impact the model's ability to generalize between different environments?" can be answered with the results of the tests:

Research Question 1

The test T3$_{\text{multip}}$ achieved the best results, increasing the mAP@0.50 by approximately 0.18 and the mAP@[0.50:0.95] by about 0.27, compared to the baseline test. This finding suggests that the model demonstrates superior generalisation capabilities when trained on a dataset comprising synthetic images from the domain of interest.

Furthermore, a comparison is made between the results of the object detection performance and the FID and CMMD values. The FID and CMMD values are utilised to quantify the synthetic-to-reality gap between synthetic and real images, as an additional evaluation method. The results of the FID and CMMD are presented in Table 6.2

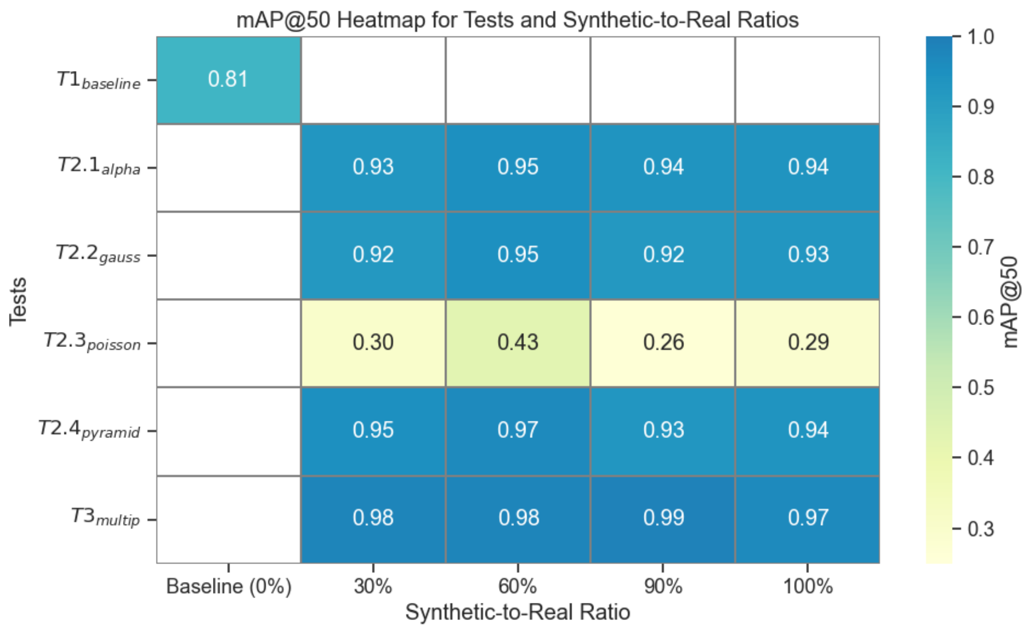


Figure 6.1: Heatmap for T1, T2 and T3. The x-axis shows the ratio of synthetic to real data. The y-axis shows the blending method. The color shows the mAP@0.50. The values ranges from 0 to 1, where higher values indicate better results. The best results per Test are highlighted in bold

and Figure 6.2, calculated against the *in-domain* dataset of the test data.

Test	**FID**	**CMMD**
$T1_{baseline}$	408.434	2.635
$T2.1_{alpha}$	210.000	0.433
$T2.2_{gauss}$	198.130	0.384
$T2.3_{poisson}$	221.460	0.506
$T2.4_{pyramid}$	178.489	0.389
$T3_{multip}$	185.293	0.330

Table 6.2: FID and CMMD for the tests T1, T2, and T3. Lower values indicate a closer distribution between synthetic and real images, indicating similar images.

For the test $T1_{baseline}$, the FID is 408.434 and the CMMD is 2.635. The FID of the test $T3_{multip}$ is 185.293 and the CMMD is 0.330. The FID of the tests between $T2.1_{alpha}$ to $T2.4_{pyramid}$ is situated between 178.489 and 221.460. The CMMD for these tests is situated between 0.389 and 0.506. The FID and CMMD of the test $T1_{baseline}$ is the highest, while the FID and CMMD of the test $T3_{multip}$ is the lowest.

Comparing the results of the FID and CMMD with the results of the self-trained model, the test $T3_{multip}$ achieved the best results, while the test $T1_{baseline}$ achieved the worst results. The tests $T2.1_{alpha}$ to $T2.4_{pyramid}$ are situated between the test $T1_{baseline}$ and the test $T3_{multip}$ in all three, the self-trained model, the FID and CMMD values. The results of the FID and CMMD values are in line with the results of the mAP@0.50

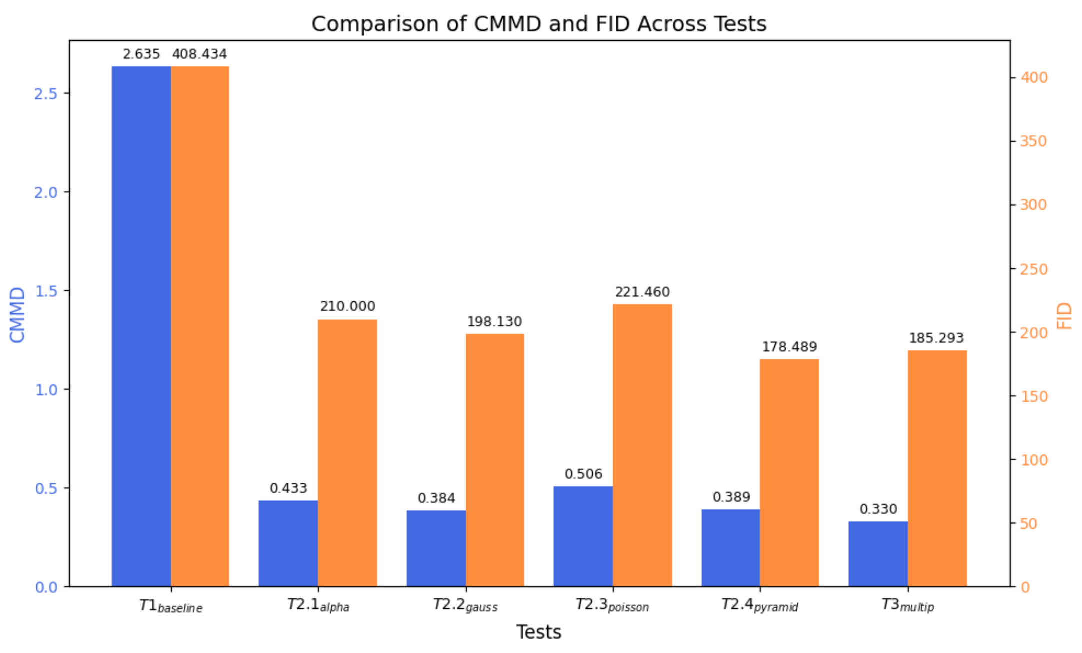


Figure 6.2: FID and CMMD for the tests T1, T2, and T3. Lower values indicate a closer distribution between synthetic and real images, indicating similar images.

and mAP@[0.50:0.95] of the self-trained model. This indicates, that the synthetic-to-reality gap measured by the FID and CMMD is similar to the results of the mAP scores.

Combining the results of the self-trained model with the FID and CMMD values, the second research question "How does the selection of the blending method influence the synthetic-to-reality gap?" can be answered:

Research Question 2

The test results of the self-trained model mostly align with the FID and CMMD values. The test T3$_{\text{multip}}$ achieved the best results and also the lowest FID and CMMD values. The baseline performance of the test T1$_{\text{baseline}}$ was not as good as the other tests and also had the highest FID and CMMD values. As all these values are used to measure the synthetic-to-reality gap and they are aligned with each other, the results indicate that the synthetic-to-reality gap is further influenced by the blending method.

Only the discrepancy of the test T2.3$_{\text{poisson}}$ is not as clear, as the FID and CMMD values are higher than the other tests, but not as high as the test T1$_{\text{baseline}}$. But in the experiment, the test T2.3$_{\text{poisson}}$ performed the worst, which should be reflected in the FID and CMMD values. It can be hypothesised that the imperfect nature of the FID and CMMD values in quantifying the synthetic-to-reality gap in this context is due to the fact that the difference between the images produced by the Poisson blending method is not only in their statistical properties, but also in their color. This could be considered a limitation of the FID and CMMD values in this context.

7 CONCLUSION

Limited real-world data is a common challenge in machine learning, particularly across different domains. This thesis explored the potential of synthetic data generation to mitigate this challenge by augmenting datasets with generated samples. The objective was to evaluate the "Cut-Paste" method in the context of microorganism detection.

The first research question investigated how well a model trained on synthetic data generalizes to real data. To assess this, synthetic images of microorganisms were generated using the "Cut-Paste" method and used to train YOLOv11, a state-of-the-art object detection model. The evaluation compared a baseline model trained only on real images to models trained with different ratios of real and synthetic data. As detailed in chapter 6, the mixed dataset approach achieved a mAP@0.50 of 0.99, significantly outperforming the baseline model (mAP@0.50 = 0.81). This suggests that synthetic data improves model generalization when it complements real data, with data from the domain of interest.

The second research question examined the impact of different blending methods on the synthetic-to-reality gap. Five blending methods were tested: Alpha blending, Gaussian blending, Poisson blending, Pyramid blending, and a multi-blend combining the best-performing methods. Since Poisson blending tended to distort image colors and reduce photorealism, it was excluded from the multi-blend. To quantify the synthetic-to-reality gap, three metrics were analyzed: The difference in the mAP scores when evaluating the self-trained YOLOv11 model on real images, the Fréchet Inception Distance (FID) and the CLIP Maximum Mean Discrepancy (CMMD).

The results in chapter 6 indicated that multi-blend images performed best, followed closely by Pyramid blending. Gaussian blending and Alpha blending yielded similar results, whereas Poisson blending performed the worst due to its alteration of image colors. The FID and CMMD values confirmed these observations, as multi-blend images had the lowest values, indicating a smaller synthetic-to-reality gap.

In summary, this thesis demonstrates that synthetic data, when appropriately blended, enhances model performance and reduces the synthetic-to-reality gap. The findings emphasize the importance of selecting suitable blending methods, with multi-blend proving most effective. These insights contribute to improving synthetic data strategies for microorganism detection and may be extended to other domains facing data scarcity.

7.1 LIMITATIONS

One limitation of this experiment could be the small dataset used to train the model. A larger dataset would have allowed a more comprehensive evaluation of the model's performance. Additionally, a more detailed analysis of the blending methods could have provided further insights into the impact of blending on the model's performance.

This could have included a more in-depth evaluation of the different blending methods and their impact on the model's performance.

It is also important to consider that the test $\mathrm{T3}_{\mathrm{multip}}$ achieved up to 0.99 mAP@0.50, which is a very high value. This could indicate an overfitting of the model on the synthetic data.

7.2 FUTURE WORK

Future works could include the integration of distractor objects or noise into the images to test the model's robustness. Especially, additional context objects like mud could be integrated into the images, in the same way as the microorganisms. The mud could also be augmented or even placed in images containing no mud. Furthermore, additional blending methods could be tested to evaluate their impact on the model's performance. In particular, the Poisson blending technique could be modified to improve its performance and make it more suitable for generating synthetic images of microorganisms. This could provide further insights into the effectiveness of different blending methods and help to identify the most suitable blending method for synthetic data generation in the context of microorganisms. Moreover, other synthetic methods like 3D models or GANs could be tested to evaluate their performance in the context of microorganisms.

Part II

Appendix

A CODE

Pyramid Blending This code snippet shows the implementation of the pyramid blending method used in the thesis.

Listing A.1: Pyramid Blending - Code

```
def _pyramid_blend(self, source, target, mask, num_levels=3):
    """
    Args:
        source: The source image as a numpy array (RGB)
        target: The target image as a numpy array (RGB)
        mask: The mask as a numpy array (grayscale) as float32 in the range [0, 1]
        num_levels: The number of levels in the Gaussian pyramid
    """
    # Initialize Gaussian pyramids for the two images and the mask
    GA = source.copy()
    GB = target.copy()
    GM = mask.copy()

    # Generate the Gaussian pyramids
    gpA = [GA]
    gpB = [GB]
    gpM = [GM]

    for i in range(num_levels):
        # Downsample
        GA = cv2.pyrDown(GA)
        GB = cv2.pyrDown(GB)
        GM = cv2.pyrDown(GM)

        gpA.append(np.float32(GA))
        gpB.append(np.float32(GB))
        gpM.append(np.float32(GM))

    # Initialize Laplacian pyramids
    # Start with the smallest Gaussian level
    lpA = [gpA[num_levels - 1]]
    lpB = [gpB[num_levels - 1]]
    gpMr = [gpM[num_levels - 1]]

    # Build Laplacian pyramids by subtracting successive Gaussian levels
    for i in range(num_levels - 1, 0, -1):
        # Get the size of the next level
        size = (gpA[i - 1].shape[1], gpA[i - 1].shape[0])

        # Compute the Laplacian by subtracting the upsampled Gaussian level from the
            current level
        LA = np.subtract(gpA[i − 1], cv2.pyrUp(gpA[i], dstsize=size))
        LB = np.subtract(gpB[i − 1], cv2.pyrUp(gpB[i], dstsize=size))

        # Append Laplacians to their respective pyramids
        lpA.append(LA)
        lpB.append(LB)

        # Append the corresponding Gaussian mask level
        gpMr.append(gpM[i − 1])

    # Blend the Laplacian pyramids using the Gaussian mask
    LS = []
    for la, lb, gm in zip(lpA, lpB, gpMr):
        # Perform weighted blending for each level
        ls = la ∗ gm + lb ∗ (1.0 − gm)
        LS.append(ls)

    # Reconstruct the final blended image by collapsing the pyramid
    ls_ = LS[0]
    for i in range(1, num_levels):
        # Get the size of the current level
        size = (LS[i].shape[1], LS[i].shape[0])
        ls_ = cv2.add(cv2.pyrUp(ls_, dstsize=size),
                            np.float32(LS[i])) # Add upsampled levels

        # Clip values to the range [0, 255] to avoid overflow when converting to uint8
        ls_ = cv2.normalize(ls_, None, 0, 255, cv2.NORM_MINMAX)

    # Convert the final blended image to uint8 for display or saving
    return Image.fromarray(np.uint8(ls_), ’RGB’)
```

Occlusion This code snippet shows the implementation of the occlusion check used in the thesis.

Listing A.2: Check for object occlusions

```
# bbox: xcenter, ycenter, width, height
def calculate_intersection(bbox1, bbox2):
    """
    Args:
        bbox1: Bounding box 1
        bbox2: Bounding box 2
    """
    x1, y1, w1, h1 = bbox1
    x2, y2, w2, h2 = bbox2
    x1_min, x1_max = x1 − w1 / 2, x1 + w1 / 2
    y1_min, y1_max = y1 − h1 / 2, y1 + h1 / 2
    x2_min, x2_max = x2 − w2 / 2, x2 + w2 / 2
    y2_min, y2_max = y2 − h2 / 2, y2 + h2 / 2
    x_overlap = max(0, min(x1_max, x2_max) − max(x1_min, x2_min))
    y_overlap = max(0, min(y1_max, y2_max) − max(y1_min, y2_min))
    return x_overlap ∗ y_overlap

def calculate_union(bbox1, bbox2):
    """
    Args:
        bbox1: Bounding box 1
        bbox2: Bounding box 2
    """
    x1, y1, w1, h1 = bbox1
    x2, y2, w2, h2 = bbox2
    area1 = w1 ∗ h1
    area2 = w2 ∗ h2
    return area1 + area2 − calculate_intersection(bbox1, bbox2)

def calculate_iou(bbox1, bbox2):
    intersection = calculate_intersection(bbox1, bbox2)
    union = calculate_union(bbox1, bbox2)
    return intersection / union

def check_for_occlusions(bboxes, threshold):
    for i in range(len(bboxes)):
        for j in range(i + 1, len(bboxes)):
            if calculate_iou(bboxes[i], bboxes[j]) > threshold:
                # Move the objects to prevent occlusions
                # ...
```

GENERATIVE AI

ERKLÄRUNG ZUR ANWENDUNG VON KI UND KI-UNTERSTÜTZTEN TECHNOLOGIEN IM SCHREIBPROZESS

Während der Vorbereitung dieser Arbeit hat der Autor DeepL Write & DeepL Translate verwendet, zur Verbesserung der Lesbarkeit. Nach der Nutzung dieses Tools/Dienstes hat der Autor den Inhalt nach Bedarf überprüft und bearbeitet und übernimmt die volle Verantwortung für den Inhalt der Veröffentlichung.

BIBLIOGRAPHY

[1] Edward Adelson et al. "Pyramid Methods in Image Processing". In: *RCA Eng.* 29 (Nov. 1983).

[2] Hassan Abu Alhaija et al. *Augmented Reality Meets Computer Vision : Efficient Data Generation for Urban Driving Scenes*. Aug. 4, 2017. DOI: 10.48550/arXiv.1708.01566. arXiv: 1708.01566. URL: http://arxiv.org/abs/1708.01566 (visited on 11/24/2024).

[3] Felipe Almeida and Geraldo Xexéo. *Word Embeddings: A Survey*. May 2, 2023. DOI: 10.48550/arXiv.1901.09069. arXiv: 1901.09069[cs]. URL: http://arxiv.org/abs/1901.09069 (visited on 02/23/2025).

[4] Claudio Salvatore Arcidiacono. "An empirical study on synthetic image generation techniques for object detectors". In: (Oct. 3, 2018). Accepted: 2018-11-13T10:07:39Z Publisher: Italy. URL: https://www.politesi.polimi.it/handle/10589/142939 (visited on 11/06/2024).

[5] Felix Assion et al. *A-BDD: Leveraging Data Augmentations for Safe Autonomous Driving in Adverse Weather and Lighting*. Nov. 19, 2024. DOI: 10.48550/arXiv.2408.06071. arXiv: 2408.06071[cs]. URL: http://arxiv.org/abs/2408.06071 (visited on 02/22/2025).

[6] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[7] Alexander Buslaev et al. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (Feb. 2020). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, p. 125. ISSN: 2078-2489. DOI: 10.3390/info11020125. URL: https://www.mdpi.com/2078-2489/11/2/125 (visited on 12/16/2024).

[8] Tamara Regina Dieter et al. "Quantifying the Simulation–Reality Gap for Deep Learning-Based Drone Detection". In: *Electronics* 12.10 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12102197. URL: https://www.mdpi.com/2079-9292/12/10/2197.

[9] Jonas Dirr et al. "Cut-paste image generation for instance segmentation for robotic picking of industrial parts". In: *The International Journal of Advanced Manufacturing Technology* 130.1 (Jan. 1, 2024), pp. 191–201. ISSN: 1433-3015. DOI: 10.1007/s00170-023-12622-4. URL: https://doi.org/10.1007/s00170-023-12622-4 (visited on 11/10/2024).

[10] Tausif Diwan, G. Anirudh, and Jitendra V. Tembhurne. "Object detection using YOLO: challenges, architectural successors, datasets and applications". In: *Multimedia Tools and Applications* 82.6 (Mar. 1, 2023), pp. 9243–9275. ISSN: 1573-7721. DOI: 10.1007/s11042-022-13644-y. URL: https://doi.org/10.1007/s11042-022-13644-y (visited on 11/23/2024).

[11] D. C. Dowson and B. V. Landau. “The Fréchet distance between multivariate normal distributions”. In: *Journal of Multivariate Analysis* 12.3 (1982), pp. 450–455. ISSN: 0047-259X. DOI: https://doi.org/10.1016/0047-259X(82)90077-X. URL: https://www.sciencedirect.com/science/article/pii/0047259X8290077X.

[12] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. *Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection*. Aug. 4, 2017. arXiv: 1708.01642. URL: http://arxiv.org/abs/1708.01642 (visited on 11/05/2024).

[13] B. Dwyer, J. Nelson, T. Hansen, et al. *Roboflow (Version 1.0) [Software]*. 2024. URL: https://roboflow.com.

[14] Mark Everingham et al. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (June 1, 2010), pp. 303–338. ISSN: 1573-1405. DOI: 10.1007/s11263-009-0275-4. URL: https://doi.org/10.1007/s11263-009-0275-4 (visited on 12/09/2024).

[15] Leon Eversberg and Jens Lambrecht. “Generating Images with Physics-Based Rendering for an Industrial Object Detection Task: Realism versus Domain Randomization”. In: *Sensors* 21.23 (2021). ISSN: 1424-8220. DOI: 10.3390/s21237901. URL: https://www.mdpi.com/1424-8220/21/23/7901.

[16] Gerd Fischer and Boris Springborn. “Eigenwerte”. In: *Lineare Algebra: Eine Einführung für Studienanfänger*. Ed. by Gerd Fischer and Boris Springborn. Berlin, Heidelberg: Springer, 2020, pp. 241–306. ISBN: 978-3-662-61645-1. DOI: 10.1007/978-3-662-61645-1_6. URL: https://doi.org/10.1007/978-3-662-61645-1_6 (visited on 02/23/2025).

[17] Marco Forte and François Pitié. *F, B, Alpha Matting*. Mar. 17, 2020. DOI: 10.48550/arXiv.2003.07711. arXiv: 2003.07711[cs]. URL: http://arxiv.org/abs/2003.07711 (visited on 01/04/2025).

[18] Gloria Bueno García et al. *Learning image processing with OpenCV: exploit the amazing features of OpenCV to create powerful image processing applications through easy-to-follow examples*. Community Experience Distilled. Birmingham Mumbai: Packt Publishing, 2015. 1 p. ISBN: 978-1-78328-765-9.

[19] Luis C. Garcia-Peraza-Herrera et al. “Image Compositing for Segmentation of Surgical Tools Without Manual Annotations”. In: *IEEE Transactions on Medical Imaging* 40.5 (May 2021). Conference Name: IEEE Transactions on Medical Imaging, pp. 1450–1460. ISSN: 1558-254X. DOI: 10.1109/TMI.2021.3057884. URL: https://ieeexplore.ieee.org/abstract/document/9350303 (visited on 12/12/2024).

[20] M.W Gardner and S.R Dorling. “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences”. In: *Atmospheric Environment* 32.14 (Aug. 1998), pp. 2627–2636. ISSN: 13522310. DOI: 10.1016/S1352-2310(97)00447-0. URL: https://linkinghub.elsevier.com/retrieve/pii/S1352231097004470 (visited on 12/23/2024).

[21] Georgios Georgakis et al. *Synthesizing Training Data for Object Detection in Indoor Scenes*. Sept. 8, 2017. arXiv: 1702.07836. URL: http://arxiv.org/abs/1702.07836 (visited on 11/06/2024).

[22] Golnaz Ghiasi et al. *Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation*. June 23, 2021. arXiv: 2012.07177. URL: http://arxiv.org/abs/2012.07177 (visited on 11/07/2024).

[23] Nikolaos Giakoumoglou, Eleftheria Maria Pechlivani, and Dimitrios Tzovaras. "Generate-Paste-Blend-Detect: Synthetic dataset for object detection in the agriculture domain". In: *Smart Agricultural Technology* 5 (Oct. 2023), p. 100258. ISSN: 27723755. DOI: 10.1016/j.atech.2023.100258. URL: https://linkinghub.elsevier.com/retrieve/pii/S2772375523000886 (visited on 11/06/2024).

[24] Afzal Godil et al. *Performance Metrics for Evaluating Object and Human Detection and Tracking Systems*. NIST IR 7972. National Institute of Standards and Technology, July 2014, NIST IR 7972. DOI: 10.6028/NIST.IR.7972. URL: https://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7972.pdf (visited on 01/01/2025).

[25] Ian J. Goodfellow et al. *Generative Adversarial Networks*. June 10, 2014. DOI: 10.48550/arXiv.1406.2661. arXiv: 1406.2661[stat]. URL: http://arxiv.org/abs/1406.2661 (visited on 12/17/2024).

[26] Han Guo et al. "Face replacement based on 2D dense mapping". In: *Proceedings of the 2nd International Conference on Image and Graphics Processing*. ICIGP 2019: 2019 2nd International Conference on Image and Graphics Processing. Singapore Singapore: ACM, Feb. 23, 2019, pp. 23–28. ISBN: 978-1-4503-6092-0. DOI: 10.1145/3313950.3313964. URL: https://dl.acm.org/doi/10.1145/3313950.3313964 (visited on 11/25/2024).

[27] Junwei Han et al. "Advanced Deep-Learning Techniques for Salient and Category-Specific Object Detection: A Survey". In: *IEEE Signal Processing Magazine* 35.1 (Jan. 2018). Conference Name: IEEE Signal Processing Magazine, pp. 84–100. ISSN: 1558-0792. DOI: 10.1109/MSP.2017.2749125. URL: https://ieeexplore.ieee.org/document/8253582 (visited on 12/09/2024).

[28] Bernardo Henz, Frederico A. Limberger, and Manuel M. Oliveira. "Independent color-channel adjustment for seamless cloning based on Laplacian-membrane modulation". In: *Computers & Graphics* 57 (June 1, 2016), pp. 46–54. ISSN: 0097-8493. DOI: 10.1016/j.cag.2016.03.004. URL: https://www.sciencedirect.com/science/article/pii/S0097849316300176 (visited on 12/22/2024).

[29] Martin Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. Jan. 12, 2018. DOI: 10.48550/arXiv.1706.08500. arXiv: 1706.08500[cs]. URL: http://arxiv.org/abs/1706.08500 (visited on 12/23/2024).

[30] Stefan Hinterstoisser et al. *On Pre-Trained Image Features and Synthetic Images for Deep Learning*. Nov. 16, 2017. DOI: 10.48550/arXiv.1710.10710. arXiv: 1710.10710. URL: http://arxiv.org/abs/1710.10710 (visited on 11/23/2024).

[31] O. Hmidani and E. M. Ismaili Alaoui. "A comprehensive survey of the R-CNN family for object detection". In: *2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet)*. 2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet). ISSN: 2771-7402. Dec. 2022, pp. 1–6. DOI: 10.1109/CommNet56067.2022.9993862. URL: https://ieeexplore.ieee.org/abstract/document/9993862 (visited on 12/18/2024).

[32] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. "Artificial neural networks: a tutorial". In: *Computer* 29.3 (Mar. 1996). Conference Name: Computer, pp. 31–44. ISSN: 1558-0814. DOI: 10.1109/2.485891. URL: https://ieeexplore.ieee.org/document/485891/?arnumber=485891 (visited on 12/23/2024).

[33] Natalia Jakubowska-Krepska et al. "Tardigrades as potential bioindicators in biological wastewater treatment plants". In: *European Journal of Ecology* 4.2 (Dec. 1, 2018). Number: 2, pp. 124–130. ISSN: 1339-8474. DOI: 10.2478/eje-2018-0019. URL: https://journals.ku.edu/EuroJEcol/article/view/11634 (visited on 11/29/2024).

[34] Jasper John Jaso. *Tardigrade Dataset*. Roboflow. 2022. URL: https://universe.roboflow.com/jasper-john-jaso/tardigrade (visited on 12/18/2024).

[35] Sadeep Jayasumana. *google-research/cmmd at master · google-research/google-research*. GitHub. URL: https://github.com/google-research/google-research/tree/master/cmmd (visited on 12/27/2024).

[36] Sadeep Jayasumana et al. *Rethinking FID: Towards a Better Evaluation Metric for Image Generation*. Jan. 25, 2024. DOI: 10.48550/arXiv.2401.09603. arXiv: 2401.09603[cs]. URL: http://arxiv.org/abs/2401.09603 (visited on 12/16/2024).

[37] Matthew Johnson-Roberson et al. *Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?* Feb. 25, 2017. DOI: 10.48550/arXiv.1610.01983. arXiv: 1610.01983. URL: http://arxiv.org/abs/1610.01983 (visited on 11/23/2024).

[38] Jaskirat Kaur and Williamjeet Singh. "A systematic review of object detection from images using deep learning". In: *Multimedia Tools and Applications* 83.4 (Jan. 1, 2024), pp. 12253–12338. ISSN: 1573-7721. DOI: 10.1007/s11042-023-15981-y. URL: https://doi.org/10.1007/s11042-023-15981-y (visited on 11/29/2024).

[39] Osama Khalil et al. "Synthetic training in object detection". In: *2013 IEEE International Conference on Image Processing*. 2013 IEEE International Conference on Image Processing. ISSN: 2381-8549. Sept. 2013, pp. 3113–3117. DOI: 10.1109/ICIP.2013.6738641. URL: https://ieeexplore.ieee.org/document/6738641/?arnumber=6738641 (visited on 11/06/2024).

[40] Rahima Khanam and Muhammad Hussain. *YOLOv11: An Overview of the Key Architectural Enhancements*. Oct. 23, 2024. DOI: 10.48550/arXiv.2410.17725. arXiv: 2410.17725. URL: http://arxiv.org/abs/2410.17725 (visited on 11/23/2024).

[41] Douwe Kiela and Léon Bottou. "Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. EMNLP 2014. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 36–45. DOI: 10.3115/v1/D14-1005. URL: https://aclanthology.org/D14-1005/ (visited on 02/23/2025).

[42] Eren Kilic. *A Review on Image Blending using Image Pyramids*. Jan. 2021. URL: https://www.researchgate.net/publication/352715359_A_Review_on_Image_Blending_using_Image_Pyramids (visited on 02/23/2025).

[43] Shaobo Lin et al. “Explore the Power of Synthetic Data on Few-shot Object Detection”. In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Vancouver, BC, Canada: IEEE, June 2023, pp. 638–647. ISBN: 9798350302493. DOI: 10.1109/CVPRW59228.2023.00071. URL: https://ieeexplore.ieee.org/document/10208358/ (visited on 11/06/2024).

[44] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48.

[45] Tony Lindeberg. “Discrete Approximations of Gaussian Smoothing and Gaussian Derivatives”. In: *Journal of Mathematical Imaging and Vision* 66.5 (Oct. 1, 2024), pp. 759–800. ISSN: 1573-7683. DOI: 10.1007/s10851-024-01196-9. URL: https://doi.org/10.1007/s10851-024-01196-9 (visited on 12/27/2024).

[46] Pingli Ma et al. “A state-of-the-art survey of object detection techniques in microorganism image analysis: from classical methods to deep learning approaches”. In: *Artificial Intelligence Review* 56.2 (Feb. 1, 2023), pp. 1627–1698. ISSN: 1573-7462. DOI: 10.1007/s10462-022-10209-1. URL: https://doi.org/10.1007/s10462-022-10209-1 (visited on 11/26/2024).

[47] Maria L. Marco. “Defining how microorganisms benefit human health”. In: *Microbial Biotechnology* 14.1 (2021). _eprint: https://enviromicro-journals.onlinelibrary.wiley.com/doi/pdf/10.1111/1751-7915.13685, pp. 35–40. DOI: https://doi.org/10.1111/1751-7915.13685. URL: https://enviromicro-journals.onlinelibrary.wiley.com/doi/abs/10.1111/1751-7915.13685.

[48] Peter Mileff. “COLLISION DETECTION IN 2D GAMES”. In: 11 (Sept. 2023), p. 10. DOI: 10.32968/psaie.2023.3.2.

[49] Daniel Mas Montserrat et al. “Training Object Detection And Recognition CNN Models Using Data Augmentation”. In: *Electronic Imaging* 29 (Jan. 29, 2017). Publisher: Society for Imaging Science and Technology, pp. 27–36. ISSN: 2470-1173. DOI: 10.2352/ISSN.2470-1173.2017.10.IMAWM-163. URL: https://library.imaging.org/ei/articles/29/10/art00005 (visited on 11/06/2024).

[50] Unites Nations. “Sustainable development goals”. In: *Sustainable development knowledge platform* (2015).

[51] Alexander Naumann et al. “Scrape, Cut, Paste and Learn: Automated Dataset Generation Applied to Parcel Logistics”. In: *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA). Dec. 2022, pp. 1026–1031. DOI: 10.1109/ICMLA55696.2022.00171. URL: https://ieeexplore.ieee.org/document/10069342 (visited on 11/23/2024).

[52] Rafael Padilla et al. “A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit”. In: *Electronics* 10.3 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10030279. URL: https://www.mdpi.com/2079-9292/10/3/279.

[53] Xingchao Peng and Kate Saenko. *Synthetic to Real Adaptation with Generative Correlation Alignment Networks*. Mar. 18, 2017. DOI: 10.48550/arXiv.1701.05524. arXiv: 1701.05524. URL: http://arxiv.org/abs/1701.05524 (visited on 12/03/2024).

[54] Patrick Pérez, Michel Gangnet, and Andrew Blake. "Poisson image editing". In: *ACM Transactions on Graphics* 22.3 (July 2003), pp. 313–318. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/882262.882269. URL: https://dl.acm.org/doi/10.1145/882262.882269 (visited on 11/16/2024).

[55] Thomas Porter and Tom Duff. "Compositing digital images". In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '84. New York, NY, USA: Association for Computing Machinery, 1984, pp. 253–259. ISBN: 0-89791-138-5. DOI: 10.1145/800031.808606. URL: https://doi.org/10.1145/800031.808606.

[56] Aayush Prakash et al. "Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019 International Conference on Robotics and Automation (ICRA). ISSN: 2577-087X. May 2019, pp. 7249–7255. DOI: 10.1109/ICRA.2019.8794443. URL: https://ieeexplore.ieee.org/document/8794443/?arnumber=8794443 (visited on 11/06/2024).

[57] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. Feb. 26, 2021. DOI: 10.48550/arXiv.2103.00020. arXiv: 2103.00020[cs]. URL: http://arxiv.org/abs/2103.00020 (visited on 02/23/2025).

[58] Luis Rangel DaCosta et al. "A robust synthetic data generation framework for machine learning in high-resolution transmission electron microscopy (HRTEM)". In: *npj Computational Materials* 10.1 (July 29, 2024). Publisher: Nature Publishing Group, pp. 1–11. ISSN: 2057-3960. DOI: 10.1038/s41524-024-01336-0. URL: https://www.nature.com/articles/s41524-024-01336-0 (visited on 11/06/2024).

[59] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. May 9, 2016. DOI: 10.48550/arXiv.1506.02640. arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640 (visited on 11/29/2024).

[60] Tal Remez, Jonathan Huang, and Matthew Brown. *Learning to Segment via Cut-and-Paste*. Mar. 16, 2018. DOI: 10.48550/arXiv.1803.06414. arXiv: 1803.06414. URL: http://arxiv.org/abs/1803.06414 (visited on 12/03/2024).

[61] Hamid Rezatofighi et al. "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Long Beach, CA, USA: IEEE, June 2019, pp. 658–666. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00075. URL: https://ieeexplore.ieee.org/document/8953982/ (visited on 01/16/2025).

[62] Adrian Rosebrock. *Intersection over Union (IoU) for object detection*. PyImageSearch. Nov. 7, 2016. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (visited on 12/09/2024).

[63] Tim Salimans et al. *Improved Techniques for Training GANs*. June 10, 2016. DOI: 10.48550/arXiv.1606.03498. arXiv: 1606.03498[cs]. URL: http://arxiv.org/abs/1606.03498 (visited on 02/22/2025).

[64] Maximilian Seitzer. *pytorch-fid: FID Score for PyTorch*. Aug. 2020. URL: https://github.com/mseitzer/pytorch-fid.

[65] Leon Sixt, Benjamin Wild, and Tim Landgraf. *RenderGAN: Generating Realistic Labeled Data*. Jan. 12, 2017. DOI: 10.48550/arXiv.1611.01331. arXiv: 1611.01331. URL: http://arxiv.org/abs/1611.01331 (visited on 12/03/2024).

[66] Hao Su et al. *Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views*. May 21, 2015. DOI: 10.48550/arXiv.1505.05641. arXiv: 1505.05641[cs]. URL: http://arxiv.org/abs/1505.05641 (visited on 02/02/2025).

[67] A. Tsirikoglou, G. Eilertsen, and J. Unger. "A Survey of Image Synthesis Methods for Visual Machine Learning". In: *Computer Graphics Forum* 39.6 (2020). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14047, pp. 426–451. ISSN: 1467-8659. DOI: 10.1111/cgf.14047. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14047 (visited on 11/23/2024).

[68] Ultralytics. *Ultralytics Documentation*. URL: https://docs.ultralytics.com/reference/engine/trainer (visited on 12/27/2024).

[69] Stanford University. *CS231n Convolutional Neural Networks for Visual Recognition*. URL: https://cs231n.github.io/convolutional-networks/ (visited on 12/30/2024).

[70] Kentaro Wada. *labelme: Image Polygonal Annotation with Python*. Publication Title: GitHub repository. 2018. URL: https://github.com/wkentaro/labelme.

[71] Xinlong Wang et al. *Adversarial Generation of Training Examples: Applications to Moving Vehicle License Plate Recognition*. Nov. 10, 2017. DOI: 10.48550/arXiv.1707.03124. arXiv: 1707.03124. URL: http://arxiv.org/abs/1707.03124 (visited on 12/03/2024).

[72] Vitor Werner de Vargas et al. "Imbalanced data preprocessing techniques for machine learning: a systematic mapping study". In: *Knowledge and Information Systems* 65.1 (Jan. 1, 2023), pp. 31–57. ISSN: 0219-3116. DOI: 10.1007/s10115-022-01772-8. URL: https://doi.org/10.1007/s10115-022-01772-8.

[73] Yunyang Xiong et al. *EfficientSAM: Leveraged Masked Image Pretraining for Efficient Segment Anything*. Dec. 1, 2023. DOI: 10.48550/arXiv.2312.00863. arXiv: 2312.00863[cs]. URL: http://arxiv.org/abs/2312.00863 (visited on 02/20/2025).

[74] Riccardo Zanella et al. "Auto-generated Wires Dataset for Semantic Segmentation with Domain-Independence". In: *2021 International Conference on Computer, Control and Robotics (ICCCR)*. 2021 International Conference on Computer, Control and Robotics (ICCCR). Jan. 2021, pp. 292–298. DOI: 10.1109/ICCCR49711.2021.9349395. URL: https://ieeexplore.ieee.org/document/9349395 (visited on 11/23/2024).