Fully-Automated Packaging Structure Recognition of Standardized Logistics Assets on Images

Zur Erlangung des akademischen Grades einer

DOKTORIN DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)

von der KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)
angenommene

DISSERTATION

von

M.Sc. Laura Dörr

Tag der mündlichen Prüfung: 02.08.2023

Hauptreferent: Prof. Dr.-Ing. Kai Furmans
Korreferentin: Prof. Dr.-Ing. Anne Meyer

Abstract

Within a logistics supply chain, a large variety of transported goods need to be handled, recognized and checked at many different network points. Often, huge manual effort is required for recognizing or verifying packet identity or packaging structure, for instance, to check the delivery for completeness. We investigate the design and implementation of an algorithm for the complete automation of packaging structure recognition. The alghorithm's objective is to accurately localize one or multiple transport units and recognize relevant characteristics, e.g. the total number and the arrangement of its packaging units, based on a single image. We propose a multi-component pipeline to solve this task of packaging structure recognition.

Our first implementation of this pipeline employs multiple deep learning models, more precisely convolutional neural networks for instance segmentation, as well as computer vision methods and heuristic components. We provide a custom data set of real-world logistics images, which we use for the training and evaluation of our method. We show that the solution is capable of correctly recognizing the packaging structure in approximately 85% of our test cases, and even more (93%) when focusing on the most common package types.

For a selected component of our pipeline, we compare the potential of using leaner custom image processing algorithms, instead of the initially implemented deep learning methods. As a result, we conclude deep learning algorithms to be the more suitable method in our case, due to their high generalization abilities and feature complexities.

Additionally, we formulate the problem of object localization based on custom feature points, as, for instance, corner points of logistics transport units. The aim is to identify object locations more accurately, as compared to when using bounding boxes, whilst constricting object shapes by geometric apriori knowledge. We propose a specific deep learning model as a solution to this task for objects representable by four corner points. The model, named TetraPackNet, is evaluated using general metrics, and use-case-specific measurements. We show the solution's applicability to our recognition pipeline and argue its relevance for other applications, like license plate recognition.

Kurzzusammenfassung

Innerhalb einer logistischen Lieferkette müssen vielfältige Transportgüter an zahlreichen Knotenpunkten bearbeitet, wiedererkannt und kontrolliert werden. Dabei ist oft ein großer manueller Aufwand erforderlich, um die Paketidentität oder auch die Packstruktur zu erkennen oder zu verifizieren. Solche Schritte sind notwendig, um beispielsweise eine Lieferung auf ihre Vollständigkeit hin zu überprüfen. Wir untersuchen die Konzeption und Implementierung eines Verfahrens zur vollständigen Automatisierung der Erkennung der Packstruktur logistischer Sendungen. Ziel dieses Verfahrens ist es, basierend auf einem einzigen Farbbild, eine oder mehrere Transporteinheiten akkurat zu lokalisieren und relevante Charakteristika, wie beispielsweise die Gesamtzahl oder die Anordnung der enthaltenen Packstücke, zu erkennen. Wir stellen eine aus mehreren Komponenten bestehende Bildverarbeitungs-Pipeline vor, die diese Aufgabe der Packstrukturerkennung lösen soll.

Unsere erste Implementierung des Verfahrens verwendet mehrere Deep Learning Modelle, genauer gesagt Convolutional Neural Networks zur Instanzsegmentierung, sowie Bildverarbeitungsmethoden und heuristische Komponenten. Wir verwenden einen eigenen Datensatz von Echtbildern aus einer Logistik-Umgebung für Training und Evaluation unseres Verfahrens. Wir zeigen, dass unsere Lösung in der Lage ist, die korrekte Packstruktur in etwa 85% der Testfälle unseres Datensatzes zu erkennen, und sogar eine höhere Genauigkeit erzielt wird, wenn nur die meist vorkommenden Packstücktypen betrachtet werden.

Für eine ausgewählte Bilderkennungs-Komponente unseres Algorithmus vergleichen wir das Potenzial der Verwendung weniger rechenintensiver, eigens designter Bildverarbeitungsmethoden mit den zuvor implementierten Deep Learning Verfahren. Aus dieser Untersuchung schlussfolgern wir die bessere Eignung der lernenden Verfahren, welche wir auf deren sehr gute Fähigkeit zur Generalisierung zurückführen.

Außerdem formulieren wir das Problem der Objekt-Lokalisierung in Bildern anhand selbst gewählter Merkmalspunkte, wie beispielsweise Eckpunkte logistischer Transporteinheiten. Ziel hiervon ist es, Objekte präziser zu lokalisieren, als dies insbesondere im Vergleich zur Verwendung herkömmlicher umgebender Rechtecke möglich ist, während gleichzeitig die Objektform durch bekanntes Vorwissen zur Objektgeometrie forciert wird. Wir stellen ein spezifisches Deep Learning Modell vor, welches die beschriebene Aufgabe löst im Fall von Objekten, welche durch vier Eckpunkte beschrieben werden können. Das dabei entwickelte Modell mit Namen TetraPackNet wird evaluiert mittels allgemeiner und anwendungsbezogener Metriken. Wir belegen die Anwendbarkeit der Lösung im Falle unserer Bilderkennungs-Pipeline und argumentieren die Relevanz für andere Anwendungsfälle, wie beispielweise Kennzeichenerkennung.

Contents

1 Introduction

1.1 Motivation and Problem Description

In logistics supply chains, numerous materials, parts and products are frequently handled by different stakeholders at different network points. Each task of transportation or asset forwarding includes noteworthy amounts of effort and expenses: Before the transport can begin, a corresponding transport order is issued, the goods need to be packaged, assembled to transportation units and marked with transport labels. During the actual transportation process, the palletized transport unit is handled several times in the outgoing goods department, during loading and unloading, and once again in the destination's goods receipt. Once the transport unit reaches its destination, it has to be identified and checked for completeness. Additionally, one or multiple turnovers at different transfer sites, where the unit needs to be handled and identified once more, are possible. During these processes, each agent may handle a large number of palletized units of similar appearance simultaneously. Fig. 1.1 shows an example of transport-ready palletized units consisting of standardized packages.

Our work contributes an essential step towards automation of the recognition and the checking of transported goods. More precisely, we introduce the task of fully-automated packaging structure recognition and propose a solution thereof. The objective of this task is the recognition of standardized transport units and their packaging structure. We introduce an image recognition pipeline to solve this task of recognizing the packaging structure of uniformly packed [1], standardized transport units on 2D images. As the process is aimed to be accessible, agile and performant at the same time, only a single RGB input image, possibly taken by a smartphone-integrated camera, is required. The pipeline is trained and evaluated using our own, manually acquired and labeled, use-case-specific dataset. We analyze different algorithmic choices for parts of the pipeline and discuss the results and their implications for the whole pipeline. Further, we design specialized deep learning models exploiting geometric characteristics given in the case of packaging structure recognition. Prospectively, these models can be applied and adapted to other use cases.

1.2 Research Questions and Objectives

In this section, we present the central research questions guiding our work to clarify the objectives of our work. We formulate three research questions as follows:

(RQ1) How can information about the constitution, assembly and condition of a packed transport unit be inferred from a 2D image?

[1] By uniformly packed, we denote transport units consisting of one specific package unit type only, and whose package unit layers have the same number of packages each. For more details, see Section 2.2.4

Figure 1.1: Example of stacked transport-ready palletized transport units.

(RQ2) Which algorithms and techniques are well-suited for the implementation of a packaging structure recognition algorithm?

(RQ3) How can available apriori knowledge about geometrical restrictions of transported materials be utilized to design specific detection algorithms, yielding improved results in packaging structure recognition?

The first research question (RQ1) concerns the construction of a solution approach to the task of packaging structure recognition. Moreover, in tackling this question, we design an image processing pipeline, which extracts packaging structure information from suitable 2D images. Using our own dataset, we evaluate our solution and demonstrate its applicability in controlled environments.

The second research question (RQ2) focuses on the algorithmic choices for our image processing pipeline. Our first implementation makes use of image recognition methods involving convolutional neural networks (CNNs), due to their use-case-agnostic character. As deep learning models require large amounts of annotated training data, we explore the possibility of using other, non-learning-based algorithms for the implementation of our pipeline. Due to the high complexity of our multi-step processing pipeline and the involved image recognition sub-steps, we focus on a single case study to reach general implications regarding this question.

Our last research question (RQ3) explores the design and implementation of specific CNN-based image recognition algorithms for the use case of packaging structure recognition. Traditional approaches rely on generic object representations, like bounding boxes and arbitrary pixel masks. When focusing on logistics transport assets, valuable apriori knowledge regarding object geometry is given. We aim to incorporate this knowledge into our models to achieve higher accuracy in the recognition and localization of relevant objects, i.e. transport and packaging units.

1.3 Scope and Organization of the Thesis

The rest of this thesis is organized as follows: Chapter 2 gives a more detailed introduction to the use case of packaging structure recognition and its relevance, as well as the prerequisites and limitations, which are necessary to ensure the task's feasibility. Chapter 3 gives an overview of the required, theoretical and technical, basics and backgrounds. Chapters 4, 5, and 6 contain the core contributions of our work: In Chapter 4, we introduce our first solution to the task of packaging structure recognition, as well as a relevant test dataset of real-world images, and evaluate our method on the data at hand. Chapter 5 gives further insights into the choice of algorithms for the recognition pipeline: We first present an alternative implementation for essential parts based on traditional computer vision approaches, rather than deep learning. Consequently, we compare the pipeline's performances for both algorithmic choices and infer the superiority of deep learning approaches for our solution approach to the use-case at hand. In Chapter 6, we propose highly specialized deep learning models, according to the given geometric apriori knowledge, and incorporate these into our recognition pipeline. In the context of packaging structure recognition, we compare the performance of these specific recognition models to generic models. Chapter 7 concludes our work by summarizing the contributions and findings, and giving guidance for further research opportunities.

We previously published parts of the results presented in this work. Some text passages, figures and equations throughout this work are citations or adoptions of our previous publications. We provide a detailed listing of citations of our previous work in Chapter A.1.

2 Packaging Structure Recognition and Cognitive Systems in Logistics

In this chapter, we thoroughly introduce the use case our work focuses on: **packaging structure recognition** (PSR). Further, we aim to provide an overview of recent developments of innovative logistics systems related to packaging structure recognition, and point out similarities and differences of our work, if possible. Part of this chapter was adopted from our previous publication Dörr et al. 2020a.

2.1 Related Work and Classification of our Work

We discuss existing research related to our use case and our solution to packaging structure recognition. We start on a very general level, considering publications concerning digitalization processes in logistics. Subsequently, we narrow our scope to technically related logistics solutions. Lastly, we focus on related work aiming to solve tasks and problems closely related to ours, i.e. concerned with the visual recognition of logistics assets.

2.1.1 Digitalization of Logistics

Digitalization and process automation triggered by recent advances in technology are of increasing relevance, also in the sector of logistics and supply chain management, as recent publications show: Wei et al. (2019) provide comprehensive insights into the ongoing digitalization of the logistics sector. The official positioning paper of the German Logistics Association (BVL) (Heistermann et al. 2017) argues that this process is important and promising, even though it requires courage and some rethinking. Sustainable digitalization of logistics networks often necessitates a move from owning data, solutions and knowledge towards sharing such resources. The chances of the ongoing digitalization process are manifold: Operative procedures can be simplified and accelerated by the use of technical devices, machines or autonomous robotic systems. The greater and quicker availability of data, like tracking and tracing information, allows for highly optimized workflows. At the same time, customer satisfaction can be improved by providing reliable data and predictions. Further, more reliable demand forecasts enable more accurate production and storage management. This non-exhaustive list of examples indicates that digitalization in logistics is very versatile regarding both applications and underlying techniques.

Starting points for digital transformation processes can be found in in-house logistics as well as in cross-company logistics and supply chain management. Likewise, all logistics sub-areas, starting from production, commissioning, shipment, storage and inventory management, to order and sales

management, are affected by the transformation. Over the past years and decades, the commissioning process, for instance, has evolved from a purely manual, paper-based process to a highly automated task performed by robots or unmanned storage and transport systems on auto-pilot. Moreover, manual data transcripts and records have been replaced by application interfaces and connected information systems and platforms.

Herold et al. (2021) give an overview of the beginning and the course of the digitalization process in logistics and supply chain management from a business perspective. Remarkable for logistics digitalization processes is also the very important and enabling role assignable to data, which is further discussed and explained by Daxböck et al. (2019).

The technological foundations of modern applications range from barcode recognition and data digitization by various sensors, over the optimization and data analytics to modern technologies like blockchain, deep learning and virtual reality. The relevant enabling technology in the case of our work is convolutional neural networks for image analysis. We discuss the role of computer-vision-related methods in logistics innovations in the next section.

2.1.2 Computer Vision and Deep Learning in Logistics

Our contribution utilizes means of image processing and artificial intelligence. Borstell (2018) gives an overview of how these methods have recently been used in various logistics applications. They propose a list of 11 categories of logistics applications of image processing. Of these categories, our use case of packaging structure recognition can be assigned to the one of "traceability and trackability", concerning the "identification and localization of logistics objects". While many applications of this category make use of 1-dimensional or 2-dimensional codes or optical character recognition (OCR) for object identification, our approach is rather concerned with image interpretation and analysis. The reason for this is that the goal of packaging structure recognition is not the pure identification of logistics objects, but rather the recognition and counting of transported units. Therefore, the application is also closely related to a second one of the 11 categories, namely "inspection and quality control of goods".

In many logistics applications of computer vision, but also other non-image-based systems, artificial intelligence or machine learning are utilized. Kerner et al. (2020) give an introduction to the topic of artificial intelligence, machine learning, and deep learning in the context of logistics. Woschank et al. (2020) conduct a comprehensive literature review regarding the topics of artificial intelligence, machine learning, and deep learning in smart logistics. Aiming to create a conceptual framework, they identify seven clusters for the categorization of applications of artificial intelligence in smart logistics:

1. Strategic and tactical process optimization
2. Cyber-physical systems in logistics
3. Predictive maintenance
4. Hybrid decision support systems
5. Production planning and control systems

6. **Improvement of operational processes in logistics**
7. Intelligent transport logistics

Our application of automated packaging structure recognition can be associated with cluster 6, "Improvement of operational processes in logistics", as it aims to simplify, for instance, the goods receipt process. The authors conclude their work by arguing that most relevant technologies are still in development and not at all in a productive-ready state. The same applies to our work as we provide foundations for the development of an artificial-intelligence-based system for application in the digitalized logistics industry.

2.1.3 Goods Receipt Automation and Logistics Unit Recognition

The goal of packaging structure recognition is to recognize and analyze logistics transport units within 2D images, in order to extract relevant information for logistics processes, like goods receipt and booking. The idea of automating parcel recognition is not new to the logistics sector, as various companies working on related applications show. A machine vision system by Zetes (2022) offers an automated reading of barcodes and labels on logistics units and archiving of transport unit images. Vitronic (2022) offer very similar machine vision systems for the automation of goods receipt and shipping. Package volumes can also be measured by additional engagement of laser scanners. Another camera-based system for automated barcode reading, sorting and dimensioning is offered by Cognex (2022). Logivations (2022) offer a vision system that additionally counts and measures objects within logistics units and can be trained by the user to recognize custom objects based on their appearance. None of these systems aims to capture the total number of packages in an assembled transport unit, as opposed to our work.

The number of scientific publications regarding logistics automation use cases is remarkably low. This may be due to the fact, that such specialized solutions are mainly developed in commercial settings, rather than in research environments. To our knowledge, no other scientific publication focusing on packaging structure recognition exists. Fraunhofer IML have developed a system for image-based automated counting of carriers (Hinxlage and Möller 2018), but do not provide further information on the technologies involved.

2.2 Introducing the Use Case of Packaging Structure Recognition

Within a logistics supply chain, a large variety of transported goods need to be handled and checked at many different network points. Often, huge manual effort is involved in the recognition or verification of packet identity or packaging structure, for instance, to check deliveries for completeness. We propose a reduction of such manual efforts by automated packaging structure recognition for logistics transport units on conventional 2D RGB images.

Figure 2.1: Illustration of the packaging structure recognition use case.

2.2.1 Motivation and Applications for PSR

In the following, we aim to motivate and demonstrate the need for and benefits of automated packaging structure recognition. Therefore, we exemplary name a few relevant logistics use cases to demonstrate the re-usability of our method:

1. Automated incoming goods transaction
2. Automated outgoing goods control
3. Automated empty package counting

We focus on the first of these use cases as our algorithm was primarily developed for such a setting. The other use cases are only briefly sketched to argue our method's relevance. The above list is not exhaustive and automated packaging structure recognition may be relevant for other use cases, inside and outside the domain of logistics (e.g. visual analysis of buildings from aerial images, or other uniformly organized cubic compositions).

Automated incoming goods transaction: Incoming goods processes are an essential part of every logistics supply chain, which are, for instance, described by Furmans and Kilger (2019). On receipt of logistics goods, the execution of corresponding booking operations is mandatory. In Germany, it is enforced by law (German Commercial Code (HGB) §377) that incoming goods have to be, at least externally, checked for completeness and damages immediately after reception. Commonly, the goods booking task is preceded by manual inspection of the transport units received: Transport labels are read, either manually or using a scanning device, packages are counted and the unit is checked for damages and tampering. In some cases, packages are opened for a more thorough examination. The steps included in incoming goods checks can vary. Nonetheless, package number

verification is necessary in all applicable cases. While systems for the automated image-based detection and reading of visible transport labels and bar codes on transport units exist, the whole process cannot be covered by such systems: In most cases, not all packages and transport labels can be captured in a single image as they may be attached to different sides of the transport unit or may be occluded. Additionally, often imaging requirements can be high for such algorithms to be able to read barcodes and text information on transport labels. Images taken from up close may be necessary and sources of motion blur need to be eliminated. If packages are arranged in a 3D pattern, it is not possible to picture all packages at once and simple counting does not suffice to find the total number of packaging units. Our method tackles this problem by recognizing two of the transport unit's distinct sides and counting packages individually for each side. As units are not only counted; but also their arrangement is captured, the total number of packages can be calculated.

Automated outgoing goods control: Similar to incoming goods control, outgoing goods need to be checked analogously before dispatching. On the one hand, individual transport units are checked for completeness, integrity and packaging instruction compliance. On the other hand, it has to be ensured that the transport units take the right track and are loaded onto the designated means of transport. This process can be supported by our method for packaging structure recognition: By automatizing the counting of packaging units, parts of the manual efforts involved can be eliminated, and additional sanity checks can be incorporated easily.

Automated empty package counting: As standardized packaging units, like "Kleinladungsträger" (KLTs) (compare Section 2.2.3), may be part of a deposit system, empty transport units are often still valuable resources, which need to be organized and repurposed: Once emptied, transport units are often cleaned and, subsequently, relocated for further use or temporarily stored. Hereby, they can be assembled to transport units or handled individually. Our method could be adapted and employed to enable automatized logging, booking or stocktaking of KLT empties.

2.2.2 The Idea of Automated Packaging Structure Recognition

We define the basic task of packaging structure recognition as the challenge of inferring a logistics transport unit's packaging structure from a single image of that unit. Here, the packaging structure consists of the following information:

- **Type** and **number** of packaging units
- **Arrangement** of packaging units
- Type of **base pallet**

In our work, the types of packaging units and base pallets that are distinguishable are limited to a small set relevant to the context of our test data. In many logistics use cases, the limitation to known packaging units and components does not compromise the applicability of our algorithm as the vast majority of transport units comply with explicit packaging standards and unknown packaging components are not to be expected. Multiple extensions and refinements to our method are possible

and intended, but not in the scope of this work. Some of these extensions are necessary to fully cover the use cases described in the previous subsection. For example, the additional detection of packaging components such as lids, security straps or transparent foils, could allow for automated checks of packaging instruction compliance.

Fig. 2.1 shows an example of the application of packaging structure recognition: In the image, one transport unit is fully visible. This unit is analyzed by our method and the results are illustrated by red and yellow drawings (red: transport unit sides, yellow: packaging unit arrangement).

In this work, we use a single RGB image as input to our algorithm. The reasons for which we opted for such a lightweight input are manifold: On the one hand, we aim to design an algorithm, which is as readily applicable as possible. Single images can be easily acquired using smartphones or similar handheld devices, but can also be provided by fixated industrial cameras or comparable equipment. Further, as we target the reduction of manual effort in incoming goods checks, complex image or video acquisition processes are not appropriate. On the other hand, the algorithmic complexity required to process multiple images, or even videos, is significantly higher. The same applies to the algorithm's computational requirements. In our opinion, the first approaches to the design of a multi-step image processing pipeline should not be unnecessarily complex. Still, with small image extraction efforts in an adequate pre-processing step, our method can be applied to video data: More precisely, a selector model could be used to extract single frames from a video, to which our packaging structure recognition algorithm is applied subsequently. Within this work, we exclusively use input images, which were manually acquired using conventional smartphone cameras. In future continuations of our work, algorithmic variations using more complex input image data may be explored.

The input image may depict one or multiple fully visible transport units. Only fully visible units are considered in our algorithm; partially visible or occluded units may be present in the image's background as this is often nearly unavoidable in logistics environments. Relevant transport units are assumed to be uniformly packed, which is common in many industrial sectors, and is also the case in the environment in which our dataset was acquired. Our assumptions regarding the packaging of transport units are more thoroughly explained in the following subsection. Given a single image as input, our algorithm produces information about the fully visible units and their packaging structure: Fully visible transport units are located, and, for each of these units, the type, number and arrangement of packaging units are extracted.

2.2.3 Terms and Definitions

For clarity, we introduce our understanding of a few important terms, which are frequently used throughout our work.

Package Unit: We use the term package unit to refer to a single container holding assets for transport or storage. This can, for instance, be a cardboard box, a wooden crate, or a standardized plastic container (like so-called **"Kleinladungsträger"** (KLT) units, which are relevant in our logistics setting). For our work, we focus on package units of regular cubic shape.

Base Pallet: A base pallet is the fundamental component of each logistics transport unit. Often made from wood or plastic, the pallet enables forklifts to efficiently and safely handle a transport unit. Standardized base pallets like the wooden **Euro pallet** are frequently used along logistics supply chains.

Transport Unit: A logistics transport unit is a completely assembled and packed set of logistics assets, ready for transport. In our case, we assume a 1-to-1 correspondence between transport units and base pallets. Moreover, each transport unit consists of several package units, which are placed on a logistics base pallet, and of adequate packaging, which allows for unified means of transport. Often, a lid is put on top of the packaging units, and straps or foil are used to secure the transport unit. Additionally, transport labels, allowing for identification, or danger warnings may be applied to a transport unit. All such components, which are safely attached for transport, are included in our definition of a logistics transport unit. Note that transport units may be handled and stored individually, or may be stacked on top of each other.

2.2.4 Prerequisites and Limitations

The proposed prototype method was trained to work in a well-defined logistics environment. All tests and evaluations were performed within the same setting. Thereby, restricted means that all types of transport units and components, like, for instance, packages and base pallets, are known beforehand and special requirements regarding the input image exist. The reasons for these restrictions are twofold: On the organizational side, large manual effort is required in the acquisition of realistic, annotated training and test data in varying logistics environments. On the technical side, difficulties arise due to the application of learning methods, which can only generalize to what they have seen in training. Image instance segmentation learning models recognize and distinguish only between those classes previously seen in training data. Evidently, there is research on few-shot or zero-shot learning working towards training algorithms that aim to distinguish instances of new classes, after having seen only very few or even no examples of that class in training (Wei et al. 2019). Still, in the scope of this work, we stick to well-proven deep learning approaches for object detection, which require all the object classes to be known beforehand. We made this choice, as the accuracy, which is generally achievable by such methods, is still higher compared to more recent few-shot-learning approaches. All prerequisites and limitations assumed valid are summarized in the following.

Material Restrictions: In logistics supply chains, the package types used can vary largely, depending on the industry sector, the transported goods and the companies involved. We limit our models for package recognition to a well-defined subset of package types, in accordance with our data set. The selected materials are particularly relevant in the automotive industry. The package types present in these images are standardized transport packages (KLT) (Verband der Automobilindustrie (VDA) 2013) of different sizes and colors (see Fig. 2.2 (a)) and so-called tray packages (see Fig. 2.2 (b)). Similar to package types, also the base pallet types present in the data have to be known beforehand. Our data contains two different types of base pallets: wooden Euro pallets (The European Pallet Association (EPAL) 2022) and reusable pallets made of plastic. Both types are of the same size (1200 x 800 mm).

(a) (b)

Figure 2.2: Examples of different transport unit types. (a) KLT units, (b) tray units.

Packaging Restrictions: We focus on **uniformly packed** transport units. This means, each transport unit may be composed of only one single package type and packages are ordered regularly in full layers. There are no gaps between neighboring package units and each level of package units has the same number of packages. In this case, the complete packaging structure can be inferred by only observing one image of the transport unit, if taken from the right perspective. For non-uniformly packed transport units, the task is in general not solvable as the information contained in one image is not sufficient to infer the unit's packaging structure.

Imaging Restrictions: In order for one or more transport units to be correctly recognized in an image, the image needs to meet the following criteria:

- Orientation: Transport units need to be **upright** in the image. Real-world vertical lines should be roughly parallel to the vertical image boundaries.
- Perspective (horizontal): Transport units are not shown in a **frontal perspective**, but in such a way that two sides are clearly visible and completely covered by the image.
- Perspective (vertical): The imaging device should be roughly **on a level** with relevant transport units, i.e. neither a significant top-down nor bottom-up view on the transport unit should be chosen. Moderate vertical view angles, as visible in part (b) of Fig. 2.2, are still acceptable.

This restriction aims to exclude such imagery in which the counting of packaging units is hindered notably.

- No occlusions: Relevant transport units are **completely visible** within the image. No parts of the unit are occluded by other objects or lie outside the image boundaries.

Note that, if both the orientation and perspective criteria are met, a left-most and a right-most **transport unit side** can be clearly identified for each transport unit in an image. Please refer to Fig. 2.1 for examples compliant with the above restrictions.

2.3 Summary

In this chapter, we introduced the problem of packaging structure recognition and corresponding use cases. To begin with, we also summarized the ongoing digitalization processes in logistics and briefly discussed conceptually and technically related work. We highlighted the possible benefits of fully-automated packaging structure recognition, and those of a prospective automated incoming goods check. The problem formulation for the scope of this work was stated, and applicable prerequisites and restrictions were defined. As the framework for our research is now set, we will move on to elaborate on the first of our previously formulated research questions (RQ1).

3 Basics of Computer Vision, Machine Learning and Artificial Neural Networks

3.1 Computer Vision and Image Processing

In the early 1920s, images were first digitized, transferred over large distances and reproduced. One of the early applications of digital images was the newspaper industry. Over the years to follow, digital image representation became more complex (increasing resolutions, increase of representable gray and color values) and more frequently and broadly used.

Since the 1970s, both **computer vision** and **image processing** have become vast research fields. According to Szeliski (2022), the field of computer vision can be distinguished from image processing by the "desire to recover the three-dimensional structure of the world from images and to use this as a stepping stone towards full scene understanding". Image processing, on the other hand, is not concerned with interpreting or recovering knowledge from digital images, but focuses on the artificial enhancement, manipulation or generation of such.

This section discusses the image processing and image analysis techniques relevant to our work. For a broad introduction into the history, applications and research on the topic, the reader may, for instance, refer to Vernon (1991), Burger and Burge (2016), or Szeliski (2022).

Convolution

Convolutions are a multi-purpose tool in classic image processing. Technically, convolution means applying a filter of fixed window size (e.g. 3×3 or 5×5) to an image by computing a weighted sum of spatially neighboring pixels at specific pixel locations. More precisely, given a **filter kernel** $K \in \mathbb{R}^{m \times n}$ and an image $\mathbf{x} \in \mathbb{R}^{w \times h}$, the convolution output $(K * \mathbf{x}) \in \mathbb{R}^{w \times h}$ is computed as:

$$(K * \mathbf{x})(i, j) = \sum_{\Delta i = -\lfloor \frac{m-1}{2} \rfloor}^{\lceil \frac{m}{2} \rceil} \sum_{\Delta j = -\lfloor \frac{n-1}{2} \rfloor}^{\lceil \frac{n}{2} \rceil} \left(K_{\Delta i, \Delta j} \cdot \mathbf{x}_{i+\Delta i, j+\Delta j} \right) \tag{3.1}$$

At image boundaries, the appliance of the convolution operation is not trivial. There are three different ways to handle image boundaries:

- **Full convolution**: The convolution kernel is applied in every pixel location where at least one cell of the kernel overlaps with an image pixel (possibly resulting in increased output image dimension).

- **Same convolution**: The convolution kernel is applied in every pixel where the kernel's center cell overlaps with an image pixel (resulting in identical output image dimension).
- **Valid convolution**: The convolution kernel is only applied in such pixels where the full kernel overlaps with image pixels (resulting in reduced output image dimension).

In case of full and same convolution, the image is expanded by adding artificial pixels outside the image boundaries, to enable kernel appliance near image boundaries. Two different strategies for choosing artificial pixel values are common:

- **Reflecting boundaries**, i.e. $\mathbf{x}_{-i,j} = \mathbf{x}_{i,j}$, $\mathbf{x}_{i,-j} = \mathbf{x}_{i,j}$, $\mathbf{x}_{w+i,j} = \mathbf{x}_{w-i,j}$, and $\mathbf{x}_{i,h+j} = \mathbf{x}_{i,h-j}$
- **Constant boundaries**, i.e. $\mathbf{x}_{-i,j} = \mathbf{x}_{i,-j} = \mathbf{x}_{w+i,j} = \mathbf{x}_{i,h+j} = c$ for a fixed choice $c \in \mathbb{R}$, e.g. $c = 0$.

The above boundary condition formulations hold for all $i, j \in \mathbb{N}, i \leq \lceil \frac{m}{2} \rceil, j \leq \lceil \frac{n}{2} \rceil$.

Depending on the kernel's values, various operations can be performed using convolutions. Fig. 3.1 illustrates examples of two significantly different processing steps using convolution: In the first example, a 3×3 smoothing kernel, K_1, with uniform entries and unit weight sum equal is applied (see Eq. 3.2). In the second example, a simple vertical line detection is performed by using a zero-sum kernel, K_2, where each column has uniform entries, as shown in Eq. 3.3.

$$K_1 = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \tag{3.2}$$

$$K_2 = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \tag{3.3}$$

Hough Transform

The **Hough Transform** is a patented method from 1962 for the detection of regular structures in binary images (Hough 1962). Its original version focusing on the detection of lines, or line segments, is relevant in our case. In answering our second research question (see Chapter 5), we make use of the Hough Transform to design an image analysis algorithm to detect transport unit sides - without the application of learning-based methods like artificial neural networks.

Application of the Hough Transform requires a **binary input image**, i.e. each pixel is either a foreground (pixel value one) or a background (pixel value zero) pixel. The idea for finding line structures in such an input image is to iterate over all foreground pixels within the image, and have each pixel vote for all possible lines passing through that pixel. Corresponding data is obtained by drawing an accumulator map in parameter space.

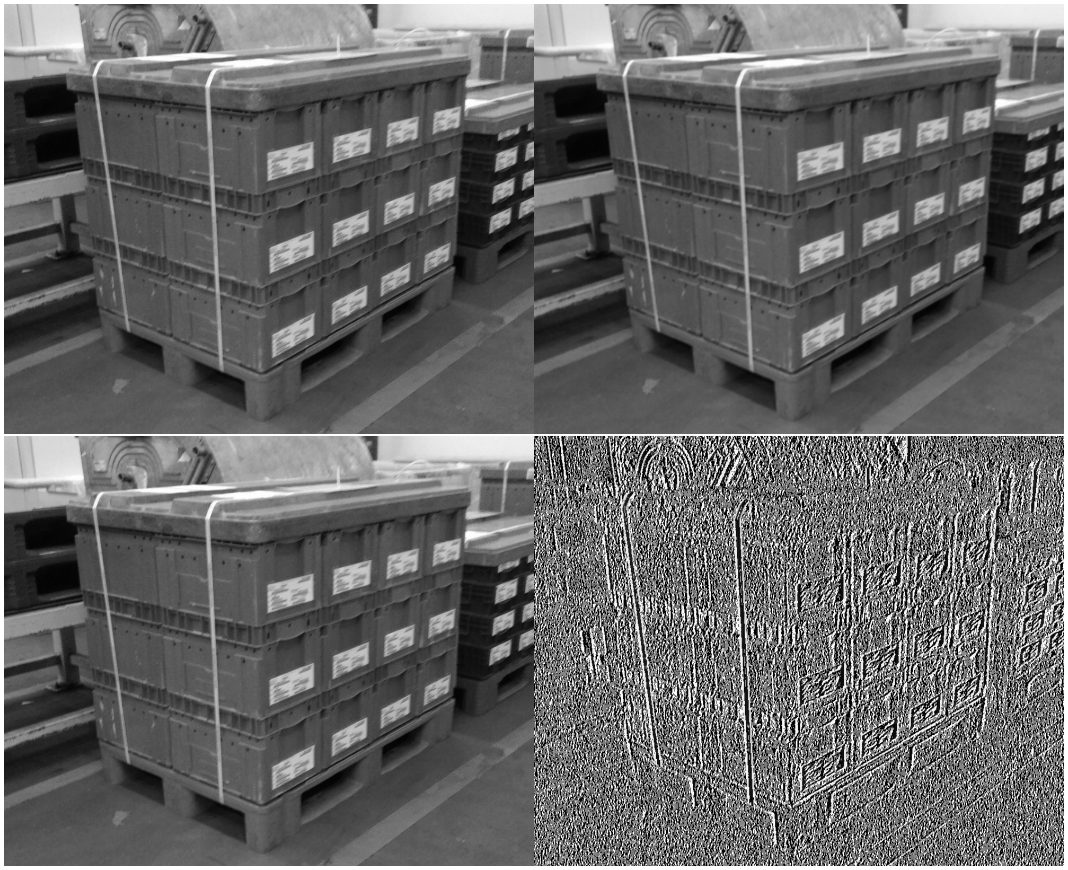


Figure 3.1: Examples for image processing operations by convolution. Top: Smoothing kernel. Bottom: Vertical line detection. Original images on the left, processed images on the right.

Every line through two points (x_1, y_1) and (x_2, y_2) in the original image's plane can be parameterized by two values $k, d \in \mathbb{R}$ with

$$y = k \cdot x + d \tag{3.4}$$

where k is the line's slope and d its y-coordinate at $x = 0$. For a single point, (x_1, y_1), in image space all lines running through that point satisfy

$$y_1 = k_i \cdot x_1 + d_i \tag{3.5}$$

This means these lines are described by parameter pairs

$$L_i = \{(k_i, d_i) | d_i = y_1 - x_1 \cdot k_i\} \tag{3.6}$$

with $k_i \in \mathbb{R}$. Once again, the geometrical shape of set L_i is a line within the 2-dimensional space spanned by parameters $k, d \in \mathbb{R}$.

If (x_1, y_1) is a foreground pixel in the image plane, this pixel contributes to any of the lines parameterized by $(k_i, d_i) \in L_i$. Thus, that point (x_1, y_1) votes for any of these lines by increasing the values for any of the parameter pairs in L_i by drawing a corresponding line on the accumulator map. This process is repeated for every foreground pixel in the input image. Afterward, line candidates within the original image can be easily identified by finding local maxima in the accumulator map.

As vertical lines cannot be parameterized in form of equation 3.4, a line representation in Hessian normal form is used in practice. This means, a line is parameterized by an angle $\theta \in [0, \pi)$ and a distance $r > 0$ as follows:

$$\cos(\theta) \cdot x + \sin(\theta) \cdot y = r \tag{3.7}$$

A more thorough discussion of the Hough transform can be found in Burger and Burge (2016).

3.2 Machine Learning and Artificial Neural Networks

This section concerns a short introduction to **convolutional neural networks** (CNNs), the state-of-the-art technology which we apply to our sub-problem regarding instance segmentation in images. First, we give a short historical overview of the superordinate research field of machine learning and artificial intelligence. Subsequently, the technical fundamentals of artificial neural networks, and specifically convolutional neural networks, are explained.

3.2.1 A Short History of Machine Learning and Artificial Intelligence

The term **machine learning** was first introduced and popularized by Arthur Samuel in 1952. Samuel (1959) worked on a computer program for playing checkers, incorporating a scoring function for arbitrary game positions and mechanisms for exploiting knowledge from previously seen positions. In his paper, Samuel assigns vast potential to the emerging technique of machine learning by stating: "Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort." Even though not properly documented, Samuel is also said to have defined machine learning as: "A field of study that gives computers the ability to learn without being explicitly programmed."

Nowadays, machine learning is mostly seen as a sub-domain of artificial intelligence (AI), on the same level as, for instance, logic-based expert systems. Even though the two buzzwords are sometimes used interchangeably, a definition of artificial intelligence is more abstract and machine learning could rather be seen as one of the most powerful tools to achieve machine intelligence.

Another popular definition of machine learning is given by Mitchell (1997): "A computer program is said to learn from experience (E) with respect to some task (T) and some performance measure (P), if its performance on T, as measured by P, improves with experience E then the program is called a machine learning program."

At about the same time as Samuel's success in teaching computers to play checkers, Rosenblatt (1958) introduced a model called the "perceptron" which was inspired by biological intelligent

systems. Thereby, the basic building block of artificial neural networks was designed. Overly high expectations were raised and exaggerated predictions regarding machine intelligence were made based on the perceptron model and other advancements of that era. This excitement was expunged shortly after when Minsky and Papert (1969) presented that the perceptron was not able to solve the not linearly separable XOR problem, and government reports by the Automatic Language Processing Advisory Committee (ALPAC) (ALPAC 1966) and by Lighthill (1972) offered dark perspectives on the research field of artificial intelligence. During the following years, a period of tremendously reduced funding and interest in AI research, which came to be known as the first "AI winter" followed.

In the 1980s, the research attention on artificial neural networks, i.e. multi-layer perceptrons, started increasing once again. This is due to significant advances made regarding the training of such complex models by the introduction of practical backpropagation by Rumelhart et al. (1986). Still, the computational resources available at that time did not yet allow for convenient and efficient training of multi-layer neural networks, given the then-available training algorithms and data. Yet again, research on artificial neural networks suffered another drawback and was widely disregarded. Still, LeCun et al. (1989) succeeded in training a convolutional neural network for the recognition of hand-written digits, using backpropagation.

In the 1990s, algorithms nowadays sometimes referred to as 'classical machine learning' became popular. More precisely, decision trees (Quinlan 1986) and random forests (e.g. summarized by Breiman (2001)), support vector machines (Cortes and Vapnik 1995), and, linear and non-linear, regression models, to name a few prominent examples, were successfully applied to various regression and classification problems.

During the early years of the 21st century, things again took a different turn. On the one hand, significant technical advancements were made, and the use of graphics processing units (GPUs) for the efficient training of complex models became possible. Simultaneously, progress regarding model design, training algorithms and training strategies was made, and large datasets, for instance regarding the task of image classification, were acquired. All these factors enabled a breakthrough of convolutional neural networks with the success of AlexNet by Krizhevsky et al. (2012) in the "ImageNet Large-Scale Visual Recognition Challenge" (ILSVRC) (Russakovsky et al. 2015), a well-known annual competition concerning the tasks of image classification and object detection. In the years to follow, research regarding artificial neural networks flourished, and remarkable results were achieved for various tasks, especially, but not exclusively, within the field of computer vision and image analysis. Until today, artificial neural networks are state-of-the-art technology for problems like object detection and instance segmentation, which are highly relevant to our work.

For more thorough elaborations on the history of AI, please refer, for instance, to LeCun et al. (2015), Toosi et al. (2021), or Fradkov (2020).

3.2.2 Deep Learning

Artificial Neural Networks

The artificial neuron, the basic building block of every **artificial neural network** (ANN), is a function that takes an arbitrary number of inputs and produces a single scalar output. This function

$f : \mathbb{R}^n \rightarrow \mathbb{R}$ is computed as a weighted sum of all input components and subsequent application of a (usually non-linear) so-called **activation function** $\sigma : \mathbb{R} \rightarrow \mathbb{R}$:

$$f(\mathbf{x}) = \sigma\left(b + \sum_{x_i \in \mathbf{x}} \omega_i x_i\right) \tag{3.8}$$

with weights $b, \omega_i \in \mathbb{R}$, $i = 1, ..., n$. The vector $\omega_i \in \mathbb{R}^n$ contains the scalar weights applied to the different input values, and $b \in \mathbb{R}$ is called bias. The values of b and ω_i are optimized during training.

The design of this function is loosely inspired by biological neurons, which are known to be arranged layerwise in a very branched fashion. As a result, each neuron is connected to multiple input and output neurons. These neurons work by the so-called all-or-none law (McCulloch and Pitts 1943): If the input signals exceed a certain threshold, the neuron fires, i.e. a signal is passed on to the succeeding neurons. Otherwise, no signal is passed on to succeeding neurons. In practice, however, the activation function σ is not necessarily modeled as a binary step function. Instead, different implementations, combining absolute values, trigonometric functions, and exponential and logarithmic functions, are used and can be reasonable. Sharma et al. (2020) introduce the most common activation functions. The model used in our experiments, Inception-v2 (Ioffe and Szegedy 2015), includes the so-called rectified linear unit (ReLU) activation function which is computed as

$$\sigma(x) = \max(0, x) \tag{3.9}$$

The most simple form of an artificial neural network consists of a single neuron only (as in the original perceptron model introduced by Rumelhart et al. (1986)). This neuron takes all model inputs as input vector x and the neuron's output, i.e. its activation value, corresponds to the ANN model's output.

To form a neural network using artificial neurons, as described above, neurons are usually arranged in layers: An **ANN layer** consists of an arbitrary number of identically built artificial neurons, each having its own weights and bias. Each neuron takes as input a vector of all outputs of the previous layer's neurons, or a vector of all input values, respectively. Analogously, the layer's neuron's output values are either used as inputs for the following layer or correspond to the model's output in the case of the final layer. This is the most basic ANN design, also called a fully-connected feed-forward network.

Other arrangements of neurons to form specialized artificial neural networks exist. Examples of such are recurrent networks, where some of the neurons are passed multiple times, or convolutional neural networks, which are relevant in most image analysis applications, and which are also applied in our case.

Convolutional Neural Networks

Compared to a standard ANN, a **convolutional neural network** has a more sophisticated design, aiming to exploit spatial dependencies in the input data. This is, for instance, especially relevant in the case of images, where neighboring pixels are undoubtedly, often closely, related.

To respect the 2D spatial adjacency of input images, or of a succeeding layer's neurons, the data has to be arranged accordingly. Thus, inputs and neuron layers are no longer represented by one-dimensional vectors, but rather by stacked two-dimensionally arranged data points. For each of the input image's color channels, one 2D data frame is contained in the input data. Intermediate layers may have an arbitrary number of channels, depending on the CNN's individual design. Intermediate layers, as well as the input image, have a consistent spatial arrangement in each layer. Thus, input images $\mathbf{x}$ of $w \times h$ pixels in width and height, respectively, and p color channels are represented as $\mathbf{x} \in \mathbb{R}^{w \times h \times p}$.

Inspired by convolutions as applied in classic image processing algorithms, each neuron applies a local filter or kernel to a spatially connected area of the previous layer, or the input layer, respectively. A local filter of window size $m \times n$, which is applied to a spatially arranged input $\mathbf{x} \in \mathbb{R}^{h \times w \times p}$, and produces an output with q different channels, can be written as $K \in \mathbb{R}^{m \times n \times p \times q}$. The convolution is succeeded by an element-wise activation function σ, as explained before. Thus, the function of a neuron within a convolutional layer of an artificial neural network can be written as

$$f_{i,j}(\mathbf{x}) = \sigma\left(K * \mathbf{x}|_{N(i,j)} + b\right) \tag{3.10}$$

with

$$N(i,j) = \left\{(i + \Delta i, j + \Delta j)\middle| -\frac{m-1}{2} < \Delta i < \frac{m}{2}, -\frac{n-1}{2} < \Delta j < \frac{n}{2}\right\} \tag{3.11}$$

$$N(i,j) \subset \mathbb{N} \times \mathbb{N} \tag{3.12}$$

$$|N(i,j)| = m \cdot n \tag{3.13}$$

$$\mathbf{x}|_{N(i,j)} \in \mathbb{R}^{m \times n} \tag{3.14}$$

$$b \in \mathbb{R}^q \tag{3.15}$$

This means, $N(i,j)$ describes the **spatial neighborhood** at location (i,j) and b is the neuron's bias term, as before. In equation 3.10, $*$ describes a channel-wise convolution operation producing q different output channels:

$$* : \mathbb{R}^{m \times n \times p \times q} \times \mathbb{R}^{m \times n \times p} \to \mathbb{R}^q \tag{3.16}$$

The activation function σ is structurally and may be computationally identical to the activation functions used in traditional ANNs.

Analogous to convolutional operations, there are common choices regarding the application of convolution windows at spatial boundaries:

- **Same padding**: Additional neurons are augmented at layer boundaries in such a way, that the window can be applied at every neuron, even at corners. (For a window of size $m \times n$, $\lfloor \frac{m}{2} \rfloor$, and $\lfloor \frac{n}{2} \rfloor$, additional units are required, respectively.) This allows for the preservation of the original spatial dimension.
- **Valid padding**: No additional units are inserted at boundaries. Consequently, the convolution operation can not be applied at locations close to the layer's boundaries, resulting in the reduction of the layers' spatial size.

Note that, contrary to convolutions in image processing, CNN's convolutional kernels are not hand-crafted to detect certain structures or perform specific operations. Instead, all kernel parameters are learned and optimized to detect features relevant to the specific task and the data at hand. More precisely, one does not deliberately design the convolutional kernels to perform a smoothing operation or detect lines (compare Section 3.1). The CNN rather establishes the most suitable convolutional kernels according to the data during the training process.

Apart from using convolutional neurons in their layers, convolutional neural networks also follow typical design patterns differing from those of traditional fully-connected networks. As is common in CNNs, convolutional layers are often alternated with pooling layers. Further, most network architectures conclude with one or multiple non-convolutional, fully connected layers to obtain output predictions.

In most cases, **pooling layers** are used to reduce the feature map sizes of layers of neurons, usually using a simple maximum or an average operation. This pooling operation is applied layer-wise to a small local neighborhood of window size $k \times l$ (often $k = l$). Widely used window sizes for pooling operations are, for instance, 3×3 or 5×5. To achieve a dimensionality reduction, the pooling operation is not performed at each location of the input feature map. Instead, it may be applied to every 2nd or every 3rd pixel in each row and column, thereby drastically reducing the number of output values, i.e. the next layer's number of neurons. The regular pattern in which the windows are applied is called the layer's stride. Analogously to convolutional layers, valid or same padding may be applied at layer boundaries. Note that pooling layers do not contain any trainable parameters and, thus, are static parts of the neural network.

More detailed explanations of the motivation, theory and architecture of convolutional neural networks are available in the literature, see e.g. Koonce (2021) or Aggarwal (2018).

Neural Network Training: Loss Function, Numerical Optimizers, Backpropagation

Loss Function: The training of artificial neural networks is a **supervised training** task. This means annotated training data $\Omega = \{(\mathbf{x}, \mathbf{y})_i\}_{i=1,\dots,s} \subset \mathbb{D} \times \mathbb{V}$ is required for the process. $\mathbb{D}$ is the problem domain, e.g., in our case, two-dimensional color images of size $h \times w$. $\mathbb{V}$ is the CNN model's target domain, for instance, a class index in case of classification tasks, or more complex structured bounding box predictors in case of object detection (see Section 3.3.2). The number of training data instances is denoted by $s \in \mathbb{N}$.

Assume a prediction model $f_\theta : \mathbb{D} \rightarrow \mathbb{V}$ with parameter vector $\theta \in \mathbb{R}^r$. In the case of ANNs and the notation used above, the parameter vector contains all the neuron's weights ω_i, ω_0. As is the case in traditional supervised machine learning, the goal of training a deep prediction model is to find model parameters $\theta \in \mathbb{R}^r$ such as to minimize the difference between the model's predictions $f_\theta(\mathbf{x})$ and the corresponding data label $\mathbf{y}$ for all training data samples $(\mathbf{x}, \mathbf{y}) \in \Omega$. To measure this difference, a so-called **loss function** L, measuring this difference for arbitrary model parameter choices, is used:

$$L : \mathbb{R}^s \rightarrow \mathbb{R} \tag{3.17}$$

$$L(\theta) = \sum_{i=1}^{s} L_i(\theta) \tag{3.18}$$

The loss function aims to incorporate a performance measure for the prediction model: The higher the loss value, the less accurate the predictions with respect to the training data. In theory, the model's performance is optimal if the loss function is at its global minimum. (Optimal in this case does not necessarily mean error-free but may be constricted by several factors. These include the model's potential, the data's separability, but also the loss function.)

Depending on the model type, data dimension and structure, and training objectives, a wide variety of different loss functions are applicable. As visible in equation 3.18, in the case of neural networks, the loss function is computed as a sum of losses for all training data samples. Thus, $L_i : \mathbb{R}^r \rightarrow \mathbb{R}$ describes the loss contribution of a single training data instance. L_i can be rewritten as:

$$L_i(\theta) = L_i(f_\theta(\mathbf{x}_i), \mathbf{y}_i) \tag{3.19}$$

The actual process of training a deep neural network corresponds to the minimization of the aforementioned loss function. As a deep learning model's function is non-linear and highly complex, minimization is performed using a numerical optimizer, e.g. gradient descent, and, more importantly, backpropagation. Both concepts are shortly reviewed in this section.

Gradient descent: Gradient descent is a classical numerical optimization algorithm that goes back as early as 1847 when it was introduced by Cauchy et al. (1847). The iterative method builds on the fact that a function's gradient points in direction of its steepest ascent. That knowledge is exploited to perform consecutive steps in descending directions thereby approaching a local minimum. For gradient descent to be able to find a function's **global minimum**, the function is required to be both differentiable and convex. Differentiability is required to compute gradients at arbitrary points of the function's domain. Convexity ensures the function has exactly one local minimum, which coincides with its global minimum.

For a differentiable and convex function $f : D \rightarrow \mathbb{R}$, with arbitrary domain D, one iteration, i.e. one gradient descent step, can be written as:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \lambda \cdot \nabla f(\mathbf{x}_n) \tag{3.20}$$

If step size $\lambda > 0$ is chosen sufficiently small, it holds

$$\mathbf{x}_{n+1} \leq \mathbf{x}_n \quad n \in \mathbb{N} \tag{3.21}$$

$$\lim_{n\to\infty} \mathbf{x}_n = \tilde{\mathbf{x}} \tag{3.22}$$

for arbitrary choice of $\mathbf{x}_0 \in D$ and global minimum $\tilde{\mathbf{x}} = \underset{\mathbf{x}\in D}{\operatorname{argmin}} f(\mathbf{x})$. In the case of a differentiable, but non-convex function f, the limit $\tilde{\mathbf{x}}$ is a local minimum of f, but not necessarily a global one.

When training a neural network using a loss function as described above, the objective is to minimize the value of a non-linear nested function including a high number of parameters. The function's gradient, i.e. its partial derivative with respect to each parameter has to be computed. These computations are not trivial and caused significant problems in the training of neural networks in their early days (Schmidhuber 2015). The remedy, in this case, is a technique called backpropagation which is explained hereafter.

Backpropagation: Backpropagation is an optimization algorithm introduced in the 1970s which was first applied to the training of artificial neural networks by Rumelhart et al. (1986). The idea is to iteratively adjust the network weights by computing a **backward pass** through the network structure based on the chain rule of differentiation: In each training step, the computed output costs are backpropagated through the network, thereby computing cost function gradients with respect to each weight within the network. This information can be used to perform gradient descent steps for all parameters in order to approach a local (or global) minimum. Analogous to gradient descent, backpropagation requires the function it is applied to, to be differentiable. Chauvin and Rumelhart (1995) is an earlier work discussing the theory of backpropagation elaborately.

While deep neural networks theoretically can be trained using vanilla gradient descent and backpropagation, usually more sophisticated gradient-descent-based optimizing algorithms are applied. Ruder (2016) gives a comprehensive overview of typical choices of such algorithms for the training of artificial neural networks. In our case, Polyak's **momentum optimizer**, which goes back to Polyak (1964), is used. This commonly applied algorithm makes use of a so-called momentum term v, which can be interpreted as incorporating a weighted average of previous steps into the current update. The update procedure can be written as

$$v_{n+1} = \mu v_n - \lambda \nabla f(\mathbf{x}_n) \tag{3.23}$$

$$\mathbf{x}_{n+1} = \mathbf{x}_n + v_{n+1} \tag{3.24}$$

with momentum parameter $\mu \in [0, 1]$ and learning rate $\lambda > 0$.

Another important point to consider when training deep learning models using gradient descent methods is the **batch size**, i.e. the amount of data considered in each iteration. While the objective is to find parameters in such a way, that optimal results are achieved for the whole set of training data, it is not necessarily the best approach to consider all the data in each training iteration. Basically, three options are available in that regard:

- **batch gradient descent**: the whole data is considered in each iteration,
- **mini-batch gradient descent**: a small number of data instances are considered in each iteration (depending on data type, model size, or available computation hardware),
- **stochastic gradient descent**: only a single instance of the data set is considered in each iteration.

Note that batch gradient descent corresponds to joint minimization of loss function L, while stochastic gradient descent alternately minimizes single loss components L_i (see Equation 3.18). In deep learning practice, batch gradient descent may be too computationally expensive to be performed on available hardware resources. At the same time, mini-batch and stochastic gradient descent methods are less likely to get stuck in local minima, as the individual update steps are randomized, depending on the current iteration's data selection. Masters and Luschi (2018) and Wilson and Martinez (2003) analyze the advantages of using smaller batch sizes.

A more thorough and detailed explanation of the process of training artificial and convolutional neural networks can, for instance, be found in Aggarwal (2018) or Goodfellow et al. (2016).

Overfitting and Regularization

One of the most prominent problems with training machine learning models is **overfitting**. This phenomenon occurs, when the model memorizes the training data beyond generalization. That means, the model does not learn to recognize general properties and features of the domain's data but is able to recognize every single training example by its individual characteristics, or even by the noise within the data. Overfitting can be diagnosed by observing the model's performances on training and validation data during the training process: In the case of over-fitting, the performance on training data is increasing further while validation performance is in decline.

The effect of overfitting generally can be observed if the model representation abilities, i.e. the number of parameters, is too great relative to the amount of training data. Consequently, it can be tackled by either choosing a smaller model, which is not able to simply memorize the whole training data set or by increasing training data variance. The latter can not only be achieved by increasing the number of available training samples, but also by incorporating **data augmentation** techniques into the data pre-processing. Algorithmic techniques to reduce overfitting, called **regularization**, exist. A traditional approach is the addition of regularization terms, which penalize excessive parameter utilization, to the loss function. Deep-learning specific procedures are, for instance, dropout (Hinton et al. 2012) or batch normalization (Ioffe and Szegedy 2015). Apart from that, overfitting can be avoided by limiting training time such that the training process is terminated when no further increases in validation performance are achieved. This concept is called early stopping. Ying (2019) gives an overview of possible remedies to overfitting.

Parameter Initialization

Another aspect not to be neglected when training deep neural networks is **parameter initialization**. Simple zero or constant initialization of parameters is not a valid choice as it can cause all of the network's neurons to behave identically (based on their position within the network), leading to redundant units. Further, the magnitude at which unit parameters are initialized is crucial for convergence speed. Goodfellow et al. (2016) discuss neural network parameter initialization in more detail and introduce common initialization strategies. As we use a transfer learning approach, and thus do not need to choose initial parameters, the problem is not discussed further here. Instead, transfer learning is explained in the following paragraph.

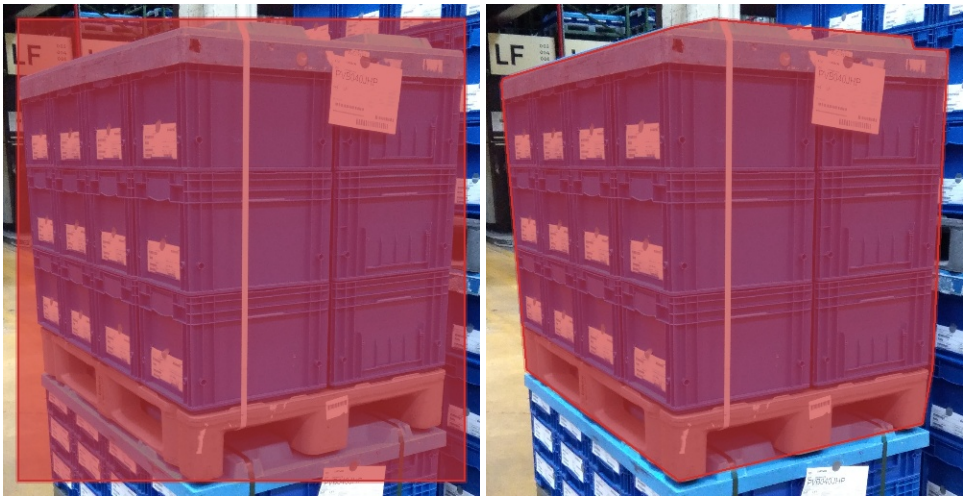
Transfer Learning

Features learned by convolutional neural networks are, to some degree, task-agnostic. That can be seen in the widespread use of a technique called **transfer learning**. This term refers to the usage of already trained neural networks for different use cases. It is frequently applied, e.g. in cases of limited data availability, or when employing synthetic training data. Pan and Yang (2009) and Weiss et al. (2016) provide comprehensive surveys on the general topic of transfer learning, including definitions and application examples. More specifically, Li et al. (2019) show that transfer learning approaches are highly beneficial for object detection tasks with limited data availability. Huh et al. (2016) analyze the usage of ImageNet for CNN pre-training and find that dataset size is not an ideal predictor for the suitability of a pre-training dataset.

It is common practice in vision applications to use models which were pre-trained on large, publicly available datasets like **ImageNet** (Russakovsky et al. 2015) or **COCO** (Lin et al. 2014). Training of such pre-trained models is then continued on a use-case-specific dataset. This is especially helpful if the amount of data available for the specific use case is limited, or if there is a **domain shift** between available training data and inference data. The latter is often the case if synthetic training data shall be used to train a model for application in real-world scenarios.

3.3 Object Detection and Instance Segmentation

In this section, we outline the general tasks of **object detection** and **instance segmentation** in two-dimensional images. While both tasks concern the identification of specific objects within images, the granularity at which object instances are localized differs significantly: Object Detectors find minimal **bounding boxes** containing instances of target objects, while Instance Segmentation models try to segment object instances as accurately as possible, i.e. by **pixel masks**. Fig. 3.2 illustrates these two types of object representations on one of our use case images. Note that the bounding box representation generally covers an image part larger than the actual object, i.e. additional pixels which do not show object parts are included in the object's bounding box. The object segmentation mask, on the other hand, includes only pixels covering the corresponding object.



(a) (b)

Figure 3.2: Illustration of object representations, i.e. localization granularity, in the tasks of (a) Object Detection and (b) Instance Segmentation. The object class to be detected are complete transport units (including base pallet, packages, and lid).

For both tasks, identical CNNs for feature extraction can be used. What differs are the subsequent layers that output either bounding box representations or pixel masks, for each categorized object instance, respectively.

In this section, both tasks, corresponding data representations and relevant detection models are explained briefly. A more thorough introduction, including historic developments and milestones, as well as state-of-the-art approaches, can, for instance, be found in Szeliski (2022).

3.3.1 CNN Architectures for Image Analysis

A groundbreaking work on image classification involving CNNs was published by Krizhevsky et al. (2012). The authors achieved top results in the 2012 **ImageNet Large Scale Visual Recognition Challenge** (ILSVRC) (Russakovsky et al. 2015), leading the competition by a huge margin (15.3% top-5-error rate [1] vs. 26.2% for the second place competitor). The ImageNet challenge is an image classification contest, providing a dataset of more than one million images showing instances of objects of 1,000 different categories. Participating algorithms are required to predict up to five object categories per image, ranked by confidence. What was remarkable in the 2012 version of the challenge was that the winning model, which is widely known as **AlexNet**, was a deep convolutional neural network. Krizhevsky et al. (2012) were the first to efficiently design and train such a model for the task of image classification.

[1] The top-5-error rate of a model corresponds to the ratio of test instances, for which the true image class was within the model's five highest-rated class predictions for that image.

AlexNet operates on images of resolution 224×224 pixels and consists of eight main layers: Five convolutional layers, followed by three fully-connected (dense) layers. The first convolutional layer is applied with stride 4 to reduce spatial image resolution. With the same objective, maximum pooling with stride 2 is applied after the first two and the last convolutional layer. Convolutional kernel sizes are 11×11 in the first, 5×5 in the second, and 3×3 in all following convolutional layers. Spatial resolution decreases, and feature number increases, as the image is passed through the model. This is illustrated in more detail by Fig. 3.3.

The architecture was slightly modified in order to split computations among two GPUs. For more details, please refer to the author's original work.

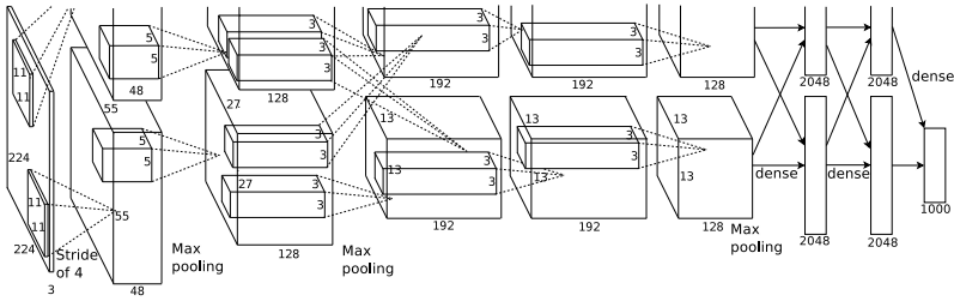


Figure 3.3: Architecture of AlexNet. Source: Krizhevsky et al. (2012)

AlexNet was adapted for many applications and more advanced algorithms in similar tasks, e.g. in Region-based CNNs for object detection (see section 3.3.2).

After the success of AlexNet in 2012, deep CNNs quickly became the go-to choice for high-performance image analysis models. Performances in recognition challenges like ImageNet and the common objects in context (COCO) object detection challenge (Lin et al. 2014) were steadily increased by further optimizations of CNN architectures.

Szegedy et al. (2015) introduced the concept of **inception**, achieving the top result in the 2014 ImageNet challenge. The idea of inception modules is to be able to capture and summarize features of different sizes at the same time. In summary, inception modules perform multiple convolutions with different kernel sizes in parallel (e.g. 1×1, 3×3, and 5×5). To avoid tremendous computational costs, 1×1-convolutions reducing the number of channels, precede the convolutional layers employing larger kernel sizes. Apart from the different convolutional layers, an additional maximum pooling layer is added in parallel. Fig. 3.4 shows the structure of a single inception module. The proposed network, GoogLeNet, consists of three conventional convolutional layers (the first of which is the only one with a larger filter size of 7×7, the filter sizes of the second and third convolution operations are 1×1 and 3×3, respectively). The first and third convolutional layers are followed by a max-pooling layer. Following these layers, a total of nine inception modules, with increasing channel numbers, are employed. These nine inception modules are interjected by another two maximum pooling layers causing a reduction of spatial dimensions. The network architecture is completed by an average pooling layer, a dropout layer, one fully-connected layer, and, finally, a softmax layer. The last layer consists of 1,000 neurons predicting class probabilities for each of ImageNet's 1,000

object categories. Image input resolution is 224×224 pixels, this spatial resolution is progressively reduced to 7×7 after the last maximum pooling operation.

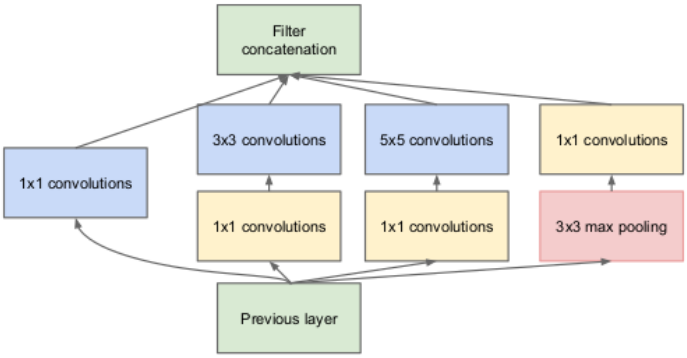


Figure 3.4: Inception module. Source: Szegedy et al. (2015)

Ioffe and Szegedy (2015) introduced a concept called **batch normalization**, and corresponding layers for deep neural networks. The operation performed by these layers is a normalization of activation values with respect to the training data batch, and the layers thereby act as powerful regularizers.

Assume $x_i, i = 1, ..., t$ are the activations of a single neuron for a training batch $\mathbb{B}$ of size t. To apply a batch normalizing transform to these activations, mean $\mu_{\mathbb{B}}$ and variance $\sigma_{\mathbb{B}}$ of the batch's activation are computed as

$$\mu_{\mathbb{B}} = \frac{1}{t} \sum_{i=1}^{t} x_i \tag{3.25}$$

$$\sigma_{\mathbb{B}} = \frac{1}{t} \sum_{i=1}^{t} (x_i - \mu_{\mathbb{B}})^2 \tag{3.26}$$

The output activations are normalized:

$$\hat{x_i} = \frac{x - \mu_{\mathbb{B}}}{\sqrt{\sigma_{\mathbb{B}} + \epsilon}} \tag{3.27}$$

Finally, the transform's output y_i is computed as

$$y_i = \gamma \hat{x_i} + \beta \tag{3.28}$$

where γ and β are learnable parameters. The scale and shift of these parameters are introduced to increase the space of transformations representable by batch normalization.

Batch normalization transforms can be added to neural networks after arbitrary neurons, each such operation introducing two additional learnable parameters.

In the same publication (Ioffe and Szegedy 2015), the authors use batch normalization to enhance GoogLeNet to an architecture often referred to as **Inception-v2**. Inception-v2 uses slightly different inception modules, replacing all 5×5 convolutions with two consecutive 3×3 convolutions. Apart from that, fully-connected layers were removed and some pooling layers were replaced by stride 2 in convolutional operations. A total of 10 inception modules are employed. Batch normalization is applied to the input of each nonlinearity (activation function). Inception-v2 does not only outperform its predecessor on ImageNet (top-5-error of 4.9% compared to 6.7%), but it also trains significantly faster needing 5 times fewer training steps, as reported by the authors.

The 2015 ImageNet challenge was won by He et al. (2016) with their Residual Network, short ResNet, adding skip connections to convolutional layers. After that, the inception architecture described above was improved further by Szegedy et al. (2016b) and Szegedy et al. (2016a).

Since 2017, additional improvements and ideas were incorporated into network designs, not all of which will be covered here. For instance, Huang et al. 2017 propose DenseNet, an architecture with additional skip connections between each pair of layers within the network. In 2019, Tan and Le 2019 introduced EfficientNet, a CNN architecture with integrated scaling methods, to achieve optimized depth, width and resolution of the network and its layers. The currently best-performing network on the ImageNet challenge data is CoCa, which was introduced by Yu et al. 2022 (CoCa reports a top-1-error of 9.0% as compared to Inception-v2 with 20.1%). Recent literature provides more sophisticated overview of the most important CNN designs, for instance, see Alzubaidi et al. 2021, or Khan et al. 2020.

However, we employ the Inception-v2 network as a feature extractor CNN in the training and evaluation of our algorithm. The reason for our choice is that the architecture is implemented and readily available in tensorflow's object detection library (Abadi et al. 2016). Additionally, its computational complexity is comparably low and its training and run times are shorter than those of many successor networks. Note that our work does not primarily focus on exploring the optimal CNN for the task of packaging structure recognition. Rather, we aim to design an appropriate algorithm for the use case, employing a reasonable CNN as image analysis tool, in the first place. We are aware, that incorporating different, or even hand-crafted, deep learning models might yield improved results. We start exploring this question in our last research question (RQ3) and we might elaborate on it further in future research.

3.3.2 Object Detection

Object Representation

In multi-class object detection, object instances are represented, or localized, by the minimal rectangular area covering the whole object. Moreover, a **bounding box** B_{x_o,y_o,w_o,h_o} of width w_o and height h_o, positioned at location (x_0, y_0), is defined as

$$B(x_0, y_0, w_0, h_0) = \{(x, y) \in P | x \in [x_0, x_0 + w_0] \wedge y \in [y_0, y_0 + h_0]\} \tag{3.29}$$

Thereby, it is assumed that x_0, y_0, w_0, h_0 are chosen such that $x_0 + w_0$ and $y_0 + h_0$ lie within image plane P.

Metrics for Object Detection

Let B_d and B_{gt} denote the bounding boxes of a model's detection on an image, and of one of the image's ground-truth annotations, respectively. To decide whether the detected object with bounding box B_d can be matched to the annotated object with bounding box B_{gt}, the **intersection over union** (IoU) of the two bounding boxes is computed as

$$\text{IoU}(B_d, B_{gt}) = \frac{\|B_d \cap B_{gt}\|}{\|B_d \cup B_{gt}\|} \tag{3.30}$$

where $\|\cdot\|$ denotes the size of the corresponding areas in 2-D Euclidean space. In the case of discretized formulations, as is the case for pixel-based digital images, the number of pixels within each set (area) can be counted. Using the IoU, and applying a decent threshold value (empirically adjusted to the individual problem, data and network), the detections found by an object detection model can be matched to the data annotations. Every detection can thus be classified as either true positive (TP) or false positive (FP) detection. Hereby, one needs to be careful to assign at most one detection to each ground-truth annotation (e.g. by application of suitable non-maxima suppression). Further, object categories need to be respected in the case of multi-class object detection. **Precision** and **Recall** are common performance measures for object detection models that can be computed based on the TP/FP-categorization: The model's precision is the number of true positive detections divided by the total number of detections. The model's recall is the number of true positive detections divided by the number of annotated objects and can be seen as the model's sensitivity.

Generally, a confidence threshold is applied when selecting the model's object proposals. By computing precision and recall for different confidence thresholds, a precision-recall-curve can be created by plotting the two values against each other on the x- and y-axis. The area under this curve (PR AUC) is a frequently used performance measure for detection models called **Average Precision** (AP). In the case of multi-class detection problems, scores are often computed separately for each object category. The **mean average precision** (mAP) is then computed as the average value of all categories' AP values. The mAP was originally introduced by the Pascal Visual Object Classes (VOC) image recognition challenge in 2007 [2] (Everingham et al. 2010). In its original version, the computation is performed for a fixed, predefined IoU threshold, e.g. 0.5. In the course of the common objects in context (COCO) object detection challenge (Lin et al. 2014), an evaluation metric using multiple IoU threshold values was established: An overall performance measure is computed by averaging the mAP value for every IoU threshold value in $\{0.5, 0.55, ..., 0.9, 0.95\}$. We denote this measure by $mAP_{0.5:0.95}$ (whereas $mAP_{0.5}$ denotes the mAP at IoU threshold 0.5). Note, that the mAP value generally decreases as the applied IoU threshold, and thereby the requirement regarding detection accuracy, increases.

[2] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

R-CNN and CNN-Based Object Detection

State-of-the-art object detectors use CNNs for feature extraction from input images and generate region proposals to localize object instances within the image. The earliest such model was the **Region-based Convolutional Network** (R-CNN) proposed by Girshick et al. (2013). R-CNN uses a selective search algorithm to generate 2,000 region proposals for an input image. For each region proposal, an image crop is created and run through a CNN feature extraction network (AlexNet, see Krizhevsky et al. (2012)). A subsequent support vector machine operates on the resulting 4,096-dimensional feature vector to produce object category predictions. Simultaneously, the CNN feature vector is fed into a bounding box regressor, which refines the initial region proposal.

One year later, Girshick (2015) improved the approach, introducing **Fast R-CNN**. The category predicting SVM and bounding box regressor were replaced by fully connected layers, which perform both tasks at once. Maybe even more important, the improved version only requires a single pass through the feature extraction CNN. Selective search is now applied to the CNN's output feature map. Accordingly, crops are taken directly from the convolutional feature map, rather than the original input image.

Another improvement to (Fast) R-CNN was proposed shortly after by Ren et al. (2017). They replaced the selective search algorithms with another CNN that produces region proposals. This region proposal network (RPN) employs a sliding window approach, as well as a set of anchor boxes: The RPN is applied to a $n \times n$ window of the feature extraction CNN's output feature map. At each location, k, anchor boxes, i.e. potential object regions of pre-defined sizes and aspect ratios are applied. For each such anchor box, the probability of that box containing an object, as well as region refining parameters, are predicted by subsequent layers. The improved algorithm was named **Faster R-CNN** by the authors.

Lin et al. (2017) introduce **Feature Pyramid Networks** (FPN) to detect objects at different scales, using Faster R-CNN on multiple feature maps of different scales.

Other CNN-based object detection algorithms, which require only a single pass through a CNN detection network exist, e.g. Redmon et al. (2016) and Liu et al. (2016). As we opted for Faster R-CNN in our implementations, these are not relevant to our work and they are not discussed here.

CornerNet

A little later, Law and Deng (2020) proposed **CornerNet**, a CNN-based object detector not relying on anchor boxes or rectangular region proposals. Instead, the locations of bounding boxes' top left and bottom right corners are predicted independently. The grouping of two detected bounding box corners to a single object instance is performed separately in a subsequent step.

Moreover, CornerNet mainly consists of two main components: A CNN feature extractor, and two parallel **corner prediction modules**. This architecture is illustrated by Fig. 3.5.

In the original work, the CNN feature extractor is an **hourglass network**, as introduced by Newell et al. (2016). The hourglass module was initially designed for the task of human pose estimation. Hourglass networks perform aggressive downsampling of the input signal by alternating convolutional and max

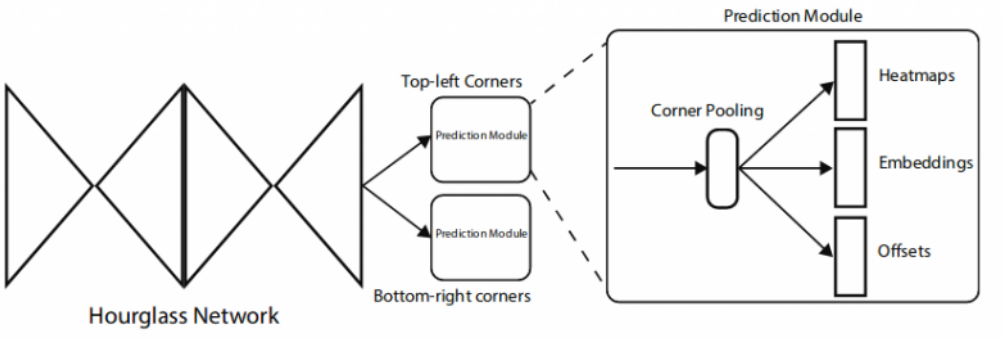


Figure 3.5: Schematic architecture of CornerNet. Source: Law and Deng (2020).

pooling layers, until a resolution as low as 4×4 pixels is reached. Subsequently, a symmetrical upsampling to the original dimension is performed. Additionally, skip layers are added between layers of identical resolution. Thus, on the upsampling path, upsampled features are combined (by addition) with features on the same scale, taken from the downsampling path. In this way, the architecture aims to detect features of different scales efficiently. Thereby, residual modules are used throughout the network, whenever convolutional layers are incorporated. For more details, please refer to Newell et al. (2016). In the case of CornerNet, two hourglass modules are used, and some adjustments are made to the modules themselves. Instead of max pooling, stride 2 convolutions are used to reduce feature resolution. Further, resolutions are reduced five times and the number of feature maps is increased at lower resolutions, compared to the original hourglass design.

The hourglass feature extraction layers are succeeded by corner prediction modules. Each corner prediction module consists of a novel kind of pooling layer, so-called corner pooling layers, followed by three parallel prediction modules. Corner pooling is based on the apriori knowledge that top left and bottom right corners of object bounding boxes are predicted: For top left corners, it can be assumed that key object features can be found toward the bottom and the right of the image, relatively. Thus, maximum pooling is exclusively performed in these directions to capture relevant features. Correspondingly, left and top direction pooling are applied in the prediction module concerning bottom right corners. Note again that corners of bounding boxes are predicted, i.e. the predicted positions themselves are not necessarily part of the object as the bounding box generally exceeds the object. The pooling layer is followed by three independent, parallel layers predicting corner locations, corner offsets and embeddings. Corner locations are predicted by scalar heat maps, indicating probabilities of corners of specific object categories. For each object category, one corner heat map is predicted. Corner offset maps are class-agnostic 2-D maps predicting sub-pixel offsets to corner locations indicated in the previously mentioned heat maps. This is necessary as the map resolution at this point is lower than the original image's resolution and high location accuracy is required. The last module predicts embedding vectors at each location. These embeddings are used to group individual corners into bounding boxes: Very similar embedding vectors indicate that two corners can be assigned to identical objects. This procedure is based on the work on Associative Embedding by Newell et al. (2017).

CornerNet reported top accuracy values on the MS COCO dataset. The authors improved their original work further in Law et al. (2020).

3.3.3 Instance Segmentation

Object Representation

In instance segmentation, objects are localized by their exact **segmentation masks**. In the case of an image of size $h \times w$, a segmentation mask S^o for an object o can be written as a binary map of the same dimensions:

$$S^o \in \{0,1\}^{h \times w} \tag{3.31}$$

$$S^o_{i,j} = \begin{cases} 1 & \text{if pixel } i,j \text{ belongs to object } o \\ 0 & \text{otherwise} \end{cases} \tag{3.32}$$

Alternatively, in the case of convex objects, instance segmentation masks can be represented by a set of points describing the mask contours:

$$P^o \subset \{0,...,w\} \times \{0,...,h\} \tag{3.33}$$

In this case $S^o_{i,j} = 1$ if and only if pixel i,j lies within the polygon described by P^o.

Metrics for Instance Segmentation

Basically, the same metrics as in object detection can be applied for the evaluation of instance segmentation models: average precision (AP) and mean average precision (mAP) are commonly used measurements.

Notably, the computation of IoU differs according to the relevant object representation. Instead of rectangular bounding box regions, segmentation masks are considered:

$$\text{IoU}(S_d, S_{gt}) = \frac{\|S_d \cap S_{gt}\|}{\|S_d \cup S_{gt}\|} \tag{3.34}$$

where S_d and S_{gt} are detected and annotated instance masks of a single image.

Mask R-CNN

Maybe the most famous CNN-based instance segmentation algorithm is **Mask R-CNN**, which is also the algorithm we employ in our work, and which was introduced by He et al. (2017). Mask R-CNN is based on Faster R-CNN and adopts most of its original architecture. The most important change is the addition of fully convolutional layers for mask prediction. Further, a so-called region of interest (RoI) align layer is added to preserve exact spatial locations despite the downsampling performed in convolutional layers of the CNN. Thus, Mask R-CNN still performs the task of object detection but additionally outputs accurate binary segmentation masks for each detected object.

Mask R-CNN's mask prediction layers run in parallel to box and class predictors. For every object category, a mask of fixed quadratic size is produced (e.g. 14×14 pixels in the original implementation using ResNet (He et al. 2016) as feature extractor network). Only the mask for the predicted object category is used further in training and inference.

The second significant novelty in Mask R-CNN is the introduction of the **RoI align layer**, which is used instead of Fast R-CNN's RoI pool layer. RoI align avoids rounding of coordinates and quantization of feature map values when constructing small feature maps of RoIs for further processing (e.g. classification). Instead, sub-pixel accuracy is used when constructing the feature map from the backbone CNN's output layer, and four values for each of the feature map's bins are found by using bilinear interpolation. The bin's value is determined using an aggregation function like maximum or average. The authors report accuracy improvements of up to 50% by the use of RoI align instead of RoI pool.

3.4 Summary

In this chapter, the theoretical backgrounds and technical fundamentals relevant to our work were discussed. In the following chapters, we will use these fundamentals to answer the three previously formulated research questions regarding the use case of packaging structure recognition.

More precisely, we introduced the basics of machine learning and deep learning, leading to the concrete architectures and techniques we employ in our learning-based image analysis components, as implemented in our image analysis pipeline to answer our first research question (see Chapter 4). We also discussed the preliminary work, which we use as fundamentals for our specialized detection models incorporating geometrical apriori knowledge. With the design of these models, which are presented in Chapter 6, we aim to answer our second research question. Further, the computer vision techniques relevant to the experiments in the scope of our second research question, regarding the algorithmic choices of our image analysis pipeline, were presented (see Chapter 5).

4 A Method for Automated Packaging Structure Recognition in Single RGB Images

In this chapter, a possible solution to the problem of automated packaging structure recognition is presented. The algorithm and part of the evaluations were previously published in Dörr et al. (2020a). However, evaluation values differ slightly due to the refactoring of the dataset.

4.1 Research Question Elaboration

The research question (RQ1), which we tackle in this chapter was already introduced in Section 1.2, and reads as follows:

(RQ1) How can information about the constitution, assembly and condition of a packed transport unit be inferred from a 2D image?

We straightforwardly answer this research question by proposing an image processing pipeline fulfilling the stated requirements: The pipeline's input is a single 2D image showing at least one packed transport unit. Its output contains information about the contained transport units and their packaging structure, i.e. the type of packaging units, and their total number and arrangement.

To prove the solution to the research question, we implement the proposed pipeline, utilizing state-of-the-art image recognition algorithms, i.e. CNNs for object instance segmentation. For training and evaluation of these models, and validation of the image processing pipeline, we employ a custom dataset of use case images, which is also introduced in this chapter.

To the best of our knowledge, we are first to formulate the task of automatic packaging structure recognition based on single RGB images. Our contributions presented in this chapter include

- Acquisition of an extensive labeled dataset of 2D images for the use case of packaging structure recognition.
- Design and implementation of an image processing pipeline for automatized packaging structure recognition, based on state-of-the-art image analysis algorithms.
- Evaluation of the proposed method on our own data.

4.2 Related Work

This section gives a brief overview of publications and techniques relevant to the proposed algorithm. Most of these have already been elaborated in Chapter 3, but are still mentioned here to summarize the fundamentals of our work. Our work makes use of recent advances in the image processing problems of object detection and instance segmentation. We employ Mask R-CNN, a state-of-the-art neural network for instance segmentation, by He et al. (2017) for the segmentation of logistics components. As feature extraction layers within the Mask R-CNN segmentation network, we use Inception-v2 (Ioffe and Szegedy 2015). An overview of deep-learning-based instance segmentation, as well as further introductions to artificial neural networks, is given by Minaee et al. (2022). As the amount of data required to train an image instance segmentation model from scratch is tremendous, a remedy often applied is transfer learning. This term refers to the idea of pre-training a prediction model on a large dataset of a general task, before fine-tuning the previously learned weights on rather few use-case-specific data. Not in itself a particular deep learning or image processing technique, transfer learning is used in various contexts (Pan and Yang 2009). The method has been applied to image object detection or instance segmentation tasks and use cases (Azizpour et al. 2015) from various domains, such as medical imaging (Shin et al. 2016), airport security (Akçay et al. 2016), and environmental engineering (Gao and Mosalam 2018), to name only a few examples. For most image processing applications, pre-training is performed either on the image classification dataset Image-Net (Russakovsky et al. (2015), Huh et al. (2016)) or on the common object in context (COCO) object detection challenge's dataset for object detection tasks (Lin et al. 2014). The latter is also the case in our work.

4.3 Multi-Step Image Processing Pipeline

The proposed algorithm for extracting the packaging structure of a transport unit from a single image is described in this section. The algorithm consists of three essential steps:

1. Transport Unit Detection: Inter-unit Segmentation
2. Packaging Unit Detection: Intra-unit Segmentation
3. Result Consolidation and Refinement

First of all, an inter-unit segmentation is performed to detect relevant transport units within the input image. A convolutional neural network (CNN) is used to find and segment all transport units which are completely visible in the image. In the next step, intra-unit segmentation is applied to each cropped transport unit image. Here, two CNNs are used to find the unit's pallet and to detect the two distinct sides as well as all packaging units. Computer vision methods are used to consolidate and refine the results. The whole process is illustrated by means of a single test image in Fig. 4.1. The original input image is depicted in part (a). Like all our images, it was acquired in a logistics environment in the automotive industry, using a conventional smartphone camera. This is in accordance with our objective to provide a readily accessible algorithm, which operates on a simple RGB image. Other image sources, like fixated industrial cameras, could be used likewise, as long as they provide images according to our prerequisites (see Section 2.2.4). All our images

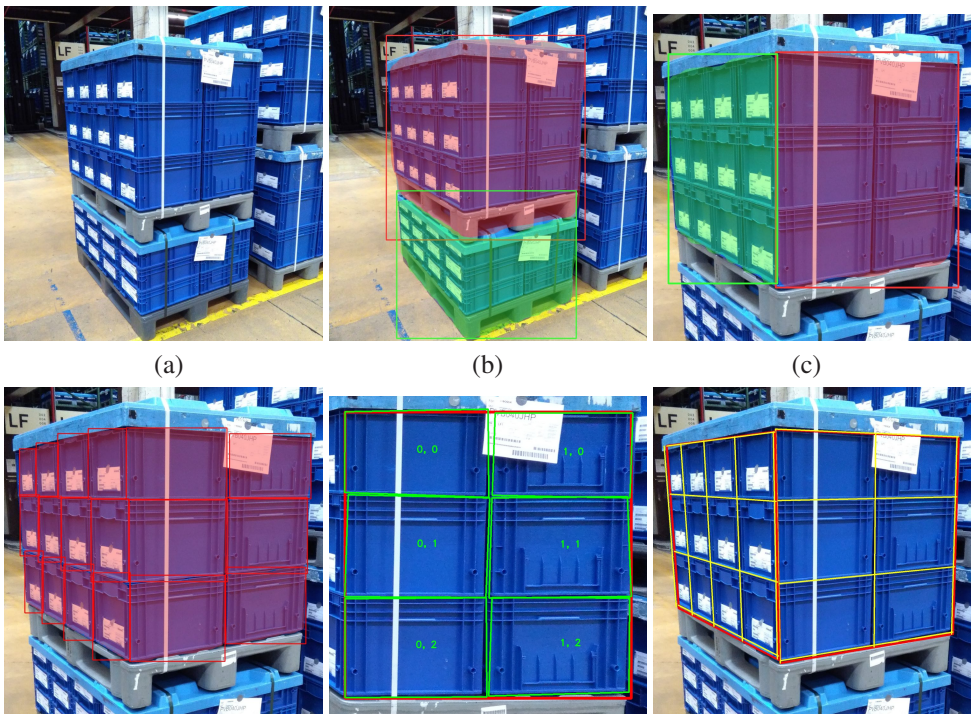


Figure 4.1: Illustration of the packaging structure recognition process. (a) Input image. (b) Inter-Unit Segmentation. (c), (d), (e) Intra-Unit Segmentation. Transport unit sides and packaging unit faces are found. Subsequently, the number and arrangement of packages for each transport unit side are determined. (f) Information Consolidation.

are downsampled in such a way, that the larger image dimension measures 1,000 pixels. The aspect ratio is preserved in this resizing operation. Note that additional resizing is generally performed before an image is passed into a neural network. For more details on our input images and dataset, please refer to Section 4.4. The input image (a) is fed into the first step of inter-unit segmentation, which detects and segments fully visible transport units (b). The output of this step are image crops showing exactly one complete transport unit. These image crops are fed into the succeeding step of intra-unit segmentation. In this step, transport unit sides (c) and package unit faces (d) are segmented. The last step of results consolidation processes these detections and determines the arrangement of package units for each transport unit side (e). This information is combined to determine the overall packaging structure (f). In the following subsections, each of these three essential steps is discussed in more detail.

4.3.1 Inter-Unit Segmentation

Given an adequate input image, the first step in our processing pipeline is the detection and segmentation of all relevant transport units within the image. Precisely speaking, we want to extract how

many transport units are fully contained in the image and which image regions and pixels belong to which of those transport units. To solve this task, a deep learning instance segmentation model is used. Namely, a Mask R-CNN (He et al. 2017) with an Inception-v2 (Ioffe and Szegedy 2015) backbone architecture is employed. The model only predicts objects of a single category: full transport unit. Apart from bounding box predictions for relevant objects, Mask R-CNN prediction models also output so-called confidence values within the interval $[0, 1]$ for each prediction and each object category. This value indicates how confident the model is, that an object of the corresponding object category is present within the bounding box.

Simple checks and filterings are performed as post-processing steps on the instance segmentation model's predictions: Apart from confidence thresholding, the detections are checked for a minimum size within the image, and detections, which have high bounding box overlaps with other higher-confidence detections, are dropped. In all our experiments, a confidence threshold of 0.5 was applied, i.e. only bounding box detections with a confidence score of 0.5 or higher are considered. This value was chosen based on a simple grid search experiment (using a sampling distance of 0.05), in which we determined the threshold value yielding the best results on our evaluation data. Further, the minimum bounding box size for transport units is set to 10% of the total image area, as we assume relevant transport units to cover a significant ratio of the image. The maximum acceptable bounding box IoU for two transport unit detections is set to 0.4. Detections that are not compliant with these thresholds are discarded. These parameters were chosen empirically based on simple comparisons and evaluations during development.

Fig. 4.1 (b) shows exemplary results for the transport unit segmentation. For each detected transport unit, a corresponding image crop is created. The crop region corresponds to the detection's bounding box, increased by a margin of 50 pixels in all four directions. This margin is employed to ensure the whole transport unit is depicted within the crop and all edges and corners are clearly visible. The image crops are used as input for the subsequent steps (compare 4.1 (c) which depicts the next step's results on the described image crop).

4.3.2 Intra-Unit Segmentation

For each image crop output by the inter-unit segmentation, the succeeding step of intra-unit segmentation is performed independently. Intra-unit segmentation aims to further analyze and segment the transport unit image. The objective is to identify the following regions:

- Base pallet
- Exactly two transport unit sides
- Package unit faces (not complete units)

Apart from the precise localization of the above components, the type of base pallet and packages are also determined. One very important aspect of this segmentation process is the identification and differentiation of the two visible, orthogonal transport unit sides. Only if both sides are segmented correctly and the detected package unit faces are assigned to the correct transport unit sides, the total package unit number can be calculated accurately. Note that package unit faces are detected

rather than complete package units. This makes a difference only for the corner stack of packages connecting the two visible transport unit sides. The reason for this decision is that we want to be able to process each transport unit side independently and to assign each package unit detection to one of the two transport unit sides unambiguously.

Analogously to the previous step, we use an instance segmentation deep-learning model to segment the desired regions within the image. As image crops are input into this model, one can assume that each image contains one base pallet and exactly two completely visible transport unit sides. Opposed to that, the number of package unit faces per transport unit side may vary significantly.

Note that transport unit side faces, and package unit faces, are of rectangular shape in the real world. Thus, their exact regions within the image can be described by four points within the image. This is under the assumption that the transformation underlying the imaging process is the projective transform. Consequently, the applied instance segmentation model's outputs, i.e. arbitrary pixel masks, are of unnecessarily high complexity for side region localization. In this first algorithm design, the model was chosen nonetheless as the vast majority of image detection methods tackle either bounding-box-based or if more accurate region information is required, segmentation-mask-based object representations. In Chapter 6, more specialized detection models aiming to find transport unit components based on a four-vertices-based representation are developed and analyzed.

Subsequent to the step of deep-learning-based image segmentation, simple consistency checks are performed. First of all, a confidence threshold of 0.5 is applied to transport unit sides and package unit detections. (Again, a simple grid search was performed to determine a suitable confidence threshold.) Afterward, all detection regions are checked for overlaps with the previously extracted transport unit region: Only detections with a bounding box of which at least 60% lies within the transport unit's bounding box are kept.

Additional post-processing of transport unit side and package unit regions is performed: All masks are cut according to the inter-unit segmentation's detection mask found for the considered transport unit. This means, any pixels detected in this step, exceeding the transport unit mask, are cut off.

For each transport unit, it is ensured that exactly one base pallet and two transport unit sides are found. In case of additional detections, only those with the highest confidence values are kept. If fewer components than required are detected, the algorithm aborts without returning any result. Transport unit side segmentation and packaging unit segmentation are illustrated in Fig. 4.1 (c) and (d), respectively.

4.3.3 Information Consolidation

The goal of this final step is to use the previously extracted segmentation information to determine the transport unit's packaging structure, i.e. to calculate the total number of packages. The algorithm works regardless of the type of packaging units present. The following steps are performed:

1. Assign each package unit face to one transport unit side
2. "Tetragonize" transport unit sides
3. Rectify both transport unit sides

4. Calculate package number

Each of the above steps is explained hereinafter.

1) Assign each package unit to one transport unit side: For each packaging unit, the intersection of its mask with both transport unit sides masks is computed. If at least one of these intersections is not empty, it is assigned to the side with the larger absolute mask intersection size. Packaging unit detections not intersecting any of the transport unit sides are discarded.

2) 'Tetragonize' transport unit side detections: To be able to calculate each transport unit side's package number in the succeeding step, the average packaging unit size has to be computed. Due to perspective distortions, transport unit sizes of identical packages vary vastly within the image depending on their position relative to the camera. Packaging unit and transport unit side regions are rectified to overcome this issue. To perform the rectification, the mask of each transport unit side is approximated by a tetragon shape. This is reasonable for the given case of cube-shaped package units and assemblies thereof, as explained above. Subsequently, the resulting tetragonal-shaped image patches are transformed to resemble rectangular regions using a suitable affine geometric transform.

To approximate the polygon describing the side's mask, an optimization problem minimizing the region difference for the detected transport unit side mask and the shape described by four corner points is considered:

$$\min_{p_1,p_2,p_3,p_4 \in P} \sum_{(i,j)\in P} \left| s_{(i,j)} - t(p_1, p_2, p_3, p_4, (i,j)) \right| \tag{4.1}$$

where $P = \{0, ..., m\} \times \{0, ..., n\}$ is the input image's pixel space, and $p_i = (x_i, y_i), i = 1, 2, 3, 4$ are four points in this pixel space, forming a tetragon (p_1, p_2, p_3, p_4). $s \in \{0, 1\}^{m \times n}$ is the binary mask describing the transport unit side as output by the segmentation model. $t : P^5 \rightarrow \{0, 1\}$ with $t(p_1, p_2, p_3, p_4, (i, j)) = 1$ if pixel (i, j) lies within the tetragon described by the corner points (p_1, p_2, p_3, p_4).

By connecting the tetragon corner points in order $p_1p_2p_3p_4p_1$, an area within the image is described. Arbitrary tetragonal shapes inside the image plane P can be described by four such pixel coordinates. On the other hand, note that the shape described in such a way is not necessarily a tetragonal one. Thus, the order in which the four corner points are connected is not arbitrary. For our work, we assume point orderings p_1, p_2, p_3, p_4 with:

$$x_1 \leq x_2 \tag{4.2}$$
$$x_3 \leq x_4 \tag{4.3}$$
$$y_1 \leq y_3 \tag{4.4}$$
$$y_2 \leq y_4 \tag{4.5}$$

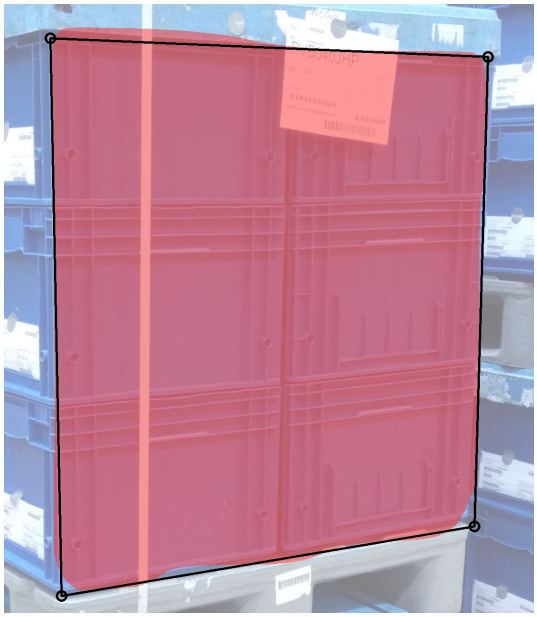


Figure 4.2: Illustration of the transport unit side mask simplification performed as sub-step of the Information Consolidation process. The detected transport unit side region is indicated in red. The four black dots indicate the four corners output by the post-processing optimization algorithm.

If these equations hold, it is ensured that the shape obtained by connecting $p_1p_2p_3p_4p_1$ is a single tetragon, and not two triangles in an hourglass-like arrangement.

In our implementation, the optimization problem in Equation 4.1 is solved numerically. OpenCV's implementation of the Douglas-Peucker algorithm (Douglas and Peucker 1973) is used on the polygon described by mask m to find suitable starting points. For our example image, this mask post-processing procedure and the results are illustrated in Fig. 4.2.

3) Rectify both transport unit sides: Using the approximated tetragon, i.e. the four transport unit side corner points, the side and corresponding package unit detections associated with this transport unit side are remapped in such a way, that the transport unit side is described by a rectangle of size $s_v \times s_h$. This is illustrated in Fig. 4.1 (e). In our experiment, we used $s_v = s_h = 1000$ for visualization purposes. (As the transform can be computed with sub-pixel accuracy, the choice of s_v and s_h is in general not relevant for result accuracy.)

4) Calculate package number: For each detected packaging unit i, the pixel mask is rectified alongside the transport unit side it is assigned to. A bounding box around the rectified pixel mask is determined, and this bounding box's size is used to approximate the package unit's extent. The height and width of packaging unit i is denoted by ps_h^i and ps_v^i, respectively. Now, the average

packaging unit height $\tilde{ps}_h$ and width $\tilde{ps}_v$ of all packages of one transport unit side, relative to the side's size, can be calculated:

$$\tilde{ps}_h = \frac{1}{|I|} \sum_{i \in I} ps_h^i \tag{4.6}$$

$$\tilde{ps}_v = \frac{1}{|I|} \sum_{i \in I} ps_v^i \tag{4.7}$$

Hereby, $I \subset \mathbb{N}$ denotes the set of packages assigned to the transport unit side.

The transport unit side's numbers of horizontally and vertically stacked transport units, n_h and n_v, are computed as

$$n_h = \lfloor \frac{s_h}{\tilde{ps}_h} + 0.5 + \delta_1 \rfloor \tag{4.8}$$

$$n_v = \lfloor \frac{s_v}{\tilde{ps}_{vh}} + 0.5 + \delta_2 \rfloor \tag{4.9}$$

Hereby, the additional summands $\delta_1, \delta_2 > 0$ account for the empirically discovered tendency towards over-estimation of package unit sizes relative to transport unit side sizes. This tendency could be due to the implementation of bounding boxes around the detected transport unit masks. Such bounding boxes easily exceed the actual units' extents in case of slight inaccuracies in the pixel masks. In our experiments, the values were set to $\delta_1 = 0.0$ and $\delta_2 = 0.1$. In the computation of the vertical package number, the additional summand δ_2 is larger than δ_1, which can be explained as follows: The packaging units in a transport unit's top row are often partly occluded by pallet covers (for example, see Fig. 4.1 (c)). Thus, these packaging units are not completely visible and their regions detected by the segmentation model are smaller than those of the other rows of packaging units. The larger choice of δ_2 helps to account for these size underestimations.

Once the horizontal and vertical package numbers are calculated for both transport unit sides, the overall package number can be determined. If the vertical package numbers for the two sides do not coincide, the algorithm stops without returning any package number result. Otherwise, the package number is calculated as

$$n = n_h^{(l)} \cdot n_h^{(r)} \cdot n_v \tag{4.10}$$

where $n_h^{(l)}$ and $n_h^{(r)}$ are the horizontal package number for the left and right transport unit sides within the image and $n_v = n_v^{(l)} = n_v^{(r)}$ is the vertical package number. Fig. 4.1 (f) exemplary shows a comprehensive illustration of the packaging structure recognition results.

This completes the description of our image analysis pipeline for packaging structure recognition. In summary, we explained the complete process of extracting one or multiple transport units' packaging structure, starting with a single RGB image. Before we move on to the training and evaluation of this pipeline, we introduce our custom dataset in the following section.

4.4 Dataset

In this section, we will describe our custom dataset for the use case of packaging structure recognition. The dataset is used both for the development of our algorithm tackling packaging structure recognition and for the evaluation of the latter. Particularly, we employ these data in the training of the learning-based methods utilized in our approach. To enable proper evaluation of the resulting models and the algorithm, part of the data set is not used in training or tuning the algorithm but is held back for final evaluations.

4.4.1 Image Acquisition Details

For the creation of our dataset, we acquired and annotated 1.000 images of one or multiple transport units in logistics environments. All images were taken in the incoming goods department of a German component supplier in the automotive sector. As is common for such logistics surroundings, transport units are spread throughout the entire location and acquiring pictures of isolated units is virtually impossible. Consequently, the vast majority of images depict additional transport units in their backgrounds. Conventional smartphones with integrated cameras were used for image acquisition (namely, the following four models were used: Samsung Galaxy S5, Samsung Galaxy S8, Huawei Y6, Huawei 701L). Depending on the device used, image aspect ratios vary slightly. All images were resized in such a way, that the larger image dimension measures exactly 1,000 pixels. Images were acquired in portrait mode, i.e. the larger image dimension is the vertical one in all cases. All images were taken indoors in well-illuminated environments.

4.4.2 Packaging Components

The packaging components contained within the dataset are restricted to a few predefined types to ensure the feasibility of the task of automatic packaging structure recognition (see Section 2.2.4). The relevant kinds of both packaging units and base pallets to be recognized are described in this section.

Transport and Packaging Units: Compliant with our prerequisites described previously (see Section 2.2.4), each transport unit may only contain packaging units of a single kind and of equal sizes. The data contains two different kinds of packaging units: KLT ("Kleinladungsträger") and tray units. The sizes of the packages of each type vary, but single transport units include only package units of the same size. Further, we assume that all transport units are packed uniformly, i.e. the packages are arranged regularly to cover the whole base pallet and each row and column of packages contains the same number of identical packages. Fig. 4.3 shows several example images for both package types.

Each image shows transport units of one packaging type exclusively, i.e. there are no images showing both packaging types. The number of images with KLT packaging types is significantly higher than for the other packaging type, as is summarized in Table 4.1.

Figure 4.3: Examples from our dataset of 1,000 images. Top row: KLT package unit type. Bottom row: Tray package unit type.

Base Pallets: Each transport unit pictured in our dataset is built upon a standardized base pallet. Though the base pallet materials and colors vary, the pallet size is always equal to 120 cm in length and 80 cm in width. Moreover, the two visually distinguishable classes of base pallets present are wooden Euro pallets and plastic pallets.

4.4.3 Annotations

Each image contains up to three fully visible and annotated transport units, which are expected to be recognized. Additionally, arbitrary other transport units may be only partly visible within an

image (these are not annotated). Data annotations consist of the following hierarchically organized information:

1. Transport unit regions
2. Transport unit side regions
3. Packaging unit regions and class
4. Base pallet regions and class

Each image contains one or more transport units. Each transport unit contains exactly two transport unit sides and one base pallet. Each transport unit side contains an arbitrary non-zero number of package unit faces. An example for image annotations is given in Fig. 4.4.

(a) (b) (c) (d)

Figure 4.4: Data annotation examples on a single image from our dataset. (a) The image and its two transport unit annotations (highlighted in red and green, respectively). (b) The two transport unit side annotations for the top transport unit are highlighted (red: left transport unit side, green: right transport unit side). (c) The packages assigned to (b)'s left transport unit side annotation are highlighted in red. (d) The base pallet of the top transport unit is highlighted in red.

4.4.4 Dataset Splits and Statistics

Of the 1000 images in our dataset, 150 images containing 163 transport units are held back for evaluation of the packaging structure pipeline. This corresponds to 15% of all images. The selection of evaluation images was performed manually with the objective to assure that images of identical transport units were not split up between evaluation and training datasets. More precisely, if several images of one transport unit were available, either all of those or none of those, were assigned to the evaluation data subset. Additionally, evaluation images were selected in such a way, that the relative numbers of KLT and tray units in both dataset splits are similar. This is verified by Table 4.1.

Table 4.1: Distribution of transport unit number and unit types within our data and its split sets.

	Training	Validation	Evaluation	Overall
Transport Unit Number				
1	636	136	137	909
2	59	14	13	86
3	5	0	0	5
Package Unit Type				
KLT	537	102	116	755
Tray	163	48	34	245

The remaining 850 images are used for training and validation of the segmentation models. To form the validation subset, 150 of those images (15% of all images) were selected randomly.

Table 4.1 shows some basic statistics for the whole dataset and the three described subsets.

4.5 Evaluation

4.5.1 Segmentation Model Training and Evaluation

Model Training

All three segmentation models in use (transport unit segmentation, side and package segmentation, base pallet segmentation) are standard Mask R-CNN (He et al. 2017) models using Inception-v2 (Ioffe and Szegedy 2015) feature extractors, as implemented in tensorflow's object detection API (Abadi et al. 2016). Once again, note that our focus in answering the research question (RQ1) was to design a viable image processing pipeline for the use-case of packaging structure recognition, not to find the single best deep learning model for those sub-tasks solving image analysis tasks. Thus, no excessive model search and hyperparameter tuning were performed in training our CNN-based models. Instead, a suitable CNN implementation with moderate computational complexity was selected and tuning was terminated once further improvements could not be easily accomplished using our data.

As our dataset of 1.000 images is rather small, heterogeneous transfer learning is used. Such approaches have proven to be a powerful solution in case of limited training data (Weiss et al. 2016). The models were initially trained on the COCO object detection challenge's dataset (Lin et al. 2014) of approximately 100 times the size of our data (123,287 annotated images). Subsequently, the models were fine-tuned on our 700 training images for 200.000 training steps using gradient descent with momentum (Qian 1999) and a batch size of one. Input image resolution was set to 512 pixels for the larger image dimension, transforming the image in such a way that the original aspect ratio was preserved. The Mask R-CNN's output mask resolution is set to 25 x 25 pixels. The batch size

Table 4.2: Training Configuration and Hyperparameters

Configuration & Hyperparameters	**Value**
Image input resolution	512 in larger dimension
Feature Extractor	Faster R-CNN Invecption-v2
Batch Size	1
Optimizer	Momentum Optimizer
Momentum Value	0.9
Training Steps	200,000
Learning Rate	Step 1 - 120k: $2 \cdot 10^{-4}$
	Step 120k - 160k: $2 \cdot 10^{-5}$
	Step 160k - 200k: $2 \cdot 10^{-6}$
Data Augmentation Options	
Random horizontal flip	Probability: 0.5
Random crop	Minimum image area: 0.95
	Probability: 0.9
Random pad	Maximum image width: 612
	Maximum image height: 612
Random brightness adjustment	Maximum deviation: 0.3
Random RGB to gray	Probability: 0.2

was chosen due to technical restrictions: The described image resizing approach results in differently sized images, which can not be processed within a single batch by the CNN implementation at hand. The number of training steps was chosen, as appeared sufficiently large to ensure convergence in our experiments, while not prolonging the training process unnecessarily. Techniques like early stopping were not implemented due to above stated reasons, even though such approaches might yield minor performance improvements in some cases.

Several image augmentation methods were used in neural network training, namely random horizontal flip, crop and pad, conversion to gray values, and brightness adjustments. We experimented with additional augmentation methods, for instance, random adjustments of saturation and contrast, but found that they did not lead to improvements in accuracy. We performed a sequence of training experiments to find a suitable image augmentation configuration for our application. More precisely, we monitored the resulting mAP of our transport unit side and package unit segmentation model when activating different data augmentation options during training, and we picked those options yielding the best results. This data augmentation configuration was applied in all our model trainings.

Table 4.2 summarizes our model configuration and hyperparameter choices.

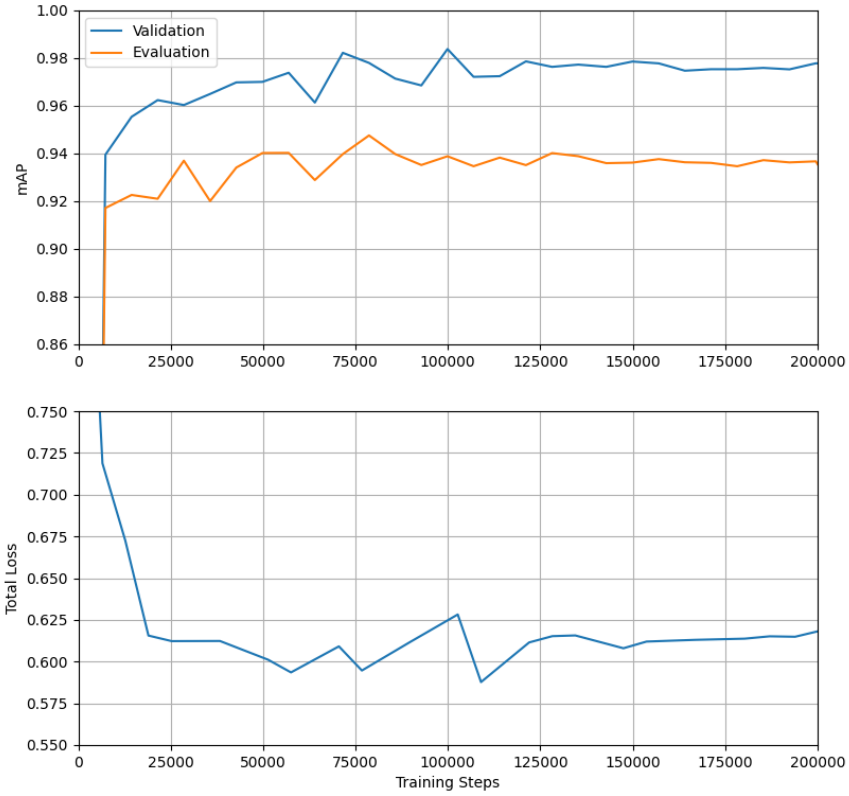


Figure 4.5: Training progress of the CNN model for transport unit segmentation.

Model Evaluation

For the trained CNN models, the model's performance under the COCO Object Detection challenge's metric is measured; i.e. the Average Precision (AP), and mean Average Precision (mAP) (see Section 3.3.2), averaged for ten different intersection over union (IoU) thresholds of 0.5 to 0.95, is computed.

The training progress of both CNN models is illustrated in Figures 4.5 and 4.6. In both cases, the top plot illustrates the mAP values on our validation data measured throughout the process of 200,000 training steps. The bottom plot illustrates the overall loss values. Note the differently scaled y-axes in both plots. AP values generally range from 0.0 to 1.0, in case of perfect detections; loss values are not easily interpretable. Still, one can observe the initially steep decrease in loss values, which subsequently reach saturation within the 200,000 training steps. Analogously, validation AP values initially increase significantly before plateauing after approximately 120,000 to 150,000

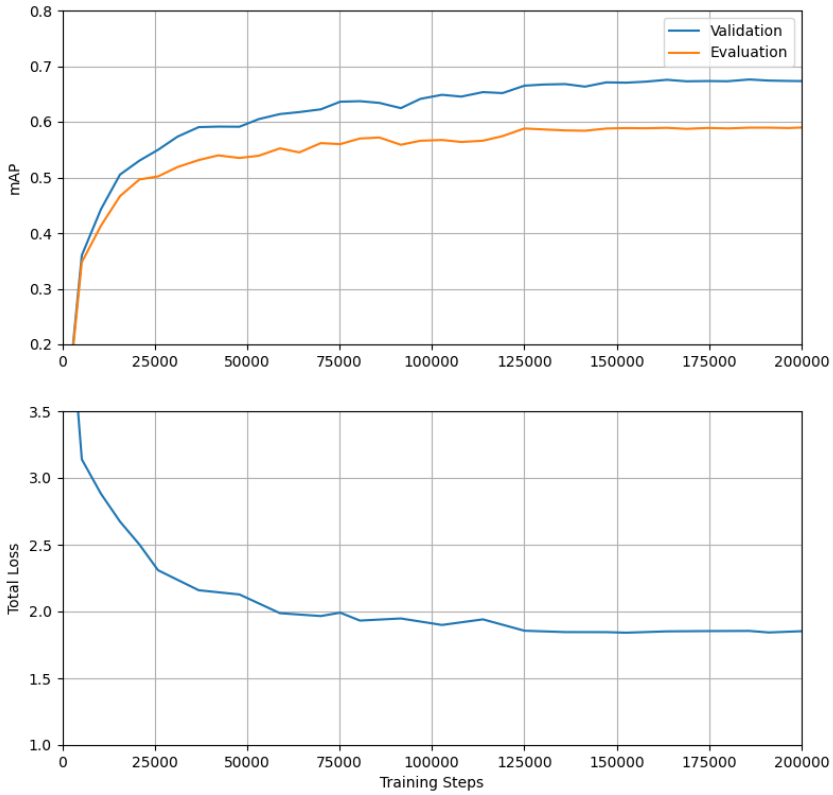


Figure 4.6: Training progress of the CNN model for transport unit side and package segmentation.

training steps. This indicates the saturation of the training process, with respect to the chosen training configuration and hyperparameters.

Considering Fig. 4.5, one observes that the model reaches its best performance on validation data after approximately 100,000 steps, and, subsequently, the model's performance declines slightly. In this case, an early stopping mechanism might yield a model with slightly better evaluation results by stopping training prematurely, when the peak in performance is reached. Still, this potentially better performance is neglectable in this case as the difference in performance measures less than 0.005 mAP points. Also, note that mAP values on validation and evaluation data do not always develop analogously in the later phases of the training.

Models were evaluated on the 150 dedicated evaluation images and on the randomly selected set of validation images of the same size. The results are listed in Table 4.3: For all three segmentation

Table 4.3: Segmentation Models' AP and mAP values on validation and evaluation data

Model/Class	**Validation images**	**Evaluation images**
Transport Units (AP)	0.978	0.935
Sides and Packages (mAP)	0.678	0.601
Sides (AP)	0.901	0.877
KLT units (AP)	0.635	0.573
Tray units (AP)	0.499	0.353
Base pallets (AP)	0.766	0.697

models the mAP is given. Additionally, the average precision for the individual classes (transport units, transport unit sides, KLT packaging units, tray packaging units and base pallets) included in the side and package segmentation model is listed.

Notable are all models' slightly superior performance values on validation data when compared to performance on evaluation data. This can be explained by the dataset split procedure: As explained previously, the set of evaluation images is hand-picked and, importantly, either all or none of the images of the same physical transport unit are contained. On the contrary, the split between training and validation data was performed randomly. Consequently, some validation images may show transport units, of which images were also contained in the training data set. Thus, the model can be expected to perform overly well on such validation images. Different images of the same transport unit may be very similar also in background elements, lighting conditions and perspective.

Statistical Significance

We perform additional training experiments to investigate the statistical significance, and the reproducibility, of our previously reported training results. These experiments are performed on the 850 images of our dataset, which were designated for training and validation. The 150 evaluation images were held back during training and only used for final evaluations.

We perform a Monte Carlo cross-validation (MCCV). Burman (1989) and Ramesan and Mathew (2015, Chapter 3.6) discuss the concept of MCCV, which is sometimes also referred to as random sub-sampling or repeated learning-testing methods. Similar approaches, like k-fold cross-validation, could be employed alternatively. We opt for MCCV due to its flexibility regarding the choices of data split ratios and training repetitions. In each MCCV iteration, a subset of a fixed number of samples is selected from the training data at random, and held back as validation data set. A model is trained on the remaining training data. This process is repeated several times to produce results regarding the statistical significance of training results, and avoid overfitting on a specific training-validation dataset split.

We perform an MCCV with 30 iterations for our two instance segmentation models, i.e. for the model for transport unit detection, as well as the model for transport unit side and package detection, In each iteration, we select 150 instances of our 850 training and validation images at random. We train both our CNN models on the 700 remaining images. Subsequently, we evaluate the trained models

on the randomly selected 150 images which were not used in training, as well as on our dedicated evaluation dataset.

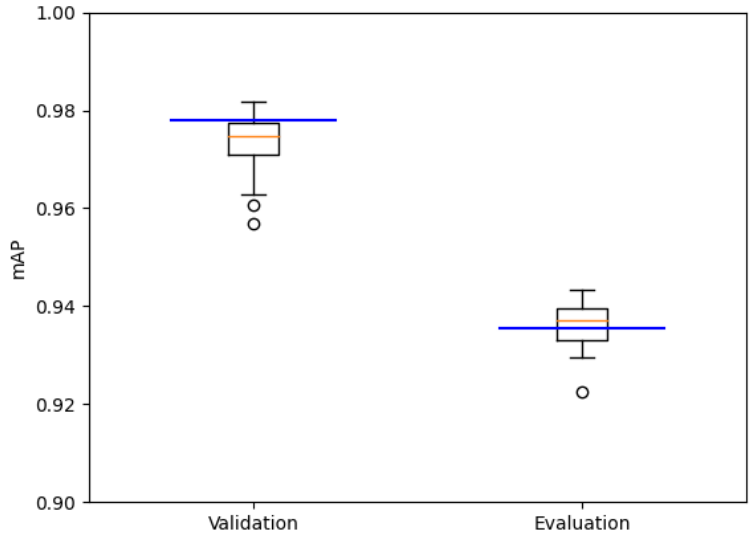


Figure 4.7: Box plot showing the evaluation statistics [mAP] for our transport unit segmentation model on validation and evaluation data.

Fig. 4.7 and Fig. 4.8 illustrate the statistics of the models' evaluation results on validation and evaluation data, utilizing box plots. As common, the box covers the range of the results' 0.25-quantile to 0.75-quantile. The orange line within the box corresponds to the median value. The whiskers cover all the data points which lie within a maximum distance of 1.5 times the interquartile range from the box. All outlier values, i.e. such values which have a greater distance from the box, are illustrated as circles. The blue horizontal lines indicate the mAP values of the models used in our PSR pipeline, as evaluated in the previous subsection. Note the different scaling and ranges of the y-axes in the two plots. Table 4.4 summarizes the same evaluation results by stating the mean and the standard deviation of the reported mAP values, for both models and datasets, respectively. Note that each statistical evaluation is based on 31 values: the 30 MCCV runs, and our initial training-validation split, as evaluated in the previous subsection.

In the case of the transport unit segmentation models, we observe very small values for the standard deviation of the reported mAP values. All reported mAP values on the evaluation data lie within the range of 0.922 and 0.944, with a mean value of 0.936. The model we use in our PSR pipeline reported an mAP of 0.935. The variance of the mAP values for the same models on their respective validation data is equivalently small. For the transport unit side and package segmentation models, the mean mAP value for the 30 MCCV models is 0.596. The mAP value for the model trained on our initial data split reads 0.601. The precisions' standard deviation on evaluation data is as low as for the transport unit detection model, whereas that of the validation data is somewhat higher.

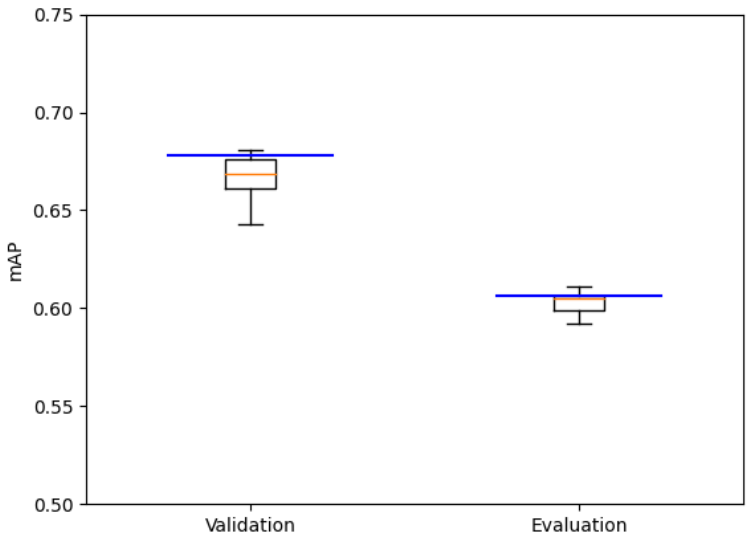


Figure 4.8: Box plot showing the evaluation statistics [mAP] for our transport unit side and package segmentation model on validation and evaluation data.

This can be explained by the models' poorer performance on the task of package unit detection, and by the inconsistent number of instances per package type in different randomly selected validation sets. As only approximately one-quarter of our training and validation images (211 out of 850) contain tray transport units, the total amount of such instances within a randomly selected validation dataset of 150 images varies broadly, and may be particularly small. Especially if the total amount of comparably error-prone tray unit examples in a validation dataset is low, a higher deviation of the resulting AP evaluation value from the mean and median values is more likely.

None of the mAP values, that we evaluated and reported in the previous chapter are ranked as outliers in our box plots. The transport unit segmentation model performs somewhat below average on evaluation data (see Fig. 4.7), whereas the transport unit side and package segmentation model performs above average on evaluation data (see Fig. 4.8). Overall, the variance in our cross-validation training results is very low, indicating the reproducibility of our results. More precisely, it shows that the random split of training and validation data does not significantly influence the model performance, and that the hyperparameters are not overfitted to our present training-validation split.

4.5.2 Pipeline Evaluation

The packaging structure recognition process is evaluated on our dedicated evaluation images, which were not used in model training or validation. In this section, isolated parts of the recognition pipeline, as well as the pipeline as a whole, are evaluated using adequate evaluation metrics.

Table 4.4: Mean and Standard Deviation of our evaluation results in the cross-validation of the two segmentation models for transport units, and sides and packages.

Model/Class	**Validation images**			**Evaluation images**		
	Mean	Median	St. Dev.	Mean	Median	St. Dev.
Transport Units (AP)	0.973	0.975	0.006	0.936	0.937	0.005
Sides and Packages (mAP)	0.665	0.666	0.014	0.596	0.597	0.005
Sides (AP)	0.894	0.892	0.010	0.870	0.871	0.008
KLT Packages (AP)	0.645	0.646	0.015	0.564	0.565	0.006
Tray Packages (AP)	0.457	0.452	0.036	0.354	0.353	0.009

Inter-Unit Segmentation

In a dedicated evaluation, we examine the results of the inter-unit segmentation, the recognition pipeline's first step, by computing an accuracy value for the resulting transport unit extractions. Note that this step does not only include inference of the segmentation model, but also the post-processing steps explained in Section 4.3.

The computed accuracy value, ACC_{inter}, measures the number of correct detections, relative to the total number of annotated transport units and erroneous detections:

$$ACC_{inter} = \frac{|TP_{inter}|}{|TP_{inter}| + |FN_{inter}| + |FP_{inter}|} \tag{4.11}$$

Where TP_{inter}, FP_{inter}, and FN_{inter}, are the true positive, false positive, and false negative transport unit detections of inter-unit segmentation. In this context, true positives are all correct detections, which can be uniquely matched to a ground-truth annotation. False positives are transport unit detections, which can not be matched to any ground-truth annotation. False negatives are transport unit annotations for which no matching detection was found. To classify all detections and annotations as true positive, false positive or false negative, the matching between ground-truth annotations and detections was performed based on their pair-wise masks' IoU values. Each detection and ground-truth annotation can only be counted in one such matching and is ignored afterward.

When applying a quite low IoU threshold of 0.5 for a detection to be counted as true positive, all transport units are detected. As the IoU requirement is increased, the accuracy value decreases only slightly, as illustrated in Fig. 4.9 and Table 4.5. This is in accordance with the model's high Average Precision values found in the previous section.

The average IoU value for a true positive transport unit detection measures 0.948 for all transport units. The average IoU for transport units with KLT-type packages measures 0.952, and is slightly superior to that of transport units with tray-type packages, which measures 0.935. Additionally, the standard deviation of the detected transport units' IoUs is slightly higher for tray units, which explains the inferior accuracy in the case of tray units for lower IoU thresholds. Table 4.6 summarizes these statistics.

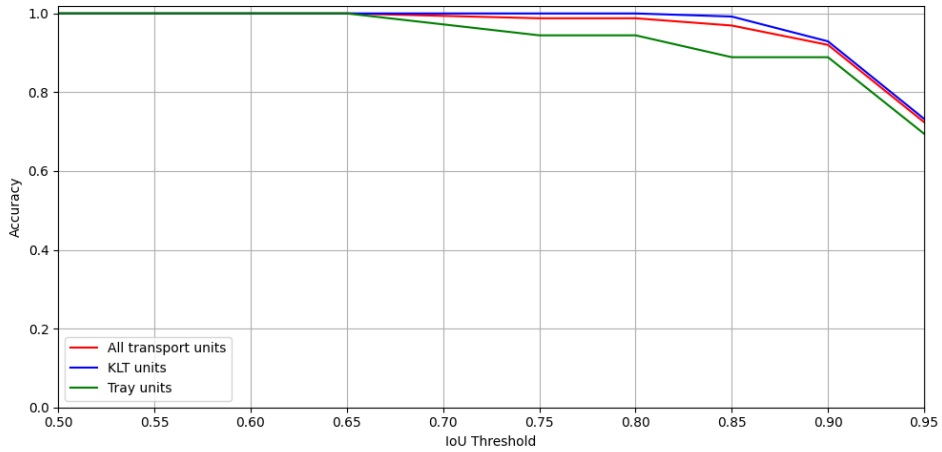


Figure 4.9: Accuracy of the step of inter-unit segmentation for different intersections over union requirements.

Table 4.5: Accuracy ACC_{inter} of the step of Inter-Unit Segmentation for different IoU value requirements.

	IoU threshold					
Image Set	0.5	0.6	0.7	0.8	0.9	0.95
Evaluation All	1.000	1.000	0.994	0.988	0.920	0.724
KLT Images	1.000	1.000	1.000	1.000	0.929	0.732
Tray Images	1.000	1.000	0.972	0.944	0.889	0.694

Recognition Pipeline

We evaluate the whole packaging structure recognition pipeline by examining how many of the recognized transport units were recognized correctly. For a transport unit to be recognized correctly, several conditions must be fulfilled:

- The transport unit was detected accurately (IoU of at least 0.5)
- The package unit type was detected correctly

Table 4.6: Statistics regarding IoU distribution of transport unit detections.

Image Set	Mean	Standard Deviation
Evaluation All	0.948	0.039
KLT Images	0.952	0.023
Tray Images	0.935	0.069

Table 4.7: Pipeline evaluation results

Image Set	Number of images	**Evaluation Error** e
Evaluation All	150	0.150
KLT Images	116	0.065
Tray Images	34	0.441

- The total number of package units was recognized correctly

A recognition error e_i for each evaluation image i is computed independently as:

$$e_i = 1 - \frac{|\mathrm{TP}_i|}{|\mathrm{TP}_i| + |\mathrm{FP}_i| + |\mathrm{FN}_i|} \tag{4.12}$$

where $|\mathrm{TP}_i|$, $|\mathrm{FP}_i|$ and $|\mathrm{FN}_i|$ are the sets of true positive, false positive and false negative results within the image. In this context, true positive means, an annotated transport unit was found and the packaging structure was recognized correctly. False positive can mean two things: Either an additional transport unit was found where there was no unit annotated, or the packaging structure of an annotated transport unit was not recognized correctly. As usual, false negatives are annotated transport units that were not recognized at all. The overall evaluation error e is computed as the mean error over all evaluation images:

$$e_I = \sum_{i \in I} \frac{e_i}{|I|} \tag{4.13}$$

where I is the set of evaluated images.

An overview of the evaluation results is given in Table 4.7. The evaluation dataset's overall evaluation error e computes to 0.15, indicating that approximately 85% of all transport units were recognized correctly. The algorithm's success rate is significantly higher for transport units with KLT packages compared to tray packages (93% vs. 56%). As is the case for the human eye, KLT package units seem to be easier to recognize and differentiate for the neural network. This can also be seen in the segmentation model's result presented previously in Table 4.3. Reasons for this fact may be the darker color and lower contrast of tray units compared to KLT units, which results in higher uniformity in the values of the pixels assignable to the tray transport unit. Additionally, the number of tray unit samples in our dataset is significantly lower than that of KLT units. This may result in the neural network emphasizing the accurate detection of KLT units over that of tray units.

Patched Recognition Pipelines

In this section, exactly the same evaluations as described in the previous section are performed on the complete recognition process, while replacing single steps of the pipeline. The goal of these evaluations is to weigh the consequences of errors in single components of the pipeline. The following modifications are explored:

- Insertion of ground-truth annotations of transport unit sides ($\mathrm{PATCH}_{\mathrm{sides}}$)

- Insertion of ground-truth annotations of packaging unit faces ($PATCH_{pack}$)
- Insertion of both transport unit side and packaging unit faces ($PATCH_{both}$)

The resulting error values on evaluation images of these modified recognition pipelines can be found in Table 4.8. Thereby, the same evaluation metric as previously introduced and applied in the pipeline evaluation (see Equation 4.13) is used.

Table 4.8: Evaluation Error e of original and patched recognition pipelines.

Image Set	Number of images	**Recognition Pipeline**			
		Original	$PATCH_{sides}$	$PATCH_{pack}$	$PATCH_{both}$
Evaluation All	150	0.150	0.170	0.050	0.000
KLT Images	116	0.065	0.099	0.052	0.000
Tray Images	34	0.441	0.412	0.044	0.000

As is expectable, when perfect transport unit side and package unit face detections are inserted ($PATCH_{both}$), the algorithm correctly recognizes all transport unit's packaging structures.

In the case of replacing packaging unit detections ($PATCH_{pack}$), the error measure for transport units of tray-type packages decreases significantly. The error value for KLT packages sees, at least, slight improvements. This helps to identify the package unit segmentation as a major source of error in our overall pipeline. At the same time, it shows that the recognition accuracy for tray units can resemble the accuracy for KLT units if reliable package unit segmentation can be achieved. Overall, in this scenario, the evaluation results in an accuracy value of 95%, and an even slightly higher value for tray packaging units.

In the scenario ($PATCH_{sides}$), interesting observations can be noted: The replacement of transport unit side detections by allegedly perfect annotations does not result in a reduced error measurement. On the one hand, this can be explained by the overall good performance of the transport unit side segmentation model, as evaluated previously. Further, note, that transport unit side segmentation and package unit face segmentation are performed simultaneously by one single neural network, using the same image features. Thus, it seems likely, that systematic errors can occur. For instance, if a transport unit side's corner is misplaced by the model, the model may be likely to also misplace the corresponding package face's corner accordingly. This can result in slightly erroneous detections canceling each other out at large. More precisely, if, for instance, both detected transport unit side masks and package face masks are too short in horizontal direction, this will still allow for the correct packaging structure to be recognized.

Manual Error Observations

Apart from the quantitative evaluations presented above, insights and model understanding can be gained through qualitative observations. To this end, some error cases are analyzed in this subsection. Altogether, 9 of 127 KLT transport units were not recognized correctly.

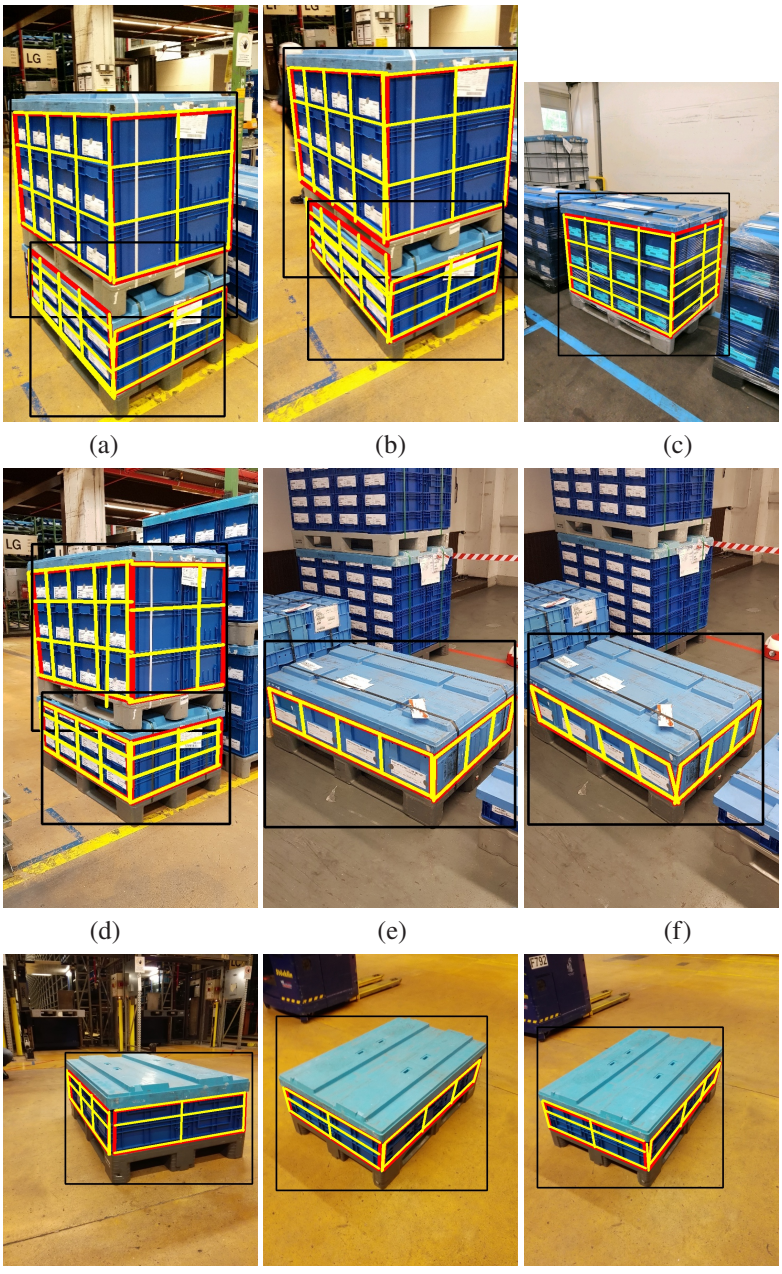


Figure 4.10: Error cases for unit type 'KLT' (exhaustive list). In (a) and (b) the lower unit is not recognized correctly. In (d), the upper unit is not recognized correctly.

All 9 error cases are illustrated in Fig. 4.10. In cases (a) and (b), the lower transport unit's left side was not recognized correctly (the detection does not span the whole transport unit side, but is shorter in the horizontal direction). As the number of packaging units in each direction is computed as the average packaging unit's size relative to the transport unit side's size, this results in an error in the calculation of the number of packaging units in horizontal, or vertical, direction, respectively. In the remaining seven cases, inaccurate package unit detections lead to errors in the results. In case (c), the detection of packaging units was hindered by the foil applied to the transport unit. The packaging strap applied to the transport unit in (e) and (f) resulted in a packaging unit being detected as two units, split by the packaging strap. In the last three cases, we argue, that the imaging perspective was far from ideal: Images (h) and (i) were taken from a significantly downward perspective (the camera was not on level with the transport unit, as is desirable). In case (g), one of the transport unit's sides is only hardly visible due to a near-frontal perspective of the transport unit. (Compare to imaging restrictions defined in Section 2.2.4.)

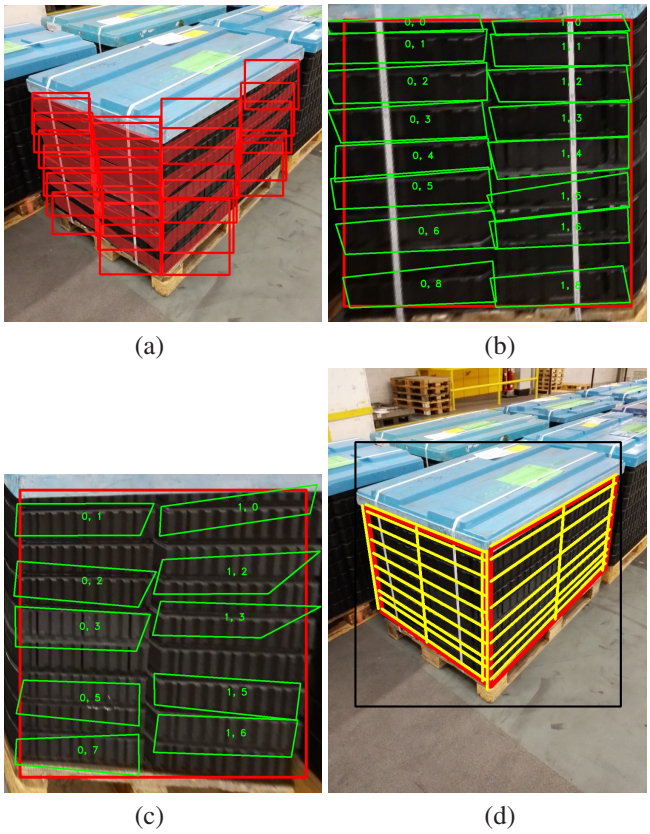


Figure 4.11: Example for erroneous packaging unit detection in case of tray packaging units. (a) Detected packaging units, (b), (c) rectified images of detected transport unit sides and assigned package detections, (d) overall results.

The error rate for images showing transport units composed of tray packages was significantly higher than for KLT packages: Errors occurred for 17 out of 36 transport units. In most of these cases, these errors were, once again, due to inaccuracies in packaging unit detection. In accordance with

previous evaluation results (compare section 4.5.1), tray packages are harder to recognize, not only for the human eye but also for the CNN detection model. Examples of such errors are illustrated in Fig. 4.11. As one can see, the package unit detections on the right transport unit side are incomplete and inaccurate. Moreover, present package unit detections are systematically too large in height, leading to an underestimation of the number of vertically stacked units, and, thus, inaccurate results.

In other cases, the packaging structure detection failed due to inaccuracies in the pipeline's first step of inter-unit detection. Corresponding examples are illustrated in Fig. 4.12. In both examples, transport units are placed next to additional, not fully visible transport units of very similar appearance (i.e. same package type and number of units). Thus, the transport unit detection does not correctly recognize the transport units' horizontal extents and assigns part of neighboring background units to a relevant unit. Note that these inaccuracies might, depending on the applied IoU threshold, not even be classified as errors in the step of inter-unit detection as the detections' IoU was nonetheless relatively high relative to the images' annotations. (The detected transport units have IoUs of 0.833 and 0.837 for examples in Fig. 4.12 (a), and IoU of 0.901 in for example in Fig. 4.12 (b).) Still, the inaccurate results of the transport unit detection allow for further inaccuracies in the succeeding steps of side and package unit detection, and in the computation of package unit numbers.

(a) (b)

Figure 4.12: Examples for erroneous packaging unit detection in case of tray packaging units, due to inaccurate detection of the whole transport unit.

Overall, our analysis shows that diverse reasons for erroneous results of our pipeline exist. On one hand, the involved instance segmentation models produce inaccurate results in some cases. This could be overcome by employing additional training data, improving the model and training setup, or, in some cases, construction and fine-tuning adequate post-processing steps. On the other hand, some errors could be resolved by more thorough enforcement of our image acquisition restrictions, i.e. enforcing desired imaging perspectives. The resolution of error cases for KLT packaging units is more easily accessible as is the case for tray units, which are visually more difficult to recognize and segment on 2D images.

4.6 Result Discussion and Assessment

Our experiments have shown that it is possible to construct an image processing pipeline, which is able to perform automated packaging structure recognition on 2D images with high success rates. The accuracies achieved in our experiments read 85% for our dataset containing two common packaging types, and 93% for the most-common packaging type of KLT units (for which a significantly higher amount of training data was available). These values may not be sufficient to completely avoid human efforts of packaging structure recognition, but can arguably be improved further. Further, common experiences in deep learning methodology teach that results can often be significantly improved by increasing the amount of training data available. Notably, our results show that the simultaneous recognition of multiple, visually distinct packaging types is possible, despite the comparably rather limited amount of training data. Accurate recognition results are possible, even in case of small occlusions due to transport labels, packaging straps or other applications to the transport unit. We achieved reliable results in a variant imaging setup, e.g. different imaging devices and backgrounds, and did not need to enforce strong limitations regarding lighting conditions or exact imaging perspective. Still, some errors prospectively could be avoided by increasing imaging requirements, such as the selection of favorable perspectives and distinguishable backgrounds (reduction of the amount of disturbing background transport units).

In the current state of our packaging structure recognition pipeline, the most frequent source of error is the step of package unit face segmentation. This step appears to be especially difficult for the tray package type. The latter is visually harder to distinguish, also for the human observer. Apart from that intuitive observation, this shows the shape, size, structure and color of package units to be crucial for their recognizability by CNN detection models.

Arguably, we have only considered a small fraction of package types and transport unit appearances relevant to the task of packaging structure recognition. A broader examination is hindered by the limited availability of relevant training and evaluation images and the high efforts required for the manual acquisition and annotation of such images. Synthetic training data generation can be a resolution to this common obstacle of data availability.

In this chapter, we attended to our first research question by designing a functional procedure to extract packaging structure information from 2D images. We also introduced a dataset for the novel use case and evaluated our method on our data. The next chapter will focus on the technical algorithms employed by our image processing pipeline, and thereby tackle the second research question.

5 Comparing Deep Learning and Computer Vision Approaches - A Case Study

In this chapter, we aim to challenge the necessity of employing learning-based approaches to obtain high-quality results for our task of packaging structure recognition. To that end, we investigate the applicability of non-learning-based, more traditional computer vision approaches to the problem at hand. Moreover, we first review literature discussing algorithmic choices for, and proposing solutions to, related image analysis tasks. As our proposed method for packaging structure recognition is a complex multi-step image processing pipeline, we explore the question of algorithmic choices based on the sub-step of transport unit side detection.

Parts of this chapter's experiments were already published in Dörr et al. (2020b). However, due to a refactoring of the dataset and minor adjustments to our implementation, evaluation values differ slightly. Interpretation and implication are not affected by these changes.

5.1 Research Question Elaboration

The research question (RQ2), which we tackle in this chapter was already introduced in Section 1.2, and reads as follows:

(RQ2) Which algorithms and techniques are well-suited for the implementation of a packaging structure recognition algorithm?

To answer this question, we investigate different algorithmic approaches toward the solution of the problem of packaging structure recognition. This question is very complex, as diverse algorithm choices and arbitrary complex algorithm compositions are possible. Thus, we cannot provide the necessary resources to prove one algorithm or solution to be ideal. We limit the scope of the question, and the considered tasks, in such a way, as to allow meaningful insights and a reliable assessment regarding suitable algorithm choices.

In the following, we describe the scope limitations, assumptions and criteria we consider in our elaboration of the research question.

First of all, we narrow the task for which the question is considered. Previously, we described our approach to packaging structure recognition to be a multi-step image processing pipeline, solving a series of related recognition tasks subsequently. In its original version, the research question considers all image recognition steps. We decide to focus on the task of transport unit side recognition in this chapter. This specific task was selected, as it appears to be most suitable for the application of

manually composed recognition algorithms, as the structure and placement of the objects (transport unit sides) to be recognized are well known.

Fundamentally, when solving image recognition task, two basic algorithmic choices are viable: Traditional computer vision or CNN-based approaches, i.e. deep learning. These are the natural options we limit our investigations to. Our original implementation uses state-of-the-art CNN methods for instance segmentation and custom post-processing operations (see Chapter 4). Thus, in this chapter, we construct an algorithm based on traditional computer vision techniques only. We compare the results and requirements of both approaches to answer the research question (RQ2).

In the implementation of our reference computer vision algorithm, we are limited regarding personnel and computing resources. Such limitations are realistic and common in the construction and implementation of industrial solutions. Theoretically, it is possible to construct computer vision algorithms performing on the same level as deep-learning-based approaches, as the fundamental techniques are often identical (CNNs use convolutional operations which are also common in traditional computer vision, compare Chapter 3). Still, in practice, the potentials of both approaches differ significantly, as it is not feasible to tune the high number of required convolutional weights and parameters manually, without the use of learning algorithms. Consequently, we aim to answer the question, of whether a competitive or at least promising computer-vision approach can be found with very limited implementation resources.

Another relevant consideration regarding the elaboration of our research question is the algorithm's ability to generalize. As we know for deep learning algorithms, such models can be trained to recognize relevant objects in highly variant appearances and environments (i.e. different perspectives, backgrounds, components), if suitable training data is available. The same is not necessarily true for hand-crafted computer vision approaches.

We summarize our contributions in this chapter as follows:

- We present a unique case study matching deep learning algorithms against traditional computer vision approaches.
- We evaluate the performance of two rivaling approaches and discuss their advantages and disadvantages.
- We conclude by finding the deep learning approach to be superior in the scope of our case study, thereby providing guidance on similar recognition tasks.

5.2 Related Work

This section gives an overview of similar object detection problems and their solution approaches using classic image processing techniques. Zou et al. (2023) give a comprehensive overview of the evolution of object detection in images, having analyzed more than 400 relevant publications. They note that the history of object detection can be split into two periods: one before and one after the rise of deep-learning-based object detectors, which was triggered by Krizhevsky et al. (2012) and related publications (compare Chapter 3, especially Section 3.2). In the earlier time period, object

detectors were constructed using handcrafted features, feature extraction methods like the scale-invariant feature transform (SIFT) (Lowe 1999), and also image transforms like Haar wavelets or the Hough transform. At the same time, most methods focused on a single use case, and one or a small number of object classes. For instance, two of the most influential works of the time, Viola and Jones (2001) and Dalal and Triggs (2005), focus on face recognition, and pedestrian detection, respectively. The highest accuracy ($mAP_{0.5}$) achieved by a model of this era before the rise of CNNs for object detection was 33.7% by Sadeghi and Forsyth (2014), as compared to 55.5% by the CNN-based model RCNN by Girshick et al. (2013) in the same year, and 89.3% by the 2022 record entry Ghiasi et al. (2021). Even though the accuracy and success of image recognition and object detection have seen an enormous increase in the later time period (see Zou et al. (2023) and Everingham et al. (2010) for details), this section is concerned with approaches not employing CNNs and deep learning.

More specifically, we consider related work aiming to detect objects of linear shapes, or based on linear edges. Line detection as a problem in itself has attracted some attention as a research field. Most approaches employ the Hough transform (Hough 1962). As early as the 1980s, image line detection studies relying on the Hough transform were published. Inigo et al. (1984) detect pathways and obstacles in images to enable autonomous vehicles to navigate more safely, and Skingley and Rye (1987) investigate the usage of the Hough transform to detect lines in radar images. Numerous improvements to the Hough transform, regarding both performance and result quality, were suggested, including the randomized Hough transform (RHT) by Xu et al. (1990), and the probabilistic Hough transform (PHT) by Kiryati et al. (1991). Aggarwal and Karl (2006) propose the formulation of image line detection as an inverse problem using the Radon transform, which basically corresponds to a continuous formulation of the Hough transform. They argue that their formulation allows for the exploitation of apriori knowledge using regularizers. More recent works also employ deep learning and CNNs for line detection in images, see for instance Lee et al. (2017) and Zhao et al. (2022). But again, as we aim to find a learning-free approach to our problem, these are not in the scope of this chapter.

Interestingly, O'Mahony et al. (2019) compare the performance of methods employing deep learning for image analysis tasks to such relying on traditional computer vision in general. They make the point, that deep learning is by far the mightier tool, and has recently led to significant progress in the domain of Digital Image Processing. Still, they conclude some tasks can be solved more efficiently by traditional computer vision means.

5.3 Computer-Vision-Based Transport Unit Side Detection

Our pipeline's first implementation, which we discussed in Chapter 4, makes use of CNNs for instance segmentation in images. The employment of such models requires large amounts of annotated training data, which is in most cases not readily available. Additionally, other disadvantages of CNNs are their computational complexity, as well as their black-box nature. Thus, we explore the construction of learning-free approaches as alternatives to CNN-based instance segmentation models.

To examine the potential of traditional computer vision approaches for automated packaging structure recognition, we exemplary consider the sub-task of transport unit side detection. The step of transport unit side detection in our image processing pipeline is visualized in Fig. 4.1 (c). This task is especially

(a) (b)

Figure 5.1: Exemplary input image and transport unit side annotations (Green color: Annotation of left transport unit side, red color: Annotation of right transport unit side.)

eligible for the application of computer vision methods, as the target structures are well-defined and visual contrast is high.

We implemented an image-processing-based approach targeting the detection of transport unit sides from cropped images showing a single logistics transport unit. Our approach is based on the Hough transform (Hough 1962), a well-established method for detecting straight lines in images. As package and transport unit faces are of regular, rectangular shapes, an image crop showing one transport unit contains many linear structures. The approach's objective is to detect these linear structures, especially the edges of packaging units, to heuristically determine the transport unit side regions within the image. Before we explain our approach step-by-step, we summarize the apriori knowledge, which we aim to exploit.

Apriori Knowledge Fig. 5.1 (b) shows an example of transport unit sides to be detected: the transport unit's left side is highlighted in green, and its right side is highlighted in red. For both transport unit sides, exactly four linear side boundaries of similar sizes are to be found. In the real world, two of these boundaries are horizontally, and the other two are vertically orientated. In general, the four boundary line segments' lengths are of significant length compared to the whole transport unit's extent. Further, three of these lines segment image regions showing packaging unit's from others, showing either transport unit lid, base pallet or image background. Thus, in general, visual differences in terms of color and structure are high and these segments are well distinguishable. The last of these line segments is a shared boundary with the other transport unit side (the left side's right boundary, and the right side's left boundary).

1. Pre-processing: The input image for the packaging structure recognition's step of transport unit side segmentation is a cropped image showing exactly one fully-visible transport unit. A very

simple pre-processing is performed on each input image to the algorithm. More precisely, images are converted to grayscale. We experimented with the standardization of images (i.e. the transformation of pixel values resulting in a distribution with pre-defined mean and standard deviation) but found that such steps did not improve the overall results. Fig. 5.1 (a) shows an exemplary input image before pre-processing.

2. Line Detection: The next step is to detect linear structures within the image. In order to be able to capture all relevant line structures, and to enhance further processing possibilities, we choose a dual approach: We aim to detect line structures depending on their orientation, processing near-vertical and near-horizontal lines separately. This also allows for the exploitation of apriori knowledge about the transport unit's geometry and the image's compositions and characteristics.

To detect qualifying horizontal and vertical structures, two different edge detection filter kernels are applied to the image, and the resulting edge images are binarized. One of these kernels aims at detecting horizontal edges, the other one on vertical edges. More precisely, the Hewitt operator was used. Amer and Abushaala (2015) introduce and compare common edge detection methods. In this case, we opted against the established edge detector by Canny (1986), which is a traditional approach and is still widely used for various tasks involving edge detection in images. The reason for our choice is that the Canny method detects edges with arbitrary orientation, whereas we aim to detect horizontal and vertical structures separately. We use the following convolutional kernels in our edge detector:

$$Pr_h = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{5.1}$$

$$Pr_v = \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \tag{5.2}$$

Equation 5.1 states the kernel for the detection of horizontal structures, and Equation 5.2 contains the kernel concerning vertical structures.

After binarization, the image foreground is restricted to the actual transport unit region using the pixel mask which is input to the step of transport unit side segmentation. This is to avoid background structures from being detected along the relevant transport unit lines. Exemplary resulting binary edge images are shown in Fig. 5.2 (a) and (b).

The binary edge images are used as input for the Hough transform in order to find linear structures. We set the Hough transform's line sampling distances to 1 pixel, and 1°, respectively. Further reduction of sampling distances did not yield improved results in our experiments. The minimum length for edge structures to be classified as lines by the Hough transform is set to 0.4 times the image height for vertical lines and 0.25 times the image width for horizontal lines. This minimum length relative to the corresponding image dimension is smaller in the case of horizontal lines, as

we assume horizontal lines to be either part of the left, or of the right transport unit side. In both cases, the corresponding transport unit side covers approximately half the image width. At the same time, vertical lines cover the whole transport unit's vertical extent, and thus a greater fraction of the overall image extent in this direction. However, we define a minimum number of both horizontal and vertical lines which need to be detected. In case fewer than the minimum acceptable number of lines are detected, the minimum length for lines is decreased and the Hough transform is repeated, iteratively, until the minimum line number is reached. This parameter is set to 25 for both classes of lines. The maximum number of lines for both horizontal and vertical lines is set to 100. Additional lines are discarded, thereby lines of greater length are preferred. Depending on viewpoint, distance and perspective, the accurate direction of horizontal and vertical lines may deviate by several degrees from 0°and 90°, respectively. Line filtering depending on the line angle is performed for vertical lines: The maximum acceptable deviation from vertical direction is set to 10° for vertical lines. Detected vertical lines with greater deviations are discarded. For horizontal lines, no filtering is performed. The procedures, thresholds and parameters described above were found to yield the best results in our experiments. The line detection results are illustrated in Fig. 5.2 (c) and (d).

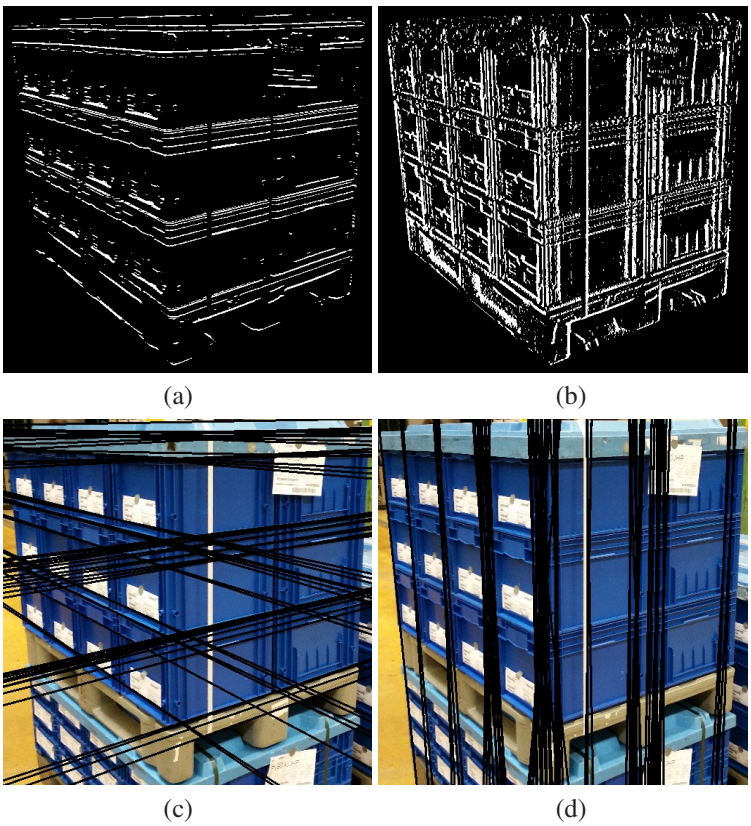


Figure 5.2: Detection of horizontal and vertical lines. (a) Binary image of horizontal edge structures, (b) binary image of vertical edge structures, (c) detected vertical lines, (d) detected horizontal lines.

3. Vanishing Point Estimation: After the line detection has been performed, we try to determine the image's vanishing points (Barnard 1983) for vertical lines and for the visible transport unit sides' horizontal lines. To do so, we use a heuristic approach, again exploiting the knowledge of the image's contents and its geometric properties. We assume that the majority of vertical line segments detected correspond to the vertical edges of the transport unit. After computing all intersection points of these vertical lines, we use the mean value of all intersection points as the first guess for the unit's vertical vanishing point. Subsequently, we drop lines that do not get sufficiently close to the current vanishing point estimate. Then, we refine the estimate based on the intersection points of the reduced set of lines. For the two vanishing points of horizontal lines on our transport unit sides, we proceed similarly. Following the computation of the intersection points of horizontal lines. After all intersection points of horizontal lines are computed, we try to find two accumulation points of horizontal line intersections: One on the left-hand side of the image and one on the right-hand side. Then, each horizontal line is either assigned to the left or right transport unit side, depending on its minimum distance to the vanishing point estimations. Once again, we refine the vanishing point positions for left-side and right-side vanishing points by considering only intersections of the lines assigned to the left side, or the right side, respectively. The step of line assignment and vanishing point computation is repeated once again to improve result quality. Fig. 5.3 (c) illustrates vanishing point estimation and line assignments.

Figure 5.3: Estimation of vanishing points.

In this step, we again rely on a suitable perspective of the input image: While generally a roughly on-level view is requested, the step of vanishing point estimation requires the view of the pallet to be slightly top-down or bottom-up. Otherwise, the transport unit's vertical structures may be depicted by virtually parallel lines within the image. In such a case, the estimation of a vanishing point of these vertical lines becomes impossible, and our approach fails. In practice, it is rather difficult to acquire such a perfectly aligned image, even if aiming to do so. Our dataset did not contain an image, for which that problem occurred.

4. Side Boundary Estimation: Based on the vanishing points and corresponding lines, we try to segment the transport unit sides. To do so, the start and end points for all horizontal line segments are determined. This is done by first creating a binary image of the line under consideration. Using the AND operator, the image is matched to the binary edge image which was created in step 1 (as input to the Hough transform). The resulting binary edge segment image is filtered in such a way, that only coherent structures of sufficient length are preserved. We then use the outermost points of the filtered binary image as the start and end points for the corresponding line segment. Employing the obtained line endpoints, we estimate the transport unit side boundaries by fitting regression lines through the corresponding endpoints of each line set and the side's corresponding vanishing point. For instance, to find the left boundary of the left transport unit side, we regress a line through the left side's vertical vanishing point and the left endpoints of all horizontal lines assigned to the left side. The regression line, in this case, is the line with minimal mean distance to applicable line endpoints and the vanishing point. First, all vertical side boundaries are determined. Fig. 5.4 (a) illustrates the vertical side boundary estimation: Red (green) lines indicate the left (right) side's line segments (as determined by the estimated endpoints), and the three black lines indicate the determined side's boundaries. Subsequently, for the upper and lower boundaries, a joint regression for both sides is performed. This is illustrated again in Fig. 5.4 (b): The side's vertical line segments are illustrated in darker red and green color, respectively, the brighter red and green lines indicate the side's boundaries. By joint regression, we mean both upper and both lower boundaries are optimized jointly, as we can assume both lines to intersect on the middle vertical side boundary, which we found previously. This exploitation of apriori knowledge leads to significantly improved results. In the last step, the transport unit side corner points are inferred by intersecting side boundary lines. Exemplary overall side detection results are shown in Fig. 5.4 (c).

(a) (b) (c)

Figure 5.4: Trasnport unit side boundary estimation and side segmentation. (a), (b): Determination of vertical and horizontal side boundary lines. (c): Resulting transport unit sides.

Table 5.1: Transport unit side segmentation evaluation results: Average IoU values.

Method	Average IoU		
	All units	KLT	Tray
CNN	0.9170	0.9113	0.9372
Image Processing	0.7914	0.7907	0.7939

Overall, there is a considerable number of thresholds and similar parameters contained in this approach. For instance, in finding binary edge images, the parameters involved are kernel sizes and patterns and binarization threshold. Further parameters are required when performing the Hough transform, e.g. the minimum length of line segments to consider, as well as distance and angle resolutions. Also, in vanishing point estimation and line assignments, and line endpoint determination, numerous threshold parameters are involved.

5.4 Results: Comparing Computer Vision and Deep Learning Approaches

In this section, we evaluate the method's performance for the task of transport unit side segmentation and compare the results to those of the CNN-based instance segmentation model. For all evaluation images, the intersection over union (IoU) of the two predicted transport unit sides, relative to the annotated sides, is computed. In a first evaluation, we compare the average IoU values of both methods. If any of the methods failed to find an annotated transport unit side completely, this is considered an IoU of 0. Table 5.1 shows the corresponding overall results, as well as the results by packaging type. The computer vision approach is not able to achieve results as good as those of the learning-based method. Averaged over all evaluation images, the average IoU of transport unit sides predicted by the computer vision approach is 0.791, compared to 0.917 for the deep learning model's predictions. Both KLT and tray units are, on average, predicted more accurately by the deep learning model.

Additionally, we compute an accuracy value as the relative number of correct detections based on given IoU thresholds. Thereby, we consider different IoU threshold values between 0.5 and 0.95. More precisely, the reported accuracy at IoU threshold 0.95 is the ratio of annotated transport unit sides, for which detections with an IoU of at least 0.95, relative to present ground-truth annotations, were found. The computed accuracy values for both methods in consideration are shown in Table 5.2.

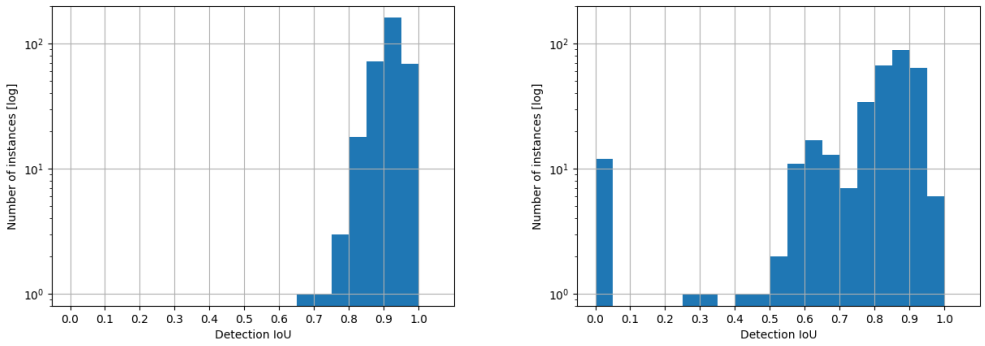
Fig. 5.5 illustrates the distribution of the two method's IoU values by means of corresponding histograms. On the x-axis the IoU value range was sectioned into 20 equidistant bins of width 0.05. The y-axis indicates the number of transport unit side instances, in each bin. Note the logarithmic scaling of the figure's y-axis, which was applied to allow for better countability of bins containing only very few instances. If a transport unit side was not detected, this is accounted for with an IoU value of 0. In part (a), we can see that the lowest reported IoU for our CNN implementation lies within the bin ranging from 0.65 to 0.7. For our image processing pipeline, 11 transport unit sides

Table 5.2: Transport unit side segmentation evaluation results: Accuracy at different IoU thresholds.

	IoU threshold					
Method	0.5	0.6	0.7	0.8	0.9	0.95
CNN	1.000	1.000	0.997	0.985	0.709	0.212
Image Processing	0.951	0.911	0.819	0.693	0.215	0.018

could not be detected at all. Of the detected transport unit sides, the lowest reported IoU value lies within the range of 0.25 to 0.3. As supported by the numbers in Tables 5.1 and 5.2, the total number of very accurately segmented transport unit sides (i.e. IoU greater than 0.9) is significantly higher for the CNN-based method. Quantifying confidence intervals for these statistics, the detected transport side IoU for the CNN algorithm is greater than 0.864 with a probability of 90%. For the image-processing-based algorithm, one has a 90% probability that the IoU exceeds a value of 0.614.

The values show that, in the current implementations, the CNN outperforms our image processing approach by a great margin.



(a) (b)

Figure 5.5: Histograms of IoU values for the two implementations of transport unit side segmentation: (a) CNN and (b) Image Processing. Note the logarithmic scale of the y-axis.

5.5 Discussion of Experiment Results and Implications

In our experiments, our computer vision algorithm for transport unit side detection proved to be very sensitive to parameter choices. Results can fluctuate significantly if single parameters are slightly varied. At the same time, it is extremely difficult to find near-optimal parameters for a whole set of images. Parameters working well in some cases may yield underwhelming results on other images, e.g. if other packaging types, packaging structure, packaging components, or different imaging conditions are present. Even though it is possible to parameter-tune the image processing algorithm to deliver precise results for single images or groups of images, we were not successful in finding

parameters yielding good results on the whole data set. The evaluation values shown are the best values achieved by systematically varying the involved parameters. The CNN, on the other hand, easily adapts to data as diverse as ours, due to the huge number of learnable parameters.

Notably, there are numerous possibilities to enhance our computer vision algorithm to yield better results in a broad selection of images. The chain of individual steps in our algorithm was constructed based on literature and existing approaches, task-specific knowledge and manual observations. Certainly, no exhaustive search of optimal processing steps was performed and a better algorithm design could most probably be found. On the other hand, images could be split into different classes, based on arbitrary compositional or visual features, and different parameter choices could be applied to each class. Further, machine learning could be introduced to enhance the quality of parameter choices, or even optimize the selection of processing steps, which are performed. Still, the implementation and optimization efforts included in such improvements are not to be neglected. Additionally, the application of machine learning approaches introduces the need for annotated training data, which we aim to eliminate in this case study.

To summarize the results of our case study, we infer that it is not feasible to replace the deep-learning-based algorithms with learning-free computer vision approaches. This does not necessarily mean, that it is not possible to solve the task of transport unit side detection on our dataset by means of traditional computer vision. Instead, we argue the effort required to find such a solution is significant, and might even exceed the effort involved in the annotation and training of our CNN model. Further, we anticipate the CNN model to be more robust against possible changes in imaging parameters and perspective, packaging components, or environmental conditions. Regarding the motivating research question (RQ2), we conclude that CNN algorithms for the implementation of our packaging recognition pipeline are the best choice, given our conditions, criteria and objectives. To our knowledge, we are the first to pit state-of-the-art deep-learning-based methods for image analysis against traditional computer vision techniques utilizing a practical case study from a real-world logistics problem. The provided insights may allow others to make similar design decisions early on when building related recognition pipelines - inside or outside the domain of logistics.

In the next chapter, we will focus on our last research question (RQ3), and move in a different research direction: We will leave traditional image processing techniques and design more sophisticated deep learning models for object detection and instance segmentation tasks within the use case of packaging structure recognition.

6 TetraPackNet: Specialized Deep Learning Approaches for Packaging Structure Recognition

The contents of this chapter were partially published in Dörr et al. (2021). This chapter does not only elaborate on the same work at full length but also contains additional information, ideas and experiments.

6.1 Research Question Elaboration

The research question (RQ3), which we tackle in this chapter was already introduced in Section 1.2, and reads as follows:

(RQ3) How can available apriori knowledge about geometrical restrictions of transported materials be utilized to design specific detection algorithms, yielding improved results in packaging structure recognition?

As we investigate this research question, our focus remains on the use case of packaging structure recognition, and the image processing pipeline, which we introduced in Chapter 4. One crucial step in this pipeline is the recognition of two visible transport unit side faces, by finding the exact positions of their four corner points. This is a reasonable objective, as transport unit side faces, as well as packaging unit faces, are of rectangular shape in the real world (compare Section 4.3). In this chapter's experiments, we aim to exploit this apriori knowledge about our detection targets in order to construct specific, purposeful detection models for the task of transport unit side detection. This is necessary, as existing detection methods usually employ generic object representations like bounding boxes, which are too inaccurate in our case, or segmentation masks, which incorporate too many degrees of freedom for our purpose. Thus, we aim to design specific image object detection models outputting exact object locations based on four characteristic feature points. The first step toward the design and implementation of such a model is the proper definition of the targeted object representation.

It is noteworthy, that similar assumptions about detection target geometry are also applicable in other logistics use cases, such as package detection or transport label detection. The same holds for non-logistics applications like the recognition of license plates or documents, and other cases where objects of regular geometric shapes need to be segmented accurately to perform further downstream processing steps, often like image rectifications or perspective transforms.

In this chapter, we present a redesigned version of the object detection model CornerNet by Law and Deng (2020), namely TetraPackNet, which segments objects by four arbitrary corner points (i.e. tetragons) instead of bounding boxes or pixel masks. We evaluate the approach on our data concerning the use case of logistics packaging structure recognition. Baselined against our previous solution to the sub-task of transport unit side corner detection, we show that TetraPackNet can achieve improved detection rates. Further, we observe that predictions made by TetraPackNet are in general very accurate and corner positions are predicted precisely. For the use case of logistics packaging structure detection, TetraPackNet represents a novel, promising method to achieve improved detection results, prospectively.

Our contributions contained in this chapter can be briefly summarized as follows:

- We introduce an object representation based on four corner points (rather than bounding boxes or pixel masks).
- We design a model, named TetraPackNet, for the detection of objects based on four corner points, on the basis of the state-of-the-art model CornerNet.
- We evaluate TetraPackNet on our use case of transport unit side detection, and find its correct detections to be of very high accuracy.
- We propose an alternative object grouping strategy, using object extent instead of predicted embeddings, which is suitable for regularly arranged rigid objects (as is the case in packaging structure recognition).
- We discuss the potentials of TetraPackNet and possible enhancements.

6.2 Related Work

This section covers existing work, which is in some way related to our approach to detecting specifically shaped objects in packaging structure recognition. Related work regarding object detection and instance segmentation, in general, was already covered in previous chapters and is not fully elaborated at this point.

Liu et al. (2020) give an overview of the task of generic object detection and the most relevant recent works regarding the topic. They also argue that metrics and the level of scene understanding have been changing progressively: where early work on the topic mostly targets image classification, the focus has gradually shifted to bounding box detection and, subsequently, to pixel-wise object segmentation. The previously mentioned authors (Liu et al. 2020) mainly focus on bounding box object representations. Minaee et al. (2022) provide a comprehensive survey into pixel-wise image segmentation, both semantic and instance-based, using deep learning. Our approach aims to move away from both of these common object representations.

Numerous less conventional approaches toward the detection of specifically shaped objects in images exist. One of these approaches is the deep-learning-based cuboid detection by Dwibedi et al. (2016). This work in the context of 3D reconstruction focuses on the detection of cuboid-shaped, class-agnostic objects and the precise localization of their vertices. We refrain from comparing to this

work for several reasons, one of which is the requirement for richer image annotations (cuboid based, eight vertices per object). Further, we do not aim to reconstruct 3D models from our images but aim to classify and interpret intra-cuboid information.

Most approaches to edge, contour and line detection are based on traditional computer vision techniques (image gradients, feature detectors like SIFT, Hough transform). Recently, Huang et al. (2018) proposed a deep-learning-based approach to detect so-called wireframes, consisting of straight line segments and junctions, in images. While wireframes are generally very relevant for our task of packaging structure recognition, we favor different detection methods. This is because the grouping and interpretation of detected wireframes to relevant objects like transport unit sides and packaging units is not at all trivial. Additionally, it may be complicated by background structures (i.e. other transport units within the image or image crop).

An application in which ideas and approaches very similar to ours are frequently employed is the popular one of human pose estimation. Conventionally, the task of human pose estimation corresponds to the prediction of a fixed number of characteristic key points, such as, for instance, head, shoulders, torso or wrists. Dang et al. (2019) provide a recent comprehensive survey on human pose estimation using deep learning methods. State-of-the-art approaches are mostly based on similar backbone CNNs as successful instance segmentation models, e.g. Inception ResNet v2 (Szegedy et al. 2016a), Stacked Hourglass (Newell et al. 2016). The succeeding network layers may differ in design. Often, heatmaps indicating keypoint positions are used (Xiao et al. 2018) (Chen et al. 2018), but fully convolutional layers resulting in keypoint position regression are also common (Luvizon et al. 2019, Toshev and Szegedy 2014).

This chapter's approach builds on CornerNet by Law and Deng (2020), a state-of-the-art model performing object detection without incorporating anchor boxes or other object position priors. Instead, corner positions of relevant objects' bounding boxes are predicted using convolutional feature maps as corner heat maps. Corners of identical objects are grouped based on predicted object embeddings, as proposed by Newell et al. (2017). This approach, which outperformed all previous one-stage object detection methods on COCO Lin et al. (2014), was further developed and improved by Duan et al. (2019) and Zhou et al. (2019). The follow-up work by Law et al. (2020), CornerNet-Lite, introduced faster and even more accurate variations of the original CornerNet method. These advancements of the original CornerNet are not in our scope, we build upon the original work (Law and Deng 2020).

Our approach makes use of recent work by Newell et al. (2017). The authors perform instance segmentation by outputting score pixel masks for each object category, where the score indicates the probability that a pixel belongs to the associated object category. By subsequent grouping of pixels of identical object categories by the similarity of predicted scalar tag values, object instances are distinguished. We employ this proposed approach in our work to group a predefined number of detected key points to object instances. Whereas the authors use traditional pixel-wise masks as object instance representations, we use keypoint-based representations.

6.3 Object Representations

6.3.1 Existing Object Instance Representations

Revisiting existing literature to object detection in 2D images, the majority of approaches employ one of the following very common generic object representations:

- Bounding Boxes
- Instance-level Pixel Masks
- Semantic Pixel Masks

The first of these representations, bounding boxes can be applied to cases, where objects need to be detected or counted in images, but exact object positions or pose are not required. In this setting representing object positions by a rectangular image area that contains the whole object instance is sufficient. Naturally, as objects are generally not rectangular-shaped, such bounding boxes can contain significant amounts of non-object areas. Technically, bounding boxes are low-dimensional simple shapes that can be parameterized by four scalar values. They can be fully described by a single pixel position (upper left corner or center, for instance) and the bounding box' width and height or, equivalently, by two pixel coordinates (bounding box upper left and lower right corner, for instance). Mathematically, a bounding box $B(x_0, y_0, w, h)$ within an image plane $P = \{0, ..., m\} \times \{0, ..., n\}$ can be written as

$$B(x_0, y_0, w, h) = \{(x, y) \in P | x \in [x_0, x_0 + w] \wedge y \in [y_0, y_0 + h]\} \tag{6.1}$$

where $x_0, y_0 \in P$ and $w, h \in \mathbf{N}$ such that $(x_0 + w, y_0 + h) \in P$.

Pixel masks are of significantly higher complexity and can, in theory, describe arbitrarily shaped areas. In instance-level segmentation, fixed-shape binary maps are often used to parameterize masks. This type of object representation is used when accurate pixel-based localization of objects in images is required. For each object within an image, a distinct binary mask is used to describe the object's extent. A pixel mask M in an image with pixel space $P = \{0, ..., m\} \times \{0, ..., n\}$ is a mapping $M : P \rightarrow \{0, 1\}$ with $M(x, y) = 1$ if and only if (x, y) belongs to the object described by pixel mask M.

Semantic masks tackle slightly different object detection scenarios, where an instance-level distinction of objects is not required. Instead, an object class is assigned to each pixel and thereby the image is segmented into object class regions. Boundaries between two instances of identical objects can not be detected. This is, for instance, adequate in many applications like autonomous driving: here, the information where obstacles might be found is crucial but the counting of or distinction between obstacles is not required. Technically, pixel masks as described above are used to represent this segmentation output. But instead of using one individual mask for each object instance present within an image, one single mask is sufficient to describe the whole image's semantics. Semantic segmentation is not sufficient for our use case requiring instance-based segmentation and will not be considered further.

6.3.2 Feature-Point Based Object Representation

For our use case, none of the previously mentioned common object representations is ideal. Bounding boxes, on the one hand, are not complex enough to accurately localize the objects of interest, i.e. transport unit sides. Pixel masks, on the other hand, are highly complex and offer a too great amount of freedom regarding output object shapes. The object representation targeted by our sub-tasks of transport unit side and packaging unit face detection consists of exactly four pixel coordinates within an image:

$$p_1 = (x_1, y_1), p_2 = (x_2, y_2), p_3 = (x_3, y_3), p_4 = (x_4, y_4) \in P \tag{6.2}$$

By connecting these points in order $p_1p_2p_3p_4p_1$, an area within the image is described. Arbitrary tetragonal shapes inside the image plane P can be described by four such pixel coordinates. Note that the order in which four points are connected is not arbitrary. For our work, we assume point orderings p_1, p_2, p_3, p_4 with:

$$x_1 \leq x_2 \tag{6.3}$$
$$x_3 \leq x_4 \tag{6.4}$$
$$y_1 \leq y_3 \tag{6.5}$$
$$y_2 \leq y_4 \tag{6.6}$$

If these equations hold, it is ensured that the shape obtained by connecting $p_1p_2p_3p_4p_1$ is a single tetragon, and not two triangles in an hourglass-like arrangement.

Notice that bounding boxes $B(x_0, y_0, w, h)$ are a special case of such tetragons with $y_1 = y_2, x_3 = x_2$, $y_3 = y_4$ and $x_1 = x_4$. Similarly, every tetragon as described above can be described by a binary pixel mask $M : P \rightarrow \{0, 1\}$.

Subsequently, we design a task-specific convolutional neural network (CNN) detecting objects by four arbitrary pixel coordinates rather than regular bounding boxes or pixel masks. To achieve that, we build upon existing work by Law and Deng (2020), Law et al. (2020), enhancing the ideas of CornerNet, moving to the higher-dimensional output object space of arbitrary tetragons. Fig. 6.1 illustrates the difference between commonly used object location representations, i.e. bounding boxes, and our four-corner-based representation. The indicated object is one of the two transport units' faces that need to be precisely localized for the motivating task of packaging structure recognition.

6.4 TetraPackNet

In this section, the architecture of our model TetraPackNet, and especially the modifications compared to role model CornerNet are elaborated. This does not only include the design of the deep learning model itself, but also the necessary post-processing steps which assemble the tetragonal detections from the raw CNN outputs.

Figure 6.1: Sample annotations. Left: Bounding box annotation. Right: Four-corner-based annotation. The example image is taken from our use-case-specific dataset.

6.4.1 Network Design

We present a novel method for four-point-based object detection, based on the recent CNN object detector CornerNet, introduced by Law and Deng (2020).

Whereas in most traditional object detectors, object locations are referenced by bounding boxes (i.e. top left and bottom right corner position), we work with more detailed locations described by four object vertices. The resulting shapes are not limited to rectangles but comprise arbitrary tetragons, i.e. four-cornered polygons.

We use model, ground-truth and loss function designs very similar to those proposed as CornerNet by Law and Deng (2020). All of these components, and our modifications for TetraPackNet, targeting tetragon-based object detection, are explained in the following sections. Fig. 6.2 gives an overview of our architecture. The additional components compared to CornerNet are highlighted.

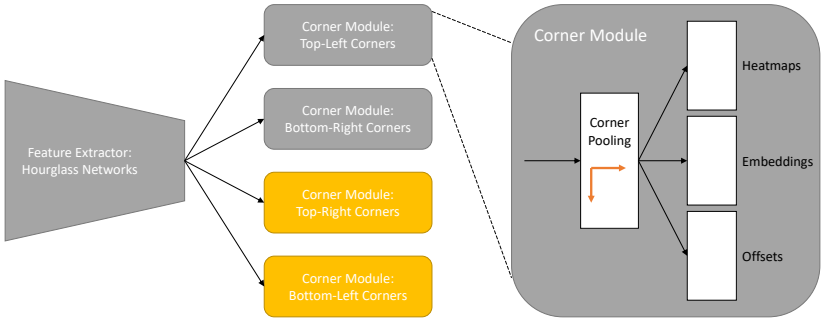


Figure 6.2: TetraPackNet architecture. Differences to CornerNet are highlighted.

As indicated, the main difference to the original architectures is the number of corner prediction modules following the backbone network: While CornerNet employs two such prediction modules resulting in adequate information to predict two diagonally opposed corners of bounding boxes, TetraPackNet requires four corner prediction modules to precisely predict all four corners of tetragon-shaped objects.

Notably, the characteristics of predicted points are fundamentally different in both models: CornerNet predicts bounding box corners, which are in general not a feature point of the object itself, but often are located outside the actual object region. TetraPackNet, on the other hand, aims to predict specific object feature points or, more precisely, object vertices.

Backbone Network

As suggested and applied by Law et al. (2020), we use an hourglass network (Newell et al. 2016), namely Hourglass-54, consisting of 3 hourglass modules and 54 layers, as backbone network. Hourglass networks are fully convolutional neural networks. They are shaped like hourglasses in that regard, that input images are downsampled throughout the first set of convolutional and max pooling layers. Subsequently, they are upsampled to the original resolution in a similar manner. Skip layers are used to help conserve detailed image features, which may be lost by the network's convolutional downsampling.

In TetraPackNet's network design, two instances of the hourglass network are stacked atop each other as follows. After an initial block of four downsampling operations (7x7 convolution, stride 2, 128 channels) and one residual block (stride 2, 256 channels), the two hourglass modules are applied subsequently. Each hourglass module reduces the spatial resolution five times while increasing the number of channels from 256 to 512. More precisely, the channel number for the input layer and the five subsequent downsampled layers are 256-384-384-384-512. Before and after the first hourglass module, Batch Normalization is applied. The input to the second hourglass module is obtained from the first module's normalized inputs and outputs by applying a residual block (256 channels) to the ReLU activation of the element-wise addition of both feature maps.

Corner Detection and Corner Modules

Following the backbone network's hourglass modules, so-called corner modules are applied to predict precise object corner positions. CornerNet utilizes two such corner modules to detect top left and bottom right corners of objects' bounding boxes. Our architecture includes four corner detection modules for the four corner types top-left, top-right, bottom-left and bottom-right. We do not detect corners of bounding boxes but precise corner locations of tetragon-shaped objects.

Analogously to the original CornerNet approach, each corner module is fully convolutional and consists of specific corner pooling layers as well as a set of output feature maps of identical dimensions. These outputs are corner heat maps, offset maps and embedding. They each work in parallel on identical input information: the corner-pooled convolutional feature maps.

We shortly revisit CornerNet's specific pooling strategy. It is based on the idea that important object features can be found when starting at a bounding box's top left corner and moving horizontally in right or vertically in downward direction. More precisely, by this search strategy, object boundaries will be hit by bounding box definition. In CornerNet, max pooling is performed in the corresponding two directions for both bounding box corner types. The pooling outputs are added to one another and the results are used as input for corner prediction components. The authors show the benefits of this approach in several detailed evaluations. In our case, where precise object corners are instead

of bounding box corners, one may argue that pooling strategies should be reconsidered. Still, for our first experiments, we retain this pooling approach. The implementation of other experimental pooling strategies is an interesting research opportunity, we will consider in future work on the topic.

The corner module's three output sets are explained in the following. For each distinguished corner type, i.e. top left, top right, bottom left and bottom right corners in our case, the model includes one heat map predicting positions where the probability for a corner of the respective type is high. As the resolution of the corner module's feature maps is lower than that of the original input image, additional location offsets are predicted for each potential corner candidate. To enable the assembly of four corresponding object corners to an object, embeddings are predicted for each corner. These embeddings aim to take such values that corners of the same object are as similar as possible, while those of corners of distinct objects differ significantly. The before-mentioned components corner pooling, corner heat map, offsets and embedding maps are combined to form a single corner prediction module.

More precisely, each corner module is a set of fully convolutional layers including corner heat maps for each object category, two offset maps for horizontal and vertical offset and, in case of one-dimensional embeddings, one embedding map. Moreover, we extended the CornerNet architecture to include four corner prediction modules instead of two. Additionally, TetraPackNet's corner prediction modules do not aim to detect bounding box corners, but vertices of tetragonal-shaped objects.

Ground-truth

Required image annotations are object positions described by the object's four corner points, i.e. top left, top right, bottom left, bottom right corner. It is required that both right corners are further right as their counterparts and, equivalently, both top corners are further up as the corresponding bottom corners. For each ground-truth object, one single positive location is added to each of the four ground-truth heatmaps. To allow for minor deviations of corner detections from these real corner locations, the ground-truth heatmaps' values are set to positive values in a small region around every corner location. As proposed by CornerNet, we use a Gaussian function centered at the true corner position to determine ground-truth heatmap values in the vicinity of that corner.

In Fig. 6.3 ground-truth heatmaps and detected heatmaps and embeddings are illustrated. The top row shows, cross-faded on the original input image, the ground-truth heatmaps for the four different corner types. There are two Gaussian circles in each corner type heatmap as there are two annotated ground-truth objects, i.e. two transport unit sides, in the image. The bottom row shows TetraPackNet's detected heatmaps (for object type transport unit side) and embeddings in a single visualization: Black regions indicate positions where the predicted heat is smaller than 0.1. Wherever the detected heat value exceeds this threshold, the color indicates the predicted embedding value. To map embedding values to colors, the range of all embeddings for this instance was normalized to the interval from 0 to 1. Afterward, Open CV's Rainbow colormap was applied (Bradski and Kaehler 2008).

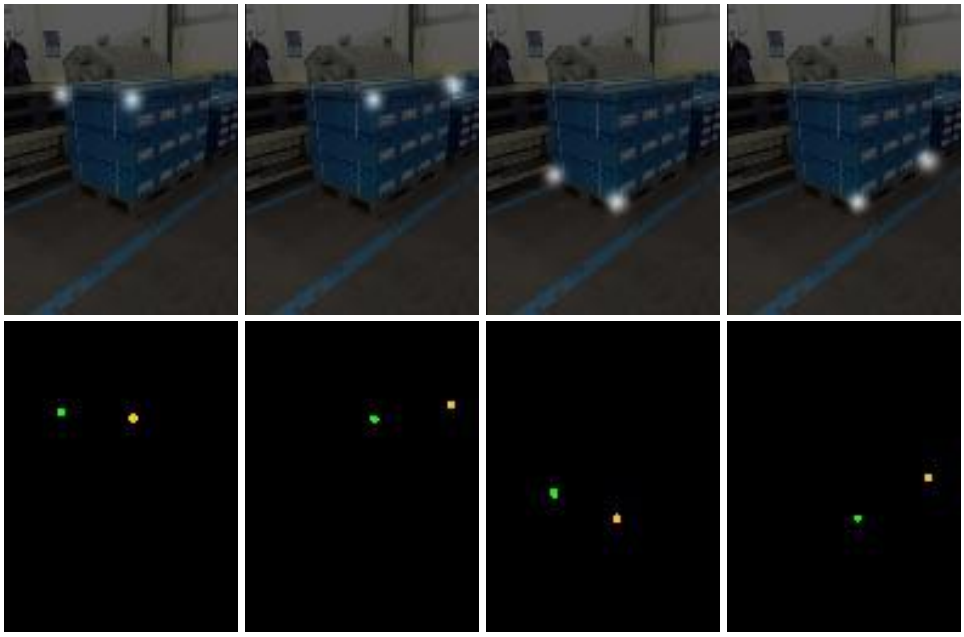


Figure 6.3: Example heatmaps. Top row: Ground-truth. Bottom row: Detected heats and color-encoded embeddings.

Loss Function

The loss function used in the training of our TetraPackNet model consists of several components:

$$L = L_{\text{det}} + w_{\text{off}} \cdot L_{\text{off}} + (w_{\text{pull}} \cdot L_{\text{pull}} + w_{\text{push}} \cdot L_{\text{push}}) \tag{6.7}$$

In our experiments, the loss component weights were set to $w_{\text{pull}} = w_{\text{push}} = 0.1$ and $w_{\text{off}} = 1.0$, as proposed by Law and Deng (2020). The individual loss components are explained in the following.

Focal Loss L_{det}: The loss term's first component L_{det} is a focal loss (Lin et al. 2018) variant, as proposed by CornerNet. This term aims to optimize heatmap corner detections by penalizing high heatmap values at points where there is no ground-truth corner location. Analogously, low heat values at ground-truth positive locations are penalized. Let y_{cij} be the ground-truth heat value for class c at location i, j. As mentioned in Section 6.4.1 ground-truth heat values are not binary, but positive-valued regions are drawn around each ground-truth corner location. To incorporate this in the focal loss formulation, an additional factor $(1 - y_{cij})$ is added at locations where $y_{cij} < 1$. The focal loss for class c at location i, j is

$$L_{\text{det}}(c, i, j) = \begin{cases} (1 - h_{cij})^{\alpha} \log(h_{cij}) & y_{cij} = 1 \\ (1 - y_{cij})^{\beta} (h_{cij})^{\alpha} \log(1 - h_{cij}) & \text{otherwise} \end{cases} \tag{6.8}$$

where h_{cij} is the predicted heat value for class c at position i, j. The hyperparameters α and β are set to $\alpha = 2, \beta = 4$. Overall, the focal loss is computed as

$$L_{\text{det}} = \frac{1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} L_{\text{det}}(c, i, j) \tag{6.9}$$

where C is the overall number of classes and $H \times W$ is the model's heat map resolution.

Offset Loss L_{off}: The offset loss L_{off} is used to penalize deviations in offset predictions in vertical and horizontal directions. As proposed in CornerNet, a simple smooth L1 Loss, comparing predicted and actual precise corner positions is used.

Pull and Push Loss L_{pull} and L_{push}: The last loss components, pull and push loss, are used to optimize the embedding values predicted at each potential corner location. As the model itself outputs candidate locations for object corners, the assignment of these corner predictions to complete objects, consisting of four corners each, must be performed subsequently. In our case, each object prediction is composed by finding a group of four suitable corner predictions (exactly one corner prediction per corner type, i.e. one top left corner, one top right corner, and so forth). The predicted embedding values serve as an indicator for corner affiliation: Corners are grouped into objects based on the similarity of their embedding values. More precisely, the objective of embeddings is to predict embedding values as similar as possible for all corners of the same object instance. At the same time, embedding values of distinct objects' corners should be as far apart as possible. To achieve the first part of this objective, embedding similarity for corners of identical objects, the pull-loss L_{pull} is used. The pull loss is computed as

$$L_{\text{pull}} = \frac{1}{N} \sum_{k=1}^{N} \sum_{i \in \{tl,tr,bl,br\}} (e_i(k_i) - e(k))^2 \tag{6.10}$$

where $k = 1, ..., N$ enumerates the ground-truth objects and $i = tl, tr, bl, br$ indicate the four corner types top left, top right, bottom left, bottom right. The position of corner i of ground-truth object k is denoted by k_i. Further, e_i denotes the embedding map for corner type i and therefore $e_i(k_i)$ is the embedding value for corner i of the ground-truth object k. The average value of the embedding values of all four corners of a ground-truth object k is given by $e(k) = \frac{1}{4} \sum_{i \in \{tl,tr,bl,br\}} e_i(k_i)$.

The push loss penalizes average embeddings of different objects being similar to each other, thereby "pushing" the embeddings of corners of different objects apart.

$$L_{\text{push}} = \frac{1}{N(N-1)} \sum_{k=1}^{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} \max\{0, 1 - |e(k) - e(j)|\} \tag{6.11}$$

6.4.2 Assembling Corner Detections to Objects

Once corner positions and their embeddings are predicted, these predictions need to be aggregated to form tetragonal object detections. Compared to the CornerNet setup, this task appears more complex as each object is composed of four corners instead of only two. However, the original grouping implementation is based on Associative Embeddings (Newell et al. 2017), which is suitable for multiple data points in general, i.e. more than two. The same approach can be applied in our case.

To obtain an overall ranking for all detected and grouped objects, the four corner detection scores as well as the similarity of their embeddings, i.e. the corresponding pull loss values, are considered. This final score for a detection p of class c consisting of four corners $p_{tl}, p_{tr}, p_{bl}, p_{br}$ is computed as

$$\frac{1}{4} \sum_{i \in \{tl,tr,bl,br\}} h_i(c, p_i) + (e_i(p_i) - e(p))^2 \tag{6.12}$$

with $e(p)$ being the average embedding for a set of four corners as before. Further, $h_i(c, p_i)$ denotes the predicted heat value for class c and corner type $i \in \{tl, tr, bl, br\}$ at position p_i.

Additionally, we only allow corners to be grouped which comply with the condition, that right corners are further right in the image than their left counterparts. Analogously bottom corners are required to be further down in the image than the corresponding top corners.

6.5 Evaluation

In this section, we examine the performance of TetraPackNet. First, TetraPackNet is trained for transport unit side detection on our use case data, and results are compared to the previously introduced baseline model. This evaluation is performed using standard and use-case-specific metrics. In an additional section, we motivate and propose an alternative corner grouping strategy and evaluate the approach.

6.5.1 Transport Unit Side Detection

To evaluate TetraPackNet for our use case, we compare its performance on the use-case-specific dataset, described in Section 4.4, to a standard Mask R-CNN model. Both models were trained for the single-class detection problem of transport unit side segmentation, as described in Section 2.2.

Setup: In both cases, the same dedicated training, validation and test splits were used. Training and evaluation were performed on an Ubuntu 18.04 machine on a single GTX 1080 Ti GPU unit.

Table 6.1: Instance segmentation evaluation results for the whole image scenario on our 150 evaluation images.

Model	AP	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$	$AP_{0.95}$
Mask R-CNN	80.8	97.0	97.0	96.0	93.3	49.0	4.0
TetraPackNet	86.1	91.6	90.4	90.4	89.4	85.7	52.5

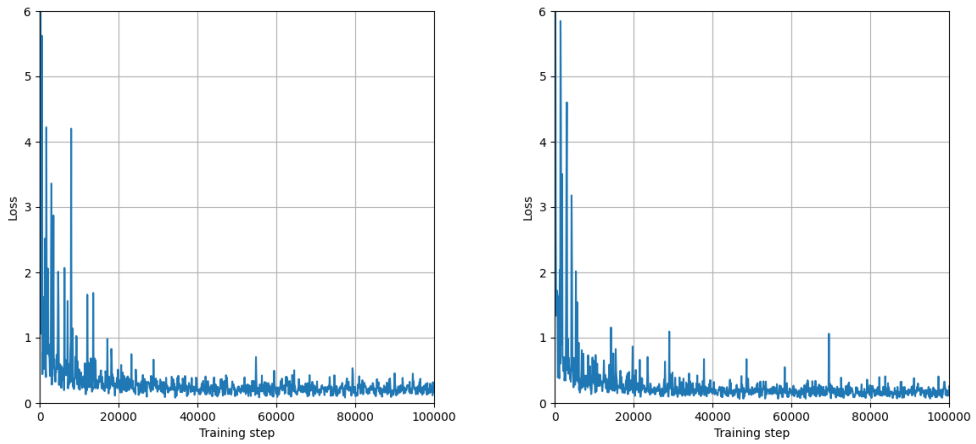
Two different training scenarios are evaluated for both models: First, the models are trained to localize transport unit sides within the full images. In a second scenario, the cropped images are used as input instead: as implemented in our packaging structure recognition pipeline (see Section 4.3), all images are cropped in such ways that each crop shows exactly one whole transport unit. For each original image, one or multiple such crops can be generated, depending on the number of transport units visible within the image. This second scenario is comparatively easier as exactly two transport unit sides are present in each image and the variance of the scales of transport units within the image is minimal.

Training Details: In both trainings, we tried to find training configurations and hyperparameter assignments experimentally. However, due to the high complexity of CNN training and its time consumption, an exhaustive search for ideal configurations could not be performed. Most likely, improvements are possible in both cases. Still, we consider the results presented in the following an affirmation of our proposed architecture TetraPackNet.

To achieve fair preconditions for both training tasks, the following prerequisites were fixed. Both models were trained for the same amount of epochs: The training of the Mask R-CNN baseline model included 200.000 training steps using a batch size of 1, whereas the TetraPackNet training included 100.000 training steps with a batch size of 2. Input resolution for both models was limited to 512 pixels per dimension. Images are resized such that the larger dimension measures 512 pixels and the aspect ratio is preserved. Subsequently, padding to a quadratic shape is performed. We implemented image augmentation methods analogously to the training of the Mask R-CNN models: random flip, crop, scale, color distortions and conversion to gray values.

Figure 6.4 illustrates the training progress of the two TetraPackNet models by plotting the loss values throughout the 100,000 training steps. As targeted in the training of CNN models, the loss values seem to saturate before the end of the training period is reached.

Standard Metric Results: As a standard evaluation metric, the COCO (Lin et al. 2014) dataset's standards are used. As is common, we report average precision (AP) as the standard metric for averaged IoU thresholds from 0.5 to 0.95 (AP). Additionally, intersection over union (IoU) at selected thresholds from 0.5 ($AP_{0.5}$) to 0.95 ($AP_{0.95}$) are stated explicitly, to allow for differentiated results interpretations. Remember, as the applied IoU threshold increases, the requirements regarding segmentation accuracy increase, and, naturally, the AP value decreases. Tables 6.1 and 6.2 show the corresponding results for the instance segmentation problem. Fig. 6.5 illustrates the same results to provide a visual intuition.



Cropped image scenario Whole image scenario

Figure 6.4: Training process illustration of TetraPackNet for both trained models.

Table 6.2: Instance segmentation evaluation results for the cropped image scenario on our 163 evaluation image crops.

Model	AP	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$	$AP_{0.95}$
Mask R-CNN	87.7	100.0	100.0	100.0	98.8	73.3	13.4
TetraPackNet	92.8	98.0	98.0	98.0	95.7	91.4	60.2

Considering the average precision values at the lowest IoU threshold examined (0.5), the baseline Mask R-CNN outperforms TetraPackNet by significant margins. E.g. in the scenario considering uncropped images (Table 6.1), Mask R-CNN's $AP_{0.5}$-value is 5.4 points higher than that of TetraPackNet (97.0 vs. 91.6). However, as the IoU threshold for detections to be considered correct increases, TetraPackNet gains the advantage. When regarding performance values at IoU threshold 0.8, Mask R-CNN, with an $AP_{0.8}$ of 93.3, still provides a higher amount of detections with bounding boxes classified as correct. At IoU threshold 0.9, the $AP_{0.9}$ for the Mask R-CNN model experiences a considerable drop to 49.0, whereas TetrPackNet's $AP_{0.9}$-value remains comparably stable at 88.4.

Very similar observations can be made for the cropped image scenario: TetraPackNet clearly outperforms the reference model Mask R-CNN when high accuracies (IoU greater than 0.8) are required. The corresponding evaluation results are shown in Table 6.2.

Overall, the results suggest that TetraPackNet does not detect quite as many ground-truth transport unit sides as our Mask R-CNN baseline model on a low accuracy basis. At the same time, the predictions made by TetraPackNet appear to be very precise, because average precision steadily remains on a high level as IoU accuracy requirements are increased. For our use case of packaging structure recognition, these should be desirable conditions, as our processing pipeline requires very accurate transport unit side predictions.

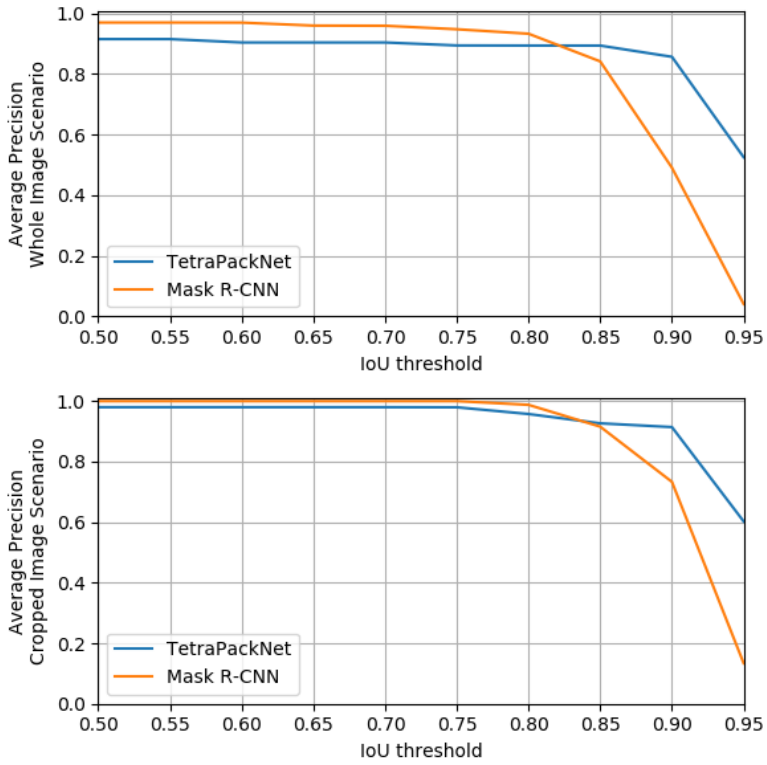


Figure 6.5: Average precisions at different IoU thresholds for TetraPackNet and Mask R-CNN baseline model. Top: Whole image scenario. Bottom: Cropped image scenario.

Use Case Specific Results: To investigate the performance and benefits of TetraPackNet for our specific use case, other metrics than standard COCO Average Precision are needed. Within our use case of packaging structure detection, the precise localization of each transport unit side's four corner points is crucial. Therefore, we baseline our results against our previous approach, which relies on Mask R-CNN to obtain transport unit side segmentation masks. In a post-processing step, four corner points giving the best approximation of these masks were found by solving a suitable optimization task. Input to the task in our image processing pipeline, and for these evaluations, are cropped images showing exactly one full transport unit.

Evaluations are performed analogously to Sec. 5.4. The average IoU values for transport unit sides are given in Table 6.3. Overall, the average IoU for TetraPackNet's detections is slightly above the average IoU of the baseline method based on Mask R-CNN and our custom post-processing. If only considering Tray units, however, TetraPackNet performs slightly inferior. Note that in this evaluation, the resulting IoU is averaged over all annotated transport unit sides, i.e. if a transport unit side is not detected at all, this corresponds to an IoU value of 0.0.

Table 6.3: Transport unit side segmentation evaluation results: Average IoU values.

Method	Average IoU		
	All units	KLT	Tray
CNN	0.917	0.911	0.937
TetraPackNet	0.937	0.938	0.933

Table 6.4: Transport unit side segmentation evaluation results: Accuracy at different IoU thresholds.

Accuracy	IoU threshold					
Method	0.5	0.6	0.7	0.8	0.9	0.95
CNN	1.000	1.000	0.997	0.985	0.709	0.212
TetraPackNet	0.988	0.988	0.985	0.957	0.902	0.672

Table 6.4 states the accuracy values of TetraPackNet, compared to our baseline method, for different IoU thresholds. Similar to the previously presented results, TetraPackNet is superior when high segmentation accuracy is required. More precisely, if an IoU of 0.9 is expected, TetraPackNet's accuracy still reads 90.2% whereas the baseline Mask R-CNN method's accuracy drops to 70.9%. The difference in accuracy increases even further if an even higher IoU of 0.95 is required (67% compared to 21.2%).

The previously stated observations can be confirmed by considering transport side detection result visualizations for both TetraPackNet and the baseline Mask R-CNN in Fig. 6.6. One can see here, that Mask R-CNN's detected masks (red) are kind of fuzzy and often cut corners or exceed the transport unit side's extent. TetraPackNet's masks (green), on the other hand, are straight by nature and align with the actual transport unit side's edges accurately. As a result, the polygon corner points placed by our post-processing (see Section 4.3) of Mask R-CNN's outputs (black circles) are less accurate than TetraPackNet's output tetragon corner points.

Manual Error Analysis

To get an understanding of the situations, in which TetraPackNet fails to segment transport unit sides correctly, we visually analyze some of these error cases. Fig. 6.7 shows manually selected error cases. The four examples reflect TetraPackNet's common sources of error, which are:

- Missing feature point detections
- Incorrect grouping of feature points
- Incorrectly located feature points
- Misclassified feature points

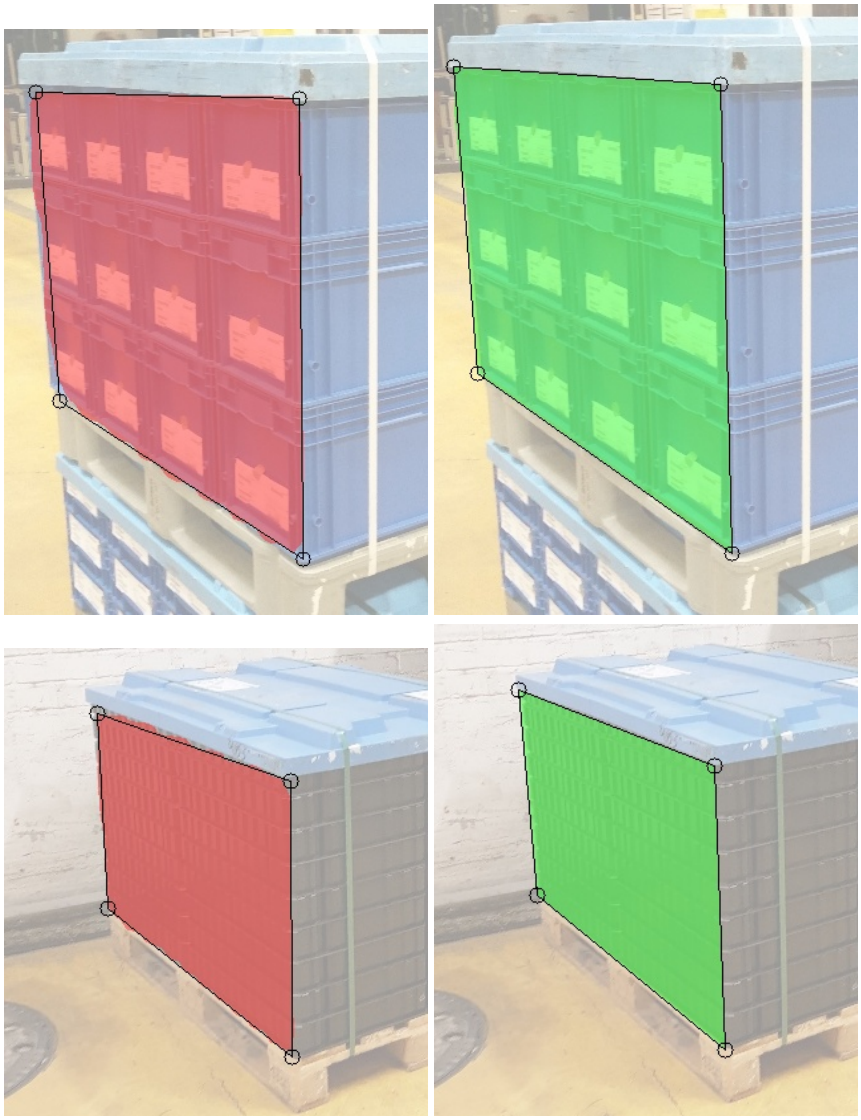


Figure 6.6: Exemplary results of both methods for transport unit side detection under consideration (TetraPackNet and Mask R-CNN). Left: Mask R-CNN, right: TetraPackNet.

(a) (b)

(c) (d)

Figure 6.7: Examples for cases in which TetraPackNet does not segment all transport unit sides correctly. (a) One transport unit side is not found at all. (b), (d) A corner point of an adjacent background transport unit side is mistakenly employed. In (d), the type of corner point is mistaken, too (top left instead of bottom left corner). (c) Corner points are misplaced by a few pixels onto the transport unit lid instead of the packaging unit corner.

Fig. 6.7, part (a), shows an example in which one of the transport unit's sides was not detected at all. In this case, the right side's bottom right corner point was not detected as a relevant feature point by TetraPackNet. Thus, even though the other three corner points were detected precisely, the model is not able to detect the transport unit side under consideration. This is one of TetraPackNet's drawbacks which could possibly be corrected by adequate, yet extensive post-processing steps: If three out of four corner points are detected properly, a reasonable approximation for the corresponding object's last corner point could be found, again by exploiting the geometrical apriori knowledge. Remember, the object of interest is known to be of rectangular shape in the real world. As we further know the image acquisition process to be a perspective transform (if disregarding camera distortions), a system of equations yielding the missing corner point position can be constructed. The implementation of post-processing steps solving this system of equations and approximating the missing corner point is not trivial and outside the scope of our work.

Another common error type is the incorrect grouping of correctly detected feature points. Fig. 6.7, part (b), depicts a corresponding example. The lower transport unit's left side is detected incorrectly, even though all of its corner points are detected in TetraPackNet's heatmaps. Additionally, the bottom left feature point of a neighboring transport unit's side is also detected (which is not an error per se, as the model is trained to detect all transport unit side corners). When grouping the detected feature points into detection objects, the model chooses the incorrect bottom left feature point to be grouped with the transport unit side's other feature point. This results in a too-large detection spanning across the neighboring transport unit.

Another error produced by TetraPackNet in our scenario is the incorrect placement of feature points. A corresponding example is illustrated in Fig. 6.7 (c). In this case, both transport unit sides' common top point was placed incorrectly (top right point of the transport unit's left side, and top left point of the transport unit's right side). It was placed in such a way that the transport unit sides include part of the transport unit's lid, instead of only including the instance's packaging units.

Lastly, part (d) of Fig. 6.7 shows an example of two different error types at the same time: The transport unit's left side is segmented incorrectly due to an error in the grouping of the detected feature points, and the misclassification of one of these feature points. More precisely, the bottom left corner point of a background transport unit side is misclassified and assigned as top left corner of the relevant transport unit side.

These examples show that the type and characteristics of errors in the case of TetraPackNet differ from those of the baseline Mask R-CNN model. Errors are more often due to background transport units. Such units lead to additional, oftentimes correct, feature point detections. These detections can result in erroneous feature point groupings.

6.5.2 Experiment: Embedding-Free Detection to Object Grouping Method

The experiment of this section is motivated by an observation made by manual examination of TetraPackNet's detection errors. In the case of multiple packaging units being (partially) visible in a single image, errors often occur in the grouping of objects from detected corner points. It seems to be the case that embeddings are not always a reliable predictor for corner assignments. Especially in the case of stacked transport units, the embeddings of both left (or right) transport unit sides seem to be

very similar for both transport units. This can be explained by the visual similarity often present for highly standardized transport units. A corresponding example is shown in Fig. 6.8. In the left image, individual detection results are indicated by differently colored tetragons. The light blue tetragon represents an example with incorrectly assigned corners. The right image shows the color encoding of the bottom-right corner embeddings. The hardly distinguishable greenish colors indicate that the embeddings for the left transport unit sides are very similar for both top and bottom transport units. This results in TetraPackNet confusing top and bottom transport units and erroneous transport unit side detections.

Figure 6.8: False assignment of object corners to objects and color-encoded embedding values of botttom right corners.

Apart from that observation, one might argue that the prediction of embeddings might not be necessary for our use case of packaging structure recognition: In our case, objects of interest are assembled in order and do not overlap. At least, this is guaranteed for the transport unit to be analyzed (both for its two transport unit sides, as well as its visible package unit faces). Background transport units may yet again disturb this condition. Nevertheless, if non-overlapping and regularly assembled objects of similar sizes are assumed, we suggest a different strategy for the grouping of transport unit sides. This strategy is based on the fact that the order of the object's feature points is fixed in the case of transport unit side detection: The right corners are further right, and the bottom corners are further down within the image. Based on these preconditions, we claim that the step of grouping detected feature points to objects can be performed by preferring valid pairs of feature points with smaller object circumference.

Formally, we implement an object grouping strategy proceeding as follows: For each given valid set of four (refined) corner points $p_{tl}, p_{tr}, p_{bl}, p_{br}$ of identical class c we compute the corresponding object extent E:

$$E(p_{tl}, p_{tr}, p_{bl}, p_{br}) = |p_{tr} - p_{tl}| + |p_{br} - p_{tr}| + |p_{bl} - p_{br}| + |p_{tl} - p_{bl}| \tag{6.13}$$

Table 6.5: Results for the two object grouping strategies for the whole image scenario on our 150 evaluation images.

Model	AP	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$	$AP_{0.95}$
Group by Embedding	86.1	91.6	90.4	90.4	89.4	85.7	52.5
Group by Extent	85.5	92.0	91.9	90.9	90.8	83.7	40.9

Table 6.6: Results for the two object grouping strategies for the cropped image scenario on our 163 evaluation image crops.

Model	AP	$AP_{0.5}$	$AP_{0.6}$	$AP_{0.7}$	$AP_{0.8}$	$AP_{0.9}$	$AP_{0.95}$
Group by Embedding	92.8	98.0	98.0	98.0	95.7	91.4	60.2
Group by Extent	91.0	97.9	97.9	97.9	93.7	86.4	51.6

We compute an extent-sensitive object score as

$$\left(\sum_{i \in \{tr,tl,br,bl\}} \frac{h_i(c, p_i)}{4} \right)^{0.25} + \left(1 - \frac{E(p_{tl}, p_{tr}, p_{bl}, p_{br})}{2 \cdot (H + W)} \right) \tag{6.14}$$

where $h_i(c, p)$ is the predicted heat value of heat map i for class c at point p, and $H \times W$ is the heatmap resolution, as before.

We rank all possible corner groupings correspondingly. Objects are accepted in this order where it is ensured that each corner point is assigned only once. Moreover, if an object grouping of four corners is accepted, all other possible groupings of lower score containing one identical corner detection are discarded.

The results for the scenario of transport unit side detection on complete images (without cropping) in terms of the COCO standard metrics are shown in Table 6.5, the results for the cropped image scenario are shown in Table 6.6. Overall, the suggested embedding-free approach "Group by Extent" performs slightly inferior to the original implementation. The new strategy's AP evaluates to 85.5, as compared to 86.1 in the case of the embedding-based approach. The analogous observation for the cropped image scenario is very similar: AP of 91.0 vs. 92.8.

Fig. 6.9 compares the results for both grouping strategies on three example evaluation images. The first two examples ((a) and (b)) show how our new object grouping strategy can indeed solve the issue of distinguishing and grouping multiple objects with similar embedding values, which was described above. The last example, 6.9 (c), illustrates a case in which our new strategy did not improve the results. In our new grouping strategy, the heat values indicating the probability for a feature point are not as important when ranking possible the best corner groupings. Even though these heats are still a factor in equation 6.14, this can in some cases lead to low-confidence false positive feature points being selected over true positive detections (depending on their relative positions within the image).

(a)

(b)

(c)

Figure 6.9: Example results of TetraPackNet with grouping via embedding (left) and grouping via object extent (right). (a), (b) Erroneous detections of the original TetraPackNet implementation could be corrected by the new grouping method. (c) The new grouping method introduces a new error as a feature point of lower heat value is preferred over the correct one.

6.6 Result Discussion and Assessment

In this chapter, we implemented a novel algorithm, TetraPackNet, for the detection of tetragonal-shaped objects in images, rather than relying on generic object representations like bounding boxes or pixel masks. Applying this approach to the task of transport unit side detection, we proved it to be a competitive option when requiring highly accurate segmentation. Our examination proved the approach to be rather susceptible to missing complete objects in some cases. At the same time, objects found by TetraPackNet are mostly very precise, which was the main objective of designing the method. To enhance TetraPackNet's recall values, elaborate post-processing steps exploiting the given apriori knowledge could be designed.

In its current state, TetraPackNet is not able to boost the overall performance of our recognition pipeline. This coincides with our experiments from Chapter 4, in which we found, that even replacing the transport unit side detections by perfect ground-truth annotations does not yield increased overall accuracy. To elevate the performance of TetraPackNet, further development toward the detection of specifically shaped objects is targeted. In the case of packaging structure recognition, enhanced detection models targeting both transport unit sides simultaneously (6 feature points), or even a whole transport unit including base pallet and lid (12 feature points) may be applicable.

For our use case of packaging structure recognition, we proposed to replace the embedding predictions adopted from the role model CornerNet in favor of an approach grouping feature points to objects by minimizing the resulting object's extent. Even though the replacement of the object grouping strategy did not yield improved overall results for the task of transport unit side detection, the results are promising and virtually at eye level with the original approach. Our experiment shows that a simplified model, which does not predict any feature point embeddings, can be employed without losing too much accuracy, even improving results in some cases. Further, we predict the approach to be even more promising for the task of package unit face detection, which is just as relevant for packaging structure recognition. This is, because the number of objects to detect in each image is significantly higher in the latter task, while the placement of these objects is just as regular (aligned, without gaps or overlaps). The higher the number of objects in an image, the harder it is to distinguish these objects' feature points via embedding values. The extent-based approach on the other hand does not suffer from a large number of objects, as long as all relevant feature points are detected. We aim to implement TetraPackNet-based package detection and verify these presumptions, in future experiments.

This concludes our work on the research question (RQ3) and packaging structure recognition. There are many opportunities and chances to continue research on the topic, which we partially already highlighted. The succeeding last chapter will elaborate on possible future work and give a comprehensive summary of our work until this point.

7 Conclusion

In this thesis, we aimed to introduce and investigate the problem of automated packaging structure recognition on 2D images. We presented the use case and its relevance for logistics processes, and developed solution approaches thereof. In this section, we summarize our findings and results. Further, we give an outlook and present directions for future work regarding packaging structure recognition.

7.1 Summary

From an economic perspective, we briefly introduced and explained the necessity and applications of packaging structure recognition; it is a relevant task throughout logistics networks, for instance in goods receipt or goods issue departments. In numerous logistics supply chains, standardized logistics transport units are handled by different stakeholders and need to be checked for completeness and integrity frequently. The manual counting and verification efforts involved in such checks can be reduced by automatizing this mandatory task of packaging structure recognition. Even though our work can not completely eliminate manual efforts from incoming goods checks, and similar use cases, it incorporates an essential step towards the comprehensive automation of such processes.

In Chapter 1, we formulated three research questions to explore the technical problem of packaging structure recognition on 2D images. To obtain a feasible problem formulation, restrictions regarding materials and transport unit structure, as well as image acquisition and composition, were determined. A large set of annotated use case images is required for testing, evaluation, and also for training purposes. As the use case of packaging structure recognition has, to our knowledge, not yet been systematically studied, and corresponding data is not readily available, we introduced our own set of 1,000 annotated images, respecting the previously defined restrictions.

Our first research question concerned the design of an image processing pipeline to solve the task of packaging structure recognition on valid 2D images. We proposed a suitable multi-step algorithm, based on state-of-the-art convolutional neural networks in Chapter 4. Further, optimization methods, heuristic approaches and computations, are incorporated into our pipeline. In a quantitative evaluation using our dedicated evaluation data, the solution achieved an accuracy value of 85%. The performance was significantly higher for the first of our two packaging types, i.e. KLT units, which are mostly larger and easier to recognize visually, also for the human observer. Most errors by the overall algorithm are due to incorrect detection of packaging units.

There are considerable drawbacks to convolutional neural networks. On the one hand, they are considered black-box algorithms and are not easily comprehensible, or handily tweakable according to one's ideas or requirements. Further, large amounts of annotated training data are required to train convolutional neural networks in a setting of supervised learning. Last, state-of-the-art deep

learning algorithms have high resource requirements, and cannot be executed swiftly on arbitrary hardware systems. For these reasons, we investigated the usage of alternative algorithms in our second research question, in Chapter 5. Namely, we considered the employment of traditional image processing techniques instead of the previously used deep learning approaches. As a comprehensive analysis of the complete image processing pipeline is not practical for complexity reasons, we focused our study on the step of transport unit side recognition. The latter task appears especially suitable for traditional computer vision approaches, as the objects under consideration (package and transport units) are rigid and outlined by geometrically straight lines. Thus, traditional image processing tools, like gradient-based methods for line detection or Hough transforms, are reasonable choices for the analysis of our transport unit images. Using the above-mentioned tools, we implemented a learning-free approach to the sub-task under consideration. Our evaluations show that convolutional neural networks are superior in segmenting transport unit sides on image crops depicting exactly one transport unit. We certainly do not claim that it is not possible to design a superior image processing approach to the latter task. However, we find image processing approaches not to be feasible when considering requirements regarding algorithm design, implementation, and tuning, as well as aspirations for robustness.

Our third and last research question focused on the specificity of the instance segmentation algorithms used in our packaging structure recognition pipeline. One goal of our work at that end was to incorporate the apriori knowledge about our objects of interest into the applied deep learning models. Moreover, we aimed to design a CNN for object detection, which can accurately segment transport unit sides by exploiting the fact, that package and transport units are rigid objects and of regular cubic shapes. In Chapter 6, we proposed a detection algorithm adapted to the segmentation of tetragonal-shaped objects, namely TetraPackNet. In our implementation of packaging structure recognition, the accurate segmentation of objects of regular geometric shapes is crucial in multiple sub-steps. For instance, we aim to segment cubically shaped packages and transport units, or more specifically, rectangular faces thereof. Commonly used detection algorithms target either bounding boxes or pixel-accurate masks. As both are not adequate object representations in our use case, we identified this as a potential for performance increases, which can be achieved through the design and employment of more specific detection methods. We designed TetraPackNet to perform object detection based on four corner points and trained the model for the task of transport unit side detection. When applying established standard metrics (Pascal VOC), the overall accuracy of TetraPackNet still trails that of the established instance segmentation model, as objects are missed completely more frequently. However, our evaluations also show, that our method is able to conduct highly accurate object segmentation. When requiring high accordance between model predictions and ground-truth annotations of transport unit sides (IoU greater than 0.8), TetraPackNet outperforms the baseline model, which is based on a Mask R-CNN instance segmentation model. This shows that TetraPackNet is a promising approach, which is especially suitable to perform high-accuracy segmentation of geometrically shaped objects.

Overall, our work substantiates the presence of further automation potentials in logistics processes: Manual effort in common operations like the checking of incoming and outgoing goods can be drastically reduced by the implementation of a system for fully-automated packaging structure recognition. Even though a 100%-solution can not (yet) be the target of such a system, and manual checks will still be required in difficult cases, significant savings regarding manual effort and time are possible. Further, we are confident that our method for automated packaging structure recognition can be

further improved by purposeful advancements to the algorithm, and also the training data, which is a significant contributor to result quality. We will elaborate on our ideas for such improvements in the following section.

7.2 Outlook and Future Work

Throughout our work, some opportunities for improvements or further experiments regarding our packaging structure recognition algorithm were identified. This is especially relevant for TetraPackNet, the novel object segmentation algorithm presented in Chapter 6. While the model's detections are generally of very high accuracy, objects are not detected by the model if only one of the necessary feature points is missed by the detector. To increase the model's detection rates, more sophisticated post-processing methods could be tailored. Given the apriori knowledge about object shapes, arrangement and alignment, missing feature points could be interpolated. This is even more applicable when adapting TetraPackNet for the task of package unit detection, as packaging units are arranged in a very regular pattern, and package unit corner points coincide with those of neighboring units. Apart from that, improved detection rates and accuracies might be achievable by transitioning from four-corner-point objects (i.e. transport unit sides or package unit faces) to downright transport unit templates consisting of even more inherent feature points (including both transport unit sides as well as base pallet and unit lid).

In its current implementation, quite a few restrictions and assumptions apply to our packaging structure recognition pipeline. Hereinafter, we reflect the restrictions relevant to the real-world application (our imaging restrictions do in our opinion not limit the applicability of our method), and propose possible solutions thereof.

First, we restricted our work to a small number of standardized package types and packaging components. While adaptation to other standardized package types, given suitable data, should not pose a problem for our learning-based algorithm, the acquisition of such data sets can be very costly. The synthetic generation of annotated training images is often seen as a resolution to this problem. Still, the performance of models trained on synthetic data generally still trails that of models employing real-world training data. In many cases, using a mixture of real-world and synthetic data can be a powerful compromise to create accurate models with the ability to detect object classes of which not a single real-world instance was seen during training. This seems to be a promising approach for the expansion of our packaging structure recognition algorithm to additional packaging types, and also packaging components or distractor objects (like transport or hazard labels, packaging foils or straps).

We maintained further restrictions which limit the automation potential opened by our solution, i.e. the assumption of uniformly packed transport units. In practice, non-uniformly packed units, i.e. units that are either composed of multiple distinct package types or whose packaging layers consist of different numbers of packages, are not unusual. To deal with such cases, additional improvements to our image processing pipeline are necessary. Most importantly, improved accuracy in transport unit side and package segmentation would be required. Currently, we exploit the knowledge that all packaging units are of identical size by using mean package size to infer package numbers in horizontal and vertical directions. If this assumption is no longer valid, it becomes essential that every

single packaging unit is segmented correctly. Further enhancements of TetraPackNet to become more robust against missing feature point detections could prove to be a fundamental step toward this goal. Additionally, a single RGB image may not be sufficient to recognize the transport unit's packaging structure in the case of non-uniformly packed units. At the very least, two images taken from opposite sides would be required to tackle such transport units. Using such two images as input for our packaging structure detection, and joining the information inferred from both inferences, can be implemented on top of our algorithm easily. In other cases, it might not be possible to infer a transport unit's packaging structure from its outer appearance without disassembling it. Consequently, the task of packaging structure detection from RGB images is no longer solvable. Other input information, e.g. video streams of a transport unit's assembly or disassembly may be a prospective remedy here (requiring enhanced recognition algorithms and pipelines).

Apart from these technical restrictions, our experiments were limited to a restricted domain of images. Although an aim for varying background settings was pursued, all images were taken in a single facility. Only two package types were employed and the visual appearance of the instances of these two package types was rather uniform (i.e. most packages were of identical manufacturing, though the color and individual structure of KLT packages, for instance, can vary). To obtain more general conclusions, similar experiments on data of broader variance have to be deduced. Such experiments are not easily accessible as they require large amounts of labeled image data. For this reason, they are not included within the scope of this work but might become the focus of future research in the domain of packaging structure recognition.

On the other hand, even better results might be achieved by limiting the approach to an even more restricted setting. If, for instance, implementing a system for automatic packaging structure recognition in an incoming goods department, a very controlled setting can be achieved easily. The camera can be fixed to a position where incoming goods can be overseen from an advantageous perspective. For example, a gate through which incoming goods need to be transported on a forklift, whilst being captured by an attached and properly adjusted camera, could be employed. In doing so, disturbances by background units can be completely avoided, if choosing proper positioning. As we have seen in our TetraPackNet experiments, many recognition errors of our packaging structure recognition components are due to background transport units of similar appearances being present in our dataset's images. Further, issues due to variations in lighting and perspective could be minimized in such a setting. The construction of such an experimental system for live packaging structure recognition is an interesting direction for future work.

Overall, we provided a fundamental advance toward the automation of packaging structure recognition, and thereby, of several logistics processes like goods receipt checks or package empties counting. Notably, our work is only one of the puzzle pieces required to fully automatize such use cases, and additional developments are required to that end. Most importantly, the (partial) recognition and reading of at least one of the attached transport labels would be required for a system to know which order a transport unit is associated with. Further, at least a visual check for damages or tampering would need to be incorporated. Additional checks, like the recognition of the base pallet's serial number or the verification of packaging instructions, may be required. Such checks could be implemented alongside, or on top, of our packaging structure recognition. Still, other process steps may be less readily automatized, for instance, manual package content inspections.

Disregarding the logistics use cases our work is focused on, it would be interesting to explore the potential of our feature-point-based object detector, TetraPackNet, for other applications. The method could be of relevance in many other cases where regularly shaped rigid objects need to be segmented accurately, e.g. detection of labels, road signs or license plates, or package detection for grasping point determination in robotics.

Bibliography

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. (2016). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467*. DOI: 10.48550/arXiv.1603.04467.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning : A Textbook*. Springer eBook Collection. Cham: Springer International Publishing. DOI: 10.1007/978-3-319-94463-0.

Aggarwal, N. and W. C. Karl (2006). "Line detection in images through regularized Hough transform". In: *IEEE transactions on image processing* 15.3, pp. 582–591. DOI: 10.1109/ICIP.2000.899595.

Akçay, S., M. E. Kundegorski, M. Devereux, and T. P. Breckon (2016). "Transfer learning using convolutional neural networks for object classification within x-ray baggage security imagery". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. Phoenix, AZ, USA, pp. 1057–1061. DOI: 10.1109/ICIP.2016.7532519.

ALPAC (1966). *Language and Machines: Computers in Translation and Linguistics*. Washington, DC: The National Academies Press. DOI: 10.17226/9547.

Alzubaidi, L., J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan (2021). "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions". In: *Journal of big Data* 8.1, pp. 1–74. DOI: 10.1186/s40537-021-00444-8.

Amer, G. M. H. and A. M. Abushaala (2015). "Edge detection methods". In: *2nd World Symposium on Web Applications and Networking (WSWAN)*. IEEE. Sousse, Tunisia, pp. 1–7. DOI: 10.1109/WSWAN.2015.7210349.

Azizpour, H., A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson (2015). "From Generic to Specific Deep Representations for Visual Recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. Boston, MA, USA, pp. 36–45. DOI: 10.1109/CVPRW.2015.7301270.

Barnard, S. T. (1983). "Interpreting perspective images". In: *Artificial intelligence* 21.4, pp. 435–462. DOI: 10.1016/S0004-3702(83)80021-6.

Borstell, H. (2018). "A short survey of image processing in logistics - How image processing contributes to efficiency of logistics processes through intelligence". In: Magdeburg, Germany. DOI: 10.13140/RG.2.2.11060.76168.

Bradski, G. and A. Kaehler (2008). *Learning OpenCV: Computer vision with the OpenCV library*. Sebastopol, CA, USA: O'Reilly Media, Inc.

Breiman, L. (2001). "Random Forests". In: *Machine Learning* 45, pp. 5–32. DOI: 10.1023/A:1010933404324.

Burger, W. and M. J. Burge (2016). *Digital Image Processing : An Algorithmic Introduction Using Java*. 2nd ed. 2016. Texts in Computer Science. London: Springer London. DOI: 10.1007/978-1-4471-6684-9.

Burman, P. (1989). "A Comparative Study of Ordinary Cross-Validation, v-Fold Cross-Validation and the Repeated Learning-Testing Methods". In: *Biometrika* 76, pp. 503–514. DOI: 10.1093/biomet/76.3.503.

Canny, J. (1986). "A Computational Approach To Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8, pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

Cauchy, A. et al. (1847). "Méthode générale pour la résolution des systemes d'équations simultanées". In: *Comptes Rendus de l'Académie des Sciences de Paris* 25.1847, pp. 536–538.

Chauvin, Y. and D. E. Rumelhart (1995). *Backpropagation: Theory, Architectures, and Applications*. New York: Psychology Press. DOI: 10.4324/9780203763247.

Chen, Y., Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun (2018). "Cascaded Pyramid Network for Multi-person Pose Estimation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA, pp. 7103–7112. DOI: 10.1109/CVPR.2018.00742.

Cognex (2022). *Logistics Industry Solutions*. Accessed: 2022-10-10. URL: https://www.cognex.com/industries/logistics.

Cortes, C. and V. Vapnik (1995). "Support-vector networks". In: *Machine Learning* 20.3, pp. 273–297. DOI: 10.1023/A:1022627411411.

Dalal, N. and B. Triggs (2005). "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. San Diego, CA, USA, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

Dang, Q., J. Yin, B. Wang, and W. Zheng (2019). "Deep Learning Based 2D Human Pose Estimation: A Survey". In: vol. 24. 6. Peking: Tsinghua University Press, pp. 663–676. DOI: 10.26599/TST.2018.9010100.

Daxböck, C., J. Kröber, and M. Bergmann (2019). "Digitized performance management along the supply chain". In: *Performance Management in Retail and the Consumer Goods Industry*. Cham: Springer, pp. 405–423. DOI: 10.1007/978-3-030-12730-5_26.

Dörr, L., F. Brandt, M. Pouls, and A. Naumann (2020a). "Fully-Automated Packaging Structure Recognition in Logistics Environments". In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. Vienna, Austria, pp. 526–533. DOI: 10.1109/ETFA46521.2020.9212152.

Dörr, L., F. Brandt, A. Naumann, and M. Pouls (2021). "TetraPackNet: Four-Corner-Based Object Detection in Logistics Use-Cases". In: *Bauckhage, C., Gall, J., Schwing, A. (eds) Pattern Recognition. DAGM GCPR 2021. Lecture Notes in Computer Science*. Vol. 13024. Springer. Cham. DOI: 10.1007/978-3-030-92659-5_35.

Dörr, L., F. Brandt, M. Pouls, and A. Naumann (2020b). "An Image Processing Pipeline for Automated Packaging Structure Recognition". In: *Forum Bildverarbeitung*. Karlsruhe, Germany: KIT Scientific Publishing, p. 239.

Douglas, D. H. and T. K. Peucker (1973). "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature". In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10.2, pp. 112–122. DOI: 10.3138/FM57-6770-U75U-7727.

Duan, K., S. Bai, L. Xie, H. Qi, and Q. Tian (2019). "CenterNet: Keypoint Triplets for Object Detection". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South), pp. 6568–6577. DOI: 10.1109/ICCV.2019.00667.

Dwibedi, D., T. Malisiewicz, V. Badrinarayanan, and A. Rabinovich (2016). "Deep cuboid detection: Beyond 2d bounding boxes". In: *arXiv preprint arXiv:1611.10010*. DOI: 10.48550/arXiv.1611.10010.

Everingham, M., L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman (2010). "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2, pp. 303–338. DOI: 10.1007/s11263-009-0275-4.

Fradkov, A. L. (2020). "Early History of Machine Learning". In: *IFAC-PapersOnLine* 53.2, pp. 1385–1390. DOI: 10.1016/j.ifacol.2020.12.1888.

Furmans, K. and C. Kilger (2019). *Betrieb von Logistiksystemen*. Heidelberg: Springer Vieweg. DOI: 10.1007/978-3-662-57943-5.

Gao, Y. and K. Mosalam (2018). "Deep Transfer Learning for Image-Based Structural Damage Recognition". In: *Computer-Aided Civil and Infrastructure Engineering* 33. DOI: 10.1111/mice.12363.

Ghiasi, G., Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. Cubuk, Q. Le, and B. Zoph (2021). "Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation". In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA, pp. 2917–2927. DOI: 10.1109/CVPR46437.2021.00294.

Girshick, R. (2015). “Fast R-CNN”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

Girshick, R., J. Donahue, T. Darrell, and J. Malik (2013). “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR.2014.81.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep learning*. http://www.deeplearningbook.org. Cambridge: MIT press.

He, K., G. Gkioxari, P. Dollar, and R. Girshick (2017). “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy, pp. 2980–2988. DOI: 10.1109/ICCV.2017.322.

He, K., X. Zhang, S. Ren, and J. Sun (2016). “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

Heistermann, F., M. ten Hompel, and T. Mallée (2017). *BVL Positionspapier: Digitalisierung in der Logistik*. https://www.bvl.de/positionspapier-digitalisierung. Bundesverband für Logistik (BVL).

Herold, D. M., M. Ćwiklicki, K. Pilch, and J. Mikl (2021). “The emergence and adoption of digitalization in the logistics and supply chain industry: an institutional perspective”. In: *Journal of Enterprise Information Management*. DOI: 10.1108/JEIM-09-2020-0382.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov (2012). “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580*. DOI: 10.48550/arXiv.1207.0580.

Hinxlage, J. and J. Möller (2018). “Ladungsträgerzahlung per Smartphone”. In: *Jahresbericht Fraunhofer IML 2018*, pp. 72–73.

Hough, P. V. (1962). *Method and means for recognizing complex patterns*. US Patent 3,069,654.

Huang, G., Z. Liu, L. V. D. Maaten, and K. Q. Weinberger (2017). “Densely Connected Convolutional Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

Huang, K., Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma (2018). “Learning to Parse Wireframes in Images of Man-Made Environments”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA, pp. 626–635. DOI: 10.1109/CVPR.2018.00072.

Huh, M., P. Agrawal, and A. A. Efros (2016). “What makes ImageNet good for transfer learning?” In: *arXiv preprint arXiv:1608.08614*. DOI: https://doi.org/10.48550/arXiv.1608.08614.

Inigo, R. M., E. S. McVey, B. Berger, and M. Wirtz (1984). "Machine Vision Applied to Vehicle Guidance". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6, pp. 820–826. DOI: 10.1109/TPAMI.1984.4767606.

Ioffe, S. and C. Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, pp. 448–456.

Kerner, S., J. Leveling, O. Urbann, L. Weickhmann, M. Otten, and M. Vogel (2020). "Anwendungsfelder von künstlicher Intelligenz in Industrie-4.0-Systemen". In: *Handbuch Industrie 4.0: Produktion, Automatisierung und Logistik*. Berlin, Heidelberg: Springer Vieweg, pp. 227–250. DOI: 10.1007/978-3-662-45537-1_121-1.

Khan, A., A. Sohail, U. Zahoora, and A. S. Qureshi (2020). "A survey of the recent architectures of deep convolutional neural networks". In: *Artificial Intelligence Review* 53.8, pp. 5455–5516. DOI: 10.1007/s10462-020-09825-6.

Kiryati, N., Y. Eldar, and A. M. Bruckstein (1991). "A probabilistic Hough transform". In: *Pattern Recognition* 24.4, pp. 303–316. DOI: https://doi.org/10.1016/0031-3203(91)90073-E.

Koonce, B. (2021). *Convolutional Neural Networks with Swift for Tensorflow : Image Recognition and Dataset Categorization*. Berkeley, CA, USA: Apress. DOI: 10.1007/978-1-4842-6168-2.

Krizhevsky, A., I. Sutskever, and G. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Neural Information Processing Systems* 25. DOI: 10.1145/3065386.

Law, H. and J. Deng (2020). "Cornernet: Detecting Objects as Paired Keypoints". In: *International Journal of Computer Vision*. Vol. 128, pp. 642–656. DOI: 10.1007/s11263-019-01204-1.

Law, H., Y. Teng, O. Russakovsky, and J. Deng (2020). "CornerNet-Lite: Efficient Keypoint based Object Detection". In: *31st British Machine Vision Conference 2020*. Virtual Conference: BMVA Press.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4, pp. 541–551. DOI: 10.1162/neco.1989.1.4.541. eprint: https://direct.mit.edu/neco/article-pdf/1/4/541/811941/neco.1989.1.4.541.pdf.

LeCun, Y., Y. Bengio, and G. Hinton (2015). "Deep Learning". In: *Nature* 521, pp. 436–44. DOI: 10.1038/nature14539.

Lee, J.-T., H.-U. Kim, C. Lee, and C.-S. Kim (2017). "Semantic Line Detection and Its Applications". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy, pp. 3249–3257. DOI: 10.1109/ICCV.2017.350.

Li, H., B. Singh, M. Najibi, Z. Wu, and L. S. Davis (2019). "An Analysis of Pre-Training on Object Detection". In: *arXiv preprint arXiv:1904.05871*. DOI: 10.48550/arXiv.1904.05871.

Lighthill, J. (1972). "Artificial Intelligence: A general survey". In: *Artificial Intelligence: A paper symposium*.

Lin, T., P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie (2017). "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

Lin, T.-Y., P. Goyal, R. Girshick, K. He, and P. Dollar (2018). "Focal Loss for Dense Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2), pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826.

Lin, T.-Y., M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick (2014). "Microsoft COCO: Common Objects in Context". In: *Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science*. Vol. 8693. Cham: Springer. DOI: 10.1007/978-3-319-10602-1_48.

Liu, L., W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen (2020). "Deep Learning for Generic Object Detection: A Survey". In: *International Journal of Computer Vision* 128.2, pp. 261–318. DOI: 10.1007/s11263-019-01247-4.

Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg (2016). "SSD: Single Shot Multibox Detector". In: *Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science*. Vol. 9905. Cham: Springer. DOI: 10.1007/978-3-319-46448-0_2.

Logivations (2022). *KI-basierte Identifikation in der Logistik*. Accessed: 2022-10-10. URL: https://www.logivations.com/de/solutions/agv/camera_identification.php#count_and_measure.

Lowe, D. G. (1999). "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2. IEEE. Kerkyra, Greece, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410.

Luvizon, D. C., H. Tabia, and D. Picard (2019). "Human Pose Regression by Combining Indirect Part Detection and Contextual Information". In: *Computers & Graphics* 85, pp. 15–22. DOI: 10.1016/j.cag.2019.09.002.

Masters, D. and C. Luschi (2018). "Revisiting small batch training for deep neural networks". In: *arXiv preprint arXiv:1804.07612*. DOI: 10.48550/arXiv.1804.07612.

McCulloch, W. S. and W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5, pp. 115–133. DOI: 10.1007/BF02478259.

Minaee, S., Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos (2022). "Image Segmentation Using Deep Learning: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7, pp. 3523–3542. DOI: 10.1109/TPAMI.2021.3059968.

Minsky, M. and S. Papert (1969). *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press.

Mitchell, T. M. (1997). *Machine Learning*. Vol. 1. 9. New York: McGraw-Hill.

Newell, A., Z. Huang, and J. Deng (2017). "Associative Embedding: End-to-End Learning for Joint Detection and Grouping". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., pp. 2274–2284.

Newell, A., K. Yang, and J. Deng (2016). "Stacked Hourglass Networks for Human Pose Estimation". In: *Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science*. Vol. 9912. Cham: Springer. DOI: 10.1007/978-3-319-46484-8_29.

O'Mahony, N., S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh (2019). "Deep learning vs. traditional computer vision". In: *Advances in Computer Vision. CVC 2019*. Springer. Cham, pp. 128–144. DOI: 10.1007/978-3-030-17795-9_10.

Pan, S. J. and Q. Yang (2009). "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10, pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.

Polyak, B. (1964). "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4, pp. 1–17. DOI: 10.1016/0041-5553(64)90137-5.

Qian, N. (1999). "On the momentum term in gradient descent learning algorithms". In: *Neural Networks* 12.1, pp. 145–151. DOI: 10.1016/S0893-6080(98)00116-6.

Quinlan, J. R. (1986). "Induction of decision trees". In: *Machine Learning* 1.1, pp. 81–106. DOI: 10.1007/BF00116251.

Ramesan, R. and J. Mathew (2015). *Hydrological Data Driven Modelling*. Cham: Springer International Publishing. DOI: 10.1007/978-3-319-09235-5.

Redmon, J., S. Divvala, R. Girshick, and A. Farhadi (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

Ren, S., K. He, R. Girshick, and J. Sun (2017). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.06, pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

Rosenblatt, F. (1958). "The perceptron: A probabilistic model for information storage and organization in the brain". In: *Psychological Review* 65.6, p. 386. DOI: 10.1037/h0042519.

Ruder, S. (2016). "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747*. DOI: 10.48550/arXiv.1609.04747.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3, pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

Sadeghi, M. A. and D. Forsyth (2014). "30Hz Object Detection with DPM V5". In: *Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science*. Vol. 8689. Springer. Cham, pp. 65–79. DOI: 10.1007/978-3-319-10590-1_5.

Samuel, A. L. (1959). "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3, pp. 210–229. DOI: 10.1147/rd.33.0210.

Schmidhuber, J. (2015). "Deep learning in neural networks: An overview". In: *Neural Networks* 61, pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

Sharma, S., S. Sharma, and A. Athaiya (2020). "ACTIVATION FUNCTIONS IN NEURAL NETWORKS". In: *International Journal of Engineering Applied Sciences and Technology* 04, pp. 310–316. DOI: 10.33564/IJEAST.2020.v04i12.054.

Shin, H.-C., H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers (2016). "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning". In: *IEEE Transactions on Medical Imaging* 35.5, pp. 1285–1298. DOI: 10.1109/TMI.2016.2528162.

Skingley, J. and A. Rye (1987). "The Hough transform applied to SAR images for thin line detection". In: *Pattern Recognition Letters* 6.1, pp. 61–67. DOI: 10.1016/0167-8655(87)90050-X.

Szegedy, C., S. Ioffe, V. Vanhoucke, and A. Alemi (2016a). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 31. DOI: 10.1609/aaai.v31i1.11231.

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015). "Going deeper with convolutions". In: *2015 IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR). Boston, MA, USA, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna (2016b). "Rethinking the Inception Architecture for Computer Vision". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

Szeliski, R. (2022). *Computer Vision : Algorithms and Applications*. 2nd ed. Texts in Computer Science. Cham: Springer. DOI: 10.1007/978-3-030-34372-9.

Tan, M. and Q. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR. Long Beach, CA, USA, pp. 6105–6114.

The European Pallet Association (EPAL) (2022). *EPAL EURO PALLET*. Accessed: 2023-01-09. URL: https://www.epal-pallets.org/fileadmin/user_upload/ntg_package/images/Produktdownloads/Produktdatenbla__tter/GB/EPAL_Euro_Pallet_800x1200mm.pdf.

Toosi, A., A. G. Bottino, B. Saboury, E. Siegel, and A. Rahmim (2021). "A Brief History of AI: How to Prevent Another Winter (A Critical Review)". In: *PET Clinics* 16.4, pp. 449–469. DOI: 10.1016/j.cpet.2021.07.001.

Toshev, A. and C. Szegedy (2014). "DeepPose: Human Pose Estimation via Deep Neural Networks". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA, pp. 1653–1660. DOI: 10.1109/CVPR.2014.214.

Verband der Automobilindustrie (VDA) (2013). *4530 VDA Einweg Kleinladungsträger System*. Accessed: 2023-01-09. URL: https://www.vda.de/de/aktuelles/publikationen/publication/4530-vda-einweg-kleinladungstr-ger-system.

Vernon, D. (1991). *Machine Vision: Automated Visual Inspection and Robot Vision*. Hemel Hempstead, Hertfordshire: Prentice-Hall International (UK) Ltd.

Viola, P. and M. Jones (2001). "Rapid object detection using a boosted cascade of simple features". In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. IEEE. Kauai, HI, USA, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

Vitronic (2022). *Warehouse Logistics & Distribution Logistics*. Accessed: 2022-10-10. URL: https://www.vitronic.com/en-us/logistics/warehouse-distribution.

Wei, F., C. Alias, and B. Noche (2019). "Applications of Digital Technologies in Sustainable Logistics and Supply Chain Management". In: *Innovative Logistics Services and Sustainable Lifestyles*. Cham: Springer, pp. 235–263. DOI: 10.1007/978-3-319-98467-4_11.

Weiss, K., T. M. Khoshgoftaar, and D. Wang (2016). "A survey of transfer learning". In: *Journal of Big data* 3.1, pp. 1–40. DOI: 10.1186/s40537-016-0043-6.

Wilson, D. R. and T. R. Martinez (2003). "The general inefficiency of batch training for gradient descent learning". In: *Neural Networks* 16.10, pp. 1429–1451. DOI: 10.1016/S0893-6080(03)00138-2.

Woschank, M., E. Rauch, and H. Zsifkovits (2020). "A review of further directions for artificial intelligence, machine learning, and deep learning in smart logistics". In: *Sustainability* 12.9, p. 3760. DOI: 10.3390/su12093760.

Xiao, B., H. Wu, and Y. Wei (2018). "Simple Baselines for Human Pose Estimation and Tracking". In: *Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science*. Vol. 11210. Cham: Springer, pp. 466–481. DOI: 10.1007/978-3-030-01231-1_29.

Xu, L., E. Oja, and P. Kultanen (1990). "A new curve detection method: randomized Hough transform (RHT)". In: *Pattern Recognition Letters* 11.5, pp. 331–338. DOI: 10.1016/0167-8655(90)90042-Z.

Ying, X. (2019). "An Overview of Overfitting and its Solutions". In: *Journal of Physics: Conference Series* 1168, p. 022022. DOI: 10.1088/1742-6596/1168/2/022022.

Yu, J., Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu (2022). "Coca: Contrastive captioners are image-text foundation models". In: *arXiv preprint arXiv:2205.01917*. DOI: 10.48550/arXiv.2205.01917.

Zetes (2022). *Zetes: Warenannahme und Versand*. de. Accessed: 2022-10-10. URL: https://www.zetes.com/en/technologies-consumables/machine-vision.

Zhao, K., Q. Han, C.-B. Zhang, J. Xu, and M.-M. Cheng (2022). "Deep Hough Transform for Semantic Line Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.9, pp. 4793–4806. DOI: 10.1109/TPAMI.2021.3077129.

Zhou, X., D. Wang, and P. Krähenbühl (2019). "Objects as points". In: *arXiv preprint arXiv:1904.07850*. DOI: 10.48550/arXiv.1904.07850.

Zou, Z., K. Chen, Z. Shi, Y. Guo, and J. Ye (2023). "Object Detection in 20 Years: A Survey". In: *Proceedings of the IEEE* 111.3, pp. 257–276. DOI: 10.1109/JPROC.2023.3238524.

List of Figures

List of Tables

List of Author's Publications

Dörr, L., F. Brandt, A. Meyer, and M. Pouls (2019). "Lean Training Data Generation for Planar Object Detection Models in Unsteady Logistics Contexts". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. Boca Raton, FL, USA, pp. 329–334. DOI: 10.1109/ICMLA.2019.00062.

Dörr, L., F. Brandt, M. Pouls, and A. Naumann (2020a). "Fully-Automated Packaging Structure Recognition in Logistics Environments". In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. Vienna, Austria, pp. 526–533. DOI: 10.1109/ETFA46521.2020.9212152.

Dörr, L., F. Brandt, A. Naumann, and M. Pouls (2021). "TetraPackNet: Four-Corner-Based Object Detection in Logistics Use-Cases". In: *Bauckhage, C., Gall, J., Schwing, A. (eds) Pattern Recognition. DAGM GCPR 2021. Lecture Notes in Computer Science*. Vol. 13024. Springer. Cham. DOI: 10.1007/978-3-030-92659-5_35.

Dörr, L., F. Brandt, M. Pouls, and A. Naumann (2020b). "An Image Processing Pipeline for Automated Packaging Structure Recognition". In: *Forum Bildverarbeitung*. Karlsruhe, Germany: KIT Scientific Publishing, p. 239.

A Appendix

A.1 Citations of Our Prior Publications

We previously published significant parts of this work in Dörr et al. 2020a, Dörr et al. 2020b and Dörr et al. 2021. The work at hand is more comprehensive than our previous publications but still draws from them in content, formulations and graphics. In this section, we specify in detail which paragraphs are complete citations, contain citations, or are partially adopted from previous publications.

Section	Element	Prior Publication	Citation Description
Abstract	§1	Dörr et al. 2020a	The first two sentences are a citation of our previous work.
Abstract	§2	Dörr et al. 2020a	The last sentence is very similar to our previous work, quantitative results differ due to a refactoring of the data set.

Section	Element	Prior Publication	Citation Description
2.1.1	§1	Dörr et al. 2020a	The first sentence is a citation of our previous work. Additional formulations may be similar, but the document at hand is more thorough and extensive than our previous work.
2.1.1	§4	Dörr et al. 2020a	The first sentence is a citation of our previous work.
2.1.2	§1	Dörr et al. 2020a	The second sentence is a citation of our previous work.
2.1.3	§1	Dörr et al. 2020a	Apart from the first sentence, the whole paragraph is a citation of our previous work.
2.1.3	§2	Dörr et al. 2020a	The paragraph is a citation of our previous work, except for the addition of the paragraph's second sentence.
2.2	§1	Dörr et al. 2020a	The first two sentences are a citation of our previous work.
2.2.1	§1	Dörr et al. 2020a	The paragraph is a citation of our previous work, except for the addition of the last sentence.
2.2.1	§2	Dörr et al. 2020a	The paragraph is a citation of our previous work.
2.2.1	§3	Dörr et al. 2020a	The paragraph is a citation of our previous work, except for the addition of the last sentence.
2.2.1	§4	Dörr et al. 2020a	The paragraph is a citation of our previous work.
2.2.2	§1	Dörr et al. 2020a	The paragraph is a citation of our previous work.
2.2.2	§2	Dörr et al. 2020a	The paragraph is a citation of our previous work.
2.2.2	Fig. 2.1	Dörr et al. 2020a	The figure was already published in our previous work.
2.2.3		Dörr et al. 2020a	The content is very similar to our previous publication, but the precise formulations differ.
2.2.4	§1	Dörr et al. 2020a	The paragraph is a citation of our previous work. Few formulations were slightly altered but the contents are in agreement.
2.2.4	§2	Dörr et al. 2020a	The paragraph is a citation of our previous work.
2.2.4	§3	Dörr et al. 2020a	The paragraph is a citation of our previous work.
2.2.4	§4	Dörr et al. 2020a	The paragraph is a citation of our previous work. Bullet point 3 was added for additional clarification.
2.2.2	Fig. 2.2	Dörr et al. 2020a	The figure was already published in our previous work.

Section	Element	Prior Publication	Citation Description
4.2		Dörr et al. 2020a	From the third sentence on, the paragraph is a citation of our previous work.
4.3		Dörr et al. 2020a	The beginning of the paragraph is a citation of our previous work. From line 12 onward, the content originates from this work.
4.3	Fig. 4.1	Dörr et al. 2020a	The figures were already published in our previous work.
4.3.1	§1	Dörr et al. 2020a	The first three sentences are citations of our previous work.
4.3.1	§2	Dörr et al. 2020a	The second sentence is an exact citation of our previous work. The remainder contains more detailed formulations as compared to the original publication.
4.3.1	§3	Dörr et al. 2020a	The first two sentences are a citation of our previous work.
4.3.2	§1	Dörr et al. 2020a	The beginning of the paragraph is a citation of our previous work. The last four sentences originate from this work.
4.3.2	§4	Dörr et al. 2020a	The last sentence is a citation of our previous work.
4.3.2	§6	Dörr et al. 2020a	The paragraph is a citation of our previous work. The second sentence was added for clarity.
4.3.3	§1	Dörr et al. 2020a	The paragraph is mostly a citation of our previous work. The enumeration was altered to allow for a more thorough explanation of the described method.
4.3.3	**1)**	Dörr et al. 2020a	The paragraph is a citation of our previous work.
4.3.3	**2)**, §1	Dörr et al. 2020a	The paragraph is mostly a citation of our previous work. The last two sentences were added for clarification.
4.3.3	**2)**, §2	Dörr et al. 2020a	The paragraph is a citation of our previous work.
4.3.3	Eq. (4.1)	Dörr et al. 2020a	The equation is a citation of our previous work.
4.3.3	**2)**, §5	Dörr et al. 2020a	The paragraph is a citation of our previous work.
4.3.3	**3)**	Dörr et al. 2020a	The first two sentences are a citation of our previous work, slight re-formulations were applied to achieve consistent wording throughout this work.
4.3.3	**4)**, §1	Dörr et al. 2020a	The last sentence is an exact citation of our previous work, the remainder of the paragraph, as well as Equations (4.6) and (4.7) were added as clarification.
4.3.3	Eq. (4.8), (4.9)	Dörr et al. 2020a	The equations are a citation of our previous work.

Section	Element	Prior Publication	Citation Description
4.3.3	**4)**, §3	Dörr et al. 2020a	The paragraph is mostly a citation of our previous work. Values of δ_1, δ_2 were adjusted in this work, reflecting changes in the data.
4.3.3	**4)**, §4	Dörr et al. 2020a	The paragraph is a citation of our previous work.
4.3.3	Eq. (4.10)	Dörr et al. 2020a	The equation is a citation of our previous work.
4.4.1		Dörr et al. 2020a	The paragraph is similar to parts of our previous work, but details differ due to changes in the dataset.
4.4.3	§1	Dörr et al. 2020a	The paragraph is a citation of our previous work.
4.5.1	§1	Dörr et al. 2020a	The paragraph is mostly a citation of our previous work. Details differ due to changes in the dataset and our method.
4.5.1	§2	Dörr et al. 2020a	The paragraph is a citation of our previous work.
4.5.1	§4	Dörr et al. 2020a	From sentence three on, the paragraph cites our previous work.
4.5.2	"**Inter-Unit Segmentation**"	Dörr et al. 2020a	The section is similar to our previous work but contains more extensive in its explanations.
4.5.2	"**Recognition Pipeline**", §1	Dörr et al. 2020a	The first sentence is a citation of our previous work.
4.5.2	"**Recognition Pipeline**", §2	Dörr et al. 2020a	The paragraph is a citation of our previous work.
4.5.2	Eq. (4.12)	Dörr et al. 2020a	The equation is taken from our previous work.
4.5.2	Eq. (4.13)	Dörr et al. 2020a	The equation is taken from our previous work.
4.5.2	"**Recognition Pipeline**", §3	Dörr et al. 2020a	The paragraph contains citations of our previous work. Exact results differ due to changes in the dataset. Additional explanations were added.
4.5.2	"**Manual Error Observations**"	Dörr et al. 2020a	The paragraph is similar to our previous work but more thorough and extensive.

Section	Element	Prior Publication	Citation Description
5.3	§3	Dörr et al. 2020b	The paragraph is mostly a citation of our previous work.
5.3	"**2. Line Detection**", §2	Dörr et al. 2020b	The paragraph's first sentence is a citation of our previous work.
5.3	"**2. Line Detection**", §4	Dörr et al. 2020b	The paragraph's first sentence is a citation of our previous work.
5.3	"**2. Line Detection**", §5	Dörr et al. 2020b	The paragraph's first sentence is a citation of our previous work.
5.3	"**3. Vanishing Point Estimation**", §1	Dörr et al. 2020b	The paragraph is mostly a citation of our previous work.
5.3	"**4. Side Boundary Estimation**"	Dörr et al. 2020b	The paragraph contains citations of our previous work (lines 1-3, lines 8-12), but further explanations were added.
5.3	Last §	Dörr et al. 2020b	The paragraph is a citation of our previous work.
5.3	Fig. 5.2, (c), (d)	Dörr et al. 2020b	The figures were already published in our previous work.
5.3	Fig. 5.3	Dörr et al. 2020b	The figure was already published in our previous work (colors were modified for better contrast).
5.3	Fig. 5.4	Dörr et al. 2020b	The figure was already published in our previous work (colors were modified for better contrast).
5.5	§1	Dörr et al. 2020b	Beginning with the paragraph's fifth sentence, it is a citation of our previous work.

Section	Element	Prior Publication	Citation Description
6.1	§4	Dörr et al. 2021	The paragraph is mostly a citation of our previous work.
6.2	§3	Dörr et al. 2021	The paragraph is mostly a citation of our previous work.
6.2	§6	Dörr et al. 2021	The paragraph is mostly a citation of our previous work.
6.4.1	Fig. 6.1	Dörr et al. 2021	The Figure was published in our previous work.
6.4.1	§1-3	Dörr et al. 2021	The paragraph is a citation of our previous work.
6.4.1	Fig. 6.2	Dörr et al. 2021	The Figure was published in our previous work.
6.4.1	**Backbone Network**	Dörr et al. 2021	The paragraphs are a citation of our previous work.
6.4.1	**Corner Detection and Corner Modules**	Dörr et al. 2021	The paragraphs are a citation of our previous work.
6.4.1	**Ground-truth**	Dörr et al. 2021	The paragraphs are a citation of our previous work.
6.4.1	**Loss Function**	Dörr et al. 2021	The paragraphs are a citation of our previous work.
6.4.1	Fig. 6.3	Dörr et al. 2021	The Figure was published in our previous work.
6.4.2		Dörr et al. 2021	The subsection is a citation of our previous work.
6.5	§1	Dörr et al. 2021	The paragraph is a citation of our previous work.
6.5.1	§1	Dörr et al. 2021	The paragraph is a citation of our previous work.
6.5.1	**Setup**	Dörr et al. 2021	The paragraph is a citation of our previous work.
6.5.1	**Training Details**, §1-2	Dörr et al. 2021	The paragraph's are mostly a citation of our previous work.
6.5.1	**Standard Metric Results**	Dörr et al. 2021	The paragraphs are mostly a citation of our previous work. Quantitative evaluation values differ slightly due to a refactoring of the data set.
6.5.1	**Use Case Specific Results**, §1	Dörr et al. 2021	The paragraph is mostly a citation of our previous work.
6.5.2	§1	Dörr et al. 2021	The paragraph is mostly a citation of our previous work.
6.5.2	§1	Dörr et al. 2021	The paragraph is mostly a citation of our previous work.
6.5.2	§3	Dörr et al. 2021	The paragraph is a citation of our previous work.
6.5.2	§4	Dörr et al. 2021	The paragraph is similar to our previous work, adaptations to the method (Eq. 6.14) were made.
6.5.2	§5	Dörr et al. 2021	The paragraph is a citation of our previous work.
6.5.2	§6	Dörr et al. 2021	The paragraph's first sentence is mostly a citation of our previous work.