TUM School of Computation, Information and Technology
Technical University of Munich

Bachelor's Thesis in Robotics, Cognition, Intelligence

Monocular Roadside 3D Perception based on Instance Segmentation

Monokulare 3D-Wahrnehmung mit Hilfe von Infrastruktursensorik basierend auf Instanzsegmentierung

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Walter Zimmer, M.Sc.

Author Bach Ngoc Doan

Date February 15, 2024 in Munich

Disclaimer

I confirm that this Bachelor's Thesis is my own work and I have documented all sources and material used.

Munich, February 15, 2024

(Bach Ngoc Doan)

Abstract

As part of the AUTOtech.agil project, this thesis aims to enhance the 2D detector node within the Providentia Mono3D system by leveraging instance segmentation models, thereby improving the system's capacity to predict real-world 3D object locations and shapes. Addressing the absence of segmentation labels in the TUM Traffic Dataset, we extend a subset with both modal and amodal segmentation annotations. To achieve this, we propose a 2D annotation interpolation pipeline capable of interpolating annotations between consecutive frames. We employ the YOLOv8x segmentation model and extensively investigate the effectiveness of pre-training on various datasets including COCO, KINS, and nuImages, along with training on the annotated frames. The best-performing model achieves a 2D mAP@[.5:.95] of 75.90 and a 3D mAP@[.10] of 18.51, which is an 18.30% and 7.53% improvement over the current implementation utilizing YOLOv7. Additionally, we employ the amodal segmentation model C2F to extend to amodal detections and evaluate its impact on final 3D perception performance, which, contrary to expectations, exhibits a decline. Further experiments are conducted across nighttime and highway scenarios, accompanied by suggestions for future improvements such as extending the training dataset on different scenarios for better model generalization.

Zusammenfassung

Im Rahmen des AUTOtech.agil-Projekts zielt diese Arbeit darauf ab, den 2D Detektor des Providentia Mono3D-Systems durch die Verwendung von Instanzsegmentierungsmodellen zu verbessern, um die 3D-Wahrnehmungsfähigkeit des realen Systems zu verbessern. Um dem Fehlen an Segmentierungsannotationen im TUM Traffic Intersection Datensatz entgegenzuwirken, erweitern wir eine Teilsatz um modale und amodale Segmentierungsmasken. Hierfür schlagen wir eine 2D-Annotationsinterpolationspipeline vor, die in der Lage ist, Annotationen zwischen aufeinanderfolgenden Frames zu interpolieren. Wir verwenden das YOLOv8x Instanzsegmentierungsmodell und untersuchen die Wirksamkeit von Vortrainings auf verschiedenen Datensätzen, darunter COCO, KINS und nuImages, sowie das Training auf annotierten Frames. Das beste Modell erreicht eine 2D mAP@[.5:.95] von 75.90 und eine 3D mAP@[.10] von 18.51, eine Verbesserung von 18,30% und 7,53% gegenüber der aktuellen Implementierung mit YOLOv7. Darüber hinaus verwenden wir das amodale Segmentierungsmodell C2F, um die sichtbaren Detektionen auf amodale Detektionen auszudehnen, und evaluieren dessen Auswirkungen auf die endgültige 3D- Wahrnehmungsleistung, die entgegen den Erwartungen einen Rückgang zeigt. Weitere Experimente werden in Nacht- und Autobahnszenarien durchgeführt, begleitet von Vorschlägen für zukünftige Verbesserungen, wie die Erweiterung des Trainingsdatensatzes auf verschiedene Szenarien für eine bessere Modellverallgemeinerung.

Acknowledgement

I would like to express my sincere thanks to my thesis advisor, Walter Zimmer, for his guidance throughout the literature research process, his support in helping me understand the existing toolchain and his valuable feedback on my work.

Contents

Chapter 1

Introduction

1.1 AUTOtech.agil

Autonomous driving has become a crucial topic of research and application in modern transportation. With the development of Artificial Intelligence (AI) and Machine Learning (ML), the integration of automated driving technologies has become vital, bringing in numerous benefits such as effectively reducing accident rates, minimizing labor costs, and promoting more efficient driving behaviors, serving as a significant tool in combating greenhouse gas emissions. While onboard Advanced Driver Assistance Systems (ADAS) have proven invaluable, these systems can only perceive the immediate environment around the vehicle, leading to blind spots and areas with limited coverage. As a result, they may fail to provide a holistic overview of the entire traffic scene, hindering their effectiveness in certain scenarios. Offboard solutions, from a road-side perspective, present a promising avenue to overcome the limitations of onboard sensors. These solutions provide real-time data beyond the field of view of onboard sensors, offering a more comprehensive and global perspective. Simulated in real-time, these solutions enable road users to make farsighted decisions, such as lane recommendations, accident warnings, and collision avoidance, utilizing both real-time information and historical data.

The Providentia project focused its research precisely on this point. Launched in 2017, the project achieved the development of an intelligent infrastructure system on the A9 highway and created a complete digital twin to enhance traffic safety and efficiency by the end of 2019. Two measurement stations (S40 and S50 in Figure 1.1), equipped with high-resolution cameras and radar systems, were mounted on overhead gantry bridges along the A9 highway near Garching. The sensor data is transmitted wirelessly via 5G, radio techniques that enable a reliable and fast connection between vehicles and intelligent infrastructure. Artificial intelligence is used to identify vehicle types and classes, and finally, the information from all the different sensors is fused. Moreover, an autonomous vehicle was also developed that can use information from the digital twin to change lanes on the highway independently and slow down to avoid traffic jams or accidents.

While highways provide more stable driving conditions for autonomous vehicles, residential areas pose a greater challenge due to their complexity. The transition from highway-centric solutions to residential areas requires innovative approaches and technologies to address the complex and dynamic nature of urban environments. Since 2020, the Providentia project has evolved into Providentia++, led by the Chair of Robotics, Artificial Intelligence, and Real-time Systems at the Technical University of Munich's Department of Informatics. The goal is to extend the deployment beyond highways into encompass residential areas. The addition of Light Detection and Ranging (LiDAR) sensors, along with the existing perception methods, provides redundant road coverage with overlapping fields of view, accurate

Figure 1.1: Overview of the test bed Providentia++ taken from [Cre+22]. This test bed spans the autobahn A9 and the highway B471 near Munich, covering a total length of 3.5 km

calibration, and robust detection and data fusion algorithms. The test stretch was expanded into the residential area, enabling the survey of intersections, traffic circles, bus stations, and other urban situations. The complete overview of the Providentia++ test bed is shown in Figure 1.1

This work is carried out in the scope of the AUTOTech.agil project funded by the Federal Ministry of Education and Research of Germany, which is a continuation of the Prodentia++ project.

1.2 Motivation

The creation of a digital twin that accurately reflects reality requires precise 3D detection. This crucial task involves building a reliable virtual representation capable of mapping the position, orientation, and dimensions of traffic participants from the real world to the simulator. Previous works in the Providentia project have proposed a Monocular 3D Object Detection (Mono3D) system to generate a three-dimensional digital twin of visible road users from the two-dimensional RGB camera input frames.

The current system uses YOLOv7 for object detection and has several limitations, such as the inability to classify some classes in the dataset like TRAILER, VAN, EMERGENCY _VEHICLE, and OTHER. Additionally, it struggles with detecting large or heavily occluded objects, which could impact the final 3D perception and automated driving safety. This thesis aims to enhance the Providentia Mono3D object perception pipeline by predicting real-world 3D object locations and 3D shapes from segmentation masks.

To achieve this goal, we aim to improve the system's instance segmentation detector. A comprehensive literature review is conducted to identify promising instance segmentation (IS) and amodal instance segmentation (AIS) methods. On the one hand, our approach involves exploring state-of-the-art (SOTA) basic instance segmentation models, outperforming YOLOv7 on the same task. On the other hand, we explore SOTA amodal instance segmenta-

tion models that can segment the invisible parts of an object in addition to the visible. We want to investigate whether leveraging amodal segmentation can enhance the system's object detection capabilities, particularly in complex traffic scenarios with significant occlusion.

Moreover, we also explore common datasets with IS and AIS annotations in the domain of autonomous driving for pre-training purposes. Different models are then pre-trained on public autonomous driving datasets and fine-tuned on our TUM Traffic Intersection Extended dataset. Since there are no labeled instance masks in the TUM Traffic Intersection dataset, a subset of frames is annotated for fine-tuning. This enables the models to classify all our categories and learn to predict from our camera settings for improved performance.

Subsequently, a 2D detector node exploiting these newly trained models is integrated into the existing toolchain. The models are optimized for inference speed using TensorRT. Performance evaluations are conducted, comparing the models against each other and against YOLOv7 in terms of 2D instance segmentation and 3D object detection, utilizing metrics such as Average Precision (AP) and mean Intersection over Union (mIoU).

In essence, this work represents a valuable contribution to the AUTOtech.agil project, effectively addressing the shortcomings of the existing system and potentially advancing the safety and efficacy of autonomous driving technologies.

1.3 Contributions

This thesis presents the following contributions:

1. We explore various state-of-the-art instance and amodal instance segmentation models for inference on the TUM Traffic Intersection dataset. Among these models, we choose the instance segmentation YOLOv8x and the amodal instance segmentation C2F model for further analysis.

2. We propose an instance segmentation labeling pipeline with a method for 2D annotation interpolation across consecutive frames of an image sequence (video). We use this pipeline to extend the TUM Traffic Intersection dataset with visible and full instance segmentation annotations.

3. We define an extended OpenLABEL annotation structure tailored for our TUM Traffic dataset to include additional modal and amodal segmentation labels. We also extend the TUM Traffic development kit (TUMTraf dev-kit) with label converters facilitating conversion between YOLO, COCO, and our TUM Traffic Dataset OpenLABEL annotation formats.

4. We leverage YOLOv8x and C2F models pre-trained on COCO and KINS datasets and fine-tuned them on the instance segmentation masks extended frames of the TUM Traffic Intersection dataset. Additionally, we train YOLOv8x from scratch on the extended TUM Traffic Intersection dataset as well as nuImages.

5. A total of around seven experiments are conducted on the trained models, utilizing different data sequences under various settings to thoroughly evaluate their 2D and 3D perception performance. Notably, the best-performing model for the TUM Traffic Intersection test sequence, across all settings, demonstrates a remarkable 3D mAP@[.10] improvement of 17.79% compared to YOLOv7, the model currently employed in the Providentia Mono3D system.

6. We export trained YOLOv8x PyTorch models to TensorRT, resulting in a significant acceleration of inference speed by approximately 2.8 times, surpassing the YOLOv7 TensorRT model's speed by around 2.3 times. Furthermore, we integrate a 2D object detector node, exploiting YOLOv8, into the existing toolchain.

1.4 Structure of the Thesis

The thesis is structured as follows: in Chapter 1, a comprehensive introduction to the AUTOtech.agil project and its related predecessors is presented, along with the motivation and main contribution of this thesis. Chapter 2 provides an extensive background introduction. This chapter covers the Robot Operating System 1 (ROS1), well-known autonomous driving datasets with instance and amodal instance segmentation annotations, common label formats for object detection tasks, and the labeling tool CVAT. Chapter 3 offers an overview of related works, including the Providentia Mono3D System, followed by an introduction to the state-of-the-art instance segmentation models. Chapter 4 describes in detail the solution approach, including inferencing using different pre-trained models, extending segmentation annotations for the TUM Traffic dataset, training and fine-tuning models, exporting to TensorRT, and integrating into the existing toolchain. Chapter 5 provides an extensive evaluation, including 2D and 3D quantitative analysis, qualitative analysis, as well as an analysis on inference speed, comparing the trained models against the baseline YOLOv7 currently used in the Providentia Mono3D system. Chapter 6 documents five additional experiments on different scenes, including intersection, nighttime, and highway scenarios, with different settings, such as time-shifted ground truth and 3D tracking, to further analyze the generalizability and additional improvements of the proposed models. Finally, Chapter 6 and Chapter 7 conclude the thesis and suggest avenues for future work.

Chapter 2

Background

This chapter provides essential background information. Firstly, Section 2.1 introduces the Robot Operating System 1 (ROS1), a middlewares utilized in the Providentia Mono3D system. Following this, Section 2.2 offers a comprehensive overview of common instance segmentation datasets for autonomous driving, including a detailed description of our TUM Traffic Intersection Dataset (Section 2.2.4). Additionally, Section 2.3 introduces some popular annotation formats in the field of object detection, describing their labeling structures and conventions. This section is followed by an exploration of the Computer Vision Annotation Tool (CVAT) in Section 2.4, which plays an important role in annotation during our research.

2.1 Robot Operating System 1 (ROS1)

The Robot Operating System, abbreviated as ROS, is not an actual operating system but a collection of software libraries and tools that support the development and reuse of code across different robotic applications. Moreover, ROS's usefulness extends beyond robots, as most of the provided tools focus on working with peripheral hardware. ROS is open-source and maintained by many people. There are two versions, ROS1 and ROS2. Since our system exploits ROS1, in this section, we introduce the concept of ROS1.

The way ROS1 operates is relatively straightforward. A ROS system consists of several independent processes called *ROS nodes*. ROS nodes can be located on different systems and can even have different architectures, making ROS genuinely flexible and adaptable to user needs. Communication is established between nodes by *ROS Master*. The *ROS Master* provides naming and registration services to the nodes, allowing them to find and communicate with each other.

There are two main ways of communication in ROS. The first way is communication through *topic*. *Nodes* can *publish* or *subscribe* to a *topic*, then send or receive ROS messages of the topic type. This is called a publisher and subscriber relationship. The second way is communication through *Service*. *Nodes* provides requestable *services*, other nodes can then send a *request* to the service to receive a *response*. The structure and type of messages between nodes are defined individually for each *topic* and *service*.

2.2 Autonomous Driving Segmentation Datasets

Autonomous driving segmentation datasets play a crucial role in advancing computer vision algorithms in this domain. Segmentation datasets can be exploited for various tasks, including object detection, instance segmentation, and semantic segmentation. This section

provides an extensive description of the commonly used instance segmentation datasets in the autonomous driving domain, as illustrated in Table 2.1. Specifically, the three datasets COCO, KINS, and nuImages utilized in this thesis are examined in more detail. Afterward, the TUM Traffic Intersection Dataset, which is utilized to fine-tune the models, is described.

Name	Classes	Annotations	Data Split
COCO	91 object classes with 80 used for instance segmentation	2d bounding box, instance- and panoptic- segmentation, captioning, keypoint, dense pose	Train: 118k, Val: 5k, Test: 41k
COCOA	same as COCO	COCO annotations plus semantic amodal segmentation	Train: 2476, Test: 1223
KINS	8 classes: pedestrian, cyclist, person-siting, car, tram, truck, van, misc	amodal instance mask, semantic label, and relative occlusion order	Train: 7474, Test: 7517
nuImages	23 classes. The supercategory Vehicle contains: Bus, Car, Construction, Emergency, Motorcycle, Trailer, Truck	2d bounding box, instance mask	Train: 67k, Val: 16k, Test: 10k
Cityscapes [Cor+16]	30 classes grouped into eight categories (flat surfaces, humans, vehicles, constructions, objects, nature, sky, and void)	semantic, instance-wise, and dense pixel annotation	Train: 2975, Val: 500, Test: 1525
Waymo Open Perception [Sun+20]	4 object classes: Vehicles, Pedestrians, Cyclists, Signs	2D and 3D bounding boxes, key point, 3D semantic segmentation, 2D video panoptic segmentation	390k frames, Train and Val: 1000 scenes, Test: 150 scenes

Table 2.1: This table provides an overview of common autonomous driving segmentation datasets including the dataset name, the object categories defined, the annotation types, and the number of samples contained in each data split.

2.2.1 COCO

Microsoft Common Objects in Context (COCO) [Lin+14] is a large-scale dataset for image classification, object detection, semantic segmentation, and instance segmentation. The dataset comprises images taken from daily life scenes with varying resolutions. It contains 2.5 million labeled instances in 328k images, covering 91 object classes, of which 80 are used for instance segmentation. Based on the 2017 COCO split, the data split is 118k training images, 5k validation images, and 41k testing images. Along with bounding boxes and per-instance segmentation masks, this dataset also provides natural language descriptions (captioning), key points, stuff image segmentation, panoptic segmentation, and dense pose annotation.

This dataset is extended to an amodal dataset in 2017 in [Zhu+17]. The extended dataset is named COCOA and includes 2476 images in the training set and 1223 images in the testing set, with additional semantic amodal segmentation annotations.

2.2.2 KINS

KITTI INStance segmentation dataset (KINS) [Qi+19] is a large-scale street scene amodal instance dataset, which is built upon Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) [GLU12]. The dataset consists of 7474 training images and 7517 testing images with eight categories: pedestrian, cyclist, person-siting, car, tram, truck, van, and misc (ambiguous vehicles). The annotations include amodal instance masks, semantic labels, and relative occlusion orders, from which initial instance masks can be easily inferred. On average, each image has 12.53 labeled instances, and each object polygon consists of 33.70 points. 53.6% are partially occluded, and the average occlusion ratio is 31.7%. The annotation format follows the COCO style, which will be described in Section 2.3.3

2.2.3 nuImages

Following the success of the nuScenes [Cae+20] dataset, in August 2020, Motional released nuImages [Cae+] with additional 2D annotations from a much larger pool of data. From a total of 1.2 million autonomous driving camera images, active learning techniques were employed to select approximately 75% challenging images according to the uncertainty of an image-based object detector. Rare classes, like bicycles, were given special focus. The remaining 25% of the images were uniformly sampled to ensure a representative dataset and avoid strong bias. After careful review, some images were discarded due to camera artifacts, because they were too dark, or because they showed pedestrians' faces. This careful curation of a dataset resulted in a diverse dataset regarding class distribution, spatiotemporal distribution, and weather and lighting conditions. The annotated images encompass rain, snow, and nighttime, which are crucial for autonomous driving applications.

The final dataset contains 93k labeled images with instance masks and 2D boxes, which results in around 800k foreground objects and 100k semantic segmentation masks. Additionally, each annotated image is accompanied by six past and six future unlabeled camera images at 2 Hz, leading to 93k video clips with 13 frames spaced out at 2 Hz.

2.2.4 TUMTraf Intersection Dataset

The TUM Traffic Intersection Dataset, previously known as the "A9 Intersection Dataset" [Zim+23], was first published in June 2023. This dataset is the second release (R2) of the A9 dataset [Cre+22], and it comprises 4.8k synchronized images and LiDAR point clouds with over 57.4k manually labeled 3D bounding boxes. The dataset was captured using two roadside cameras and two LiDAR mounted on intersecting gantry bridges. The data labels follow the OpenLABEL format [Hag20], which is discussed in detail in Section 2.3.1.

This dataset consists of four continuous camera and labeled LiDAR scenes captured at a frame rate of 10 Hz. Each scene is recorded from two camera perspectives, denoted as south1 and south2, yielding a total of eight frame sequences. Scenes S1 and S2 each consist of 600 frames, depicting a daytime scenario at dusk. Scene S3 consists of 2400 frames captured during daytime with sunshine. Scene S4 contains an additional 1200 frames recorded at night and during heavy rain [Cre+22].

It differentiates ten categories: CAR, BUS, TRUCK, TRAILER, VAN, MOTORCYCLE, BICYCLE, PEDESTRIAN, EMERGENCY_VEHICLE, and OTHER. The dominant category is CAR, followed by TRUCKS, TRAILER, VAN, and PEDESTRIAN by approximately the same order of magnitude. The remaining five classes are present in slightly smaller numbers. Of all the objects in the dataset, 78.2% were classified as NOT OCCLUDED, 16.1% as PARTIALLY OCCLUDED, 0.8% as MOSTLY OCCLUDED, and 4.9% UNKNOWN.

2.3 Label Format

Each model and each dataset has its own specific requirements regarding data structure and label format. Understanding these aspects is critical for training deep learning models and managing datasets. This section introduces three well-known annotation formats used in this work, including OpenLABEL, YOLO, and COCO Format.

2.3.1 OpenLABEL Format

The Association for Standardization of Automation and Measuring Systems (ASAM) with Deepen AI released the OpenLABEL [Hag20], a standard designed to specify an annotation format that is flexible enough to support the development of automated driving features while ensuring interoperability among different systems and providers. The annotation structure of OpenLABEL is defined using JSON schema, and annotation files are stored in .json format.

The TUMTraf dataset has fully adopted this annotation format, with each frame having one .json label file. The general structure of one point cloud 3D label file is shown in Listing 2.1. Each object has one unique object identifier. The object data is a collection of "name," "type," which defines the category of the object, and "cuboid," which describes the 3D bounding box.

Listing 2.1: Illutration of OpenLABEL Annotation JSON Structure

```
{"openlabel:" {
     "metadata": {...},
     "coordinate_system": [...],
     "frames": {
       "frame_id" :{
         "frame_properties" :{...},
         "objects" :{
           "object1_id" :{
             "object_data": {
               "name" :str,
               "type" :str,
               "cuboid" :{
                 "name": "shape3D",
                 "val": [...],
                 "attributes": {...}
               }
             }
           },
           "object2_id" :{...}
           "object3_id" :{...}
           ...
         }
       }
     }
   }
}
```

2.3.2 YOLO Format

The YOLO label format is tailored for YOLO (You Only Look Once) models. For each frame, there is one annotation text file (.txt). Each row in the text file corresponds to one object instance in the image. The format for one object bounding box is

```
<class-index> <x_center> <y_center> <width> <height>
```

whereas for the bounding coordinates of the object's segmentation mask is

```
<class-index> <x1> <y1> <x2> <y2> ...  <xn> <yn>
```

The bounding box coordinates and segmentation mask points are normalized to the range [0, 1]. While YOLO's compact label format simplifies annotation storage, users must ensure consistency and distinguish between bounding box and mask annotations.

2.3.3 COCO Format

Common Objects in Context (COCO) is a well-known dataset format used by Microsoft, Google, and Facebook [Hag14]. This format also exploits the JSON structure. In this format, the labels of all objects from all frames will all be included in one .json file, a collection of "info," "licenses," "images," "annotations," and "categories".

```
{
        "info"        : info,
        "images"      : [image],
        "annotations" : [annotation],
        "licenses"    : [license],
}
```

The "annotations" section contains a list of every individual object annotation. The structure of one object detection annotation is shown in Listing 2.2. Each annotation includes information such as the category label, bounding box coordinates, segmentation mask, and additional metadata. The segmentation mask can be represented either as a run-length-encoded bit mask or as a list of polygon contour points, providing flexibility in representing object boundaries. The 'iscrowd' field specifies whether the segmentation is for a single object or for a cluster of objects, while the 'area' field indicates the area of the object mask, measured in pixels.

Listing 2.2: Illutration of COCO Annotation JSON Structure

```
annotation  {
        "id"          : int,
        "image_id"    : int,
        "category_id" : int,
        "segmentation": RLE or [polygon],
        "area"        : float,
        "bbox"        : [x_top_left,y_top_left,width,height],
        "iscrowd"     : int
}
```

2.4 Computer Vision Annotation Tool (CVAT)

Computer Vision Annotation Tool [SMZ19] is a web-based, open-source annotation tool initially developed by Intel and now maintained by OpenCV. CVAT allows users to annotate images with various types of shapes, including boxes, polygons, polylines, and points. It supports importing and exporting annotations in multiple formats, such as YOLO, MS COCO, and KITTI. One of its standout features is its ability to perform automatic labeling using serverless deep learning models like YOLO, RCNN, Face Detection, and Segment Anything. This feature can significantly speed up the labeling process for large datasets.

Chapter 3

Related Work

This chapter provides an overview of the related literature. Firstly, Section 3.1 provides a concise summary of prior works on the Providentia Mono3D System. Following this, Section 3.2 delves into the task of instance segmentation, introducing state-of-the-art deep learning models utilized for this task. These models are presented within a hierarchical taxonomy, offering insight into their categorization and methodologies. In particular, two state-of-the-art (SOTA) instance and amodal instance segmentation models, YOLOv8 and Coarse-to-Fine Segmentation (C2F-Seg), which are exploited in this work, are introduced in deeper detail in this section.

3.1 Providentia Mono3D System

The Providentia Monocular 3D Perception task (Mono3D) describes the process of detecting three-dimensional objects from a single two-dimensional RGB camera output frame. This Mono3D toolchain is described in the works "Real-Time Monocular 3D Object Detection to Support Autonomous Driving" by Leon Blumenthal [Blu22] and "Monocular 3D Object Detection Using HD Maps" by Joseph Birkner [Bir23]. In [Blu22], the Providentia Mono3D detecter splits the detection task into two separate steps: a 2D instance segmentation step and a 3D lifting step. This so-called two-stage detection model is adapted and refined in the thesis of Joseph [Bir23]. The first work focused on highway scenarios, where the yaw value (the direction of travel) is fixed. The second work addressed and generalized to the more urban scenarios where the yaw value is variable. The proposed approach was then evaluated on the TUM Traffic Intersection Dataset. This dataset has been covered in detail in Section 2.2.4.

For each object, Mono3D estimates the basic 3D perception values, including birds-eye-view (BEV) positions X/Y, BEV size L/W, object height H, and headling angle (yaw) θ. The elevation of the object over the road surface Z, pitch ϕ, and roll γ values are set to 0 always. Along with these basic values, Mono3D also provides an identifier for the object across multiple frames I, the planar speed $\delta X/\delta Y/\delta Z$, and category C. These make up a total of fourteen dimensions in the output, as shown in Table 3.1.

The Mono3D detection flow consists of three process nodes as shown in Figure 3.1: a camera driver node, a 2D instance segmentation node, and a 3D detection node. The 2D instance segmentation node continuously receives full-resolution $1920x1200$ RGB frames from the camera driver node. This communication exploits the Enhanced Communication Abstraction Layer (eCAL) [23] shared memory as middleware for throughput and high performance. The second node exploits the YOLOv7 [WBL23] instance segmentation model for detection and publishes an array of detections containing a confidence score, a category, a 2D

Variable	Description
X/Y	Position along the longitudinal/lateral axes
Z	The elevation of the object over the road surface ($= 0$)
$L/W/H$	Length/Width/Height of the object
$\gamma/\phi/\theta$	Roll/Pitch/Yaw angles, yaw determines travel direction ($\gamma = 0, \phi = 0$)
C	category
I	Object Identifier across multiple frames
$\delta X/\delta Y/\delta Z$	Derivatives of the position variables := speed

Table 3.1: The fourteen-dimensional output of Mono3D per object

bounding box, and a per-pixel instance mask to downstream processes via Robot Operating System (ROS) [Qui+09]. The 3D detection node subscribes to the instance segmentation node, receives the 2D instance detections, and starts with the so-called 3D lifting stage. This stage adapts and develops the L-Shape Fitting Method for Vehicle Pose Detection from LiDAR [Zha+17] using HD map and tracking algorithm to stabilize the 3D detections and speed estimations. The currently chosen tracking algorithm is Simple Online and Realtime Tracking (SORT) [Bew+16], a lightweight tracking algorithm specifically designed for real-time applications. Finally, the final fourteen-dimensional output of the entire flow is published via ROS to downstream applications, such as sensor fusion, visualization, or autonomous vehicles.

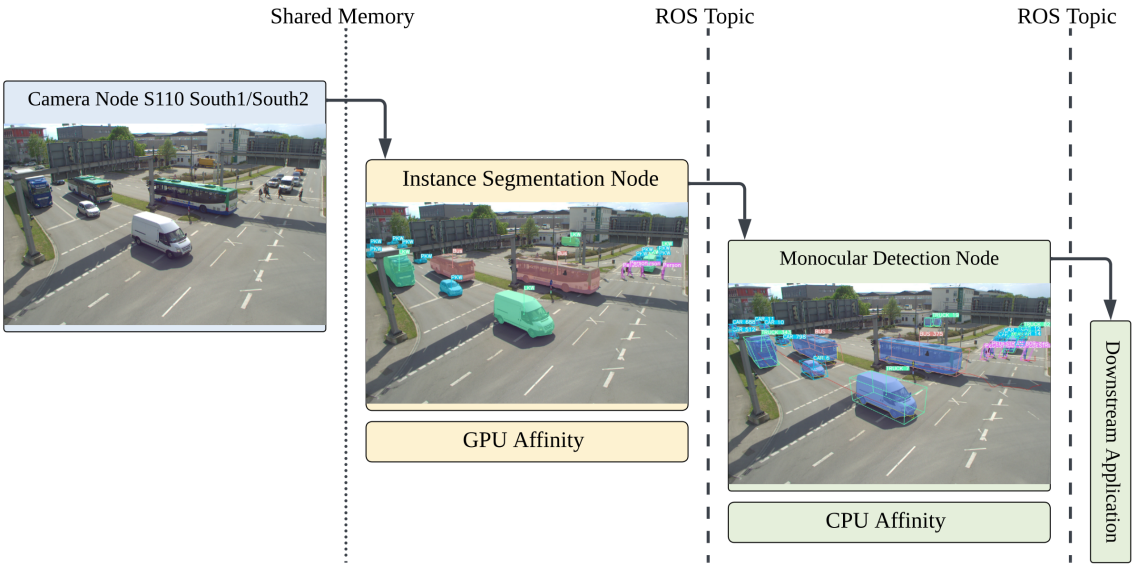


Figure 3.1: Providentia Mono3D system architecture from [Bir23]. The camera driver publishes frames to Shared Memory, which are picked up from the instance segmentation node. The 2D detections from each frame are published to the Monocular Detection Node through ROS. The final 3D perception outputs are then published via ROS to downstream applications.

This separation of the detection pipeline into multiple concurrent processing nodes provides the flexibility to evaluate and optimize each stage independently. Our work focuses on enhancing the initial stage of the 2D detector.

3.2 Instance Segmentation models

Instance segmentation (IS) is a fundamental computer vision task that strives to detect, classify, and predict per-pixel segmentation masks of every individual object within an image. Fully-supervised instance segmentation models fall into three primary categories: single-stage, two-stage, and multi-stage object detection models as illustrated in Figure 3.2 [GBK22].

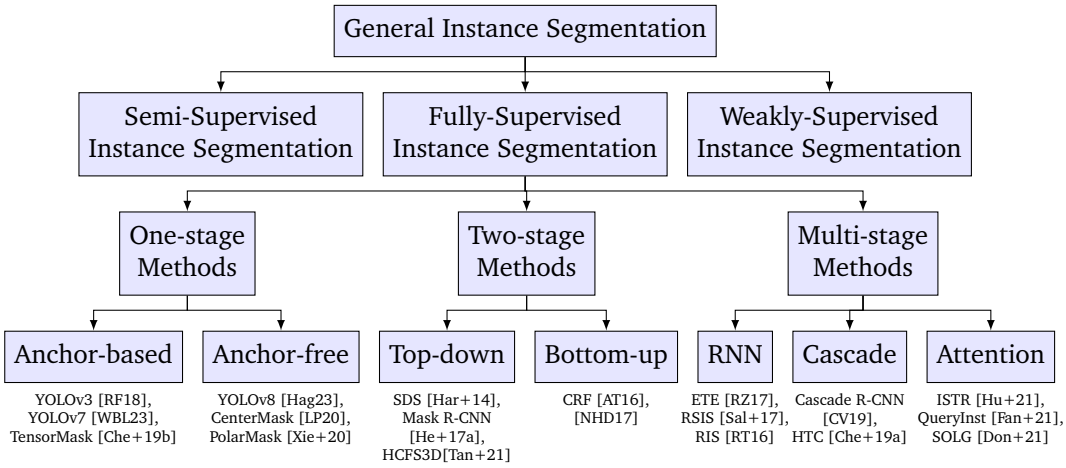


Figure 3.2: Taxonomy of Instance Segmentation Methods taken from [GBK22]

3.2.1 Two-stage object detection models

Two-stage object detection models, leveraging region-based convolutional neural networks (R-CNNs), employ two rounds of the input image. The first round generates a series of proposals or potential object locations. In the second round, these proposals are refined to make conclusive predictions, including mask estimation and classification.

The two-stage detection framework has become a classical model in both 2D and 3D object detection. Two-stage instance segmentation methods can be further divided into top-down and bottom-up methods. Top-down methods, such as SDS [Har+14], Mask R-CNN [He+17a], and HCFS3D [Tan+21], first predict regions of interest and then perform segmentation within each bounding box. In contrast, bottom-up methods such as CRF [AT16] and [NHD17] first map each pixel as a vector embedding and then group them into different instances through clustering methods. Compared to one-stage object detection, this class of models provides more precise detections. However, they are more memory and computationally expensive, which makes them unsuitable for real-time applications [Wan+22]. Moreover, the sequential execution of detection and segmentation does not comprehensively consider their correlation. [GBK22].

3.2.2 Multi-stage object detection models

To fully exploit useful reciprocal information about the relationship between detection and segmentation, many researchers are turning to multi-stage models. These models typically have either a multi-stage cascade structure or are based on recurrent neural networks (RNNs) or self-attention mechanisms.

- The multi-stage cascade structure (Cascade R-CNN [CV19], HTC [Che+19a]) performs instance segmentation stage by stage.
- The RNN-based methods (ETE [RZ17], RSIS [Sal+17], RIS [RT16]) segment instances one by one.
- The self-attention-based instance segmentation methods (ISTR [Hu+21], QueryInst [Fan+21], SOLG [Don+21]) refine query boxes and mask predictions with the recurrent refinement strategy [GBK22].

Compared with two-stage methods, the multi-stage instance segmentation methods can achieve better performances. However, these methods are even more computationally expensive. [GBK22].

3.2.3 Single-stage object detection models

Considering the limitations of the two classes of models mentioned above, it is reasonable to perform both detection and segmentation in a single architecture to comprehensively consider the relationship between them and speed up inference time. One-stage instance segmentation methods swiftly analyze the entire image and make predictions in just one go. Although these models may not be as accurate as the two-stage models and might struggle with detecting smaller objects, they are computationally efficient and have better generalization capabilities [Wan+22]. They are ideal for real-time detection in resource-limited settings.

Single-stage object detection models can further be categorized into anchor-based methods and anchor-free methods. Single-stage anchor-based instance segmentation methods such as YOLOv3 [RF18], YOLOv7 [WBL23] and TensorMask [Che+19b], are similar to the two-stage top-down methods, they first generate a set of candidate regions then segment each instance in their corresponding positive bounding box. Single-stage anchor-free instance segmentation methods such as YOLOv8 [Hag23], CenterMask [LP20], and PolarMask [Xie+20] locate instances directly in the pixel-level without anchors.

Since the Providentia Mono3D System currently utilizes YOLOv7 [WBL23] for instance segmentation and this work exploits YOLOv8, in the following, we will look deeper into the state-of-the-art You Only Look Once (YOLO) [Red+16] family. YOLO is an object detection and image segmentation system that has revolutionized the field of computer vision. YOLO is known for its small and simple architecture and fast inference speed and is thus suited for real-time object detection tasks. YOLO was first introduced in 2015, and since then, several new versions of the same model have been proposed. The general YOLO framework consists of three main components:

- **Backbone**: A convolutional neural network that mainly extracts essential image features.
- **Neck**: A collection of neural network layers that combine and mix features before passing them to the next stage for prediction.
- **Head**: Output layers that consume features from the neck and generate predictions. At the end of the process, non-maximum suppression (NMS) is used to filter out overlapping detections.

The YOLOv8 exploited in this work is the latest version of YOLO, featuring five variants based on the number of parameters, namely nano(n), small(s), medium(m), large(l), and

extra large(x) fig. 3.3. All variants can be used for object detection, segmentation, and classification. YOLOv8 incorporates several new features and improvements for enhanced performance, flexibility, and efficiency. The updates in YOLOv8 include the following:

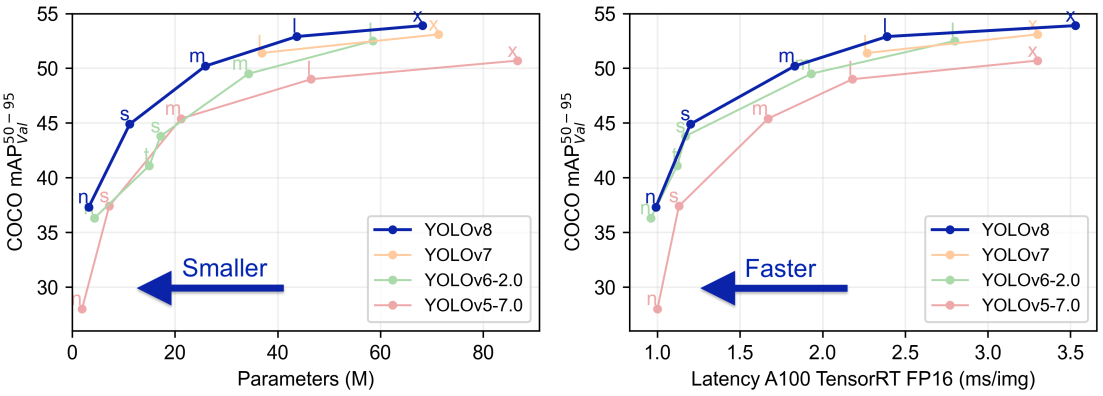


Figure 3.3: YOLO parameters and latency comparison from [Hag23]. Several other performance tests have shown that YOLOv8 outperforms YOLOv7 in terms of speed and accuracy.

1. The switch to anchor-free detection head: The head module switched from anchor-based to anchor-free and adapted the current mainstream decoupled structure, separating the classification and detection heads. Anchor-free models directly predict an object's center instead of the offset from a known anchor box. Anchor-free detection reduces the number of box predictions, which speeds up the Non-Maximum Suppression (NMS) post-processing. Experimental results in [Lv+23] show that with equivalent accuracy, YOLOv7 produces around three times more predicted boxes in comparison to YOLOv8.

2. The convolutional blocks undergo modifications to expedite the training process and improve gradient flow.

3. Exploitation of Mosaic augmentation: Mosaic augmentation is the process of combining four images into a single mosaic image. This is done by resizing the four images, stitching them together, and then taking a random cutout. This technique enhances the model's ability to learn objects in new locations, partial occlusion, and with greater variation in surrounding pixels. However, it has been shown that using Mosaic augmentation throughout the entire training regime may have an adverse effect on prediction accuracy. Thus, YOLOv8 applies Mosaic augmentation only during training and turns it off before the last ten epochs.

4. YOLOv8 offers several developer-convenience features, from an easy-to-use CLI to a well-structured Python package.

3.2.4 Amodal Instance Segmentation

Instance segmentation methods are restricted to detecting and segmenting the visible part of the objects, which can lead to poor performance in situations where objects are heavily occluded. Under such conditions, instances may be wholly missing, or the 2D bounding boxes and segments may be truncated, causing errors in downstream processes. In our case, these errors can affect the accuracy of 3D perception. To address the limitations of IS, amodal

instance segmentation (AIS) techniques aim to predict the complete shapes of objects, including both visible and invisible parts. The full mask of an object is referred to as an amodal mask, while the visible mask is known as a modal or inmodal mask.

In this related work study, several survey AIS papers were consulted. Some of these papers are worth highlighting for reference. The Bilayer Convolutional Network (BCNet) [KTT23] exploits a bilayer structure for occluding objects (occluders) and partially occluded instances (occludees). This approach naturally decouples the boundaries of both instances and considers the interaction between them during mask regression. AISFormer [Tra+22] enhances the extraction of the long-range dependency via transformer and explicitly models the complex coherence between occluder, visible, amodal, and invisible masks within an object's regions of interest by treating them as learnable queries. The self-supervised Amodal Video Object Segmentation (SaVos) [Yao+22] offers a solution for video amodal segmentation. SaVos leverages spatiotemporal consistency and dense object motion to explain away occlusion. Coarse-to-Fine Segmentation (C2F-Seg) [Gao+23] model consists of two modules: a transformer-based module for predicting coarse amodal masks and a CNN-based refinement module for obtaining fine amodal masks. The paper of this model is published in ICCV 2023. The authors of C2F-seg conducted a comparison with several state-of-the-art models, including PCNet [Zha+20], Mask R-CNN [He+17b], ORCNN [Fol+19], VRSP [Xia+21] and AISformer on the KINS and COCOA datasets and report the results in Table 3.2. This indicates that the C2F-Seg outperforms other state-of-the-art models on both datasets across average precision (AP) and mean intersection over union (mIoU) metrics.

Methods	KINS					COCOA				
	AP	AP@.5	AP@.75	mIoUfull	mIoUocc	AP	AP@.5	AP@.75	mIoUfull	mIoUocc
PCNet	29.1	51.8	29.6	78.02	38.14	–	–	–	76.91	20.34
Mask R-CNN	30.0	54.5	30.1	–	–	28.0	53.7	25.4	–	–
ORCNN	30.6	54.2	31.3	–	–	28.0	53.7	25.4	–	–
VRSP	32.1	55.4	33.3	80.70	47.33	35.4	56.0	38.7	78.98	22.92
AISformer	33.8	57.8	35.3	81.53	48.54	29.0	45.7	31.8	72.69	13.75
C2F-Seg	**36.5**	**58.2**	**37.0**	**82.22**	**53.60**	**36.6**	**57.0**	**38.5**	**80.28**	**27.71**

Table 3.2: C2F-Se performance comparison on the KINS and COCOA reported by [Gao+23]. The AP columns show AP@[.5:.95]

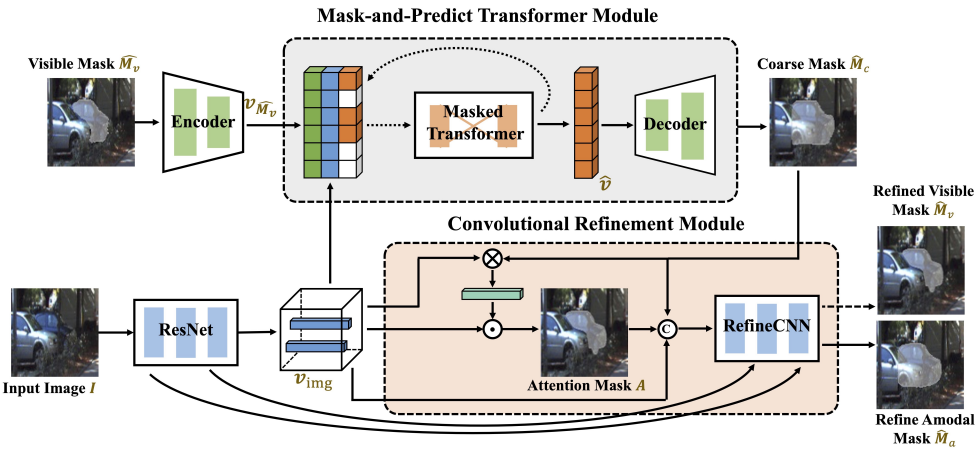


Figure 3.4: The architecture of C2F-Seg taken from [Gao+23]. The transformer module predicts coarse amodal masks from visual features and visible segments, while a convolutional refinement module refines the coarse predictions using visual features to generate final amodal segmentation mask predictions. Inference provides an estimate of the amodal mask based on input visible masks.

The following shows the Coarse-to-Fine Amodal Segmentation with Shape Prior (C2F-seg) model in more detail. C2F-Seg generates amodal segments progressively through two phases, as shown in Figure 3.4. The first coarse segmentation phase takes as inputs ResNet visual feature, vector-quantized visible segments, and ground-truth amodal segments masked in a high ratio. In this phase, a transformer is adopted and trained to reconstruct the masked tokens of the amodal segments. The reduction of learning space from pixel-level image space to low-dimension vector-quantized latent space makes the learning process easier and accelerates the inference process. The second refinement phase takes as inputs the coarse-predicted amodal segments from the first phase and the visual features. A semantic-inspired attention module is constructed as an initial stimulus, which gradually injects the visual features into the segments through convolution layers to finally arrive at a more precise amodal object segmentation. The learning of visible masks is used as an auxiliary task in training; inference only provides an estimate of the amodal mask based on the input of visible masks.

The C2F_seg framework used for experiments in [Gao+23] was implemented on the PyTorch platform and utilized pre-detected visible detections by AISFormer [Tra+22]. The model enlarges bounding boxes of visible regions twice and uses them to crop the image and mask inputs. The inputs are then all resized to 256×256, and various data augmentation techniques, including morphology dilation, erosion, and Gaussian blur, are applied.

Chapter 4

Solution Approach

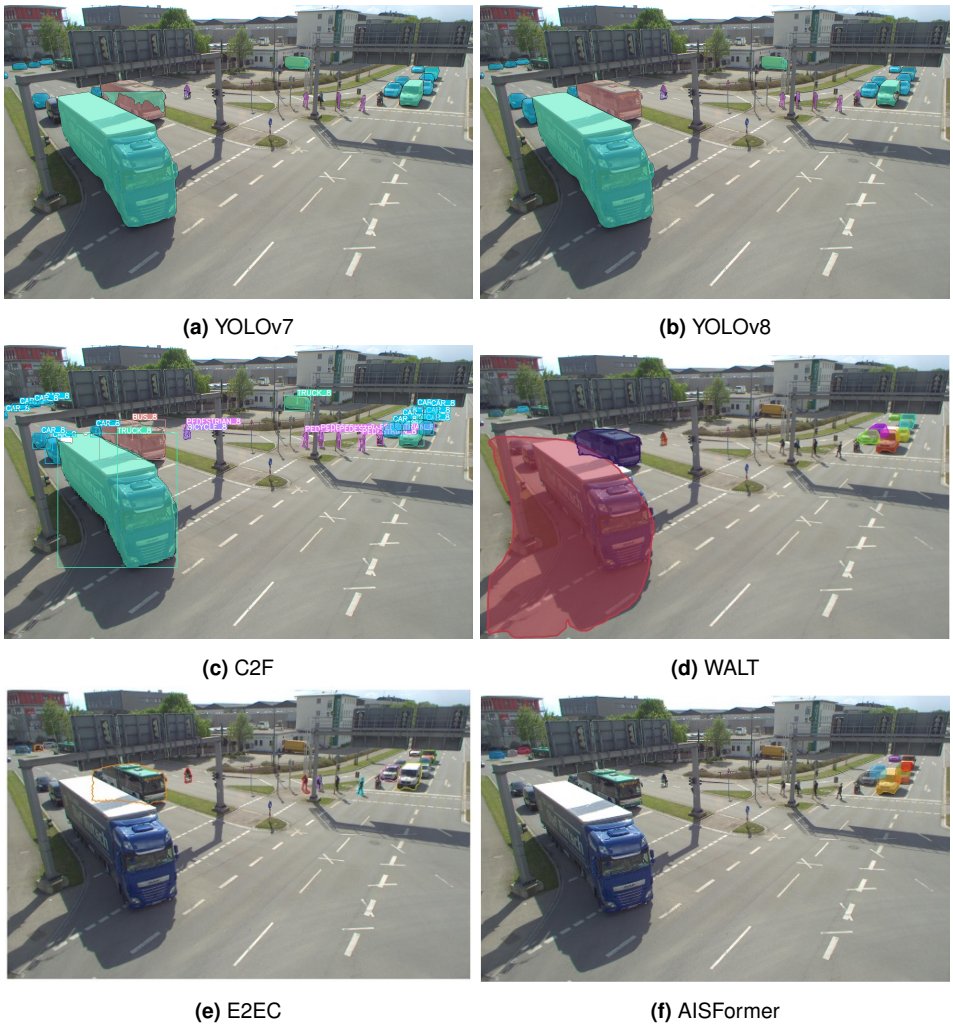
The 2D object detector of Providentia Mono3D currently relies on YOLOv7 pre-trained on COCO dataset. It downscales the incoming frames from 1920x1200 to 1280x1280 or even 640x640 (with padding) to speed up the process to meet the system's real-time requirement. However, this downscaling sacrifices detection accuracy, particularly for small objects farther away from the camera. Moreover, the COCO dataset annotates eighty classes, but only six classes overlap with our TUM Traffic dataset. The Providentia 2D detector filters six COCO sub-classes: CAR, BUS, TRUCK, MOTORCYCLE, BICYCLE, and PEDESTRIAN. The VAN, TRAILER, EMERGENCY_VEHICLE, and OTHER classes from the A9 Dataset cannot yet be recognized. This poses some limitations. Firstly, occlusion with classes that are not considered in our dataset will cause big blobs in the detected mask. Secondly, the classes TRAILER, VAN, EMERGENCY_VEHICLE, and OTHER are not included in COCO and will either not be detected or detected with another class label.

To address these limitations, in this work, we fine-tune the models on our TUM Traffic Intersection dataset. This approach not only addresses class mismatches but also improves detection by training models to understand our camera's perspective, considering factors such as view angle and distance. However, since the TUM Traffic dataset lacks segmentation labels, additional steps are required to extend the dataset with segmentation annotations.

This chapter elaborates on all the steps taken to achieve the aforementioned goals. Firstly, different state-of-the-art segmentation models are adapted to perform inference on the TUM Traffic Intersection dataset using publicly available pre-trained model weights on COCO and KINS datasets in Section 4.1. This provides insights into the general performance and limitations of the models and pre-trained datasets. Subsequently, Section 4.2 outlines the process of extending the TUM Traffic dataset labels with 2D visible and amodal instance segmentation masks and bounding boxes. This section is divided into subsections, beginning with an introduction to the structure of the newly extended OpenLABEL label format (Section 4.2.1). Following this, a description of a simple 2D annotation interpolation pipeline is presented in Section 4.2.3 to speed up the annotation process. The annotation procedure for frames is then described in detail in Section 4.2.4. Finally, Section 4.3 delves into the training of five different YOLO and C2F models and discusses their effectiveness.

4.1 Pre-trained Models on TUMTraf Intersection Dataset

After reviewing the literature for state-of-the-art modal and amodal instance segmentation models, promising and suitable models with publicly available pre-trained weights are chosen and adapted to perform inference on the TUMTraf Intersection Dataset. The models have been either pre-trained on COCO/COCOA or KITTI/KINS datasets. Figure 4.1 and Ta-



(a) YOLOv7 **(b)** YOLOv8

(c) C2F **(d)** WALT

(e) E2EC **(f)** AISFormer

Figure 4.1: Illustration of YOLOv7, YOLOv8, C2F_Seg, WALT, E2EC, and AISFormer models on an image in the TUMTraf Intersection dataset. YOLOv7 and YOLOv8 were trained on the COCO dataset; C2F, WALT, E2EC, and AISFormer were trained on the KINS dataset.

ble 4.1 demonstrate the results obtained from the instance segmentation models YOLOv7 and YOLOv8, alongside the amodal instance segmentation models C2F_Seg, WALT, E2EC and AISFormer, on an image taken from TUM Traffic Intersection Dataset. The two YOLOv7 and YOLOv8 shown here were trained on COCO dataset, while the weights of the other four AIS models were trained on the KINS dataset. For a fair comparison, C2F_Seg receives as input the visible detections of YOLOv8 and is, therefore, just an amodal detection extension of the IS model YOLOv8.

Overall, the results on the TUMTraf Intersection Dataset show that IS models pre-trained on COCO can detect more objects than AIS models pre-trained on KINS. The advantage of COCO, with its larger dataset containing 2.5 million labeled objects, has likely contributed to the higher detection rate. Additionally, YOLOv8 outperforms its predecessor, YOLOv7. E2EC and AISFormer struggle to detect large objects, possibly due to being trained on frames with smaller resolutions.

Furthermore, the YOLO models are one-stage object detection models and are much faster in inference, especially after being exported to TensorRT. In contrast, the AIS models require longer inference times and are thus not suitable for real-time detection. Detailed analysis results of inference speeds are provided in Section 5.4.

Model	# person	# vehicle
Ground Truth	6	19 (16 cars/vans, 1 motorcycle, 2 buses, 1 truck, 1 bicycle)
YOLOv7_coco	4	16 (10 cars, 1 motorcycle, 1 bus, 4 trucks)
YOLOv8x_coco	6	19 (13 cars, 1 motorcycle, 1 bus, 3 trucks, 1 bicycle)
C2F_Seg	6	19
WALT	1	11
E2EC	4	8

Table 4.1: Number of detections generated from the inference of YOLOv7, YOLOv8, C2F_Seg, WALT, E2EC, and AISFormer models, as shown in Figure 4.1. The 'Ground Truth' denotes the actual count of persons and vehicles present. The YOLO models, particularly YOLOv8 trained on COCO, exhibit the highest detection rates. E2EC and AISFormer encounter challenges in detecting large objects, while WALT faces difficulties in identifying pedestrians.

4.2 Segmentation Annotation for TUMTraf Intersection Dataset

The TUM Traffic dataset provides 3D bounding box labels but lacks segmentation masks. Therefore, it cannot be directly used to train instance segmentation models. To address this limitation, this work extends the TUM Traffic Intersection dataset by annotating visible and full instance masks and bounding boxes. Additionally, we adjust the OpenLABEL annotation schema of the TUMTraf dataset to accommodate these new annotations.

4.2.1 Extended Annotation Formats

Extended OpenLABEL Format

First of all, we extend the OpenLABEL annotation format, described in Section 2.3.1, of the TUM Traffic dataset to include additional full instance segmentation annotations. Each "object_data" now includes additional "bbox" and "poly2d" attributes containing two bounding boxes and polygons. The visible and amodal bounding boxes and masks are differentiated by the "full_..." and "visible_..." name attributes. We describe the instance masks by the bounding coordinates around the mask areas. The structure of one extended annotated object is shown in Listing 4.1.

Listing 4.1: Illutration of the extended OpenLABEL Annotation JSON Structure

```
"object_id" :{
  "object_data": {
    "name" :str,
    "type" :str,
    "cuboid" :{...},
    "bbox" :[
    {
      "name": "full_bbox",
      "val": [x_center, y_center, width, height]
    },
    {
      "name": "visible_bbox",
      "val": [x_center, y_center, width, height]
    }
    ],
    "poly2d" :[
    {
```

```
          "name": "full_mask",
          "val": [x1, y1, x2, y2, x3, y3, ...]
        },
        {
          "name": "visible_mask",
          "val": [x1, y1, x2, y2, x3, y3, ...]
        }
        ]
    }
  }
```

Extended COCO Format

For compatibility with the CVAT labeling tool, which does not support the OpenLABEL format, we convert labels to and from COCO format for importation into and exportation out of CVAT. Inspired by the annotation structure of the KINS dataset, we also extend the COCO annotation format to store the additional amodal segmentation labels. The structure of one extended object annotation is shown in Listing 4.2. The modifications involve replacing the single "segmenation" attribute with two separate "i_segm" and "a_segm" attributes describing the visible (inmodal) and full (amodal) instance masks. The attributes "area" and "bbox" are replaced by "a_area", "i_area", "a_bbox" and "i_bbox" accordingly. Here, we also describe the instance masks by the bounding coordinates around the mask areas to ease the conversion to and from our OpenLABEL format. Additionally, the "iscrowd" attribute is set to 0, indicating that the annotation refers to a single object.

Listing 4.2: Illutration of the extended COCO Annotation JSON Structure

```
annotation  {
        "id"          : int,
        "image_id"    : int,
        "category_id": int,
        "i_segm"      : [polygon],
        "a_segm"      : [polygon],
        "i_area"      : float,
        "a_area"      : float,
        "i_bbox"      : [x_min,y_min,width,height],
        "a_bbox"      : [x_min,y_top_min,width,height],
        "iscrowd"     : 0
}
```

4.2.2 Annotation Format Converters

Different stages of the work requires different label formats. In particular, YOLOv8 mandates data labels in YOLO format with a specific folder structure. The Providentia Mono3D Toolchain operates with data in OpenLABEL format. Labels are imported and exported in COCO format to and from CVAT for labeling purposes. Therefore, we implement different annotation format converters to facilitate the transformation of labels from the TUM Traffic Dataset OPENLabel format to YOLO and COCO formats, and vice versa.

Several considerations were taken into account during the implementation of these converters, as well as in the utilization of labels in these formats. Firstly, we address the variance in bounding box specifications. The "x_center" and "y_center" values of OpenLABEL format has to be converted to "x_min" and "y_min" values of COCO format. While YOLO also utilizes

bounding boxes with "x_center" and "y_center" values, however, all coordinates specified in this annotation format are normalized to the range of $[0, 1]$. Secondly, as the YOLO format does not inherently include attribute names to distinguish between bounding box and mask annotations, additional logic is implemented to check this. A polygon requires a minimum of three vertices, which is six values, that is why this check is valid. Furthermore, the converters offer the flexibility to choose whether to convert only visible annotations, only amodal annotations, or both.

4.2.3 2D Annotation Interpolation Pipeline

Leveraging the structure of the TUMTraf Intersection dataset, which comprises eight sequences, we develop a simple 2D annotation interpolation pipeline that can interpolate 2D annotations between consecutive frames, thereby speeding up the labeling process. To interpolate between two annotated frames, we formulate the matching of object annotations as a Linear Assignment Problem (LAP) and exploit the Jonker-Volgenant algorithm to match objects of the first frame to the objects of the second frame. The Jonker-Volgenant algorithm [JV88] is faster than the famous Hungarian algorithm, with a time complexity of $O(n^3)$ compared to $O(n^4)$. The pipeline then interpolates annotations between matched objects, interpolating each point of the polygon contour individually for precise label interpolation.

In particular, given a sequence with some frames annotated. The 2D annotation interpolation pipeline first sorts the frames in ascending chronological order, then for each pair of consecutive labeled frames, the object annotations are interpolated from the first frame to the second frame by following the following steps:

1. A 2D distance matrix D of shape $N \times M$ is created, with N and M as the amount of the object's annotation of the first frame and second frame, respectively. Each cell $D[i, j]$ represents the straight-line distance, also known as Euclidean distance, of the center of the i-th object from the first frame to the center of the j-th object from the second frame. In cases where the categories of the two objects are not identical, the distance is set to infinity. The distance between a visible annotation and an amodal annotation is also set to infinity.

$$D(i,j) = \begin{cases} \sqrt{(x_{\text{center}_i} - x_{\text{center}_j})^2 + (y_{\text{center}_i} - y_{\text{center}_j})^2} & \text{if category}_i = \text{category}_j \\ \infty & \text{otherwise} \end{cases}$$

2. This matrix is then given to the `scipy.optimize.linear_sum_assignment()` function, exploting the Jonker-Volgenant function, to match each object annotation of the first frame to one object annotation of the second frame with minimal distance.

3. For each pair of matched object annotations, the visible and full masks are interpolated. As already mentioned, the masks are stored in forms of the bounding coordinates around the mask areas. We again match each polygon coordinate of the first object mask with one polygon coordinate of the second object mask. However, here, only a simple search for nearest neighbor is done. Let $M_0 = \{(x_{0i}, y_{0i})\}_{i=0}^{(}n-1)$ and $M_1 = \{(x_{1i}, y_{1i})\}_{i=0}^{(}m-1)$ be the set of n and m bounding coordinates of the two matched object masks. Then the nearest neighbor of one coordinate $\{(x_{0i}, y_{0i})\}$ of the first mask if calculated as:

$$\text{Nearest Neighbor}((x_{0i}, y_{0i}), M_1) = \underset{j}{\operatorname{argmin}}\, d((x_{0i}, y_{0i}), (x_{1j}, y_{1j}))$$

with

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

4. Then linear interpolation is used to interpolate each single polygon point.

$$x_k = x_0 + \frac{x_1 - x_0}{n+1} \cdot k$$

$$y_k = y_0 + \frac{y_1 - y_0}{n+1} \cdot k$$

With n being the number of unannotated frames in the middle, $k \in [1, n]$, and (x_0, y_0) and (x_1, y_1) being the matched polygon point from the first frame and the second frame, respectively.

5. Finally, bounding boxes are calculated from the masks, object category, and other attributes are copied over.

This 2D detection interpolation pipeline goes beyond simple copying and pasting of bounding boxes and masks. Each point of the polygon contour of the mask is interpolated individually. Therefore, not only the position but also the shape and rotation of the object are interpolated, providing much more accurate labels.

However, this pipeline is not flawless and still requires manual refinement afterward. One limitation is that a single false matching can trigger a cascade of erroneous matches, given that each object from the second frame can only be matched with one object from the first frame. To mitigate this challenge, future research could explore more sophisticated solutions for data association, facilitating more precise matching between objects for interpolation. Another limitation lies in the current use of linear interpolation, which assumes a constant velocity of road users between frames, which may not always hold true. To address this limitation, future work could leverage tracking algorithms to monitor the moving velocities of road users and interpolate the 2D masks to more precise locations.

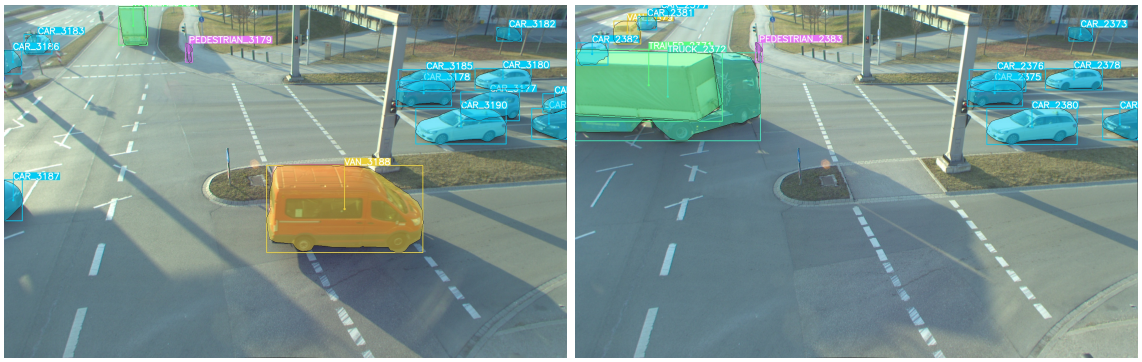
4.2.4 Annotating Instance Masks

To begin, we annotate approximately every tenth frame of the dataset, excluding the test set, with both modal and amodal instance masks, covering nearly 10% of the dataset. However, instead of entirely manual labeling, we leverage the YOLOv8x segmentation model pretrained on COCO to automatically detect the visible masks of objects. Subsequently, we import these detections into the Computer Vision Annotation Tool (CVAT), where the visible masks are manually refined and extended to amodal masks. The object categories are also adjusted to align with the TUMTraf Dataset specifications. We have taken great care to ensure the accurate annotation of the full mask by searching for the object's occluded parts in another frame where they are visible. A sequence with around 10% annotated frame is then given to the 2D detection interpolation pipeline so that the remaining frames in the middle can be automatically annotated. Afterward, we double-check the annotations and correct the erroneous ones.

Within the limited timeframe of this bachelor thesis, we have only interpolated and refined the visible and full segment annotations for three sequences, including R02_S01 from both south1 and south2 cameras and R02_S02 from south1 camera with the addition of around 10% of frames from the remaining sequences excluded test frames. In total, 1238 frames from the TUMTraff Intersection Dataset are annotated. Some examples are shown

in fig. 4.2. These annotated frames are then partitioned into training, validation, and testing sets comprising 1022, 125, and 91 frames, respectively, adhering to the original split of the TUMTraff Intersection Dataset. Looking ahead, there is potential to enhance and extend this annotation pipeline to cover the remaining TUMTraf Intersection Dataset and potentially the entire TUMTraf Dataset with segmentation annotations. In the future, this annotation pipeline can be further improved and utilized to extend the remaining TUMTraf Intersection Dataset and even the entire TUMTraf Dataset with segmentation annotations.

R02 S01 south1



R02 S01 south2

Figure 4.2: Four frames from release 02 sequence 01 of TUMTraf Dataset with the south1 and south2 camera, respectively. The frames are visualized with the object's full instance masks, full bounding boxes, and categories. The missing classes from COCO, like TRAILER and VAN, are also annotated.

4.3 Instance Segmentation Node

Similar to the Instance Segmentation Node of the Providentia Mono3D system, we implement a 2D detector node utilizing the YOLOv8 instance segmentation model. The interaction between the camera driver node and the 3D detection node remains unchanged. This 2D detector node receives full-resolution 1920x1200 RGB frames from shared memory and publishes the 2D detection results to the 3D detector through ROS. The input frames, however, are not downscaled before inference. Research has demonstrated that maintaining full resolution during inference yields significantly better performance compared to downsampling (as discussed in section 5.3).

The new library style of YOLOv8 has significantly simplified interaction with the model. Models in various formats, such as Pytorch (.pt) or TensorRT (.engine/.trt), can be loaded in the same way. After inference, we apply non-maximum suppression (NMS) with a threshold of 0.65 to filter out heavily overlapped detections. Subsequently, detections with lower

confidence are filtered based on class-wise confidence thresholds. To differentiate between detection results of model weights trained on different datasets, including COCO, TUMTraf, and NuImages, the 2D detector node implements additional logic to correctly interpret category indices, utilizing different index-to-category mappings for each dataset.

For each input frame, the detector node publishes an output array of detected instances containing 2D bounding boxes, 2D segmentation masks, and confidence scores to downstream processes. The detected 2D segmentation instance masks can currently be transmitted in two formats: bit-packed masks or polygon contours. The former utilizes bit-packing to serialize pixel mask arrays for each instance. This results in 1920*1200 pixels per frame for full resolution, with each pixel being one bit, resulting in a total of 285000 bytes per frame. Alternatively, the latter approach transmits only the image coordinates (x,y) of the polygon points of the detected mask's contour, with each x or y coordinate being an unsigned-8-bit integer, requiring only 2 bytes per point. Hence, up to 142500 polygon points can be sent before reaching the level of bit-packed masks. Considering that each frame typically contains 20 to 30 objects, allowing each object to use 4750 points to describe the polygon contour seems excessive. Therefore, we advocate for the second option as a better choice. However, since the current 3D detector expects bit-packed masks as input, a complete transition to polygon contours is not feasible yet. A future work would be to adjust the 3D detector's message reception to eliminate the need for sending bit-packed masks entirely.

Additionally, 2D visualizations and OpenLABEL detection files can generated for further analysis and debugging.

4.3.1 Training different YOLOv8 models

Training different YOLOv8 models is a crucial aspect of this work. We choose the YOLOv8x, the largest model that promises the best accuracy, and train different model weights and evaluate their performance on the TUMTraf Intersection Dataset as well as their general generalization capabilities. The above-mentioned annotated split of the TUMTraf Intersection Dataset is converted to YOLO's required data structure as preparation for training.

We focus on training two main models. The first YOLOv8x model is trained from the annotated TUMTraf Intersection Dataset from scratch for a total of 300 epochs on a full image resolution of 1920. The second utilizes the pre-trained YOLOv8x model published by Ultralytics, which was initially trained on COCO with an image size of 640 for 500 epochs, and fine-tunes it on the TUMTraf Intersection Dataset with an image size of 1920. Both models demonstrate improved performance in both 2D and the subsequent 3D detections. Particularly noteworthy is the first model trained on TUMTraf from scratch, which not only outperforms the existing 2D detector based on YOLOv7 on the TUMTraf Intersection Dataset but also exhibits superior generalization to other scenarios, such as nighttime and highway scenes. Compared to YOLOv7, it achieves a 2D mAP@[.5:.95] difference of over 18% and a 3D mAP@[.10] difference of almost 8%. Further details of the evaluations are elaborated in chapter 5.

Additionally, we fine-tune a model on the TUMTraf Intersection Dataset at a resolution of 640 to validate that training and detecting from higher image resolutions yield higher detection accuracy. In particular, the model weight with the highest accuracy at resolution 640 still performs approximately 16.5% worse than the model weight with the highest accuracy at resolution 1920.

To further enhance performance, we want to explore nuImages for pre-training. As described in Section 2.2, nuImages is a significantly larger dataset compared to COCO, making it a potentially better candidate for pre-training. However, due to its immense size, training on nuImages requires more time per epoch and may require many epochs to converge. Fol-

lowing the training of the YOLOv8x model from scratch on nuImages with a resolution of 1280 for 50 epochs, we assess its performance. It is evident that pre-training on nuImages for 50 epochs is not enough to yield any improvement compared to the published model pre-trained on COCO. We are currently continuing the training of this model in pursuit of improved results.

4.3.2 TensorRT Optimization

To preserve the real-time performance of the Providentia system, we export the models to TensorRT for accelerating inference. NVIDIA's TensorRT [Van16] is a high-performance deep-learning inference library designed to optimize and accelerate the inference of deep neural networks on NVIDIA GPUs. It employs various techniques, such as weight and activation precision calibration, to achieve FP32, FP16, or INT8 quantization with a significantly lower model footprint. Layer and tensor fusion, combined with kernel auto-tuning, further maximizes the GPU utilization. We exported the model using FP16 quantization and optimized it for inference on TUMTraf full-resolution frames of size 1920.

The exportation of the YOLOv8x model from Pytorch to TensorRT format significantly boosts the inference speed, nearly tripling it from 10 frames-per-second (FPS) to 28 FPS for inference at an image resolution of 1920x1920, and from 66 FPS to 200 FPS for an image resolution of 640x640.

4.3.3 Training C2F model

Since the YOLOv8x model trained on the TUM Traffic Intersection dataset from scratch has achieved the best performance improvement in visible 2D segmentation, we leverage its detections to extend them to amodal masks using the C2F-seg model. First, we adapt and fine-tune the C2F-seg amodal segmentation model on the TUMTraf Intersection Dataset. Subsequently, we give it the visible detections as inputs and generate amodal detections, which are then passed to the subsequent 3D detector to produce the final 3D bounding boxes.

It is intriguing to explore whether the additional information provided by amodal masks can enhance the final 3D perception results. However, the results are somewhat surprising; despite receiving visible detections from the YOLO model as input, the C2F model achieves a lower improvement in 3D performance (only 1.90% compared to the 7.53% improvement of YOLOv8x trained on TUMTraf Intersection). This suggests that extending to amodal masks using C2F has even worsened the final 3D perception performance. Further in-depth analysis of this phenomenon would be valuable to uncover the underlying reasons.

Additionally, regarding inference time, C2F requires, on average, around 22 milliseconds to extend one visible mask to an amodal mask. For a frame containing 20 objects, this would add an additional 440 milliseconds on top of the visible segmentation time. There may be potential for improvement when C2F can be exported to TensorRT in future work.

Chapter 5

Evaluation

This chapter conducts a comprehensive evaluation of different instance segmentation YOLO models and amodal instance segmentation C2F models trained on different datasets, including COCO, KINS, nuImages and TUMTraf Intersection Dataset. Section 5.1 provides an overview of all models to be evaluated. Section 5.2 introduces the evaluation metrics chosen for both the 2D and 3D detection stage. Mean Average Precision (mAP) and mean Intersection over Union (mIoU) are utilized for 2D detection evaluation, while 3D mAP is used for assessing the final 3D perception results. Next, the 2D quantitative results of the trained models are discussed in Section 5.3 and compared against the baseline model YOLOv7. Subsequently, the inference speed is compared in Section 5.4. Section 5.5 demonstrates the effectiveness of different trained segmentation models on the final 3D perception results, with some qualitative results shown in Section 5.6.

All experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU card. In the tables, small green and red numbers indicate improvement and reduction compared to the YOLOv7 baseline model, respectively. Bold and underlined values denote the highest value, while bold values represent the second-highest value in each column.

5.1 Comparison Objectives

Table 5.1 provides an overview of all models compared in this work. In this chapter, we define the model symbol as follows: If the model symbol contains only one dataset name, it means that the model is trained only on that particular dataset. On the other hand, if the model symbol contains two dataset names, it indicates that the model is pre-trained on the first dataset and fine-tuned on the second dataset.The YOLOv7_coco and the YOLOv8x_coco models are publicly available pre-trained models on COCO with an image size of 640, they are trained for 30 epochs and 500 epochs, respectively. YOLOv7_coco is utilized in the Providentia Mono3D system and is referred to as a baseline model in this chapter. C2Fseg_coco and C2Fseg_kins are published weights of the C2F segmentation model trained on KINS and COCOA datasets using a batch size of 16 with a total of 45k and 10k iterations, respectively. The remaining four models are trained within the scope of this thesis. The YOLOv8x_coco_tumtraf_640 is pre-trained on COCO and then fine-tuned on TUMTraf Intersection dataset with image resolution of 640 for 150 epochs. The YOLOv8x_coco_tumtraf_1920 is pre-trained on COCO and then fine-tuned on TUMTraf Intersection dataset with image resolution of 1920 for 250 epochs. The YOLOv8x_tumtraf is trained from scratch on the TUMTraf Intersection dataset with image resolution of 1920 for 250 epochs. Lastly, YOLOv8x_nuImg is trained on nuImages with a resolution of 1280 for 50 epochs. Continuous training of this model is necessary because nuImages is a vast dataset, and therefore, 50 epochs are not sufficient for significant

improvement over COCO.

Model Symbol	Trained on	image size
$YOLOv7\_coco$	COCO	640^2
$YOLOv8x\_coco$	COCO	640^2
$YOLOv8x\_tumtraf$	TUMTraf	1920^2
$YOLOv8x\_coco\_tumtraf\_640$	COCO, finetuned on TUMTraf	640^2
$YOLOv8x\_coco\_tumtraf\_1920$	COCO, finetuned on TUMTraf	640^2, 1920^2
$YOLOv8x\_nuImg$	nuImages	1280^2
$C2Fseg\_coco$	COCO dataset	(*)
$C2Fseg\_kins$	KINS	(*)
$C2Fseg\_kins\_tumtraf$	KINS, finetuned on TUMTraf	(*)

Table 5.1: An overview of instance segmentation models compared in this work. (*) : as described in Section 3.2, C2Fseg models crop input images based on visible regions and resize each region of interest (ROI) to 256^2 px.

5.2 Evaluation Metrics

In order to assess the effectiveness of the 2D and 3D object detectors, we utilize widely recognized metrics commonly employed in the literature on object detection and instance segmentation. Specifically, we employ mean average precision (mAP) and mean Intersection over Union (mIoU) for evaluating our 2D object detector. For our 3D object detector, we utilize 3D mAP as the primary metric.

5.2.1 Intersection over Union

Intersection Over Union (IoU) is a metric used to determine the degree of geometric overlap. This metric is calculated by dividing the overlap area by the union area between the ground truth and prediction. A perfect overlap yields a maximum IoU value of 1, whereas no overlap results in 0. Mean IoU is obtained by taking the average of the IoUs of each class.

$$\textbf{IoU} = \frac{GT \cap Pred}{GT \cup Pred}$$

In basic 2D or 3D object detection tasks, IoU is calculated as the overlap area divided by the union area between the ground truth 2D or 3D bounding box and the predicted 2D or 3D bounding box. In most cases, this IoU is used as an intermediate step in determining the True Positive (TP), False Positive (FP), or False Negative (FN) detections. To make this determination, IoU is first calculated between the prediction and the ground truth. If the IoU value is greater than a predetermined threshold (such as 0.5), the prediction is classified as TP; otherwise, it is classified as FP.

In instance segmentation task, where predictions are segmentation masks, IoU analysis occurs at the pixel level. The definition of TP, FP, and FN is slightly altered as it is not based on a predefined threshold. A True Positive, in this case, is the number of pixels of intersection between the ground truth and the predicted mask. This is mathematically equivalent to the logical AND operation of both masks. A False Positive is the predicted area outside the ground

truth, which is the logical OR of the ground truth and prediction, minus the ground truth. A False Negative is the number of pixels within the ground truth area that the model failed to predict, computed as the logical OR of the ground truth and prediction, minus the prediction masks. These definitions yield the following expressions:

$$\begin{aligned}\mathbf{TP}_{\text{mask}} &= GT_{\text{mask}} \cap Pred_{\text{mask}},\\ \mathbf{FP}_{\text{mask}} &= (GT_{\text{mask}} \cup Pred_{\text{mask}}) \setminus GT_{\text{mask}},\\ \mathbf{FN}_{\text{mask}} &= (GT_{\text{mask}} \cup Pred_{\text{mask}}) \setminus Pred_{\text{mask}}\end{aligned}$$

5.2.2 Average Precision

Average precision (AP) utilizes the concept of Precision and Recall. Precision represents the ratio of true positives and the total number of predicted positives, while Recall denotes the ratio of true positives and the total number of actual positive samples. Precision reflects the degree of confidence that a model has in classifying a sample as Positive, while Recall indicates the number of positive samples correctly identified by the model. In essence, precision measures the quality, while recall measures the quantity.

$$\textbf{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \textbf{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The Precision-Recall (PR) curve presents the tradeoff between precision and recall values for different thresholds. The average precision (AP) of a category is obtained by calculating the area under the PR curve of that category. The mean average precision (mAP) is simply the average of all AP values across different categories. To differentiate between different IoU thresholds, it's common to specify the IoU threshold after mAP. For example, $mAP@[.5]$ denotes mAP at an IoU threshold of 0.5, whereas $mAP@[.5 : .95]$ represents the average mAP across different IoU thresholds ranging from 0.5 to 0.95, with a step of 0.05, i.e., 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, and 0.95. $mAP@[.5 : .95]$ offers a more comprehensive evaluation by considering a broader range of IoU thresholds, capturing both high and low overlap between predicted and ground truth bounding boxes.

5.3 2D Quantitative Analysis

5.3.1 YOLO Models Quantitative Analysis

Table 5.2 examines the 2D visible instance segmentation mean average precision (mAP) and mean intersection over union (mIoU) of several YOLO models on the segmentation mask annotated TUMTraf Intersection test set. A confidence threshold of 0.25 and image sizes of 1920, 1280, and 640 are used. The results reveal some noteworthy observations.

- Firstly, comparing the two pre-trained model weights on COCO, the YOLOv8x_coco outperforms YOLOv7_coco across all image resolutions. This reaffirms the statements made when YOLOv8 was published.
- Secondly, the detection accuracy is proportional to image resolution. All accuracies drop as image resolution decreases from 1920 to 640. The only exception is the model

Image resolution of 1920^2 px

Model	mAP@[.5]	mAP@[.5:.95]	mIoU
YOLOv7_coco (Baseline)	82.70	57.60	85.36
YOLOv8x_coco	77.20 -5.50	61.20 +3.60	89.51 +4.15
YOLOv8x_tumtraf	**94.50** +11.80	**75.90** +18.30	**91.51** +6.15
YOLOv8x_coco_tumtraf_640	44.40 -38.30	27.50 -30.10	80.80 - 4.56
YOLOv8x_coco_tumtraf_1920	**89.80** +2.60	**68.10** +10.50	**90.58** +5.22

Image resolution of 1280^2 px

Model	mAP@[.5]	mAP@[.5:.95]	mIoU
YOLOv7_coco (Baseline)	78.50	53.40	85.72
YOLOv8x_coco	80.10 +1.60	58.50 +5.10	88.50 +2.78
YOLOv8x_tumtraf	**93.80** +15.30	**71.70** +18.30	**90.00** +4.28
YOLOv8x_coco_tumtraf_640	92.00 +13.50	63.50 +10.10	85.30 -0.42
YOLOv8x_coco_tumtraf_1920	**95.90** +17.40	**74.30** +20.90	**88.60** +2.88

Image resolution of 640^2 px

Model	mAP@[.5]	mAP@[.5:.95]	mIoU
YOLOv7_coco (Baseline)	67.00	42.80	**85.36**
YOLOv8x_coco	**70.50** +3.50	**46.00** +3.20	82.12 -3.24
YOLOv8x_tumtraf	66.40 -0.60	42.90 +0.10	80.67 -4.69
YOLOv8x_coco_tumtraf_640	**86.10** +19.10	**59.40** +16.60	**85.97** +0.61
YOLOv8x_coco_tumtraf_1920	34.00 -33.00	18.80 -24.00	67.97 -17.39

Table 5.2: A comparative quantitive results of the YOLO instance segmentation models on the segmentation annotation extended test set. The models are evaluated on image sizes of 1920, 1280, and 640 with a confidence threshold of 0.25.

weight YOLOv8x_coco_tumtraf_640, which is pretrained on COCO and fine-tuned on TUMTraf Intersection Dataset with an image size of 640 and hence performs better for smaller image sizes. However, overall, the model weight with the highest accuracy at resolution 640 still performs approximately 16.5% worse than the weight with the highest accuracy at a resolution of 1920.

- Overall, YOLOv8x_tumtraf at an image resolution of 1920 achieves the highest accuracy, showing a significant improvement of 18.30% mAP@[.5:.95] compared to the baseline model YOLOv7. YOLOv8x_coco_tumtraf_1920 achieves the second highest of +10.50% against the baseline. These results demonstrate the effectiveness of training on the TUMTraf Intersection Dataset.

Training from scratch achieves better performance on the test set than fine-tuning from pre-trained weight on COCO. This is likely due to the limited number of frames annotated with segmentation masks in the TUMTraf Intersection Dataset, allowing models trained from scratch to better adapt and overfit to this specific dataset. Further investigation in the ablation study section will explore performance across the entire dataset, as well as on other datasets with varying scenes and camera settings, revealing the generalizability of the models.

5.3.2 C2F Models Quantitative Analysis

Model	invisible mIoU	full mIoU
C2Fseg_cocoa	31.49	78.71
C2Fseg_kins	**33.73**	**82.44**
C2Fseg_kins_tumtraf	**77.29**	**91.99**

Table 5.3: A quantitative comparison of different C2F amodal instance segmentation models on the segmentation annotation extended test set. Full mIoU represents the mIoU of the complete object mask, incorporating both visible and occluded parts. All models utilize ground truth instance segmentation masks as visible detection inputs.

Table 5.3 compares the performance of different C2F amodal instance segmentation models on the segmentation annotation extended test set. Full mIoU represents the mIoU of the entire object mask, including both visible and occluded parts, while invisible mIoU specifically represents the mIoU for the occluded part of the objects. To ensure a fair comparison, all three models receive the ground truth instance segmentation masks as visible detection inputs.

The results demonstrate that the C2F model weight pre-trained on the KINS dataset outperforms the pre-trained on the COCOA dataset. Additionally, fine-tuning on the TUMTraf Dataset has significantly improved the performance. The subsequent analysis in Section 5.5 will provide insights into whether amodal masks have a significant impact on the final 3D perception results.

5.4 Inference Speed

Real-time is a critical aspect of autonomous driving, where the goal is to achieve higher accuracy while maintaining real-time inference speed. The Frames Per Second (FPS) measurement is commonly used to evaluate the efficiency of the methods. A higher FPS indicates faster inference. The baseline YOLOv7 model, as stated in [Bir23], achieves frame rates between 55 and 60 FPS at a resolution of 640^2 px, 22 FPS at 1280^2 px, and only 12 FPS at 1920^2 px on the TUMTraf Intersection Dataset. The YOLOv8 model not only achieves better performance but also accelerates in speed. In our study, we measure the inference speed of these models on the TUMTraf Intersection Dataset, and the results are presented in Table 5.4.

Notably, the largest YOLOv8x TensorRT models achieve FPS rates of 26 to 28 at the full 1920^2 px resolution, marking a 233% improvement over the YOLOv7 TensorRT. Furthermore, the FPS increases to 62 and 200 as the resolution decreases to 1280^2 px and 640^2 px, respectively. Additionally, we document the inference speed of PyTorch models, observing around threefold acceleration in inference speed through exporting to TensorRT.

The C2F model extends visible masks to amodal masks, and hence, the inference speed per frame is contingent upon the number of object masks to be extended within each frame. C2F models require approximately 22 milliseconds to extend a visible mask to an amodal mask for one instance. With an average of 15 to 20 objects per frame in the TUMTraf Intersection Dataset, this translates to 330 to 440 milliseconds per frame, which is 2.3 to 3 FPS. However, as C2F relies on visible masks as input, which must be predicted beforehand, the computational cost of visible mask estimation must also be factored in. For example, utilizing predicted visible masks from YOLOv8x as input into C2F would result in a total inference time of 365 to 475 milliseconds per frame, equivalent to 2.1 to 2.7 FPS. Consequently,

Model	Model Format	Image Resolution		
		640^2px	1280^2px	1920^2px
YOLOv7	TensorRT	52	22	12
YOLOv8x	PyTorch	66	22	10
YOLOv8x	TensorRT	**200**	**62**	**28**

Table 5.4: A Comparison of model inference speed (FPS) across various image resolutions. The results highlight significant acceleration in inference speed with the YOLOv8 model, especially after exporting to TensorRT.

the C2F amodal instance segmentation model appears unsuitable for real-time applications. Nevertheless, there is potential for improved speed by exporting the C2F PyTorch model to TensorRT.

5.5 3D Perception Performance Analysis

Table 5.5 presents quantitative comparisons of 3D perception on the test sequence of TUM-Traf Intersection Dataset from both camera south1 and camera south2. The 2D detections from the existing baseline 2D detector based on YOLOv7 and from the proposed 2D detector based on YOLOv8 models are sent to the toolchain 3D detector as ROS messages. The final 3D output detections are then evaluated with Precision, Recall, and Average Precision metrics. The experimental setup aligns with the parameters defined in [Bir23], utilizing original LiDAR labels L_0 and non-tracking T_0, with an image resolution of 1920 and a confidence threshold set to 0.25. The results reveal some noteworthy observations.

- The integration of the YOLOv8 2D detector has yielded notable improvements over the baseline YOLOv7 detector, with the most significant improvement observed when utilizing the YOLOv8 model trained from scratch on the TUMTraf dataset (+7.53% 3D mAP@[.10] improvement against the baseline YOLOv7). The model pre-trained on COCO and subsequently fine-tuned on TUMTraf also showcases performance gains, although slightly less significant (+5.58% 3D mAP@[.10] improvement against the baseline). A closer look at the YOLOv8x_coco_tumtraf_1920 reveals a decline in both Precision and Recall, which is causing a significant drop in the AP of the BUS category. Further investigation shows that this model often detects two large adjacent objects as a single object mask, as shown in Figure 5.1.

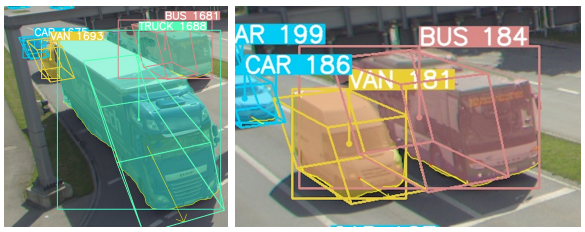


Figure 5.1: An illustration of YOLOv8x_coco_tumtraf_1920 model detecting two adjacent large objects as a single object mask.

- The TRUCK and VAN categories show remarkable performance enhancements across all trained models, while it's noteworthy that the VAN class is not included in the COCO dataset. However, the PEDESTRIAN category experiences diminished performance, characterized by a significant decrease in precision despite a relatively stable recall rate. This anomaly likely arises from a substantial increase in false positive (FP)

Classes	Precision	Recall	AP@[.10]
CAR	30.54	17.27	25.45
TRUCK	1.63	1.52	1.20
VAN	2.28	0.02	0.00
BUS	48.52	30.88	43.37
PEDESTRIAN	7.06	7.79	6.21
BICYCLE	12.62	30.62	11.61
mAP@[.10]			**10.98**

(a) YOLOv7_coco (Baseline): trained on COCO for 30 epochs at resolution 640 x 640

Classes	Precision	Recall	AP@[.10]	△Baseline
CAR	32.81	17.31	28.52	+3.08
TRUCK	10.74	5.32	8.63	+7.43
VAN	2.28	0.02	0.00	0.00
BUS	50.80	31.87	45.87	+2.50
PEDESTRIAN	6.53	11.04	5.98	-0.23
BICYCLE	11.80	30.18	11.54	-0.07
mAP@[.10]			**12.60**	**+ 1.62**

(b) YOLOv8x_coco: trained on COCO for 500 epochs at resolution 640 x 640

Classes	Precision	Recall	AP@[.10]	△Baseline
CAR	29.51	17.11	29.51	+4.06
TRUCK	24.27	13.99	24.27	+23.07
VAN	28.43	12.33	28.43	+28.43
BUS	43.51	30.54	43.51	+0.14
PEDESTRIAN	4.03	7.70	4.03	-2.18
BICYCLE	18.37	30.26	18.37	+6.76
mAP@[.10]			**18.51**	**+ 7.53**

(c) YOLOv8x_tumtraf: trained on TUMTraf Intersection dataset for 250 epochs at resolution 1920 x 1920

Classes	Precision	Recall	AP@[.10]	△Baseline
CAR	34.51	16.88	29.14	+3.69
TRUCK	16.08	9.57	12.33	+11.13
VAN	33.58	12.16	26.94	+26.94
BUS	40.47	19.52	34.52	-8.85
PEDESTRIAN	5.17	7.47	4.55	-1.76
BICYCLE	29.09	16.69	27.43	+15.82
mAP@[.10]			**16.86**	**+ 5.88**

(d) YOLOv8x_coco_tumtraf_1920: YOLOv8x_coco fine-tuned on TUMTraf Intersection dataset for 250 epochs at resolution 1920 x 1920

Classes	Precision	Recall	AP@[.10]	△Baseline
CAR	30.70	16.55	23.77	-1.68
TRUCK	0.00	0.00	0.00	-1.20
TRAILER	6.41	0.62	2.06	+2.06
BUS	5.77	1.15	3.18	-40.19
PEDESTRIAN	5.64	7.14	5.64	-0.57
BICYCLE	13.64	30.62	12.62	+1.01
mAP@[.10]			**5.91**	**- 5.07**

(e) YOLOv8x_nuImg: trained on nuImages dataset for 50 epochs at resolution 1280 x 1280

Classes	Precision	Recall	AP@[.10]	△Baseline
CAR	16.43	16.83	14.42	-11.03
TRUCK	16.48	10.44	13.39	+ 12.19
VAN	30.97	12.11	24.26	+ 24.26
BUS	33.59	19.05	28.90	-14.48
PEDESTRIAN	3.06	4.85	2.52	-3.71
BICYCLE	20.01	30.26	19.57	+ 7.96
mAP@[.10]			**12.88**	**+ 1.90**

(f) C2Fseg_kins_tumtraf: pre-trained on KINS then fine-tuned on TUMTraf Intersection dataset for 30 epochs

Table 5.5: This table presents the 3D detection quantitative results on the test sequence of TUMTraf Intersection Dataset from both camera south1 and camera south2. To make the table more readable, classes with Precision, Recall and AP of 0 are not shown here. Classes with labels in ground truth but no detections have an AP value of 0 and are still included in the mAP calculation. However, classes without labels in ground truth and with no detections are excluded and do not contribute to the mAP.

predictions. Upon visualizing the ground truth 3D labels against the model predictions, as shown in the first image of Figure 5.2, it becomes apparent that the 3D bounding box labels of the TUMTraf Intersection Dataset do not cover all the pedestrian instances. The model can correctly identify the presence of three pedestrians at the top left (depicted in red), which are not annotated in the ground truth (depicted in green), consequently contributing to a considerable rise in false positives. This issue appears not only in the PEDESTRIAN category but also in other classes, as is evident in the remaining images.

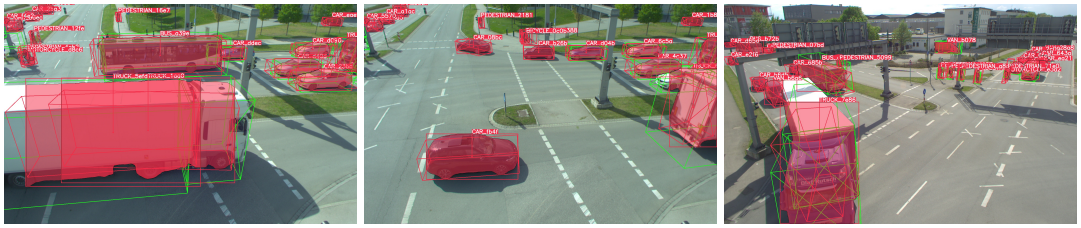


Figure 5.2: An Illustration comparing ground truth (green) with YOLOv8x_tumtraf predictions (red). These frames are from the TUMTraf Intersection Dataset test sequence. The frames reveal that the labels of the TUMTraf Intersection Dataset are incomplete, failing to label all objects in the image. Consequently, the models produce false positives when identifying unlabeled objects.

- YOLOv8 model trained on nuImages for 50 epochs still is not powerful enough to surpass the performance achieved by the model trained on COCO for 30 epochs. Given the extensive scale of the nuImages dataset, further training is deemed necessary to realize its full potential.

- The C2F model trained on KINS and fine-tuned on TUMTraf also demonstrates moderate performance gains in the TRUCK and VAN categories. However, there is also a notable decline in performance for the BUS and PEDESTRIAN categories. Overall, the trained C2F has a smaller improvement (+1.90% 3D mAP@[.10] against the baseline YOLOv7) compared to the improvement of trained YOLOv8x models even though it receives the visible detections from YOLOv8x as input. This indicates that extending to amodal has even worsened the final 3D perception performance.

5.6 Qualitative Analysis

Figure 5.3 presents the qualitative comparison of the models on the TUMTraf Intersection Dataset. The first row (a) showcases the results obtained from the existing 2D detector based on YOLOv7, utilized in the Providentia Mono3D system. Subsequent rows (b,c,d) present the outcomes of the newly implemented 2D detector leveraging different YOLOv8 model weights. The final row (e) displays the results of the amodal mask extended 2D detection obtained from the trained C2F model. Several notable observations emerge:

- The existing YOLOv7 model, pre-trained on COCO, exhibits limitations in detecting large objects, as evidenced by its inability to detect the truck and bus in the first image. Although YOLOv8 models pre-trained on COCO demonstrate improved performance in this regard, they still face challenges in detecting extremely large objects, often detecting only a portion of the object, as depicted in the first frame of the second row. This limitation can be attributed to their training on COCO with an image size of 640, which constrains their ability to detect objects larger than 640^2 pixels. In contrast, models fine-tuned on the TUMTraf Dataset with full image resolution showcase enhanced capability in detecting large objects, although sometimes requiring multiple overlapping

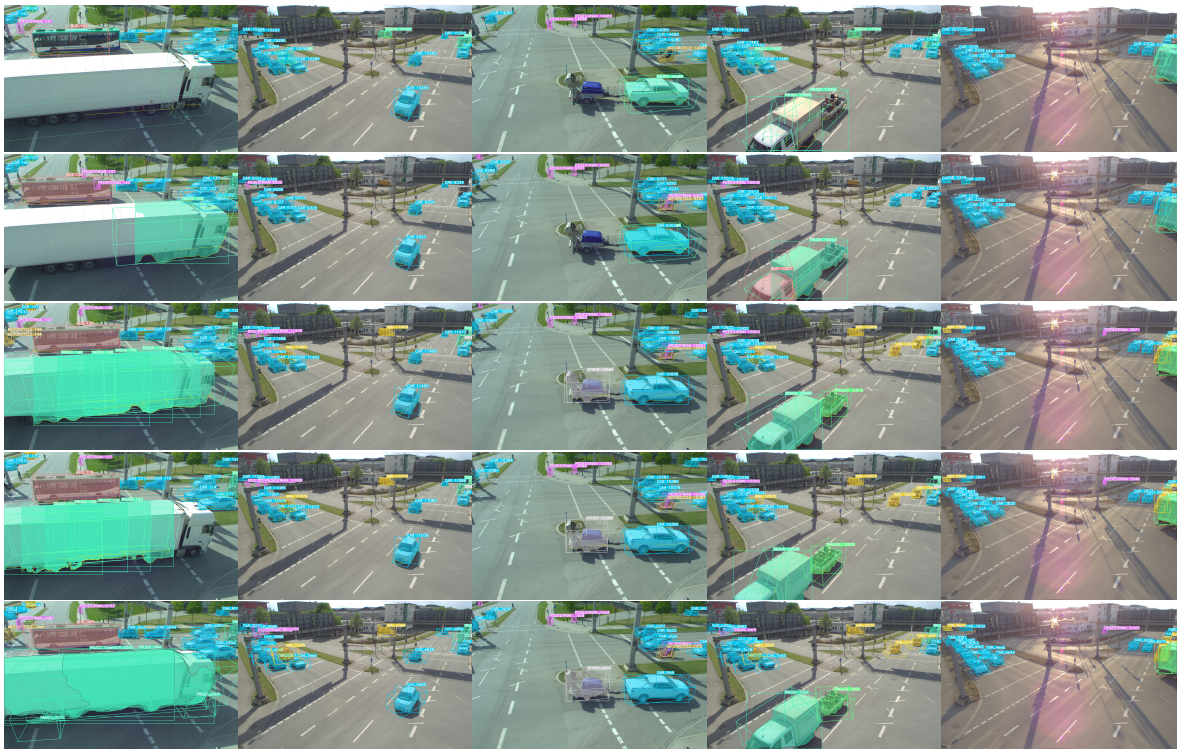


Figure 5.3: Qualitative comparison results of different models on TUMTraf Intersection Dataset. From top to bottom: a) $YOLOv7\_coco$, b) $YOLOv8x\_coco$, c) $YOLOv8x\_tumtraf$, d) $YOLOv8x\_coco\_tumtraf\_1920$, e) $C2F\_kins\_tumtraf\_1920$. The visualizations include 2D bounding boxes, 2D masks, 3D bounding boxes, and category labels.

masks to detect the entire object. Nevertheless, detecting with multiple overlapping masks is still superior to the inability to detect the object at all.

- Models fine-tuned on the TUMTraf Dataset with full resolution (b, c, d) demonstrate superior performance in detecting pedestrians, as evidenced by the accurate detection of three small pedestrians at the top left corner of the second image in each row. While these pedestrians are scarcely detectable by the first two models, they are accurately identified by the latter.

- Models fine-tuned on the TUMTraf Dataset exhibit the capability to distinguish between ten classes, surpassing the six classes recognized by models trained solely on COCO. This expanded class recognition facilitates the differentiation of vans, depicted in yellow, from trucks in green and cars in blue. Furthermore, it enables the detection of trailers in green in the fourth image and OTHER objects in gray in the third image.

- To ensure a fair comparison, visible detections from YOLOv8x fine-tuned on the TUMTraf Dataset are utilized as inputs for C2F. The resulting full masks provided by C2F facilitate the detection of both visible and invisible objects, as demonstrated in the fourth image, where overlapping full masks of cars on the left side are observed. The resulting full masks provided by C2F allowed for the detection of both visible and invisible objects. This can be seen in the fourth and the fifth images, where the full masks of cars on the left side overlap each other.

  The bottom contours of occluded objects are accurately detected using amodal masks, as demonstrated in Figure 5.4. However, given that the 3D bounding boxes can already be inferred accurately with visible masks alone, these additional amodal masks do not

significantly impact the final 3D perception performance. Additionally, the TUMTraf Intersection Dataset does not exhibit heavy occlusions. As outlined in Section 2.2.4, only 16.1% of instances are PARTIALLY OCCLUDED, and 0.8% are MOSTLY OCCLUDED. Therefore, in the overall context, amodal segmentation is unlikely to yield substantial benefits.

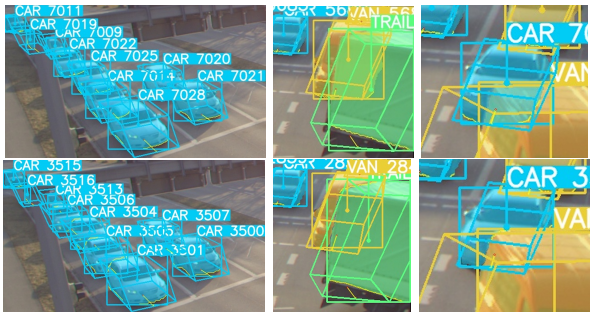


Figure 5.4: Illustrations comparing bottom contour extraction from amodal masks generated by C2F (first row) and from visible masks only by YOLO models (second row). Bottom contours of occluded objects can be detected accurately from amodal masks.

Chapter 6

Experiments

6.1 Performance on Night sequence

Figure 6.1: Night sequence: Qualitative comparisons of night sequence prediction of baseline detector $YOLOv7\_coco$ (first row) and the proposed detector with model weight $YOLOv8x\_tumtraf$ (second row) and with model weight $YOLOv8x\_coco\_tumtraf\_1920$ (third row).The visualizations include 2D bounding boxes, 2D masks, 3D bounding boxes, and category labels.

Table 6.2 and Figure 6.1 demonstrate the results on the night scenario sequence (S04) of the TUMTraf Intersection Dataset. The model trained from scratch on the TUMTraf Intersection Dataset achieves a notable performance enhancement, surpassing the existing 2D detector based on YOLOv7 by over 10%. Conversely, the model pre-trained on COCO and subsequently fine-tuned on the TUMTraf Intersection Dataset exhibits a modest improvement of 1.83%. Additionally, the performance of the YOLOv8x_coco pre-trained weight from Ultralytics is also assessed, showing an improvement of only 0.57% over the baseline.

The qualitative results demonstrate that the YOLOv8 models provide more stable predictions compared to the baseline YOLOv7. While YOLOv7 can identify vehicles, its predictions are unstable as objects can sometimes be identified and sometimes not. In contrast, YOLOv8 can predict objects with stability throughout the sequence, as illustrated in the first two frames.

YOLOv8x_coco_tumtraf_1920 has stable detections but performs notably worse than the YOLOv8x_tumtraf model. It exhibits buggy predicted object masks (frame 3 of the third row), with several objects remaining undetected (frames 1 and 2 of the third row).

YOLOv7_coco (Baseline)

Classes	Precision	Recall	AP@[.10]
CAR	26.85	9.92	20.21
TRUCK	0.46	0.07	0.26
VAN	0.00	0.00	0.00
BUS	22.58	8.85	19.68
mAP@[.10]			**8.03**

YOLOv8x_tumtraf

Classes	Precision	Recall	AP@[.10]	△Baseline
CAR	34.26	14.13	28.16	+ 7.95
TRUCK	4.68	4.56	3.86	+ 3.60
VAN	22.33	17.45	20.97	+ 20.97
BUS	40.88	29.24	38.30	+ 18.62
mAP@[.10]			**18.26**	**+ 10.23**

YOLOv8x_coco_tumtraf_1920

Classes	Precision	Recall	AP@[.10]	△Baseline
CAR	27.16	12.34	21.05	+ 0.84
TRUCK	4.18	3.94	3.45	+ 3.19
VAN	10.59	5.475	8.59	+ 8.59
BUS	18.23	13.98	16.21	- 3.47
mAP@[.10]			**9.86**	**+ 1.83**

Table 6.2: Night sequence: 3D detection quantitative comparisons on the night sequence.

Remarkably, YOLOv8x_tumtraf outperforms others in the night sequence of the TUMTraf Intersection Dataset, offering stable predictions and significantly improved identification of large vehicles.

6.2 Performance on TUMTraf Intersection Dataset

Model	S01	S02	S03	S04	Average	△Baseline
YOLOv7_coco (Baseline)	23.91	15.79	11.87	8.03	12.91	
YOLOv8x_coco	20.60	13.82	12.01	8.60	15.20	+ 2.29
YOLOv8x_tumtraf	**41.27**	**17.48**	**17.50**	**18.26**	**20.66**	+ 7.75
YOLOv8x_coco_tumtraf_1920	**32.42**	**17.91**	**21.03**	**9.86**	**19.27**	+ 6.36

Table 6.3: Entire TUMTraf Intersection Dataset: 3D mAP@[.10] comparisons across all sequences of the TUMTraf Intersection Dataset. Sequence 03 constitutes the largest portion of the TUMTraf Intersection Dataset, accounting for 50% of the dataset with 2400 frames, followed by the night sequence 04, analyzed previously, with 1200 frames (25%). Sequences 01 and 02 each contain 600 frames, collectively representing the remaining 25%. Thus, the average mAP values are weighted based on a ratio of 1:1:4:2 for sequences 01 to 04, respectively. This weighting ensures a fair representation of each sequence's contribution to the overall performance evaluation.

Table 6.3 presents the 3D mean Average Precision comparison of the YOLOv8 model weights with the baseline model YOLOv7 across all sequences of the TUMTraf Intersection Dataset. Once more, the model weight YOLOv8x_tumtraf showcases the most significant performance improvement. The improvement percentages are not very different from the 3D quantitative analysis of the test sequence alone in Section 5.5.

6.3 Performance on Highway

Section 6.3 illustrates the inference results of different YOLO models on one frame sequence taken from the measurement station S40 on the A9 highway. Comparing the two model

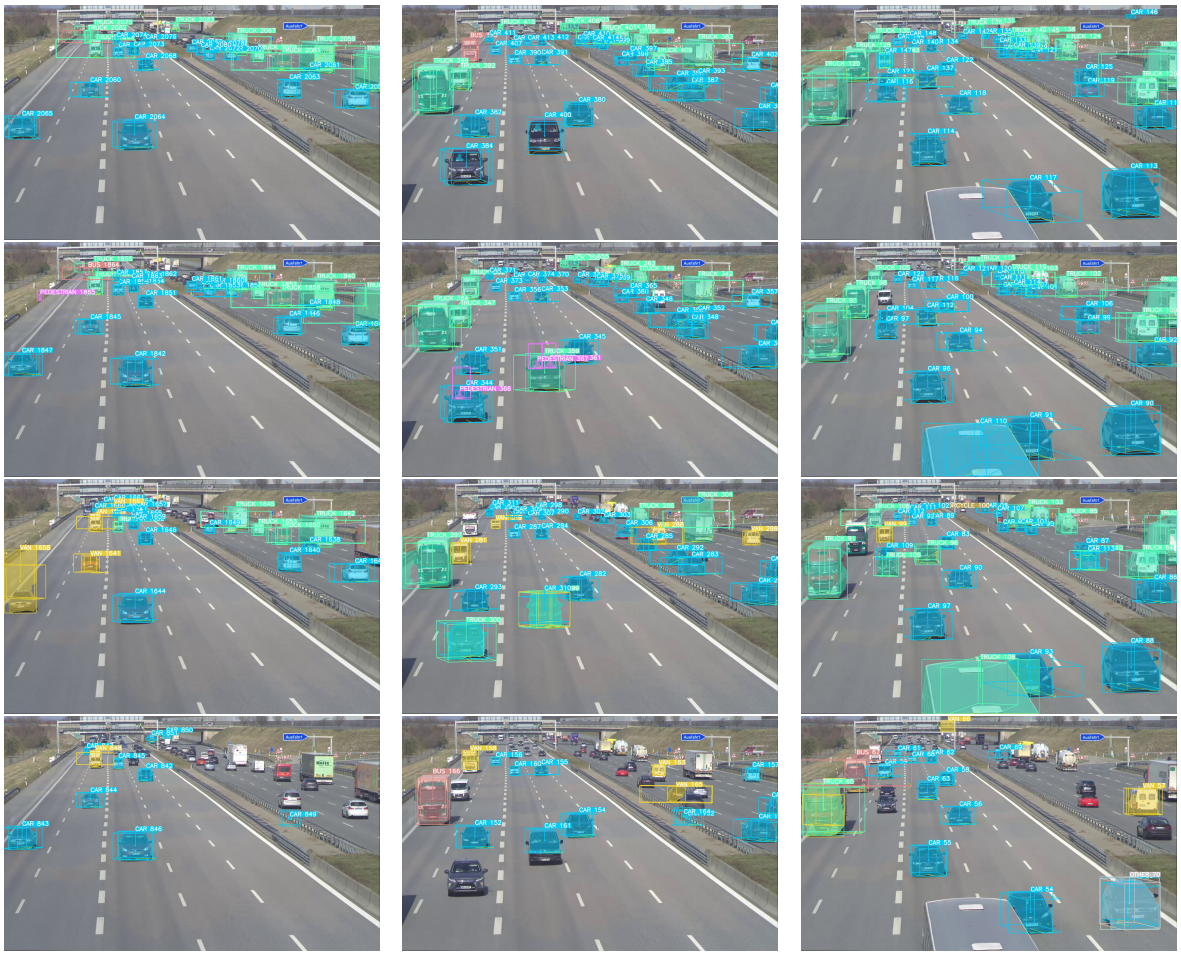


Figure 6.2: Highway: Qualitative comparisons on one high way sequence. Baseline detector YOLOv7_coco (first row) and the proposed detector with model weight YOLOv8_coco (second row), YOLOv8x_tumtraf (third row) and with model weight YOLOv8x_coco_tumtraf_1920 (fourth row).The visualizations include 2D bounding boxes, 2D masks, 3D bounding boxes, and category labels.

weights trained on COCO, YOLOv8x _coco again outperforms YOLOv7_coco. Interestingly, YOLOv8x_coco can even identify humans sitting inside the vehicles. Overall, YOLOv8x_coco also outperforms the other two models, YOLOv8x _coco_tumtraf_1920 and YOLOv8x_tumtraf on the highway, by detecting the most objects.

In this scenario, the models fine-tuned on TUMTraf Intersection Dataset do not perform well. Compared to YOLOv8x_coco, the YOLOv8x_tumtraf model weight, trained solely on TUMTraf Intersection Dataset, can detect slightly less objects, especially the small vehicles much further away from the camera.

Additionally, sometimes the detected mask is too large, as for the yellow van on the right of the first frame and the truck on the top left of the third frame. Occasionally, two adjacent large objects are also identified as a single object mask, such as the detected "TRUCK 103" in the top left corner of the fourth image. YOLOv8x_coco_tumtraf_1920 performs poorly in this scenario with too few detections.

This confirms that training and fine-tuning on TUMTraf Intersection Dataset will lead to performance boost in intersection scenarios but will not generalize well to other scenarios like highways. Therefore, to achieve the same performance boost as in intersection sequences, annotations for highway sequences also have to be extended with segmentation mask labels for training the models.

6.4 Time-shifted ground truth labels and 3D tracker

Model + Time-shifted	mAP@[.10]	△ non Time-shifted	△ Baseline
YOLOv7_coco (Baseline)	15.20	+ 4.22	
YOLOv8x_coco	17.77	+ 5.17	+ 2.57
YOLOv8x_tumtraf	**33.31**	+ 14.80	+ 18.11
YOLOv8x_coco_tumtraf_1920	**22.26**	+ 5.40	+ 7.06

Table 6.5: Shifted ground truth: 3D detection quantitative comparisons on the test sequence of TUMTraf Intersection Dataset from both camera south1 and camera south2 with time-shifted ground truth labels. △non-shifted column shows the improvement achieved by using time-shifted ground truth compared to non time-shifted. △baseline shows the improvement against the baseline model YOLOv7_coco with time-shifted ground truth.

The TUMTraf Dataset provides 3D LiDAR labels. These labels, however, might be slightly shifted because of sensor delay between the RGB camera and the LiDAR sensor frame. The LiDAR label shifting technique, described in [Bir23], involves estimating spatial velocity to correct label positions based on the known synchronization error time delta. Table 6.5 shows the quantitative evaluation after correcting the 3D LiDAR labels. These time-shifted LiDAR labels lead to an improvement in the overall performance compared to using original LiDAR labels. This enhancement demonstrates the importance of addressing sensor delay between the camera and LiDAR sensor frames, as it helps mitigate inherent inaccuracies in the LiDAR labels, ultimately enhancing evaluation accuracy.

Model + Time-shifted + polyMOT tracker	mAP@[.10]	△ no tracker	△Baseline
YOLOv7_coco (Baseline)	16.23	+ 1.03	
YOLOv8x_coco	18.24	+ 0.47	+ 2.01
YOLOv8x_tumtraf	**34.02**	+ 0.71	+ 17.79
YOLOv8x_coco_tumtraf_1920	**30.80**	+ 8.54	+ 14.57

Table 6.6: Shifted ground truth + polymot 3D tracker: 3D detection quantitative comparisons on the test sequence of TUMTraf Intersection Dataset from both camera south1 and camera south2 with time-shifted ground truth labels and 3D polyMOT tracker. △no tracker column shows the improvement achieved by using 3D polyMOT tracker with time-shifted ground truth compared to only time-shifted ground truth. △baseline shows the improvement against the baseline model YOLOv7_coco with time-shifted ground truth labels and 3D polyMOT tracker.

Table 6.5 shows the additional quantitative improvement achieved when using a 3D tracker. The tracker employed in this study is Poly-MOT (a Polyhedral Framework for 3D Multi-Object Tracking) [Li+23], which has been exploited and integrated into the toolchain as part of a bachelor thesis by Vitus Becker. Poly-MOT tracks detections in 3D space and enhances the stability of vehicle position estimates by continuously predicting and updating the status of objects within a video sequence.

By using the Poly-MOT 3D tracker and evaluating against the time-shifted ground truth labels, the 3D mAP@[.10] can be further improved up to 15.51%.

Chapter 7

Conclusion

In conclusion, our thesis, conducted as part of the AUTOtech.agil project, has been dedicated to enhancing the 2D detector within the Providentia Mono3D object perception pipeline, with a focus on leveraging instance segmentation models to improve the system's final 3D perception performance, particularly on the TUM Traffic Intersection dataset.

Initially, we extend 1238 frames of the TUMTraf Intersection dataset with both modal and amodal instance masks to address the challenge of lacking segmentation annotations. Additionally, we present a simple approach for 2D annotation interpolation, utilizing algorithms like Jonker-Volgenant and linear interpolation to speed up the annotation process by at least five times.

Furthermore, our exploration into the state-of-the-art YOLOv8 segmentation model has yielded promising results. We extensively examine the effectiveness of pre-training on various datasets including COCO, KINS, and nuImages as well as fine-tuning on the annotated frames of the TUMTraf Intersection Dataset. Our experiments demonstrate the superiority of YOLOv8x pre-trained on COCO over the baseline YOLOv7 used in the Providentia live system, showing improvements of 3.6% in 2D mAP@[.5:.95] and 1.53% in 3D mAP@10. The fine-tuning of YOLOv8x on the TUMTraf Intersection dataset further improves performance, achieving a 10.5% improvement in 2D mAP@[.5:.95] and 5.88% in 3D mAP@10 compared to the baseline. Notably, this model excels in detecting large objects, addressing a significant limitation of the baseline YOLOv7 detector. The most significant performance boost, however, is achieved by the YOLOv8x model trained solely on the TUMTraf Intersection Dataset, with an 18.30% improvement in 2D mAP@[.5:.95] and 7.53% in 3D mAP@10 compared to the baseline. Together with the use of time-shifted ground truth labels and Poly-MOT 3D tracker, the best-trained model achieves a 3D mAP@10 of 34.02% on the test sequence of the TUMTraf Intersection dataset, which is a 17.79% improvement compared to the baseline YOLOv7 model trained on COCO.

However, since all annotated frames come from the TUMTraf Intersection Dataset only, the trained models are overfitting on this intersection setting. This has been proven in our experiments on a highway sequence.

In terms of speed, YOLOv8x exhibits accelerated inference speed. Once exported to TensorRT, this model achieves an inference speed of 200 FPS at a 640x640 resolution and 28 FPS at a 1920x1920 resolution, which is a 2.3 times speedup compared to the YOLOv7 model. This advancement allows for inference on full-resolution frames of 1920 while maintaining the system's real-time characteristics. The inference on full-resolution frames of 1920 has been shown to have 16.5% higher 2D mAP@[.5:.95] than inference at 640 resolution. This highlights the significance of resolution in object detection accuracy, especially for small and far away object detection.

Additionally, we investigate the performance implications of segmenting full object masks instead of only visible ones using the amodal segmentation model C2F. Surprisingly, our

experimental results indicate that utilizing amodal masks leads to a lower final 3D perception performance on the TUMTraf Intersection test set compared to solely relying on visible object masks detected by YOLOv8x.

Chapter 8

Outlook and Future Work

In this chapter, we outline potential future work to address identified improvements from our implementation and evaluation process:

- As mentioned in Section 4.2.3, the proposed 2D annotation interpolation pipeline utilizing the Jonker-Volgenant algorithm still sometimes generates erroneous object matching, requiring manual refinement afterward. Moreover, the currently used linear interpolation assumes a constant velocity of road users between frames. Future work could explore more advanced data association algorithms and leverage the existing Poly-MOT tracker to interpolate the 2D masks to more precise locations.

- Currently, only a limited number of frames, over one thousand, from the TUM Traffic Intersection dataset are annotated. While effective for training models specific to this intersection, this limited dataset poses a risk of overfitting. Consequently, models trained on this data exhibit improved performance at the intersection but suffer decreased performance on highway scenes compared to the baseline model YOLOv7. To address this limitation and train more generalized models, it is essential to annotate the remaining frames of the TUM Traffic Intersection Dataset and other releases of the TUM Traffic Dataset, such as the R00 and R01 release containing highway scenes and extended highway scenes in adverse weather conditions. The proposed 2D annotation interpolation pipeline can be refined and utilized for this task.

- Another potential approach is to train separate models specifically tailored for intersection and highway scenes. By doing so, each model can be optimized for its respective environment, potentially leading to improved performance in both settings. Particular attention should be paid to overfitting prevention. The model can overfit quickly on the small amount of annotated frames. To prevent this, closely monitor the validation performance.

- The YOLOv8 model pre-trained on nuImages for 50 epochs is still not powerful enough to have any improvement over the model pre-trained on COCO. Given the vastness of the nuImages dataset and its potential benefits for model performance, our ongoing efforts involve continuing to pre-train the YOLOv8x segmentation model on nuImages. We are aiming for 200 epochs, and when the performance improves compared to pre-train on COCO, we will also fine-tune the model on the TUM Traffic Intersection dataset.

- The trained YOLOv8 segmentation models can be exploited to run 2D multiple-object tracking. Ultralytics has implemented track algorithms, including BoT-SORT and ByteTrack, for the YOLOv8 model. The tracker produces the same output as segmentation, with an additional object ID that remains consistent for each object across consecutive

frames. Therefore, after training a good and generalized model, future work can exploit 2D multiple-object tracking and investigate the performance of the two supported track algorithms. Exploiting a 2D multiple-object tracker with YOLOv8 is expected to enhance the overall performance by providing more stable predictions.

- As discussed in Section 4.3, transmitting segmentation instance masks in the form of polygon contours may offer greater efficiency compared to using bit-packed masks. Therefore, future work could adjust the 3D detector's message reception to accept polygon contours, eliminating the need to transmit bit-packed masks altogether.

- As revealed in Section 5.5, the 3D bounding box labels of the TUMTraf Intersection Dataset are incomplete, resulting in false positives during model evaluation. Future work should involve revising these labels, particularly for categories such as PEDESTRIAN, MOTORCYCLE, and BICYCLE, to ensure more accurate model evaluation and performance assessment.

- A YOLOv8 segmentation model trained on full-resolution frames can detect big objects successfully, however, with multiple overlapping masks. Future work can implement post-processing, which does not filter out the overlapping masks but merges them to receive one final mask with good coverage over the detected object. Objects from the frames taken at the intersection do not overlap much with each other. Therefore, an IoU threshold of around 0.6 to 0.7 can be used to decide if two masks are referring to the same object and should be merged.

- Current evaluations of the C2F amodal segmentation model do not show improvement over only segmenting the visible part. The current C2F architecture crops frames based on each object's region of interest (ROI), resizes them to 256x256, extends them to amodal masks, and rescales them back to their original size within the original frame. However, as the system currently operates on the full resolution of 1920, 256x256 is relatively small, even for an object in the frame. This means that after upscaling a predicted amodal mask to its original size, the mask contour can be very unsmooth, which can affect the following bottom contour extraction step of the 2D-to-3D lifting stage of the system. In future work, the architecture of C2F can be adjusted to scale the ROIs to a bigger size.

Bibliography

[AT16] Arnab, A. and Torr, P. H. "Bottom-up instance segmentation using deep higher-order crfs". In: *arXiv preprint arXiv:1609.02583* (2016).

[Bew+16] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. "Simple online and realtime tracking". In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[Bir23] Birkner, J. "Monocular 3D Object Detection Using HD Maps". 2023.

[Blu22] Blumenthal, L. "Real-Time Monocular 3D Object Detection to Support Autonomous Driving". 2022.

[Cae+20] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. "nuscenes: A multimodal dataset for autonomous driving". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[Cae+] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. *Motional nuImages*. URL: https://www.nuscenes.org/nuimages (visited on 01/31/2024).

[CV19] Cai, Z. and Vasconcelos, N. "Cascade R-CNN: High quality object detection and instance segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 43.5 (2019), pp. 1483–1498.

[Che+19a] Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al. "Hybrid task cascade for instance segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4974–4983.

[Che+19b] Chen, X., Girshick, R., He, K., and Dollár, P. "Tensormask: A foundation for dense object segmentation". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 2061–2069.

[Cor+16] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[Cre+22] Creß, C., Zimmer, W., Strand, L., Fortkord, M., Dai, S., Lakshminarasimhan, V., and Knoll, A. "A9-Dataset: Multi-Sensor Infrastructure-Based Dataset for Mobility Research". In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 965–970. DOI: 10.1109/IV51971.2022.9827401.

[Don+21] Dong, B., Zeng, F., Wang, T., Zhang, X., and Wei, Y. "Solq: Segmenting objects by learning queries". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 21898–21909.

[23] *Eclipse eCAL enhanced Communication Abstraction Layer*. 2023. URL: https://eclipse-ecal.github.io/ecal/ (visited on 01/25/2024).

[Fan+21] Fang, Y., Yang, S., Wang, X., Li, Y., Fang, C., Shan, Y., Feng, B., and Liu, W. "Instances as queries". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 6910–6919.

[Fol+19] Follmann, P., König, R., Härtinger, P., Klostermann, M., and Böttger, T. "Learning to see the invisible: End-to-end trainable amodal instance segmentation". In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 1328–1336.

[Gao+23] Gao, J., Qian, X., Wang, Y., Xiao, T., He, T., Zhang, Z., and Fu, Y. "Coarse-to-fine amodal segmentation with shape prior". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 1262–1271.

[GLU12] Geiger, A., Lenz, P., and Urtasun, R. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.

[GBK22] Gu, W., Bai, S., and Kong, L. "A review on 2D instance segmentation based on deep neural networks". In: *Image and Vision Computing* 120 (2022), p. 104401.

[Hag14] Hagedorn, N. *COCO Data Format*. 2014. URL: https://cocodataset.org/#format-data (visited on 01/21/2024).

[Hag20] Hagedorn, N. *ASAM OpenLABEL Concept Paper*. 2020. URL: https://www.asam.net/index.php?eID=dumpFile&t=f&f=3876&token=413e8c85031ae64cc35cf42d0768627514868b2f#_introduction (visited on 01/17/2024).

[Hag23] Hagedorn, N. *Ultralytics Ultralytics YOLOv8 Docs*. 2023. URL: https://docs.ultralytics.com (visited on 01/18/2024).

[Har+14] Hariharan, B., Arbeláez, P., Girshick, R., and Malik, J. "Simultaneous detection and segmentation". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*. Springer. 2014, pp. 297–312.

[He+17a] He, K., Gkioxari, G., Dollár, P., and Girshick, R. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[He+17b] He, K., Gkioxari, G., Dollár, P., and Girshick, R. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[Hu+21] Hu, J., Cao, L., Lu, Y., Zhang, S., Wang, Y., Li, K., Huang, F., Shao, L., and Ji, R. "Istr: End-to-end instance segmentation with transformers". In: *arXiv preprint arXiv:2105.00637* (2021).

[JV88] Jonker, R. and Volgenant, T. "A shortest augmenting path algorithm for dense and sparse linear assignment problems". In: *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*. Springer. 1988, pp. 622–622.

[KTT23] Ke, L., Tai, Y.-W., and Tang, C.-K. "Occlusion-Aware Instance Segmentation Via BiLayer Network Architectures". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[LP20] Lee, Y. and Park, J. "Centermask: Real-time anchor-free instance segmentation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 13906–13915.

[Li+23] Li, X., Xie, T., Liu, D., Gao, J., Dai, K., Jiang, Z., Zhao, L., and Wang, K. "Polymot: A polyhedral framework for 3d multi-object tracking". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023, pp. 9391–9398.

[Lin+14] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[Lv+23] Lv, W., Xu, S., Zhao, Y., Wang, G., Wei, J., Cui, C., Du, Y., Dang, Q., and Liu, Y. "Detrs beat yolos on real-time object detection". In: *arXiv preprint arXiv:2304.08069* (2023).

[NHD17] Newell, A., Huang, Z., and Deng, J. "Associative embedding: End-to-end learning for joint detection and grouping". In: *Advances in neural information processing systems* 30 (2017).

[Qi+19] Qi, L., Jiang, L., Liu, S., Shen, X., and Jia, J. "Amodal instance segmentation with kins dataset". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3014–3023.

[Qui+09] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al. "ROS: an open-source Robot Operating System". In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[Red+16] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[RF18] Redmon, J. and Farhadi, A. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[RZ17] Ren, M. and Zemel, R. S. "End-to-end instance segmentation with recurrent attention". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6656–6664.

[RT16] Romera-Paredes, B. and Torr, P. H. S. "Recurrent instance segmentation". In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*. Springer. 2016, pp. 312–329.

[Sal+17] Salvador, A., Bellver, M., Campos, V., Baradad, M., Marques, F., Torres, J., and Giro-i-Nieto, X. "Recurrent neural networks for semantic instance segmentation". In: *arXiv preprint arXiv:1712.00617* (2017).

[SMZ19] Sekachev, B., Manovich, N., and Zhavoronkov, A. "Computer vision annotation tool: a universal approach to data annotation". In: *Intel [Internet]* 1 (2019).

[Sun+20] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2446–2454.

[Tan+21] Tan, J., Wang, K., Chen, L., Zhang, G., Li, J., and Zhang, X. "HCFS3D: Hierarchical coupled feature selection network for 3D semantic and instance segmentation". In: *Image and Vision Computing* 109 (2021), p. 104129.

[Tra+22] Tran, M., Vo, K., Yamazaki, K., Fernandes, A., Kidd, M., and Le, N. "AISFormer: Amodal Instance Segmentation with Transformer". In: *arXiv preprint arXiv:2210.06323* (2022).

[Van16] Vanholder, H. “Efficient inference with tensorrt”. In: *GPU Technology Conference*. Vol. 1. 2. 2016.

[WBL23] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 7464–7475.

[Wan+22] Wang, Y., Ahsan, U., Li, H., Hagen, M., et al. “A comprehensive review of modern object segmentation approaches”. In: *Foundations and Trends® in Computer Graphics and Vision* 13.2-3 (2022), pp. 111–283.

[Xia+21] Xiao, Y., Xu, Y., Zhong, Z., Luo, W., Li, J., and Gao, S. “Amodal segmentation based on visible region segmentation and shape prior”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 4. 2021, pp. 2995–3003.

[Xie+20] Xie, E., Sun, P., Song, X., Wang, W., Liu, X., Liang, D., Shen, C., and Luo, P. “Polarmask: Single shot instance segmentation with polar representation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 12193–12202.

[Yao+22] Yao, J., Hong, Y., Wang, C., Xiao, T., He, T., Locatello, F., Wipf, D. P., Fu, Y., and Zhang, Z. “Self-supervised Amodal Video Object Segmentation”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 6278–6291.

[Zha+20] Zhan, X., Pan, X., Dai, B., Liu, Z., Lin, D., and Loy, C. C. “Self-supervised scene de-occlusion”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 3784–3792.

[Zha+17] Zhang, X., Xu, W., Dong, C., and Dolan, J. M. “Efficient L-shape fitting for vehicle detection using laser scanners”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 54–59.

[Zhu+17] Zhu, Y., Tian, Y., Metaxas, D., and Dollár, P. “Semantic amodal segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1464–1472.

[Zim+23] Zimmer, W., Creß, C., Nguyen, H. T., and Knoll, A. C. “A9 Intersection Dataset: All You Need for Urban 3D Camera-LiDAR Roadside Perception”. In: *arXiv preprint arXiv:2306.09266* (2023).