Informatics

Einstufige Posenerkennung von verletzlichen Verkehrsteilnehmern

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Fabian Windbacher, BSc
Matrikelnummer 11809633

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Mag. Dr. Margrit Gelautz

Wien, 1. Oktober 2022

Fabian Windbacher   Margrit Gelautz

Informatics

Single-Stage Human Pose Estimation of Vulnerable Road Users

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Fabian Windbacher, BSc
Registration Number 11809633

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Mag. Dr. Margrit Gelautz

Vienna, 1st October, 2022

Fabian Windbacher   Margrit Gelautz

Erklärung zur Verfassung der Arbeit

Fabian Windbacher, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Oktober 2022

Fabian Windbacher

Danksagung

Danke Margrit, Matej, Michael, Georg und Nicolas für eure Hilfe!

Diese Arbeit wurde von dem Projekt SmartProtect (Projektnummer 879642) unterstützt, welches von dem Bundesministerium für Klimaschutz (BMK) im Rahmen des FTI-Programms Mobilität der Zukunft finanziert und der Österreichischen Forschungsförderungsgesellschaft (FFG) abgewickelt wurde.

Acknowledgements

Thank you Margrit, Matej, Michael, Georg and Nicolas for your help!

This thesis was supported by the project SmartProtect (Project Number 879642), which is funded through the Austrian Research Promotion Agency (FFG) on behalf of the Austrian Ministry of Climate Action (BMK) via its Mobility of the Future funding programme.

Kurzfassung

Die Posenerkennung von ungeschützten Verkehrsteilnehmern ist eine wichtige Aufgabe autonomer Fahrzeuge. Informationen über die Körperhaltung von Menschen im Straßenverkehr können dazu beitragen, deren Absichten einzuschätzen. Dies, wiederum, kann der effektiven Steuerung des Fahrzeuges dienen. Der Verkehrskontext unterscheidet sich von anderen üblichen Anwendungsfällen von Posenerkennung, etwa durch sein charakteristisches Szenenbild, die Schwierigkeit der Datensammlung, und die zusätzliche Verwendung von LiDAR-Sensoren. Einstufige Methoden für die Posenerkennung wurden in diesem Gebiet bisher wenig erforscht. Generell wurden derartige Ansätze bisher für weniger akkurat befunden. Sie haben jedoch andere vorteilhafte Eigenschaften, etwa das Potenzial für geringe Latenzzeiten. Wir untersuchen daher eine ausgewählte einstufige Methode zur Posenerkennung im Verkehrskontext. Erst kürzlich wurden mehrere geeignete, domänenspezifische Datensätze veröffentlicht. Die Positionen der Körperteile der beinhalteten Posen sind jedoch meist nicht mit Tiefeninformationen versehen. Nur das *Waymo Open Dataset* lokalisiert eine relativ kleine Anzahl an Posen auch in 3D. Deswegen haben wir uns entschienen, das Waymo Open Dataset zu erweitern. Hierfür nutzen wir die Bounding-Boxes, mit welchen alle sichtbaren Personen annotiert sind. Wir wenden eine zweistufige Methode für Posenerkennung auf diese Bounding-Boxes an. Zusätzlich nutzen wir die Tiefeninformation der verfügbaren LiDAR-Punktwolken. So erstellen wir mehr als eine Million Posen mit 3D Gelenkspositionen. In einem Vergleich mit den originalen Posendaten, beträgt der MPJPE weniger als 10 cm. Als Nächstes untersuchen wir eine ausgewählte einstufige Methode für Posenerkennung: *KAPAO*, ein neuartiger Ansatz, der sich durch eine besonders schnelle Inferenz auszeichnet. Wir untersuchen ihn anhand von 2D Verkehrsteilnehmer-Datensätzen. Wir variieren Trainings- und Inferenzparameter, wählen verschiedene initiale Modellgewichte, und modifizieren die Modellarchitektur. Unsere Resultate für die Datensätze *Tsinghua-Daimler Urban Pose* und *Berkley DeepDrive 100K* können mit den besten veröffentlichten Ergebnissen mithalten. Mit dem erweiterten Waymo Open Dataset und vielversprechenden KAPAO-Konfigurationen, formulieren wir zum Zweck der 3D-Posenschätzung *KAPAO 3D*: eine Variante von KAPAO, welche zusätzlich die Tiefe von Körperteilen vorhersagt. Als Vergleichswert dient ein Uplifting-Ansatz. Dieser führt 2D KAPAO Vorhersagen in 3D über, indem den Körperteilen naheliegende LiDAR-Punkte interpoliert werden. KAPAO 3D liefert etwas schlechtere Ergebnisse in Metriken wie AP, AR und MPJPE. Der visuelle Vergleich der beiden Ansätze zeigt jedoch, dass KAPAO 3D im Allgemeinen plausiblere Posen erzeugt.

Abstract

Human pose estimation of vulnerable road users is an important perception task for autonomous vehicles. Understanding the pose of traffic participants can provide valuable input for intention prediction, which, in turn, can guide the actions of the vehicle. This autonomous driving context for human pose estimation has a number of special characteristics, such as its distinctive scenes, the inherent difficulty of data collection and the prominence of LiDAR sensors. Single-stage human pose estimation approaches have hardly been studied in this setting so far. While they have generally been less accurate than two-stage methods in the past, they showed other desirable qualities, such as the potential for low-latency applications. We propose to study a designated single-stage method in the autonomous vehicle domain. Recently, multiple public benchmark datasets were released for that specific purpose. Depth information for the poses, however, is still largely unavailable. To our knowledge, only the *Waymo Open Dataset* localizes a small number of poses in 3D. Therefore, we decide to extend the Waymo Open Dataset. To that end, we leverage 2D and 3D bounding boxes that are present for any visible person in the dataset. Using a state-of-the-art two-stage model on those bounding boxes, as well as depth information from the LiDAR point clouds, we create more than one million poses with 3D joint positions. Evaluating the quality on a holdout set of original labels, we report an MPJPE of less than 10 cm. Next, we focus on our single-stage model of choice: *KAPAO*. It is a state-of-the-art human pose estimation method, which stands out due to its inference speed. We study its performance on 2D vulnerable road user benchmark datasets. We vary training and inference parameters, choose different initial checkpoints, and even consider an architecture modification. Evaluating on *Tsinghua-Daimler Urban Pose* and *Berkley DeepDrive 100K*, we find KAPAO to be competitive with the best reported results. Having access to a large-scale dataset and promising configurations of KAPAO, we finally study 3D pose estimation in the domain. We propose *KAPAO 3D*, a variant of KAPAO that additionally predicts the depths of joints. This is compared against a baseline uplifting approach, in which 2D KAPAO predictions are lifted into 3D using close-by LiDAR points in a post-processing step. KAPAO 3D performs slightly worse than the baseline in metrics like AP, AR and MPJPE. Closer visual inspection of the errors made, however, shows that the 3D model generally produces more plausible poses.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Problem Statement

Perception is a central feature of autonomous vehicles (AV). To successfully navigate the world, it is paramount that the machines sense their environment and other actors accurately. Vulnerable road users (VRUs), such as pedestrians and cyclists, whose safety can only be guaranteed if they are detected in time are of particular interest.

Mere detection may not always provide enough information. Human pose estimation (HPE) gives more fine-grained insight into the pose and status of the person. Here, the positions of individual body parts (also referred to as joints or keypoints), such as the head, shoulders, elbows, wrists, knees, and feet, are estimated. Several tasks involving VRUs can greatly profit from such information, including gesture recognition [FZG+20], crossing prediction [CYQW19] and trajectory estimation [LJN+19].

Often, a distinction is made between the following two sets of approaches to HPE: *two-stage methods* first detect people, and apply pose estimators to the cropped image patches containing individuals; *single-stage* methods process the whole image and output poses of multiple people. Traditionally, the former achieve better accuracy. The latter, however, usually perform better in crowded scenes [CWP+18] and are potentially more desirable in terms of simplicity and efficiency.

Applications of HPE vary in their setting. The AV context has distinct characteristics that differ from those of other typical HPE domains. For one, VRUs often appear small in the image, when they are located far from the camera and near the edges of typical driving scenes. In these cases, there is frequent use of LiDAR sensors, which produce sparse depth maps of their surroundings.

The outdoor setting makes automatic collection of accurate data (as produced by motion capture systems, for example) difficult. Additionally, since the distance and angle between the car and VRUs have an impact on the associated danger level, estimation of joint positions in 3D is desirable (as opposed to only localizing the keypoints within a 2D image). Large scale pose annotation by humans is significantly more difficult in 3D, however, due to the difficulty of navigating 3D space effectively on 2D displays. The modalities which allow for 3D annotation (e.g. point clouds) are less familiar than images for 2D, and navigation within 3D space is less intuitive and generally more laborious.

Recently, several HPE datasets focusing on traffic scenes have been published. The datasets *Tsinghua-Daimler Urban Pose (TDUP)* [WYW$^+$21], *Berkley DeepDrive 100K (BDD100K)* [YCW$^+$20], and *Waymo Open Dataset (WODS)* [SKD$^+$20] provide 2D human pose labels (i.e., position information of multiple joints that make up a pose) in a diverse set of scenes captured from cars in motion. WODS additionally includes a small number of 3D labels and LiDAR data. Its video sequences are fully annotated with 2D and 3D bounding boxes (i.e., each identifiable person's location is defined by a bounding box). Poses, however, are not exhaustively annotated: in one frame none, some or all people may have pose labels. In a naive application such partial annotation could lead to single-stage methods being wrongly penalized for the detection of unlabelled individuals.

Still, single-stage HPE methods have not been studied extensively in the AV domain. Leaderboards of the mentioned 2D benchmark datasets are largely dominated by two-stage approaches[1,2]. The two single-stage approaches listed for TDUP [1] are consistently outperformed by the other, two-stage methods. And for 3D VRU HPE, we are not aware of any pre-existing work[3].

This may be related to the limited availability of 3D HPE data. WODS features only a small number of 3D pose labels and has the undesirable characteristic that human poses are only selectively annotated. We aim to address the lack of single-stage applications by first addressing the limited 3D data availability. This aim of the thesis is explained in more detail in the following section.

1.2 Aim of the Work

In this work, we want to study single-stage HPE of VRUs. To overcome the limitations of 3D pose label availability, we first extend WODS with automatically generated labels. These generated labels are subsequently referred to as pseudo-labels, since they are automatically produced and not guaranteed to be entirely correct. Then, we experiment with multiple configurations of a state-of-the-art single-stage HPE model on the previously mentioned 2D VRU benchmark datasets. Finally, we adapt that single-stage method for the 3D HPE task on VRUs. The extended WODS, that we created for this purpose, enables its training and evaluation.

[1]`http://urbanpose-dataset.com/info/Datasets/198` [retrieved on 18.8.2022]

[2]`https://github.com/SysCV/bdd100k-models/tree/main/pose` [retrieved on 18.8.2022]

[3]Works such as [ZSG$^+$21, FGS$^+$21, KJZ$^+$18, GWH19] consider 3D VRU HPE, but do not fit our conception of single-stage

We address the following research questions:

1. What is the accuracy of our generated 2D and 3D (pseudo-)labels for the Waymo Open Dataset?

2. How does our chosen single-stage human pose estimator perform across selected 2D vulnerable road user pose estimation benchmarks?

3. How accurate is our 3D adaptation of the single-stage human pose estimator from Question 2 applied to the dataset from Question 1?

To enable the study of 3D pose estimation, we extend WODS [SKD+20] under Question 1. For Research Question 2, a 2D single-stage pose estimator, namely *KAPAO (Keypoints as Poses and Objects)* [MVWM21], is evaluated in multiple configurations on the 2D VRU datasets TDUP [WYW+21] and BDD100K [YCW+20]. Having established a suitable dataset and a single-stage estimator for Questions 1 and 2, respectively, we now have the means to address Question 3, the analysis of that pose estimation method adapted to 3D. To that end, we modify the architecture to directly regress 3D poses with and without LiDAR input data. We compare this to a baseline method, that lifts 2D predictions to 3D using the depth provided by LiDAR data in a post-processing step.

To answer the research questions formulated above, we make use of several evaluation metrics. Performance in a particular 2D or 3D setting can be characterized via various HPE metrics, such as object keypoint similarity (OKS) based average precision (AP) [LMB+14], log average miss rate (LAMR) [DWSP12], or mean per joint position error (MPJPE) [IPOS14]. 2D ground truth exists in the form of public benchmark datasets, such as TDUP [WYW+21] and BDD100K [YCW+20]. The (extended) WODS [SKD+20] provides LiDAR data and 3D HPE labels.

1.3 Methodological Approach

In this section, we outline the procedure we apply to answer our research questions. We perform evaluations to answer all three questions using quantitative experiments. These experiments are in the form of predictive performance tests on holdout splits of suitable datasets. Furthermore, results are checked qualitatively, by visually inspecting random data samples.

First, we extend the Waymo Open Dataset. As a first step towards generating exhaustive 3D labels, we expand the 2D label coverage. To that end, an established two-stage HPE model architecture, *HRNet* [SXLW19], is chosen. We train it on the original 2D human pose labels of WODS. Then, we apply the trained pose estimator to the image patches defined by ground truth person bounding boxes, replacing the initial person detector stage. Note that ground truth person bounding boxes are present for all individuals and in all frames, which is not the case for the original joint annotations. As a result

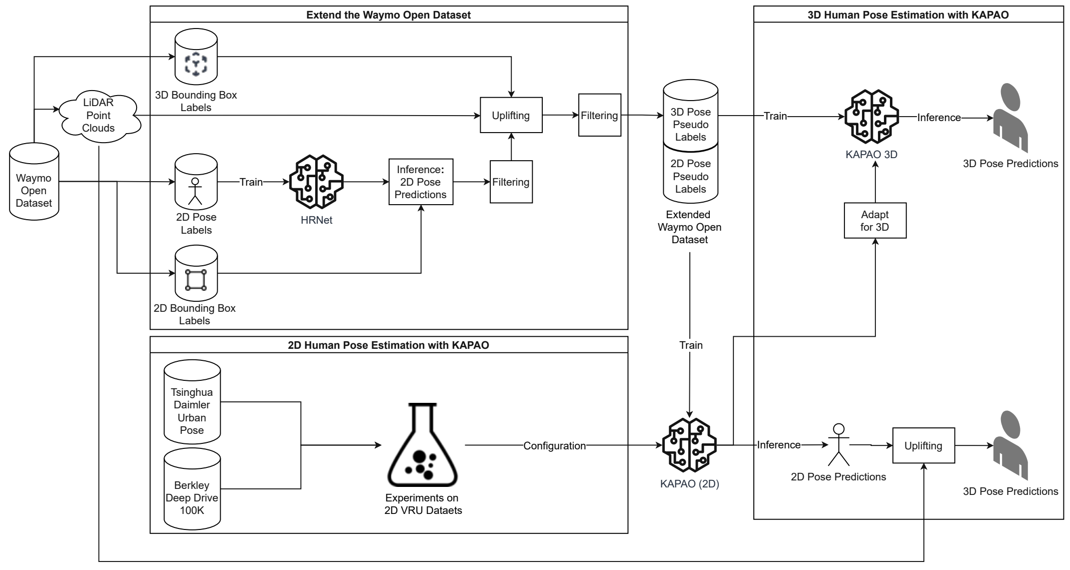


Figure 1.1: Overview of the content of this thesis.

of having those ground truth boxes, the pose estimation takes place exclusively on the image patches that contain exactly one person, effectively reducing the problem to the joint localization of a single person. This allows us to consider the models' predictions as automatically generated 2D pseudo-labels. Furthermore, a lifting procedure based on LiDAR data [ZSG+21] is employed to transform 2D keypoints to 3D. After filtering potentially bad predictions using various heuristics, we have access to more than one million 2D and 3D pseudo-labels. The accuracy of the generated labels is evaluated by comparing against a holdout set of original 2D and 3D HPE annotations, addressing Research Question 1.

As the basis for further, single-stage analysis, we choose *KAPAO* [MVWM21]. It is a recently published, single-stage HPE architecture that reportedly shows a particularly good accuracy-latency trade-off. We train the chosen model on the selected 2D VRU HPE benchmark datasets, TDUP and BDD100K. We adjust the training procedure, hyperparameters or even architectural details as needed to achieve good performance. Different model variants associated with these adjustments are evaluated regarding their speed and accuracy on the provided evaluation sets, addressing Research Question 2.

The established KAPAO variants, in conjunction with the extended WODS, provide the basis for tackling question three. We now focus on the task of 3D HPE. We propose KAPAO 3D: an adaptation of the original KAPAO architecture that additionally predicts the depth of keypoints. Two variants of that model are considered. One that only takes monocular images into account. The other additionally takes a sparse depth map provided by the LiDAR sensors as input. We compare those against a lifting baseline, which uses the original KAPAO to predict 2D poses and lifts those to 3D using close LiDAR points in a post-processing step. Our proposed methods are evaluated and compared regarding their accuracy on a holdout split of the extended WODS.

An overview of this work's procedure is given in Figure 1.1. The individual steps are grouped with respect to the research question they pertain to. Elements outside of the classifications illustrate how the major blocks of this work are interrelated.

1.3.1 Structure

Finally, we present this work's structure. In Chapter 2 we outline the current state-of-the-art, establish some important notions and contextualize our work within the literature. Then, in Chapters 3, 4, and 5 we present our main findings regarding the extension of WODS, 2D HPE, and 3D HPE that allow us to answer the Research Questions 1, 2, and 3, respectively. Finally, in Chapter 6 we give concluding thoughts and outline promising continuations of this work.

CHAPTER 2

Related Work

In this chapter, we discuss existing work that is relevant to this thesis. First, we summarize the state of the art in human pose estimation, where we describe KAPAO in more detail. Then, we introduce the datasets we apply our models to. Finally, we explain the evaluation metrics that are used throughout this work.

2.1 Human Pose Estimation

HPE is the task of localizing the joints (also referred to as keypoints) of a person. This can be performed in various settings, on different input modalities and with 2D or 3D target coordinate spaces. When we refer to HPE in this thesis, we always consider multi-person pose estimation, where, in principle, the joints of any number of people ≥ 0 are localized and grouped into individual poses. In contrast, single-person pose estimation expects exactly one person to be depicted in a given image. In terms of input modalities, we restrict ourselves to images and LiDAR point clouds only. We consider both 2D HPE, i.e., estimating image coordinates in pixels, and 3D HPE, where the joints' depths in relation to some real space (e.g. meters) are additionally estimated.

In the literature, distinctions between HPE methods are made in various ways, including *single-stage* vs. *two-stage* and *top-down* vs. *bottom-up*. The exact definitions of these partitions differ. We adopt the conceptions given in [MVWM21]. Therefore, *single-stage* methods rely on only one pass of one neural network to estimate poses from a given image. *Two-stage* methods, on the other hand, split the task into two separate neural networks. Generally, the task is split in a *top-down* fashion (therefore we use *top-down* and *two-stage* interchangeably). The first stage detects individual people (i.e., a person detector) and the second estimates the joints of a person based on the image patch defined by the detection of the initial stage (i.e., a single-person pose estimator). *Bottom-up* methods use a neural network to predict all joints in the image (plus additional grouping information) and form poses in non-neural post-processing steps. Therefore, bottom-up methods are considered a subset of single-stage methods. In this work, we are primarily concerned with single-stage methods.

2.1.1 Single-Stage Human Pose Estimation

We outline the current state of the art in single-stage 2D HPE methods. First, we list prominent bottom-up methods, such as *HigherHRNet* [CXW+20], which is built on the backbone network *HRNet* [SXLW19]. It outputs keypoint heatmaps that are combined to form poses using associative embeddings [NHD17]. Variants, such as *CenterGroup* [BKL21] and *SWAHR* [LWH+21], adapt and improve HigherHRNet further. *DEKR* [GSX+21] extends existing approaches that regress person centers and keypoint offsets with adaptive convolution operations to achieve competitive performance. For 3D HPE there exist similar bottom-up conceptions [FLC+20, ZMZ+18a].

KAPAO [MVWM21] represents a single-stage method that is not bottom-up. It adapts the prominent object detection framework *YOLOv5* [JCS+22] for HPE, by modelling poses and keypoints as objects to detect. Grouping-free single-stage approaches have been conceived for 3D as well [ZMZ+18b, WNQ+22].

KAPAO

Keypoints and Poses as Objects (KAPAO) [MVWM21] is a recently proposed single-stage human pose estimation method. It deviates from the common approach of regressing keypoint heatmaps. Instead, it adapts a grid-based detection framework popular in object detection.

The architecture is illustrated in Figure 2.1. An Image (I) is fed into a feature extraction network (N), such as a CNN. The network outputs a number of output grids of different resolutions ($\hat{G}$). Each grid cell of each grid can make a number of object predictions, as is common in grid-based object detection.

KAPAO differs in the nature of the object definitions. KAPAO models two kinds of objects: pose and keypoint objects. The latter are defined in the usual object detection fashion: the class (i.e., what kind of keypoint is detected), size (a constant value provided as a hyperparameter) and location of a bounding box enclosing the keypoint. The pose objects represent the whole pose at once. Those extend a person object (i.e., one where the above definition relates to the bounding box enveloping the person of interest) with additional (x_i, y_i) coordinates for each keypoint i in the pose. To take advantage of efficient matrix computation, keypoint objects are padded with zeros to match the size of pose objects.

These pose and keypoint predictions ($\hat{O}^p$ and $\hat{O}^k$, respectively) are made by the cells of the output grids ($\hat{G}$). Then, non-maximum suppression (NMS) is performed to eliminate duplicate predictions. The resulting pose objects can additionally be refined with the keypoint predictions' locations using a simple matching algorithm (ϕ).

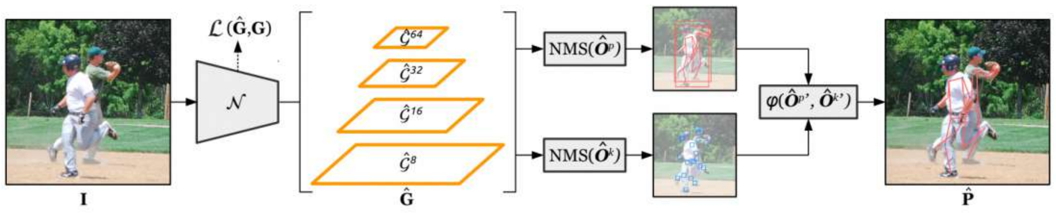


Figure 2.1: Illustration of the KAPAO architecture. Reproduced from [MVWM21].

The authors report state-of-the-art performance with a particularly good accuracy-latency tradeoff for KAPAO.

2.1.2 Two-Stage Human Pose Estimation

We outline the state of the art regarding two-stage HPE as well. Among the most prominent approaches are heatmap-based ones [SXLW19, FXTL17, XWW18, CWP+18, LWY+19, ZZD+20]. Each joint is modelled as a 2D Gaussian distribution centered at its coordinates. The model then outputs a confidence grid for each joint, which is penalized with respect to that distribution during training. Inference is performed by selecting the maximum value of the grid. Notably, HRNet [SXLW19], which we employ in Chapter 3, is such a method.

Heatmap-based approaches rely on high resolution output grids to keep the quantization error low. This, however, can make them computationally inefficient. Direct keypoint regression methods do not rely on such output grids. They aim to directly predict the joint coordinates in various ways. These methods achieved competitive performance recently by disentangling horizontal and vertical coordinates [LYZ+21], employing a specialized loss function [LBZ+21], and using transformers [LWZ+21]. Top-down approaches with similar ideas exist for 3D HPE [MSM+18, MCL19, RWS20].

Furthermore, there is previous work on HPE of VRUs, such as [WFX+19]. Some of the previously listed VRU HPE datasets also report the performance of baseline methods [YCW+20, WYW+21][1]. There have also been endeavours to optimize for TDUP specifically [KRC+21].

Then, there are methods that incorporate LiDAR data as well [ZSG+21, FGS+21]. These focus specifically on 3D HPE in the AV context. Both papers address the limitations in dataset availability by employing weak supervision through pseudo-labels. Our work differs from those publications by employing a single-stage method.

2.2 Datasets

We discuss the datasets relevant to this work. Find a comparison of relevant characteristics between the VRU datasets in Table 2.1.

[1]Reported baselines are mostly two-stage. The few that are single-stage underperform in comparison.

Characteristics	TDUP	BDD100K	WODS	WODS++
#2D-Labels	$\sim$ 75K	$\sim$ 20K	$\sim$ 200K	($\sim$ 1.2M)
#3D-Labels	0	0	$\sim$ 10K	($\sim$ 1M)
Camera	✓	✓	✓	✓
LiDAR	✗	✗	✓	✓

Table 2.1: Characteristics of the different VRU HPE datasets [WYW+21, YCW+20, SKD+20] based on training and validation splits. Parentheses in the case of WODS++ are added as a reminder that those are pseudo-labels.

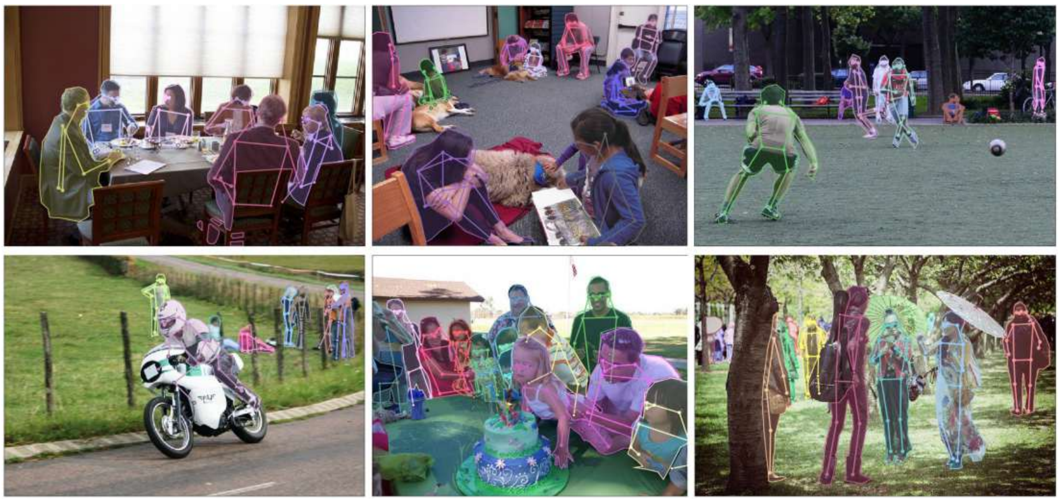


Figure 2.2: Collection of sample images with pose (and segmentation) labels from COCO. Colors are randomly assigned to the different people. Reproduced from [LMB+14].

2.2.1 Common Objects in Context

Common Objects in Context (COCO) [LMB+14] is a human-annotated dataset for object detection, segmentation and captioning. Importantly, it also provides human pose labels. This part of the dataset is commonly referred to as *COCO Keypoints*. We show a collection of sample images in Figure 2.2.

We do not use it in this work for training or evaluation, since it does not specifically focus on the AV domain. However, due to its prominence as a benchmark (e.g. [MVWM21, SXLW19, CXW+20]) it has relevance to our work via the data format and evaluation procedure it established. Furthermore, we use checkpoints trained on its pose and object detection labels as initial weights in Chapter 4.

2.2.2 Tsinghua-Daimler Urban Pose

Tsinghua-Daimler Urban Pose (TDUP) [WYW+21] is a large-scale 2D VRU pose estimation dataset. It contains monocular images collected from cars in urban Chinese

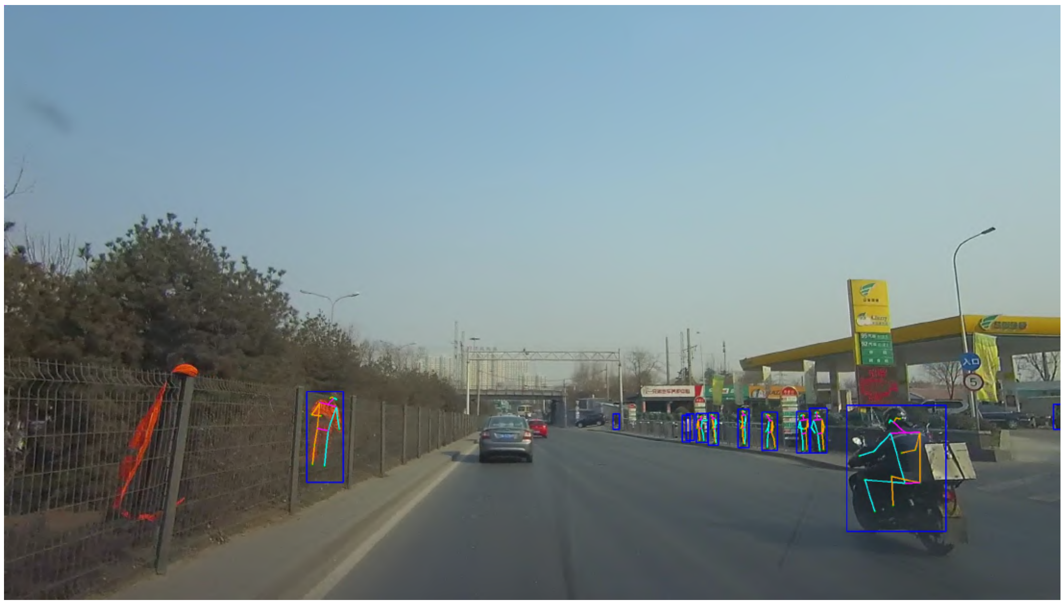


Figure 2.3: Sample image with pose labels from TDUP. Reproduced from [WYW+21]. Colors distinguish left, right, torso and head keypoints.

environments. TDUP consists of $21,000$ images with $\sim 90,000$ person annotations, $\sim 75,000$ of which additionally have joint positions. The dataset is split into train, validation and test set. Annotations are not publicly available for the test set but prediction quality can be evaluated via submission to an evaluation server. We display a sample image and its labels in Figure 2.3.

More than half of the poses in the dataset are of riders; i.e., people that ride a motorcycle, bike, or similar. It uses the same keypoint definitions (i.e., specifics of which joints are annotated) as COCO Keypoints. This dataset is used to evaluate our methods for 2D HPE.

2.2.3 BDD100K

Berkley DeepDrive 100K (BDD100K) [YCW+20] is a large-scale dataset for heterogenous driving tasks. It consists of $100,000$ sequences collected across the United States. It is hand-labelled to support a diverse set of different tasks including object detection, tracking, and segmentation. Importantly, it was extended with VRU pose labels at the end of 2021. We provide a sample image in Figure 2.4.

There are $\sim 27,000$ pose instances across $\sim 14,000$ images. These are split into train, validation and test set. Annotations are not publicly available for the test set but prediction quality can be evaluated via submission to an evaluation server. BDD100K's keypoint definitions are similar to COCO and TDUP. The keypoints have a similar

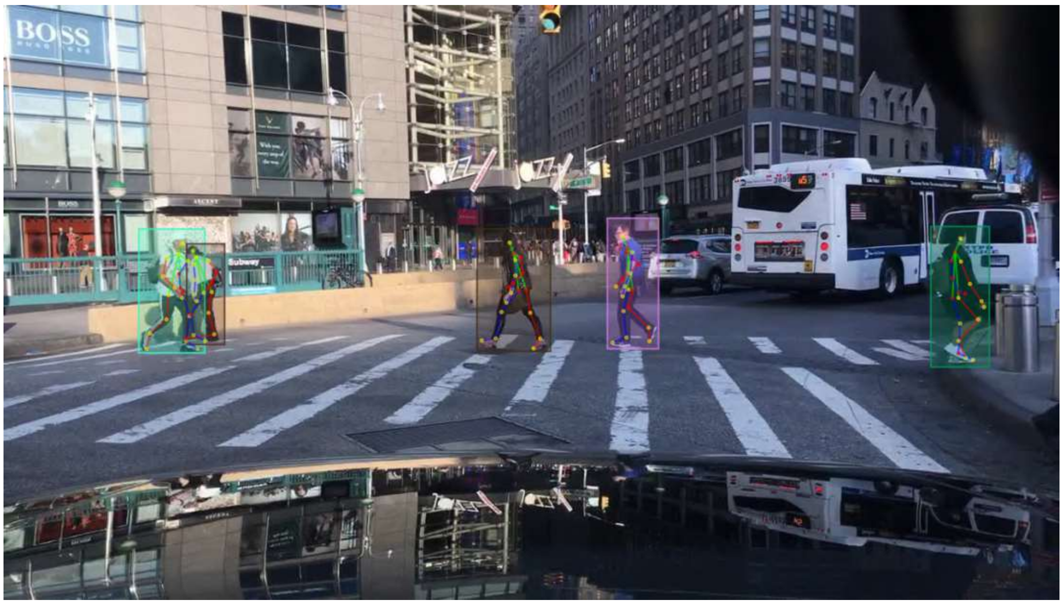


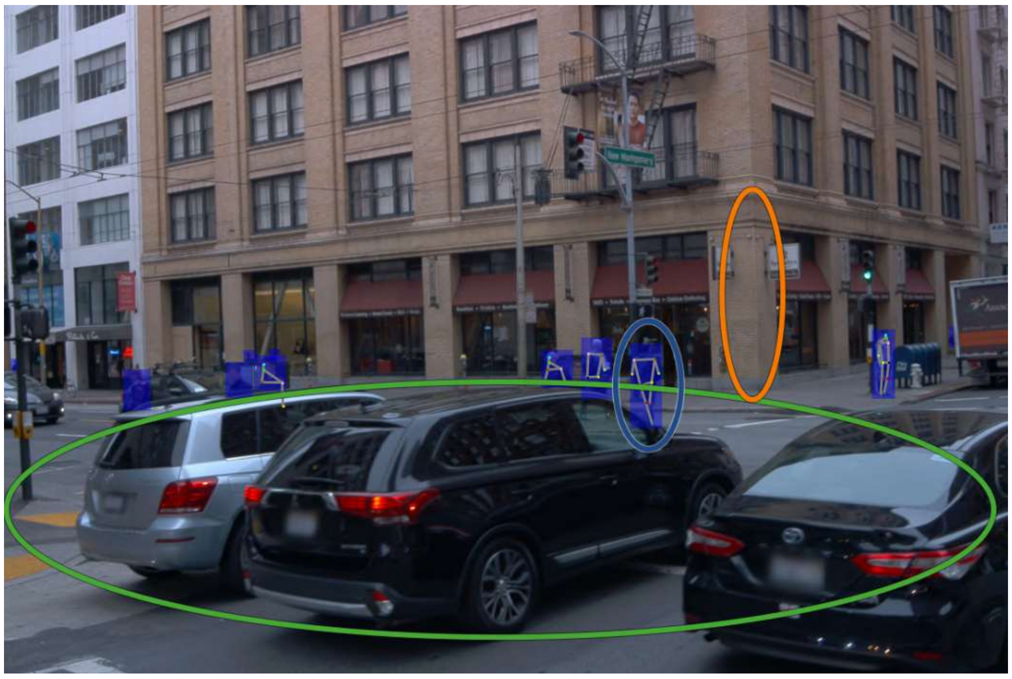
Figure 2.4: Sample image with pose labels from BDD100K. Box colors are randomly assigned to the different people. Skeleton colors distinguish left, right and torso (including head) keypoints. Reproduced from [YCW+20].

granularity and also span the whole body, but do not relate to the exact same body parts. This dataset is also used to evaluate our methods for 2D HPE.

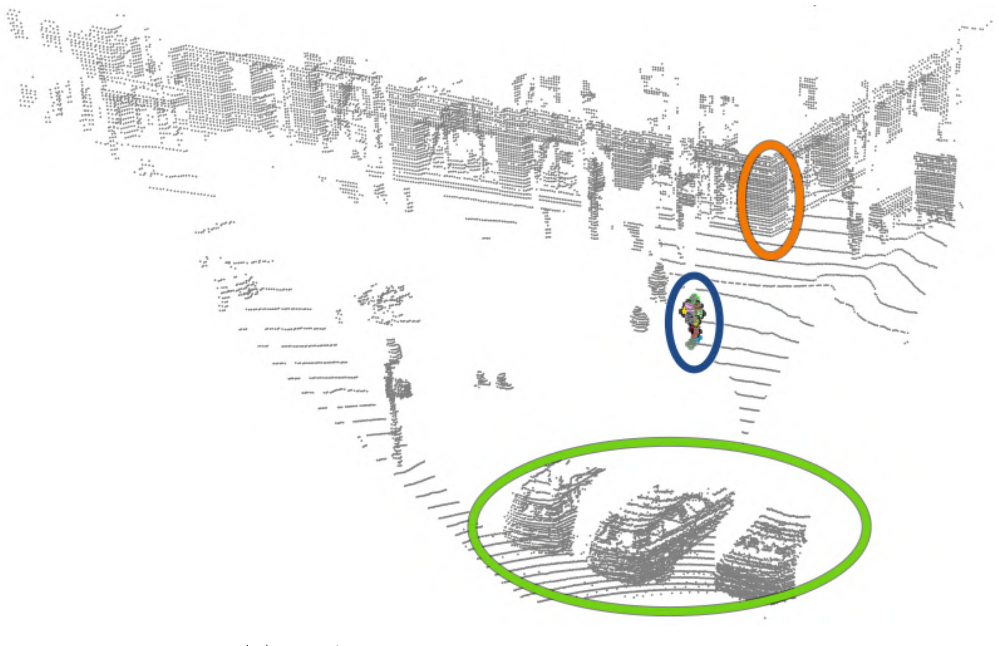
2.2.4 Waymo Open Dataset

Waymo Open Dataset (WODS) [SKD+20] is a large-scale dataset developed to aid in autonomous driving tasks. It is split into *Motion* and *Perception* to serve different applications. When we speak of WODS, in the context of this thesis, we refer to the *Perception* part.

WODS consists of $1,150$ sequences that are 20 seconds long, aligned sensor data from LiDAR and cameras, and various 2D and 3D labels. It was extended in March 2022 with human keypoint annotations. It contains $\sim 170,000$ 2D labels and $\sim 10,000$ 3D annotations split into train and validation sets. As opposed to the object detection labels (2D and 3D bounding boxes), pose labels are not always present. In a particular frame none, some or all people can have pose annotations. We provide an example scene in Figure 2.5. We extend this dataset with large-scale 3D HPE labels in Chapter 3.



(a) Image with 2D pose labels.



(b) LiDAR pointcloud with 3D pose labels.

Figure 2.5: Sample scene from WODS [SKD$^+$20]. Each keypoint type has a different color, consistent across people. Colored ellipses mark corresponding objects in image and point cloud to aid orientation.

2.3 Evaluation Metrics

We discuss the HPE evaluation metrics relevant to this work. Their choice is primarily determined by the datasets used. Therefore, other prominent metrics (e.g. [FMJZ08, YR13]) are omitted here.

2.3.1 Object Keypoint Similarity

Object keypoint similarity (OKS) is a metric used to measure the similarity between a predicted and target pose [LMB+14]. It is generally used as part of a further, aggregate performance measure. We provide the formula to compute the OKS score in Equation 2.1.

$$s_{OKS} = \frac{\sum_{i \in J} e^{-\frac{d_i^2}{2s^2k_i^2}} \delta(v_i > 0)}{\sum_{i \in J} \delta(v_i > 0)} \tag{2.1}$$

Here, $\boldsymbol{d}$ is a vector of euclidean distances between corresponding joints of two poses. The vector $\boldsymbol{v}$ denotes the annotation state of the joints (here, we assume that fully visible is coded as 2, occluded but annotated as 1, and unannotated as 0). The evaluator function $\delta(v_i > 0)$ is 1 if the boolean expression evaluates to true; and 0 otherwise. Therefore, the metric only takes into account joints that are annotated. In the exponent, sk_i acts as the standard deviation; where s denotes the object scale and k_i a pre-estimated joint-specific positional variation. Finally, J is the index set encompassing all joints (i.e., $|J| = dim(d)$).

2.3.2 Average Precision and Average Recall

Average precision (AP) is an aggregate metric adopted from object detection and occurs in several variations. It represents the area under the precision-recall curve. We provide the formula to compute AP in Equation 2.2. It uses precision (pr) and recall (re), which are defined in Equations 2.3 and 2.4, respectively.

$$AP = \frac{1}{|R|} \sum_{r \in R} max_{re(c) \leq r} pr(c) \tag{2.2}$$

$$pr(c) = \frac{tp(c)}{tp(c) + fp(c)} \tag{2.3}$$

$$re(c) = \frac{tp(c)}{tp(c) + fn(c)} \tag{2.4}$$

Here, $tp(c)$, $fp(c)$, and $fn(c)$ refer to the count of true positives, false positives, and false negatives, respectively, at some confidence threshold c. Which of these classes a prediction belongs to is generally determined by the OKS score at a certain threshold. The set of reference points R is equally distributed in $[0, 1]$, such as $r \in \{0, 0.1, ..., 0.9, 1\}$.

To fit the HPE task, the original intersection-over-union (IoU) similarity measure is replaced with the OKS measure explained previously. In the COCO Keypoints [LMB+14] benchmark, AP represents an averaged score over the OKS match thresholds $\{0.5, 0.55, ..., 0.9, 0.95\}$ (unless indicated otherwise with e.g. $AP_{.50}$, which means only the AP score at the OKS threshold of 0.5 is considered). The same holds for TDUP [WYW+21] and BDD100K [YCW+20].

Average recall (AR) works similarly. However, where AP considered the area under the precision-recall curve, AR considers the recall-OKS curve (and originally in object detection the recall-IoU curve).

2.3.3 Log Average Miss Rate

Log average miss rate (LAMR) is a measure of the miss-rate as a function of the false positives per image, also adapted from object detection [DWSP12]. We provide the formula to compute LAMR in Equation 2.5. It depends on the miss-rate and false-positives-per-image, which are given in the Equations 2.6 and 2.7, respectively.

$$LAMR = \exp\left(\frac{1}{|F|}\sum_{f\in F}\log\Big(mr(argmax_{fppi(c)\leq f}\, fppi\,(c))\Big)\right) \tag{2.5}$$

$$mr(c) = \frac{fn(c)}{tp(c) + fn(c)} \tag{2.6}$$

$$fppi(c) = \frac{fp(c)}{\#img} \tag{2.7}$$

Here, $fn(c)$, $tp(c)$, and $fp(c)$ again refer to the count of false negatives, true positives, and false positives at a confidence threshold c, respectively. F is a set of thresholds, acting as a limit on the $fppi(c)$ maximization (generally 9 evenly spaced thresholds in log space between $10^{-}2$ and 10^{0}). Matching of prediction and target (for determining the confusion matrix) may again be done using OKS. In TDUP [WYW+21] the confidence threshold is 0.5.

2.3.4 Mean Per Joint Position Error

Mean per joint position error (MPJPE) is an aggregate distance measure over all poses [IPOS14]. For each pair of predicted and target pose, the L^2-distance between corresponding joints is computed. Then, this measure is averaged over all joints, and over all poses. We also consider variants where the distances are only averaged across a subset of joints to give insight into body-part specific accuracy.

CHAPTER 3

Extending the Waymo Open Dataset

This chapter describes the process and results of extending the Waymo Open Dataset. First, the procedure is outlined. Then, the experiment setup is discussed and the results are presented.

3.1 Implementation

As discussed previously, 3D pose labels are not available on a large scale. The only dataset to feature them at all, to our knowledge, WODS, only provides a small number of them. Additionally, WODS does not annotate scenes fully with pose labels, which can be detrimental to naive applications of single-stage HPE methods. Therefore, as part of our work, we aim to generate exhaustive pose pseudo-labels for WODS. To that end, the existing (complete) bounding box ground truth is leveraged. We train a single-person pose estimator on the original keypoint labels, and infer the poses of all people (as located by their bounding box labels). We call our newly generated extended dataset *WODS++*.

3.1.1 Extraction

First, we acquire the Waymo Open Dataset, specifically version 1.3.2 of the Perception variant[1]. We then iterate over the training and validation set and extract the following information:

1. All 2D person and cyclist objects are extracted. This always includes their bounding boxes, and, if available, keypoint information. Further, if available, the associated 3D object information is stored, that is, bounding box and keypoints (if available).

[1] `https://waymo.com/open/download/` [retrieved on 8.6.2022]

2. We save the image of each camera that has labels in a given frame. Additionally, we store the 3D LiDAR points in the field of view of that camera, as well as their 2D projections on the corresponding image.

The original WODS annotations are split into a training and a validation set. We further split the original validation set into: one for model selection and one for gauging the final performance. We randomly partition the original validation set's images into two sets of equal size. Subsequently, they are referred to as the *development* and *test* split, respectively.

3.1.2 Modelling

We then fit a two-stage HPE model on the extracted 2D keypoints. The HRNet [SXLW19] architecture is chosen. Specifically, the mmpose COCO implementation of HRNet[2], which achieves top scores on the COCO Keypoints benchmark[3]. Dataset specific parameters are adjusted to fit the needs of WODS, such as the keypoint configuration and their estimated standard deviations (which are taken from the WODS toolkit[4]). Other minor adjustments include:

1. adjusting the batch size to 16 in training to fit within hardware limits.
2. setting the number of training epochs to 250.
3. computing evaluation metrics every 5 epochs, as opposed to every 10.

Refer to Section 3.2 for a discussion of the results and Figure 3.5 in particular for a visualization of the training progress over epochs.

3.1.3 Label Generation

Inference

Having trained the model, we now use it to create the pseudo-labels. We make use of the bounding box annotations of WODS, that are present for all people, in all frames. They serve as a replacement for the initial person detector stage, that is usually needed with two-stage approaches. We infer poses using our trained HRNet on each image patch defined by a ground truth bounding box.

[2]https://github.com/open-mmlab/mmpose/blob/4f5ec97c9991d4fda1c2ff8133bab8337e9663f7/configs/body/2d_kpt_sview_rgb_img/topdown_heatmap/coco/hrnet_w48_coco_384x288.py [retrieved on 13.06.22]

[3]https://mmpose.readthedocs.io/en/latest/benchmark.html [retrieved on 13.06.22]

[4]https://github.com/waymo-research/waymo-open-dataset [retrieved on 13.06.22]

Filtering

While in principle we can estimate poses for every person that has an associated bounding box, we filter the results; i.e., we remove certain inferred poses. This is necessary, since there are cases where person detection is feasible, yet pose estimation is not (e.g. for very small depictions of people). Furthermore, the benefit of using generated keypoint labels with particularly low confidence for the training or evaluation of other models is questionable.

We use multiple heuristics to do this. We list the chosen filtering conditions and give the reasoning behind our decisions:

1. Inspired by TDUP [WYW+21], we only provide labels for bounding boxes with some minimum height. To choose the threshold value, we look at the boxes for which original labels are provided. This is for two reasons: **a)** we cannot meaningfully evaluate our model performance for sizes that are not well reflected in the original labels, and **b)** a small number of annotations below a certain size may reflect that detection of keypoints is infeasible (due to low resolution, for example). Therefore, we plot a histogram of the height of bounding boxes that contain at least 7 (i.e, half of all) keypoints (see Figure 3.1). We choose a minimum bounding box height of 60px. This is for two reasons: **a)** this decision is in line with TDUP, and **b)** there are only few labels below that threshold. The latter may by itself indicate that pose estimation at that size is infeasible. Furthermore, the model's ability to estimate poses accurately at that resolution may be diminished due to the lack of training data.

2. After limiting the bounding boxes for which pose estimation is performed, we provide limits on the keypoint confidence. We only keep the joints that have an associated confidence above a certain threshold. We decide on the threshold value by plotting OKS/ACC performance at various threshold values across the confidence range $[0, 1]$ (see Figure 3.2b). We find that for most of the range (< 0.8) the performance degrades gradually. Any threshold in this range seems adequate. Therefore, we choose the intuitive middle point: 0.5. This is a point at which we do not yet lose a significant portion of our labels, yet already achieve about half of the possible improvement.

3. Finally, we omit poses based on the number of keypoints left after the previous filtering step. Visual inspection showed that when only few keypoints passed the confidence threshold, the people were often either heavily occluded, truncated or visibility was low (e.g. very dark scenes, or blending into the background). We found the former setting to be associated with predictions of subpar quality, and predictions in the latter setting were hard to verify (as our human ability of locating the joints failed). Therefore, we omit such heavily limited pose information completely. A minimum number of 5 joints (more than a third) of a pose need

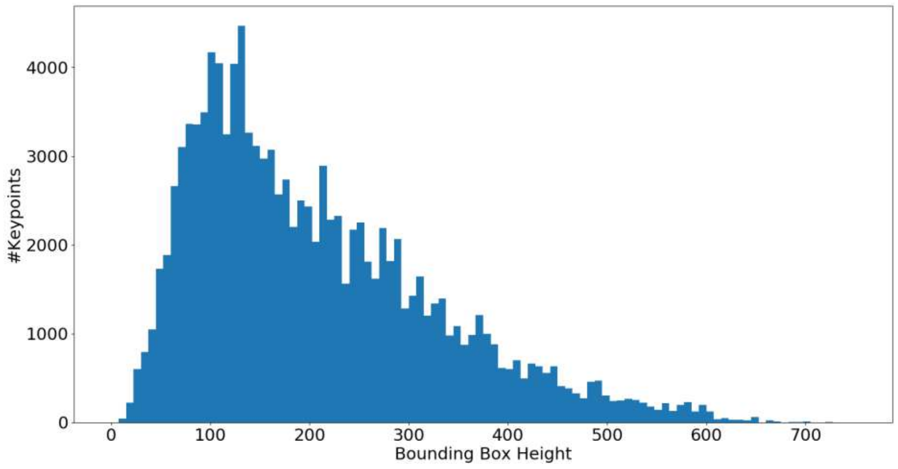


Figure 3.1: Histogram of the number of keypoints over the range of bounding box heights in the WODS++ development set.

to be identifiable to not be filtered altogether. This additional filtering step also brought about a minor improvement in the evaluation metrics.

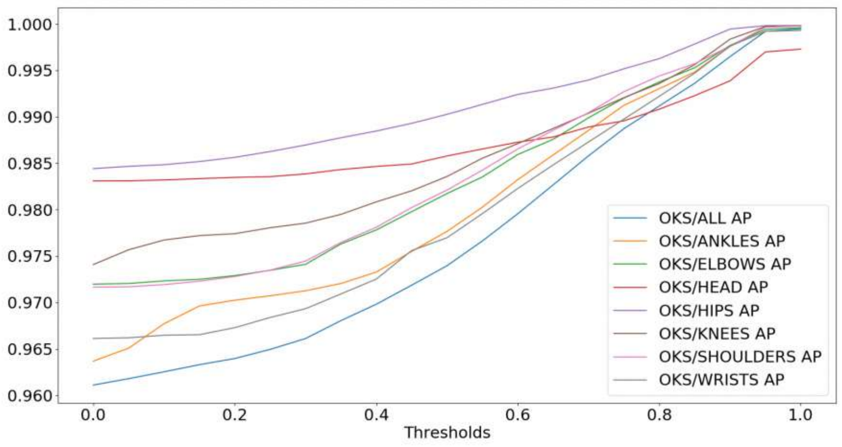
Up-Lifting

The generated 2D keypoints are lifted into 3D, leveraging WODS' LIDAR data. We use the procedure laid out in [ZSG+21]. The formula is given in Equation 3.1. The weight factors it relies on are defined in Equation 3.2.

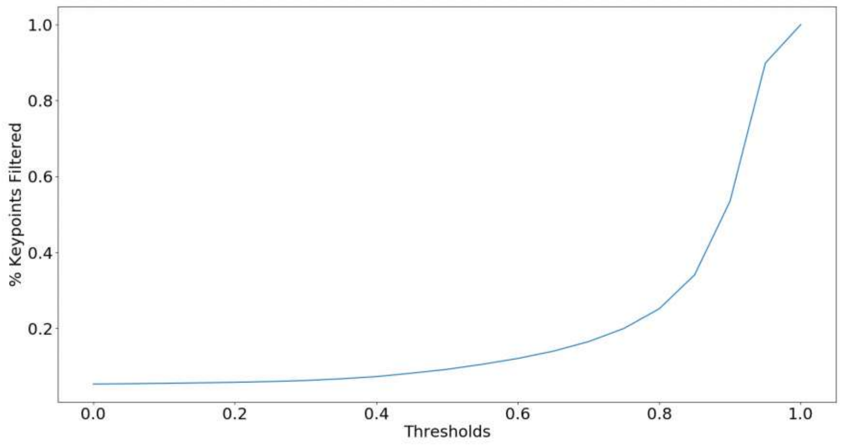
$$\tilde{\boldsymbol{y}}_{\boldsymbol{k}} = \sum_{i=1}^{N} \alpha_{ik} \boldsymbol{x}_{\boldsymbol{i}} \tag{3.1}$$

$$\alpha_{ik} = \frac{exp(-\tau \| \boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{(p)}} - \boldsymbol{y}_{\boldsymbol{k}} \|_2)}{\sum_{j=1}^{N} exp(-\tau \| \boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{(p)}} - \boldsymbol{y}_{\boldsymbol{k}} \|_2)} \tag{3.2}$$

Here, k identifies a joint and i a LiDAR point. The 3D keypoint estimate $\tilde{y}_k$ is a weighted sum of LiDAR points x_i The sum is from 1 to N, where N is the number of LiDAR points contained in the 3D bounding box of that individual. The weight factor α_{ik} is the result of a softmax operation over the negative L^2-norms between the 2D joint labels $\boldsymbol{y}_{\boldsymbol{k}}$ and the 2D projections of the LiDAR points $\boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{(p)}}$. Therefore, points that lie closer to the joint on the image plane have a higher weight. The smoothness of the softmax operation is controlled with a temperature parameter τ.



(a) OKS/ACC performance of our HRNet [SXLW19] model at various confidence thresholds.



(b) Percentage of ground truth keypoints made ineligible due to filtering at various confidence thresholds.

Figure 3.2: Confidence threshold study on the WODS++ development set.

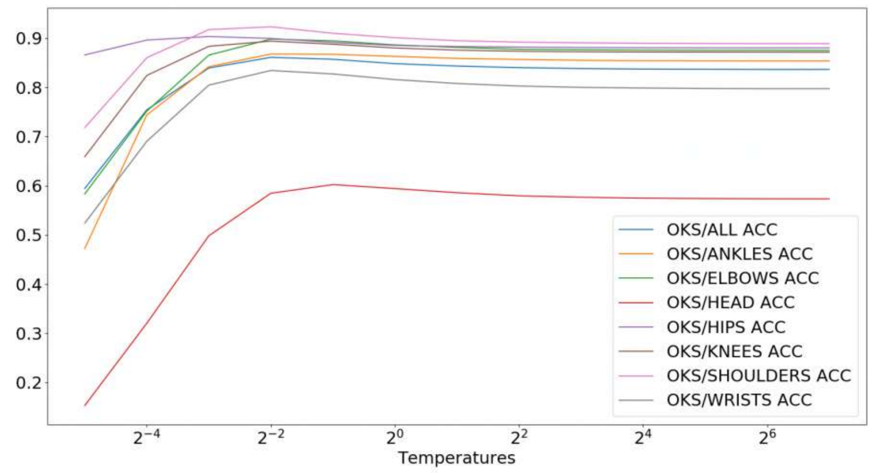


Figure 3.3: OKS/ACC 3D scores on the WODS++ development after the LiDAR lifting procedure at various temperatures.

We choose the temperature $\tau = 0.25$. We evaluated the prediction performance on the WODS++ development set at the following temperatures: $\tau_k = 2^k, k \in \{-5, -4, ..., 6, 7\}$. The results are visualized in Figure 3.3.

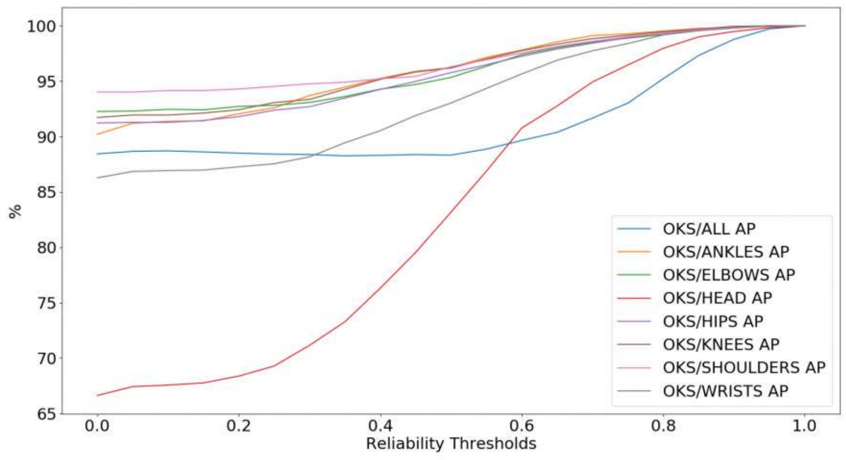
Since not all individuals have an associated 3D bounding box, a significant number of 2D poses cannot be lifted. As we will see in Chapter 5, limiting the points considered for the uplifting operation in 3D is necessary to avoid major deviations.

Additionally, we omit instances where less than 14 points are contained in the bounding box. The underlying intuition is that the existence of at least one LiDAR point per keypoint is necessary for any suitable estimation.

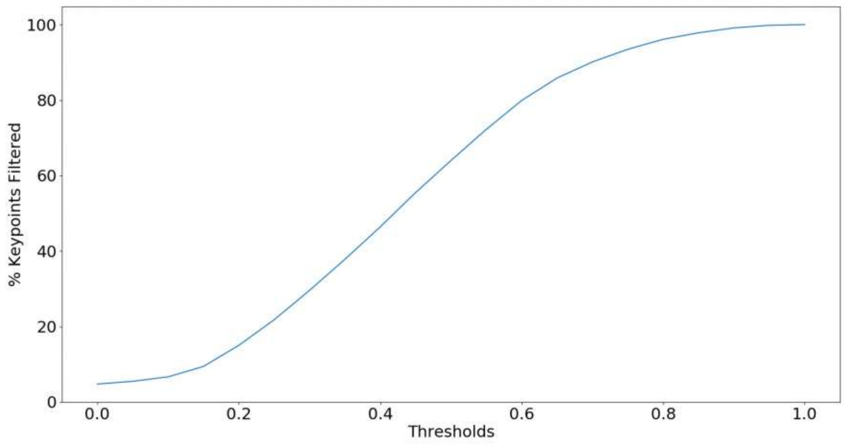
Keeping to [ZSG+21], a reliability threshold is employed. When a keypoint has no close enough point, we omit it. The reliability score is determined by the closest LIDAR point on the image plane, as shown in Equation 3.3.

$$r_k = exp(-\tau\, min_i \|\boldsymbol{x_i^{(p)}} - \boldsymbol{y_k}\|_2) \tag{3.3}$$

This was determined in an analogous fashion to the other hyperparameters we chose so far; i.e., we evaluate each threshold at steps of size 0.05 in the range $[0, 1]$. Again, we keep track of how much ground truth is lost. The results are visualized in Figure 3.4. The graphs show the performance measure percentages (Figure 3.4a) and the percentage of filtered out keypoints (Figure 3.4b) across thresholds. These graphics indicate that no particularly favorable trade-off can be achieved. Accuracy improvements come with the loss of large quantities of labels. However, since the ground truth is hand-annotated only on selected point clouds, it seems likely that in their choice annotators gravitated



(a) 3D OKS/ACC performance of our lifting procedure at various reliability thresholds.



(b) Percentage of ground truth keypoints made ineligible due to filtering at various reliability thresholds.

Figure 3.4: Reliability threshold study on the WODS++ development set.

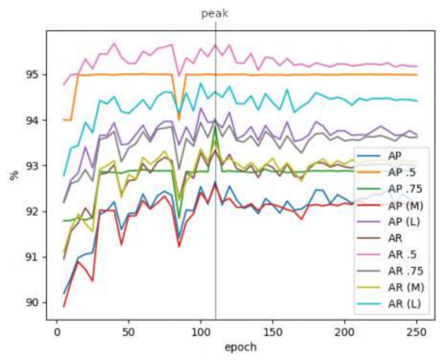
to particularly well-formed ones. On such point clouds, the lower reliability thresholds rarely take effect. Based on this intuition, we choose the relatively low value of 0.15.

3.2 Experiments and Results

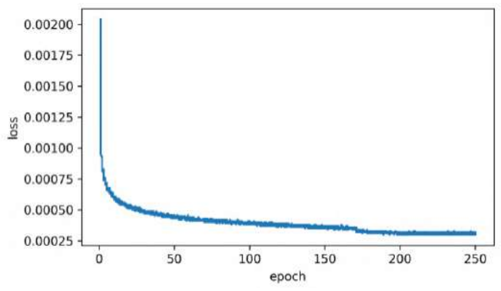
First, we note some general statistics of the original and the extended dataset in Table 3.1. The number of labels is vastly increased to over 1 million, for both 2D and 3D.

Metrics	2D		3D	
	Original	Extended	Original	Extended
#Poses	173.143	1.242.332	11.609	1.035.184
Avg. Poses per Image	5.1	5.7	1.3	4.8
Avg. Keypoints per Pose	8.0	9.6	11.5	9.5
Avg. Height (2D) & Depth (3D)	190.3px	158.9px	14.7m	28.2m

Table 3.1: Statistics of the original [SKD+20] and extended WODS. Only images that contain at least one pose label are considered. We give the height of 2D bounding boxes in pixels and for 3D we give the euclidean distance to the box center.



(a) AP and AR, evaluated after each epoch on the development set.



(b) HRNet's heatmap training loss.

Figure 3.5: HRNet [SXLW19] training progress on WODS [SKD+20].

Furthermore, we extended to smaller depictions of humans, as evidenced by the smaller average height and higher average depth of labels.

The model is trained for 250 epochs. Every 5 epochs we evaluate the model on the development data split. The best model is chosen based on COCO-style AP which we find at epoch 110 (see Figure 3.5a). At that point, training loss has mostly, but not yet completely, levelled off (see Figure 3.5b).

While this is not explicitly stated, we assume that what is referred to as *"internal dataset"* in [ZSG+21] has large overlap with the keypoint annotations that were published shortly after the paper. Note that the authors of [ZSG+21] are affiliated with *Waymo*, the publishers of WODS. The number of annotations we extracted do not match the statistics mentioned in the paper exactly, but they are of a similar magnitude[5]. As such, we believe a comparison between accuracy metrics from [ZSG+21] and our results is meaningful. While this uncertainty is undesirable, [ZSG+21] is the only point of reference in terms

[5]Reported [ZSG+21]: 197, 381 2D pose labels. Extracted: 173, 526

of performance to our knowledge[6]. We report the results in the OKS/ACC[7] metric proposed in the same work. Also their choice of keypoint standard deviation coefficients used for the OKS/ACC calculation is not explicitly stated. However, due to the authors' affiliation to Waymo, we assume that they use the ones provided in the WODS toolkit.

We report our results in the subsequent Tables (3.2, 3.3, and 3.4) and Figures (3.6 and 3.7). In all these Tables, the *Filtered* column refers to the performance after applying the filtering conditions we introduced in Section 3.1.3. In the *Filtered* setting, we only compare to ground truth for which we have filtered predictions (i.e., we do not incorporate false negatives). It should be noted that the filtering only reduces the number of predictions and does not change them, and that the metrics only consider instances where ground truth exist. Therefore, the filtered set would necessarily perform worse, if we did not reduce the ground truth to match the predicted set. This would hinder meaningful insight into the quality improvement.

Furthermore, we split results into groups of two related body parts. That is generally the left and right version of a given joint, but for the head it is forehead (respectively, head-center in the 3D setting) and the nose. The OKS calculation between target and prediction is only performed on the joints of the respective group. The *overall* category (last row in the tables) considers all joints.

Note that the definition of keypoints between 2D and 3D differs slightly. Whereas WODS' 2D keypoint definition considers the forehead, in 3D the center of the head is specified. Lower performance in metrics relating to the head accuracy may be due to this disparity. The phenomenon of reduced scores for the head group is also observed in [ZSG$^+$21].

In Table 3.2 we report the OKS/ACC performance of the 2D pseudo-labels on the test split. We report scores $\geq 94\%$ both filtered and unfiltered, and for any body part group. The results reported in [ZSG$^+$21] are consistently out-performed.

Next, we consider 3D OKS/ACC performance in Table 3.3. Here, we report values over 80% for all groups except *head*, with 57.5% unfiltered and 68.1% filtered, respectively. Still we outperform performance metrics reported in [ZSG$^+$21] in all regards.

We also quantify 3D results in terms of MPJPE in Table 3.4. Our method still generally performs better than the reported from [ZSG$^+$21], but not in all groups. In hip, head, and shoulder our unfiltered MPJPE is slightly higher. Interestingly, the MPJPE for the head group is the lowest among all groups in the filtered setting and the method from [ZSG$^+$21], and third lowest in our unfiltered setting. This stands in contrast to the previous results from Table 3.3. A possible reason may be that the head keypoints have comparatively small variations, making OKS-based metrics more strict, while lowering MPJPE.

[6] At this point in time. [13.6.22]

[7] OKS/ACC is the average of multiple OKS based accuracy measures. For each threshold in $[0.05, 0.95]$ with steps of size 0.05 a prediction is considered a match for the purposes of accuracy computation if it is above the threshold. Practically, this is a variant of COCO's [LMB$^+$14] AP with a wider range of thresholds and prediction to ground truth matching being unnecessary.

Part	OKS/ACC@2D (%)			
	Ours	Ours[Filtered]	Camera [ZSG+21]	Multi-Modal [ZSG+21]
head	97.3	98.5	75.1	72.2
shoulder	95.9	98.2	83.4	87.9
elbow	96.4	98.3	82.6	84.8
wrist	96.2	98.0	79.0	79.2
hip	97.9	99.0	88.0	92.4
knee	97.0	98.4	85.9	90.1
ankle	95.8	97.8	84.2	88.7
overall	94.2	97.4	78.2	82.9

Table 3.2: OKS/ACC@2D performance comparison between our HRNet [SXLW19] and the reported values from [ZSG+21]. Metrics are computed on the WODS++ test split.

Part	OKS/ACC@3D (%)			
	Ours	Ours[Filtered]	Camera [ZSG+21]	Multi-Modal [ZSG+21]
head	57.5	68.1	24.5	29.7
shoulder	90.2	92.2	65.4	76.9
elbow	88.3	91.0	65.6	72.5
wrist	81.2	84.7	46.0	47.0
hip	87.9	89.6	57.7	74.8
knee	87.8	90.8	65.4	78.0
ankle	85.5	90.2	62.7	72.3
overall	84.4	87.3	51.7	63.1

Table 3.3: OKS/ACC@3D performance comparison between the lifted predictions of our HRNet [SXLW19] and the reported values from [ZSG+21]. Metrics are computed on the WODS++ test split.

Finally, we present our pseudo-labels on selected frames. In Figure 3.6 we show our 2D visualizations in a complex traffic scene. In Figure 3.7 we show a selection of corresponding images and LiDAR point clouds overlaid with the generated 2D and 3D labels, respectively. The keypoint size in the point-cloud visualization is proportional to its estimated standard deviation (default behaviour in the WODS toolkit).

Part	MPJPE@3D (cm)		
	Ours	Ours[Filtered]	Multi-Modal [ZSG+21]
head	9.1	7.0	8.4
shoulder	8.5	7.6	8.7
elbow	8.6	7.5	8.9
wrist	10.2	8.7	13.2
hip	12.4	11.8	12.1
knee	10.1	8.7	11.1
ankle	12.2	9.4	11.1
overall	10.2	8.8	10.3

Table 3.4: MPJPE performance comparison between the lifted predictions of our HR-Net [SXLW19] and the reported values from [ZSG$^+$21]. Metrics are computed on the WODS++ test split.

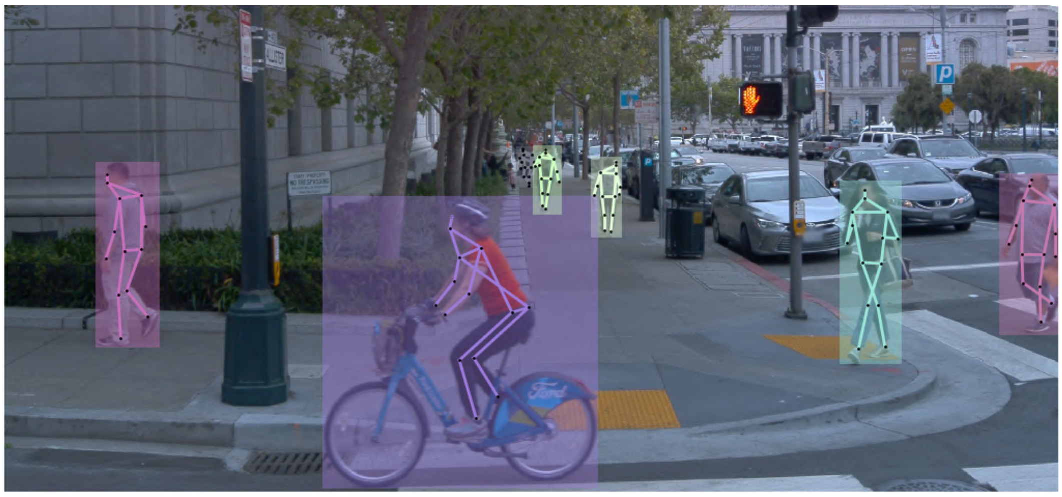


Figure 3.6: Pseudo-labels on a selected frame from the WODS++ development set.

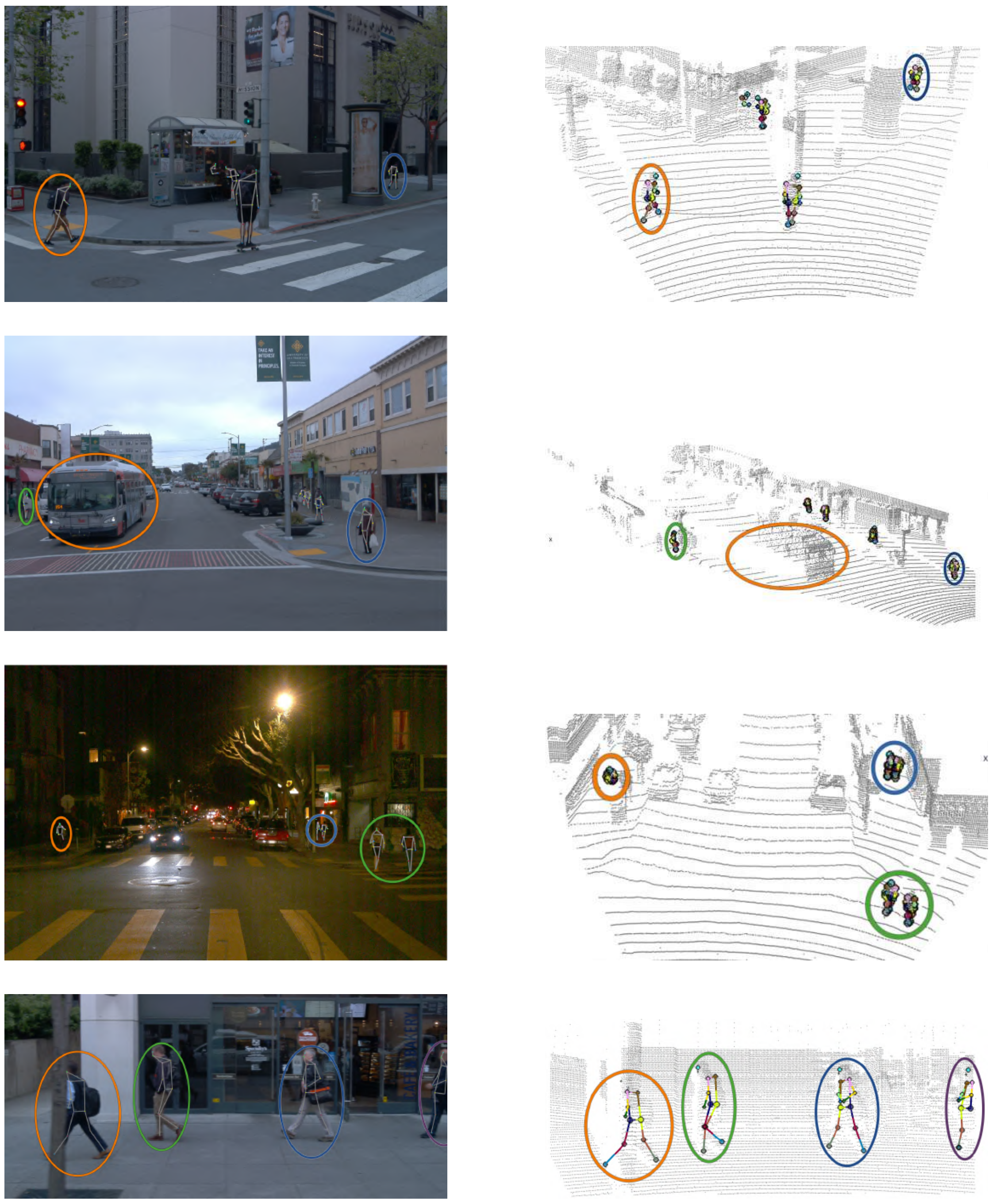


Figure 3.7: 2D and 3D pseudo-labels visualized on four selected scenes from the WODS++ development set. Colored ellipses mark corresponding objects in image and point cloud to aid orientation. Each keypoint type has a different color, consistent across people.

CHAPTER 4

Human Pose Estimation of Vulnerable Road Users in 2D

In this chapter, we perform single-stage 2D HPE on VRU datasets. The KAPAO [MVWM21] architecture is used. We explore multiple settings: varying inference hyperparameters, adjusting training data preprocessing strategies, choosing different initial model weights, and also modifying the structure slightly. We primarily experiment on the datasets TDUP [WYW$^+$21] and BDD100K [YCW$^+$20]. First, the implementation of the configurations is explained in detail in Section 4.1. Then, we report our experimental results in Section 4.2.

4.1 Implementation

Our aim is to study a single-stage 2D HPE method on VRU benchmarks. In an attempt to achieve competitive performance, we experiment with a number of different configurations of KAPAO [MVWM21].

We build on the KAPAO-L architecture, the largest version of KAPAO with the highest accuracy. It, in turn, is based on the architecture and weights of YOLOv5-L [JCS$^+$22], trained on the COCO object detection [LMB$^+$14] dataset.

4.1.1 Training Procedure

YOLOv5 [JCS$^+$22], the basis of KAPAO, provides an extensive suite of data augmentation steps. The following modifications are performed during training by default:

- random left-right flipping
- perturbations in the individual dimensions of the HSV color space

- mosaicking, i.e., creating a 2-by-2 grid of four individually augmented images for the model input
- random re-scaling
- random translation

The first two points are considered unproblematic. The latter three, however, may be disadvantageous in the AV context. Therefore, we consider a setting in which those are disabled. This is motivated by the following concerns:

- Mosaicking destroys the previously present scene homogeneity, which is a consequence of the image always being shot from the car (and, thereby, generally from the street, often with sidewalks or buildings to the side, etc.).
- VRUs often appear of small size in the image, such that scaling down may make HPE infeasible.
- In many cases, VRUs are located towards the edges of the scene. Random translation could then result in the primary targets of interest being cut off.

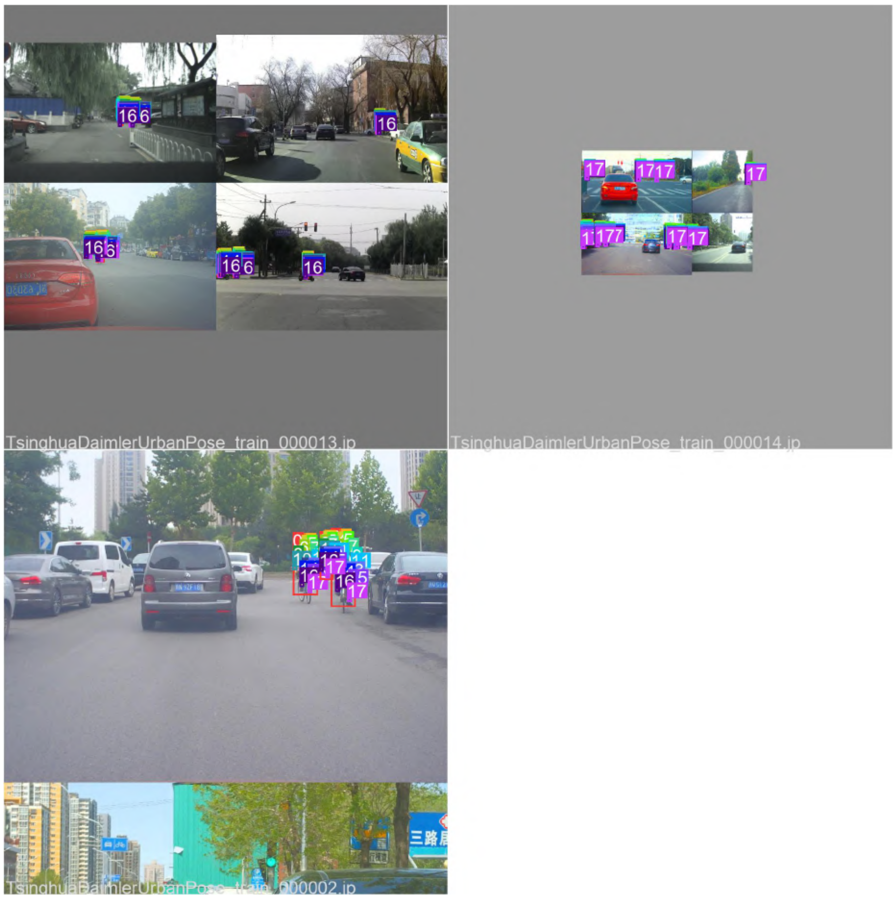
Furthermore, we reconsider how keypoints are modelled in KAPAO. The default keypoint bounding box size is 5% of the larger of the two image dimensions (64px at a total width of 1280px). That would result in a keypoint's bounding box frequently being wider than the person it belongs to. We reduce it to the keypoint bounding box size 1% (12.8px).

Finally, by default, images are padded such that they are square. Since the datasets of interest mostly have a consistent aspect ratio that is quite far from square, we can increase efficiency by avoiding this step. So, padding is only applied to the next closest permissible size[1].

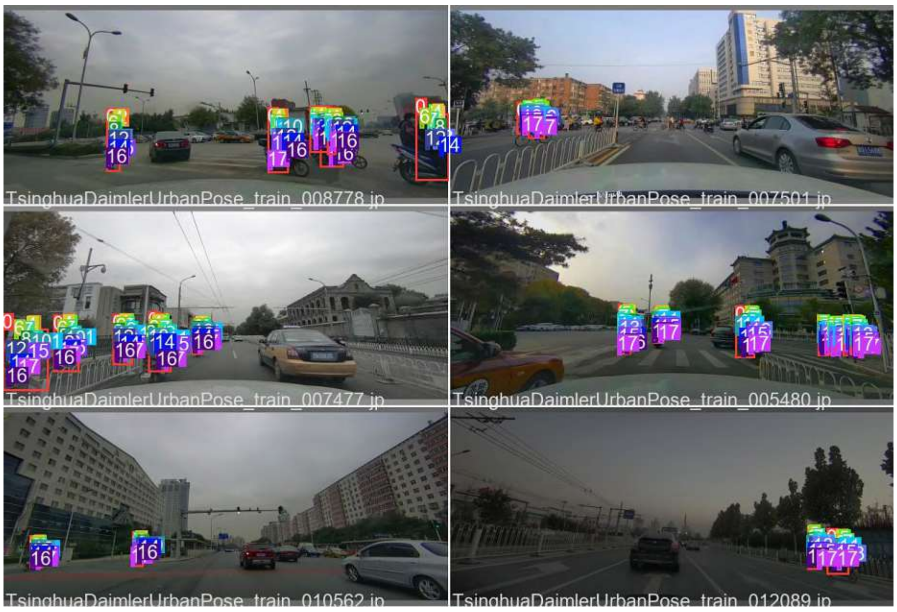
Find a comparison of example training batches between the default and our customized setting in Figure 4.1. The gray borders represent the padding that is added after down-scaling the input images. The customized setting (Figure 4.1b) shows the driving scenes as they are, aside from possible flips and color perturbation. The default setting (Figure 4.1a) additionally applies the problematic augmentations discussed previously: mosaicking (all three images in the batch), scaling (top-right image), and translation (top-right and bottom image).

When comparing results, we denote the original training setting with the additional identifier `{TD}`. Also, we introduce `{PT}`, for pretrained only, when we do not train on the target dataset at all.

[1]Due to the repeated use of strided convolutions, YOLOv5 places limitations on permissible input sizes. The default architectures require multiples of 32px or 64px.


TsinghuaDaimlerUrbanPose_train_000013.jp
TsinghuaDaimlerUrbanPose_train_000014.jp
TsinghuaDaimlerUrbanPose_train_000002.jp

(a) Training batch of 3 images with extensive data augmentation.


TsinghuaDaimlerUrbanPose_train_008778.jp
TsinghuaDaimlerUrbanPose_train_007501.jp
TsinghuaDaimlerUrbanPose_train_007477.jp
TsinghuaDaimlerUrbanPose_train_005480.jp
TsinghuaDaimlerUrbanPose_train_010562.jp
TsinghuaDaimlerUrbanPose_train_012089.jp

(b) Training batch of 6 images when limiting data augmentation.

Figure 4.1: Training batch samples from TDUP [WYW+21] with the original default training settings (top) and our customized setting (bottom). Numbers and colors identify the different keypoints.

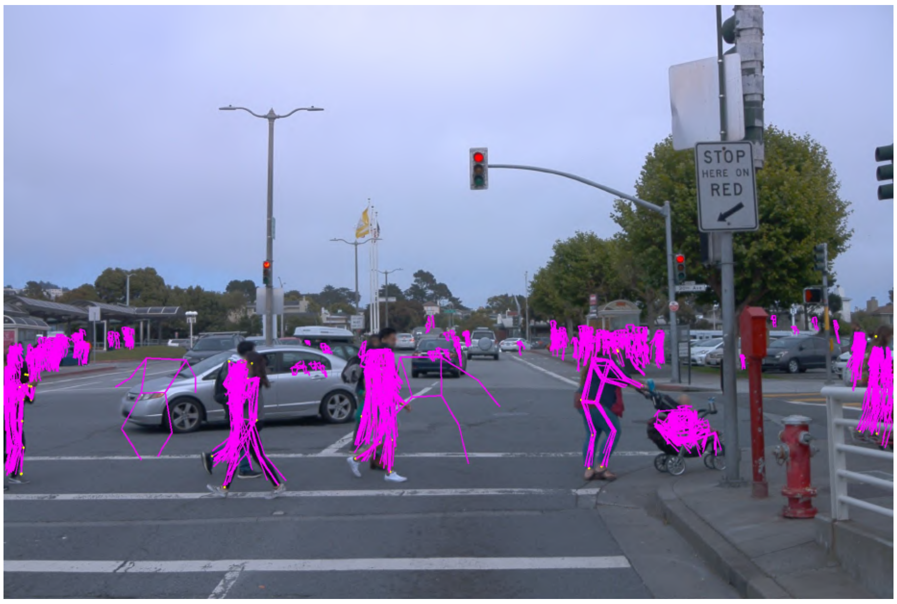


Figure 4.2: Inference result of the default validation settings (with a low confidence threshold) using the public KAPAO-L [MVWM21] checkpoint pretrained on COCO Keypoints [LMB$^+$14] on a scene from WODS [SKD$^+$20].

4.1.2 Inference Settings

A number of hyperparameters need to be chosen for practical application of the model. These include confidence and detection overlap thresholds. Their choice has considerable impact on the number of predictions made. Therefore, they determine the precision-recall trade-off. In addition to the inherent complexity of optimizing these parameters, optimality may differ depending on the evaluation benchmarks' performance metrics.

The default validation parameters provided by KAPAO[2] are presumably tuned for the prominent COCO Keypoints [LMB$^+$14] dataset. Notably, they use a confidence threshold of 0.001. This stands in contrast to their inference demo[3], where they use a value of 0.7, thus predicting individuals at a significantly more conservative rate. The evaluation specfics may have motivated their choice of the former, more liberal setting.

COCO's evaluation toolkit limits the considered pose predictions to only the top 20, based on the confidence score associated with the predictions. Additional predictions do not influence the score, and are discarded by the metric. Therefore, the degree to which false positives are penalized is limited. When false positives are not penalized accordingly, the metric may not reflect effectiveness in a practical use-case properly. We show the low

[2]https://github.com/wmcnally/kapao/blob/ad507c2a00de330eb40d58cf0f4614d82d3c86b5/val.py [retrieved on 20.5.2022]

[3]https://github.com/wmcnally/kapao/blob/ad507c2a00de330eb40d58cf0f4614d82d3c86b5/demos/image.py [retrieved on 20.5.2022]

confidence threshold resulting in a particularly unnatural outcome in Figure 4.2. In this example, the model produces many more pose predictions than there are people depicted.

The demo threshold, on the other hand, was found to be quite restrictive. Therefore, we introduce an adjusted, more lenient version. We use demo settings with the confidence threshold reduced to 0.15. This configuration is considered the default setting in our experiments. We denote KAPAO's original default validation setting with `(VD)` when comparing results.

4.1.3 Initial Checkpoints

There are also multiple options for the choice of initial object weights. As in many other deep learning applications, starting training from pretrained checkpoints is recommended for YOLOv5[4].

Model checkpoints are available for both YOLOv5[5] and KAPAO[6]. We use the KAPAO-L checkpoint that was trained on the COCO Keypoints dataset. It, in turn, started off from the YOLOv5-L checkpoint. Since this checkpoint was already trained on a large-scale HPE dataset, it may improve final model performance.

We can go another step further in this chain of checkpoints. After training on one of our 2D VRU datasets, the resulting weights may be used to initialize the model for training on the other dataset. In this case, we do not only start with HPE weights, but the specialized AV domain is also known beforehand.

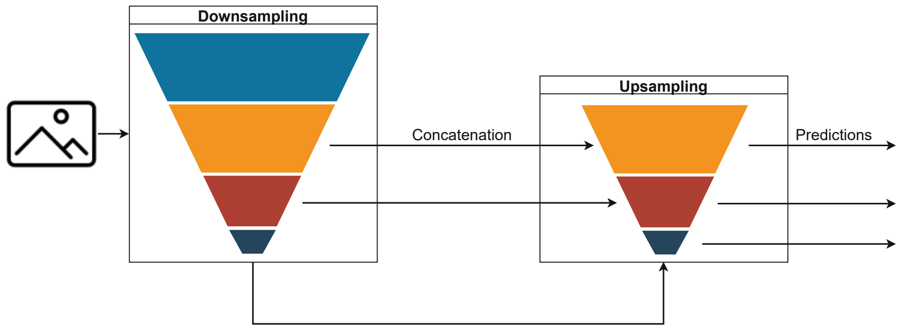
As the default setting in our experiments, we start from the COCO Keypoints KAPAO-L checkpoint. We denote setting YOLOv5-L, trained on COCO Detection, as our initial checkpoint with `[YOLO]`. We also start training with the KAPAO weights trained on the respective other VRU benchmark, which is indicated with `[TDUP]` and `[BDD]`, respectively. We also use a checkpoint trained on WODS++, marked as `[W++]`. This is to establish a performance baseline for the experiments in Chapter 5. Furthermore, it may give insight into the dataset's utility as a training set.

4.1.4 Architecture

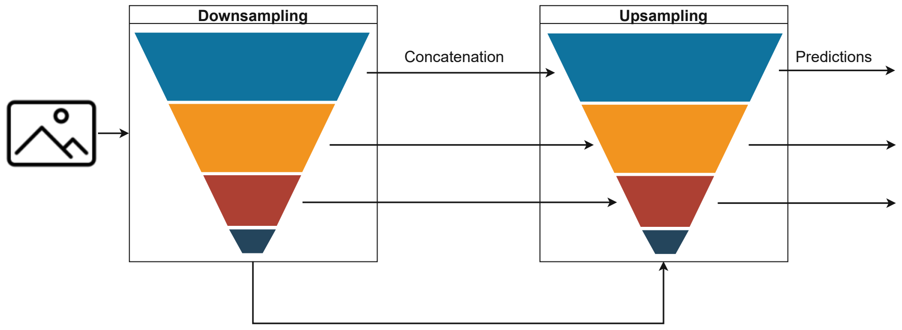
As mentioned previously, VRUs often appear small in traffic scenes. We do not just detect VRUs, but their individual joints. KAPAO models these joints as objects with bounding boxes. Therefore, the concern arises whether the underlying YOLOv5 architecture is equipped to handle that.

Therefore, we first consider the default YOLOv5 (and therefore KAPAO) architecture (see Figure 4.3a). On a conceptual level, YOLOv5 first applies a number of layers that build on convolutions, to extract increasingly high-level image features (top-down

[4]https://docs.ultralytics.com/tutorials/training-tips-best-results/ [retrieved on 20.5.2022]
[5]`https://github.com/ultralytics/yolov5` [retrieved on 22.8.2022]
[6]`https://github.com/wmcnally/kapao` [retrieved on 22.8.2022]



(a) Default YOLOv5 [JCS+22] architecture.



(b) Customized YOLOv5 [JCS+22] architecture. A higher resolution output grid is added, and the previously lowest resolution grid is removed, as in [BTCB21].

Figure 4.3: Conceptual illustrations of the default and the customized YOLOv5 [JCS+22] architecture.

phase). Afterwards, in the bottom-up phase, the features are again up-sampled and concatenated with corresponding features from the top-down phase. This results in a number of output-grids at different resolutions. Each cell of each grid is responsible for detecting objects. These output grids have a resolution of $\{\frac{w}{r} \times \frac{H}{r} | r \in R\}$, where w and h denote the input image's width and height, respectively. R is the resolution ratio set ($R = \{8, 16, 32, 64\}$ in YOLOv5-L).

We decide to experiment with a higher grid resolution. We shift KAPAO-L's output grids by one order of 2, i.e., $R' = \{4, 8, 16, 32\}$. This is achieved by adding another up-sample operation and adjusting the layer concatenation accordingly. This approach is inspired by [BTCB21]. An illustration of the adjusted architecture is given in Figure 4.3b. Note how in the upsampling step the lowest resolution grid during upsampling (dark blue) has no prediction output arrow (which is present in Figure 4.3a), but there is an additional grid on top (light blue) that does output predictions.

Alternatively, we could just add the additional output grid, instead of shifting the resolution set. This is not done for two reasons, **a)** then the adjusted model would be strictly more complex than the original one (whereas our modification trades the lowest resolution grid for a higher resolution one), which also comes at additional computational cost, and **b)** in the experiments of [BTCB21] such a modification did not perform any better than the shifting one we employ.

When discussing experiments, we use the additional identifier `+XS`, to denote the architecture variant with the high resolution output grid that we discussed here (and which intuitively is responsible for handling extra small objects). When no such identifier is present, the default KAPAO-L architecture is used.

4.2 Experiments and Results

We perform experiments to evaluate our different settings. To that end, we refer to the public benchmark datasets TDUP and BDD100K. Furthermore, we report metrics computed on the 2D labels of WODS++ to help contextualize results in Chapter 5.

We summarize the model setting notation:

- We refer to our KAPAO models plainly as `KAPAO` (or `K` to be space-efficient in tables).
- The default configuration, without any additional identifier:
  (a) applies the customized training procedure
  (b) uses the slightly adjusted demo validation setting
  (c) starts from the COCO KAPAO-L checkpoint
  (d) has the unmodified KAPAO-L architecture
- `{TD}` identifies variants where, instead of (a), the original default training procedure is applied.
- `{PT}` identifies variants where, instead of (a), no training is performed and the unchanged pretrained checkpoint is used for inference.
- `(VD)` identifies variants where, instead of (b), the KAPAO default validation parameters are used for inference.
- `[YOLO]`, `[BDD]`, `[TDUP]`, and `[W++]` identify variants where, instead of (c), the YOLOv5-L checkpoint, or the otherwise default setting model trained on BDD100K, TDUP, or WODS++, respectively, is used.
- `+XS` denotes variants where, instead of (d), the adjusted architecture geared towards detecting small objects is chosen.

All trainings are performed for 75 epochs, evaluating the current weights on the validation (or, as applicable, development) set after each epoch. We keep track of the best model version, as determined based on its AP (as computed by the COCO toolkit). That one is saved and used in the subsequent evaluations.

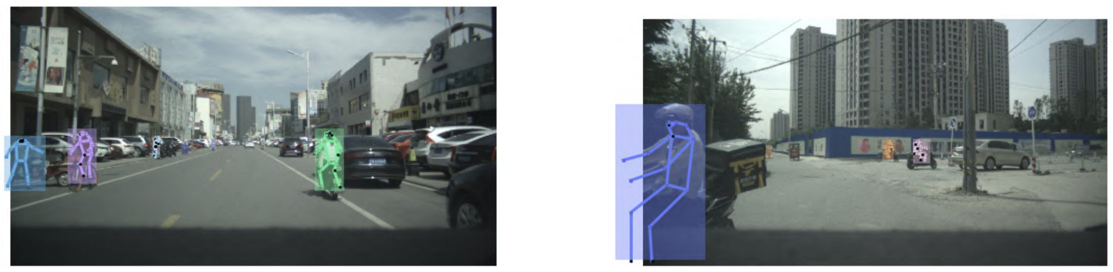


Figure 4.4: Two examples of annotations having out-of-frame coordinates in TDUP [WYW+21].

In our datasets, it is possible to have annotations beyond the image frame. We show two examples of such cases in Figure 4.4. To make maximal use of the data, we clip coordinates to the image bounds, for both bounding boxes and keypoints. For keypoints, we additionally set the visibility label to 0 (indicating a missing annotation), if clipping was necessary. This ensures that the within-frame keypoints of clipped poses are still usable and no ground truth is located out-of-frame (a requirement of the underlying KAPAO/YOLOv5 framework).

Latency is measured as the time needed to run inference on a single image. For our results, an RTX 2080 Ti GPU and an Intel i7-11700@2.50GHz CPU were used.

4.2.1 Tsinghua-Daimler Urban Pose

We evaluate our models on TDUP [WYW+21]. We report our findings in Table 4.1.

First, we provide a selection of results from the official TDUP website[7], and then our own. Note that the former are presumably evaluated on the test-set. Our values, however, are computed on the validation-set, since we found test-set result submissions to not work properly[8]. This is an advantage for our models, since we saved the best model in terms of validation set performance. The extent to which this improves our performance metrics is unclear. For reference, we find in Section 4.2.3 that the performance difference between validation and test set on 2D WODS++ is ~ 1 percentage-point.

Among the models reported on the website, we selected the best overall approach for display: AlphaPose [FXTL17]. It is a top-down method, whose initial object detection step is performed by YOLOv3 [RF18] in this particular case. Additionally, we show the performance measures of other single-stage (specifically, bottom-up) methods in HigherHRNet [CXW+20] and PifPaf [KBA19].

The results are computed using the TDUP evaluation toolkit, provided in the dataset download package[7]. Labels are partitioned into three sets: *Reasonable*, *Small* and

[7]http://urbanpose-dataset.com/info/Datasets/198[retrieved on 4.8.2022]
[8]At the time of writing: 4.8.2022

Architecture	LAMR ↓				AP ↑				ms
	Reas.	Sm.	Occ.	Comb.	Reas.	Sm.	Occ.	Comb.	
AlphaPose	**26.33**	38.43	58.75	34.03	**70.41**	**52.33**	**24.27**	**59.37**	230.2
HigherHRNet	33.91	42.84	64.40	40.90	53.39	34.90	13.71	43.61	2110
PifPaf	36.49	56.63	64.55	44.12	57.15	29.30	17.75	46.49	79.3
K{PT}(VD)	46.67	58.09	77.13	53.91	50.7	23.89	8.65	39.61	525.4
K{PT}	45.31	57.99	76.78	52.87	49.92	22.83	8.5	38.93	55.9
K{TD}(VD)	62.35	62.91	65.87	63.11	51.39	30.26	18.12	42.62	179.8
K{TD}	55.43	75.01	63.51	59.0	51.87	20.86	20.51	42.5	55.0
K(VD)	27.27	41.03	**54.49**	34.09	65.51	42.28	23.55	54.79	113.0
K	26.75	40.42	54.50	**33.67**	65.12	42.52	23.9	54.62	51.5
K[YOLO]	31.2	45.4	58.84	38.15	55.78	36.3	18.83	46.44	50.7
K[BDD]	27.95	42.27	56.63	35.12	64.04	41.31	22.28	53.41	52.3
K[W++](VD)	28.4	**37.61**	57.87	35.22	66.22	45.25	22.65	55.41	36.4
K[W++]	28.27	38.15	57.91	35.18	65.36	45.24	22.31	54.74	**34.8**
K+XS	27.18	40.15	**54.49**	33.94	65.03	44.13	23.41	54.61	64.8

Table 4.1: TDUP [WYW+21] evaluation metrics. The horizontal line separates reported approaches [FXTL17, CXW+20, KBA19] and our KAPAO [MVWM21] variants. Bold values are the best overall, underlined values are best in class.

Occluded. The former two differentiate between the size of human depictions, whereas the latter accounts for all people that are significantly occluded or truncated. Both LAMR and AP are computed on these sets individually. The *Combined* score gives a weighted average of these: with *Reasonable*, *Small*, and *Occluded* scores being weighted with 0.7, 0.1, and 0.2, respectively. These weights are chosen by the benchmark authors with the stated intent to model practical relevance. For example, people depicted small tend to be far away and their detection may, therefore, be less important in safety applications.

Analyzing the results in Table 4.1, we find that the better KAPAO variants beat the single-stage baselines consistently. This holds across label sets, AP, LAMR, and latency. Furthermore, they can compete with the AlphaPose model in terms of LAMR. In terms of AP, however, KAPAO cannot match AlphaPose, except for the *Occluded* class (compare column *Occluded*-AP of the *AlphaPose* and `K` rows in Table 4.1). However, where the other single-stage methods underperform by $>$ 10 percentage-points, KAPAO only does so by $\sim$ 5 percentage-points.

Among KAPAO settings, we find that the default training procedure (`{TD}`) and the pretrained checkpoint (`{PT}`) perform significantly worse than the others (compare rows of the respective variant with the *K* row in Table 4.1). Both of these results are expected: the former intuitively increases the difficulty by destroying the traffic scene homogeneity (see Section 4.1.1), and the latter is disadvantaged by having no knowledge of the dataset specifics at all. Notably, these configurations also bring increased latency, particularly at lower confidence thresholds (`(VD)`). That additional time is required by the post-processing; i.e., NMS and the KAPAO-specific pose-keypoint matching. This increase in post-processing latency indicates that these less effective models also seem to be less specific in what grid cells hold the respective predictions; requiring more intensive post-processing.

Applying a lower confidence threshold at inference (`(VD)`) has a mixed impact. It improves some metrics and worsens others, in a way that we found to not always be consistent across different initial checkpoint choices. Additionally, we find that the latency increases due to more extensive post-processing with more predictions (compare the *ms* column of any KAPAO configuration row in Table 4.1 with its `(VD)` variant). Even though `K` performs well, indicating a good fit, we find that the latency doubles with `K(VD)`. Interestingly, this is not observed with `K[W++]` and `K[W++](VD)`, where the discrepency is only 2.4ms.

Next, we discuss the initial checkpoint choice. Expectedly, starting from YOLOv5 as opposed to the default (KAPAO-L trained on COCO Keypoints) decreases performance (compare rows `K` and `K[YOLO]` in Table 4.1). This is expected, as the default initial checkpoint started from the YOLOv5 checkpoint itself and is additionally trained on a dedicated pose estimation dataset. Starting from the BDD100K checkpoint also decreases performance (compare rows `K` and `K[BDD]` in Table 4.1). Possibly, this is due to the model overfitting to the comparatively small, domain-specific dataset. Due to the large-scale and general purpose nature of COCO Keypoints dataset, on the other hand, models may not overfit to the same extent. Starting from the WODS++ checkpoint, however, does provide a benefit in AP (except *Occluded*-AP) and *Small*-LAMR (compare rows `K` and `K[W++]` in Table 4.1). This may be a consequence of the dataset's much larger size compared to BDD100K. Additionally, the `[W++]` setting also brings about a significant decrease in latency. The change in inference time is again attributable to the post-processing step. It seems like the inverse of the phenomenon observed previously in this section with the weaker model variants `{PT}` and `{TD}`.

Finally, we consider the XS architecture variation (`+XS`). The results overall are in line with its design considerations: performance for the *Small* class are slightly improved without significantly deteriorating performance otherwise. The differences to the standard configuration (plain `K`) are marginal. However, `K[W++]` beats `K+XS` in *Small* class performance, by 2 percentage-points in LAMR and $\sim$ 1 percentage-point in AP, at no additional computational expense compared to default `K`. The `+XS` architecture has increased latency (64.8 ms versus the default `K`'s 51.5 ms), which is due to the architecture modification itself, rather than post-processing. Therefore, the `+XS` configuration compares unfavourably overall.

4.2.2 Berkley Deep Drive 100K

Next, we report our results on BDD100K's 2D HPE labels. Performance metrics are displayed in Table 4.2.

Architecture	BDD100K Validation Set									
	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_M	AR_L
HRN48	50.3	64.4	55.9	**48.0**	74.3	72.2	92.8	79.1	69.9	78.1
ResNet-101	48.1	63.1	53.3	45.7	71.6	69.5	91.4	76.2	67.1	75.8
HRN48(DARK)	50.3	64.3	56.2	47.9	**74.5**	72.2	92.8	**79.7**	69.9	**78.5**
MobileNetV2	45.2	63.0	49.9	43.0	66.9	65.6	91.4	71.1	63.6	70.9
HRN32(UDP)	49.5	63.5	55.0	**48.0**	73.9	**72.4**	**93.2**	**79.7**	**70.3**	77.9
K	50.9	77.2	56.2	44.6	63.5	61.6	84.8	68.5	57.9	71.7
K(VD)	51.9	79.4	57.3	45.5	64.5	66.1	90.4	73.7	63.2	73.8
K[YOLO]	45.0	73.7	49.4	38.0	57.7	55.9	81.6	62.1	52.7	64.7
K[TDUP]	50.8	77.1	56.3	44.5	62.4	61.2	84.8	68.3	57.8	70.5
K[TDUP](VD)	51.6	79.1	57.1	45.4	63.7	65.2	89.8	72.9	62.5	72.7
K[W++]	51.5	78.8	56.7	45.9	63.0	62.0	85.3	68.7	58.7	70.7
K[W++](VD)	**52.5**	**80.2**	**57.6**	46.8	64.5	66.4	90.7	73.1	63.8	73.2
K+XS	48.2	75.1	52.6	41.0	62.9	60.0	83.9	66.7	56.5	69.2

Table 4.2: BDD100K [YCW+20] evaluation metrics (validation set). The horizontal line separates publicly reported approaches [SXLW19, HZRS16, SHZ+18, ZZD+20, HZGH20] and our KAPAO [MVWM21] variants. Bold values are the best overall, underlined ones are best in class. HRNet-w*XX* is abbreviated as HRN*XX*.

Table 4.2 is set up similarly to Table 4.1. Our models are evaluated using the BDD100K toolkit[9]. For comparison, we show the highest performing variant of each architecture listed in the publicly reported results[10] [SXLW19, HZRS16, SHZ+18, ZZD+20, HZGH20]. There are no single-stage methods among the published results, which we could use for reference. The published top-down approaches use a Cascade R-CNN [CV18] with an R-101-FPN backbone as the initial detection stage, which achieves 32.69 AP in the person detection task on the BDD100K validation set. Since no inference times are provided with the publicly reported results, we omit latency measures in the table. Note, however, that the inference times of our model configurations are similar to the ones reported for TDUP in Table 4.1. We again cannot compare test set results, since we found that the evaluation server does not process our submissions[11]. However, since the published results report validation- and test-set scores separately, we can fairly compare their and our validation set scores.

Overall, the KAPAO variants beat or match the top-down methods in terms of AP. In AP_{50} we report a significant gain of more than 15 percentage-points over the best reported method (*HRN48* [SXLW19]). Our best variant performs worse by 10 percentage-points

[9] `https://github.com/bdd100k/bdd100k` [retrieved on 22.8.2022]
[10] `https://github.com/SysCV/bdd100k-models/tree/main/pose` [retrieved on 23.5.2022]
[11] Documented in the following issue: `https://github.com/bdd100k/bdd100k/discussions/268` [retrieved on 5.8.2022]

Architecture	WODS++ 2D									
	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_M	AR_L
Dev: K	85.1	89.6	87.3	79.8	91.6	87.8	91.7	89.6	85.6	92.6
Dev: K (VD)	87.7	93.1	90.2	82.3	94.7	92.1	96.4	94.1	89.8	96.1
Test: K	84.1	89.0	86.5	80.8	90.1	86.6	90.4	88.4	84.6	91.6
Test: K (VD)	86.3	91.6	88.9	82.7	93.1	90.2	94.3	92.2	87.8	95.0

Table 4.3: AP and AR computed on the WODS++ 2D labels. *Dev:* and *Test:* indicate that the metrics were computed on the development and test split, respectively.

in the AP_L metric, however. KAPAO is also consistently inferior to all the displayed public results in terms of AR.

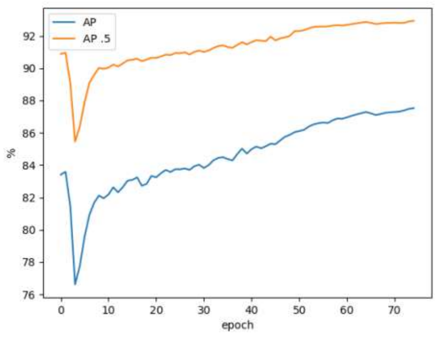
We now discuss the differences between the various KAPAO settings. Generally, the results are consistent with the ones of Section 4.2.1. Initializing the model with the weights pretrained on COCO Keypoints (plain `K`) is better than taking the more specific TDUP (`[TDUP]`) or the object detection checkpoint YOLOv5 (`[YOLO]`). Using the WODS++ checkpoint (`WODS++`) leads to improved performance over default `K`. The XS architecture setting (`+XS`) also performs slightly worse across the metrics. Wheras there was a special class for small people depictions in Section 4.2.1 in which `+XS` performed better than the default `K`, there is no such distinction here. We find the performance increase provided by the low confidence threshold validation parameters (`(VD)`) to be more significant here than on TDUP (see Section 4.2.1).

We do not evaluate the untrained (`{PT}`) variant, or the one trained with the default procedure (`{TD}`). The former is infeasible due to BDD100K's keypoint definition being different to the COCO Keypoints datasets that the `{PT}` model is trained on. The `{TD}` variant is omitted to avoid wasting computational resources on a setting that the previous findings have indicated to perform significantly worse.

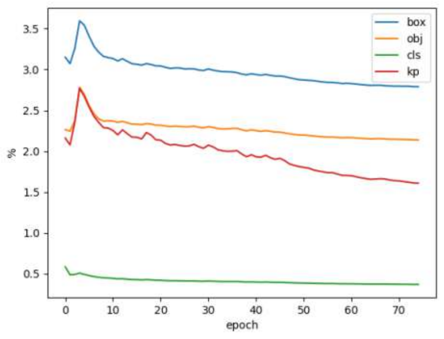
4.2.3 Extended Waymo Open Dataset

Finally, we evaluate KAPAO on the 2D WODS++ labels. This is the dataset whose creation we detailed in Chapter 3. There are no baselines available. We primarily perform this evaluation to introduce a performance baseline of KAPAO on the dataset. This is intended to help contextualize results of Chapter 5 related to this.

We split the WODS++ evaluation set (i.e., the one having the sequences of the original WODS' evalution set) into a *development* and a *test* set. The partition is performed on the recording sequence level, as opposed to the frame level. This is to avoid having highly similar frames (as commonly found within a sequence) distributed between the sets, thus maintaining independence. We assign 50% of the data to each set. Note that due to differences in the sequence characteristics, the resulting frame and annotation counts differ between the sets.



(a) The primary COCO metrics of KAPAO [MVWM21] evaluated on the WODS++ development split over the epochs.



(b) The different loss coefficients of KAPAO [MVWM21] computed on the WODS++ development split over the epochs.

Figure 4.5: Performance measures during the training of KAPAO [MVWM21] on the 2D labels of WODS++.

We keep the settings the same as previously. As opposed to the previous datasets, 75 epochs were not found to be sufficient for convergence here. Seeing the development of losses and performance metrics across epochs, it seems likely that the model could improve even further with longer training (see Figure 4.5). In Figure 4.5a we show the AP and $AP_{.5}$ computed on the development set over the epochs. In Figure 4.5b we show the different training loss coefficients over the epochs, where *box* refers to the loss applied to the bounding box estimation, *obj* penalizes the score indicating the presence of an object of interest, *cls* penalizes the class of object detected (pose or a specific keypoint type), and *kp* refers to the loss applied to the keypoint position estimation as part of the pose object prediction. We did not report similar training records for TDUP or BDD100K (see Sections 4.2.1 and 4.2.2, respectively), since the models seemed to reach peak performance well within the 75 epochs. Still, we do not train further in this case for three reasons: **a)** to keep the setting aligned with the other datasets, **b)** an improvement of a few percentage-points is hard to meaningfully interpret without established baselines, **c)** to save computational resources. Note with respect to the last point that training on the large scale extended WODS dataset is quite compute-intensive. On a single GPU (RTX 2080 Ti), an epoch takes multiple hours.

We report the results in Table 4.3. We mark the settings additionally with *Dev:* and *Test:* to refer to evaluation on the development and test sets, respectively. The former is used during training to keep track of the best model version. We use the COCO evaluation suite[12] with the WODS toolkit's keypoint standard deviations[13].

[12]`https://github.com/cocodataset/cocoapi` [retrieved on 5.8.2022]
[13]`https://github.com/waymo-research/waymo-open-dataset` [retrieved on 13.06.22]

We observe a difference of $\sim 1\%$ percentage-point between the development and test set (compare the `Dev:` rows with their corresponding `Test:` rows in Table 4.3). On either set, the `(VD)` setting improves performance metrics by $\sim 2-4$ percentage-points. However, while the `(VD)` setting improves performance metrics, it may not be effective in practical use, due to the large number of false-positives. On the development set, which has $\sim 170,000$ ground truth annotations, the `(VD)` setting makes $\sim 3,100,000$ pose predictions. Our default configuration (plain `K`), for comparison, produces $\sim 140,000$ predictions. For practical purposes, producing $\sim 80\%$ as many poses as there are ground truth labels (default `K`) seems distinctly more plausible than producing ~ 18 times as many (`(VD)`).

CHAPTER 5

Human Pose Estimation of Vulnerable Road Users in 3D

Previously, in Chapter 4, we established KAPAO variants for traffic datasets in 2D. Furthermore, in Chapter 3, we created a large-scale dataset containing both 3D annotations and LiDAR data. Building on that, we now study the performance of KAPAO adapted for use in 3D pose estimation. Specifically, we examine the following settings:

- We train the original KAPAO on WODS++. Then, we lift its 2D predictions to 3D using the LiDAR point clouds (similar to the procedure proposed in Section 3.1.3).
- A variant of KAPAO is devised that estimates 3D joint locations, based on the image input alone.
- That same 3D KAPAO variant is used, but the input image is enriched with depth data from LiDAR point clouds.

First, we specify the design details of the variants listed above. Also, we discuss how to represent the 3D coordinates to suit the models. Finally, we elaborate on the experiments performed and the results obtained.

5.1 Implementation

Applying KAPAO to said settings requires adjustments in the framework. We will specify them subsequently. Additionally, as in the 2D case previously, we need to accommodate the dataset-specific keypoint definitions. We use the same number, order and variation coefficients as for 2D HPE on WODS++ (see Section 4.2.3).

5.1.1 3D KAPAO Architecture

The prediction of 3D joint positions requires changes to the core model. YOLOv5 was extended for 3D bounding box localization in [MKD+22]. We choose an analogous approach for the 3D localization of keypoints within KAPAO. Specifically, we adopt the following modifications:

- The model is extended to additionally predict the depth of keypoints (see Section 5.1.4 for specifics on the representation chosen for this). This is accomplished by adding additional targets to the complex pose objects that KAPAO regresses. The keypoint objects are only trained to estimate their respective depth. The 2D localization is kept unchanged.

- We adopt the L^1-loss for the distance estimation. It is specified in Equation 5.1.

$$l_{dist} = \frac{1}{|I_B|} \sum_{i \in I_B} |\hat{z}_i - z_i|) \tag{5.1}$$

  Here, I_B is the index set of keypoints in a given batch. The variables z_i and $\hat{z}_i$ represent the true and predicted depth values of a keypoint i, respectively.

  Per default, KAPAO's loss is already a weighted sum of individual loss components. We simply extend it with our additional loss l_{dist}. The combined loss function for the 3D setting is given in Equation 5.2.

$$l_{3D} = l + \delta_{dist} l_{dist} = \delta_{obj} l_{obj} + \delta_{box} l_{box} + \delta_{cls} l_{cls} + \delta_{kps} l_{kps} + \delta_{dist} l_{dist} \tag{5.2}$$

  We add the distance loss l_{dist} weighted by the factor δ_{dist} to the original combined loss l. That previous aggregate loss consists of the objectness (l_{obj}), box (l_{box}), class (l_{cls}), and keypoint (l_{kps}) loss. Each is weighted by their respective factor $\delta_{(\cdot)}$. In practice, the loss is additionally scaled by the number of images in the batch.

- Associated utilities are modified as needed to accommodate the architectural changes. Notably, the target definition changes. This requires adjustment in the loading of labels and subsequent transformation steps.

We call the model with the described architecture *KAPAO 3D*. Without any additional notation, we only use images as input.

5.1.2 Additional LiDAR Input

We now consider LiDAR point clouds to complement images as our input data. We decide on an early fusion approach. Specifically, we enrich the three-channel RGB image with an additional (sparse) depth channel D based on the LIDAR data. It is formalized in Equation 5.3.

$$\forall(i,j) \in \{1, ..., w\} \times \{1, ..., h\} : D(i,j) = \begin{cases} z(i,j), & \text{if } (i,j) \in P \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

Here, P is the set of 2D projections of all LiDAR points. The image height and width is given by w and h, respectively. The function $z(i,j)$ denotes the smallest z among points (x,y,z) that project to the image coordinates (i,j). Note that most of the time only one point projects onto a particular pixel. In case multiple points project onto the same pixel, the closest one is chosen.

We implement this setting in conjunction with the 3D architecture described previously. This variant is referred to as *LiDAR KAPAO 3D*.

5.1.3 2D Prediction Lifting

Next, we present our uplifting approach based on the default 2D KAPAO architecture. A similar procedure to the one described in Section 3.1.3 is employed. As before, we estimate the 3D location of a joint using a weighted sum of the closest points around the 2D keypoint prediction. The weighting is based on the L^2 distance of the projected LiDAR points to the 2D keypoint on the image plane. In contrast to the setting in Section 3.1.3, however, we do not limit the LiDAR points based on 3D bounding box information. Using that additional information would give an unfair advantage to this approach in comparison to KAPAO 3D.

Still, we limit the points considered in the weighted sum. Instead of the 3D bounding box, we take an estimated 2D bounding box. We consider any LiDAR point that is projected within the rectangle, whose lower and upper corner are given by the minimum and maximum keypoint coordinates, respectively. The box' corners are specified in Equations 5.4 and 5.5.

$$box_{lower} = (min_{i \in I}\, x_i, min_{i \in I}\, y_i) \tag{5.4}$$

$$box_{upper} = (max_{i \in I}\, x_i, max_{i \in I}\, y_i) \tag{5.5}$$

Here, I is the set of keypoint indices. Then, x_i and y_i refer to the i'th keypoint's coordinates.

A LiDAR point with its image plane projection at $p = (x,y)$ is considered in the uplifting operation only if $box_{lower} \leq p \leq box_{upper}$. We call this approach *KAPAO*↑, to indicate that it is the default KAPAO setting with subsequent uplifting.

Alternatively, we simply selected the depth of the closest LiDAR point to a particular keypoint. In this case, the performance metrics were worse. Therefore, the approach was discarded.

5.1.4 3D Coordinate Representation

WODS provides 3D annotations and point cloud coordinates in vehicle space. Its axes originate in the center of the vehicle and point forward (the direction of driving), to the side and upwards. The same scene, viewed through different cameras (as one could imagine happening for the side cameras after a 180° degree turn of the vehicle), would then be associated with different coordinates. This representation does not suit camera-based localization methods, like the one we employ.

Therefore, we transform all 3D coordinates to camera space. This coordinate space originates at the center of a particular camera lens. The x and y axes form the plane parallel to the lens and image plane, and the z-axis is perpendicular to that; i.e., it represents the depth. We always transform LiDAR points and 3D labels to the current camera of interest; i.e, the one that corresponds to the currently processed image.

Furthermore, we choose a root-relative representation. Inspired by previous work [MCL19, NFZY19], we split up the task of depth estimation into: **a)** the absolute depth estimation of the root, and **b)** the depth estimation of the other joints relative to the root. Therefore, all non-root joint depths are represented as their displacement from the root joint depth. The root-joint depth itself stays unaltered. This is formalized in Equation 5.6.

$$z_i' = z_i - z_r \forall i \in I \setminus \{r\} \tag{5.6}$$

Here, $\boldsymbol{z}$ and $\boldsymbol{z'}$ represent the original and root-relative joint depth vectors, respectively. I is the index set of all keypoints for a particular pose, with r being the index of the designated root-joint.

We remove all poses whose root joint is not annotated, since we cannot calculate the displacement of the other joints in this case. Note that the choice of the root joint could therefore result in a bias as to which kinds of poses the model can predict well.

Since we filter out all root-less pose annotations, we choose the most frequently available joint in WODS++ as the root. That is the right shoulder, available for 90.5% of poses (estimated on the development set).

5.2 Experiments and Results

5.2.1 Evaluation Metrics

We discuss the methods used to evaluate the proposed methods. The COCO [LMB+14] dataset, and its corresponding suite of evaluation tools[1] provide the prominent OKS-based AP and AR metrics, in various settings. These can naturally be extended to the 3D use-case: we compute the euclidean distance of 3D points, instead of 2D points,

[1] `https://github.com/cocodataset/cocoapi` [retrieved on 5.8.2022]

when computing the OKS scores. The same generalization has been performed for the OKS/ACC metric in [ZSG+21].

Since these metrics try to summarize pose estimation quality in one measure, they combine penalization of joint position inaccuracies, complete detection misses, and false positives. It can be hard to derive conclusions about the model performance in practical terms. Therefore, we also provide the MPJPE that can be more easily interpreted as the joint position error in matched poses. In calculating the MPJPE we rely on the prediction-target matching that is done in the process of calculating the COCO AP and AR metrics. Additionally, we provide what we call the *root joint depth error (RJDE)* and the *non-root joint depth error (NRJDE)*. The former is the L^1 distance of the absolute depth (i.e., the z-coordinate of the root joint) between target and prediction poses. The latter is the average of the L^1 distances of the relative depths (i.e., the z-coordinates of the non-root joints) between target and prediction joints. We defined these metrics to have separate performance measures for estimating the absolute depth of a person, and for predicting the relative positioning of their joints.

Note that we cannot directly re-use the evaluation tools provided in the WODS toolkit[2], since the tools expect predictions and targets to be pre-aligned. The toolkit does not provide functionality to match a prediction to its corresponding target (as opposed to the COCO evaluation tools, for example).

For the 3D KAPAO variant, another transformation is necessary before the metrics can be computed. The model estimates joint locations at pixel coordinates (x, y) and depth z in (root-relative) meters. This mix of quantities (i.e., pixels and meters) is not suitable for interpretable evaluation quantities. Therefore, we use the sensor intrinsics (given in WODS) to transform the image space coordinates and depth to a meter-based (x, y, z) coordinate in camera space. However, at the time of writing, we encountered an issue with the projection tools provided by the dataset, such that there is a slight deviation in results[3]. Therefore, we ensure that our ground-truth is generated using those same projection tools. Then, target and prediction are projected to the same (slightly wrong) space. The L^2 distance between the space of the data provided in WODS (seemingly correct by visual inspection) and the result of the projection is about 5 cm (estimated on the development set).

5.2.2 Model Specifics

We specify the variants in detail. First, we consider the uplifting approach KAPAO↑. The 2D prediction model is already available. We use the KAPAO model established in Section 4.2.3. Specifically, we choose the default variant, plain `K`, according to the

[2] `https://github.com/waymo-research/waymo-open-dataset/blob/e0cc6134ceb2e0910386ad882eb1216ce596b505/waymo_open_dataset/metrics/python/keypoint_metrics.py` [retrieved on 29.7.2022]

[3] `https://github.com/waymo-research/waymo-open-dataset/issues/525` (also issues 521, 493, 146, and 255) [retrieved on 27.7.2022]

conventions introduced in that section. The lifting procedure detailed in Section 5.1.3 is performed with a softmax temperature of 0.25. Analyzing a range of temperatures as in Section 3.1.3, we find that the performance barely changes in this case. Therefore, we keep to the same temperature value that we have chosen in that section (0.25).

The KAPAO 3D variant is trained for 75 epochs on the 3D labels of WODS++. As an initial checkpoint, we use the 2D KAPAO variant that underlies KAPAO↑. All hyperparameters are kept the same. However, there is a new additional hyperparameter: the δ_{dist} that scales the loss applied to the depth prediction. We choose a value of 0.025, which is equal to the 2D keypoint estimation loss scaling factor δ_{kps}. Also, in contrast to the 2D variants considered, we omit the optional pose and keypoint object merging specific to KAPAO. The original paper [MVWM21] reports a minor improvement using that additional step. Due to the lack of baselines on our self-created dataset and the added complexities of 3D prediction, we avoid it in favor of the simpler approach; i.e., taking the unmodified pose object predictions.

The training progress is visualized in Figure 5.1. In Figure 4.5a, we show the AP and AP$_{.5}$ computed on the development set over the epochs. In Figure 4.5b, we show the different training loss coefficients over the epochs, where *box* refers to the loss applied to the bounding box estimation, *obj* penalizes the score indicating the presence of an object of interest, *cls* penalizes the class of object detected (pose or a specific keypoint type), *kp* penalizes the 2D keypoint positions contained in the pose object, and *depth* refers to the KAPAO 3D specific loss applied to the depth information of the pose object prediction. While the best performance was observed after the last epoch, progress seems to have levelled off. Therefore, we do not expect much progress to be achieved with additional epochs. Interestingly, there are two major drops in performance across the epochs. The reasons for the one at epoch ~ 5 is unclear. The one at epoch 25, however, is most likely an artifact of stopping and resuming training at that point.

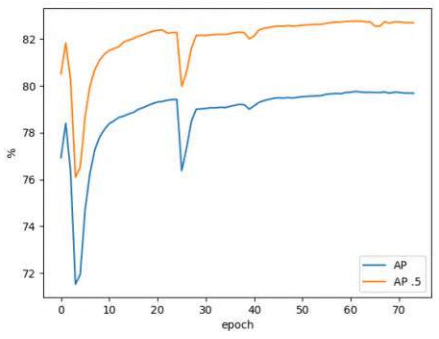
Our LiDAR-KAPAO 3D variant has the same specification as KAPAO 3D. It only differs in the additon of the sparse depth channel. We attempt two trainings: one initialized in the same way as KAPAO 3D and the other with the final trained weights of KAPAO 3D.

Note that both KAPAO 3D and KAPAO↑ showed improvements using more lenient confidence threshold settings (recall `(VD)` from Section 4.2). However, consistent with the discussion in Section 4.2.3, the predictions vastly outnumbered the ground truth, making such configurations impractical.

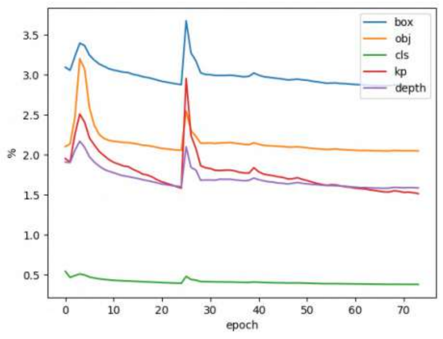
5.2.3 Results

For evaluation, we use the development and test-set split established in Section 4.2.3. Subsequent results are reported on the test-set, the one that has not been used in modelling at all.

First, we find that the LiDAR-KAPAO 3D variant does not perform any better than the image only KAPAO 3D. With neither of the two initial weight configurations that we



(a) The primary COCO metrics of KAPAO 3D evaluated on the WODS++ development split over the epochs.



(b) The different loss coefficients of KAPAO 3D computed on the WODS++ development split over the epochs.

Figure 5.1: Performance measures during the training of KAPAO 3D on the 3D labels of WODS++.

tried did it surpass the performance measures obtained with the image-only version. We suspect that the reason lies in the sparse representation of the data: with only a small fraction of the depth channel having data other than 0, it may be hard for the model to meaningfully incorporate the depth information. Considering that it requires more information than the image-only KAPAO 3D, but performs no better, we do not consider it promising and disregard it in the subsequent comparisons.

In terms of latency, the methods perform similarly. Recall that, for the purposes of inference, KAPAO 3D only differs to the original 2D architecture by adding prediction targets (i.e., the keypoint depths). Therefore, inference speed is comparable to the reports in Section 4.2.1 at ~ 40 ms. The 2D variant has the additional lifting overhead, where measurement depends on when and how the data is loaded. The uplifting procedure itself has negligible latency of < 1 ms.

We focus on the comparison between KAPAO$\uparrow$ and KAPAO 3D. First, we consider the COCO-like AP and AR metrics discussed previously. Results are reported in Table 5.1. Both approaches perform quite similarly across all metrics. The baseline is almost always better, but only by a small margin (< 1 percentage-point). Without knowing general performance baselines, however, the absolute values are hard to interpret.

Less promising results are observed with MPJPE. Results are reported in Table 5.2. We consider overall and body-part specific MPJPE as mentioned in Section 3.2. Predictions are on average more than 3 meters off the target. The baseline method again generally performs better. Interestingly, this is not the case for the head and ankle. It may be that those joints are particularly often occluded or often have background LiDAR points close by; a prominent source for uplifting errors that we will discuss shortly. Also, considering the space we operate in (on the scale of 10s of meters), the results again seem relatively close between the methods.

Method	WODS++ 3D Test-Set									
	AP	AP_{50}	AP_{75}	AP_M	AP_L	AR	AR_{50}	AR_{75}	AR_M	AR_L
KAPAO↑	76.0	78.3	76.5	70.3	89.3	80.9	82.7	81.1	75.8	90.7
KAPAO 3D	75.4	78.0	75.6	70.8	89.4	80.5	82.5	80.9	75.3	90.6

Table 5.1: Performance comparison in terms of AP and AR between the uplifting model KAPAO↑ and the direct 3D pose estimation model KAPAO 3D on the WODS++ test split.

Part	MPJPE (cm)	
	KAPAO↑	KAPAO 3D
head	357	338
shoulder	344	354
elbow	290	350
wrist	290	342
hip	313	353
knee	317	357
ankle	357	338
overall	315	349

Table 5.2: Performance comparison in terms of MPJPE between the uplifting model KAPAO↑ and the direct 3D pose estimation model KAPAO 3D on the WODS++ test split.

To analyse the results more closely, we now consider the (N)RJDE. Results are given in Table 5.3. The mean RJDE is another confirmation of the previous observations: KAPAO↑ is slightly more accurate at an offset of 283 cm compared to the 293 cm of KAPAO 3D. It also seems like a large fraction of the MPJPE we observed is due to the depth error, which also intuitively represents the more challenging task. The NRJDE, on the other hand, is quite different from these other metrics. It is much lower for KAPAO 3D than KAPAO↑. As we will also see later in the visualizations shown in Figures 5.2, 5.3, and 5.4, KAPAO 3D generally produces much more plausible poses. It may misplace them in the scene by a few meters, but the relative joint estimation is quite robust. This cannot be said for KAPAO↑. As we hinted to before, KAPAO↑ often makes mistakes by using the depth of occlusion or background objects for uplifting. We report the standard deviations of the error measures to indicate the large fluctuations in accuracy, that are particularly strong for KAPAO↑.

Finally, in Figures 5.2, 5.3, and 5.4, we visualize our predictions on sample scenes. Looking at multiple samples, we can discern a few patterns in their errors. Both KAPAO↑ and KAPAO 3D seem to identify people correctly overall. There are consistently more

Architecture	RJDE (cm)		NRJDE (cm)	
	Mean	Std. Dev.	Mean	Std. Dev.
KAPAO↑	283	528	143	257
KAPAO 3D	293	516	9	7

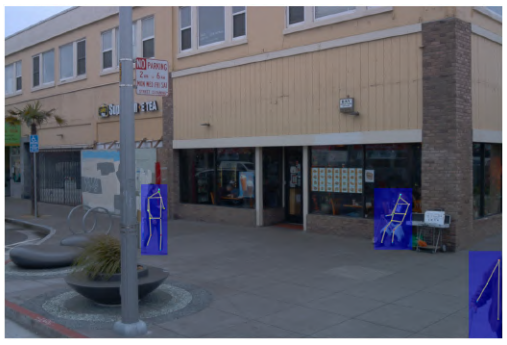
Table 5.3: Performance comparison in terms of RJDE and NRJDE between the uplifting model KAPAO↑ and the direct 3D pose estimation model KAPAO 3D on the WODS++ test split.

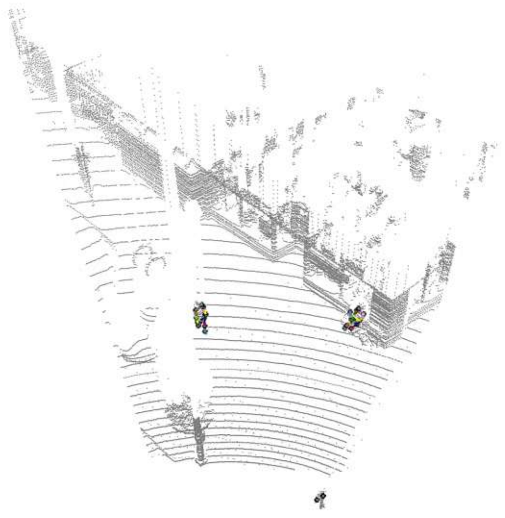
predictions than ground truth. Those surplus predictions, however, do not seem entirely wrong. In those cases, depictions of humans really seem to be present most of the time. Settings in which this frequently occurs are: people inside of cars (or behind any glass), reflections or when the person is on the edge of distinguishability.

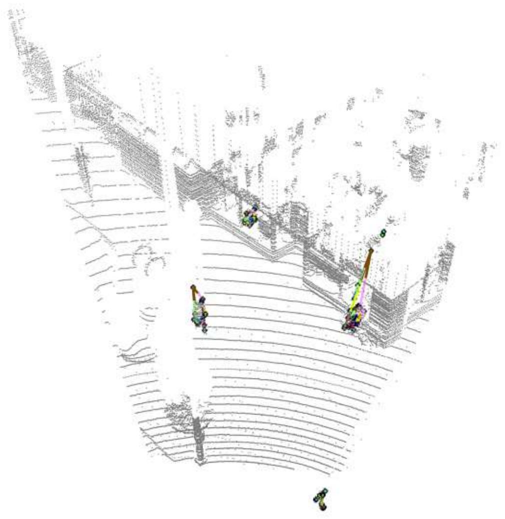
The two methods are quite different in their depth estimation, however. KAPAO↑ tends to estimate most joints of a person accurately, but fails for some. These failings are quite extreme and make for a very unnatural visualization. They either occur due to occlusion or because LiDAR points of the background are mistakenly used for uplifting. The former is a fundamental issue of the approach, which at best could be detected and omitted. The latter could possibly be improved based on heuristics, or also removed based on plausibility checks.

KAPAO 3D, on the other hand, estimates poses that are natural and very close to the ground truth. However, it struggles to place them at the correct absolute depth. This can be seen by focusing on the displacement between the point clouds of the humans and the predictions. In that sense, its limitations are similar to the ones a human has in this regard: the general position of the depicted people, as well as their individual poses, can be estimated; but quantifying the distance exactly is difficult. Without seeing the point cloud, we could hardly tell that the predictions do not match the ground truth.

This is consistent with what we observed in relation to the (N)RDJE in Table 5.3. In terms of root joint localization, both approaches can lead to wrong results (as we have seen, however, in quite different ways). The individual, relative pose information is estimated much more accurately by KAPAO 3D.


(a) The input image with 2D ground truth


(b) 3D ground truth


(c) KAPAO↑ predictions

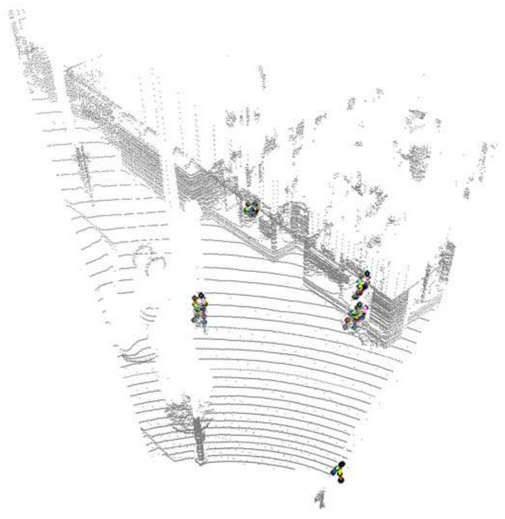
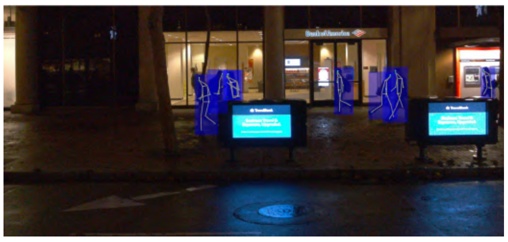
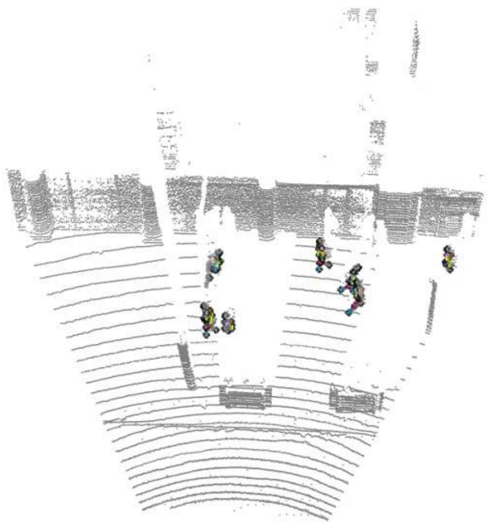
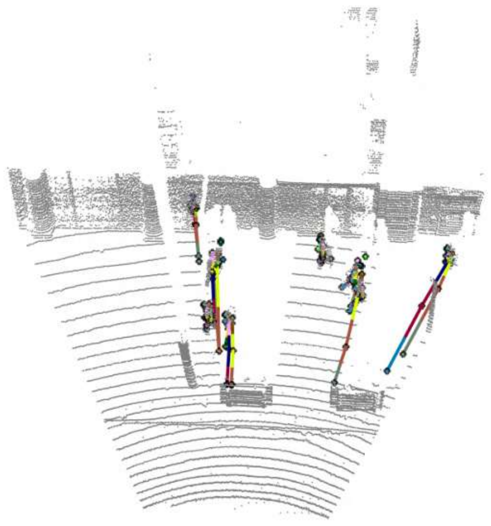

(d) KAPAO 3D predictions

Figure 5.2: First sample scene from the WODS++ test set and our 3D predictions. Each keypoint type has a different color, consistent across people and subfigures.


(a) The input image with 2D ground truth


(b) 3D ground truth


(c) KAPAO↑ predictions

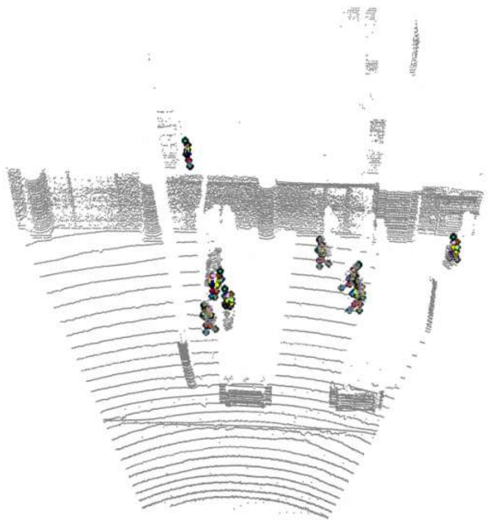
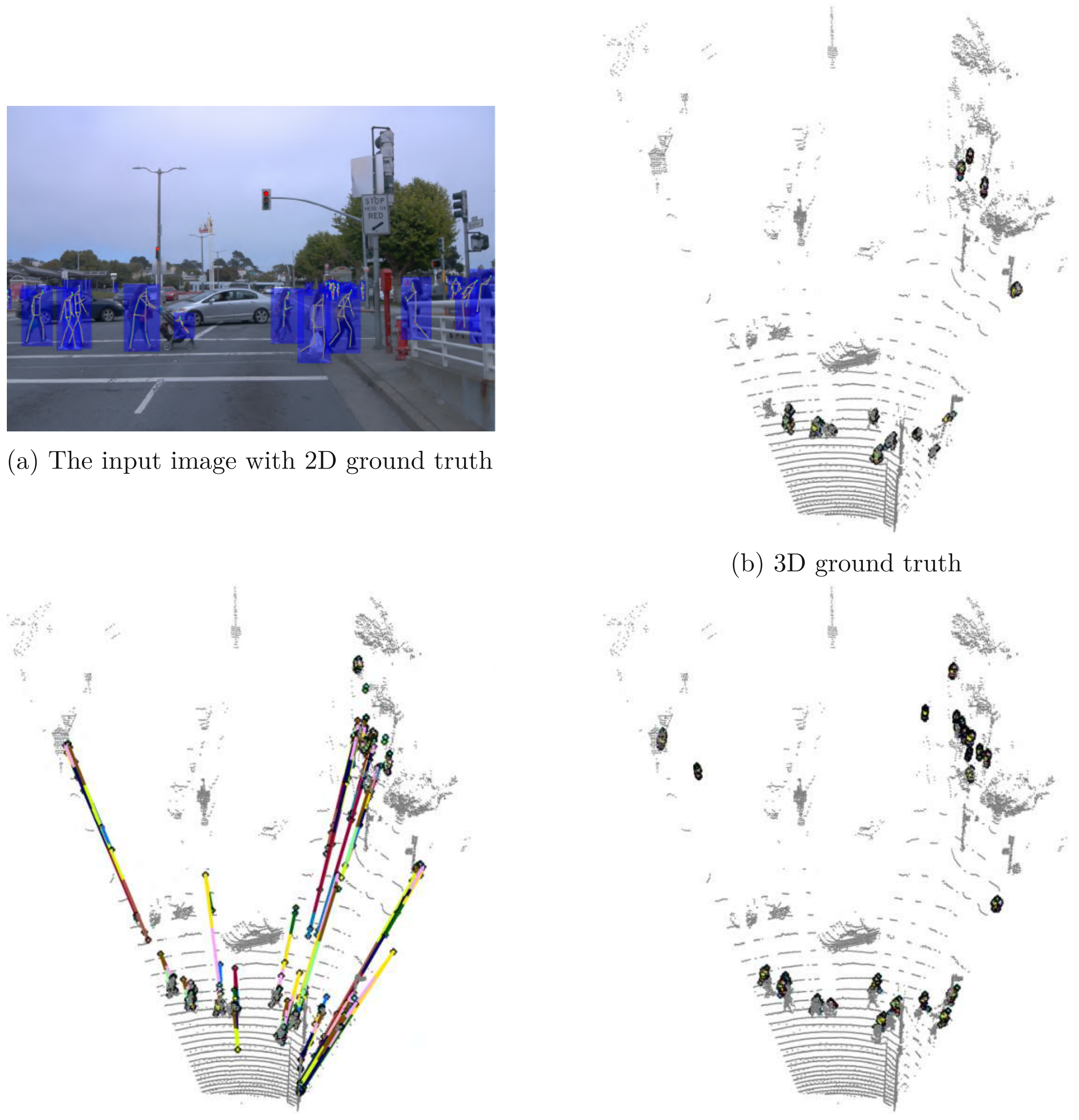

(d) KAPAO 3D predictions

Figure 5.3: Second sample scene from the WODS++ test set and our 3D predictions. Each keypoint type has a different color, consistent across people and subfigures.



(a) The input image with 2D ground truth

(b) 3D ground truth

(c) KAPAO↑ predictions

(d) KAPAO 3D predictions

Figure 5.4: Third sample scene from the WODS++ test set and our 3D predictions. Each keypoint type has a different color, consistent across people and subfigures.

CHAPTER 6

Conclusion

6.1 Discussion

In this work, we studied KAPAO for application in the VRU HPE task. We separately considered 2D and 3D localization. For the former setting, public benchmark datasets are available, and KAPAO can be applied in its original design. For 3D, however, the model architecture requires adjustment and applicable datasets are not as easily available.

Therefore, as a first step, we extended the WODS to have large-scale 3D human pose annotations. We leveraged the existence of 2D & 3D bounding box labels for all identifiable humans as a replacement for the initial object detection stage of a two-stage pose estimator. That two-stage pose estimator, namely HRNet, was trained on the existing WODS pose labels and applied to the exhaustive box labels. The 2D pose predictions were then lifted to 3D using the LiDAR data included with the dataset. After a number of filtering steps to increase the label quality, we were left with over one million 2D and 3D pose labels. We evaluated their quality on a holdout set consisting of original WODS pose labels. Our performance metrics are consistently better than those reported on a related dataset [ZSG+21]. In absolute terms, we achieve a 3D MPJPE of less than $\sim$ 10 cm. Visual inspection of samples indicate that the generated poses are consistently plausible.

Next, we studied KAPAO using 2D VRU HPE benchmark datasets. Specifically, we considered the datasets TDUP [WYW+21] and BDD100K [YCW+20]. We experimented with various changes pertaining to the training procedure, the inference hyperparameters, initial model checkpoints, and even the architecture. Our best settings outperformed other single-stage methods consistently, and were generally slightly worse, but competitive with the best reported top-down approaches. Additionally, KAPAO has a lower latency than any reported competitive approach.

We also compared our KAPAO variants among themselves. We found that image augmentations performed by default during training in the KAPAO framework, such as mosaicking, led to worse performance. The default inference setting with a low confidence threshold may perform well on prominent evaluation metrics; but it may be impractical, due to the high-rate of false positives it produces. By varying initial checkpoints, we found that pre-training on the large-scale HPE datasets (COCO Keypoints and the extended WODS) performed best. Starting from another small domain-specific VRU dataset (BDD100K and TDUP) degraded performance. Checkpoints from HPE datasets, irrespective of scale, led to better performance than ones pre-trained on object detection data (COCO Detection) only. Then, considering higher resolution output grids resulted in a slight improvement in the estimation of small poses, a decrease in performance overall, and a significant cost in latency, thus making for an unfavorable trade-off.

Finally, we performed 3D HPE using KAPAO. We use the extended WODS that we created for this purpose. Primarily, two ways of adapting KAPAO for 3D were considered. For one, we extend KAPAO's architecture to estimate poses in 3 dimensions directly (KAPAO 3D). The other variant uses KAPAO's default 2D predictions and estimates their depth using the LiDAR data in a post-processing step (KAPAO↑). A third variant, where we added LiDAR information as an additional sparse depth channel to the image input of KAPAO 3D, could not outperform the image-only KAPAO 3D. It was, therefore, not considered any further. We found that across popular metrics, such as AP and MPJPE, KAPAO↑ outperformed KAPAO 3D slightly. KAPAO 3D shows desirable qualities, however, when considering the depth estimation error in more detail, it still has slightly less accuracy in positioning the pose in absolute terms in space. However, it produces poses that are much more accurate relative to the root joint. Visual inspection of the results confirm that KAPAO 3D produces plausible poses, comprehends the scene correctly at large, but does not manage to place them precisely at the right distance from the camera.

6.2 Future Work

We identified various potential extensions of the work presented, that we did not explore within the scope of this thesis. We list them subsequently.

A general way of improving the approach would be to consider sequence information. We only considered data that pertains to one specific frame. However, in WODS specifically (and in practical autonomous driving settings) video sequences are available. This could improve estimation accuracy, and even bring additional qualities, such as stable predictions across frames.

We evaluated the extended WODS based on the original version. We did not, however, compare the extended against the original in terms of their utility as training sets. This would help to determine the value of the extended WODS, and pseudo-labels more generally. Due to the size of these datasets, however, such a study requires the availability of extensive computational resources.

In general, the pseudo-label generation for extending WODS can be improved. The top-down model acting as the label generator could be further fine-tuned. So can the subsequent filtering steps. In our estimation, however, rethinking the procedure for creating the 3D labels is most promising. We lifted 2D labels using a simple procedure based on surrounding LiDAR points. Incorporating more complex neural methods here could lead to even higher accuracy.

Furthermore, the examination of initial checkpoints for 2D KAPAO could be extended. We considered the datasets only in a sequential way: the model that was trained on one dataset is used as the initial checkpoint for training on another. Better characteristics in terms of generalization may be achieved by mixing the datasets and training on the merged set jointly.

Next, we identified a number of additional paths to explore with the application of KAPAO in 3D. For one, the LiDAR based KAPAO 3D was only studied in the form of one specific modality-fusion approach. Intuitively, it should be possible to use the additional LiDAR data to bring an improvement. Possibly, features of the point cloud could be extracted and merged with latent image representations within the network at a slightly later point. Another way may be to impute missing values into the depth channel, making it less sparse.

Finally, we found KAPAO 3D to have some desirable characteristics in prediction. The related evaluation was limited to the VRU HPE task only. The model architecture, however, is not specific to that domain, and could be just as well suited for any other 3D HPE setting.

List of Figures

List of Tables

Acronyms

AP average precision. 3, 14, 15, 25, 35, 37, 39–41, 46–50, 56, 61

AR average recall. 15, 40, 46, 47, 49, 50, 61

AV autonomous vehicles. 1, 2, 9, 10, 30, 33

BDD100K Berkley DeepDrive 100K. 2–4, 11, 12, 15, 29, 35, 38–41, 55, 56, 59, 61

COCO Common Objects in Context. 10, 11, 15, 18, 29, 32, 33, 35, 38, 40, 41, 46, 47, 49, 56, 59

HPE human pose estimation. 1–5, 7–12, 14, 15, 17, 18, 29, 30, 33, 38, 43, 55–57, 61

IoU intersection-over-union. 15

LAMR log average miss rate. 3, 15, 37

MPJPE mean per joint position error. 3, 15, 25, 47, 49, 50, 55, 56, 62

NMS non-maximum suppression. 8, 38

NRJDE non-root joint depth error. 47, 50, 51, 62

OKS object keypoint similarity. 3, 14, 15, 19, 21–23, 25, 26, 46, 47, 59, 61

RJDE root joint depth error. 47, 50, 51, 62

TDUP Tsinghua-Daimler Urban Pose. 2–4, 9–11, 15, 19, 29, 31, 35–37, 39–41, 55, 56, 59–61

VRU vulnerable road user. 1–4, 9–11, 29, 30, 33, 55–57, 61

WODS Waymo Open Dataset. 2, 3, 5, 12, 13, 17, 18, 20–28, 32, 33, 35, 36, 38, 40, 41, 43, 46–57, 59–62

Bibliography

[BKL21] Guillem Brasó, Nikita Kister, and Laura Leal-Taixé. The center of attention: Center-keypoint grouping via attention for multi-person pose estimation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11833–11843, 2021.

[BTCB21] Aduen Benjumea, Izzeddin Teeti, Fabio Cuzzolin, and Andrew Bradley. YOLO-Z: improving small object detection in yolov5 for autonomous vehicles. *CoRR*, abs/2112.11798, 2021.

[CV18] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6154–6162, 2018.

[CWP$^+$18] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7103–7112, 2018.

[CXW$^+$20] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S. Huang, and Lei Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5385–5394, 2020.

[CYQW19] Pablo Rodrigo Gantier Cadena, Ming Yang, Yeqiang Qian, and Chunxiang Wang. Pedestrian graph: Pedestrian crossing prediction based on 2d pose estimation and graph convolutional networks. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2000–2005, 2019.

[DWSP12] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[FGS$^+$21] Michael Fürst, Shriya T. P. Gupta, René Schuster, Oliver Wasenmüller, and Didier Stricker. Hperl: 3d human pose estimation from rgb and lidar. In *25th International Conference on Pattern Recognition (ICPR)*, pages 7321–7327, 2021.

[FLC+20] Matteo Fabbri, Fabio Lanzi, Simone Calderara, Stefano Alletto, and Rita Cucchiara. Compressed volumetric heatmaps for multi-person 3d pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7202–7211, 2020.

[FMJZ08] Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. Progressive search space reduction for human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[FXTL17] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2353–2362, 2017.

[FZG+20] Zhijie Fang, Wuqiang Zhang, Zijie Guo, Rong Zhi, Baofeng Wang, and Fabian Flohr. Traffic police gesture recognition by pose graph convolutional networks. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1833–1838, 2020.

[GSX+21] Zigang Geng, Ke Sun, Bin Xiao, Zhaoxiang Zhang, and Jingdong Wang. Bottom-up human pose estimation via disentangled keypoint regression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14676–14686, 2021.

[GWH19] Renshu Gu, Gaoang Wang, and Jenq-neng Hwang. Efficient multi-person hierarchical 3d pose estimation for autonomous driving. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 163–168, 2019.

[HZGH20] Junjie Huang, Zheng Zhu, Feng Guo, and Guan Huang. The devil is in the details: Delving into unbiased data processing for human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5699–5708, 2020.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[IPOS14] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014.

[JCS+22] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, 2022.

[KBA19] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. Pifpaf: Composite fields for human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11969–11978, 2019.

[KJZ+18] Viktor Kress, Janis Jung, Stefan Zernetsch, Konrad Doll, and Bernhard Sick. Human pose estimation in real traffic scenes. In *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 518–523, 2018.

[KRC+21] Chandan Kumar, Jayanth Ramesh, Bodhisattwa Chakraborty, Renjith Raman, Christoph Weinrich, Anurag Mundhada, Arjun Jain, and Fabian B. Flohr. Vru pose-ssd: Multiperson pose estimation for automated driving. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):15331–15338, 2021.

[LBZ+21] Jiefeng Li, Siyuan Bian, Ailing Zeng, Can Wang, Bo Pang, Wentao Liu, and Cewu Lu. Human pose regression with residual log-likelihood estimation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11005–11014, 2021.

[LJN+19] Junwei Liang, Lu Jiang, Juan Carlos Niebles, Alexander G. Hauptmann, and Li Fei-Fei. Peeking into the future: Predicting future person activities and locations in videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5718–5727, 2019.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, 2014.

[LWH+21] Zhengxiong Luo, Zhicheng Wang, Yan Huang, Liang Wang, Tieniu Tan, and Erjin Zhou. Rethinking the heatmap regression for bottom-up human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13259–13268, 2021.

[LWY+19] Wenbo Li, Zhicheng Wang, Binyi Yin, Qixiang Peng, Yuming Du, Tianzi Xiao, Gang Yu, Hongtao Lu, Yichen Wei, and Jian Sun. Rethinking on multi-stage networks for human pose estimation. *CoRR*, abs/1901.00148, 2019.

[LWZ+21] Ke Li, Shijie Wang, Xiang Zhang, Yifan Xu, Weijian Xu, and Zhuowen Tu. Pose recognition with cascade transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1944–1953, 2021.

[LYZ+21] Yanjie Li, Sen Yang, Shoukui Zhang, Zhicheng Wang, Wankou Yang, Shutao Xia, and Erjin Zhou. Is 2d heatmap representation even necessary for human pose estimation? *CoRR*, abs/2107.03332, 2021.

[MCL19] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10132–10141, 2019.

[MKD+22] A. Mauri, R. Khemmar, B. Decoux, M. Haddad, and R. Boutteau. Lightweight convolutional neural network for real-time 3d object detection in road and railway environments. *Journal of Real-Time Image Processing*, 19(3):499–516, jun 2022.

[MSM+18] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. Single-shot multi-person 3d pose estimation from monocular rgb. In *International Conference on 3D Vision (3DV)*, pages 120–130, 2018.

[MVWM21] William J. McNally, Kanav Vats, Alexander Wong, and John McPhee. Rethinking keypoint representations: Modeling keypoints and poses as objects for multi-person human pose estimation. *CoRR*, abs/2111.08557, 2021.

[NFZY19] Xuecheng Nie, Jiashi Feng, Jianfeng Zhang, and Shuicheng Yan. Single-stage multi-person pose machines. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6950–6959, 2019.

[NHD17] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 2274–2284, 2017.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[RWS20] G. Rogez, P. Weinzaepfel, and C. Schmid. Lcr-net++: Multi-person 2d and 3d pose detection in natural images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(05):1146–1161, 2020.

[SHZ+18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018.

[SKD+20] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurélien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin

Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2443–2451, 2020.

[SXLW19] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5696, 2019.

[WFX$^+$19] Sijia Wang, Fabian B. Flohr, Hui Xiong, Tuopu Wen, Baofeng Wang, Mengmeng Yang, and Diange Yang. Leverage of limb detection in pose estimation for vulnerable road users. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 528–534, 2019.

[WNQ$^+$22] Zitian Wang, Xuecheng Nie, Xiaochao Qu, Yunpeng Chen, and Si Liu. Distribution-aware single-stage models for multi-person 3d pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13096–13105, 2022.

[WYW$^+$21] Sijia Wang, Diange Yang, Baofeng Wang, Zijie Guo, Rishabh Verma, Jayanth Ramesh, Christoph Weinrich, Ulrich Kreßel, and Fabian B. Flohr. Urbanpose: A new benchmark for vru pose estimation in urban traffic scenes. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1537–1544, 2021.

[XWW18] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Computer Vision – ECCV 2018*, pages 472–487, Cham, 2018. Springer International Publishing.

[YCW$^+$20] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2633–2642, 2020.

[YR13] Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, 2013.

[ZMZ$^+$18a] Andrei Zanfir, Elisabeta Marinoiu, Mihai Zanfir, Alin-Ionut Popa, and Cristian Sminchisescu. Deep network for the integrated 3d sensing of multiple people in natural images. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[ZMZ$^+$18b] Andrei Zanfir, Elisabeta Marinoiu, Mihai Zanfir, Alin-Ionut Popa, and Cristian Sminchisescu. Deep network for the integrated 3d sensing of multiple people in natural images. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[ZSG$^+$21] Jingxiao Zheng, Xinwei Shi, Alexander Gorban, Junhua Mao, Yang Song, Charles R. Qi, Ting Liu, Visesh Chari, Andre Cornman, Yin Zhou, Congcong Li, and Dragomir Anguelov. Multi-modal 3d human pose estimation with 2d weak supervision in autonomous driving. *CoRR*, abs/2112.12141, 2021.

[ZZD$^+$20] Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. Distribution-aware coordinate representation for human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7091–7100, 2020.