Faculty of Engineering
Cooperative Automated Transportation Systems Laboratory

Master Thesis

Human Body Pose Detection Using Depth Sensors

by

Kerim Turacan

Date of submission: March 5, 2024

Autor:

Kerim Turacan
Friedrich-Ebert-Straße
Stockstadt a. Main

Matrikel-Nr.: 2218375

Studiengang: Angewandte Forschung in den Ingenieurwissenschaften
Vertiefungsrichtung: Angewandte Informatik

Prüfer:

Prof. Dr.-Ing. Konrad Doll
Labor für Kooperative automatisierte Verkehrssysteme

Technische Hochschule Aschaffenburg
Fakultät Ingenieurwissenschaften
Würzburger Straße 45
D-63743 Aschaffenburg

Ehrenwörtliche Erklärung

Kerim Turacan
Friedrich-Ebert-Straße
Stockstadt a. Main

Hiermit erkläre ich, dass ich die von mir vorgelegte Arbeit mit dem Thema "*Human Body Pose Detection Using Depth Sensors*"' selbstständig verfasst habe, dass ich die verwendeten Quellen, Internet-Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen und Bildern –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Aschaffenburg, March 5, 2024

_____

Kerim Turacan

Contents

Chapter 1

1 Introduction

Human Pose Estimation (HPE) is one of the most actively researched areas in computer vision that aims to identify and locate human body joints in an image or video. Typically, it is the inference of joint coordinates and the reconstruction of a human skeletal representation. [11] HPE has a multitude of technological domains, including Virtual Reality (VR)/ Augmented Reality (AR), AI-assisted healthcare, security, surveillance, and autonomous/intelligent vehicles (AV/IV).

Autonomous driving and driving assistance systems can greatly benefit from advancements in HPE. By accurately detecting and tracking the positions and orientations of human body joints, autonomous vehicles can better understand the actions and intentions of pedestrians and other road users. For example, by using cameras and sensors to detect and track the movements of pedestrians, an autonomous vehicle can predict where they are going and adjust its own path accordingly to avoid any potential collisions. Moreover, HPE can also be used in the development and testing of AVs. By accurately simulating human movements and behaviors, researchers can create more realistic driving scenarios and evaluate the performance of their autonomous driving algorithms.
A major challenge for automotive driving sensors is to provide accurate, real-time, minimal-latency sensing of a vehicle's surroundings while it is traveling at high speed [6]. Currently, the Mercedes-Benz DRIVE PILOT [1] is the only SA3-certified vehicle in the U.S., which uses a LiDAR-based vision system. The industry's push for more automated driving features heavily relies on advanced sensing, and almost all manufacturers exploring autonomous driving view LiDAR as a critical enabling technology.

Despite the high level of interest in human pose estimation in the wild (in outdoor environments), only a few papers approached outdoor 3D keypoint detection using LiDAR point clouds. A primary factor is the necessity of a large amount of high-quality outdoor ground-truth labels for training a pedestrian pose estimation model. The process of labeling 3D human keypoints in point cloud data is expensive, time-consuming and error-prone. Although there are a few existing point cloud datasets [2] with ground-truth human poses, they are often limited in terms of quantity, resolution, diversity for

long-tail scenarios or region-specific settings (a dataset collected in one geographical or urban setting might not generalize well to different settings).

Pose estimation using point cloud data presents distinct challenges compared to traditional image perception. The structure of point clouds is fundamentally different due to their unordered nature. Unlike images, which are composed of pixel grids with consistent and structured spatial relationships, point clouds consist of a collection of points scattered in three-dimensional space without any inherent order. Furthermore, point clouds are characterized by their sparsity. Such data pose challenges for directly processing the data with an end-to-end model. In AV scenarios, the time and space complexities in performing real-time tasks are very important. Recent networks that directly process each point individually from the point cloud (i.e., 3D CNNs) are computationally expensive. The time and space complexities of the 3D CNN grow cubically with the resolution of the input 3D volume. [67] In AV scenarios, this can quickly become critical. Thus alternative approaches need to be considered.

1.1 Literature Review

This section presents an overview of concepts for 3D human pose estimation models, including different input modalities. Early approaches in this field have used model-driven methods [13, 34, 26, 44, 48, 41], which rely on the use of a mathematical, pre-defined model of the human body to estimate the position of body joints by fitting the model to the input image data, incorporating handcrafted features, perspective relationships, and geometric constraints. These methods often rely on iterative optimization and may require sensitive initialization to converge to the optimal pose. While model-based approaches offer a wide range of solutions, they tend to be computationally expensive, may not generalize well to complex or uncontrolled environments, and require a significant amount of rich, visual information, usually obtained from a multi-camera system. Therefore, as a result, they may not perform as well in all scenarios, thus not in the AV scenario.

1.1.1 3D HPE on monocular images

In recent years, deep neural learning methodologies have been most dominantly used, particularly for image-to-3D-HPE by a single camera or multiple cameras. Unlike multi-camera systems that provide a rich source of visual information by capturing the subject from various angles, monocular image-based estimation relies on a single viewpoint. This constraint presents unique challenges, primarily due to the loss of depth information, making the task of accurately predicting 3D human poses more complex. Since the monocular 3D pose estimation problem inherently suffers from predicting the 3D human pose on an absolute scale, most studies relax this problem to predict root relative distances by defining the pelvis as the origin or root node. This is referred to as either a root-relative, person-centric, or ego-centric 3D human pose estimation problem, whereas the absolute 3D pose estimation problem is referred to as a camera-centric 3D

HPE problem. Compared to the root-relative 3D human pose, the absolute 3D human pose is beneficial for applications such as surveillance systems and autonomous vehicles, where real-scale human motion and body size are crucial.

In monocular 3D pose estimation, two main approaches have emerged for supervised learning:

1. Models that learn to regress the 3D pose directly from image features [54, 24, 38, 33, 65] and
2. Pipeline approaches where the problem is split up, with the 2D pose first estimated and then lifted into 3D [29, 40, 31, 22, 56, 39, 10, 5, 9].

1. Direct regression 3D HPE from monocular images

The direct regression method allows for predicting 3D human poses directly from images without first detecting 2D keypoints as an intermediate step by extracting contextual and pose information. These architectures are end-to-end trainable using a single cost function. However, these advantages come with notable challenges:

- The datasets require pairs of images and their corresponding precise 3D annotation.
- They are sensitive to target environment changes since these models learn the mapping from images to 3D poses based on the specific characteristics of the training data.
- The complexities of human motion and 3D spatial relationships must be accurately modeled in the cost function.

Tekin et al. [54] developed a technique combining an auto-encoder with traditional CNN that learns complex dependencies between different body parts. Furthermore, Karticioglu et al. [24] extended this model by incorporating Long Short-Term Memory (LSTM) networks, enhancing the model's capability to maintain temporal consistency across frames, thus improving the dynamic pose estimation accuracy. Pavlakos et al. [38] proposed a novel method that utilizes a fine discretization of 3D space surrounding the subject, employing a CNN to predict the likelihood for each joint within a voxelized space. Newell et al. [33] introduced the Stacked Hourglass Networks, a significant architectural innovation designed to iteratively refine pose estimates through a process of pooling and upsampling across multiple scales, allowing the model to capture both local and global skeletal information. Their work laid the groundwork for subsequent research in the field, including further variations and improvements on the stacked hourglass design, such as the adaptations by Xu et al. [65], which introduced Graph Stacked Hourglass Networks. This takes advantage of the skeletal structure's graph nature, enabling more effective processing and interpretation of human poses by leveraging the connectivity and relationships between joints.

2. Multi-step 3D HPE from monocular images

As CNNs have become more prevalent, 2D joint estimation has become increasingly reliable and many recent works have looked to exploit this using a pipeline approach, where the 2D poses are generated by a state-of-the-art 2D detector and then subsequently lifted in a next step. The lifting methods are divided into two categories: absolute [39, 10, 5, 9] and root-relative [29, 40, 31, 22, 56] pose estimation approaches. The latter is more prominent in research. Both methodologies will be covered, focusing on the lifting step.

Root-relative 2D-to-3D pose lifting: The root-relative approach estimates the normalized-scale 3D human pose while predicting an ego-centric pose instead of predicting the real-scale 3D pose. They use either image or 2D poses as input. The normalization typically includes both the 2D inputs and 3D outputs getting normalized. The specific normalization procedure often varies. One simple implementation is denoted in [29] by subtracting the data by its mean and dividing by the standard deviation. Since they do not predict the global position of the 3D pose, they are zero-centering the 3D poses around a root joint, typically the hip or pelvis joint. This is why these methods are referred to as root-relative approaches. The baseline is mostly a small multilayer perception (MLP), which aims to predict the root-relative depth of each joint. Many works added additional anatomical constraints for generating "physically logical" poses, such as limb length ratios, and joint-angle limits.

Commonly, by applying inverse normalization, the real scale pose can be recovered, but the translation, thus the real position in 3D space, cannot. This also means that in most works, normalization parameters need to be recalculated, and the model needs to be trained for every new training data gathered.

Instead of using normalized pixel coordinates as input, Miura et al. [31] converted the 2D joint locations to 3D unit vectors for known camera intrinsic parameters. The 3D unit vectorization confines the impact of camera optics by the module parameters according to the intrinsic characteristics. This means that the amount of training data can be synthetically upscaled by transforming 3D ground-truth joint positions, e.g., from publicly available 3D mocap datasets, into 3D unit vectors that are image-independent in the training phase. This advantage alleviates the data collection and training burden due to changes in camera optics and setups.

This methodology is fine for tasks subsequent to HPE, e.g., pose or action recognition, but in applications where true depth estimation and the absolute position of objects are required (e.g., IV), post-processing architectures are additionally needed, or alternatively, lifting methods that consider the absolute position.

Absolute 2D-to-3D pose lifting: The absolute-scale approach estimates a real-scale 3D human pose using a 2D pose as the input. Pavllo et al. [39] incorporate a regression of the 3D trajectory of the person. Combining the root-relative 3D pose with the additional prediction of the global root joint position results in an absolute 3D pose. Note, that this trajectory is achieved in contiguous video data rather than on a single-image basis. The authors of [10] use a pose-lifter that yields the canonical root depth (i.e., camera or focal-length independent Z-depth) and the root-relative pose. The

absolute 3D pose can be reconstructed with perspective projection for known focal length information. Unfortunately, they evaluated the root-relative pose and root-depth separately and not the reconstructed absolute pose. Using Z-depth estimations also affects the horizontal and vertical cartesian components x and y of the absolute root joint location and thus alters the final pose location. To disentangle the Z-depth interdependencies from the other components, [5] use a spherical coordinate system that naturally separates the depth component from the angular components. This separation allows for the independent adjustment of an object's depth without directly affecting its angular position, thereby reducing the compounded errors that can occur in cartesian coordinates due to perspective distortion. A very recent work [9] used two baseline networks to simultaneously predict the absolute root and root-relative distance in combination with a correction network that refines the predicted results. This model considers relevant constraints, including body length symmetry and directional constraints.

1.1.2 3D HPE from point clouds

There has not been a lot of work on 3D HPE from LiDAR information due to the lack of ground-truth 3D human pose annotations paired with LiDAR data, especially from in-the-wild point clouds. Many datasets have fitted pose information into point clouds from SMPL [27] models, with subjects being relatively close to the capture system, which are less noisy than outdoor equivalents. Zhou et al. [68] use 3D point clouds generated by indoor 2D depth maps and directly regress the 3D pose by adapting DGCNN [62] and PointNet [42]. However, the final pose is ego-centric, and due to the utilization of a depth camera setup, the human data points are captured in short-range and indoor environments.

The Waymo dataset [52] is a diverse autonomous driving dataset, which includes 2D and 3D keypoint labels with correspondence between 2D (camera) and 3D (LiDAR) labels (pedestrian only). Few works utilize this dataset for 3D HPE from point clouds but mostly rely on weak supervision.

Zheng et al. [66] propose one of the first approaches targeting the AV setting. The network takes an RGB image as input and generates a keypoint heatmap. The heatmap gets smoothed by Gaussian kernel and sampled at the locations of the 2D projections of the point cloud. These features are then concatenated with the point coordinates and are fed into a 3D regression network, which includes a segmentation branch to introduce stronger supervision.

Weng et al. [64] very recently introduced a self-supervised learning method by first training a transformer-based keypoint predictor and body part segmentation predictor supervised on a small synthetic dataset (3D keypoint labels are generated using the SMPL model, and no manual labeling is needed). The next stage refines the predictors by unsupervised loss functions using unlabeled in-the-wild data (a large amount of training data is needed for the unsupervised stage; they used 200,000 samples).

1.2 Scope of this work

This work presents a novel absolute 2D-to-3D pose lifting methodology using LiDAR data as the single input modality suitable in a AV setting. This two-step approach comprises a 2D pose estimator and a subsequent 2D-to-3D lifting network, similar to the approaches introduced in Section "Absolute 2D-to-3D pose lifting". In the first stage, LiDAR data get transformed into an image-like structure and fed into a 2D pose estimator. Inspired by the baseline lifting network of [9], this stage "lifts" the joint predictions of the 2D joint estimator into 3D. Similar to [5] and [31], instead of using normalized 2D pixel coordinates as the input of the lifting network, they get transformed into unit direction vectors. As both stages are independent of each other, the lifting network does not require manually labeled 3D annotations from point clouds; rather, they can be synthetically generated. The only requirement is that the underlying skeleton model and representation must be identical. Indeed, pairs of labeled data can be easily generated by augmenting 3D joints with viewpoint changes or various poses and reprojecting them onto image coordinates.
Because of time limitations, the input samples of the 2D pose estimator are synthetically generated using an autonomous driving simulation environment. Testing on real-world LiDAR datasets, e.g., on the Waymo dataset [52], only requires fine-tuning the 2D pose estimator. This has been quantitatively tested by applying convolution on the projected LiDAR data (more details about this transformation will be covered in Chapter 3.2) with smoothing kernels, such as the Gaussian kernel, before feeding it into the 2D pose estimator. However, as this has not been qualitatively tested, this work serves as a foundational concept and establishes vital groundwork. Extending this to real-world data remains an essential progression in realizing the full potential of this research.

The main contributions of this work are as follows:

- Projection of LiDAR data to a grid-based (image-like) structure using HHA-inspired channel encoding
- 2D keypoints predicted by the 2D estimator are converted into unit direction vectors, serving as the input for a lifting architecture. This architecture comprises two regression networks that predict both the one-dimensional root distance and the root-relative distance of other joints, ultimately generating the final 3D output pose.

The subsequent sections of this thesis are delineated into five principal chapters. In chapter 2, the foundational concepts and technologies used in this work are introduced; particularly basics about HPE and human body representations and an overview of LiDAR technology. Chapter 3, furnishes an in-depth exploration of the dataset used in this work, followed by the core part of this work, Chapter 4, about the details of the approach and its implementation. In chapter 5, the performance of the contributed 3D HPE model gets evaluated. Finally, the last chapter 6 concludes with the contribution of this work and a discussion of improvements and future directions.

Chapter 2

2 Basics

This chapter gives a comprehensive overview of the task of estimating the human pose and how the human pose can be constructed and visually represented. Additionally, application areas where HPE benefits are presented. With the use of the LiDAR sensor as the primary sensor type used in this work, an in-depth overview of LiDAR technology, device variants, and functionalities are elucidated.

2.1 Human Pose Estimation

Human Pose Estimation (HPE) is a computer vision task that involves detecting and localizing the human body joints or keypoints in an image or video. The objective of HPE is to identify the keypoints that define the human body's pose, such as the head, neck, shoulders, elbows, wrists, hips, knees, and ankles. These are typically represented as 2D or 3D coordinates and are used to create a representative model of the human body.
There are three types of approaches to modeling the human body in HPE (illustrated in Fig. 2.1):

- Skeleton-based model: This approach involves representing the human body as a set of connected points that form a skeleton-like structure. The connections between the joints are defined by a set of rules or constraints that ensure the anatomical correctness of the model.
- Contour-based model: This approach involves representing the human body as a set of contours or shapes that define the body's silhouette. The keypoints are then detected by analyzing the contour shape and its properties, such as curvature and orientation.
- Volume-based model: This approach involves representing the human body as a 3D volume or voxel grid, where each voxel corresponds to a small 3D element of the human body. The keypoints are then detected by analyzing the 3D shape of the volume and its properties, such as density and shape.

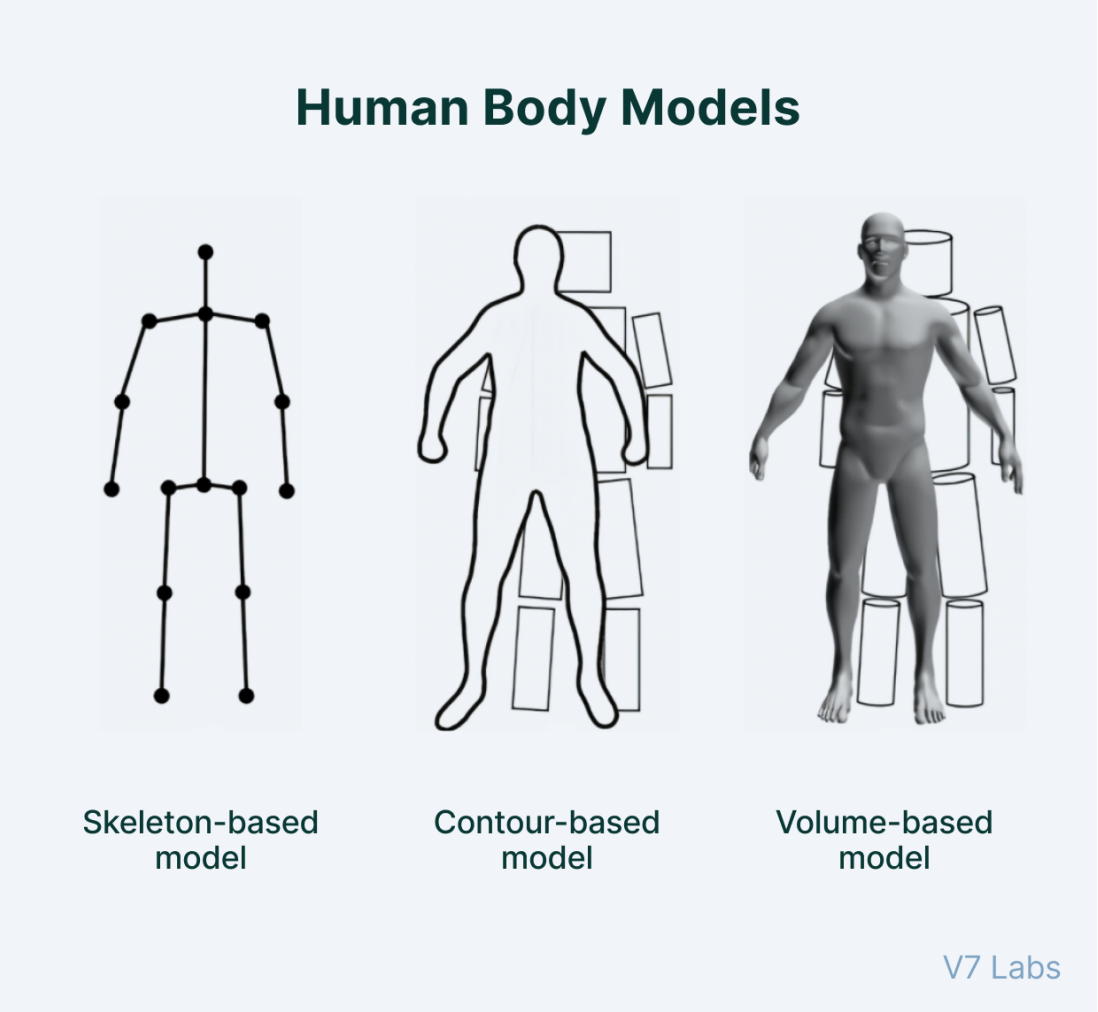


Figure 2.1: Types of approaches to model the human body [3]

Some of the popular HPE applications are:

- Action recognition: Recognizing and classifying human actions, such as walking, running, jumping, and dancing.
- Human-computer interaction: Enabling natural and intuitive interaction between humans and computers, such as gesture recognition and virtual try-on.
- Sports analysis: Tracking the movement and posture of athletes during games or training sessions, enabling coaches and athletes to analyze and optimize their performance.
- Gaming: Enabling immersive and interactive gaming experiences, such as motion-controlled games and virtual reality games.
- Surveillance: Detecting and tracking human activity in surveillance videos, enabling security personnel to monitor and respond to potential threats.
- Medical diagnosis: Detecting and monitoring conditions related to posture and movement, such as scoliosis and Parkinson

HPE can also have applications in the field of intelligent vehicles and autonomous driving. For instance, HPE can be used to detect and track the poses of pedestrians, cyclists, and other objects on the road, enabling autonomous vehicles to better understand and respond to their surroundings. Inside the driver's cabin, HPE can be used to monitor the posture and movement of the driver, enabling intelligent vehicles to detect and respond to fatigue or other impairments.

2.2 LiDAR

The LiDAR sensor, also known as Light Detection and Ranging sensor, is a device that utilizes lasers to detect and locate objects in space. Unlike radar sensors use electromagnetic waves, LiDAR sensors use optical measuring methods. The sensor comprises a transmitter, receiver, and evaluation unit with a high temporal resolution, which is responsible for emitting focused light pulses, detecting reflected light beams, and measuring the distance, speed, and orientation of objects.

LiDAR sensors use a method called time-of-flight (TOF) to measure distance, further elaborated in Section 2.2.2. In this method, the emitted light pulses bounce off surrounding objects and return to the receiver unit. The time taken for the pulse to travel to the object and back is measured and used to determine the distance between the object and the sensor. Overall, LiDAR sensors provide highly accurate and detailed information about the surrounding environment, making them useful in a wide range of applications such as self-driving cars, robotics, and mapping.

2.2.1 Overview of techniques

LiDAR sensors employ a modulated laser beam as a carrier to measure distance, but only within its instantaneous field of view (FOV). In order to generate a 3D point cloud with cartesian (x,y,z)-coordinate information, the laser must be pointed in all directions of the desired FOV.

LiDAR devices can be categorized based on the method used to scan the laser beam, listed in Figure 2.2. Scanning LiDAR devices scan the laser beam successively, while Flash LiDAR devices acquire distance information simultaneously.

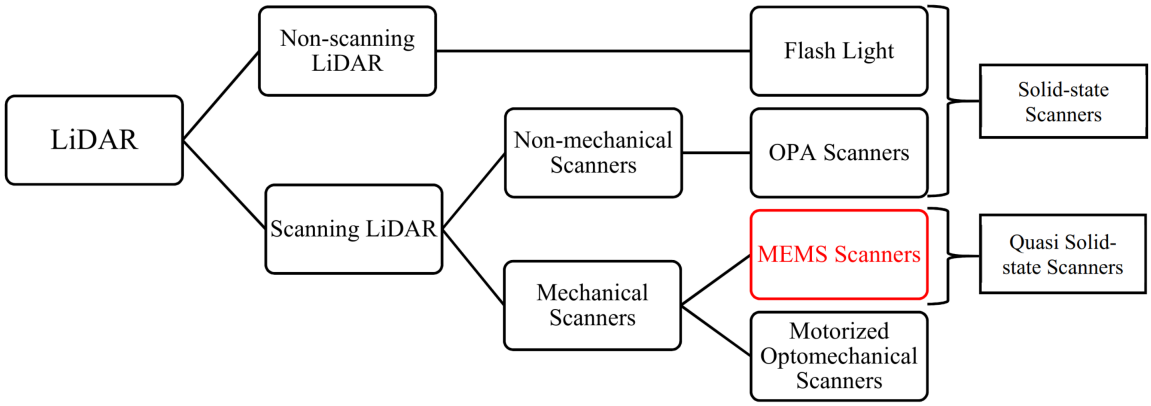


Figure 2.2: LiDAR scanning types [60]

Flash LiDAR

Flash LiDAR sensors use a flood illumination approach that scatters the laser beam by the optics (diffuser and lens) in such a way that it illuminates the entire scene (limited by the FOV) at once, similar to a camera with a flashlight (see Fig. 2.3a) [23]. This approach works faster than scanning systems, is inexpensive, easy to manufacture with different aperture [18] angles, and belongs to the solid-state system category. Using only solid-state components eliminates the need for moving parts and makes them more compact, resistant to vibration, and lower in cost.

In contrast to scanning LiDAR systems, the laser is spread out by the diffuser and not focused on a single point, i.e., the output energy is distributed over an area and is limited in signal-to-noise ratio. This can significantly limit the measurement range or require high laser power. Additionally, the resolution of the detector array-based Flash LiDAR is limited by the size and density of the detector array. [55] Therefore, Flash LiDAR systems are primarily used as short-range LiDAR.

OPA scanner

Optical Phased Array (OPA) scanners are solid-state and deflect the laser beam by optical phase control, making them a more popular and mature technology compared to Flash LiDAR. An integrated OPA device (see Fig. 2.3b) typically consists of an array of closely spaced emitters that can be individually addressed to control the optical phase. By controlling the magnitude of the phase difference, the position of the maxima can be adjusted, enabling beam steering in one dimension (azimuth angle).

In an 1D-OPA, beam deflection in the direction of the other dimension (elevation angle) is achieved by controlling the wavelength. In contrast, the emitters in a 2D-OPA can be operated at the same wavelength but require phase control over a much larger number of emitters. [14] OPA receivers can electronically shape, steer, and selectively collect a received beam from any desired direction, allowing them to suppress unwanted light from other directions and improve sensitivity.

OPAs also have the advantage of requiring low drive voltage and being simple to implement. However, efficiency tradeoffs exist between wide-angle control and high resolution in horizontal phase control, as efficiency is typically lower at wider angles. [60]

Motorized optomechanical ccanner

Motorized optomechanical scanners, also known as spinning LiDAR, use a rotating mechanism to spin the scanner, equipped with multiple lasers and photodetectors stacked vertically (also known as channels), which produces a full 360-degree scan. While this type of scanner can be developed with a long-range, wide field of view and high scanning speed, they are not energy efficient and are susceptible to mechanical shock and wear. In addition, their vertical resolution is fixed and dependent on the number of transmitter and receiver channels, making high vertical resolution costly.

For applications where a concentrated horizontal field of view is required, direct use of spinning LiDAR can be inefficient, as a large portion of the laser beams must be diverted away from the target scene and discarded. However, the most obvious advantage is the 360-degree FOV. Particularly, on vehicular operation sites, a single LiDAR unit can be equipped on top of a car and a complete view of the car's surroundings is obtained. Solid-state LiDARs, in contrast, are fixed in place and typically have a field of view of 120° or less. Achieving comparable coverage with a solid-state sensor takes at least four units.

Another advantage of rotating LiDARs is that they can emit laser pulses at a higher power than stationary sources. With a rotating sensor, the laser is focused in a particular direction for only a fraction of its 360-degree rotation, making it safer for the human eye, when compared to a scanning solid-state device, where 100% of the laser light could enter the eye. Additionally, they generally have a larger aperture, which means more emitted energy and range can minimize the effects (e.g. dead-bug problem [37]) of opaque materials on the image sensor, giving them an advantage in environments with contaminants such as dust, rain, mud, snow, and others.

MEMS Scanner

MEMS (Micro-Electro-Mechanical System) mirrors are microscale devices that can direct, modulate, switch, and control the phase of light. MEMS LiDARs have already found commercial success in projectors, displays, and fiber optic communications applications.

In a MEMS mirror-based LiDAR system, only one mirror plate with a diameter in the range of 1-7mm is moved, while the rest of the components of the system are stationary. Therefore, MEMS-LiDAR is often referred to as quasi-static LiDAR and is a compromise between solid-state LiDAR and spinning LiDAR.
MEMS scanners have several advantages over other LiDAR technologies, including their

small size, low cost, and fast scanning speed. They can also produce high-resolution 3D images with accurate distance information. However, MEMS scanners also have limitations, such as limited scanning range and difficulty in controlling the mirror's angle precisely. [60]

LiDAR types outlook

Solid-state LiDAR sensors have package space, cost, and robustness advantages (less susceptible to mechanical wear and have a longer lifespan). They eliminate moving components for beam alignment and can be based on phased arrays or MEMS mirrors. In addition, they can provide high-resolution 3D scans with accurate distance information, making them suitable for a wide range of applications such as robotics, autonomous navigation, and environmental monitoring.

Despite these advantages, spinning LiDAR remains superior for many applications that require a high emphasis on the range while maintaining high resolution and the largest possible field of view. Their unique selling point is their ability to provide up to 360-degree visibility with a single LiDAR unit. Therefore, the choice of LiDAR technology depends on the specific application requirements, and both spinning LiDAR and solid-state LiDAR sensors have their place in the market.

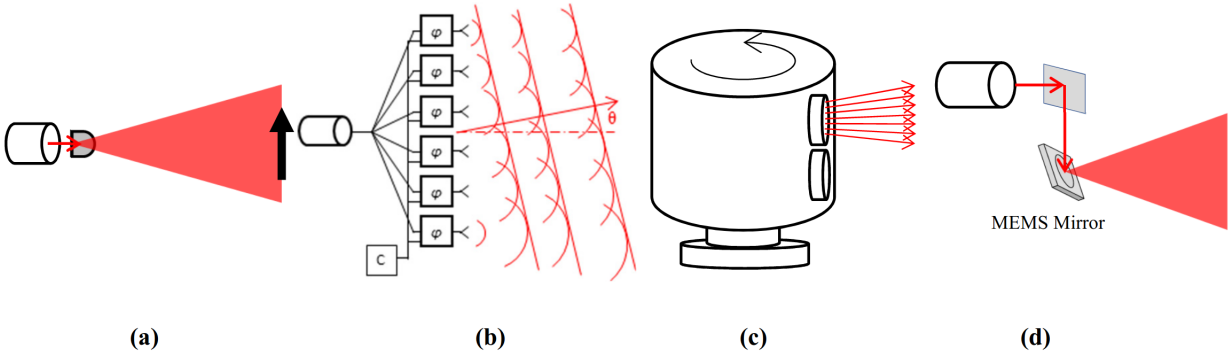


Figure 2.3: (a) Flash LiDAR, (b) Optical Phased Array (OPA) scanner, (c) Spinning LiDAR, (d) MEMS scanner

2.2.2 Runtime measurement

The LiDAR sensor measures the transit time of a light pulse to determine the distance to a reflecting object. This measurement principle is based on the time-of-flight (TOF) of the light pulse, which is the time it takes for the pulse to travel from the LiDAR sensor to the object and back to the sensor.

The distance between the LiDAR sensor and the reflecting object is calculated using the transit time and the speed of light in vacuum, which is approximately $c_0 = 299.792.458 \frac{m}{s}$:

$$d = \frac{c_0}{n_{Br}} \cdot \frac{ToF}{2} \tag{2.1}$$

Here, the refractive index n_{Br} of the medium is taken into account, which describes the relationship between the speed of light in a vacuum and the speed of light propagation in the medium. The refractive index generally depends on the wavelength of the light and is approximately equal to 1 in air.

The distance determination also takes into account that the light pulse must travel the path from the LiDAR sensor to the object and back to the sensor, hence the factor of $1/2$ in the equation. This calculation is used in all LiDAR sensors to determine the distance to objects in their field of view.

2.2.3 Remission

The received signal from a LiDAR sensor provides information about the remission of the detected object, which refers to the portion of the laser pulse that is reflected back to the sensor. Each material has a specific remission that depends on its surface properties such as color and structure. When a laser pulse hits a surface, the energy is partially absorbed by the material, and the remaining energy is reflected back to the sensor.

Most surfaces reflect the laser beam diffusely in all directions [50]. The maximum detection range of a LiDAR sensor depends largely on the remission of the object. The greater the remission, the greater the possible detection range [49, 20]. Light surfaces, for instance, reflect the laser beam better than dark surfaces, and they can be detected by the LiDAR sensor over longer distances [50].

2.2.4 Beam and reflection angle

The angle of reflection of a laser beam corresponds to the angle of incidence, and optimal reflection occurs when the laser beam hits a surface perpendicular. If it strikes at an angle, there is a corresponding loss of energy and range. Reflective surfaces can also be problematic because an almost complete deflection of the laser beam can result in the detection of another object hit by the deflected laser beam instead of the reflective surface.

Laser beams diverge, which means that less light power is irradiated onto the object per unit area as the distance increases. The same propagation conditions apply to reflected light. An object is fully hit by the laser if it is at least as large as the diameter of the laser beam. Objects smaller than the diameter of the laser beam cannot reflect all the energy of the laser light, resulting in a loss of energy and less range than theoretically possible. [63]

As the distance increases, the laser beam widens, increasing the diameter of the measurement spot on the surface of the object, which can result in erroneous measurements. Unfavorable outdoor or weather conditions such as hoarfrost, dust, moisture, snow, or rain can also lead to reflections and consequently result in registered measuring points on the sensor, leading to unwanted object detections.

2.2.5 LiDAR coordinate system

The sensor expresses each individual measurement point in sensor coordinates (radius r, elevation ω, azimuth α). In relation to a cartesian coordinate system, the elevation angle ω is measured in the zy-plane from the y-axis, and the azimuth angle α is measured in the xy-plane from the y-axis. The azimuth angle depends on the position at which the laser is fired and is registered at the time of firing, while the elevation angle for a laser beam is specified by the sensor.

The cartesian coordinates (x, y, z) can then be calculated using the following formulas:

$$\begin{aligned} x &= r \cdot \cos\omega \cdot \sin\alpha \\ y &= r \cdot \cos\omega \cdot \cos\alpha \\ z &= r \cdot \sin\omega \end{aligned} \tag{2.2}$$

The cartesian coordinate system is easy to manipulate and is used in many applications, e.g., for rendering 3D point clouds. To project points in cartesian coordinates back to spherical coordinates, the equations below are used:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \arcsin\frac{z}{r} = \arctan\frac{z}{\sqrt{x^2 + y^2}} \\ \phi &= \arctan\frac{y}{x} \end{aligned} \tag{2.3}$$

The relationship between the two coordinate systems is depicted in Figure 2.4, where a point in three-dimensional space using one of a spinning LiDAR's laser beams is illustrated.

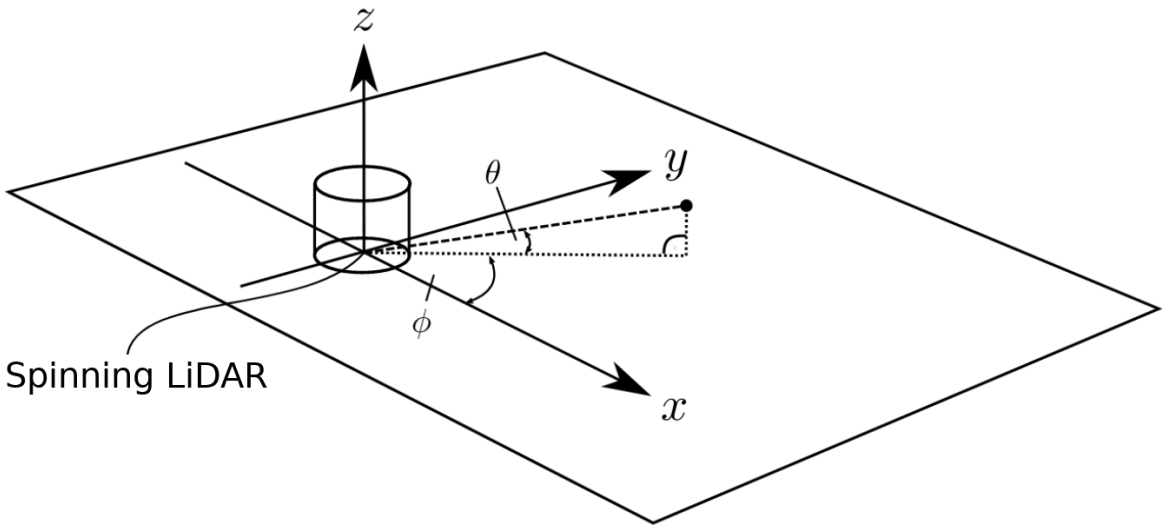


Figure 2.4: Coordinate system of a spinning LiDAR. ϕ as the azimuth angle and θ as an inclination angle from the xy-plane [45]

Additional LiDAR characteristics will be covered in Section 3.2.2 and 3.3, including the impact of sensor resolution on point cloud density and the challenges posed by the occlusion effect.

3 Chapter 3

Dataset description and preparation

This chapter details the foundation and preparation of the dataset utilized in this research. Here, the simulation environment aims to create realistic driving scenarios and simultaneously implement LiDAR sensor measurements. Subsequently, the input representation of the 2D pose estimator is presented.

3.1 An overview of the CARLA simulator and data acquisition

Simulation environments offer unparalleled precision in controlling and manipulating every aspect of a scene. This controlled setting sidesteps the unpredictable variables inherent to real-world testing, such as adverse weather conditions, changes in lighting, and system challenges like sensor noise and faulty measurements. This aspect of simulation is beneficial when dealing with LiDAR sensor data, which is susceptible to influence from external environmental conditions.

For this proof-of-concept research, the CARLA simulator [7, 12] has been chosen, an open-source modular framework used for research and by the automotive industry in the field of autonomous driving. It is built on the robust Unreal Engine 4 (UE4) [15] and is released under the MIT license, ensuring its accessibility and adaptability for various research purposes.

Compared to using open-source autonomous driving datasets, which include LiDAR measurements, selecting the CARLA simulator for creating driving scenarios has several advantages:

1. **Realism and Detail**: CARLA's environments are designed to mimic real-world urban landscapes closely, providing detailed simulations that include realistic road networks, traffic patterns, and pedestrian behaviors.
2. **Comprehensive Sensor Suite**: It includes simulations of a wide array of sensors, including RGB, Depth, Semantic Segmentation cameras, and LiDAR.

3. **Customization and Flexibility**: Users can create customized driving scenarios, including varying weather conditions, time of day, and traffic densities.
4. **Scalability and Efficiency**: Simulation allows for the rapid testing of autonomous driving algorithms across a vast number of scenarios, including rare or dangerous situations that would be difficult or unethical to replicate in real life.
5. **Open-Source Community**: CARLA benefits from contributions by a global community of researchers and developers. This collaborative environment fosters innovation, with continuous updates and improvements to the simulator,

3.1.1 The working mechanism of CARLA

CARLA operates on a client-server system. The server side is responsible for maintaining the state of every actor and the world's state, including physics computation and graphics rendering. On the other hand, the client side comprises one or multiple clients that connect to the server, requesting data and sending commands to control the logic of actors in the scene and set world conditions.

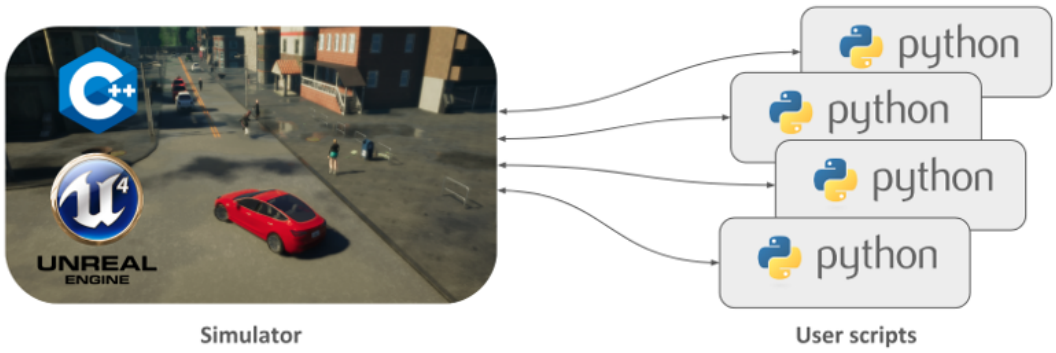


Figure 3.1: Basic structure of the simulator [8]

3.1.2 Data acquisition using the CARLA simulator

The data acquisition process in this work commenced with the generation of driving scenarios using CARLA's Python API. Utilizing the diverse maps available in CARLA, 801 unique scenes were created for this study. These scenes were subsequently divided into a training and testing set following an 80/20 split.

Autonomous agents, called "walkers" and "vehicles", were spawned within each scene. These agents are outfitted with controller logics (i.e., out-of-the-box autopilot feature found in CARLA), enabling them to navigate through the environment autonomously.

A (semantic) LiDAR sensor was attached to the rooftop of one of the vehicles in each scene. This sensor simulates a spinning LiDAR using ray-casting [16] for each laser in

every step that exposes all the information about the ray-cast hit. The raw data retrieved by the semantic LiDAR includes [51]:

- Coordinates of the point (as the normal LiDAR does).
- The cosine between the angle of incidence and the normal of the surface hit
- Instance and semantic ground-truth. Basically, the index of the CARLA object hit (typeID) and its semantic tag.

The points are computed by adding a laser for each channel distributed in the vertical FOV. The rotation is simulated by computing the horizontal angle that the LiDAR rotated in a frame. After a full rotation specified by the horizontal FOV, the raw data can be represented as a point cloud (see Figure 3.3).
The sensor is configured with the following attributes: vertical FOV=180°, horizontal FOV=360°, channels=1024, frequency=10Hz.

The interval of saving the point cloud in each scene is set to 0.2Hz (i.e., every 5 seconds) to minimize redundancy. Given the high computational demands of ray-casting, the actual runtime exceeds the simulated timeline, preventing real-time operation. Consequently, this simulation is employed solely to generate an annotated automotive LiDAR dataset rather than integrating the HPE model directly with the simulation.
Though these configurations seem too high-resolution and not to be found in current state-of-the-art LiDAR, the vertical FOV divided by the number of channels yields the vertical angular resolution between each channel or ray-cast, which is approximately 0.18° and draws inspiration from the Ouster OS2 LiDAR system [36]. Further details on why this large vertical FOV is chosen are explained in Chapter 4.1.4.

The coordinates of the points are adapted from Unreal Engine, characterized by a left-handed coordinate system, where the x-axis points forward, the y-axis points to the right, and the z-axis extends upward, as illustrated in Figure 3.2. To facilitate seamless integration with prevalent mathematical and computational frameworks in scientific analyses, the y-coordinates were inverted to represent a right-handed cartesian coordinate system.

Figure 3.2: Left: Front view of the vehicle with sensor coordinate system (UE)
Middle: Rear view
Right: Point cloud

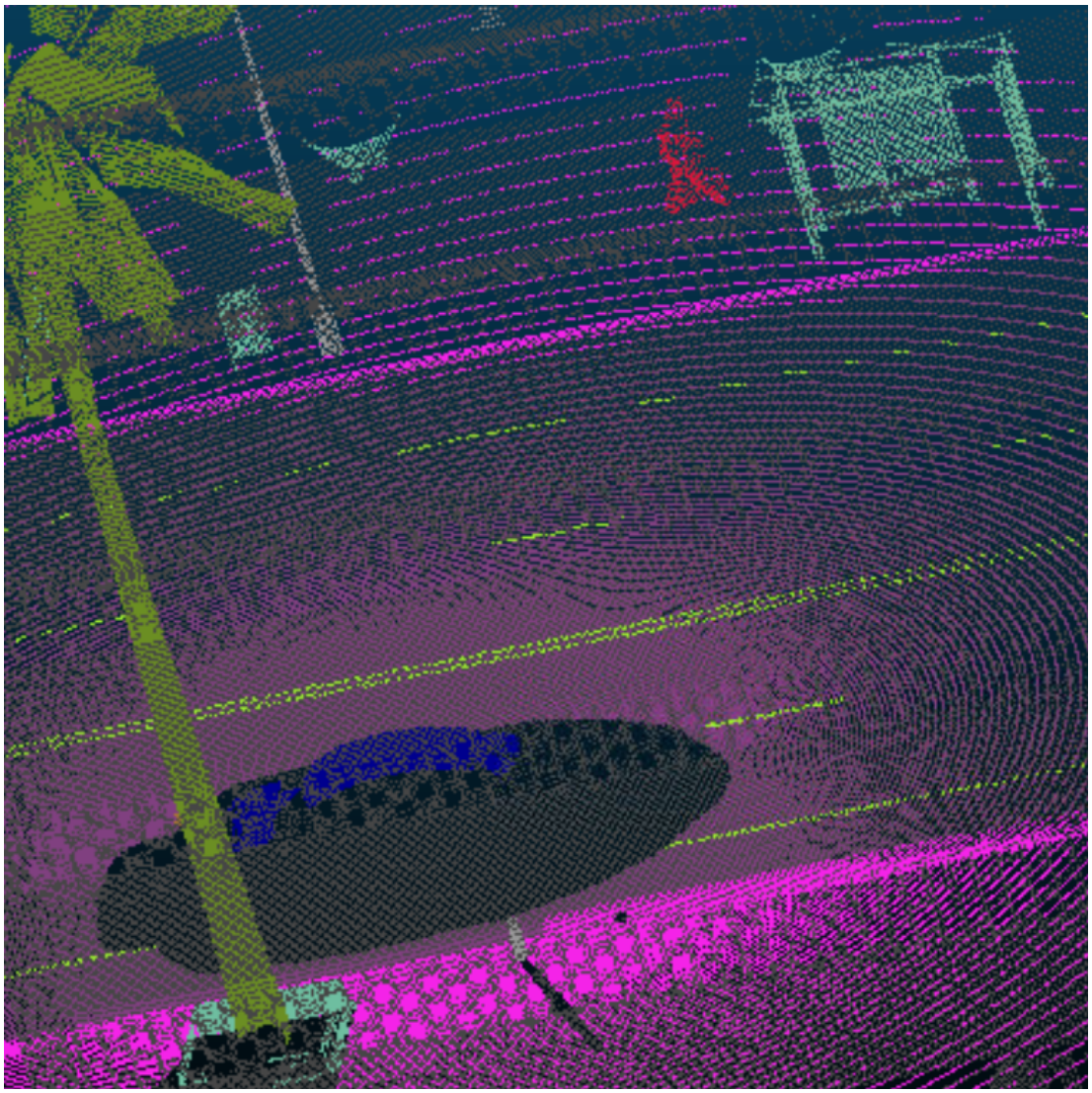
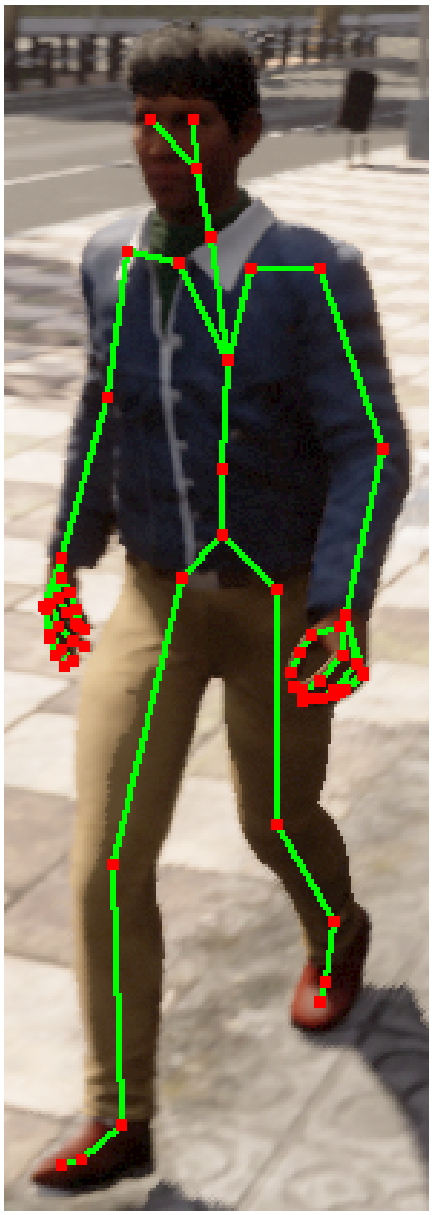


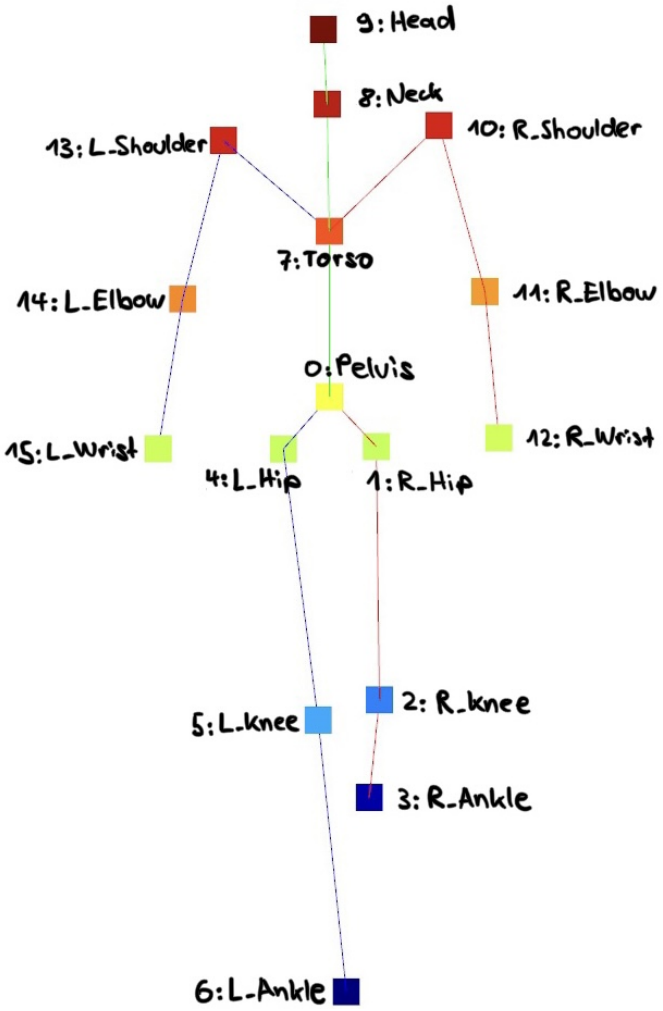
Figure 3.3: Semantically colored point cloud cutout

In addition to the raw data, additional metadata were derived from both static and dynamic actor properties obtained from the world state in CARLA to provide a more detailed understanding of each scene. This includes the motion state, velocity, acceleration, 3D bounding box extent, location, rotation, semantic tag, and typeID.

The typeID, which is present in both raw data and metadata, maintains consistency for the same actors, ensuring that these identifiers are matched and coherent across both data representations.
The additional parameter number of wheels was extracted for vehicles to provide more specific information about the vehicle type.
For walkers, a detailed skeleton structure consisting of 66 joints was obtained. This structure (see Fig. 3.4a), unique to CARLA, provides the 3D location of each joint, offering a comprehensive representation of the walker's body. Here, the joint locations of walkers are the only relevant information in the metadata.



(a) CARLA skeleton with 66 joints



(b) Custom skeleton with 16 joints

Figure 3.4: CARLA joints

3.1.3 Custom human skeleton structure

For human joint detection in driving scenarios, the 66-joint representation appears overly detailed. Given the sparse nature of LiDAR measurements, there is no need for that much detail. Simplifying the skeleton by focusing on the most relevant joints is more pragmatic. As it stands, CARLA natively supports only its distinct skeleton representation, making it incompatible with direct transformation to popular skeleton representations like Human3.6M [21] or SMPL [28]. This work utilizes selected CARLA joints to construct a custom skeleton structure to avoid the complexities and potential errors associated with converting to another joint representation. This representation is

adapted from the Human3.6M skeleton structure [21] (see Fig. 3.5), extracting a total of 16 joints from the CARLA joints, as illustrated in Figure 3.4b.

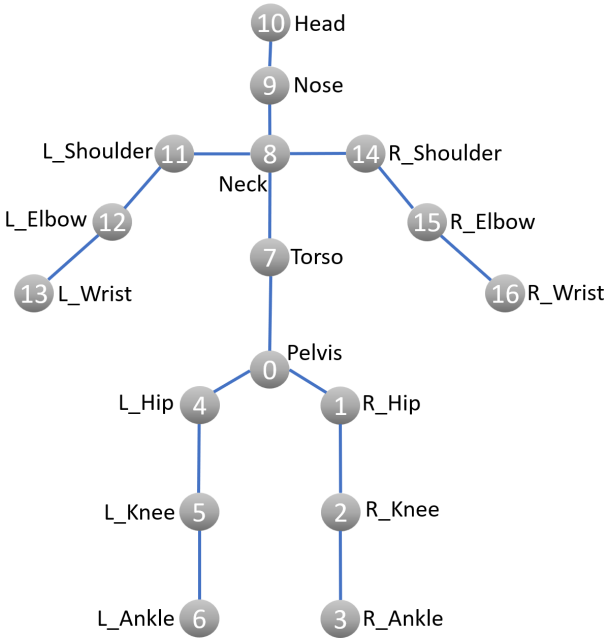


Figure 3.5: Human3.6M joint indices

3.2 Spherical Projection

Point clouds are fundamentally irregular, are not sensitive to the order of the data, and thus lack structure. This means that the model for processing point cloud data needs to be invariant to different permutations of the data. PointNet models (e.g., PointNet [42] and PointNet++ [43]) treat each point individually, learning the mapping from 3D to potential features without taking advantage of geometry. As briefly mentioned in the introductory chapter 1, the time and space complexities of the 3D CNN grow cubically with the resolution of the input 3D volume, which is critical for desired real-time inference time.

Instead of downsampling the point cloud and consequently losing relevant information, a widely adopted method to structure LiDAR data is through spherical projection [57], which is designed to map spherical coordinates (ϕ, θ) to image coordinates (u, v), forming what is known as a spherical projection image. Particularly, equirectangular projection is used here, a specific type of spherical projection, which ensures uniform angular resolution across both horizontal and vertical directions of the image. This uniformity means that each pixel in the resulting image corresponds to the same degree of coverage in both the longitude and latitude directions. This characteristic is particularly beneficial when working with LiDAR data, as it allows for preserving spatial relationships and structural integrity that might be lost through downsampling or other forms of data compression.
It is important to note that while angular resolution remains constant, point density

can vary across the image. This variation is most pronounced near the poles, where the projection inherently stretches, leading to potential distortion.

3.2.1 Mathematical Formulation

Continuing with the mathematical context, the point cloud's cartesian coordinates (x,y,z) get firstly transformed into spherical coordinates using Equation 2.3. The matrix multiplication of transformation matrix K (size: 3×3) and matrix X, containing the spherical coordinates of N points being projected (size: $3 \times N$), defines the non-discretized pixel coordinate matrix U (size: $3 \times N$).

This is mathematically expressed in Equation 3.1 [45, 19].

$$\underbrace{\begin{bmatrix} \vec{u} \\ \vec{v} \\ \vec{1_N} \end{bmatrix}}_{U} = \underbrace{\begin{bmatrix} \frac{1}{\Delta\phi} & 0 & c_\phi \\ 0 & -\frac{1}{\Delta\theta} & c_\theta \\ 0 & 0 & 1 \end{bmatrix}}_{K} \cdot \underbrace{\begin{bmatrix} \vec{\phi} \\ \vec{\theta} \\ \vec{1_N} \end{bmatrix}}_{X} \tag{3.1}$$

The matrix elements of K consist of scaling factors $(\frac{1}{\Delta\phi}, \frac{1}{\Delta\theta})$ and shifts (c_ϕ, c_θ) that synchronize the range and origin of the spherical coordinates with the image coordinates. The exact meaning and definition of these constants are described in more detail below.

$$\Delta\phi = \frac{\text{F\_H}}{\text{I\_W}}\,, \tag{3.2}$$

where F_H is the horizontal field of view and I_W the image width. This implies that for each pixel increment in the image coordinate u, the spherical coordinate ϕ increases by $\Delta\phi$ degrees.

$$\Delta\theta = \frac{\text{F\_V}}{\text{I\_H}}\,, \tag{3.3}$$

where F_V is the vertical field of view and I_H the image height. $\Delta\theta$ represents the degree change in the spherical coordinate θ per pixel increment in the vertical direction v of the image.

This study's transformation parameters were meticulously selected to emulate the equirectangular image format and align with the vertical angular resolution of the Ouster OS2 LiDAR. However, these parameters can be adjusted to modulate the projection's granularity, allowing for a more detailed or broader representation.

3.2.2 Sensor resolution impact of $\Delta\theta$ and $\Delta\phi$

$\Delta\theta$ and $\Delta\phi$ play crucial roles in determining several characteristics of the output image. Firstly, the range of points' spherical coordinates is constrained by $-\frac{\text{F\_H}}{2}$ to $+\frac{\text{F\_H}}{2}$ for the ϕ coordinates, and $-\frac{\text{F\_V}}{2}$ to $+\frac{\text{F\_V}}{2}$ for θ. Secondly, the denominators I_W and I_H

set the final image dimensions. Finally, the whole expressions affect how the 3D points are projected onto 2D pixel coordinates - data points falling outside these parameters can be selectively excluded - and define the image's granularity, including the clarity and detail of the human pose representation.

Figure 3.6 [45] illustrates the influence of varying configuration settings and point cloud densities on a uniform human pose representation. The triplet of point clouds in the top left exemplifies the sensor measurement resolution employed in this study (from Eq. 3.3: $\Delta\theta = \frac{180}{1024} = \frac{22.5}{128}$). With the increasing distance of objects, the point cloud quickly becomes sparse, making it challenging to accomplish HPE successfully due to the diminished details available about the target.

Figure 3.6: Left to right: Vertical Layers [128, 64, 32]
Top to bottom: Vertical FOV [22.5°, 45°, 90°]
Colors (Range): Orange (10m), Green (20m), Blue (40m);
Resolution used in this work: Row 1, Column 1

3.2.3 Formulation of the spherical projection image

As an equirectangular image is targeted, $\Delta\theta$ and $\Delta\phi$ must be equal. A straightforward approach is to set the number of channels (i.e., 1024) as the image's height. As the horizontal FOV is configured twice the vertical FOV, the image width is set to 2048, preserving the uniform angular resolution of approx. 0.18°.

The inclusion of the negative sign $-\frac{1}{\Delta\theta}$ in K reverses the increment direction of the θ coordinate, aligning it with the descending v-coordinate in the image space (see Fig. 3.7). This adjustment ensures that as θ increases from the xy-plane, the v-coordinate increases

from the top of the image, maintaining a coherent mapping between the spherical and image coordinate systems.
The shifts c_ϕ and c_θ are determined to align the origin of the spherical coordinates ($\phi = 0, \theta = 0$) with the center of the image ($c_\phi = \frac{\text{I\_W}}{2}, c_\theta = \frac{\text{I\_H}}{2}$). The different orientations and origins of the image coordinate system and the spherical coordinate system are illustrated in Figure 3.7.

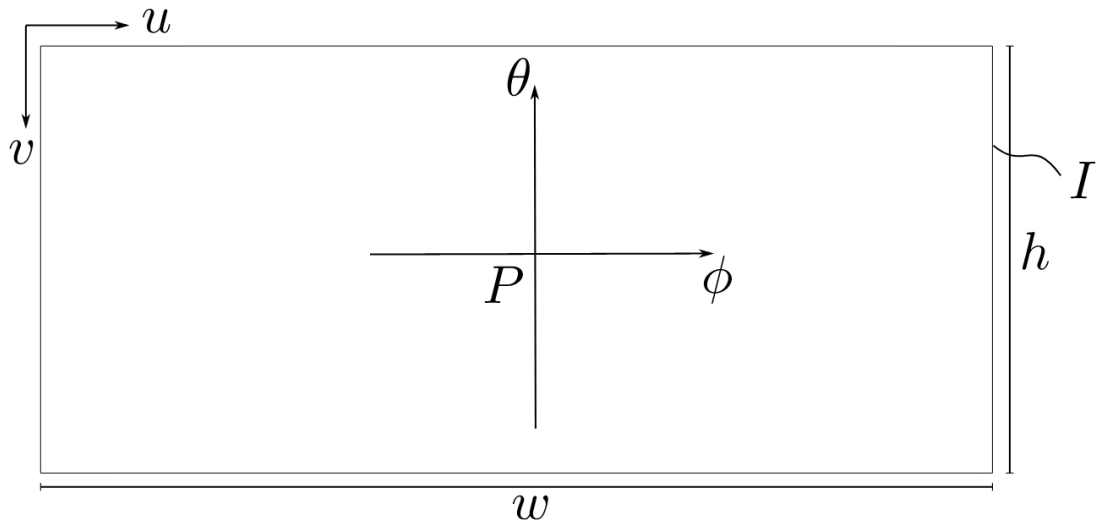


Figure 3.7: Spherical image I with principal point P, height h, width w [45]

After this alignment, the pixel coordinates undergo a discretization process. This step transforms the coordinates from floating-point values into discrete pixel values, aligning them with the pixel grid. This can result in multiple points being projected onto a single pixel. A selection criterion addresses this, prioritizing points with the shortest Euclidean distance r to the sensor and thereby incorporating the occlusion effect of LiDAR measurement (see Sec. 3.3 for more detail on the occlusion characteristics of LiDAR). It's essential to note that any shifts or rotations in the point cloud can alter the arrangement of points closest to the origin within the grid, impacting the final configuration.
At this stage, each (u,v)-coordinate maps to a specific point in the original point cloud and its corresponding raw data. The specific input image channel encoding used for the 2D keypoint estimator is covered in Chapter 4.1.

Generation of ground-truth labels

The filtered raw data is now used to generate the ground-truth labels: bounding box, instance segmentation mask, and class label. The 3D cartesian keypoint coordinates are obtained from the metadata by typeID matching. The projection of these, using Equation 2.3 and 3.1, yield the 2D pixel coordinates.

3.3 Occlusion in LiDAR data

In addition to the impact of sensor resolution settings on LiDAR-based HPE, as discussed in Section 3.2.2, occlusion effects play a significant role in the quality of LiDAR data.

These originate from the intrinsic characteristics of LiDAR technology, where detection points are generated exclusively from the surfaces. Consequently, points beyond the initial contact surface are obscured in line with the sensor's directional path. Occlusions occur through self-occlusion - where an object's parts obscure its other parts - or through mutual occlusion, involving the obstruction of objects or individuals by one another, a common scenario in crowded spaces or during interactions.
This phenomenon frequently results in the capture of incomplete or distorted human figures, complicating the interpretation of human shapes.

A notable complication arises with human joints. Unlike surface detection points, human joints require a nuanced depth estimation that penetrates beyond visible layers into the body's interior.

Illustrated by Figure 3.8, the issue of half a body being obscured showcases the inherent challenges in accurately estimating posture and joint positions. The spatial relationship between the sensor and the subject heavily influences the occlusion effect, making it prone to errors without additional external interventions, e.g., additional sensor perspective or tracking techniques over multiple frames.

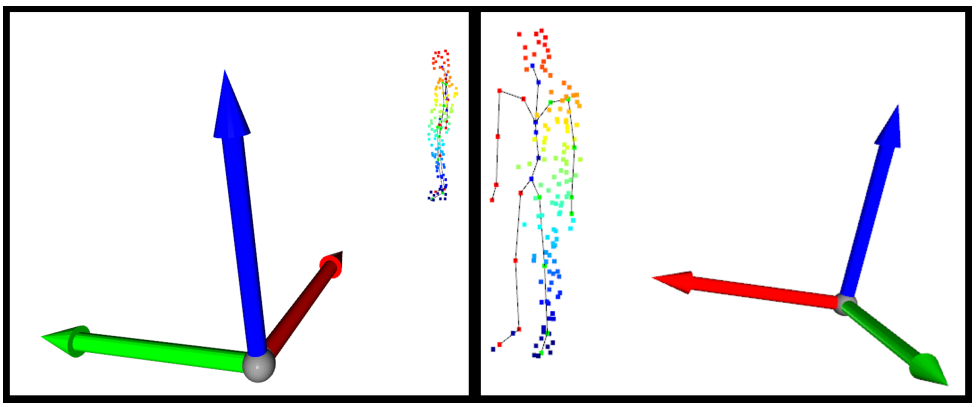


Figure 3.8: Self-occlusion showcase; red coordinate axis indicates the sensor's viewing direction

Chapter 4

4 Methodological approach

This work introduces a novel absolute multi-step HPE approach that leverages LiDAR data, diverging from traditional camera-based inputs such as RGB, Depth, and omni-directional cameras, which are detailed in Section 1.1.1. The initial phase employs a straightforward 2D pose estimator to determine 2D joint locations. However, rather than normalizing the predicted pixel coordinates based on the image's dimensions, this approach utilizes the projection matrix described in Section 3.2.1 and transforms 2D pixel coordinates into 3D space, employing the direction vector toward each point as the input for the lifting network. The foundation of this lifting network is an adaptation of [9]. The nuanced methodology will be detailed in the subsequent sections of this chapter, including the enrichment of the spherical projection image from Section 3.2.3 and the implementation details of the two stages of this multi-step HPE approach.

4.1 Input image construction

The spherical projection image's grid positions point or map to specific points in the point cloud. Here, the most relevant information from its connected raw and metadata is the physical measurement of 3D point locations, from which mathematical relationships can be derived, e.g., direct point-to-sensor distances, height, and angle correlations between points or from point-to-sensor. These can be similarly found in the HHA encoding scheme [17], used for alternatively representing depth maps.

In the original HHA encoding:

- **H** represents **H**orizontal disparity, encoding the disparity observed in the horizontal direction.
- **H** denotes **H**eight above ground, capturing the relative height of each pixel from the ground.
- **A** signifies the **A**ngle with gravity, which encodes the angle between the surface normal and the gravitational direction.

4.1.1 Channel 1: Euclidean distance

The HHA encoding originates from stereo vision, where time-of-flight calculation is not available to obtain depth directly. As disparity is intrinsically tied to pixel differences in stereo images, it is fundamentally dependent on the specific camera setup, and even nuanced alterations in camera configurations can induce pronounced disparities.
To avoid this dependency, [53] utilizes Z-depth. Z-depth, by definition, measures the distance of objects along a single axis from the sensor - usually the axis perpendicular to the camera's imaging plane in the direction it's facing (i.e., typically the z-axis, but in this setup visualized in Fig. 3.2, the reference axis would be the straightforward pointing x-axis). While it still relies on the disparity for its computation $Z = \frac{f \cdot B}{d}$ - where f is the focal length of the camera and B is the baseline or the distance between the two camera centers - once converted to Z-depth, it is advantageous due to simplicity, universality (allows integration of various input modalities), and direct applicability.

However, it is crucial to distinguish between the depth measurement approaches, especially when considering LiDAR systems. Unlike stereo vision, LiDAR systems directly measure depth through time-of-flight, as outlined in Chapter 2.2.2. With cartesian coordinates encoded in the raw data, the distance d is more accurately represented by the Euclidean distance, as shown in Equation 2.1 and 2.3. This method provides a comprehensive 3D spatial measurement, offering a significant advantage over Z-depth due to its robustness against changes in observation angles.

Figure 4.1 illustrates the difference between these two methods of distance measurement in 2D. In automotive contexts, where the ego-vehicle undergoes constant translation and rotation, stereo vision's Z-depth readings fluctuate for the same static object due to orientation changes. As depicted in Figure 4.1, rotating the coordinate axes affects Z-depth values, whereas Euclidean distances remain consistent, underscoring the superiority of Euclidean distance for maintaining stable depth perceptions in dynamic environments.

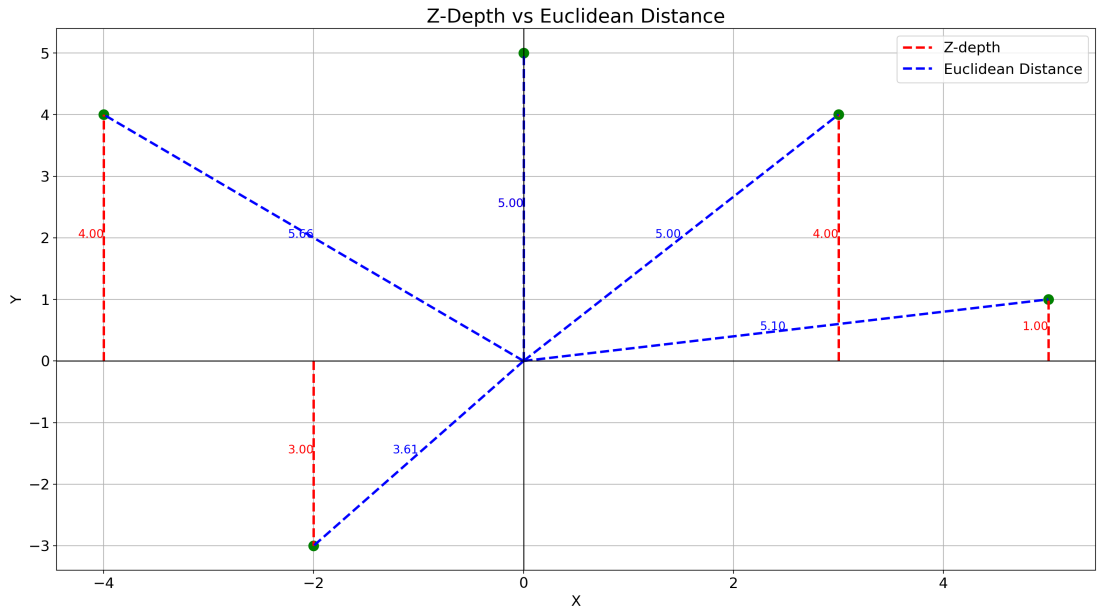


Figure 4.1: Comparison of Z-depth (in red) and Euclidean distance (in blue) in 2D alongside representative data points (in green) and their respective distance measurements

Considering this, the Euclidean distance has been selected as the primary channel for the custom HHA encoding.

4.1.2 Channel 2: Height above ground

With the known mounting position of the sensor, the operational height of the sensor to the ground plane is fairly consistent throughout the recordings. The z-value of the points, subtracted from the sensor height, is defined as pixel values for the second channel. This contextual information about the respective heights above the ground could be useful, e.g., to easily distinguish between a tall vehicle and an overhead signboard or at which height levels people are mostly located.

4.1.3 Channel 3: Direction cosine of the local surface normals

The last channel represents the angle between the surface normal (at each spatial location) and the gravitational direction. This angle provides crucial information about the orientation of surfaces in the scene.

Previous works [4, 46] try to estimate the normal vector by performing Principal Component Analysis (PCA). The PCA minimization problem can be seen as finding the eigenvector associated with the smallest eigenvalue of the covariance matrix C, which is equivalent to solving the following optimization problem:

$$\min_{\vec{n}} \quad \vec{n}^T C \vec{n} = \min_{\vec{n}} \quad \vec{n}^T \left(\frac{1}{k} \sum_{i=1}^{n} (p_i - \bar{p}) \cdot (p_i - \bar{p})^T \right) \vec{n} \tag{4.1}$$

subject to $\|\vec{v}\| = 1$ (ensures that $\vec{v}$ is a unit vector), where $p_i - \bar{p}$ represents the centered data points and $\bar{p}$ is the mean of the points in the local k-number neighborhood of the point of interest.

Instead of using an iterative optimization approach, direct mathematical computation in ordered point clouds is introduced by [47]. Before delving into the implementation details, the mathematical context of a local surface normal needs to be established.

For better understanding, the following mathematical notations are shown in Figure 4.2.
Let z=f(x,y) be a two-variable real-valued function that generates the surface S. For a fixed value of $y = y_0$, the partial derivative $\frac{\partial f(x, y_0)}{\partial x}$ yields a curve C_1, which intersects with the plane $y = y_0$. For any defined value $x = x_0$, the partial derivative $\frac{\partial f(x = x_0, y = y_0)}{\partial x}$ returns the slope of the tangent line T_1 of C_1. Similarly, let C_2 be the curve obtained by intersecting the plane $x = x_0$ with S, then the tangent line T_2 can be obtained by $\frac{\partial f(x = x_0, y = y_0)}{\partial y}$. The tangent plane on S to the point $P_C(x_0, y_0, f(x_0, y_0))$ is the plane that contains both T_1 and T_2 and can be mathematically formulated as:

$$z - z_0 = \frac{\partial f(x = x_0, y = y_0)}{\partial x}(x - x_0) + \frac{\partial f(x = x_0, y = y_0)}{\partial y}(y - y_0) \tag{4.2}$$

Finally, the cross-product $T_1 \times T_2$ yields the normal $\vec{n}$ to the surface S at (x_0, y_0, z_0).

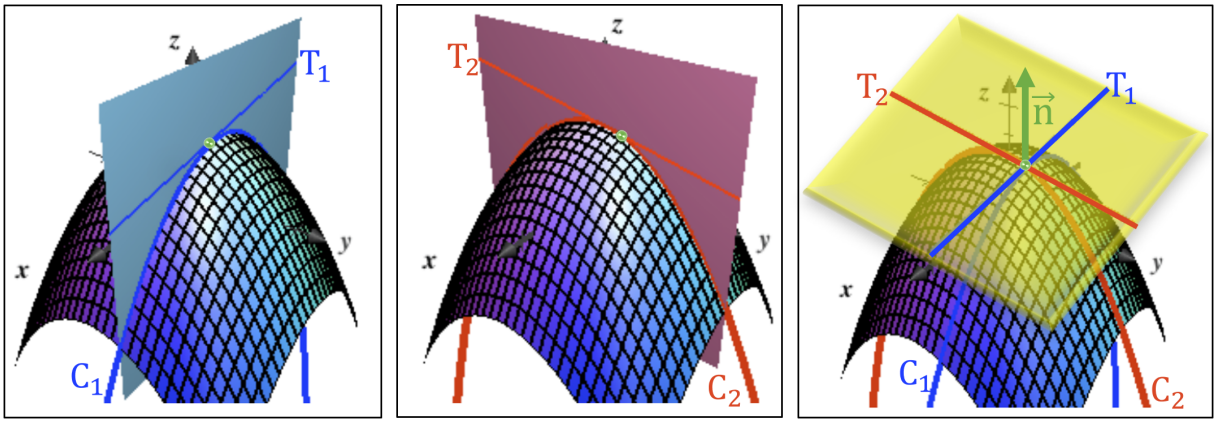


Figure 4.2: Surface with tangent lines that are part of the tangent plane at point $(x_0, y_0, f(x_0, y_0))$, from which the surface normal $\vec{n}$ is established
Left to right: Surface S with plane $y = y_0$. The intersection of the plane and the surface yields curve C_1. T_1 is the tangent line to curve C_1; Surface S with plane $x = x_0$. The intersection of the plane and the surface yields curve C_2. T_2 is the tangent line to curve C_2; Surface S alongside curves (C_1, C_2), tangent lines (T_1, T_2), tangent plane (colored in yellow) and the targeted normal $\vec{n}$ (colored in green) [30]

In summary, the tangent plane to the central objective point P_C consists of infinite tangent lines that lie directly on the tangent plane. To calculate the normal to the surface

and likewise to the tangent plane, which goes through the point P_C, a total of two tangent lines that intersect in P_C are needed, denoted as (T_1, T_2).

When dealing with a point cloud, without an explicit definition of the surface function, the tangent plane and thus $\vec{n}$ cannot be directly computed. However, T_1 and T_2 can be expressed as vectors $(\vec{t}_1, \vec{t}_2)$ and the tangent plane as an unlimited small surface. Besides point P_C, a total of two neighboring points (P_a, P_b) are needed to define the tangent vectors. (P_a, P_b) can be approximated by finding the two closest points to P_C while having the steepest spatial coordinate change in order to approximate a tangent. This is an extended explanation of the approach used by the authors of [47], where this context is established in an equivalent ordered point cloud projection image $I_{x,y,z}$ to calculate normal $\vec{n}$ of the central point of interest P_c mapped at image coordinates $I_{x,y,z}(u = u_0, v = v_0)$, which is illustrated in Figure 4.3 [47].

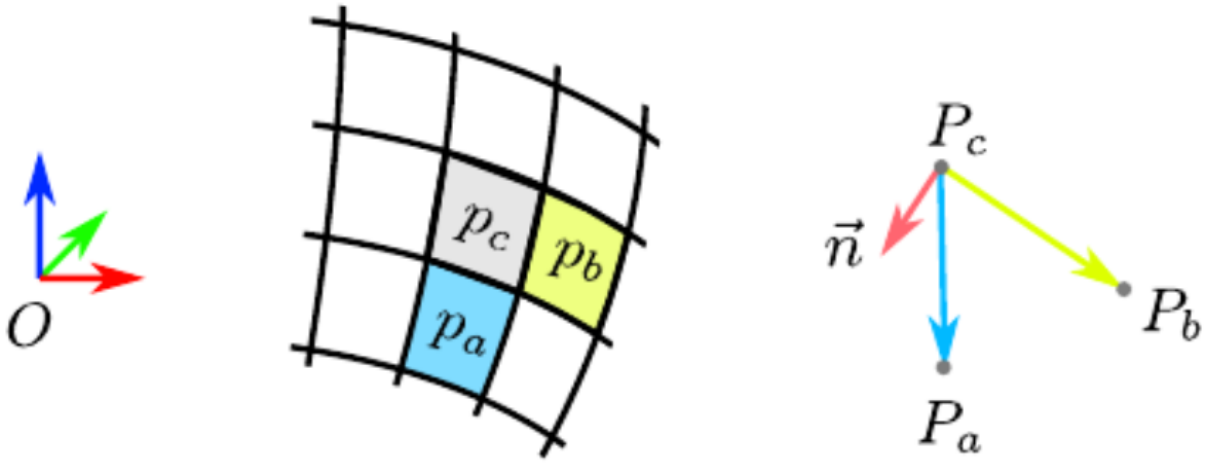


Figure 4.3: From left to right: The coordinate system of the sensor; The pixel p_c and its neighboring pixels p_a and p_b in $I_{x,y,z}$; The corresponding 3D cartesian points (P_c, P_a, P_b) along with the normal vector $\vec{n}$ [47]

Here, the point of interest is pixel $p_c = (u_0, v_0)$ with the corresponding 3D point $P_c = I_{x,y,z}(u_0, v_0) = (x_0, y_0, z_0)$. Its neighboring pixels are defined as $p_b = (u_0 + 1, v_0)$ and $p_a = (u_0, v_0 + 1)$, corresponding to $P_b = I_{x,y,z}(u_0 + 1, v_0)$ and $P_c = I_{x,y,z}(u_0, v_0 + 1)$. Given these points, the unit surface normal at P_c can be calculated (magnitude not of interest):

$$\vec{n} = \frac{\overrightarrow{P_cP_b} \times \overrightarrow{P_cP_a}}{\left\| \overrightarrow{P_cP_b} \times \overrightarrow{P_cP_a} \right\|_2} \tag{4.3}$$

The vectors $\overrightarrow{P_cP_b}$ and $\overrightarrow{P_cP_a}$ are equivalent to the tangent vectors $(\vec{t}_1, \vec{t}_2)$. However, this approach relies on the geometric relationships between just a few points (specifically 3), which may not fully capture the local surface orientation, and approximate the tangents best. This is where image gradients come into play, a more robust alternative incorporating a broader and weighted array of spatial locations. Generally used for edge detection to detect the highest intensity changes in an image, when applied to a spatial image, i.e., $I_{x,y,z}$, it measures the variation in spatial coordinates across the image, providing insights into the surface's geometry encoded within. By mapping the

direction of the steepest change in (x, y, z)-coordinates, gradient kernels, such as the Scharr kernels K_u and K_v, effectively translate the spatial variations of each cartesian axis component into vectors that describe the surface's local geometry. This is achieved through specific configurations of the kernel:

- Horizontal gradient detection (K_u):

$$K_u = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \tag{4.4}$$

- Vertical gradient detection (K_v)

$$K_v = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{4.5}$$

These kernels prioritize direct horizontal or vertical neighbor pixels over diagonal ones, enhancing sensitivity to spatial changes along the primary axes.

When convolved with $I_{x,y,z}$, K_u and K_v compute gradients that effectively map the steepest change in the (x,y,z)-coordinates, transforming spatial variations into directional vectors $\overrightarrow{P_cP_b}$ and $\overrightarrow{P_cP_a}$ tangent to the surface. This aligns with Equation 4.3 and allows the calculation of $\vec{n}$.

With $\vec{n}$ obtained, the angle $\cos(\theta)$ between $\vec{n}$ and the gravitational direction $\vec{e_z}$ is defined as $\cos(\theta) = \dfrac{\vec{n} \cdot \vec{e_z}}{|\vec{n}| \cdot |\vec{e_z}|}$, with $\vec{e_z} = \langle 0, 0, 1 \rangle$. As the magnitudes of the two vectors are equal to 1, the equation simplifies to $\cos(\theta) = \vec{n} \cdot \vec{e_z}$, which is essentially the z-component of $\vec{n}$. This value, $\cos(\theta)$ provides insights into the orientation of the surface:

- A value of "1" signifies that the surface normal points directly upwards, aligned with the positive z-axis. This indicates a horizontal surface, like the ground plane, facing upwards.
- A value of "-1" suggests that the surface normal is pointing directly downwards, opposite to the positive z-axis, and aligned with the gravitational direction. This indicates a horizontal surface facing downwards.
- A value of "0" for $\cos(\theta)$ indicates that the surface normal is perpendicular to the z-axis and parallel to the xy-plane. This signifies a vertical surface, akin to a wall or facade.

Applying the convolution with the Scharr kernel on $I_{x,y,z}$ is visualized in Figure 4.4, from which the surface normals provide a good visual understanding of which points are part of the ground and about abrupt surface slope differences.

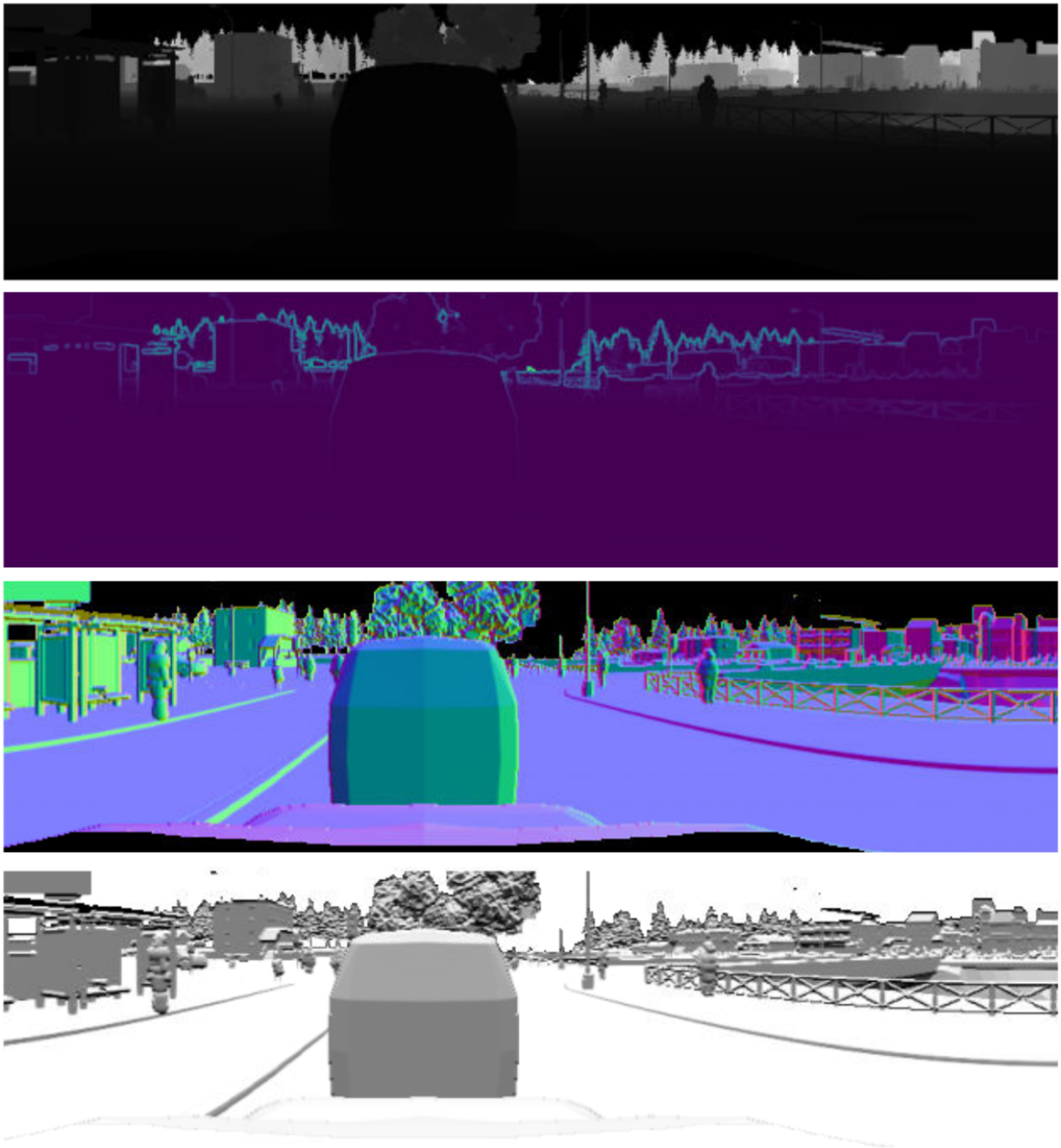


Figure 4.4: From top to bottom: Depth image; Gradient magnitude of x-channel; Surface normal estimation $\vec{n}$; Normalized $\cos(\theta)$: $\frac{\cos(\theta)+1}{2}$

4.1.4 Normalization of channel values and construction of the finalized input image

Finally, the channel values undergo normalization, and the resulting images are subjected to a series of augmentations. The Euclidean distances get normalized relative to the sensor's operational range, delineated with a maximum distance of 250 meters. This choice presents a deviation from the approach described in [53], wherein depth values

are uniformly transposed onto a grayscale spectrum, ranging from 0 to 255. Although this transformation is common, it inherently limits the depth resolution and potentially leads to the loss of critical depth information.
For the representation of height above the ground, values are normalized against a threshold of 50 meters, encapsulating all relevant objects within the scene. $\cos(\theta)$ is normalized with $\frac{\cos(\theta)+1}{2}$, where $\cos(\theta) \in [0,1]$. The channel values are stored in a 32-bit floating-point format to ensure precision and minimize information loss.

The image augmentations include random horizontal flipping, sub-image cropping, and a final resizing operation. In this context, sub-image cropping is utilized to form output images based on the desired vertical FOVs by solving Equation 3.3 to I_H $= \frac{\text{F\_V}}{\Delta\theta}$, where $\Delta\theta$ is constant by initialization (fixed sensor configuration, see Chapter 3.1.2). This is why the sensor's vertical FOV was originally configured to be overly large, to accommodate all possible desired FOVs found in LiDAR applications. In this work, the FOVs are inspired by the Ouster OS2 [36] and Ouster OS1 [35], which are 22.5° and 45°, respectively. This deduces the current image dimensions of 1024 × 2048 (I_H × I_W) to 128 × 2048 (see Fig. 4.5) and 256 × 2048 (see Fig. 4.6) while maintaining the full 360° horizontal FOV.

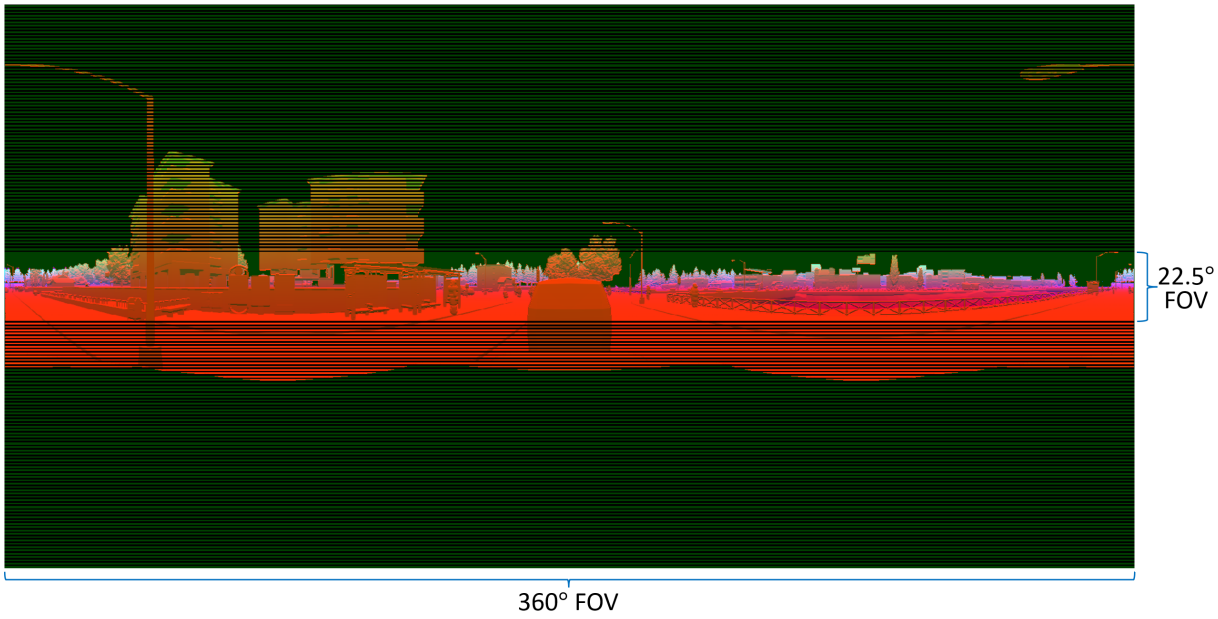


Figure 4.5: Cropped image with a FOV of 22.5°

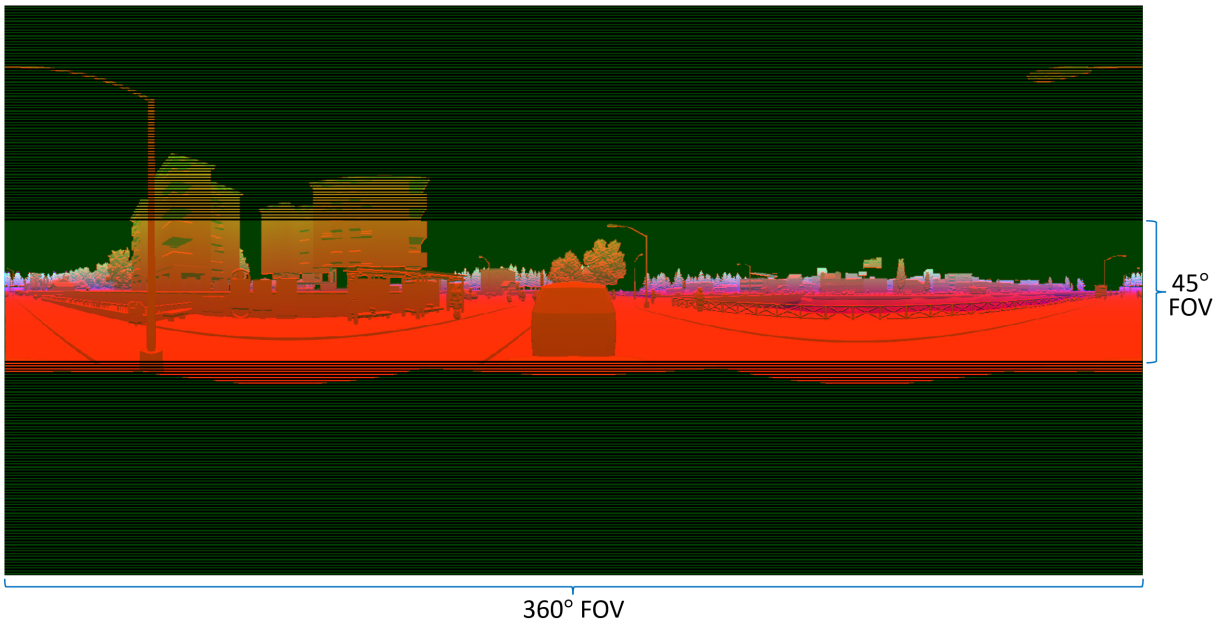


Figure 4.6: Cropped image with a FOV of 45°

The final phase in the augmentation sequence involves resizing the cropped images to standardized dimensions. This dimension is predicated on the maximum FOV used in the sub-image cropping step, set at 256×2048.
The input image is now formulated and ready to be input into the keypoint detection model.

4.2 Keypoint Detection Model

Feature extraction over the input image was performed using a ResNet-50-FPN backbone that yields a set of convolutional feature maps to extract the ROIs. A box head performs object classification and bounding box regression (format: XYXY) using the feature maps provided by the backbone. An optional mask head yields instance segmentation masks, and a keypoint detection head predicts specific key points on the objects detected by the box head network. This keypoint head is designed to predict the 16 keypoints delineated by the adapted custom CARLA skeleton structure (see Section 3.1.3) and operates with a pooling resolution of 14×14. It produces dense heatmaps for each of the 16 keypoints, signifying the confidence that this particular location is the center of a specific keypoint. Once these heatmaps are generated, they are subjected to a sigmoid activation function, ensuring the normalization of confidence scores between 0 and 1. A series of sophisticated transformations and refinements (e.g., peak detection, refinement, thresholding) produce the final set of detected keypoints $p = [\vec{p}_1, \ldots, \vec{p}_N]$, where $N = 16$ and the i-th 2D keypoint is represented by $\vec{p}_i = \langle u_i, v_i \rangle^T$. The network architecture is depicted in a simplified form in Figure 4.7.

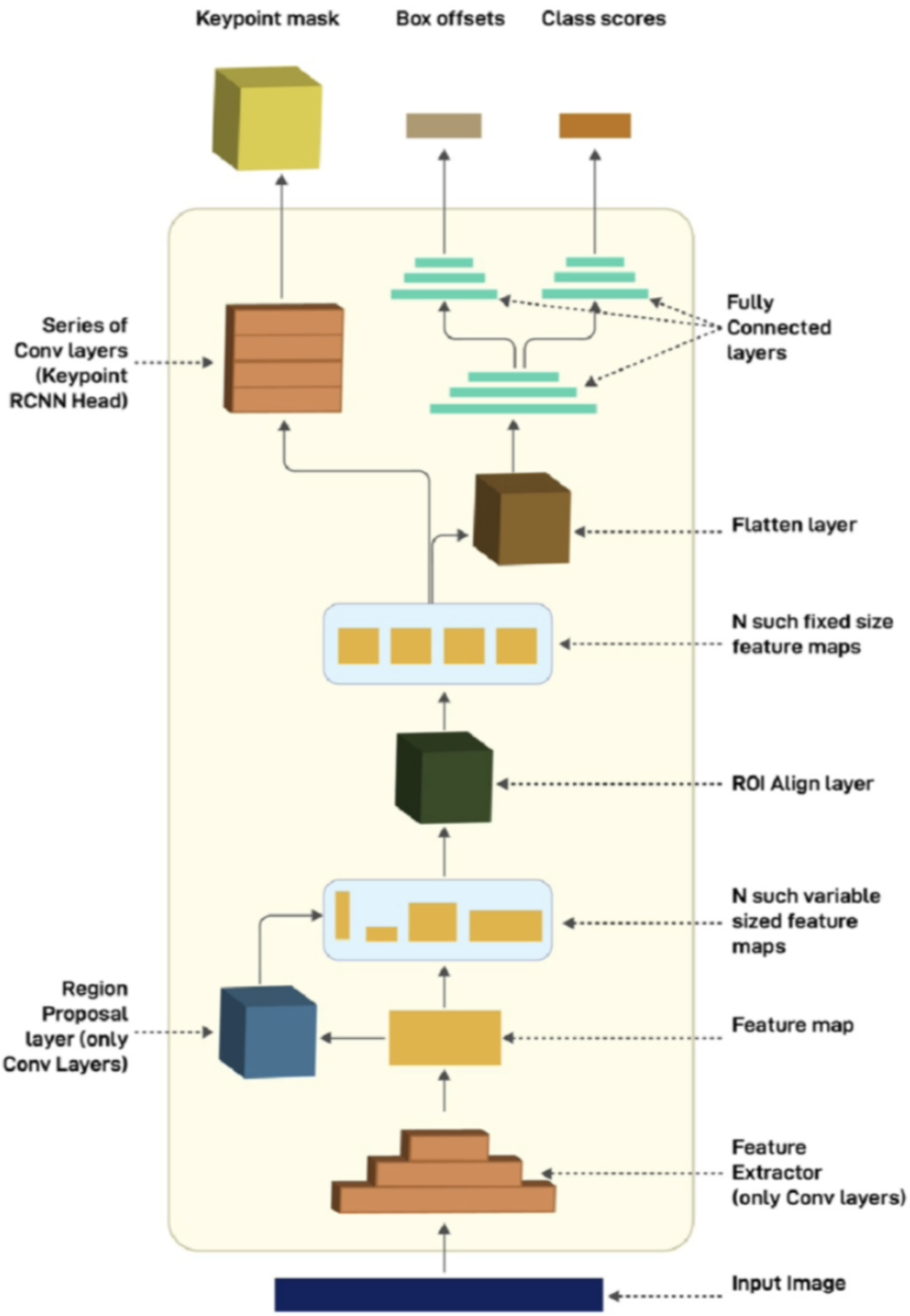


Figure 4.7: Simplified representation of the keypoint detection model with a ResNet-FPN backbone including the keypoint head [59]

4.3 2D-3D-Lifting Network

Instead of normalizing the pixel coordinates p based on image dimensions, this work makes use of the projection matrix K, introduced in section 3.2.1, and transforms the 2D pixels into 3D space, specifically into unit direction vectors (see Eq. 4.7) being the input of the lifting network.

The equation 3.1 is solved for X:

$$X = K^{-1} \cdot U = K^{-1} \cdot \begin{bmatrix} u_1 & \dots & u_N \\ v_1 & \dots & v_N \\ 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} \phi_1 & \dots & \phi_N \\ \theta_1 & \dots & \theta_N \\ 1 & \dots & 1 \end{bmatrix} = \begin{bmatrix} \vec{\phi} \\ \vec{\theta} \\ \vec{1_N} \end{bmatrix} \tag{4.6}$$

The spherical coordinate matrix X is used to calculate the unit direction vectors, represented as cartesian coordinate vectors $\vec{u_x}, \vec{u_y}, \vec{u_z}$:

$$\begin{aligned} \vec{u_x} &= \sin\left(\frac{\pi}{2} - \vec{\theta}\right) \circ \cos\left(\vec{\phi}\right) \\ \vec{u_y} &= \sin\left(\frac{\pi}{2} - \vec{\theta}\right) \circ \sin\left(\vec{\phi}\right) \\ \vec{u_z} &= \cos\left(\frac{\pi}{2} - \vec{\theta}\right) \end{aligned} \tag{4.7}$$

The combined vectors can be represented in matrix format $U_{xyz} = \left[\vec{u_x}, \vec{u_y}, \vec{u_z}\right]^T$ or preferably in vector format:

$$\vec{u}_{xyz} = \langle u_{x1}, u_{y1}, u_{z1}, \dots, u_{xN}, u_{yN}, u_{zN} \rangle \tag{4.8}$$

to be directly applicable in Neural Networks, as they require flattened input.
Note, that the 3D unit vectorization is also applicable for camera intrinsic matrices $K_C = \begin{bmatrix} fx & 0 & \text{I\_W}/2 \\ 0 & fy & \text{I\_H}/2 \\ 0 & 0 & 1 \end{bmatrix}$, where fx is the focal length of the horizontal direction and fy is the focal length of the vertical direction. When using K_C in 4.6, [9] refers to X as normalizing the 2D keypoints, yet $\vec{u}_{xyz}$ can still be constructed with Equation 4.7. [31] has shown that converting the 2D joint locations to 3D unit vectors provides better accuracy and estimation results in ego-centric HPE. To the best of current knowledge, no paper has addressed this representation for absolute HPE.

An enhancement has been made by concatenating the median depth (median_d) of the target instance to $\vec{u}_{xyz}$, which translates to:

$$\vec{u}_{xyz,d} = \langle u_{x1}, u_{y1}, u_{z1}, \dots, u_{xN}, u_{yN}, u_{zN}, \text{median\_d} \rangle \tag{4.9}$$

This is available from the Euclidean input channel (see Section 4.1.1). By filtering the pixels that fall either inside the bounding box extent or, for more accurate results, are part of the instance segmentation mask, the median value from that data points is computed. Quantitative testing revealed faster training convergence, reduced pose error, and training loss with the median depth-enhanced $\vec{u}_{xyz,d}$, compared to its absence.

As the 3D pose translation is the most challenging and error-prone part of absolute HPE, providing a depth estimate of the object's distance mitigates spatial ambiguities. However, qualitative validation with expanded datasets to evaluate performance differences has not been conducted due to time constraints.

The 2D-3D-lifting framework aims to predict a one-dimensional radial distance for each joint. The radial distance of 3D joints is defined by:

$$\vec{r} = \langle r_1, \ldots, r_N \rangle \,, \quad r_i = \sqrt{X_i^2 + Y_i^2 + Z_i^2} \tag{4.10}$$

where N is the number of joints, and i-th radial distance $r_i \in \vec{r}$ is calculated by the i-th 3D joint location $P_i \in P$ with $P_i = \langle X_i, Y_i, Z_i \rangle^T$.
The i-th 3D pose P_i can be recovered with the unit direction vectors $\vec{u}_{xyz_i} \in \vec{u}_{xyz}$ by $P_i = r_i \cdot \vec{u}_{xyz_i}$.

Aligned with previous works on absolute HPE, the radial distance vector $\vec{r}$ is decomposed into relative distances of joints to the root joint $\vec{r}_{rel}$ - predicted by the (a) Relative Distance Network through $f_{rel}(\vec{u}_{xyz,d})$ - and the absolute distance to the root joint r_{Root}, estimated by the (b) Root Distance Network through $f_{Root}(\vec{u}_{xyz,d})$:

$$\vec{r} = \langle r_1, \ldots, r_N \rangle = \vec{1_N} r_{Root} + \vec{r}_{rel} \tag{4.11}$$

where $r_{Root} \in \mathbb{R}^1$ and $\vec{r}_{rel} \in \mathbb{R}^N$. These two regression networks are adopted from [9] and shown in Figure 4.8.

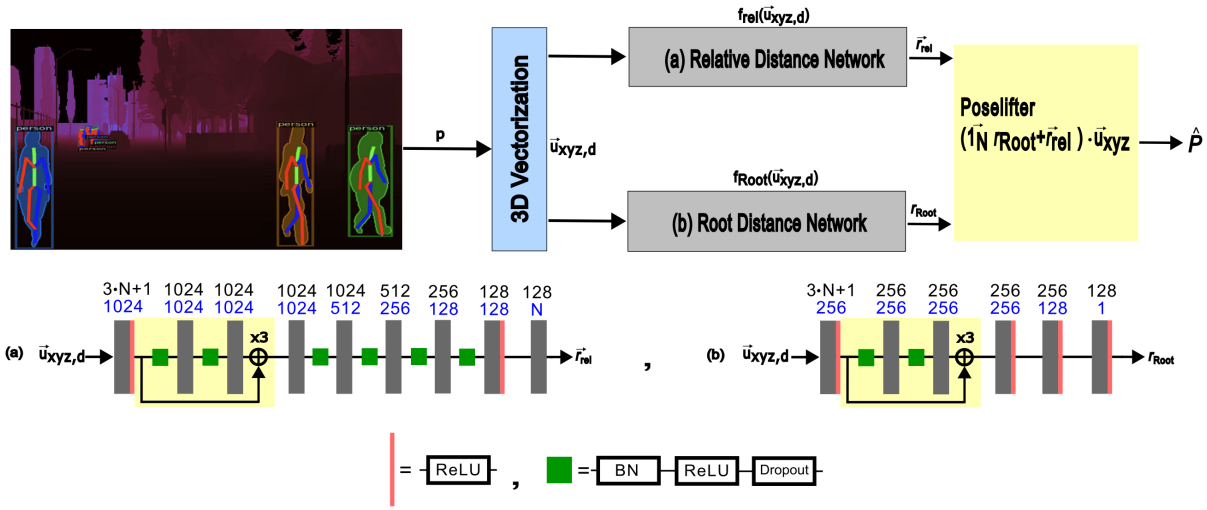


Figure 4.8: Architecture of the lifting network, consisting of the two regression networks (a) Relative Distance Network and (b) Root Distance Network

4.3.1 Loss Function

The framework is end-to-end trainable and equipped with a loss function $\mathcal{L}$ that combines pose error $\mathcal{L}_P$, distance error $\mathcal{L}_D$ and joint relation constraints $\mathcal{L}_C$:

$$\mathcal{L} = \lambda_P \cdot \mathcal{L}_P + \lambda_D \cdot \mathcal{L}_D + \lambda_C \cdot \mathcal{L}_C \tag{4.12}$$

where the coefficients λ are control parameters.

The pose error measures the L1-loss (MAE) between predicted 3D poses $\hat{P}$ and the ground-truth 3D poses P.

$$\mathcal{L}_P = \frac{1}{N} \sum_{i=1}^{N} \|\hat{P} - P\| \tag{4.13}$$

The distance loss measures the MAE between the predicted joint radial distances $\hat{\vec{r}}$ and the label distances $\vec{r}$.

$$\mathcal{L}_D = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\vec{r}} - \vec{r}\| \tag{4.14}$$

The joint relation constraints capitalize on the inherent symmetrical structure of the human body, particularly focusing on the bones and their connecting joints.
Given this bilateral symmetry of the human body, where each side of the body mirrors the other in terms of bone structure and joint connections, the bone length symmetry loss term minimizes the lengths of matching bones on the body halves.
Additionally, the directional constraint loss term limits the bone orientation. This reduces the likelihood of generating anatomically unrealistic bone orientations. The directional vectors of bones implicitly address angular constraints too, ensuring that the orientation and movement of bones remain within physiologically realistic bounds established by the ground-truth poses. The implementation of joint relation constraints is dependent on the skeleton human body model used. The considered bones of the custom CARLA skeleton representation are shown in Figure 4.9, where associated bone pairs are colored the same. The juxtaposed bones are defined as vectors $\vec{b_i}$ and $\vec{b_j}$, from which the length difference between the predicted bone counterparts $(\hat{\vec{b}}_i, \hat{\vec{b}}_j)$ can be calculated as well as the directional difference, presented as $\mathcal{L}_{C_{i,j}} \in \mathcal{L}_C$:

$$\mathcal{L}_{C_{i,j}} = \left(\|\hat{\vec{b}}_i\| - \|\vec{b_j}\| \right) + \|\hat{\vec{b}}_i - \vec{b_j}\| \tag{4.15}$$

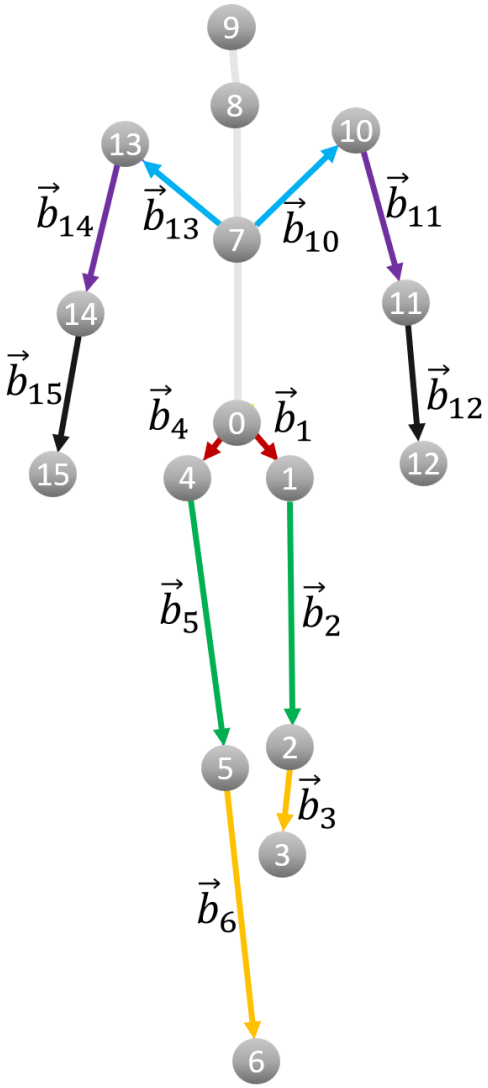


Figure 4.9: Bone pair symmetry of the custom CARLA skeleton. The selected bones subjected to the joint relation constraint loss are considered 3D vectors. The bilateral symmetrical bone vectors are colored the same.

4.3.2 Implementation

The training is run using the Adam optimizer with a batch size of 64. The model weights are randomly initialized and trained in two phases: Firstly with ground-truth labels for 10 epochs at a learning rate of LR $= 0.001$ and then fine-tuned @LR=1E-4 with $\vec{u}_{xyz,d}$ inputs originating from the 2D estimator predictions for instances that are in the range $\leq 40m$ to the sensor. This threshold is based on the approximate stopping distance calculation (see Eq. 4.16, integrating the reaction distance and braking distance) when driving at 50 km/h (German speed limit in urban areas) with a 10% tolerance.

$$D_{total} \approx v \cdot t + \frac{v^2}{2\mu g} \tag{4.16}$$

Where:

- v is the initial velocity of the car in m/s (here: 55km/h),
- t is the reaction time in seconds (mean estimate of 1.5 seconds),
- μ is the coefficient of friction between the tires and the road (it varies but a common value is 0.7 for dry asphalt),
- g is the acceleration due to gravity ($9.81 m/s^2$),
- D_{total} is the total stopping distance and calculates with the given conditions to approx. 40m.

The weights of the loss function are empirically selected, where λ_P and λ_D are set to 1. $\lambda_C = 0.25$ at the first training stage and 1 when fine-tuning, where a stronger emphasis on realistic poses is established by the joint relation constraints while learning to reduce bias introduced by the predicted 2D keypoints.

The first training stage can be operated independently of the 2D pose estimator. This allows for the use of synthetic training data, which has the advantage of being scalable in terms of volume by augmenting the 3D poses in cartesian coordinates with various translations and rotations. From the 3D pose labels, $\vec{u}_{xyz}$ is established from Equations 2.3 and 4.7. The missing median distance parameter in $\vec{u}_{xyz,d}$ can be obtained by calculating the median distance of the 3D joints and adding some bias. Here, the instance segmentation mask outputs of the 2D pose estimator can be used to zero-center the point clouds based on the root joint locations. Then, the standard deviation of Euclidean distances is computed for each instance. Finally, the bias is defined as a bidirectional variable fraction of the global (mean) standard deviation.

When using the prediction results from the 2D estimator (for both training and testing), the performance of the lifting network is extremely dependent on the quality of the prediction results. This is why a crucial preliminary step involves instance matching before the lifting network processes any predicted instance, utilized by the Hungarian algorithm [25, 32]. This process pairs predicted instances with their closest ground-truth counterparts based on the similarity of their bounding box locations, measured using the Intersection over Union (IOU) metric, and handled as true positives for IOU ≥ 0.5.

Chapter 5

5 Evaluation

This chapter addresses the performance of the proposed methods, firstly assessing the 2D estimator, followed by evaluating the lifting network. This work utilizes a custom synthetic dataset and makes comparison to other works unfeasible. However qualitative assessment of the evaluation results can be made.

5.1 Evaluation of the 2D keypoint estimator

Before delving deeper into the detailed aspects of the employed metrics, it is vital to address the inherent limitations that potentially compromise the model's precision in the 2D keypoint estimation task, which also affects the lifting model.

1. **Sparse data representation and restricted pixel coverage of objects:**
   A notable limitation in the current model is the sparse data representation, particularly when compared to its high-resolution RGB counterparts. This aspect manifests as a restricted pixel coverage of objects, potentially introducing errors in the detection process. Mainly, when analyzing objects situated at considerable distances, the relatively small appearance (see Fig. 5.1, left and right image) of these objects might induce a bias or diminish the accuracy in detection. Furthermore, the model might encounter difficulties distinguishing between adjacent joints that are closely aligned or even situated at the same pixel location.

2. **Overlapping instances and malformed skeleton structure:**
   The issue of overlapping instances presents a significant challenge, particularly in scenarios where it becomes difficult to recognize individual entities distinctly. This phenomenon is illustrated in Figure 5.1 (middle image), where the overlapping instances could not be distinguished during prediction. Thus, the keypoint head would generate multi-modal heatmaps (multiple spikes of similar magnitudes). Because the set of keypoints is of fixed size and the actual number present in that bounding box area is larger, the quality of the final set of keypoints $\hat{p}$ is arbitrary and mostly results in the formation of highly distorted and malformed skeletons. While this problem is less critical at greater distances, where the car's vision system

has extended response time to react, it becomes considerably more significant at closer proximities. The complexities of overlapping instances can also be observed when individuals are positioned closely behind one another or when there are greater overlaps.

3. **Interchanging symmetrical pairs of joints:**
   Due to the bilateral anatomical skeleton structure, i.e. symmetrical pairs of joints on opposing sides of the body (e.g., the left and right ankle), they can get incorrectly interchanged during prediction. This phenomenon affects various joints, including the hip, shoulder, knee, elbow, wrist, and ankle, with the level of vulnerability to precision degradation increasing in the order listed. This presents a unique challenge, especially for joints with a high degree of freedom, leading to increased localization errors in image-wise detection.

In order to counteract missing keypoints in the dataset and provide a uniform joint presence, only fully visible instances and complete skeletons are considered in the evaluation.

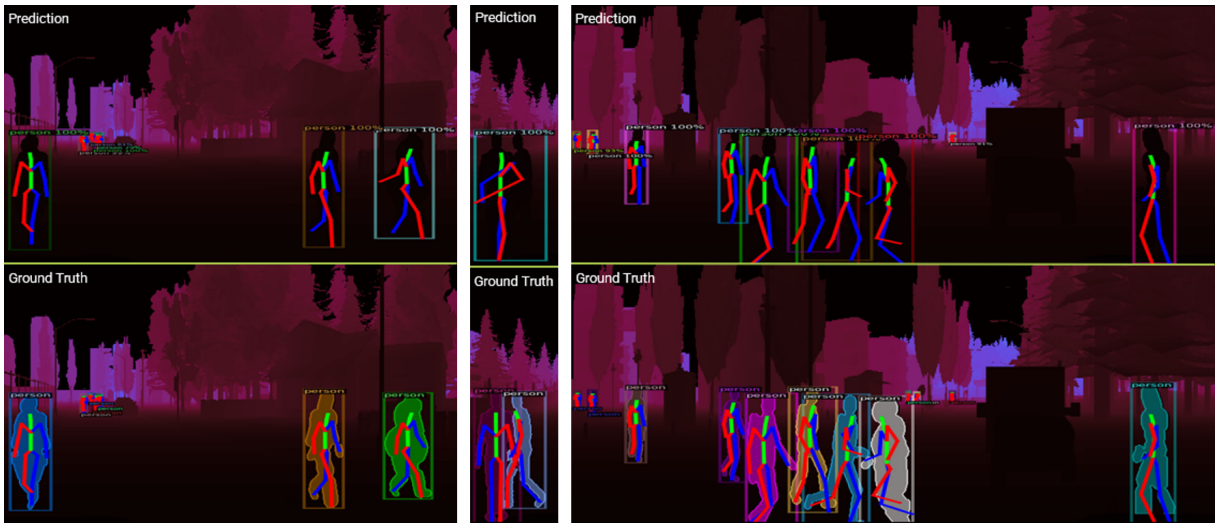


Figure 5.1: Comparison of predicted keypoints and ground-truth, illustrated with underlying bone connections

Employed metrics

The employed evaluation metrics are Mean Per Joint Position Error (MPJPE) [61] and Percentage of Correct Keypoints (PCK) [61], which are adapted to incorporate scale-invariant measurement to reduce the sensitivity of the metrics of varying object sizes and of image resizing effects (see Section 4.1.4).

Traditionally, MPJPE finds its primary application in 3D space evaluations. However, it can be adapted to evaluate pixel location differences. The error is calculated as the pixel distance relative to the BBOX diagonal length, resulting in a scale-invariant and

standardized measurement.

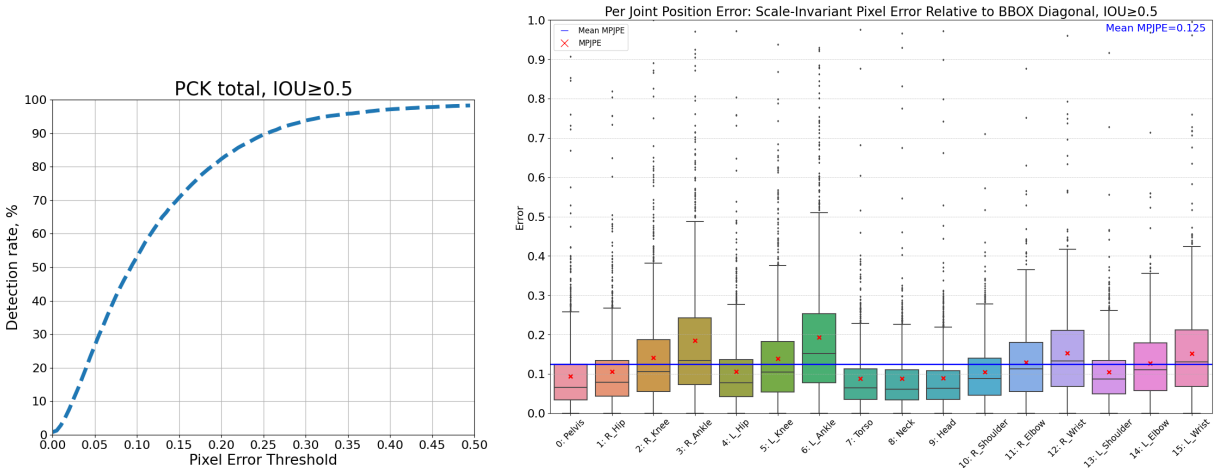
$$\text{MPJPE} = \frac{\left(\frac{1}{N}\sum_{i=1}^{N}\|p_i - \hat{p}_i\|_2\right)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{5.1}$$

where N is the number of joints, and $p_i \in p$ is the i-th ground-truth 2D joint and $\hat{p}_i$ is the equivalent i-th predicted keypoint. The utilized bounding box format is XYXY, which depicts the upper-left coordinates (x_1, y_1) and bottom-right coordinates (x_2, y_2), from which the bounding box diagonal is calculated (represented in the denominator of Equation 5.1). The final error is accumulated and averaged for the set of instances over all frames in the test set.

Concurrently, the PCK metric classifies a detected joint as correct if the distance between the predicted and actual location resides within predetermined error thresholds. The traditional PCKh metric measures the accuracy of keypoint detection relative to the head size to preserve proportional consistency with the body size. However, owing to the constraints in pixel granularity, selecting a larger reference size becomes imperative. In this context, the thresholds are established as fractions relative to the bounding box diagonal (referenced as "Pixel Error Threshold" in Figure 5.2a and 5.3a).

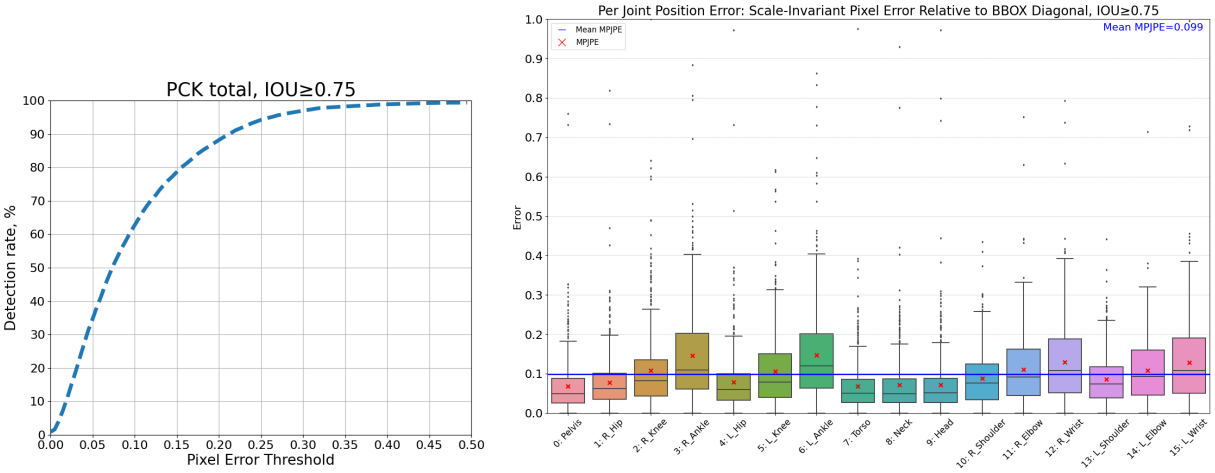
5.1.1 Results

The results are shown with respect to IOU thresholds 0.5 and 0.75. Stricter instance matching @IOU=0.75 (see Figure 5.3) naturally presents a more refined outcome with a total of 611 walkers being evaluated compared to 1107 instances @IOU=0.5 (see Figure 5.2).



(a) PCK **(b)** MPJPE

Figure 5.2: Metrics @IOU=0.5



(a) PCK **(b)** MPJPE

Figure 5.3: Metrics @IOU=0.75

The results at both IOU levels show a notable similarity. The discrepancies in joint position errors are vividly illustrated through a boxplot, encapsulating variations in the performance of all 16 joints that might not be immediately discernible when relying solely on the overall MPJPE, denoted as Mean MPJPE.

The interquartile range (IQR) and the outer quartiles represented by the 1st and 4th quartiles of the boxplot accentuate the disparities in error susceptibility across various joint locations, particularly pronounced at joints characterized by a high degree of freedom. With an extensive range of motion, these joints allow for more complex and varied movements compared to other joints in the human body. Thus exhibiting a pronounced variability in error distribution, as evidenced by larger interquartile ranges and broader spreads in the outer quartiles compared to joints with limited mobility. This phenomenon can be attributed to the complex nature of the movements associated with these joints, which often involve a combination of complex rotational and translational motions, making accurate keypoint detection considerably more challenging.

Despite these challenges, the extent of the errors remains within an acceptable range, with an average MPJPE of approximately 12.5% at IOU=0.5 and 10% at IOU=0.75.

Upon observing the data points that constitute outliers, transcending the boundaries of the 4th quartile, it becomes evident that these are situated at considerable distances from the sensor. The small bounding box area exacerbates this issue, as even a small deviation can constitute a significant portion of the object's representation in the image, thereby leading to disproportionately high error values.

Following the analysis of the MPJPE boxplot, examining the PCK curve is an additional tool in evaluating the accuracy of keypoint detections. The model is adept at achieving a detection rate of 65% across both mean MPJPE benchmarks. The more pronounced slope observed at an IOU of 0.75 is a significant marker of the model's enhanced sensitivity and improved performance at higher IOU thresholds. Additionally, when the pixel error threshold is synchronized with the mean MPJPE at an IOU of 0.5, the detection

rate climbs to a respectable 75%. Given the limitations discussed earlier, these results can be regarded as quite encouraging, indicating a solid foundation upon which further refinements can be built in future works.

5.2 Evaluation of the 2D-to-3D lifting network

The quantitative evaluation of the proposed lifting method is based on the MPJPE metric. Equation 5.2 compares the ground-truth 3D pose $\hat{P}$ with the predicted pose P and measures the distance error in meters.

$$\text{MPJPE} = \frac{1}{N}\sum_{i=1}^{N}\left\|P_i - \hat{P}_i\right\|_2 \tag{5.2}$$

Similar to the previous Section 5.1, the joint position errors are presented for each joint in the boxplot 5.4 using the predicted and as true positives (i.e., @IOU$\geq$ 0.5, see Chapter 4.3.2) declared 2D joints of the 2D estimator as inputs for the lifting network. By including distance filtering a total of 742 instances were evaluated, compared to 1107 instances at the same IOU-threshold used in the 2D estimator testing.

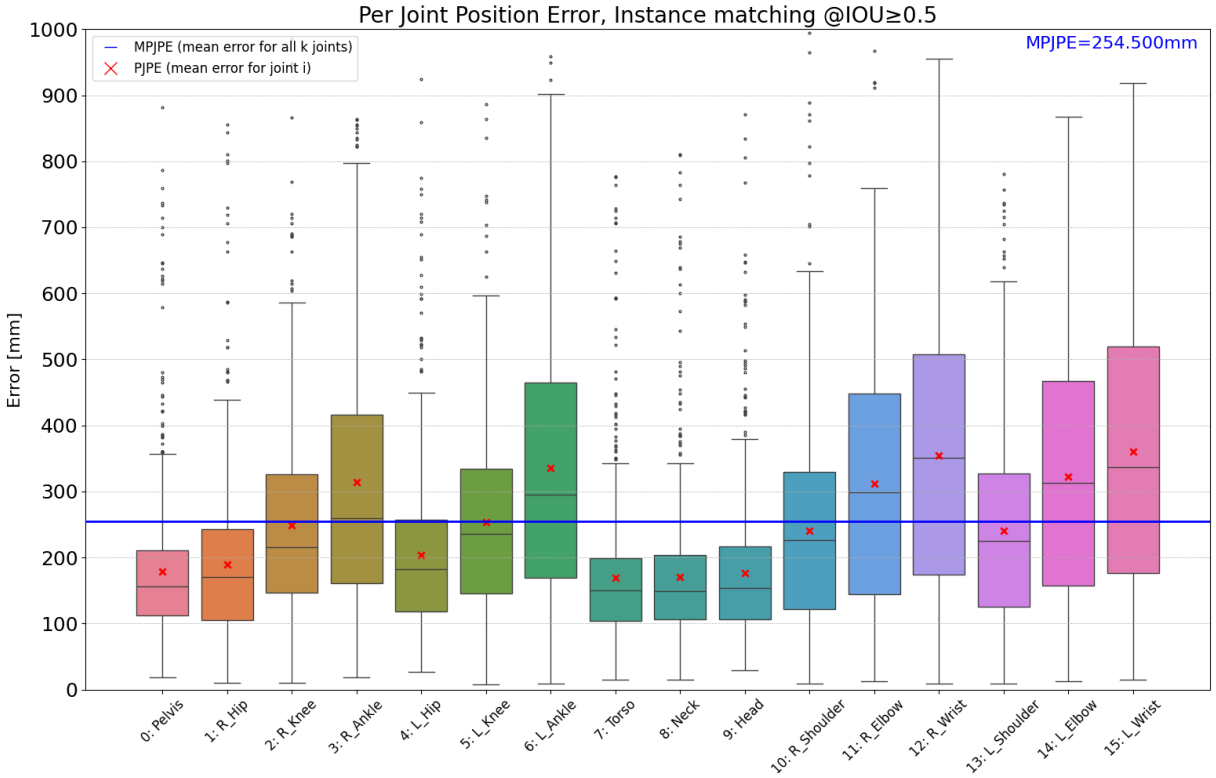


Figure 5.4: Per joint position error boxplot @IOU$\geq$ 0.5

The MPJPE over all joints is 25.5cm. Here, the joints with a high degree of freedom have the highest error and interquartile range. This observation is similar to the boxplots referring to the 2D estimator, but the IQR seems to be larger as well as having higher

4th quartiles. Testing showed that 3.5% of the test subjects have a MPJPE over 1 meter. With higher IOU-thresholds the MPJPE does not decrease be significant margins. This implies that the model training is satisfied for the current training data available. Qualitative 3D pose visualizations are shown in the following Figures, where the ground-truth pose $\hat{P}$ are colored in RGB, and the predicted pose P in CMY (Cyan, Magenta, Yellow).

Although utilizing unit direction vectors enhances the representation of pixel locations, the accuracy of the final pose heavily relies on the predicted radial distance, denoted as $\hat{\vec{r}}$. Even minor adjustments to this depth estimate can yield a pose that appears plausible or consistent with learned models, yet it may still be incorrect. Points that appear to align from one perspective do not actually align when viewed from another angle due to discrepancies in radial distance. This situation is shown in Figure 5.5, where $\hat{P}$ and P from the sensor (front perspective) seem to match, but when rotated and observed from a side perspective, the radial distance difference can quickly reach the decimeter range.

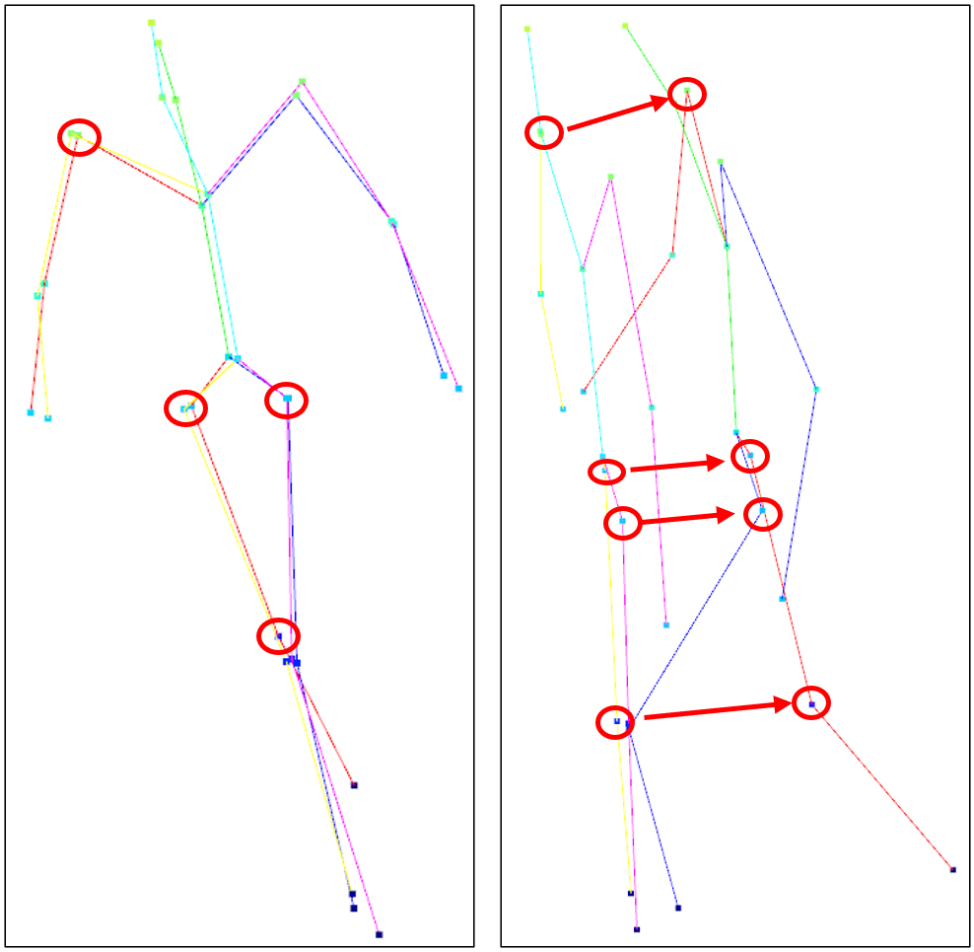


Figure 5.5: Radial distance error with highlighted observation points. GT pose in RGB colors, predicted pose in CMY; left: front perspective; right: side perspective

In Figure 5.6, the problem of instance fusion originating from the false predictions of the 2D estimator is shown. The lack of separation of these instances leads to the largest

errors, which increase the MPJPE to a non-negligible extent.

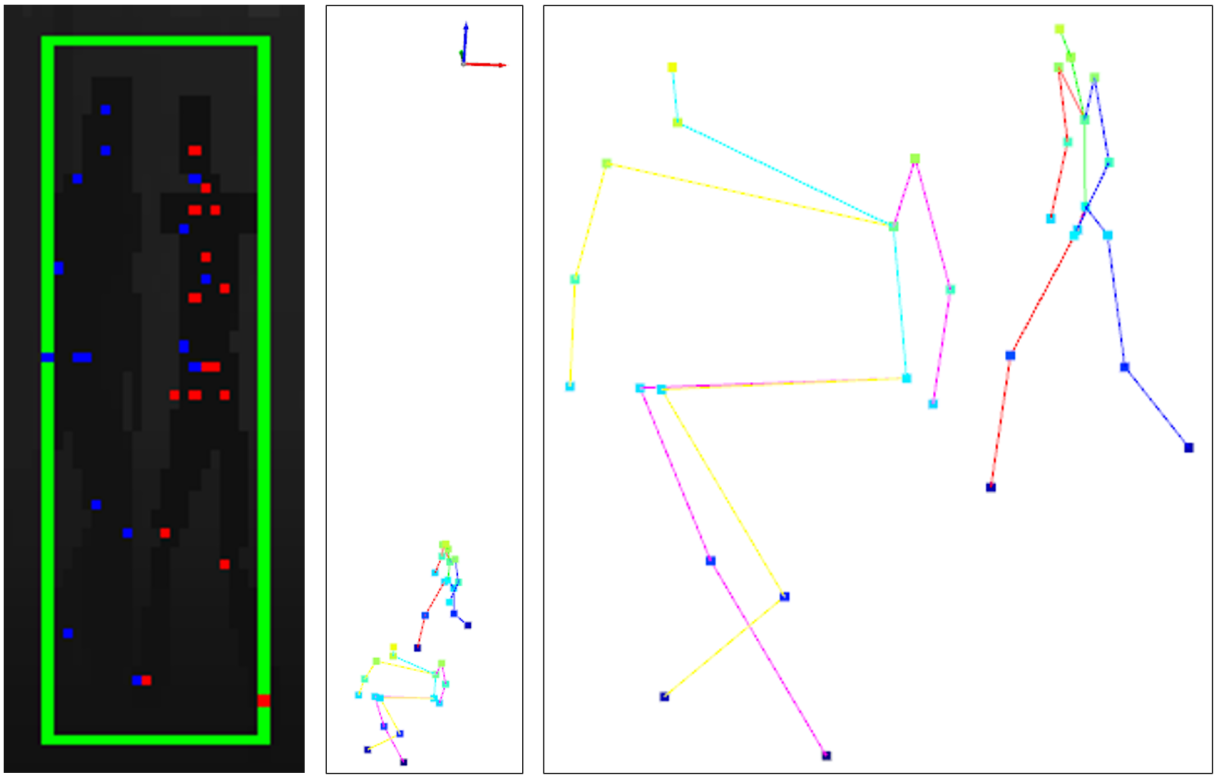


Figure 5.6: Missing separation of instances caused by the 2D estimator. In this example, MPJPE=2m at an object distance of 31 meters. Left to right: Depth map with bounding box in green, ground-truth 2D joints in red, predicted joints in blue; GT pose (RGB) and predicted pose (CMY) in 3D with respect to the sensor coordinate system; Close-up 3D poses

Opposing these imperfect prediction examples, the following Figure 5.7 shows a prediction result that can be considered good, although it still has a 22% higher MPJPE than the average MPJPE. Mostly the joints with high DOF suffer from higher accuracy, as they are most susceptible to errors.

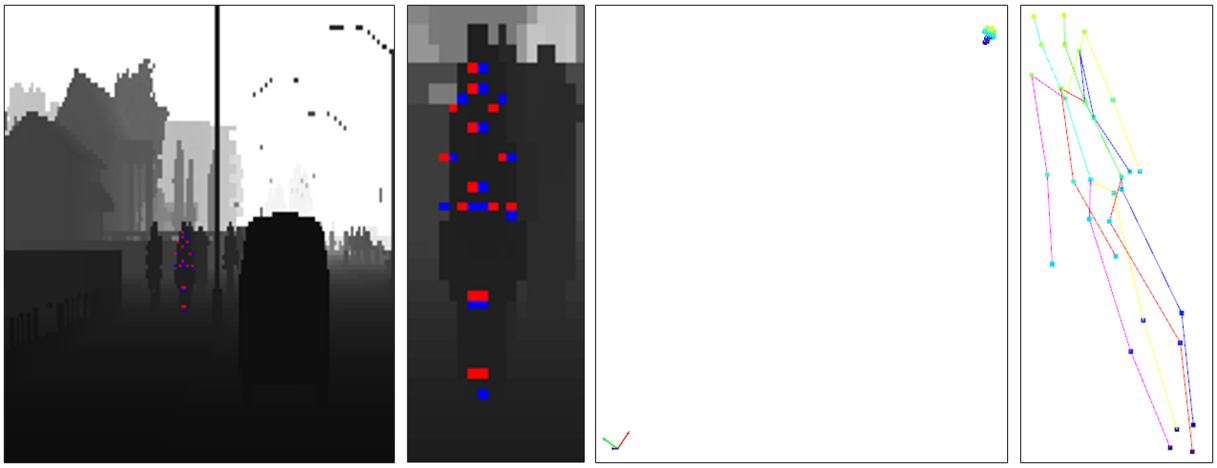


Figure 5.7: Pose comparison with MPJPE=31cm at an object distance of 31meters. Left to right: Depth map with ground-truth 2D joints in red, predicted joints in blue at the 2D estimator stage; 2D joints comparison close-up (red: GT, blue: prediction); GT pose (RGB) and predicted pose (CMY) in 3D with respect to the sensor coordinate system; Close-up 3D poses

Overall, three key challenges further complicate the absolute HPE task, extending beyond the limitations already mentioned at the beginning of Section 5.1. This discussion also includes potential avenues for solutions.

- **Sparse data representation with restricted pixel coverage of objects:**
  The challenge here revolves around the limitations imposed by the current technological capabilities of LiDAR devices. These devices, even the state-of-the-art models, can only provide a limited resolution. This limitation results in a sparse representation of objects within the point cloud data, directly impacting the accuracy of 2D keypoint detection when this data is projected into images. The core issue is that with initial low-resolution data, achieving high accuracy in the predictions is challenging.
- **Crowded scenes and overlapping instances in the employed CARLA simulations:**
  The created environments in CARLA often feature very crowded scenes with multiple overlapping objects. In such scenarios, separating individual instances via object detection becomes increasingly difficult, if not impossible, at higher levels of overlap and negatively affects the 2D keypoint prediction. This issue is exacerbated when attempting to lift these detections into 3D space, introducing biases in the training process and inaccurately inflating error metrics. To mitigate these issues, two potential solutions are proposed. Firstly, integrating a correction network within the lifting network, as suggested by [9]. Secondly, applying unsupervised instance segmentation techniques directly to the point clouds offers an alternative approach. This method involves extracting the bounding box area of the image and then selectively zeroing out segmented pixels not belonging to the focal objects by using a binary segmentation mask. Consequently, only the pixels belonging to the object, along with a less biased segmentation cut-out, are used

for 2D keypoint estimation. This has been effectively demonstrated in a preceding work [58].

- **Small dataset:**
  In comparison to other works, the dataset used in this project appears to be relatively small. The initial data generation phase did not clearly highlight the need for a larger dataset, a realization that only became apparent during the evaluation phase of the lifting model. Time constraints significantly hindered the ability to generate a larger dataset. A more extensive dataset could have potentially improved the accuracy of the models, particularly by enhancing the model's ability to generalize better in the face of biased predictions from the 2D estimator.

5.2.1 Performance comparability

Following the analysis, it becomes evident that the current implementation faces limitations in comparability with existing works in Human Pose Estimation. It is essential to recognize that this research is pioneering a unique approach that leverages LiDAR or depth data in an image format to address 3D keypoint detection. Despite these challenges, the results remain informative and lay the foundation for further advancements in LiDAR-based HPE systems.

Chapter 6

6 Outlook

Overall, the study has successfully demonstrated the potential of HPE, utilizing a representation of LiDAR data as image structures. The CARLA simulator was crucial in generating realistic driving scenarios, supplemented with LiDAR measurements across various environments. The work acknowledges the limitations of the proposed LiDAR data representation and comprehensively evaluates the proposed keypoint detection model. The proposed method incorporates a simple and effective absolute 2D-to-3D human pose-lifting framework and the prediction results are within the range of expectations.

However, it's important to note that this simulation lacks certain real-world complexities, such as noise and potential false measurements, offering a somewhat idealized scenario. This forms a critical area of focus for future work, as understanding and mitigating the impact of these factors is vital for developing a robust and reliable model.
As this research area evolves, future efforts may explore incorporating genuine LiDAR data of autonomous driving datasets, enabling more direct comparability with existing benchmarks.

List of Abbreviations

AV	Autonomous Vehicle
FOV	Field of View
HPE	Human Pose Estimation
IOU	Intersection over Union
IQR	Interquartile range
IV	Intelligent Vehicle
LiDAR	Light Detection and Ranging
MAE	Mean Absolute Error
MEMS	Micro-Electro-Mechanical System
OPA	Optical Phased Array
TOF	Time-of-Flight

Bibliography

[1] AG, Mercedes-Benz G.: *Certification for SAE Level 3 system for U.S. market.* 2023. – URL https://group.mercedes-benz.com/innovation/product-innovation/autonomous-driving/drive-pilot-nevada.html. – Zugriffsdatum: 2024-02-26

[2] AUTONOMOUS, Jeremy; T.: *5 Best LiDAR Datasets to Learn and Process Point Clouds Data*. 2022. – URL https://www.thinkautonomous.ai/blog/lidar-datasets/. – Zugriffsdatum: 2024-02-26

[3] BARLA, Nilesh: *A Comprehensive Guide to Human Pose Estimation*. 2023. – URL https://www.v7labs.com/blog/human-pose-estimation-guide. – Zugriffsdatum: 2023-02-27

[4] BEDKOWSKI, Janusz: Understanding 3D shapes being in motion. In: *Journal of Automation, Mobile Robotics and Intelligent Systems* 1 (2013), 01, S. 42–46

[5] BERTONI, Lorenzo ; KREISS, Sven ; ALAHI, Alexandre: *Perceiving Humans: from Monocular 3D Localization to Social Distancing*. 2021

[6] BROWN, Ed: *Advanced LiDAR - On the Road to SAE Level 3 Partially Automated Driving*. 2022. – URL https://www.techbriefs.com/component/content/article/47086-advanced-lidar-on-the-road-to-sae-level-3-partially-automated-driving. – Zugriffsdatum: 2024-02-26

[7] CARLA: *Understanding Region of Interest - Part 2 (RoI Align)*. 2020. – URL https://carla.org/. – Zugriffsdatum: 2023-08-09

[8] CARLA: *CARLA - The simulator*. 2023. – URL https://carla.readthedocs.io/en/latest/start_introduction/. – Zugriffsdatum: 2023-08-09

[9] CHANG, Inho ; PARK, Min-Gyu ; KIM, Je W. ; YOON, Ju H.: Absolute 3D Human Pose Estimation Using Noise-Aware Radial Distance Predictions. In: *Symmetry* 15 (2023), Nr. 1. – URL https://www.mdpi.com/2073-8994/15/1/25. – ISSN 2073-8994

[10] CHANG, Ju Y. ; MOON, Gyeongsik ; LEE, Kyoung M.: *PoseLifter: Absolute 3D human pose lifting network from a single noisy 2D human pose*. 2020

[11] DESMARAIS, Yann ; MOTTET, Denis ; SLANGEN, Pierre ; MONTESINOS, Philippe: A review of 3D human pose estimation algorithms for markerless motion capture. In: *Computer Vision and Image Understanding* 212 (2021), S. 103275. – URL https://www.sciencedirect.com/science/article/pii/S1077314221001193. – ISSN 1077-3142

[12] DOSOVITSKIY, Alexey ; ROS, German ; CODEVILLA, Felipe ; LOPEZ, Antonio ; KOLTUN, Vladlen: CARLA: An Open Urban Driving Simulator. In: *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, S. 1–16

[13] EROL, Ali ; BEBIS, George ; NICOLESCU, Mircea ; BOYLE, Richard D. ; TWOMBLY, Xander: Vision-based hand pose estimation: A review. In: *Computer Vision and Image Understanding* 108 (2007), Oktober, Nr. 1-2, S. 52–73. – URL https://linkinghub.elsevier.com/retrieve/pii/S1077314206002281. – Zugriffsdatum: 2023-02-22. – ISSN 10773142

[14] FATEMI, Reza ; ABIRI, Behrooz ; KHACHATURIAN, Aroutin ; HAJIMIRI, Ali: High sensitivity active flat optics optical phased array receiver with a two-dimensional aperture. In: *Opt. Express* 26 (2018), Nov, Nr. 23, S. 29983–29999. – URL http://opg.optica.org/oe/abstract.cfm?URI=oe-26-23-29983

[15] GAMES, Epic: *Unreal Engine 4*. 2014. – URL https://www.unrealengine.com/. – Computer software

[16] GAMES, Epic: *Traces with Raycasts*. 2024. – URL https://docs.unrealengine.com/5.3/en-US/traces-with-raycasts-in-unreal-engine/. – Zugriffsdatum: 2023-02-27

[17] GIRSHICK, Ross ; DONAHUE, Jeff ; DARRELL, Trevor ; MALIK, Jitendra: *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014

[18] HANS-PETER, Willig: *Apertur*. 2022. – URL https://www.cosmos-indirekt.de//Physik-Schule/Apertur. – Zugriffsdatum: 03-09-2022

[19] HATA, Kenji ; SAVARESE, Silvio: *CS231A Course Notes 1: Camera Models*. 2021. – URL https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf. – Zugriffsdatum: 2023-03-03

[20] INDIREKT cosmos: *Remission (Physik)*. 2022. – URL https://physik.cosmos-indirekt.de/Physik-Schule/Remission_(Physik). – Zugriffsdatum: 28-07-2022

[21] IONESCU, Catalin ; PAPAVA, Dragos ; OLARU, Vlad ; SMINCHISESCU, Cristian: Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), Nr. 7, S. 1325–1339

[22] JAHANGIRI, Ehsan ; YUILLE, Alan L.: *Generating Multiple Diverse Hypotheses for Human 3D Pose Consistent with 2D Joint Detections*. 2017

[23] JENOPTIK AG: *Innovative LiDAR-Technologien: optische Module und Komponenten für LiDAR-Sensoren*. 2022. – URL https://www.jenoptik.de/produkte/lidar-sensoren-technologien. – Zugriffsdatum: 28-07-2022

[24] KATIRCIOGLU, I. ; TEKIN, B. ; SALZMANN, M. ; LEPETIT, Vincent ; FUA, Pascal: Learning Latent Representations of 3D Human Pose with Deep Neural Networks. In: *International Journal of Computer Vision* 126 (2018), S. 1326–1341. – ISSN 0920-5691

[25] KUHN, Harold W.: The Hungarian Method for the Assignment Problem. In: *Naval Research Logistics Quarterly* 2 (1955), March, Nr. 1–2, S. 83–97

[26] LA GORCE, M. de ; FLEET, D. J. ; PARAGIOS, N.: Model-Based 3D Hand Pose Estimation from Monocular Video. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), September, Nr. 9, S. 1793–1805. – URL http://ieeexplore.ieee.org/document/5719617/. – Zugriffsdatum: 2023-02-22. – ISSN 0162-8828, 2160-9292

[27] LOPER, Matthew ; MAHMOOD, Naureen ; ROMERO, Javier ; PONS-MOLL, Gerard ; BLACK, Michael: SMPL: a skinned multi-person linear model, 11 2015

[28] LOPER, Matthew ; MAHMOOD, Naureen ; ROMERO, Javier ; PONS-MOLL, Gerard ; BLACK, Michael J.: SMPL: A Skinned Multi-Person Linear Model. In: *ACM Transactions on Graphics (TOG)* 34 (2015), Nr. 6, S. 1–16

[29] MARTINEZ, Julieta ; HOSSAIN, Rayat ; ROMERO, Javier ; LITTLE, James J.: *A simple yet effective baseline for 3d human pose estimation*. 2017

[30] MATHONLINE: *Tangent Planes to Surfaces*. 2024. – URL http://mathonline.wikidot.com/tangent-planes-to-surfaces. – Zugriffsdatum: 2024-02-28

[31] MIURA, Teppei ; SAKO, Shinji: Simple yet effective 3D ego-pose lift-up based on vector and distance for a mounted omnidirectional camera. In: *Applied Intelligence* 53 (2022), 05, S. 1–13

[32] MUNKRES, James: Algorithms for the Assignment and Transportation Problems. In: *Journal of the Society for Industrial and Applied Mathematics* 5 (1957), Nr. 1, S. 32–38. – URL https://doi.org/10.1137/0105003

[33] NEWELL, Alejandro ; YANG, Kaiyu ; DENG, Jia: *Stacked Hourglass Networks for Human Pose Estimation*. 2016

[34] OIKONOMIDIS, Iason ; KYRIAZIS, Nikolaos ; ARGYROS, Antonis: Efficient model-based 3D tracking of hand articulations using Kinect. In: *Procedings of the British Machine Vision Conference 2011*. Dundee : British Machine Vision Association, 2011, S. 101.1–101.11. – URL http://www.bmva.org/bmvc/2011/proceedings/paper101/index.html. – Zugriffsdatum: 2023-02-22. – ISBN 978-1-901725-43-8

[35] OUSTER: *OS1, High Resolution Imaging LIDAR*. 2019. – URL https://www.dataspeedinc.com/app/uploads/2019/10/Ouster-OS1-Datasheet.pdf. – Zugriffsdatum: 2023-02-28

[36] OUSTER: *OS2, Long-Range High-Resolution Imaging Lidar*. 2021. – URL https://data.ouster.io/downloads/datasheets/datasheet-revd-v2p0-os2.pdf. – Zugriffsdatum: 2023-03-01

[37] OUSTER: *The Dead Bug Problem*. 2022. – URL https://ouster.com/blog/beam-aperture-and-the-dead-bug-problem/. – Zugriffsdatum: 03-09-2022

[38] PAVLAKOS, Georgios ; ZHOU, Xiaowei ; DERPANIS, Konstantinos G. ; DANIILIDIS, Kostas: *Coarse-to-Fine Volumetric Prediction for Single-Image 3D Human Pose*. 2017

[39] PAVLLO, Dario ; FEICHTENHOFER, Christoph ; GRANGIER, David ; AULI, Michael: *3D human pose estimation in video with temporal convolutions and semi-supervised training*. 2019

[40] PHAM, Huy H. ; SALMANE, Houssam ; KHOUDOUR, Louahdi ; CROUZIL, Alain ; ZEGERS, Pablo ; VELASTIN, Sergio A.: *A Unified Deep Framework for Joint 3D Pose Estimation and Action Recognition from a Single RGB Camera*. 2019

[41] PONS-MOLL, Gerard ; ROSENHAHN, Bodo: Model-Based Pose Estimation. (2011)

[42] QI, Charles R. ; SU, Hao ; MO, Kaichun ; GUIBAS, Leonidas J.: *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017

[43] QI, Charles R. ; YI, Li ; SU, Hao ; GUIBAS, Leonidas J.: *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017

[44] QIAN, Chen ; SUN, Xiao ; WEI, Yichen ; TANG, Xiaoou ; SUN, Jian: Realtime and Robust Hand Tracking from Depth. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA : IEEE, Juni 2014, S. 1106–1113. – URL https://ieeexplore.ieee.org/document/6909541. – Zugriffsdatum: 2023-02-22. – ISBN 978-1-4799-5118-5

[45] REICHERT, Hannes ; HETZEL, Manuel ; SCHRECK, Steven ; DOLL, Konrad ; SICK, Bernhard: Sensor Equivariance by LiDAR Projection Images. In: *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, S. 1–6

[46] SANCHEZ, Julia ; DENIS, Florence ; COEURJOLLY, David ; DUPONT, Florent ; TRASSOUDAINE, Laurent ; CHECCHIN, Paul: Robust normal vector estimation in 3D point clouds through iterative principal component analysis. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 163 (2020), 05, S. 18–35

[47] SCHRECK, Steven ; REICHERT, Hannes ; HETZEL, Manuel ; DOLL, Konrad ; SICK, Bernhard: *Height Change Feature Based Free Space Detection*. 2023

[48] SCHRÖDER, Matthias ; MAYCOCK, Jonathan ; RITTER, Helge ; BOTSCH, Mario: Real-time hand tracking using synergistic inverse kinematics. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on* IEEE (Veranst.), 2014, S. 5447–5454

[49] SICK: *Laserscanner Planungshilfen für die Perimeter- und Objektüberwachung*. 2022. – URL https://www.siedleralarm.ch/fileadmin/user_upload/www.siedleralarm.ch/PDF/Sick_Laserscanner_Alarm_Planungshilfe.pdf. – Zugriffsdatum: 28-07-2022

[50] SICK AG: *Remission*. 2022. – URL https://www.sick.com/de/de/glossar/remission/g/p544247. – Zugriffsdatum: 29-07-2022

[51] SIMULATOR, CARLA: *Sensors reference; Semantic LIDAR sensor*. 2024. – URL https://carla.readthedocs.io/en/latest/ref_sensors/#semantic-lidar-sensor. – Zugriffsdatum: 2023-02-27

[52] SUN, Pei ; KRETZSCHMAR, Henrik ; DOTIWALLA, Xerxes ; CHOUARD, Aurelien ; PATNAIK, Vijaysai ; TSUI, Paul ; GUO, James ; ZHOU, Yin ; CHAI, Yuning ; CAINE, Benjamin ; VASUDEVAN, Vijay ; HAN, Wei ; NGIAM, Jiquan ; ZHAO, Hang ; TIMOFEEV, Aleksei ; ETTINGER, Scott ; KRIVOKON, Maxim ; GAO, Amy ; JOSHI, Aditya ; ZHANG, Yu ; SHLENS, Jonathon ; CHEN, Zhifeng ; ANGUELOV, Dragomir: Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020

[53] TAN, Fang ; XIA, Zhaoqiang ; MA, Yupeng ; FENG, Xiaoyi: 3D Sensor Based Pedestrian Detection by Integrating Improved HHA Encoding and Two-Branch Feature Fusion. In: *Remote Sensing* 14 (2022), Nr. 3. – URL https://www.mdpi.com/2072-4292/14/3/645. – ISSN 2072-4292

[54] TEKIN, Bugra ; KATIRCIOGLU, Isinsu ; SALZMANN, Mathieu ; LEPETIT, Vincent ; FUA, Pascal: *Structured Prediction of 3D Human Pose with Deep Neural Networks*. 2016

[55] TKBW: *Flash LiDAR*. 2023. – URL https://www.transformationswissen-bw.de/fileadmin/media/Publikationen/2020/Technologiekalender/Technologiesteckbrief_Flash_LiDAR__T108__TKBW_Update_2023.pdf. – Zugriffsdatum: 2024-02-02

[56] TOME, Denis ; RUSSELL, Chris ; AGAPITO, Lourdes: Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Juli 2017. – URL http://dx.doi.org/10.1109/CVPR.2017.603

[57] TOPIWALA, Anirudh: *Spherical Projection for Point Clouds*. 2020. – URL https://towardsdatascience.com/spherical-projection-for-point-clouds-56a2fc258e6c. – Zugriffsdatum: 2023-02-28

[58] TURACAN, Kerim: *Methoden zur Segmentierung von LiDAR Punktwolken*. 2022

[59] VOLLMER, Andreas ; VOLLMER, Michael ; LANG, Gernot ; STRAUB, Anton ; KÜBLER, Alexander ; GUBIK, Sebastian ; BRANDS, Roman C. ; HARTMANN, Stefan ; SARAVI, Babak: Automated Assessment of Radiographic Bone Loss in the Posterior Maxilla Utilizing a Multi-Object Detection Artificial Intelligence Algorithm. In: *Applied Sciences* 13 (2023), Nr. 3. – URL https://www.mdpi.com/2076-3417/13/3/1858. – ISSN 2076-3417

[60] WANG, Dingkang ; WATKINS, Connor ; XIE, Huikai: MEMS Mirrors for LiDAR: A review. In: *Micromachines* 11 (2020), 04, S. 456

[61] WANG, Jinbao ; TAN, Shujie ; ZHEN, Xiantong ; XU, Shuo ; ZHENG, Feng ; HE, Zhenyu ; SHAO, Ling: Deep 3D human pose estimation: A review. In: *Computer Vision and Image Understanding* 210 (2021), 05, S. 103225

[62] WANG, Yue ; SUN, Yongbin ; LIU, Ziwei ; SARMA, Sanjay E. ; BRONSTEIN, Michael M. ; SOLOMON, Justin M.: *Dynamic Graph CNN for Learning on Point Clouds*. 2019

[63] WEBER, Harald: *FUNKTIONSWEISE UND VARIANTEN VON LiDAR-SENSOREN*. 2018. – URL https://cdn.sick.com/media/docs/5/25/425/whitepaper_lidar_de_im0079425.pdf. – Zugriffsdatum: 28-07-2022

[64] WENG, Zhenzhen ; GORBAN, Alexander S. ; JI, Jingwei ; NAJIBI, Mahyar ; ZHOU, Yin ; ANGUELOV, Dragomir: *3D Human Keypoints Estimation From Point Clouds in the Wild Without Human Labels*. 2023

[65] XU, Tianhan ; TAKANO, Wataru: *Graph Stacked Hourglass Networks for 3D Human Pose Estimation*. 2021

[66] ZHENG, Jingxiao ; SHI, Xinwei ; GORBAN, Alexander ; MAO, Junhua ; SONG, Yang ; QI, Charles R. ; LIU, Ting ; CHARI, Visesh ; CORNMAN, Andre ; ZHOU, Yin ; LI, Congcong ; ANGUELOV, Dragomir: *Multi-modal 3D Human Pose Estimation with 2D Weak Supervision in Autonomous Driving*. 2021

[67] ZHOU, Yufan ; DONG, Haiwei ; EL SADDIK, Abdulmotaleb: Learning to Estimate 3D Human Pose From Point Cloud. In: *IEEE Sensors Journal* 20 (2020), Oktober, Nr. 20, S. 12334–12342. – URL http://dx.doi.org/10.1109/JSEN.2020.2999849. – ISSN 2379-9153

[68] ZHOU, Yufan ; DONG, Haiwei ; EL SADDIK, Abdulmotaleb: Learning to Estimate 3D Human Pose From Point Cloud. In: *IEEE Sensors Journal* 20 (2020), Oktober, Nr. 20, S. 12334–12342. – URL http://dx.doi.org/10.1109/JSEN.2020.2999849. – ISSN 2379-9153